

# Cincom

## **gOOi**

User's Guide

P39-5020-00



---

## gOOi™ User's Guide

© 1997, 1999, 2000, 2001 Cincom Systems, Inc.  
All rights reserved

This document contains unpublished, confidential, and proprietary information of Cincom. No disclosure or use of any portion of the contents of these materials may be made without the express written consent of Cincom.

The following are trademarks, registered trademarks, or service marks of Cincom Systems, Inc.:

AD/Advantage®  
C+A-RE™  
CINCOM®  
Cincom Encompass®  
Cincom Smalltalk™  
Cincom SupportWeb®  
CINCOM SYSTEMS®  
  
gOOi™

iD CinDoc™  
iD CinDoc Web™  
iD Consulting™  
iD Correspondence™  
iD Correspondence Express™  
iD Environment™  
iD Solutions™  
intelligent Document Solutions™

MANTIS®  
Mindspeed™  
MindspeedXML™  
SPECTRA™  
SUPRA®  
SUPRA® Server  
Visual Smalltalk®  
VisualWorks®

All other trademarks are trademarks or registered trademarks of:

Acucobol, Inc.  
AT&T  
Compaq Computer Corporation  
Data General Corporation  
Gupta Technologies, Inc.  
International Business Machines Corporation  
JSB Computer Systems Ltd.

Micro Focus, Inc.  
Microsoft Corporation  
Systems Center, Inc.  
TechGnosis International, Inc.  
The Open Group  
UNIX System Laboratories, Inc.

or of their respective companies.

Cincom Systems, Inc.  
55 Merchant Street  
Cincinnati, Ohio 45246-3732  
U.S.A.

PHONE: (513) 612-2300  
FAX: (513) 612-2000  
WORLD WIDE WEB: <http://www.cincom.com>

---

### Attention:

Some Cincom products, programs, or services referred to in this publication may not be available in all countries in which Cincom does business. Additionally, some Cincom products, programs, or services may not be available for all operating systems or all product releases. Contact your Cincom representative to be certain the items are available to you.

---

## Release information for this manual

The *gOOi User's Guide*, P39-5020-00, is dated January 31, 2001. This document supports Release 4.0.01 of gOOi in the IBM MVS and VSE, OpenVMS/Alpha, OpenVMS/VAX, Windows NT and Windows 95/98/2000 environments.

### We welcome your comments

We encourage critiques concerning the technical content and organization of this manual. Please take the [survey](#) provided with the online documentation at your convenience.

FAX: (513) 612-2000  
Attn: gOOi Support

E-mail: [helpna@cincom.com](mailto:helpna@cincom.com)

Phone: 1-800-727-3525

Mail: Cincom Systems, Inc.  
Attn: gOOi Support  
55 Merchant Street  
Cincinnati, OH 45246-3732  
U.S.A.



# Contents

<b>About this book</b>	<b>xi</b>
Using this document.....	xi
Document organization.....	xii
Conventions.....	xiv
Related documentation.....	xvii
Educational material .....	xvii
<b>Introduction</b>	<b>19</b>
Overview of gOOi .....	19
Before you begin.....	20
gOOi components.....	21
How gOOi works.....	23
Pre-Generation .....	24
Generation .....	24
Post-Generation .....	25
Runtime .....	25
<b>Just-In-Time GUI display</b>	<b>27</b>
Introduction .....	27
Features of Just-In-Time GUI display.....	28
Common gOOi functions (Pop-Up Menu) .....	31
Additional keyboard considerations for JIT.....	33
Using the title bar close options with JIT .....	34
JIT pop-ups.....	34
Character Display for 3270 data streams .....	37
Character Display for VT data streams .....	39

<b>Preparing to generate gOOi forms</b>	<b>41</b>
Loading gOOi .....	42
Specifying gOOi generation options and settings .....	45
Setting the screen ID .....	46
Application bundling .....	48
Using MANTIS Dynamic CONVERSE .....	49
Keeping forms open for reuse .....	49
Selecting the Host Connection .....	51
Specifying the Host Connection Profile .....	52
Specifying a TCP/IP Connection .....	53
Specifying an Emulator Connection .....	55
Specifying host field attribute matching .....	59
Form and Color options .....	62
Using hotspots .....	72
Creating a user toolbar .....	76
Spacing toolbar items .....	79
Specifying headers and footers .....	79
Pointing to prompter files .....	80
Using environment-dependent tools .....	82
Converting BMS and MFS source files into UEF format .....	82
gOOi restrictions for non-MANTIS host applications .....	82
BMS/MFS conversion procedure .....	83
Capturing IBM mainframe host screens .....	86
Locating screen IDs .....	89
Creating a template for function key mapping .....	91
Modifying a function key map file .....	100
Copying a function key map file .....	101
Editing your loadable application .....	102
Creating an application template for visual items .....	103
Application template inheritance .....	104
Steps for creating an application template for visual items .....	105
<b>Downloading screens from MANTIS</b>	<b>111</b>
Methods for transferring UEF screen images .....	111
Using the Host Monitor .....	112
Using FTP .....	117
Creating .exp files with the File Splitter .....	118
gOOi restrictions for MANTIS applications .....	118
UEF file splitting steps .....	119

---

<b>Generating gOOi forms</b>	<b>125</b>
Overview of gOOi forms generation .....	125
Specifying gOOi form components.....	127
gOOi form generation restrictions .....	127
MANTIS users .....	129
Form component specification steps.....	130
Generating gOOi application forms .....	141
Viewing a gOOi form .....	146
Viewing an ungenerated form using Preview .....	146
Viewing a generated gOOi form without a host connection.....	147
Viewing a generated gOOi form with a host connection.....	148
<b>Using integration wizards</b>	<b>149</b>
Word Wizard tool.....	149
Excel Wizard tool.....	153
<b>Customizing gOOi forms</b>	<b>159</b>
Overview of customization.....	159
Bringing a gOOi form into ObjectStudio Designer .....	161
Basic customizations .....	164
Moving fields.....	164
Changing the size and shape of fields.....	167
Grouping fields graphically .....	180
Changing the form title and/or background .....	185
Additional customizations .....	190
Using check boxes for Y/N fields.....	190
Using spin buttons for numeric fields .....	191
Putting data in the status line .....	192
Using property pages .....	197
Example of an event-driven customization.....	203
Creating an event-driven button with an existing Smalltalk method.....	203
Creating an event-driven button with Smalltalk .....	206

<b>Host Navigation</b>	<b>213</b>
Overview of Host Navigation .....	213
Host Navigation methods and attributes .....	214
Object: Controllers generated by the Application Generator.....	220
Methods .....	221
Attributes .....	225
Object: Host Objects generated by the Application Generator .....	226
Methods .....	226
Attributes .....	227
Object: GOOIHostMonitorController .....	227
Methods .....	227
Attributes .....	229
Object: gOOi Session .....	230
Methods .....	230
Attributes .....	236
Object: GOOIJITController (Just-In-Time Controller) .....	237
Methods .....	237
Attributes .....	242
Object: GOOIJITCustomController (Just-In-Time Custom Controller) .....	243
Methods .....	244
Attributes .....	247
Object: GOOIVTEmulatorCustomController .....	248
Methods .....	248
Attributes .....	257
<b>Deploying a gOOi application</b>	<b>259</b>
gOOi application deployment steps .....	260
Using the standard Program Generator.....	261
Using the Small Program Generator.....	267
<b>PC CONTACT file access</b>	<b>269</b>
Overview of PC CONTACT file access.....	269
Options for PC CONTACT .....	270
Save/Open display options .....	271
Numbered files floating point field format .....	272
Designing a MANTIS view for PC CONTACT.....	273
Create or update file views.....	274
Update file view layout .....	277
Sample PC file view design.....	280
Sample program .....	285

---

PC CONTACT supported file types .....	286
Sequential BASIC files .....	286
Sequential TEXT files .....	286
Sequential DIF files .....	287
Numbered files .....	292
<b>IBM mainframe considerations</b> .....	<b>293</b>
Platform considerations .....	293
<b>Emulator considerations</b> .....	<b>297</b>
How gOOi recognizes emulators .....	297
Emulators .....	297
PC3270 .....	297
EXTRA! .....	298
RUMBA .....	298
KEA! .....	298
Generic EHLLAPI .....	299
Reflection .....	300
Reflection for UNIX and Digital .....	300
<b>Screen Registry and AD/Advantage</b> .....	<b>301</b>
Screen registry and AD/Advantage .....	301
<b>gOOi class files and names</b> .....	<b>303</b>
gOOi classes .....	303
Emulator Communication class files .....	303
Telnet class files .....	303
TN3270 class files .....	303
TNVT class files .....	304
Generator class files .....	304
Run-time class files .....	304
gOOi user interfaces .....	304
Supporting classes .....	305
Utility class .....	305
Just-In-Time classes .....	305
Messages .....	305
BMS converter .....	305
MFS converter .....	306
UEF Generator .....	306
Default templates .....	306

<b>Rules for screen IDs and MANTIS prompter IDs</b>	<b>307</b>
Screen ID rules .....	307
Screen ID options.....	308
Prompter ID rules.....	310
<b>Host-PC translation tables</b>	<b>311</b>
Translation tables.....	311
<b>gOOi error messages</b>	<b>315</b>
GOOIMessages class .....	315
<b>Using the UEF Generator</b>	<b>327</b>
Using the UEF Generator.....	327
<b>Glossary of terms</b>	<b>329</b>
<b>Index</b>	<b>333</b>

---

# About this book

---

---

## Using this document

The purpose of this guide is to assist developers who want to generate Windows<sup>®</sup>-compatible interfaces for host applications (for example, MANTIS<sup>®</sup>, IBM<sup>®</sup> CICS, and AD/Advantage).

gOOi<sup>™</sup> forms are generated through ObjectStudio<sup>®</sup>, an object-oriented development tool offered by Cincom.

Before generating gOOi forms, you may need to be familiar with MANTIS and its Universal Export Facility (UEF), IBM Message Formatting Service (MFS), or IBM CICS Basic Mapping Support (BMS). For information about these products, refer to *AD/Advantage MANTIS Administration OpenVMS/UNIX*, P39-1320, or *AD/Advantage MANTIS Facilities OS/390, VSE/ESA*, P39-5001.

ObjectStudio Designer lets you modify and extend gOOi forms (see “*Overview of customization*” on page 159 for specific information). For more information about ObjectStudio Designer, refer to the *ObjectStudio User Interface Guide*, P40-3205.

You can also use gOOi for Java<sup>™</sup> to Web-enable MANTIS<sup>®</sup> and AD/Advantage host applications. For more information about gOOi for Java, refer to the *gOOi for Java User's Guide*, P39-5021.

## Document organization

The information in this manual is organized as follows:

### **Chapter 1—Introduction**

Provides a high-level discussion of gOOi capabilities.

### **Chapter 2—Just-In-Time GUI display**

Provides directions for using the automatic GUI display feature.

### **Chapter 3—Preparing to generate gOOi forms**

Discusses preparation for gOOi forms generation and provides step-by-step procedures for tools that may be needed before generating gOOi forms.

### **Chapter 4—Downloading screens from MANTIS**

Discusses ways to get Universal Export Facility (UEF) descriptions of MANTIS screens from the host to the PC.

### **Chapter 5—Generating gOOi forms**

Provides step-by-step procedures for generating gOOi forms (interfaces) for host application screens.

### **Chapter 6—Using integration wizards**

Provides step-by-step procedures for using wizard tools to integrate a host application with a Microsoft product (Word or Excel) on the PC.

### **Chapter 7—Customizing gOOi forms**

Provides step-by-step procedures for extending gOOi forms using ObjectStudio's Designer tool. The chapter also shows how visual items can be assigned to the gOOi generation process as a parent class, causing all generated forms to inherit the same visual items.

### **Chapter 8—Host Navigation**

Describes how to programmatically take control from gOOi (for example, the Host Monitor) and move through host screens yourself.

### **Chapter 9—Deploying a gOOi application**

Provides step-by-step procedures for deploying a gOOi application to an end-user workstation.

### **Chapter 10—PC CONTACT file access**

Describes how to use PC CONTACT to upload/download data between MANTIS on the mainframe and a personal computer.

**Appendix A—IBM mainframe considerations**

Provides information specific to users of IBM mainframes.

**Appendix B—Emulator considerations**

Provides the settings necessary for gOOi to recognize a specific emulator.

**Appendix C—Screen Registry and AD/Advantage**

Provides information specific to AD/Advantage users.

**Appendix D—gOOi class files and names**

Provides a list of class files and class names used by gOOi.

**Appendix E—Rules for screen IDs and MANTIS prompter IDs**

Provides rules for the unique identifiers required by gOOi for all screens and MANTIS prompters.

**Appendix F—Host-PC translation tables**

Provides the key mappings used in the default IBM and ASCII controllers provided with this product.

**Appendix G—gOOi error messages**

Provides descriptions and user actions for error messages generated by gOOi.

**Appendix H—Using the UEF Generator**

Provides step-by-step procedures for creating a UEF (Universal Export Facility) file from gOOi forms, then uploading the file to the host and importing it into MANTIS.

**Glossary of terms**

**Index**

## Conventions

The following table describes the conventions used in this document series:

Convention	Description	Example
Constant width type	Represents screen images and segments of code.	<pre>PUT 'customer.dat' GET 'miller\customer.dat' PUT '\DEV\RMT0'</pre>
Slashed b ( <i>b</i> )	Indicates a space (blank).  The example indicates that four spaces appear between the keywords.	<pre>BEGIN<b>bbb</b>SERIAL</pre>
Brackets [ ]	Indicate optional selection of parameters. (Do not attempt to enter brackets or to stack parameters.) Brackets indicate one of the following situations:	
	A single item enclosed by brackets indicates that the item is optional and can be omitted.  The example indicates that you can optionally enter a WHERE clause.	<code>[WHERE <i>search-condition</i>]</code>
	Stacked items enclosed by brackets represent optional alternatives, one of which can be selected.  The example indicates that you can optionally enter either WAIT or NOWAIT. (WAIT is underlined to signify that it is the default.)	<pre>[<u>WAIT</u> NOWAIT]</pre>

Convention	Description	Example
Braces { }	<p>Indicate selection of parameters. (Do not attempt to enter braces or to stack parameters.) Braces surrounding stacked items represent alternatives, one of which you must select.</p> <p>The example indicates that you must enter ON or OFF when using the MONITOR statement.</p>	<pre>MONITOR {ON          OFF}</pre>
<u>Underlining</u> (In syntax)	<p>Indicates the default value supplied when you omit a parameter.</p> <p>The example indicates that if you do not choose a parameter, the system defaults to WAIT.</p>	<pre>(WAIT) (NOWAIT)</pre>
	<p>Underlining also indicates an allowable abbreviation or the shortest truncation allowed.</p> <p>The example indicates that you can enter either STAT or STATISTICS.</p>	<pre>STATISTICS</pre>
Ellipsis points...	<p>Indicate that the preceding item can be repeated.</p> <p>The example indicates that you can enter multiple host variables and associated indicator variables.</p>	<pre>INTO :host-variable [:ind- variable],...</pre>
UPPERCASE Lowercase	<p>In most operating environments, keywords are not case-sensitive and they are represented in uppercase. You can enter them in either uppercase or lowercase.</p> <p>In the UNIX operating environment, keywords are case-sensitive and you must enter them exactly as shown.</p>	<pre>COPY MY_DATA.SEQ HOLD_DATA.SEQ  cp *.QAR /backup</pre>

Convention	Description	Example
<i>Italics</i>	<p>Indicate variables you replace with a value, a column name, a file name, and so on.</p> <p>The example indicates that you must substitute the name of a table.</p>	FROM <i>table-name</i>
Punctuation marks	<p>Indicate required syntax that you must code exactly as presented.</p> <p>( ) parentheses            . period            , comma            : colon            ' ' single quotation marks</p>	<p>(<i>user-id</i>, <i>password</i>, <i>db-name</i>)</p> <p>INFILE 'Cust.Memo' CONTROL            LEN4</p>
⇒ (Right arrow)	Indicates that you should select each choice that is separated by an arrow, in sequence.	Select File ⇒ Save as
+ (Plus sign)	Indicates that you should hold down the first key and press the second key.	CTRL+S

### Mouse button conventions

*Mouse button 1* refers to the primary mouse button—on a right-handed mouse, the button on the left; on a left-handed mouse, the button on the right.

*Mouse button 2* refers to the secondary mouse button—on a right-handed mouse, the button on the right; on a left-handed mouse, the button on the left.

## Related documentation

Below are listed the manuals referenced in this document that will be helpful with your understanding of gOOi.

### General use

- ◆ *AD/Advantage MANTIS Language OpenVMS/UNIX*, P39-1310
- ◆ *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300
- ◆ *gOOi for Java User's Guide*, P39-5021
- ◆ *ObjectStudio User Interface Guide*, P40-3205
- ◆ *ObjectStudio Smalltalk User's Guide*, P40-3202
- ◆ *ObjectStudio User's Guide*, P40-3201

### Educational material

MANTIS educational material is available from your regional Cincom education department.



# 1

---

## Introduction

---

---

### Overview of gOOi

gOOi (pronounced gooey ('gü-ē)) is a Graphical User Interface (GUI) design tool that allows you to extend almost any host application, such as MANTIS, IBM CICS, and AD/Advantage, into the Microsoft® Windows GUI environments.

gOOi can quickly generate the appropriate Windows code corresponding to your host application without making changes to your host system. We recommend, but do not require, that your host applications have unique screen identifiers in a standard location. If your application uses MANTIS prompts, gOOi requires a unique identifier (enclosed within parentheses) at the end of each prompt description.

## Before you begin

Before installing gOOi, make sure the latest version of ObjectStudio, including ObjectStudio Designer and Modeling Tool, is installed on your PC. Please visit the Cincom Web site ([www.cincom.com](http://www.cincom.com)), or contact your Cincom representative for ObjectStudio version information.

If you have a TCP/IP network, gOOi does not require any additional software. If you do not have a TCP/IP network, gOOi includes interfaces for many of the popular host emulators, such as:

- ◆ Attachmate Corporation's Extra!®
- ◆ Attachmate Corporation's KEA!®
- ◆ IBM® Corporation's Personal Communications
- ◆ Wall Data Corporation's Rumba®
- ◆ WRQ, Inc.'s Reflection®
- ◆ WRQ, Inc.'s Reflection2

gOOi also supports IBM mainframe emulators other than those in the preceding list via two generic EHLLAPI interfaces. If your emulator is not listed here, contact Cincom to see if your emulator is supported.

After installing ObjectStudio, you can install gOOi according to the on-screen setup instructions provided on the installation media. When you have verified that the required software has been successfully installed, you are ready to create a gOOi application (see “[Generating gOOi application forms](#)” on page 141)



---

For best results with gOOi, we recommend that you set your display settings for the desktop area at 800 by 600 pixels, or a higher resolution of 1024 by 768 pixels. We do not recommend a resolution of 640 by 480 pixels.

---

## gOOi components

When you purchase gOOi, you receive several components that work together to generate custom-designed forms for developing a loadable application for ObjectStudio. The contents of a complete gOOi package include the following:

Component	Description
Application Generator	Generates a gOOi application containing Windows interfaces for all the host screens that you specify
Host Monitor	Monitors a host session and automatically matches each host screen to the proper gOOi form, or presents a Just-In-Time GUI display when no form is available
Template Hierarchy Browser	Creates template subclasses for developing function key maps
Settings	Defines host screen configuration settings and gOOi form generation options
Screen ID Locator	Determines a suitable location to add a screen ID
File Splitter	Selects all screen (and prompter, if applicable) elements in a UEF file and places them in separate .exp files
Dynamic Screen Capture	Generates a UEF file for an IBM mainframe host screen that can be used as input to the Application Generator

Component	Description
BMS Converter	Converts a BMS file into a UEF file
MFS Converter	Converts an MFS file into a UEF file
Function Keys Definition	Creates menu items that invoke function key commands
Word Wizard	Integrates host data into a Word for Windows document
Excel Wizard	Integrates host data into an Excel spreadsheet
Just-In-Time Display (JIT)	Displays a Windows presentation for a host screen without any form generation (requires a TCP/IP network and an IBM mainframe host)
Screen Registry tool	Associates one or more screen IDs with a gOOi form that has a name that does not match the screen ID



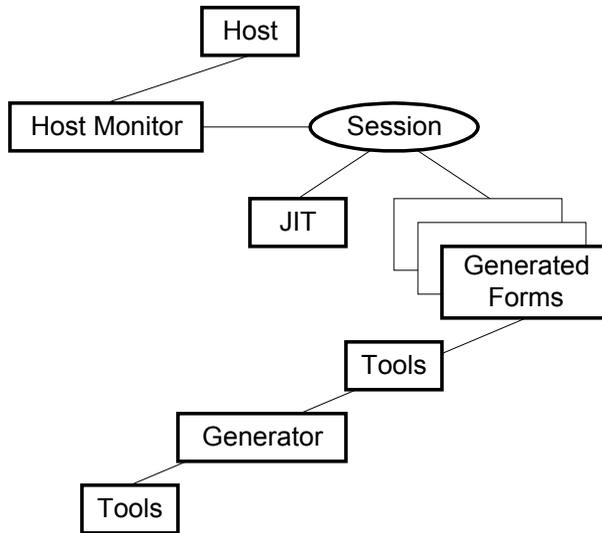
---

The Universal Export Facility (UEF) includes a specification for describing host screens.

---

## How gOOi works

The following illustration depicts a high-level view of gOOi:



## Pre-Generation

You can control the appearance of generated forms through the generation options provided with the Application Generator (see “[Specifying gOOi generation options and settings](#)” on page 45 for specific information).

Configuration options provide inherited (parent class) properties for gOOi form generation. gOOi templates provide the ability to automatically add GUI controls (for example, buttons) to your gOOi forms to enhance the user interface.

Because host applications are often function key driven, gOOi includes a tool that lets you define pull-down menus for invoking commands associated with specific terminal function keys.

gOOi uses MANTIS Universal Export Facility (UEF) format input files to generate forms. The BMS/MFS Converter tools take an input file that describes host BMS/MFS screens and outputs a file with equivalent descriptions of these screens in UEF format. This output file can then be input into the File Splitter tool to create one file per screen for input to the Application Generator.

gOOi can also dynamically capture any 3270-style host screen for conversion into a gOOi form. The Dynamic Screen Capture tool creates a UEF format file from the currently displayed IBM mainframe host screen.

## Generation

Host screen definitions in UEF format are input to the gOOi Application Generator, which outputs gOOi forms.

Each form consists of two ObjectStudio classes: a controller class with the visual display, and a host object class with host screen information. The controller class part of the form can be customized for enhanced display and/or processing.

The host object contains information specific to the host screen, such as field location and size, and can be regenerated independently of the controller if the host screen changes. Thus, you do not lose your visual display changes when the underlying host screen changes.

## Post-Generation

After generating a gOOi application, you can use ObjectStudio's Designer tool to customize the forms and improve the usability of your application.

ObjectStudio Designer allows you to move fields and enhance forms with resizable drop-down lists, slider bars, push buttons, and other GUI controls. For more information about ObjectStudio Designer, refer to the *ObjectStudio User Interface Guide*, P40-3205. gOOi provides an easy way for developers to employ object-oriented programming. gOOi generates standard ObjectStudio Smalltalk code or Java code that you can add to as needed. Custom code can also be attached to other Windows events.

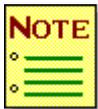
You can then use the Host Navigation feature of gOOi to combine multiple host screens into a single gOOi form. You can also utilize this feature to automate processing of host screens that do not require user interaction, such as file uploads or downloads.

The Excel and Word wizards are examples of post-generation tools. These wizards provide an easy way to integrate your generated form controllers with these popular desktop tools.

Cincom can provide the necessary training or professional services for implementing the object-oriented capabilities of gOOi and ObjectStudio. For more information, contact your Cincom representative.

## Runtime

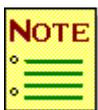
When the Host Monitor is started, it establishes a session with the host and starts monitoring host screen activity. gOOi attempts to find a matching form for each host display. If gOOi finds one, it displays that form. If no matching form is available and the Just-In-Time GUI display (JIT) feature is available, gOOi presents a JIT window.



---

You can override the display of both generated forms and JIT. See "[Host Navigation](#)" on page 213 for details.

---



---

When using the direct gOOi TCP/IP connection with MANTIS for OpenVMS/UNIX, start MANTIS with the 'gooi' parameter.

---



# 2

## Just-In-Time GUI display

### Introduction

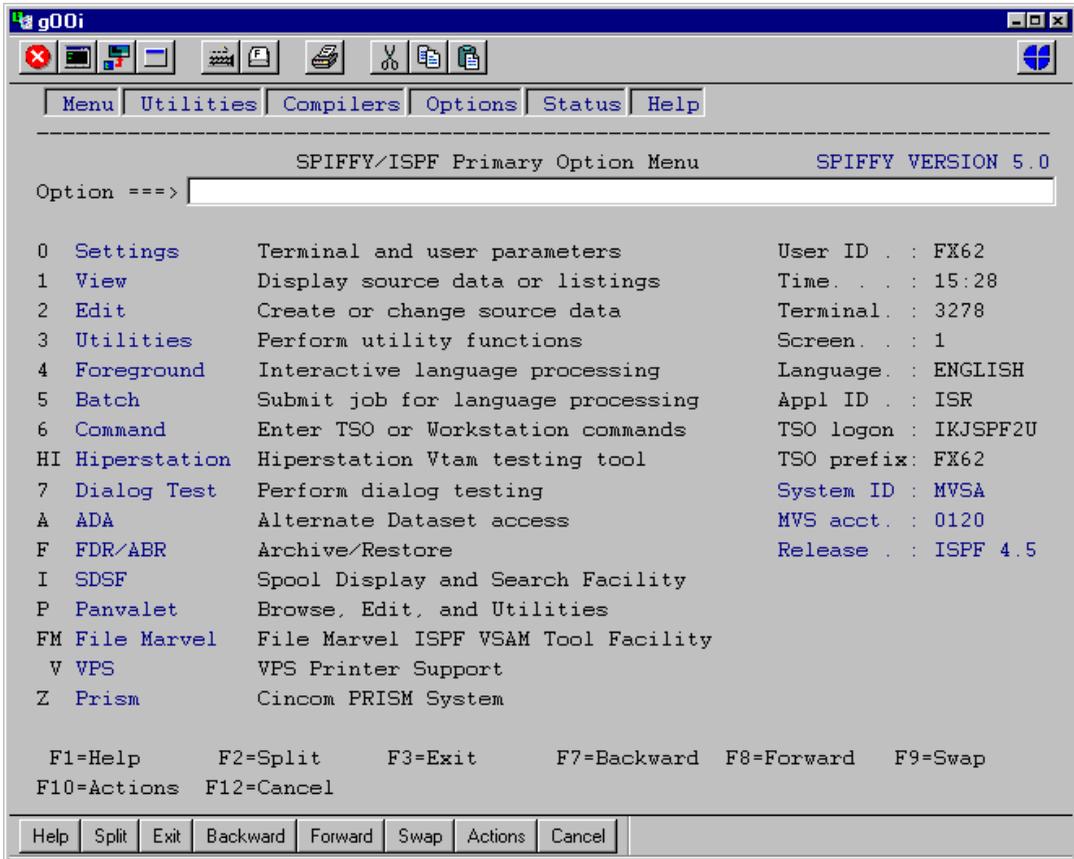


Just-In-Time is only available with hosts that support a 3270 data stream. If you are working with a OpenVMS/UNIX host, please see "[Character Display for VT data streams](#)" on page 39.

gOOi's Just-In-Time (JIT) GUI display feature offers a graphical interface for your host application without any application generation. This feature allows you to be productive with gOOi immediately, and with little or no effort. You can then gradually evolve your host applications into a richer graphical environment using the other features of gOOi and ObjectStudio. You can choose to enhance your most used, or most complex, applications and leave the others alone.

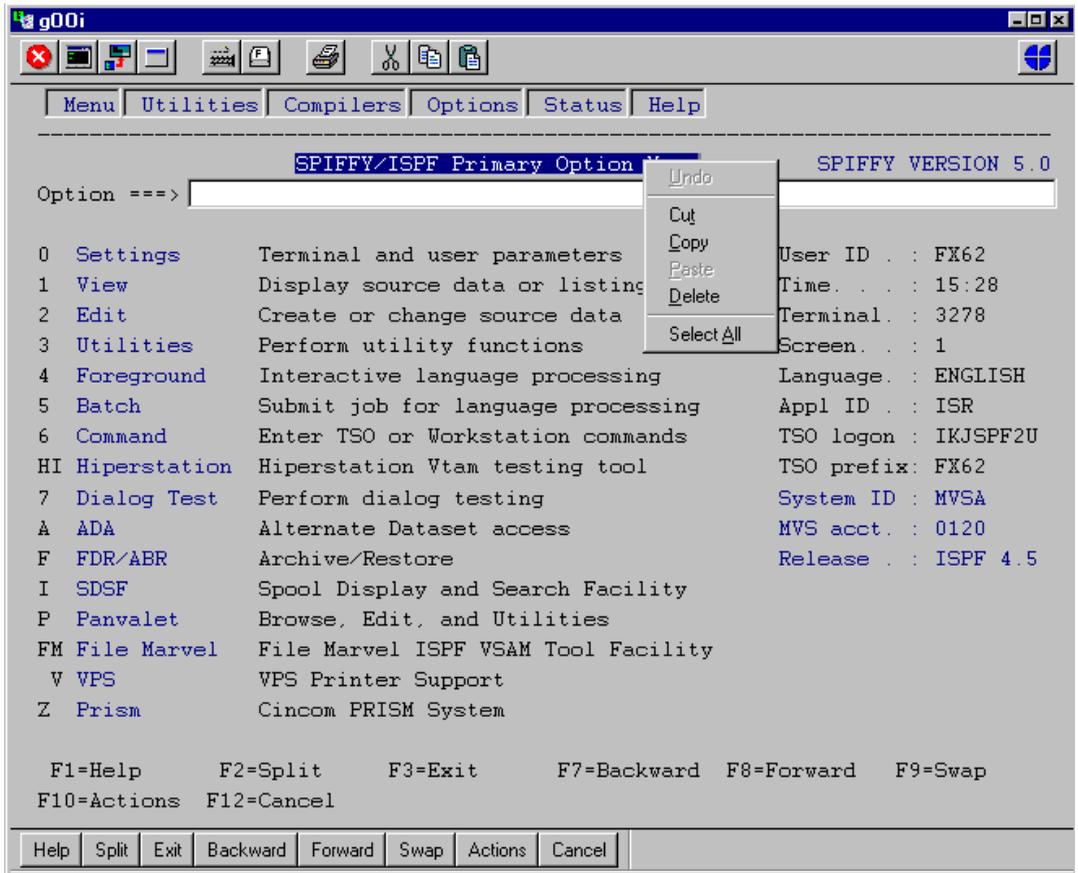
## Features of Just-In-Time GUI display

The following window shows a sample Just-In-Time (JIT) display:



The preceding window displays some of the features of JIT:

- ◆ The main body of the window consists of static text and entry fields (only one , Option, in this case).
- ◆ The field color for both static and entry fields can be modified via Form/Field Colors of Settings.
- ◆ Text from any individual field on a single line can be copied to the clipboard by highlighting the desired text and pressing Ctrl+C, or via a pop-up that is displayed by clicking mouse button 2 (for example, right mouse button), as shown in the following window:



- ◆ To copy text from multiple fields or lines to the clipboard, drag the pointer from left to right across the window while holding down mouse button 1 (for example, left mouse button). A rectangle forms a lasso around the fields/lines as you move the mouse. When you release mouse button 1, the text inside the rectangular lasso is copied to the clipboard.

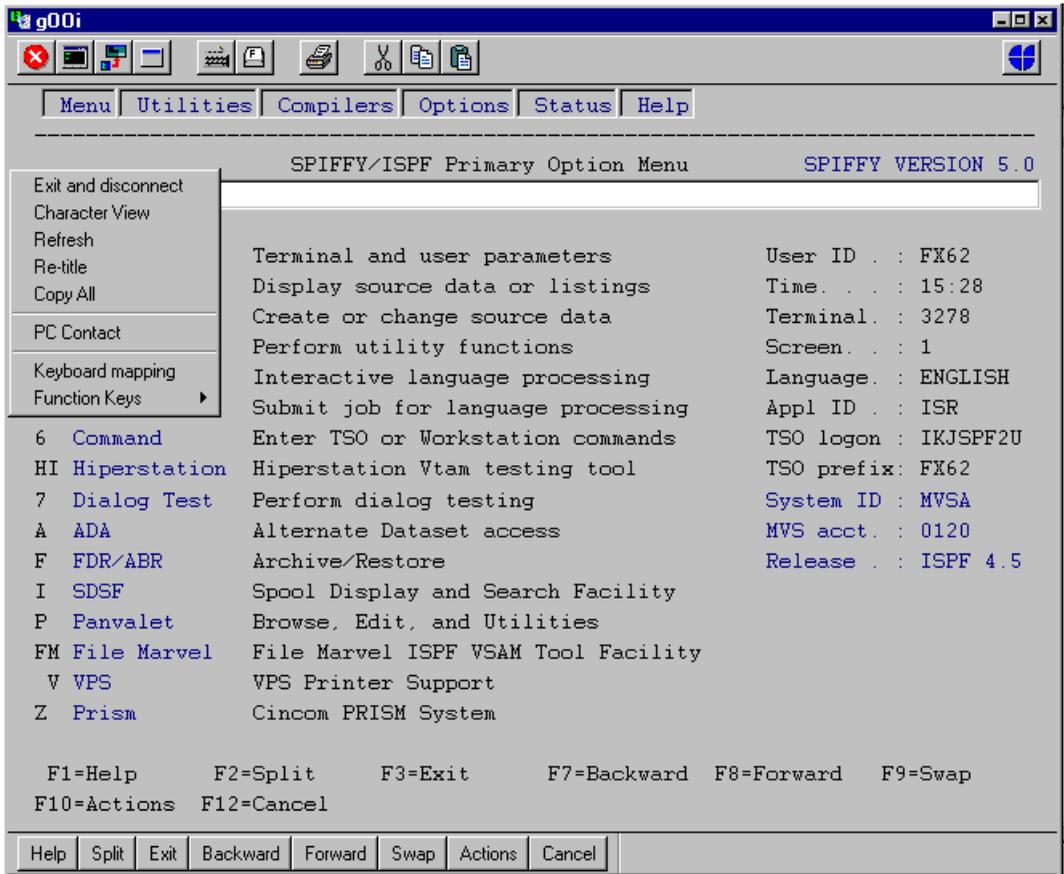
The preceding window also shows two toolbars that can be included as part of the JIT window. The toolbar at the top of the window is a sample user toolbar that is distributed with gOOi (jitnet.tb).

From left to right, the toolbar icons represent the following actions:

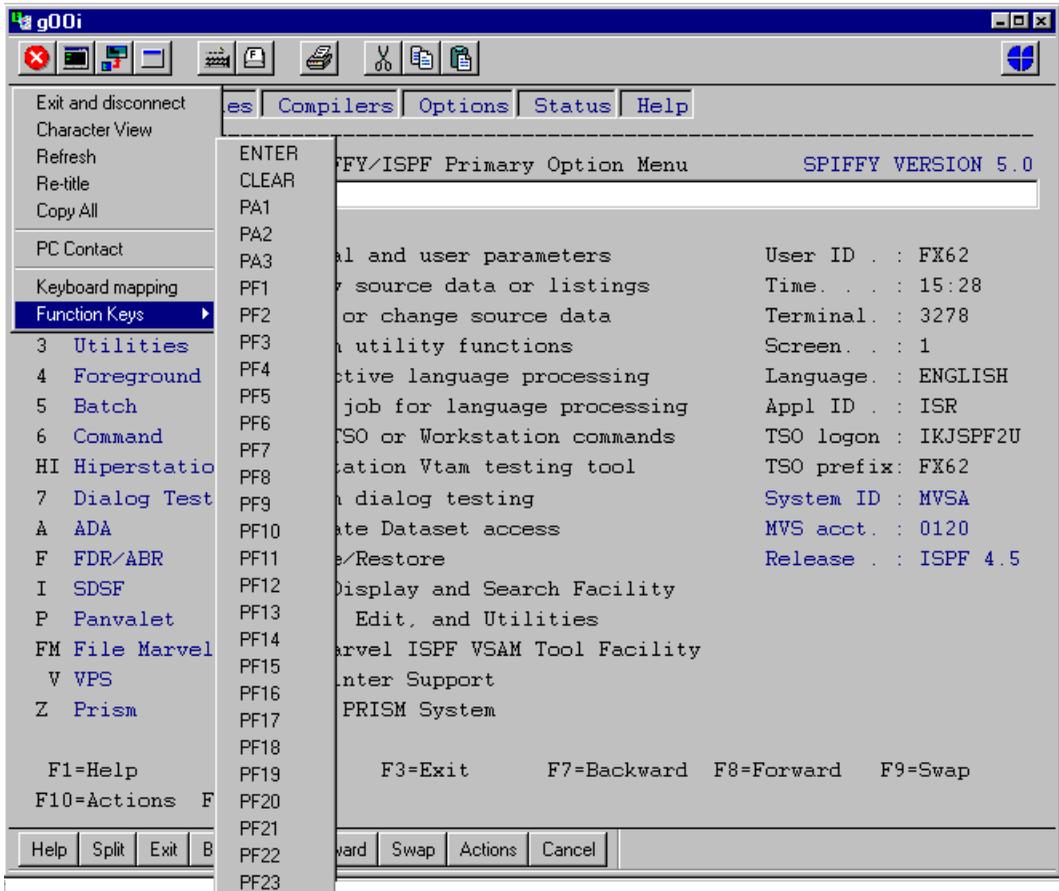
Button	Shortcut key	Action
	Alt+F4	Exit and disconnect from the host
	Alt+D	Present the gOOi character display window
	Alt+R	Refresh the current JIT window
	Alt+T	Retitle the current JIT window
	Alt+K	Present the keyboard mapping window
	Alt+F	Present a list of function keys for selection
	Ctrl+P	Print the current JIT window
	Ctrl+X	Cut the selected text to the clipboard
	Ctrl+C	Copy the selected text to the clipboard
	Ctrl+V	Paste from the clipboard to the current cursor position (paste will continue to subsequent entry fields if the size of the clipboard text exceeds the size of the entry field where the cursor is positioned)
		Present the Cincom web site using your Web browser

## Common gOOi functions (Pop-Up Menu)

Regardless of whether you elect to implement a user toolbar with JIT, a pop-up menu of common functions is always available by clicking mouse button 2 near the window border (outside any of the fields), as shown in the following example:



To provide an alternative to the keyboard, the FUNCTION KEYS option displays a pop-up that allows a choice of common keys, as shown in the following example:



The toolbar at the bottom of the window is created dynamically from the contents of the display, according to patterns defined in the Hotspots setting of gOOi. In the preceding window, the toolbar includes eight buttons because the display contains eight data items that match the defined pattern of F##=@.

The example has a title of “gOOi”. You may want to change this, especially if you have multiple JIT sessions. This title can be modified on the startup shortcut with the -GT startup option. The following example sets the JIT title to “gOOi – TSO”:

```
C:\Program Files\Cincom Systems\gOOi\gOOi.exe -"GTgOOi - TSO"
```

## Additional keyboard considerations for JIT

The behavior of the HOME key depends on the status of the field where the cursor is positioned when this key is pressed:

- ◆ If the field is highlighted when HOME is pressed, the cursor is positioned at the beginning of the field and the field will no longer be highlighted.
- ◆ If the field is not highlighted but the cursor is not at the beginning of the field, the cursor is moved to the beginning of the field.
- ◆ If the field is not highlighted and the cursor is at the beginning of the field, the cursor is moved to the first unprotected field on the display.

The behavior of the UP ARROW and DOWN ARROW keys is as follows:

- ◆ These keys have no effect if the cursor is in the first row of the form and UP ARROW is pressed, or if the cursor is in the last row of the form and DOWN ARROW is pressed.
- ◆ If the form has an entry field that begins in the same column of the row directly above (UP ARROW) or below (DOWN ARROW) the current field, the cursor is moved to that entry field regardless of any other fields on these preceding rows.
- ◆ Otherwise, the cursor is moved to the nearest preceding (UP ARROW) or following (DOWN ARROW) row that has an entry field. If this row contains multiple entry fields, the cursor is positioned in the first entry field from the beginning of the row.

## Using the title bar close options with JIT

There are two standard ways to close a window from the title bar:

- ◆ Select Close from the control menu (accessible via the icon  in the left corner of the title bar) or via the pop-up menu by clicking mouse button 2 in the body of the title bar.
- ◆ Click the Close button () in the right corner of the title bar (or use keyboard shortcut ALT+F4 for closing a window).

For both of the former approaches, JIT sends the escape sequence specified in Settings to the host. This sequence should be set to the key that is most commonly used to back out of a host screen, such as PF3 or PA2 (see “[Specifying the Host Connection Profile](#)” on page 52).

The Profile section of a host connection specifies the profile name, whether the host connection is via gOOi TCP/IP or through an emulator, and an escape sequence. GOOi includes two sample host connection profiles, SampleMainframe and SampleDigitalUNIX. Both of these profiles use the gOOi TCP/IP connection. You can modify a sample for use with your host, or create a new profile.

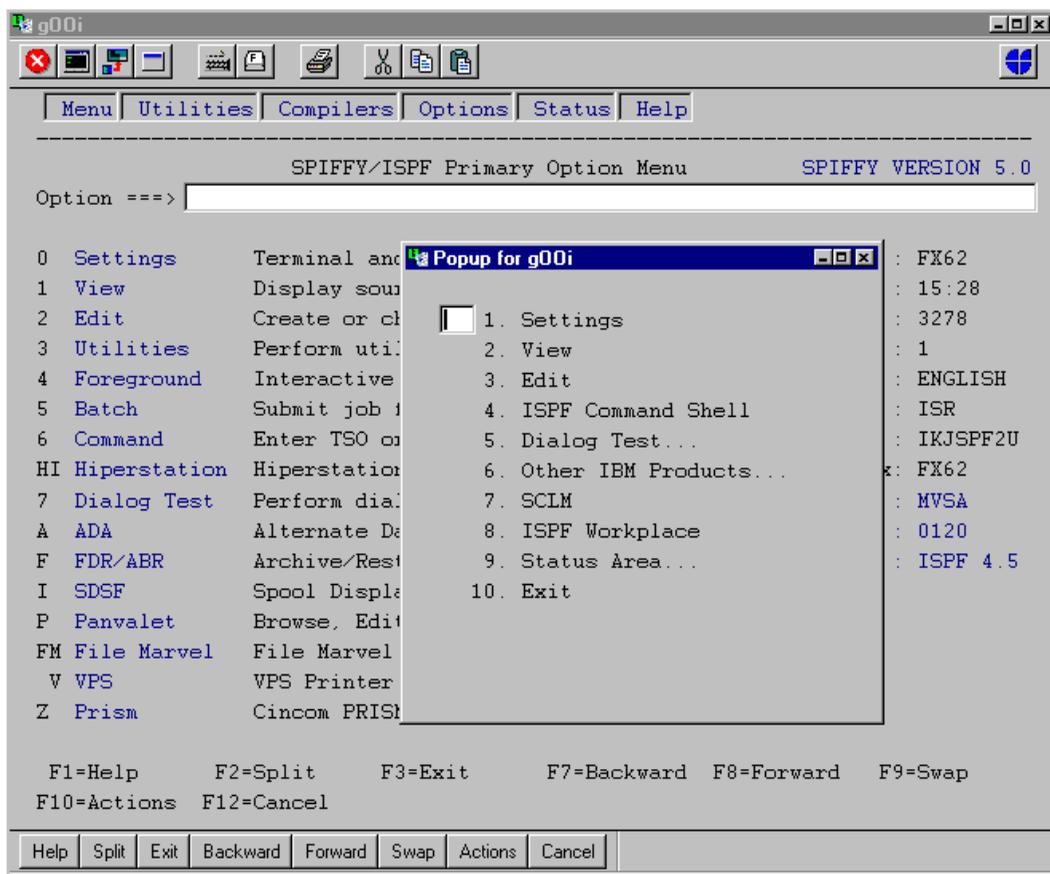
The connection type (TCP/IP or emulator) determines which of the other two sections of the host connection is active. A host connection profile can include information for both connection types, but gOOi connects to the host based on which of the two radio buttons is selected.

## JIT pop-ups

Before the JIT window is presented, gOOi checks to see if the host application includes a pop-up. To make this determination, gOOi looks for the four corners of the pop-up according to the following default patterns:

Corner	Pattern 1	Pattern 2
Top left	+ ---	. ---
Top right	--- +	--- .
Bottom left	+ ---	' ---
Bottom right	--- +	--- '

If gOOi finds a match for all four corners for any of the defined patterns, a pop-up displays and the underlying JIT form is disabled, as in the following example:



The pop-up remains displayed until the user enters the appropriate response (for example, PF3) to continue processing.

If any of the defined patterns causes a pop-up to display when the host application does not have a pop-up, it is easy to remove the pattern from JIT. For example, if pattern 1 is invalid for your host application(s), edit the gooi.ini file to remove this pattern. By default, the JIT section of gooi.ini includes four parameters that identify pop-up borders:

```
bottomLeftPopupBorder=+-----, '-----  
bottomRightPopupBorder=-----+, '-----'  
topLeftPopupBorder=+-----, .-----  
topRightPopupBorder=-----+, '----- .
```

To remove the second pattern, edit gooi.ini so that these parameters appear as follows:

```
bottomLeftPopupBorder=+-----  
bottomRightPopupBorder=-----+  
topLeftPopupBorder=+-----  
topRightPopupBorder=-----+
```

You can also edit gooi.ini to replace a pattern that is not applicable for your host application(s), or add an additional pattern. Patterns are delimited with commas, so any additional patterns that you insert must be preceded by a comma. Changes to gooi.ini for JIT take effect the next time the Host Monitor is started.



The color settings for the Character Display are:

- ◆ **Black.** Background
- ◆ **Green.** Protected, normal intensity
- ◆ **White.** Protected, high intensity
- ◆ **Yellow.** Unprotected, normal intensity
- ◆ **Cyan.** Unprotected, high intensity

The toolbar at the top of the Character Display has the following choices:

Button	Description
	Connects you to the host via TCP/IP
	Disconnects you from the host
	Presents the keyboard mapping window
	Presents a list of function keys for selection
	Toggles you to the JIT window
	Presents the Host Monitor

The status line at the bottom of the Character Display provides the following items of information:

- ◆ Connection status (Connected or Disconnected)
- ◆ Message (for example, “Press Reset to unlock the keyboard”)
- ◆ Insert status (empty or INS)
- ◆ Communication status (for example, SEND, X, PRO if trying to key into protected field)
- ◆ Cursor position (row, column)

## Character Display for VT data streams



The VT Character Display is only available when using the gOOi TCP/IP connection. If you are running gOOi with an emulator, the normal emulator display is the equivalent of the Character Display.

The gOOi VT Character Display is accessible via the mouse button 2 pop-up menu that was described earlier in this chapter. This display is a window containing a traditional VT character cell presentation:

The screenshot shows a window titled 'gOOi' with a menu bar (File, Edit, MiscKeys, FunctionKeys, Help) and a toolbar with icons for file operations. The main content area displays the following text:

```

M A N T I S

FACILITY SELECTION

Run A Program ..... 1          Transfer Facility ..... 12
Display A Prompter ..... 2      Edit MANTIS Messages ..... 13
Design A Program ..... 3        Directory Facility ..... 14
Design A Screen ..... 4         Universal Export Facility .. 15
Design A File Profile ..... 5    Update Shared Entity List .. 16
Design A Prompter ..... 6       Update Language Codes ..... 17
Design A User Profile ..... 7    MANTIS Maintenance ..... 18
Design An Interface ..... 8     Spectra ..... 19
Design An Ultra File View .. 9   Search Facility ..... 20
Design An External File view. 10 List of Current MANTIS Users. 21
Sign On As Another User .... 11  MANTIS security Patch Info . 22
                                Exit MANTIS ..... CANCEL

                                :   :

```

At the bottom of the window, there is a status bar with 'gto' on the left and '20.40' on the right.

You may want to go to the VT Character Display to check host response when testing a gOOi form, or to verify a host row/column position.

The toolbar at the top of the Character Display has the following choices:

Button	Description
	Copies selected text to the clipboard
	Pastes text from the clipboard
	Prints the display

The status line at the bottom of the Character Display provides the following items of information:

- ◆ Host machine to which this session is connected
- ◆ Cursor position (row, column)

# 3

## Preparing to generate gOOi forms

Before using gOOi to create a graphical interface for your host application, set the appropriate configuration options to ensure satisfactory results.

Follow these steps when preparing to generate gOOi forms:

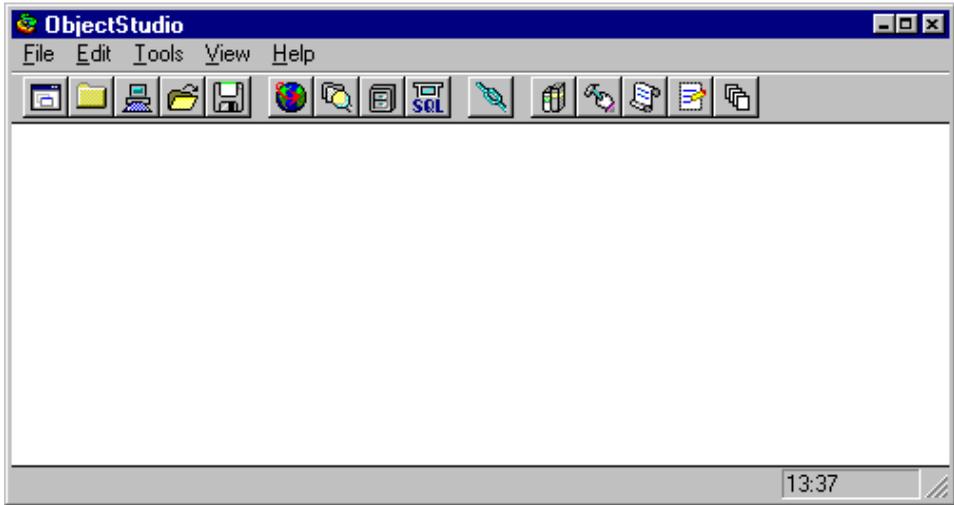
1. Load gOOi (see [“Loading gOOi”](#) on page 42 for more information).
2. Specify host screen configuration settings and gOOi form generation options (see [“Specifying gOOi generation options and settings”](#) on page 45 for more information).
3. (Optional) Perform pre-generation tasks depending upon your environment (see [“Using environment-dependent tools”](#) on page 82 for more information).
4. (Optional) Create a template for function key mapping. Default templates are provided (see [“Creating a template for function key mapping”](#) on page 91 for more information).
5. (Optional) Create a template for visual items (see [“Creating an application template for visual items”](#) on page 103 for more information).
6. (Optional) Create .exp files using the File Splitter (see [“Creating .exp files with the File Splitter”](#) on page 118 for more information).

After following these steps to configure gOOi, you are ready to generate gOOi forms.

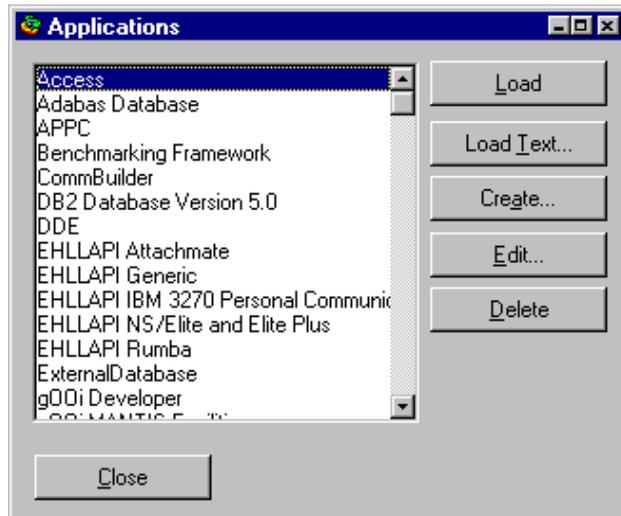
## Loading gOOi

To load gOOi, perform the following steps:

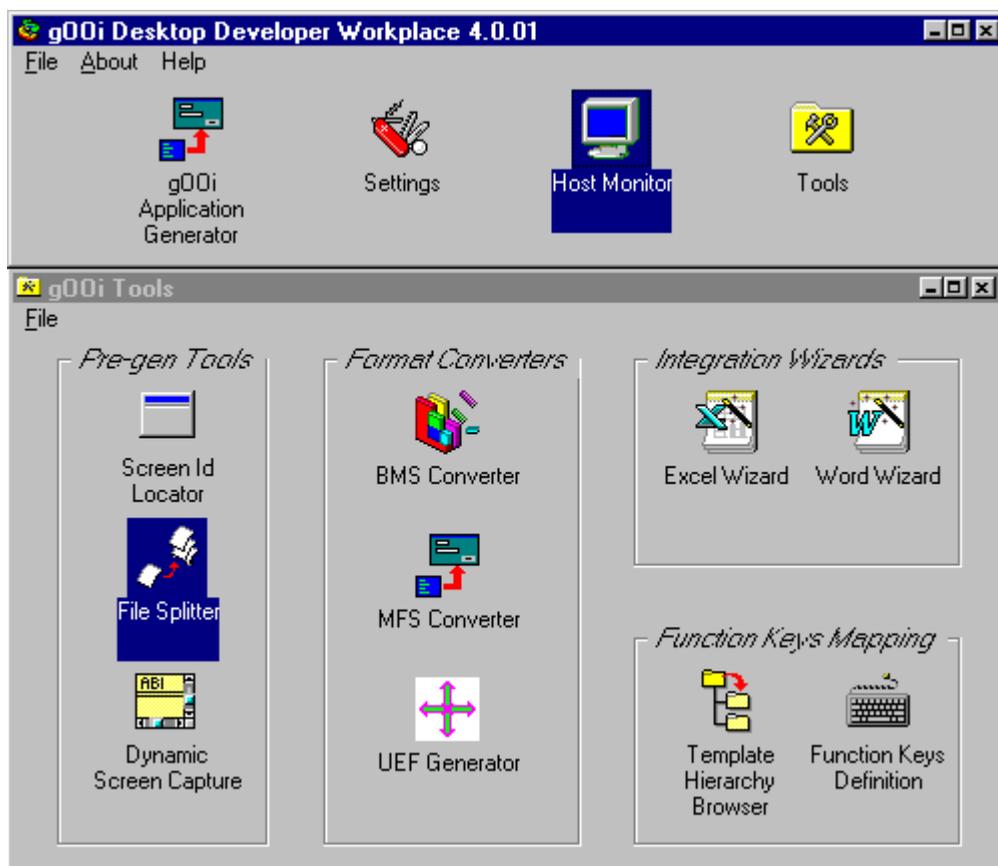
1. Run ObjectStudio. The following window displays:



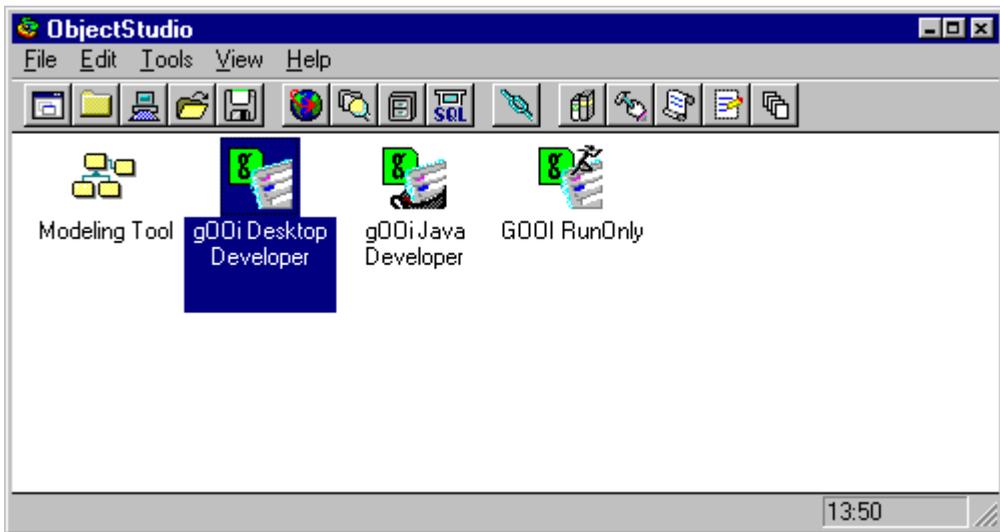
2. Click the mouse button 2 in the Work Area. A pop-up menu displays.
3. Choose Load Application. The following window displays:



4. Select gOOi Developer and click **LOAD**, or double-click on gOOi Developer. The following gOOi Workplace and gOOi Tools windows display:



- When you close the gOOi Workplace window, you are returned to the ObjectStudio Workplace Desktop. The desktop now includes a gOOi Developer icon in the workplace, as shown in the following example:

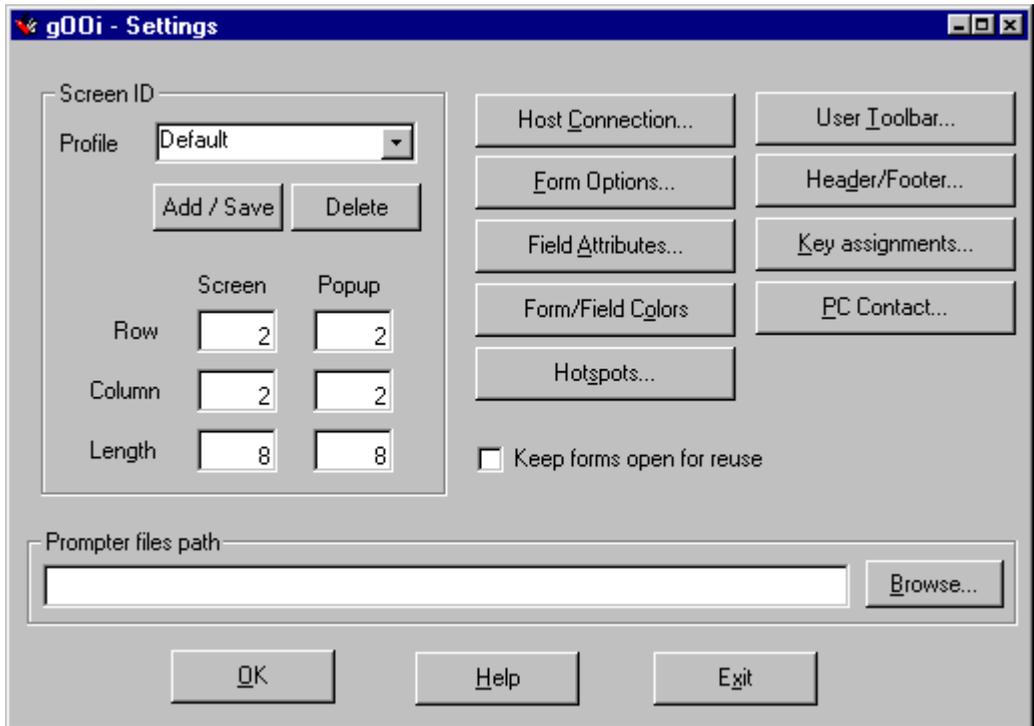


- Specify your host screen configuration settings and gOOi form generation options.

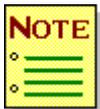
## Specifying gOOi generation options and settings

To specify host screen settings, perform the following steps:

1. At the Workplace Desktop, double-click the gOOi Developer icon.
2. Double-click the Settings icon to display the following window:



3. Specify the appropriate screen ID values to store in the profile (see “[Setting the screen ID](#)” on page 46) It may be useful to set up multiple profiles, since gOOi can dynamically switch between screen ID profiles at runtime.
4. Verify or change the other configuration settings for this profile as appropriate.



When you click **OK**, your specified values are written to the `gooi.ini` file in the directory where ObjectStudio is installed. You can update these values through the Settings facility for particular client workstations. The **Exit** button closes this window without saving changes.

The following describes the tasks you can perform using the Settings window.

## Setting the screen ID

The screen ID values specify the exact location and size of the screen ID for your host screens. This allows gOOi to automatically recognize the screens on the host (see “Screen ID rules” on page 307 for more information about screen IDs). gOOi can generate an application that includes screens that do not conform to your specifications, but the generator will issue warnings. The host navigation feature of gOOi can be used to identify host screens that do not have screen IDs

To specify screen ID settings:

1. Open the gOOi-Settings window, as shown in “Specifying gOOi generation options and settings” on page 45.
2. Select a profile name from the list, or enter a new name. (The profile name should begin with a letter and should not include any spaces or trailing blanks.)



---

gOOi comes with two preset screen ID profiles: MANTIS and AD/Advantage.

---

3. Enter the screen ID row locations for Screen and Pop-up. This value must be less than the height of the host screen.
4. Enter the screen ID column locations for Screen and Pop-up. This value must be less than the width of the host screen.
5. Enter the screen ID lengths for Screens and Pop-ups, respectively. This value must not exceed the host screen width.
6. Select Add / Save to save profile information to the gooi.ini file.

7. Do one of the following:

- Continue with other gOOi-Settings tasks.
- Click  to close the gOOi-Settings box and save the changes.
- Click  to close without saving the changes.

You can set up multiple screen ID profiles so that host applications can use different screen ID locations.

For example, all your Inventory screens might have a screen ID at row 1, column 70, whereas all your Payroll screens might have a screen ID at row 2, column 2. In this case, you would set up an Inventory profile and a Payroll profile with these values.

If you do not have screen IDs on your host screens, the gOOi Screen ID Locator can help you determine a suitable location to add these IDs (see “[Locating screen IDs](#)” on page 89 for more information).



---

gOOi attempts to generate forms with names comprised of the screen ID suffixed with Controller. For example, the Application Generator produces a form name of PAY001Controller for a host screen with ID PAY001.

If there is no valid screen ID on the host screen, however, gOOi will assign the form the name of the host screen suffixed with Controller. For example, if host screen QUERY1 lacks a screen ID, the Application Generator produces a form named QUERY1Controller. In lieu of a screen ID for QUERY1, host navigation code will then be required for gOOi to identify QUERY1 at runtime and display the gOOi form.

Lastly, it is possible with gOOi to associate a host screen that does have a screen ID with a form of a different name. In this instance, gOOi might detect a screen ID of CUST01, but display an associated form named CLIENT1. “[Screen Registry and AD/Advantage](#)” on page 301 explains how to implement this association.

---

## Application bundling



---

This feature is only available with the gOOi TCP/IP connection. It cannot be used when gOOi is running with an emulator.

---

Screen IDs are the normal method of host identification for gOOi. gOOi dynamically searches *all* saved screen ID profiles to find a profile that matches a host screen ID.

If gOOi cannot find a match at the location specified by the first screen ID profile, it searches the next profile, then the next, and so on, until a match is found. If a screen ID match is not found and the host screen is not identified via host navigation, gOOi reverts to the profile originally used and presents JIT (where applicable) or the character display.

With this feature, gOOi allows multiple applications to be bundled into one executable, if desired. It also allows for multiple screen ID locations within the same application or major application function.

Application bundling of screen ID profiles implements an MRU (most recently used) technique for screen ID searching. With this technique, when a profile changes, gOOi is not likely to search multiple profiles again until the application or major application function changes, thus increasing efficiency.

If you *do not* wish to use application bundling, simply specify only one screen ID profile for gOOi, and profile searching will not occur.

## Using MANTIS Dynamic CONVERSE

We do not recommend that you use MANTIS Dynamic CONVERSE (CONVERSE with (row, column) specification other than the default of (1, 1)) to display host screens (with their screen IDs) at a position other than the one at which they were designed. However, if *all* screen IDs in a single application are dynamically CONVERSEd to the *same* position, you can determine the row and column offset with the following algorithm:

*(designed-value - 1) + conversed-value = CONVERSE setting*

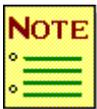
Apply this algorithm to both row values and column values. For example, if a host screen ID is defined at row 2, column 2 in the UEF file, and dynamically CONVERSEd at row 5, column 2, you must enter row 6, column 3 on the Settings screen:  $(2-1)+5 = 6$ ;  $(2-1)+2 = 3$ .

If multiple screens are being displayed through CONVERSE WAIT SET, they can be generated together, with each one using a different display row and column. For example, if screen1 is conversed at row 1, column 1; and screen2 is conversed at row 4, column 49; then a single gOOi panel could be generated consisting of screen1 (row 1, column 1) and screen2 (row 4, column 49).

Be careful to specify exactly the screen ID location that is valid for your application. Otherwise, gOOi cannot identify the host screens and invoke their corresponding gOOi forms.

## Keeping forms open for reuse

The Keep forms open for reuse option retains gOOi forms in memory for faster access.



---

This increases the amount of memory used by gOOi.

---

To keep forms open for reuse:

1. Open the gOOi-Settings window, as shown in “[Specifying gOOi generation options and settings](#)” on page 45.
2. Select the ‘Keep forms open for reuse’ check box.
3. Do one of the following:
  - Continue with other gOOi-Settings tasks.
  - Click  **OK** to close the gOOi-Settings box and save the changes.
  - Click  **Exit** to close without saving the changes.

If a gOOi application has many forms, you may not be able to keep them all open due to limited memory. Therefore, generate your forms with this option on and turn the option off for smaller or little-used forms.

For example, TESTFORM was generated with this option set. Follow these steps to turn off the option for TESTFORM:

1. Ensure that the application that includes TESTFORM is loaded into ObjectStudio.
2. In Class Browser, select TESTFORMHostObject from list of classes.
3. Select customInitialize from the list of methods.
4. Change the line in the source code that reads: “self keepOpen: true.” to instead read: “self keepOpen: false.” and save the method using Method/Save on the menu.



---

The Keep forms open for reuse option must be on in Settings for this option to be in effect at run time. If the option is off at run time, the gOOi forms will not be kept open even if the option was on at generation time.

---

## Selecting the Host Connection

You can choose the type of host connection on which gOOi will interact with the host. Follow these steps to choose the host connection:

1. Open the gOOi-Settings window (see “[Specifying gOOi generation options and settings](#)” on page 45).



Another route to get to the Host Connection selection of gOOi is via the File menu of the Host Monitor.

2. Click **Host Connection**, and the following window is displayed:

3. Specify a Profile and its connection data, either TCP/IP or Emulator. Click **OK** to close the Host Connections window and save the changes, or **Cancel** to exit without saving the changes.

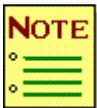
## Specifying the Host Connection Profile

The Profile section of a host connection specifies the profile name, whether the host connection is via gOOi TCP/IP or through an emulator, and an escape sequence. gOOi includes two sample host connection profiles, SampleMainframe and SampleDigitalUNIX. Both of these profiles use the gOOi TCP/IP connection. You can modify a sample for use with your host, or create a new profile.

The connection type (TCP/IP or emulator) determines which of the other two sections of the host connection is active. A host connection profile can include information for both connection types, but gOOi connects to the host based on which of the two radio buttons is selected.

### Escape sequence

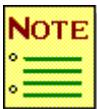
The Escape sequence entry is the host character sequence used to cancel or quit your application. It is used when you close a gOOi form, a prompter window, or a pop-up window. In such cases, gOOi must close not only your graphical window, but also the corresponding host screen.




---

The gOOi TCP/IP connection for an IBM mainframe can recognize a logical key name, such as PA2 (the default value) or PF3, or the EHLLAPI equivalents of these logical keys. When using gOOi with an emulator that supports EHLLAPI, the proper mnemonic must be used, such as @y for PA2. This value can vary with each OpenVMS/UNIX host emulator (see “[Host-PC translation tables](#)” on page 311 for a list of mnemonics).

---




---

The gOOi TCP/IP connection for an OpenVMS/UNIX host uses a 'F1 -' for the escape sequence. This setting represents a keyboard sequence of the F1 key followed by the minus (-) key.

---

Since gOOi sends the escape sequence that is specified in Settings when a form is closed via the upper right corner button, this specification should be set to the key that is most commonly used by your applications to exit from a display. For example, use PF3 (TCP/IP connection) or @3 (emulator) if PF3 is typically the key pressed to exit from a display.

You may need to handle some displays differently. For example, if the application flow is from form A to B to C to D to E, form C may need PF12 to close and the other four forms may need PF3. If the user closes form C via the top right corner button, the PF3 sent because of the @3 in Settings will not have the desired affect.

It is easy to override the @3 as needed in individual forms (such as form C) by adding the escapeSequence class method to the form, as shown in the following example:

```
escapeSequence  
^ '@c'.
```

In the preceding example, @c EHLLAPI code will cause PF12 to be sent to the host if the upper right corner button is pressed.

## Specifying a TCP/IP Connection

The TCP/IP section of a host connection requires a host address, port number, connect timeout value, and character display model. An optional start up file can also be designated.

### Host address

The host address can be either a logical name such as MVS or UNIX1, or an IP address such as 12.1.2.60.

### Port number

There is usually a standard port used to connect to a host machine, such as 23. Please check with your system administrator if you are unsure what to specify for this setting.

### Connect timeout value

The Connect timeout value is the time period (in milliseconds) that gOOi will wait when trying to connect to the host. An error message is displayed if gOOi cannot connect within this time limit.

### Character display model

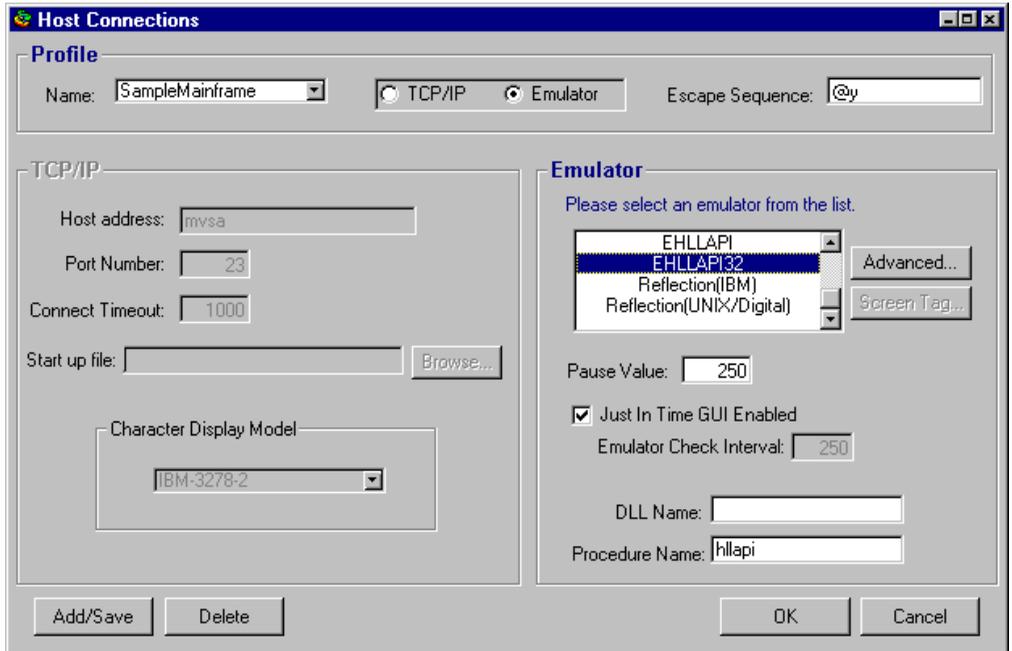
Be sure to specify the display model that corresponds to your environment. The VT220 selection is for running with an OpenVMS/UNIX host, and the IBM-3278-2 and IBM-3278-2-E choices are for connecting to an IBM mainframe.

## **Start up file**

The start up file entry allows you to specify a file that contains code for automating the repetitive task of logging on to a host system. A start up file uses the host navigation feature of gOOi to enter the data and keystrokes necessary to bypass host screens that are normally displayed during logon. gOOi includes two sample files that demonstrate implementations of the start up feature. The gOOiStartUpSampleUNIXDigital.txt file logs onto a UNIX host and signs on to MANTIS as the MASTER user. The gOOiStartUpSampleMainframe.txt file logs on to an IBM mainframe host, signs on to CICS, and starts the MANT transaction.

## Specifying an Emulator Connection

The Emulator section of a host connection lists the external emulators that gOOi supports. If EHLLAPI or EHLLAPI32 is selected, the display changes as shown in the following example:



To use generic EHLLAPI support, you must specify the name of your emulator's EHLLAPI DLL, along with the EHLLAPI procedure name within that DLL. The default procedure name is hllapi (the name is case sensitive).

### Pause value

The Pause value is the time period (in milliseconds) that the host must be idle to consider work completed.

For OpenVMS and UNIX, there is no function or flag to verify that the host is not running a task. For IBM mainframe emulators with EHLLAPI, the system clock may flicker, falsely indicating that the host data transmission is complete. Therefore, a Pause value is needed for gOOi to know when data transmission from the host is complete.

The default pause value is 250 milliseconds. Initially accept the default value. If you notice unexpected behavior in your gOOi interface, increase this value. If your gOOi application is working properly but you would like to improve performance, try decreasing this value. The optimum value depends on your host and your network speed. Use as low a value as possible without interrupting an incomplete process.

### Just In Time GUI Enabled

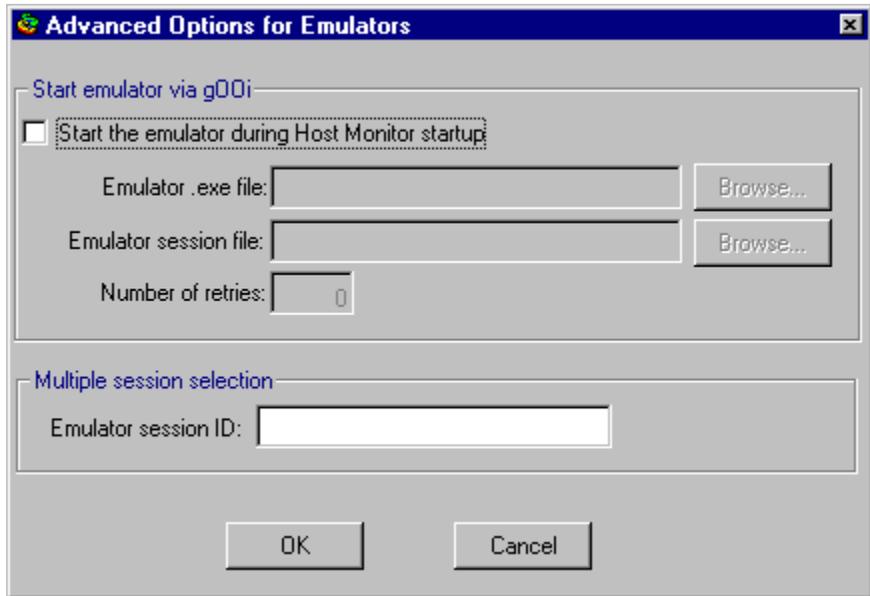
The Just In Time (JIT) window of gOOi is available when running with any of the IBM mainframe emulators that gOOi supports. This option is on by default, and can be turned off if JIT is not desired.

### Emulator Check Interval

The Emulator Check Interval specifies how often gOOi checks the emulator presentation space for host screen activity. The default value is 250 milliseconds. If gOOi does not seem to respond quickly to host changes, try decreasing this value. The optimum value depends on your host and your network speed.

### Advanced Options for Emulators

The Advanced button brings up the following window:



The check box selects whether the emulator should be started during gOOi Host Monitor start up (emulator .exe and session files must be designated). If you select this check box, specify a number of retries that is large enough to allow the emulator to fully initialize before the Host Monitor attempts to connect.

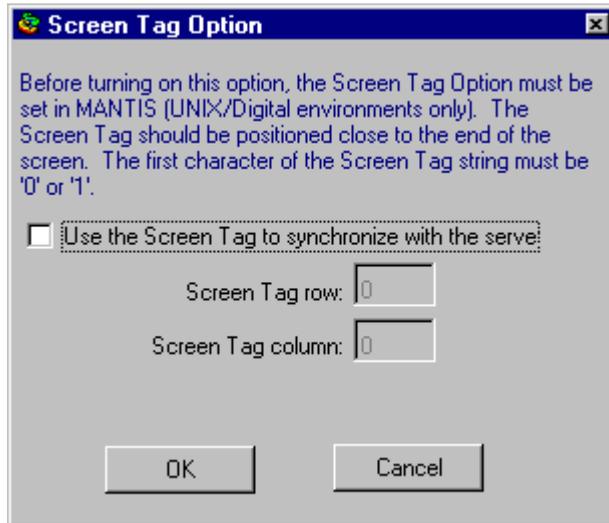
The entry field for multiple session selection designates the session name to which gOOi should connect. This field can be used to ensure that gOOi connects to the desired host session when multiple sessions are active. In the case of Reflection for UNIX/Digital, the session name specified on this dialog must match the name specified via the 'Setup>View Settings' menu of the emulator. Within this emulator menu selection, choose 'OLE Server Name' to enter the session name.

The session name is a single character for the other emulators supported by gOOi. Please see "[Emulator considerations](#)" on page 297 for a description of how to set this name for the emulator you are using.

## Screen Tag option



When using the direct gOOi TCP/IP connection with MANTIS for OpenVMS/UNIX, start MANTIS with the 'gooi' parameter. Do not use this Screen Tag option.



The Screen Tag option applies only to MANTIS running in an OpenVMS/UNIX environment. Please refer to the Options Compiler section of the *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300 for a description of the Screen Tag Option. If this option is used, the screen tag must be positioned close to the end of the screen, and the first character of the Screen Tag string must be '0' or '1'.

This option can improve performance and reliability when running gOOi with MANTIS for OpenVMS/UNIX via an external emulator. Host synchronization problems will occur if this option is turned on in gOOi but not in MANTIS, as gOOi will watch for this tag but will not find it.

## Specifying host field attribute matching

This setting determines whether you want gOOi to match host field attribute settings. This option instructs gOOi to interrogate settings on the host, then set the fields on the gOOi form accordingly.

To interrogate host field attribute settings, check any or all of the attributes you want to use from the host.

If you don't select any attributes, gOOi ignores the host field attributes.



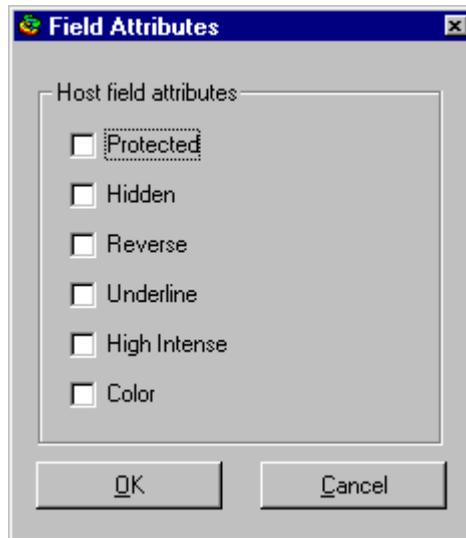
---

Field attributes do not apply to OpenVMS/UNIX environments. In addition, the gOOi TCP/IP connection for IBM mainframes does not support the Reverse, Underline, and Color attributes.

---

To specify field attributes:

1. Open the gOOi-Settings window, as shown in “[Specifying gOOi generation options and settings](#)” on page 45.
2. Click **Field Attributes**. The following window displays:



3. Check or uncheck the appropriate boxes to select field attributes:

Attribute	Description
Protected	Prevents the field from being edited
Hidden	Prevents the field from being displayed
Reverse	Displays the field with reversed foreground and background colors
Underline	Displays a line under the field
High Intense	Displays the field with brighter intensity
Color	Displays the field in color

Field attribute settings apply to all forms that you generate, but individual forms can be customized by modifying the attributes *method* in ObjectStudio's Designer or Class Browser. The generated form name consists of the screen ID name suffixed with Controller (for example, screen ID WHSE001 has a form name of WHSE001Controller).



Running with all of the host field attributes unchecked improves performance. If no attributes are selected, gOOi does not query the emulator for these attributes when displaying the gOOi form. The potential impact is most significant for host screens with many fields.

Be careful to select those attributes that are dynamically set for host screens. For example, if a host screen field is dynamically protected or unprotected during application execution, select the Protected check box. This ensures that gOOi looks at the host data stream to determine the proper response.

If only one form out of a group of forms needs an attribute selected, modify the attributes method for that form's controller. For example, to interrogate the Protected attribute you would change the method from:

```
^ {}
```

to:

```
^ {#Protected}.
```

With this strategy, gOOi only queries the emulator for a particular attribute when presented with a relevant form. You retain a performance benefit for the rest of the forms because gOOi does not issue emulator queries for attributes for them.

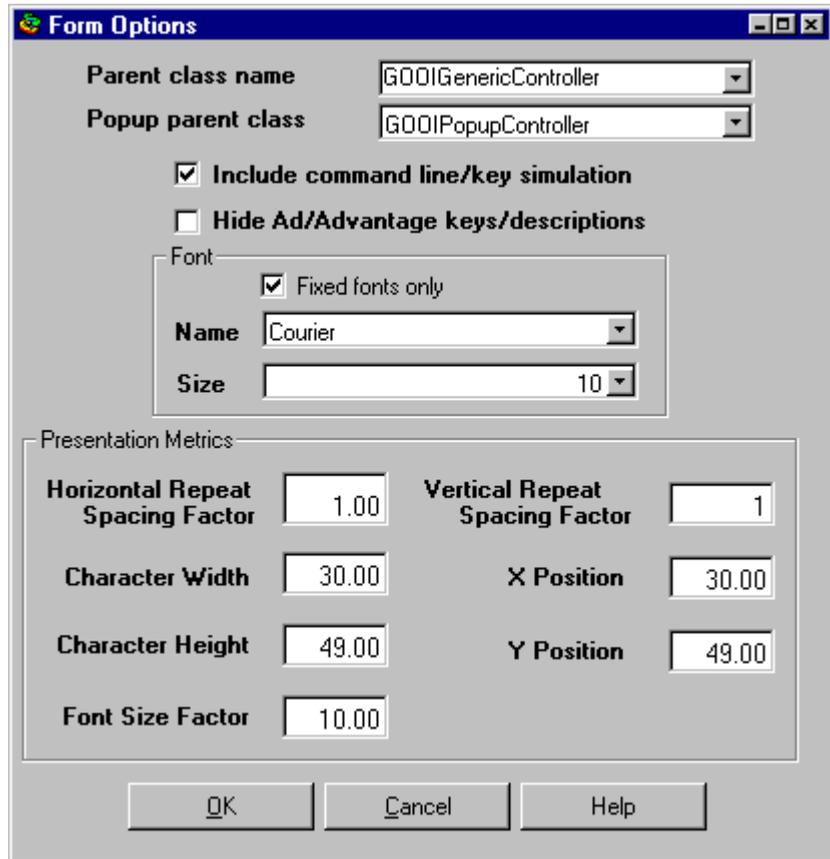
4. Click **OK** to return to the Settings screen.
5. Do one of the following:
  - Continue with other gOOi-Settings tasks.
  - Click **OK** to close the gOOi-Settings box and save the changes.
  - Click **Exit** to close without saving the changes.

## Form and Color options

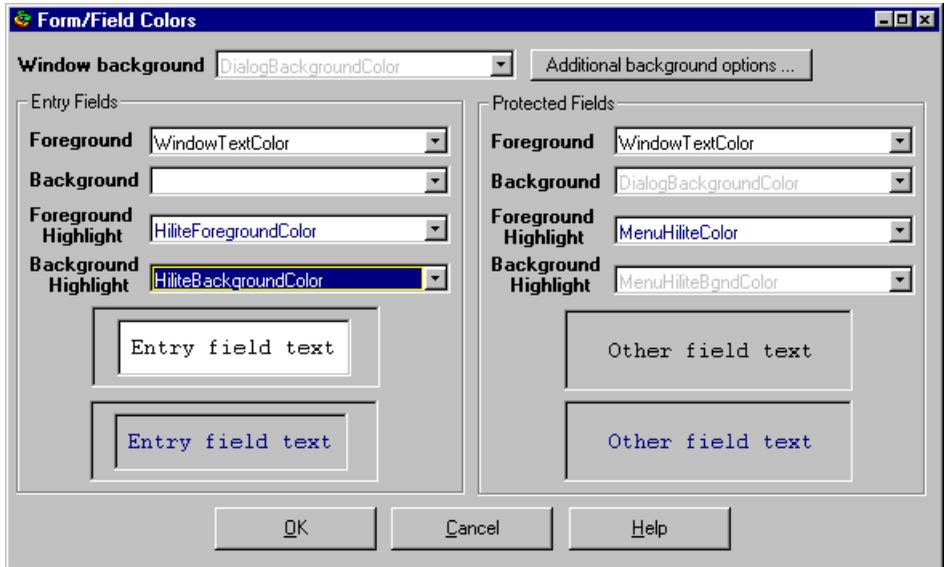
Form and Color options provide parameters to define how to generate your gOOi forms and to change the appearance of the JIT window.

To specify Form and Color options:

1. Open the gOOi-Settings window, as shown in “[Specifying gOOi generation options and settings](#)” on page 45.
2. Click **Form Options**. The following window displays:



The Form/Field Colors window also displays when you select Form Options:



- Using the pull-down menu in the Form Options window, set the parent class name.

If your host application requires function keys and you want to retain function key operation, a parent class (template) can specify your function key mapping. gOOi is delivered with these sample templates:

- IBM3191Controller for IBM hosts (the GOOIGenericController default parent class has key mapping for IBM hosts, but is not a modifiable template)
- GOOIVTTerminalController for OpenVMS/UNIX hosts when using the gOOi direct TCP/IP connection
- ASCIIterminalController for use with KEA!
- Reflection2TerminalController for use with Reflection2

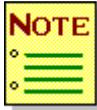
See “[Creating a template for function key mapping](#)” on page 91 for information about creating a new subclass to define function key mapping. If in doubt, you can try the appropriate sample. You can always return later to specify a subclass, then regenerate your application.



---

The default GOOIGenericController parent class has key mapping for using gOOi with IBM hosts. A different parent class, such as GOOIVTTerminalController, is necessary for appropriate key mapping when using gOOi with OpenVMS/UNIX hosts.

---



The default parent class of GOOIGenericController gives function key mapping as defined in the Key assignments of Settings. If this default is used, each generated form will have menu entries for PFkeys and MiscKeys as follows:

MiscKeys	PFkeys
CLEAR (Pause)	PF1 (F1)
ENTER (ENTER)	PF2 (F2)
HOME (HOME)	PF3 (F3)
PA1 (PageUp)	PF4 (F4)
PA2 (PageDown)	PF5 (F5)
PA3 (ShiftPause)	PF6 (F6)
	PF7 (F7)
	PF8 (F8)
	PF9 (F9)
	PF10 (F10)
	PF11 (F11)
	PF12 (F12)
	PF13 (ShiftF1)
	PF14 (ShiftF2)
	PF15 (ShiftF3)
	PF16 (ShiftF4)
	PF17 (ShiftF5)
	PF18 (ShiftF6)
	PF19 (ShiftF7)
	PF20 (ShiftF8)
	PF21 (ShiftF9)
	PF22 (ShiftF10)
	PF23 (ShiftF11)
	PF24 (ShiftF12)

Any changes you make to the key assignments will be reflected in the menu entries. These menu entries are only appropriate for IBM mainframe hosts. The sample template for your emulator or user-created parent classes should be used for OpenVMS/UNIX hosts.

The parent class name can also be used to specify a template whose controls will be copied to gOOi forms during generation. For example, using ObjectStudio Designer, you could build a template named Template1Controller that has a Next button and a Previous button. As long as Template1Controller is specified as the parent class, these two buttons are copied to all the forms you generate. See [“Creating an application template for visual items”](#) on page 103 for further information.

4. Set a pop-up parent class name. The parent class name must begin with an uppercase, alphabetic character.

If your MANTIS application uses pop-ups and you want to retain function key operation, a parent class (template) can specify your function key mapping. gOOi is delivered with these sample pop-up templates:

- IBM3191PopupController for IBM hosts (the GOOIPopupController default parent class has key mapping for IBM hosts, but is not a modifiable template)
- GOOIVTPopupController for OpenVMS/UNIX hosts when using gOOi with the direct TCP/IP connection
- ASCIIPopupController for use with KEA!
- Reflection2PopupController for use with Reflection2



---

The default GOOIPopupController parent class has key mapping for using gOOi with IBM hosts. A different parent class, such as GOOIVTPopupController, is necessary for appropriate key mapping when using gOOi with OpenVMS/UNIX hosts.

---

See “[Creating a template for function key mapping](#)” on page 91 for information about creating a new subclass to define function key mapping. If in doubt, you can try the appropriate sample. You can always specify a subclass later, and then regenerate your application.



---

The default pop-up parent class of GOOIPopupController gives function key mapping as defined in the Key assignments of Settings. If this default is used, each generated pop-up will have menu entries for PFKeys and MiscKeys as shown in the preceding note concerning GOOIGenericController.

---

5. Check whether to include command line/key simulation (MANTIS only). This simulation notifies gOOi to generate two entry fields on line 24 of your generated gOOi forms:
  - The first field (positions 1–72) is for the MANTIS command line.
  - The second field (positions 74–79) is for the MANTIS key simulation.

This is necessary because these two fields are not described in the UEF file created from your MANTIS application.

This option does not apply if you generate your MANTIS screen as an FUL display. If your MANTIS screen is not FUL and you want to retain these fields, select this check box. Alternatively, the end user can press ALT+TAB to access the host screen if running with an emulator, or ALT + D to access the Character Display if using the gOOi TCP/IP connection. After entering the data, the end user can press ENTER to return to gOOi from the emulator, or ALT + D to toggle back from the Character Display.

6. Check whether to hide AD/Advantage keys/descriptions. If you are using gOOi with AD/Advantage, selecting this check box causes the PF keys and descriptions from the host screen not to display on the gOOi form.
7. Set font properties. Text properties allow you to govern the use of fonts and colors in your generated forms. Use the following guidelines to adjust these values:

Field	Default	Description
Fixed fonts only	(selected)	Restricts the font selection to the nonproportional fonts available on your system.
Font name	Courier	Specifies the font for your entry fields and static text. The list is determined by the fonts on the developer's workstation. If the font is not available on the end-user workstation, Windows substitutes the closest font.
Font size	10	Specifies the font's point size for entry fields and static text.



Host screens use nonproportional fonts to ensure uniform text alignment (each character takes up the same amount of space). Use a nonproportional font for gOOi forms to maintain proper text spacing. Nonproportional is the gOOi default.

For more information, see “[Changing a field’s color, font, and/or justification](#)” on page 169.

8. Set color properties. Color properties let you assign different colors to your screens, fields, and text as shown in the following table:

Field	Default	Description
Window background color	Dialog Background Color	Specifies the window background color
Entry Field background color	Entry Field Color	Specifies the background color for entry fields
Entry Field foreground color	Window Text Color	Specifies the foreground color for entry fields
Highlighted Entry Fields background color	Hilite Background Color	Specifies the background color for highlighted entry fields
Highlighted Entry Fields foreground color	Hilite Foreground Color	Specifies the foreground color for highlighted entry fields
Protected Field background color	Dialog Background Color	Specifies the background field color for protected fields
Protected Field foreground color	Window Text Color	Specifies the foreground field color for protected fields
Highlighted Protected Fields background color	Menu Hilite Bgnd Color	Specifies the background color for highlighted protected fields
Highlighted Protected Fields foreground color	Menu Hilite Color	Specifies the foreground color for highlighted protected fields

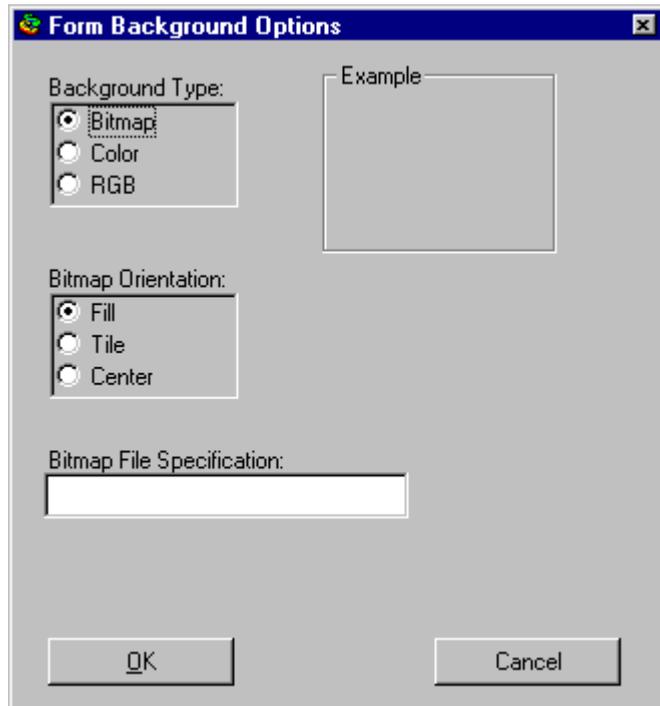
When you select a color, the effect displays in the example box. For more information, see “[Changing a field’s color, font, and/or justification](#)” on page 169



Some color choices are fixed, while others are standard parameters of each Windows configuration. This means that if you specify these standard parameters, your gOOi forms default to the colors specified on end-user workstations. This has the advantage of causing gOOi forms to display in the colors that each end user prefers.

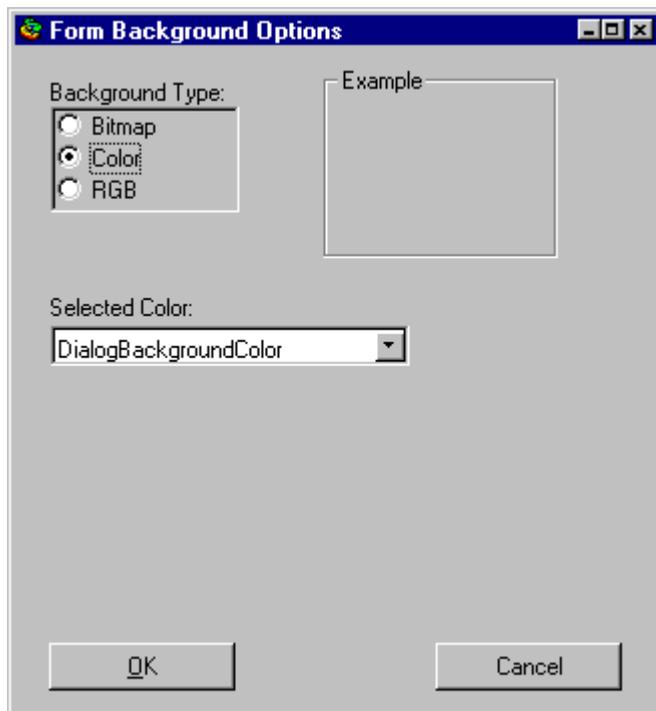
This same consideration means that the appearance of your color choices in the lists are affected by your Windows settings. Thus, some choices may appear blank because they do not contrast with the field color. However, if you click these apparently blank options, they will take effect.

- Set a background that will be assigned to your gOOi forms during generation. The window changes according to the Background Type that you choose. If you select ‘Bitmap’, you will see the following:



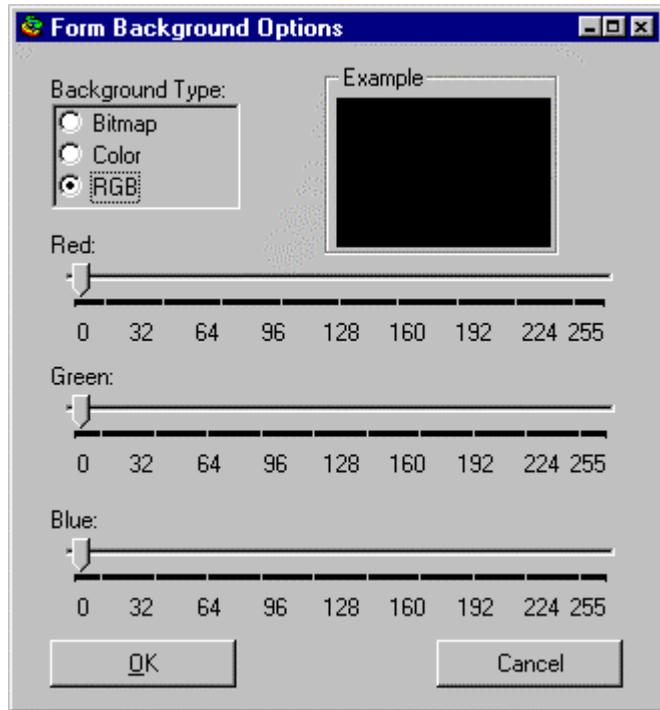
You can specify the Bitmap File to use.

If you select 'Color', you will see the following:



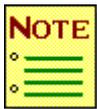
You can choose from a drop down list of logical colors such as DialogBackgroundColor, and physical colors such as White. Logical colors are adjusted to the color settings of the PC where gOOi is deployed. This means that DialogBackgroundColor could be gray on one PC, but white on a different PC.

If you select 'RGB', you will see the following:



You can adjust the amount of Red, Green, and Blue to create a custom color. The Example shows the effects of the changes that you make to the three slider bars.

10. Set presentation properties. Presentation properties allow you to easily adjust the appearance of generated forms.



We recommend that you accept the default values for presentation properties. After generating and viewing the resulting gOOi interfaces, you can use the following guidelines to assess any necessary adjustments.

Use the following guidelines for setting presentation properties:

Setting	Units	Default	Description
Horizontal Repeat Spacing Factor	0.1 mm	1.75	Specifies the gap between fields repeated horizontally across the screen.
Vertical Repeat Spacing Factor	1.0 mm	1.00	Specifies the gap between fields repeated vertically down the screen.
Character Height	0.1 mm	48.00	Specifies the height of boxes containing text characters (static or dynamic).
Character Width	0.1 mm	27.00	Specifies the width of boxes containing text characters (static or dynamic).
X Position	0.1 mm	27.00	Sets the horizontal dispersion value for all fields displayed on a gOOi form. Adjust this factor in conjunction with Y Position.
Y Position	0.1 mm	48.00	Sets the vertical dispersion value for all fields displayed on a gOOi form. Adjust this factor in conjunction with X Position.
Font Size Factor		10.00	Governs the size of controls relative to the font selected. Adjust this factor in conjunction with Font size to prevent overlap.

11. Click **OK** to save any changes and return to the Settings window.

### Using hotspots

Hotspots are host screen data descriptions of how the user can invoke a command via a function key. gOOi can recognize these contextual function keys and generate a toolbar for them.

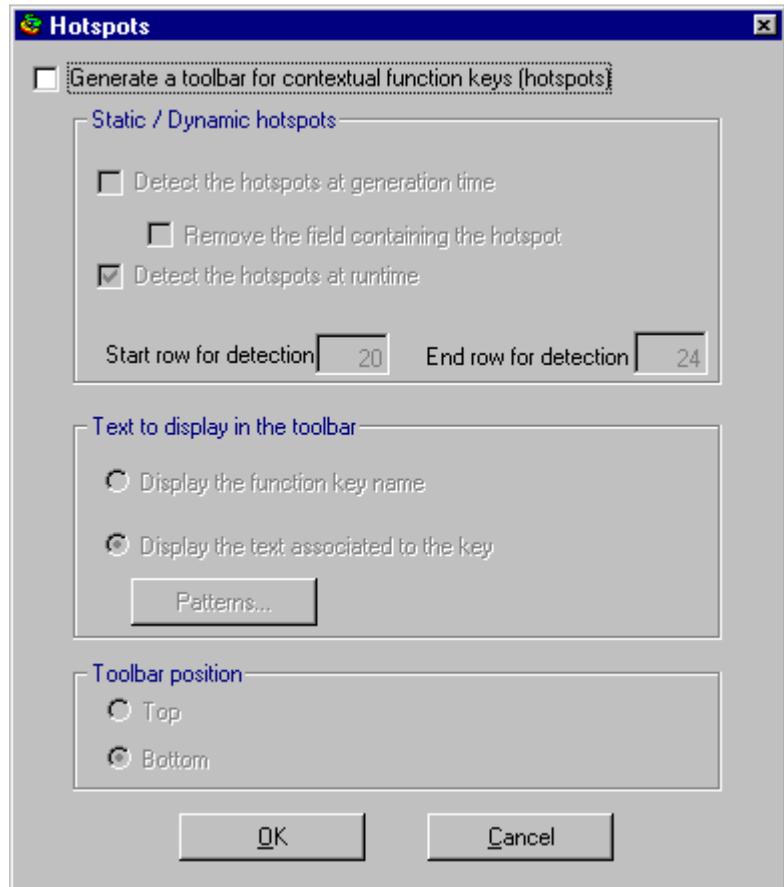
For example, a host screen might contain a line that reads:

PF1=Help PF3=Exit PF7=Backward PF8=Forward

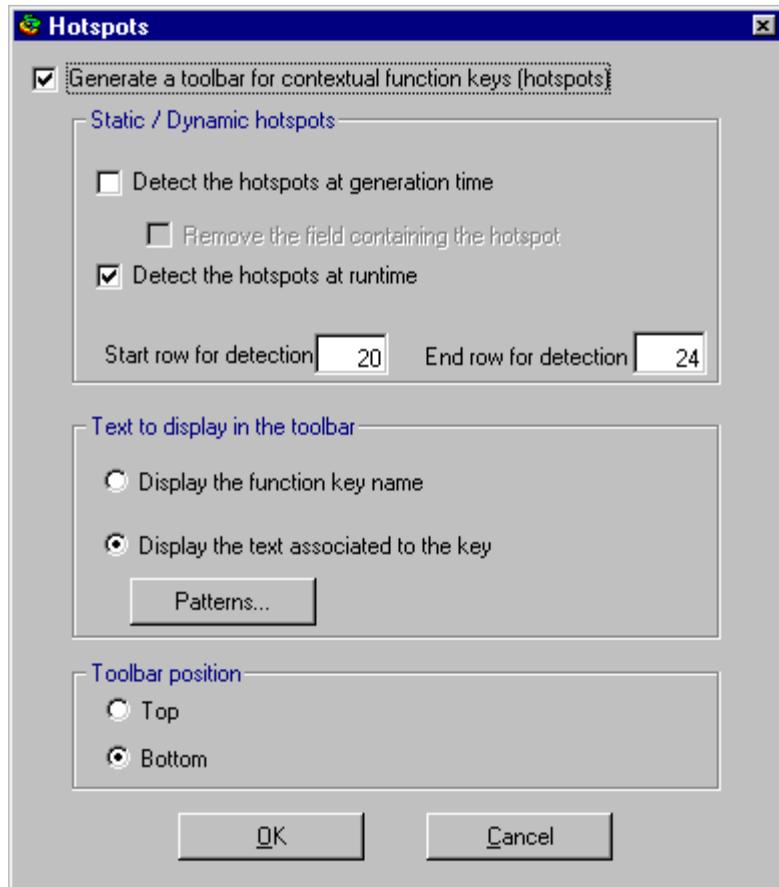
gOOi can automatically generate a toolbar that will contain four buttons for these hotspots. In this example, the toolbar would have buttons that read: **Help**, **Exit**, **Backward**, and **Forward**. When the user selects one of these toolbar buttons at run time, the appropriate function key (for example: PF1 for Help) is sent to the host.

To specify hotspot settings:

1. Open the gOOi-Settings window, as shown in “[Specifying gOOi generation options and settings](#)” on page 45.
2. Click **Hotspots**. The following window displays:



3. To enable hotspots, select the 'Generate a toolbar for contextual function keys (hotspots)' check box. The hotspot options become accessible as shown in the following example:



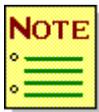
4. Specify whether to generate the toolbar when the application is generated (static) or at run time (dynamic). Detecting hotspots at run time is useful if your host screens have large fields for function key instructions that are filled by the host application. Select the 'Remove the field containing the hotspot' check box (static only) if you only want a toolbar button and not the on-screen command description.

## 5. Specify the button label.

- Choose either the function key name (for example, PF3) or the command description (for example, Exit).
- If you display text associated with the key, specify the text patterns that gOOi should look for on the host screen. In the pattern definition:
  - # represents a number,
  - @ represents text to extract to the button title,
  - ^ indicates that the text to extract is on the next line.
- Examples of text patterns are shown in the following table:

Pattern	Matches the strings:
PF## - @	PF01 - Next PF02 - Previous PF1 - Next PF2 - Previous
Press %ENTER% to @	Press ENTER to Continue
Please press F## to @	Please press F1 to Continue
F##^	F1 Help

Click **Patterns** to view example pattern information.



The pattern (Please press F## to @) does not match the string “please press F1 to Continue” because patterns are case-sensitive.

6. Select the toolbar position (top or bottom of the form).
7. Click **OK** to return to the Settings window.

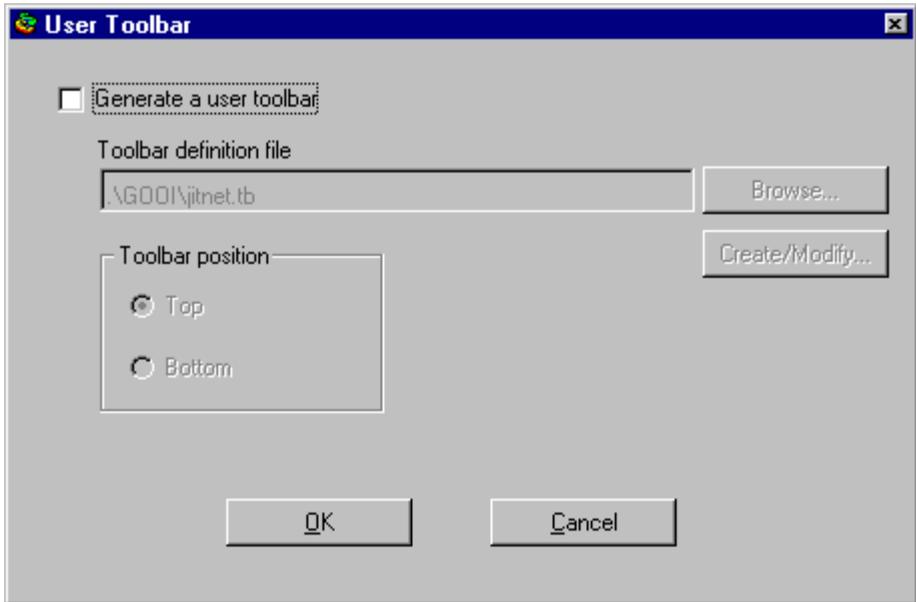
8. Do one of the following:
  - Continue with other gOOi-Settings tasks.
  - Click **OK** to close the gOOi-Settings box and save the changes.
  - Click **Exit** to close without saving the changes.

## Creating a user toolbar

You can build custom toolbars that invoke keystrokes, start other programs, or perform specified actions such as showing the host screen. Toolbar items can be either buttons or icons, and can have associated tooltips.

To create a user toolbar, perform the following steps:

1. Open the gOOi-Settings window, as shown in “[Specifying gOOi generation options and settings](#)” on page 45.
2. Click **User toolbar**. The following window displays:



3. To create a toolbar, select the 'Generate a user toolbar' check box. This enables access to related options.

4. Enter a toolbar definition file name or click **Browse** to locate an existing file. Click **Create/Modify** to build a new toolbar.



The jitnet.tb file that is installed in the gOOi subdirectory of ObjectStudio is a sample of a user toolbar.

5. Select a toolbar position (top or bottom). The default toolbar position is at the top of the gOOi form.
6. Click **OK**. The following window displays:

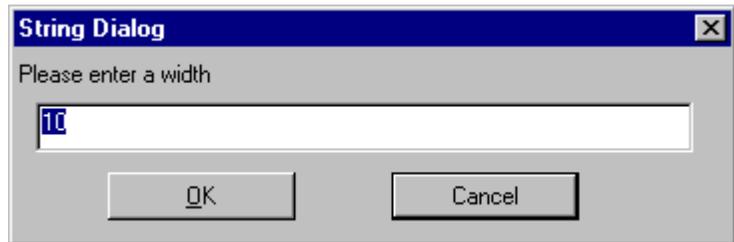


To locate and use an existing toolbar definition file, click **Load file**.

7. Select whether the toolbar item to be added will:
  - Invoke a keystroke (select a key from the list)
  - Start a program (enter a file name and path or click **Browse** to locate the program)
  - Perform an action (select an action from the list, or click **Define** to specify an action); any Smalltalk method that is available to your gOOi forms (for example, through parent classes) may be specified
8. Select whether the toolbar item should be represented as:
  - A button (enter the button label)
  - An icon (enter the icon file name, or click **Select** to locate a file)
9. Enter an item description (tooltip; optional, but recommended), or click **Default** to automatically insert the action (for example, send keystroke ENTER) specified in step 7.
10. Click **Add item**. The button is added to the Resulting Toolbar at the bottom of the window. Make the desired changes:
  - To remove an item from the Resulting toolbar, select the item and click **Remove item**.
  - To add space between toolbar items, click **Add gap** (see “Spacing toolbar items” on page 79).
11. Repeat steps 7–10 for each item that you want to add to the toolbar, then click **Save file**.
12. Click **Exit** to return to the Settings window.
13. Do one of the following:
  - Continue with other gOOi-Settings tasks.
  - Click **OK** to close the gOOi-Settings box and save the changes.
  - Click **Exit** to close without saving the changes.

## Spacing toolbar items

Clicking **Add gap** on the gOOi Toolbar definition window (see “[Creating a user toolbar](#)” on page 76) provides an interface for specifying how much space to insert between toolbar items, as shown in the following example:



Enter a value or accept the default (10) and click **OK**.

## Specifying headers and footers

AD/Advantage screens are composed of a header, body, and footer. By default, the header displays at row 1, the body at row 4, and the footer at row 21. You can modify these specifications.

The header/footer option may also be used for host screens other than AD/Advantage. If your host screens use a standard header and footer, modify the preset AD/Advantage specifications to fit your standards.

To specify header and/or footer settings:

1. Open the gOOi-Settings window, as shown in “[Specifying gOOi generation options and settings](#)” on page 45.
2. Click **Header/Footer**, then verify the following:
  - The row values for the header, body, and footer are preset and should be changed only if you have changed the No of rows in Header-Screen to a value other than 3 in the #PARM transaction of AD/Advantage.
  - The Header file and Footer file fields are preset to the location of the standard AD/Advantage header and footer. If you have not customized the AD/Advantage header or footer, verify that the specified locations are correct for your installation.

3. If you have customized the header and/or footer:
  - a. Export the customized ADV\_HEADER and/or ADV\_TRAILER from the Master user using the Universal Export Facility (UEF).
  - b. Download the UEF files to your PC.
  - c. Process these files through gOOi generation and specify their location in the Header file and Footer file fields. (For specific information, see “[UEF file splitting steps](#)” on page 119 and “[Specifying gOOi form components](#)” on page 127.)
4. Click  to return to the Settings window.
5. Do one of the following:
  - Continue with other gOOi-Settings tasks.
  - Click  to close the gOOi-Settings box and save the changes.
  - Click  to close without saving the changes.

## Pointing to prompter files

If your MANTIS application uses prompters, gOOi automatically generates them as scrollable list boxes containing the prompter text. When you generate a gOOi application, the text is generated on the PC and stored in files with a .pmt extension. gOOi looks for these files in whatever path you specify in the Prompter files path field.



---

**Warning:** gOOi treats the prompter files path as global data. (It becomes part of a single gooi.ini file.) Therefore, all MANTIS applications you intend to deploy to a workstation must have the same prompter path. If different paths are specified for two gOOi applications and they are both deployed to the same workstation, the first gooi.ini file will be overwritten. However, you can specify different prompter files paths for different workstations.

---

To specify the prompter files path:

1. Open the gOOi-Settings window, as shown in “[Specifying gOOi generation options and settings](#)” on page 45.
2. Enter a prompter files path, or click  to search for a file. You can specify a path that does not yet exist.
3. Do one of the following:
  - Continue with other gOOi-Settings tasks.
  - Click  to close the gOOi-Settings box and save the changes.
  - Click  to close without saving the changes.

## Using environment-dependent tools

Some of the following tools may be required before you can continue preparing to generate gOOi forms. Whether these tools are mandatory or optional will depend upon your environment.

### Converting BMS and MFS source files into UEF format

Before gOOi can generate forms from CICS or IMS, they must first be put into Universal Export Facility (UEF) format. From this common format, gOOi will generate forms.

If your host application is in MANTIS, go to “[Creating a template for function key mapping](#)” on page 91. If your host application is in IBM CICS Basic Mapping Support (BMS) format or IBM Message Formatting Service (MFS) format, convert it into Universal Export Facility (UEF).

To convert BMS files into UEF files, use the BMS Converter. Before doing so, review the following list of restrictions for host applications to determine whether you need to adjust your host application.

### gOOi restrictions for non-MANTIS host applications

The following restrictions apply to non-MANTIS host applications:

- ◆ Every host application screen and pop-up screen must have a unique identifier at the same specified location. This screen ID must have at least 2 characters (0–9, A–Z). The ID must begin with a letter and cannot have embedded or trailing blanks.
- ◆ gOOi does not support wrap-around capabilities for 24 x 80 domain screens. An example of such a screen would be one in which horizontal repeats cause a field to span multiple lines.
- ◆ Currently, gOOi does not support a host screen greater than 24 x 80.
- ◆ Two screen components cannot be placed side-by-side.

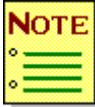


---

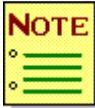
You do not have to define every screen to generate a gOOi application. If the gOOi HostMonitor encounters an undefined screen in your application, it presents the Just-In-Time display (if enabled) or the host screen. However, you must define all pop-ups. If an undefined pop-up is encountered, your application will become unsynchronized with the host.

---

## BMS/MFS conversion procedure



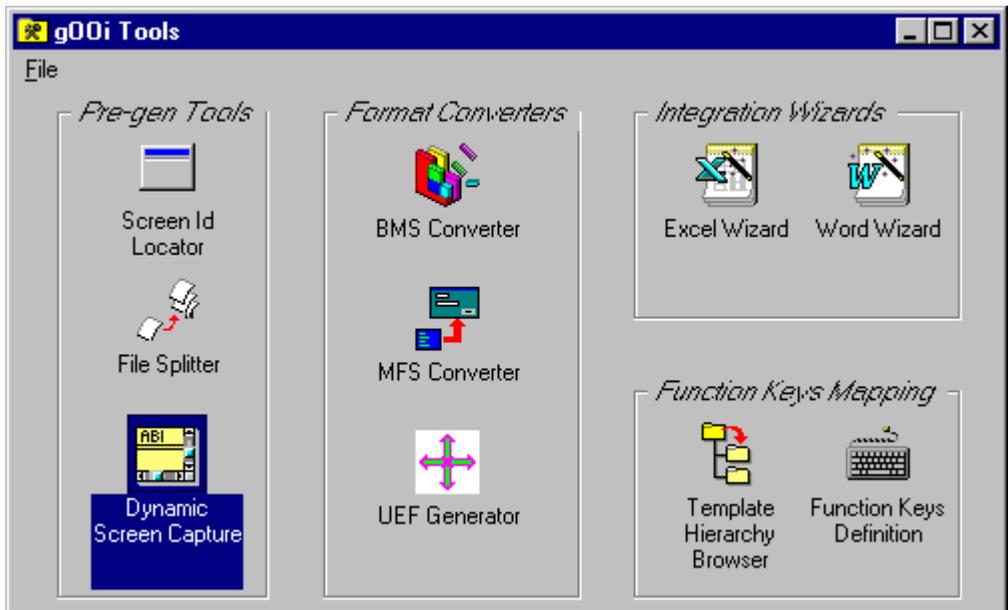
The most efficient input for the BMS or MFS Converter is a set of BMS/MFS source files containing one screen per file. You can convert BMS/MFS files containing multiple screens with an extra procedure.



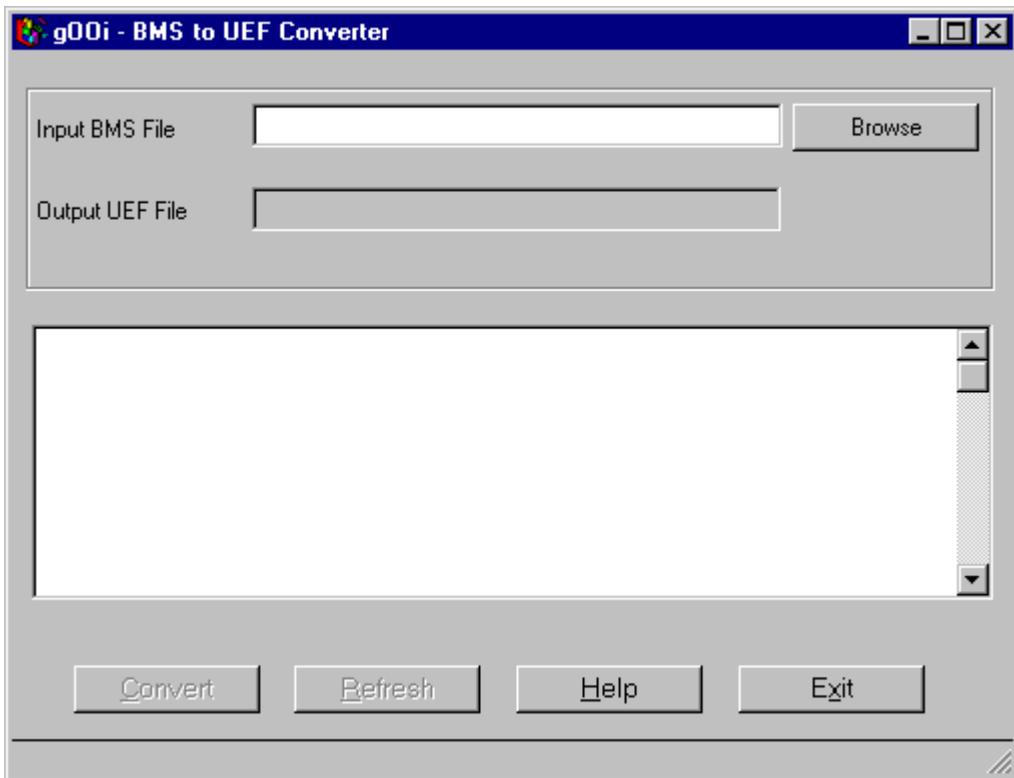
When you download your BMS/MFS source files to your PC, do not give them file extensions. The BMS or MFS Converter adds an .exp extension to each file during conversion and places these files in the same directory.

Follow these steps to convert BMS/MFS files:

1. At the gOOi Workplace window, double-click the Tools icon. The following gOOi Tools window displays:



2. Double-click either the BMS Converter or the MFS Converter icon.  
The Converter window displays:



3. Click **Browse** to specify the location of your BMS/MFS files. Select the first file in the directory and click **Open**. The path and file name display in the Input BMS/MFS File field.

4. Determine whether you can convert the files as a group or whether you must convert them individually (the Converter uses each existing file name to create a UEF file with an .exp extension):
  - If your BMS/MFS file names do not follow consistent naming conventions, you must convert each file individually. To do this, open each file in the Browse window so that the path and file name show up in the Input BMS/MFS File field. Click **Convert**. As you convert files, a list of completed conversions displays in the middle of the Converter window.
  - If your BMS/MFS file names are consistent, you can convert your files as a group. To do this, open the first file in the Browse window so that the path and file name show up in the Input BMS/MFS File field. Edit the file name in the path to the common file name characters and add an asterisk (\*) as a wild card. For example, if all your BMS/MFS file names begin with MAP (for example, MAP24, MAPEND, MAP4ID), edit the file name in the path to read "...MAP\*". Click **Convert**. All BMS/MFS files are converted in succession. A list of the completed conversions displays in the middle of the Converter window.
5. Determine whether you need to use the File Splitter:
  - If you know that there was only one screen per BMS/MFS source file, you are now ready to begin specifying your gOOi forms with your newly converted .exp files. Proceed to ["Specifying gOOi form components"](#) on page 127.
  - If you know that (or are unsure whether) there were multiple screens in any of your BMS/MFS source files, use the File Splitter on your newly converted .exp files (see ["UEF file splitting steps"](#) on page 119).

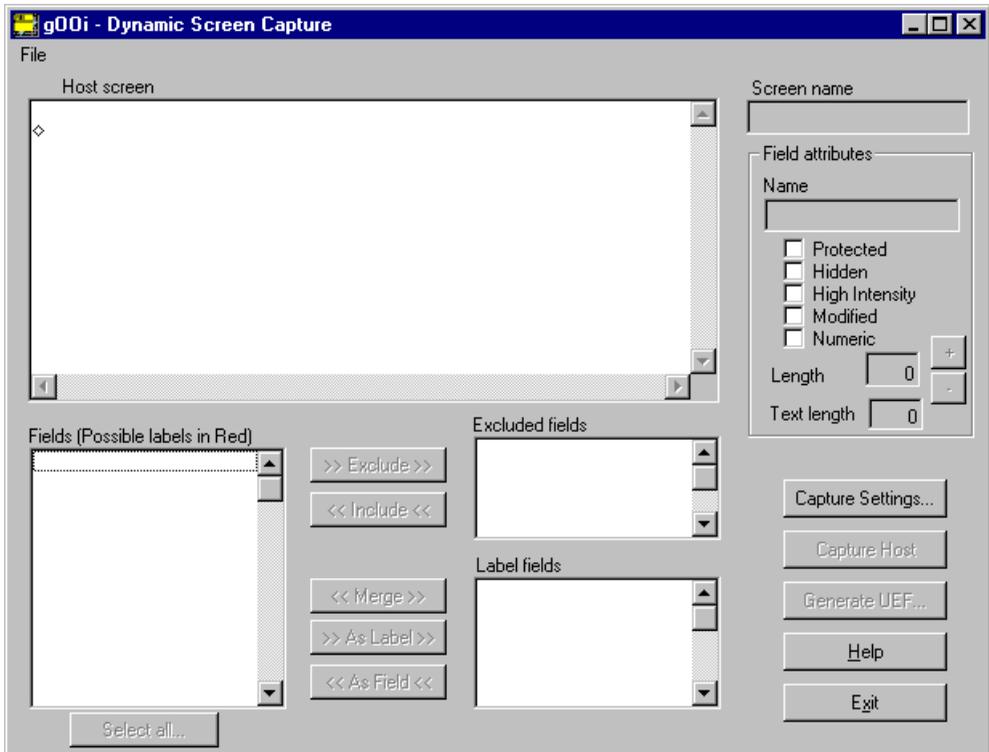
## Capturing IBM mainframe host screens

The Dynamic Screen Capture tool can be used to generate a gOOi form for any IBM mainframe host screen that has a screen ID. This tool creates a UEF file that the Application Generator can use.

The gOOi HostMonitor should be started before using this tool. The host screen that you capture needs a screen ID at the row and column defined in your current profile.

Follow these steps to open the Dynamic Screen Capture tool:

1. At the gOOi Workplace window, double-click the Tools icon. The gOOi Tools window displays.
2. In the gOOi Tools window, double-click the Dynamic Screen Capture icon. The following window displays:

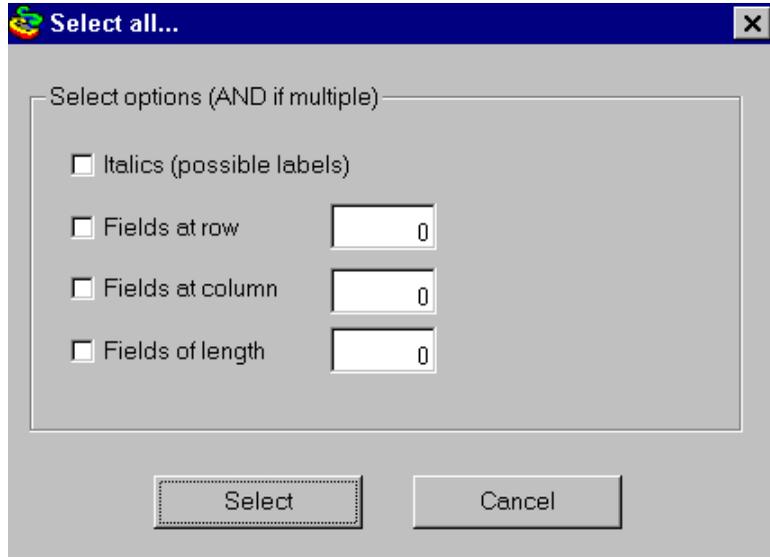


The following table describes controls that may be useful for some host screens:

Control	Description
Exclude	Specifies that certain fields are not to be included in the gOOi form.
Include	Reverses an excluded field.
Merge	Overrides the field extraction that occurs during Capture Host processing and allows you to select and combine two fields. For example, selecting Enter and Name and clicking <b>Merge</b> creates a single label field.
Field attributes	Shows the attributes detected for the currently selected field. You can override these attributes, if desired. For example, if you know that the host screen field only accepts numeric input, select numeric if this box is not selected.

3. Click **Capture Settings** and verify the emulator and session ID.
4. Click **Capture Host** to get a snapshot of the current host screen. The host screen image displays in the Host screen area. All fields extracted from the host screen image display in the Fields area. When you select a field, the corresponding area in the Host screen image is highlighted.
5. Select fields individually, or select multiple fields at once using one of the following procedures:
  - Select a label from the Fields list and click **As Label** to make it a label field. The Label fields area is initially empty because labels cannot be distinguished from protected fields in the 3270 host stream. However, fields that might be labels are listed in italics. Verify that the indicated length is correct before designating a field as a label.

- Click **Select** all to select multiple fields. The following window displays:



- Select the appropriate check boxes and click **Select**. For example, if you know that all the fields listed in italics for row 2 are labels, select italics and the fields at row 2—the italicized fields for row 2 are all highlighted.
6. Click **Generate UEF** to create an .exp file in UEF format. This file can then be input to the Application Generator to create a gOOi form corresponding to the host screen.



---

Before exiting the tool, you should save the captured host screen to a file by selecting File ⇒ Save. This will make it easier to update the generated gOOi form later. The captured host screen can be retrieved by selecting File ⇒ Open.

---

## Locating screen IDs

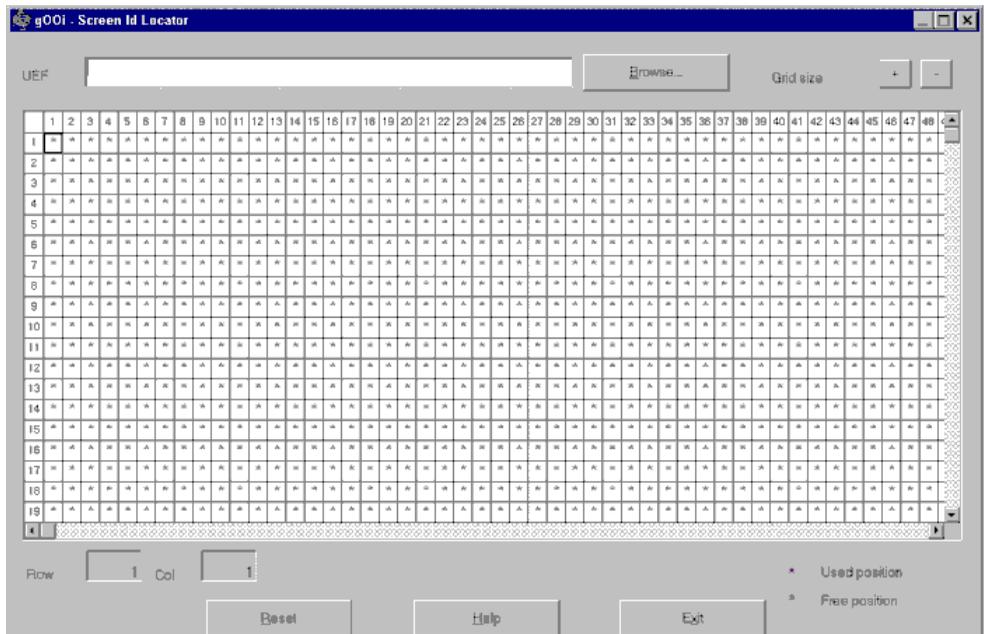
If you do not have screen IDs on your host screens, the Screen ID Locator can determine a suitable location to add these IDs.

The input for this tool is a UEF file containing the screen definitions for which screen IDs are needed. The Screen ID Locator parses the file and marks every field position. The result displays as a grid. Cells occupied by red asterisks represent screen field positions. Cells occupied by green asterisks represent screen positions where no field is defined. A set of consecutive green cells is a good candidate for a screen ID location.



AD/Advantage users do not need the Screen ID Locator when using gOOi with AD/Advantage. AD/Advantage screen IDs are predefined in gOOi.

At the Workplace window, double-click Tools, and then click **Screen ID Locator**. The following window displays:



Use the following procedure to parse a UEF file:

1. Click **Browse**.
2. Select the UEF file containing the screen definitions and click Open. The Screen ID Locator parses the file and displays the results in the grid.
3. To see a cell row and column number, select it. The coordinates are displayed at the bottom of the window. To magnify or reduce the view of the grid, click “+” or “-”.

When you identify the location where you intend to insert screen IDs in your host application, click **Exit** to return to the gOOi Workplace window. Specify your intended screen ID location on the Settings window.

---

## Creating a template for function key mapping

---



---

This feature does not apply to the Just-In-Time window, which always uses the Key assignments from Settings.

---

A template is a named group of properties that provides inherited behaviors for generated forms. Templates can include key mapping, visual items, and methods. This section addresses the most basic inherited behavior required by most applications: function key mapping.

To learn the procedures for providing inherited visual items and methods, see [“Creating an application template for visual items”](#) on page 103.

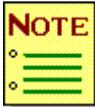


The default parent classes of GOOIGenericController and GOOIPopupController in Settings give function key mapping as defined in the Key assignments of Settings. If these defaults are used, each generated form and pop-up will have menu entries for PFkeys and MiscKeys as follows:

MiscKeys	PFkeys	PFkeys
CLEAR (Pause)		PF1 (F1)
ENTER (ENTER)		PF2 (F2)
HOME (HOME)		PF3 (F3)
PA1 (PageUp)		PF4 (F4)
PA2 (PageDown)		PF5 (F5)
PA3 (ShiftPause)		PF6 (F6)
		PF7 (F7)
		PF8 (F8)
		PF9 (F9)
		PF10 (F10)
		PF11 (F11)
		PF12 (F12)
		PF13 (ShiftF1)
		PF14 (ShiftF2)
		PF15 (ShiftF3)
		PF16 (ShiftF4)
		PF17 (ShiftF5)
		PF18 (ShiftF6)
		PF19 (ShiftF7)
		PF20 (ShiftF8)
		PF21 (ShiftF9)
		PF22 (ShiftF10)
		PF23 (ShiftF11)
		PF24 (ShiftF12)

Any changes you make to the key assignments will be reflected in the menu entries. These menu entries are only appropriate for IBM mainframe hosts. The sample template for your emulator or user-created parent classes should be used for OpenVMS/UNIX hosts. If the function key mapping built from key assignments is suitable for your environment, you can skip this section.

gOOi allows you to establish function key operation for your applications on a global scale, or by application. For example, the ESC key on the PC keyboard can equate to the IBM mainframe PA2 key in *all* your applications. Alternatively, some applications could use Esc to equate to PA2 while others could equate PGDN (Page Down) with PA2.



---

If you use the IBM3191Controller sample template for IBM mainframe hosts, your gOOi application automatically inherits methods for keys PA1–PA3, and PF1–PF24.

---

To create your own template for function key mapping, subclasses of GOOIGenericController and GOOIPopupController must be established to override the key assignments that are provided by these two classes.

The Template Hierarchy Browser makes it easy to create subclass templates of these general controllers and to define function key operation for those subclasses. The default sample templates delivered with gOOi were created in this manner. If the sample template is not exactly what you need, you can create a subclass and modify that.



---

We do not recommend modifying a Cincom-provided controller; you should create a subclass instead.

---

Because gOOi is object-oriented, you define key mapping once for all affected screen and pop-up interfaces. All newly-generated interfaces inherit from the template subclass.

If you later add a new function key to your application, it is unnecessary to create a new subclass template. You can go directly into the Function Keys Definition window from the gOOi Workplace and map the new key to your existing template. Key mapping performed through Function Keys Definition is dynamically inherited—it is unnecessary to regenerate your forms.

Every mapped function key displays in a pull-down menu on the generated form. To execute a function, the user can either select the menu item or press the defined key sequence. The gOOi developer must define the name of the menu as well as the name of the menu item. For example, you can specify that when F1 is pressed, the application sends an @1 command to the host (for IBM hosts). You can attach this sequence to the Show List item in the Action menu.

The Function Keys Definition tool also allows you to export and import these key mappings to and from text files.

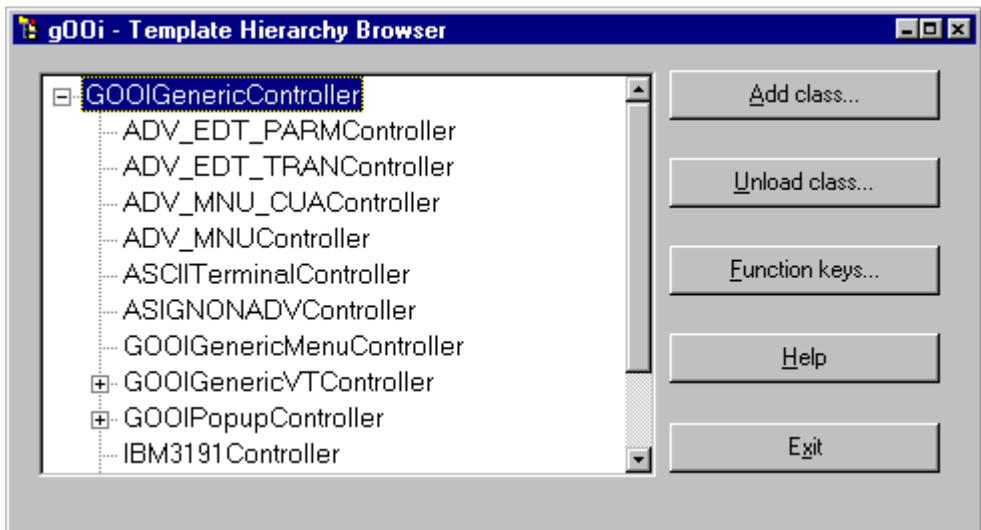


You can store a key mapping in a text file without attaching it to a particular class by using the Export and Import options of the File menu.

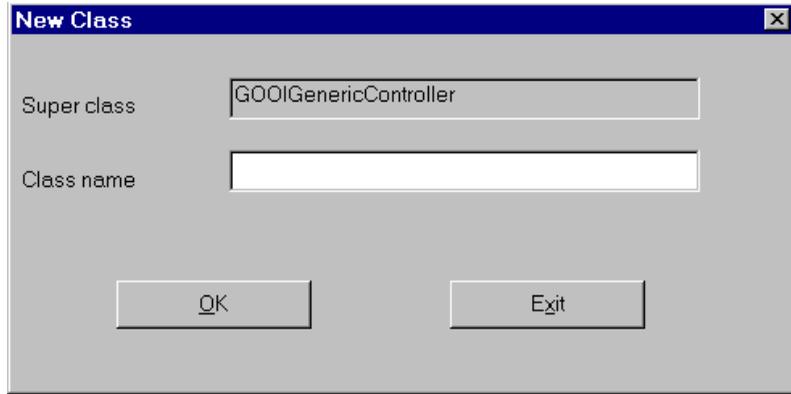
The following steps create a subclass of the class GOOIGenericController (for example, MyController). Later, you select your subclass as the parent class when you specify your form generation options. The same steps apply to creating a subclass of the GOOIPopupController or the IBM3191Controller. For information about modifying or copying a template subclass for function key maps, see “[Modifying a function key map file](#)” on page 100 and “[Copying a function key map file](#)” on page 101.

To create a template subclass for function key maps, perform the following steps:

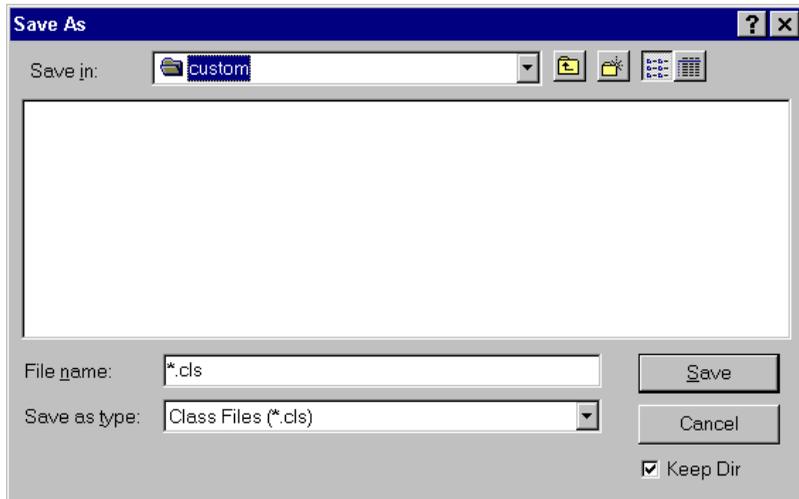
1. In the gOOi Workplace window, double-click the Template Hierarchy Browser icon to open the Browser window. If you click the small box to the left of the GOOIGenericController class name, the class tree expands. A further expansion reveals all the controllers delivered with the gOOi product. The following example displays the deliverables for an IBM user:



2. Select GOOIGenericController in the tree view box and click **Add class**. The following window displays:



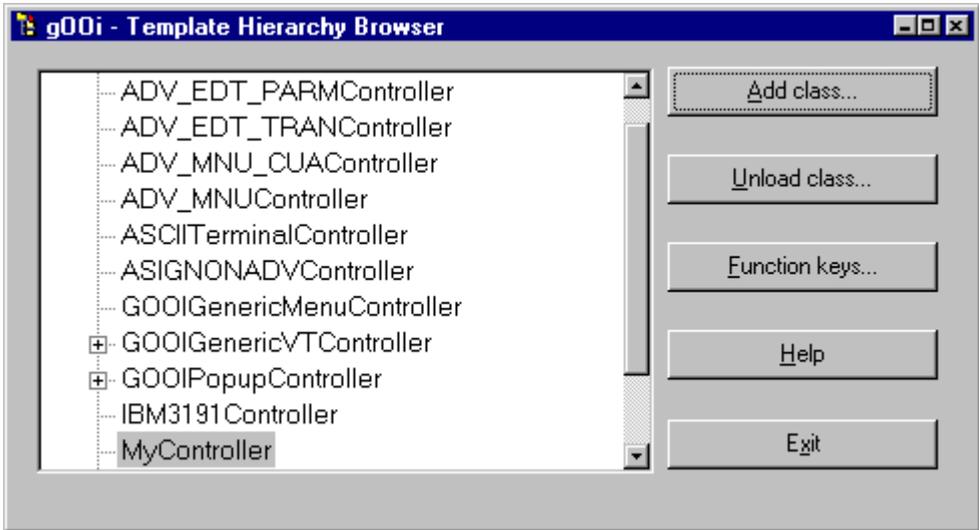
3. In the Class name field, enter the class name. A class name should begin with a capital letter. gOOi automatically appends "Controller" to your class name. Therefore, if you specify a class name of "My" gOOi creates a class called "MyController".
4. Click **OK**. The Save As window displays:



5. Save your controller in the \GOOI\CUSTOM subdirectory of ObjectStudio. The default installation path is:

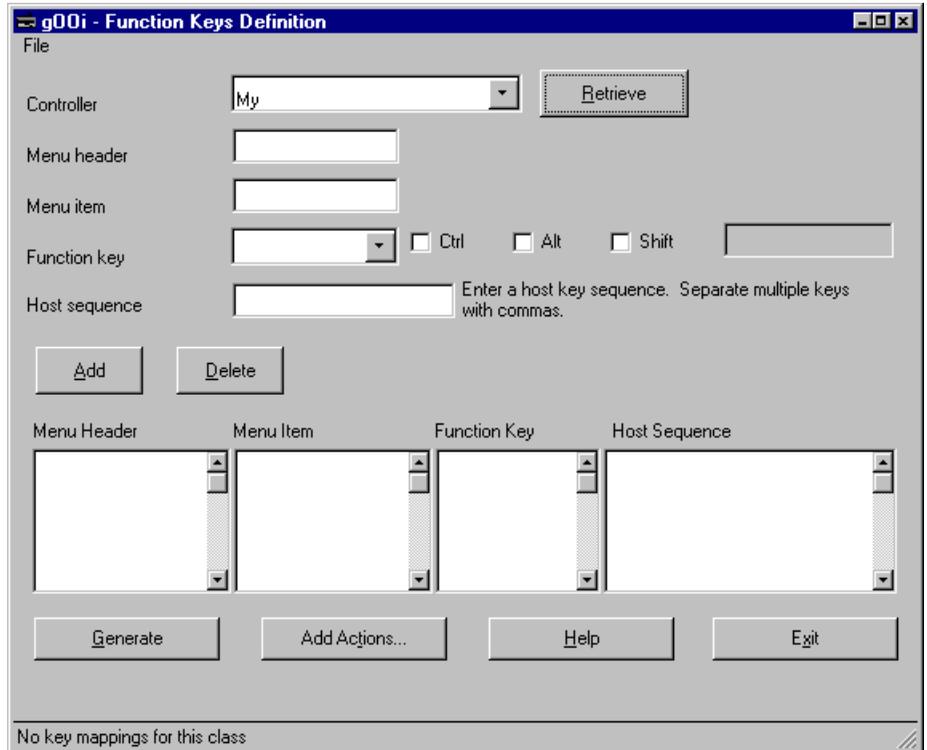
...PROGRAM FILES\OBJECTSTUDIO\GOOI\CUSTOM

Saving your file in the CUSTOM directory ensures that any controllers you create are not overwritten by gOOi product upgrades. Make sure the file type is .cls. Click **Save**. When you return to the Template Hierarchy Browser window, your newly-created class displays in the class tree as follows:

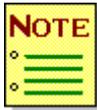


6. To create a pop-up controller subclass:
  - a. Select GOOIPopupController and click **Add class**.
  - b. Perform steps 3–5.
  - c. After completing steps 7–11 for your GOOIGenericController subclass, repeat steps 6–11 for your GOOIPopupController subclass.

7. Select your new class (for example, MyController) and click **Function keys**. The Function Keys Definition window displays. Your controller name is in the Controller field (minus the Controller suffix):



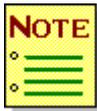
8. Perform the following steps for every function key you want to map (an example window follows):
  - a. Enter a name in the Menu header field. The menu header is the name that displays on the menu bar (for example, Record).
  - b. Enter a name in the Menu item field. A menu item is a command that displays in a menu (for example, PAGE UP).
  - c. Select a key in the Function key drop-down list (for example, PAGE UP). This is the key that will execute the command. To define a keystroke combination (for example, CTRL+F3), click the combination key box(es) and select or enter a character in the leftmost field. Your combination displays in the rightmost field.
  - d. In the Host sequence field, enter the command to send to the host when the selected function key or key combination is pressed.



---

**IBM mainframe:** A default IBM3191Controller and a default IBM3191PopupController are provided. The key mapping for these controllers reflects the “[Host-PC translation tables](#)” on page 311. This table is also provided in Help. Default ASCII controllers are also provided. See “[Host-PC translation tables](#)” on page 311 for their key mapping.

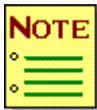
---



---

**OpenVMS/UNIX gOOi TCP/IP:** The values for the Host sequence field are specified as the sequence of individual keys pressed, separated by commas. For example, specify PF10 as F2,1,0, which represents pressing the F2 key, then the 1 key, then the 0 key. Retrieve the gOOiVTTerminal controller to see examples of Host sequences.

---



---

**Reflection for Digital/UNIX emulator:** The values for the Host sequence field can be found in the rwinapi.txt file that is within the emulator’s directory structure. Retrieve the Reflection2Terminal controller to see examples of using these values in Host sequences.

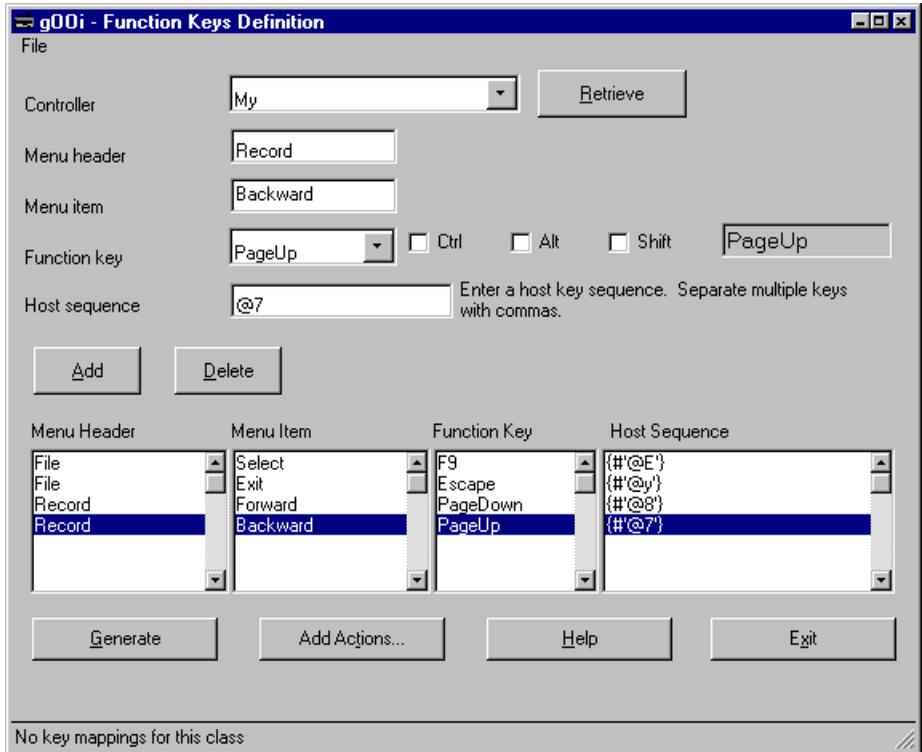
---



**KEA! emulator:** The values for the Host sequence field can be found using the KEA! SmartPad definition window (in KEA!, select Options ⇒ SmartPad). When the KEA! keyboard is visible, selecting a key sequence using your mouse fills the Value field. Retrieve the ASCIITerminal controller to see examples of using these values in Host sequences.

- e. Click **Add**. The values that you entered display in the list at the bottom of the window.

Repeat steps 8a through 8e for each function key you want to map. The following window displays a key mapping in progress for the example MyController template:



9. When you have defined all the key mappings for your controller, click **Generate**. When prompted to confirm the update, click **Yes**. Your controller is saved to the directory you specified in step 5 and you return to the Function Keys Definition window. See “[Editing your loadable application](#)” on page 102 for information about loading a new controller with the loadable application.
10. Exit the Function Keys Definition window. You return to the Template Hierarchy Browser window.
11. At the Browser window, click **Exit**. You return to the gOOi Workplace. You are now ready to create your .exp files with the File Splitter (for MANTIS applications) or to convert your host screens files into UEF format (for non-MANTIS applications).

## Modifying a function key map file

You can modify a controller that already has key mappings.



---

You can store a key mapping in a text file without attaching it to a particular class by using the Export and Import options of the File menu.

---

The instructions in this section assume that the Function Keys Definition window is displayed (see step 9 in “[Creating a template for function key mapping](#)” on page 91).

To modify a key map, follow these steps:

1. In the Controller field, select the name of the controller you want to modify and click **Retrieve**. The function key mappings are listed at the bottom of the window.
2. Double-click the key map record you want to change. The listed values display in their appropriate fields in the upper part of the window.
3. Delete the key map record from the list, modify the values in the fields, and click **Add**.

To remove a mapping, select the mapping from the list and click **Delete**. The file is updated according to your changes.

## Copying a function key map file

You can copy function key definitions from one controller to another. The instructions in this section assume that the Function Keys Definition window is displayed (see step 9 in “[Creating a template for function key mapping](#)” on page 91).

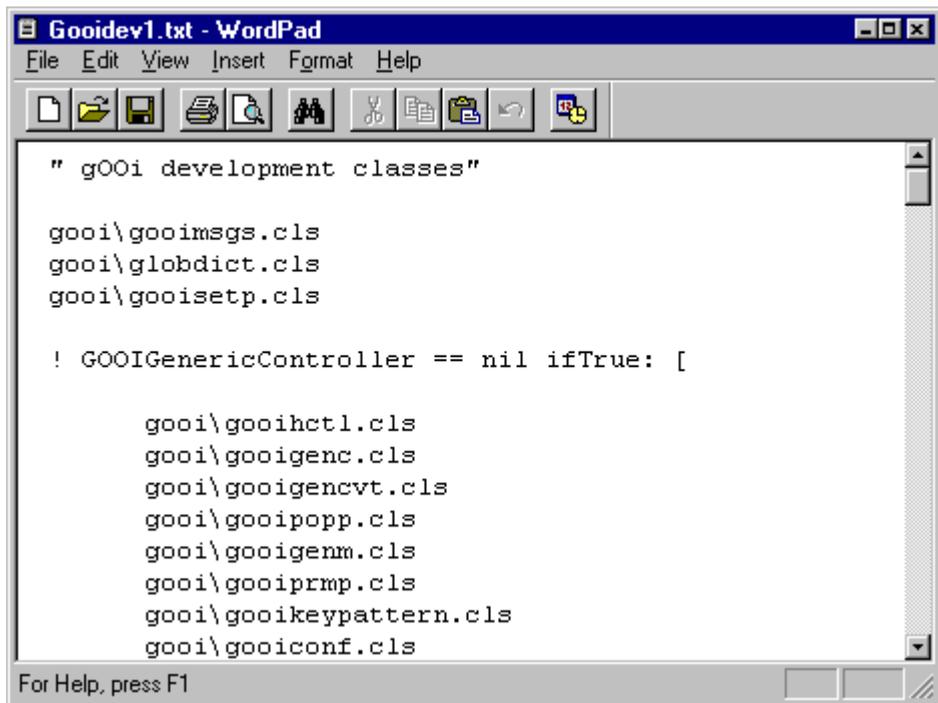
To copy key definitions, follow these steps:

1. Select the source controller in the drop-down list and click **Retrieve**. The function key mappings for that controller display at the bottom of the window.
2. In the **Controller** field, select the name of the target controller and click **Generate**. The key mappings are copied to your target controller.

## Editing your loadable application

After creating a new controller (for example, creating a template subclass), when you load gOOi, your new controller displays in the available controller hierarchy. On a subsequent loading of gOOi, you have to explicitly load the controller (using Load file) or edit your loadable application.

The gOOi Developer application is contained in the `gooidev1.txt` file, and the gOOi Runtime application is listed in the `gooirun1.txt` file. These files are located in the directory where ObjectStudio is installed. The following example shows the contents of `gooidev1.txt`:



```

" gOOi development classes"

gooi\gooimsgs.cls
gooi\globdict.cls
gooi\gooisetsp.cls

! GOOIGenericController == nil ifTrue: [

    gooi\gooihctl.cls
    gooi\gooigenc.cls
    gooi\gooigencvt.cls
    gooi\gooipopp.cls
    gooi\gooigenm.cls
    gooi\gooiprmp.cls
    gooi\gooikeypattern.cls
    gooi\gooiconf.cls
  
```

For Help, press F1

To add your new controller class to this list of classes, insert a line that specifies the proper name and location. For example, if you have put the new controller class into the custom directory of ObjectStudio under the name of `funckeys.cls`, you would add this line to the file list:

```
custom\funckeys.cls
```

Be sure to add this line after `'gooi\gooigenc.cls'`, which is the file for superclass `GOOIGenericController` and must precede your new file.

## Creating an application template for visual items

A gOOi application often parallels a host application. For example, if you have a Credit application on the host made up of 40 screens, you would likely define all 40 of these screens to gOOi under an application name of Credit. You may want all of the generated gOOi forms for Credit to have common visual items for consistency in appearance and user interaction. gOOi makes it easier to achieve this consistency by using an application template.

Visual items can be assigned to an application template that is used by the gOOi Application Generator. The template is an interface created via the ObjectStudio Designer that contains visual items such as buttons. When the generator is run, all gOOi forms created during generation will inherit the visual items from the template.

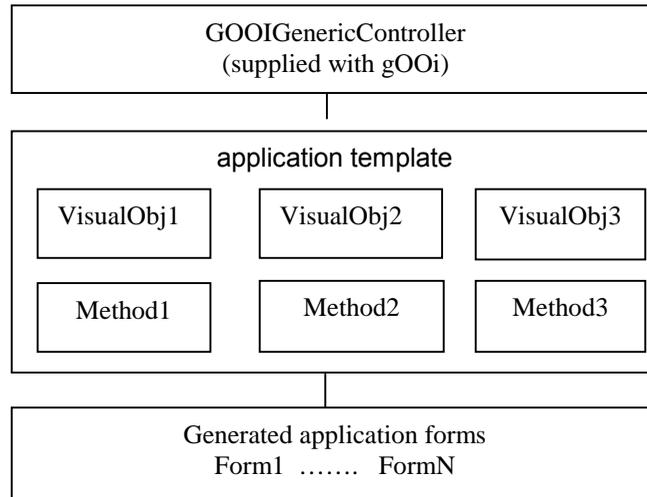
For example, you want OK and Cancel buttons on all the gOOi forms in an application. First, using the ObjectStudio Designer, you add these push buttons into an ObjectStudio interface. Next, you specify this interface in 'Settings ⇒ Form Options ⇒ Parent class name'. Finally, run the Application Generator and it will use this interface as a template for visual items. All gOOi forms generated with this template will include OK and Cancel buttons. Specific actions can be assigned to these buttons after the forms are generated.

## Application template inheritance

Application template inheritance operates under the following conditions:

- ◆ Because gOOi is an object oriented system, gOOi forms inherit behaviors from superclasses in the class hierarchy. The application template is the next level higher superclass of all the gOOi forms in a gOOi application. gOOi forms also inherit behaviors, but not visual objects, from the superclasses that are above the application template class in the class hierarchy.
- ◆ gOOi looks for visual objects only in the parent class (or pop-up parent class) specified in Settings ⇒ Form Options.
- ◆ Your specified parent class becomes the *superclass* of the gOOi forms that you generate for your gOOi application.
- ◆ Key mapping and methods (including methods associated with visual objects) exhibit dynamic inheritance. In other words, if you change key mapping or method properties, the new properties are immediately and automatically inherited by their subclasses. No regeneration of the gOOi application is required.
- ◆ Visual objects exhibit static inheritance. In other words, if you add visual objects to a template, they are only copied into gOOi forms during gOOi application generation.
- ◆ The application template has a specific name requirement. The name must be the name of the gOOi application. This name is case sensitive, so gOOi will not recognize a template named 'CREDIT' during generation of an application named 'Credit'.
- ◆ If you do not create an application template prior to application generation, the gOOi Application Generator will create one according to the name requirement described above. This template is created so that it is easy to add behaviors (methods) in a single place that immediately become available via inheritance to all the gOOi forms in the application. The template produced by the generator is a stub class that contains only a few necessary methods which do not affect the gOOi forms.

The following illustration shows an example class hierarchy. The application template class provides both static and dynamic inherited behaviors to a gOOi application:



In this case, the application template has visual objects and is specified as the parent class. The visual objects from the template are copied to the gOOi application forms during generation. The methods are common functionality that was added after generation.

If changes are made to the template methods and visual objects after form generation, the gOOi application dynamically inherits the method changes. Changes to template visual objects will not take effect unless the forms are regenerated.

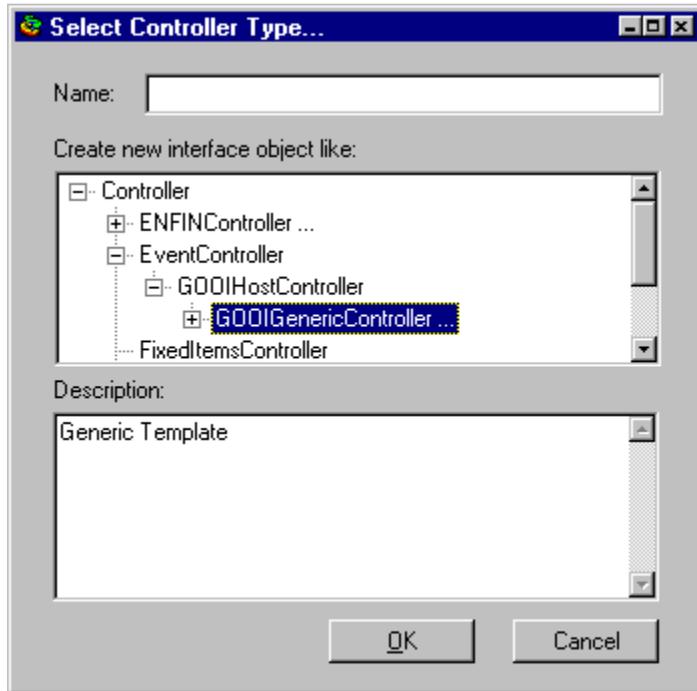
## Steps for creating an application template for visual items

Creating an application template for assigning visual items to a gOOi application involves the following tasks:

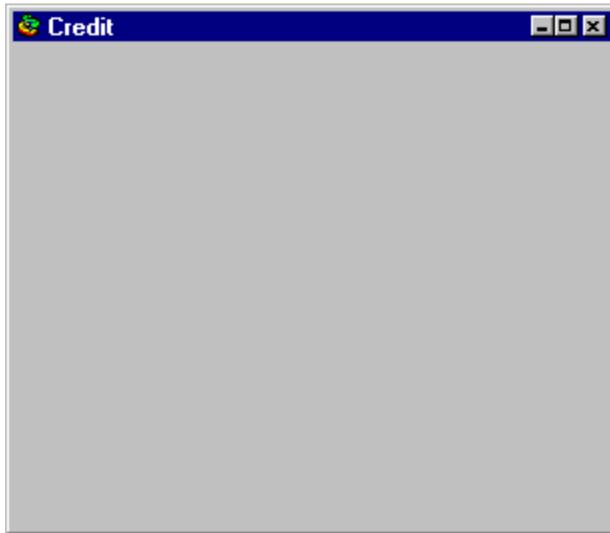
1. Creating an interface (controller)
2. Adding visual items to the interface
3. Making your interface the parent class for generation
4. Generating and testing application forms

Details for each of these steps follow:

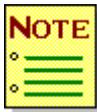
1. From the ObjectStudio desktop, run the Designer by clicking the 'Create new Interface' icon, or by selecting File ⇒ New ⇒ Interface from the menu. The 'Select Controller Type...' window displays. Expand the hierarchy under EventController until the interface with your key mapping appears in the window, then click on this interface to select it. If you are using the gOOi direct TCP/IP connection, the default key mapping is in GOOIGenericController (IBM) or GOOIGenericVTController (OpenVMS/UNIX). In the below example, GOOIGenericController has been selected:



Enter your gOOi Application name into the Name field. Name is a case sensitive field, so 'CREDIT' is not equivalent to 'Credit'. Click **OK**, and the following empty ObjectStudio Designer interface window displays:



2. Add one or more visual items (for example, OK and Cancel buttons) to the controller. Appropriate actions can later be assigned to the item(s) following form generation. Create a button as follows:
  - a. In the ObjectStudio Designer, select Formitem ⇒ New Item from the menu. You see the New Item window.
  - b. Enter a button name in the Name field. Click the Button icon. When you click **OK**, the button displays on the interface.



---

When you click an icon, the type description (for example, Button) displays near the top of the window. If you click in the empty space between the icons with mouse button 2, then click **List**, the icons and their associated descriptions are presented in alphabetic order.

---

- c. Select File ⇒ Save. Save the interface as a .cls file using the Application name. For example, save a template for the Credit application as Credit.cls.
- d. Select File ⇒ Exit. Your new controller displays as an icon in the ObjectStudio Work Area.



---

The ObjectStudio Designer automatically puts your new interface in the Work Area. However, if you exit ObjectStudio and return or make changes to your controller, you must explicitly load the controller file by clicking mouse button 2 in the Work Area and selecting 'Load file...'. After generation, this template interface class is included in the list of gOOi forms that make up an application.

---

3. Make your new controller the parent class for generation as follows:
  - a. With your new controller in the ObjectStudio Work Area, double-click the gOOi Developer icon. The gOOi Developer Workplace window displays.
  - b. Double-click the Settings icon, and the Settings window displays.
  - c. Click **Form Options**, and the Form Options window displays.
  - d. Select the name of your new template interface from the Parent class name drop down list. The name will be suffixed with 'Controller', as in 'CreditController'. This action directs the Application Generator to use your new controller as the template for generating gOOi forms. Click **OK** to return to the Settings window.
  - e. Click **OK** to return to the gOOi Developer Workplace.

4. Test and generate the application forms as follows:
  - a. Double-click the gOOi Application Generator icon. The Application Generator window displays.
  - b. Specify an Application Name that matches the name used for the template class. The name is case sensitive.
  - c. Click **Browse** to locate and open an existing .exp file to test your new template.
  - d. With your .exp file name in the File name field, click **Add item to form** to assign the file to Form 001 in the lower part of the window.
  - e. Click **Preview** to display the gOOi form. Check the form to verify that the visual items from the template are displayed. It is likely that the visual template items will not be displayed at the location where you want them to be. Item location is easy to change after generation. It is also possible that you may not see the template visual items that were copied into the gOOi form. This situation can occur when the position of the template items conflicts with items generated from the host screen file. In this case, try going back into the Designer for the template and moving its visual items to other locations, then rerun Preview. In the event that problem continues, please check that the template is correctly specified in Settings as the 'Parent class name'.
  - f. Once you have verified via Preview that the visual template is working, you can proceed to build the application layout and generate your gOOi application. Please see ["Generating gOOi forms"](#) on page 125 for details on this process.

With your new template class specified as the parent class, all gOOi forms in the application will be generated with the visual items from the template. You can then assign method to these items. For example, it is easy to specify that the ENTER key is sent to the host when the OK button is clicked. For information about creating an event-driven button, see ["Additional customizations"](#) on page 190 and ["Example of an event-driven customization"](#) on page 203.



# 4

---

## Downloading screens from MANTIS

---

---

### Methods for transferring UEF screen images

Before you use the Application Generator to create gOOi forms, you must save the desired MANTIS screen images in UEF format on your desktop (that is, the PC or workstation). There are two methods for transferring UEF screen images from the host to the PC: via the Host Monitor or using FTP.

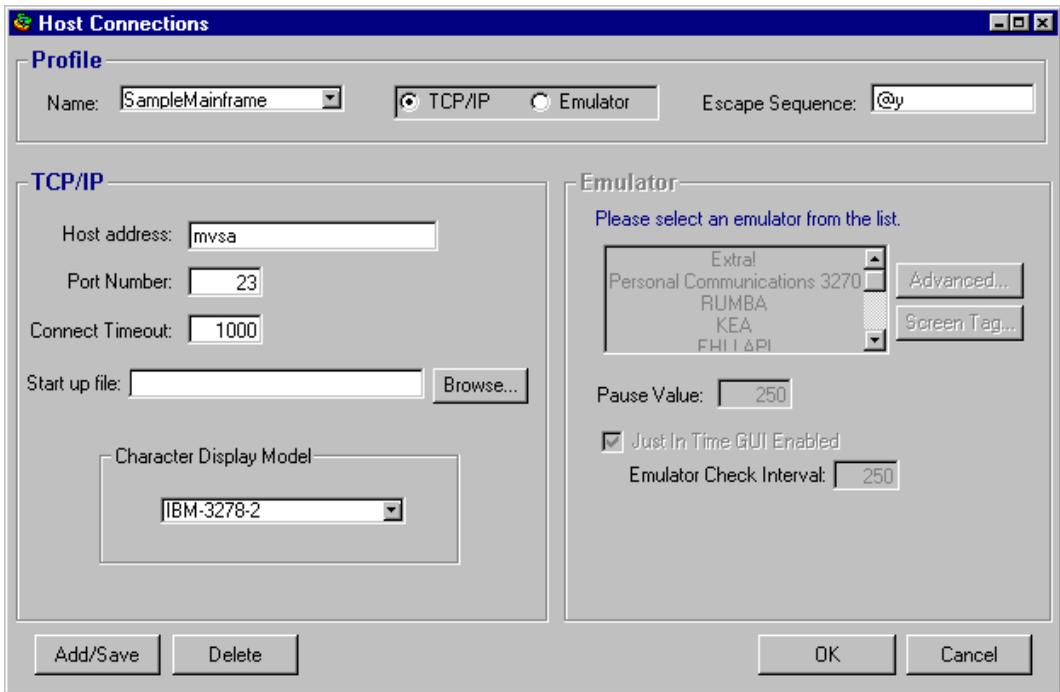
## Using the Host Monitor



The gOOi forms shown in this section are from the IBM mainframe, and may differ slightly in appearance for OpenVMS/UNIX.

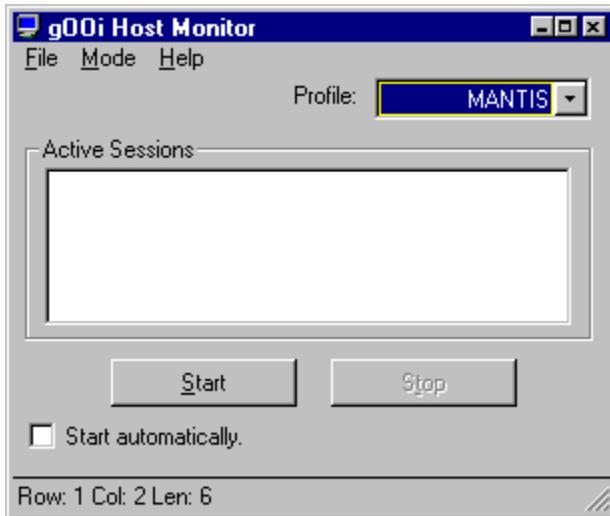
To obtain UEF images of screens using the Host Monitor, you must first specify how the Host Monitor is to connect to the host, either through a direct gOOi TCP/IP connection, or via an emulator. To obtain UEF images of screens using the Host Monitor:

1. Start the Host Monitor, then choose File ⇒ Host Connection from the menu. The following window displays:

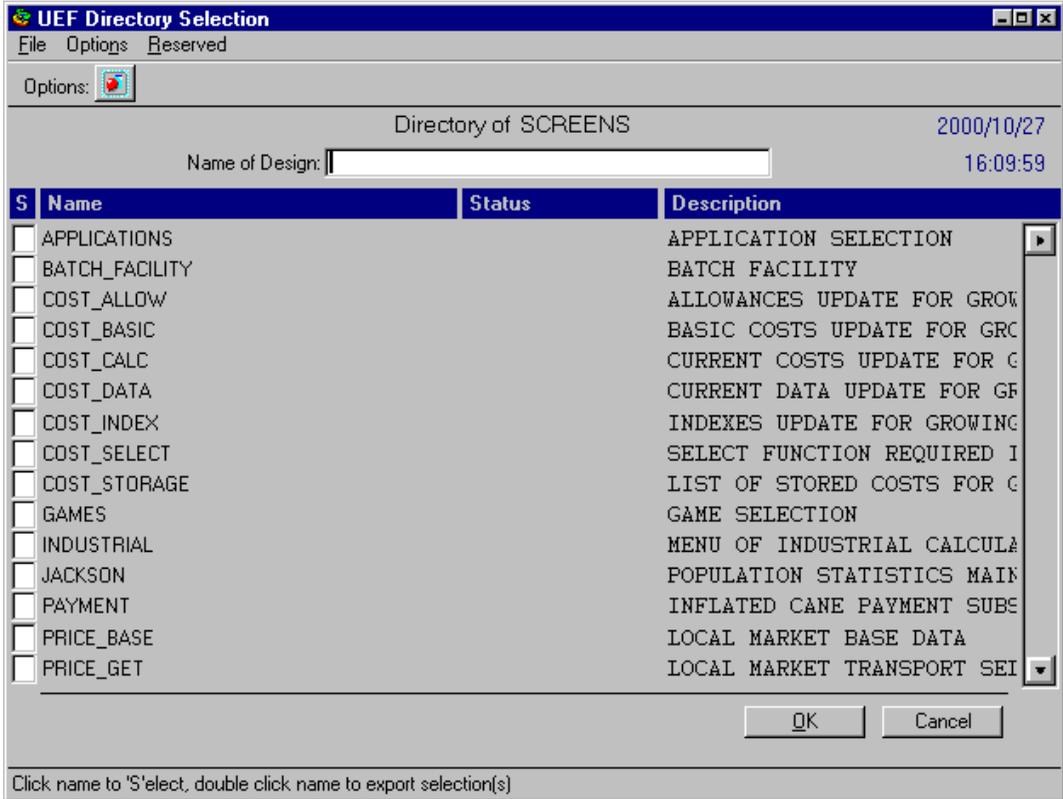


2. Specify the connection parameters for your environment, then click **OK**.

- Click **Start** on the following window to begin a host session:

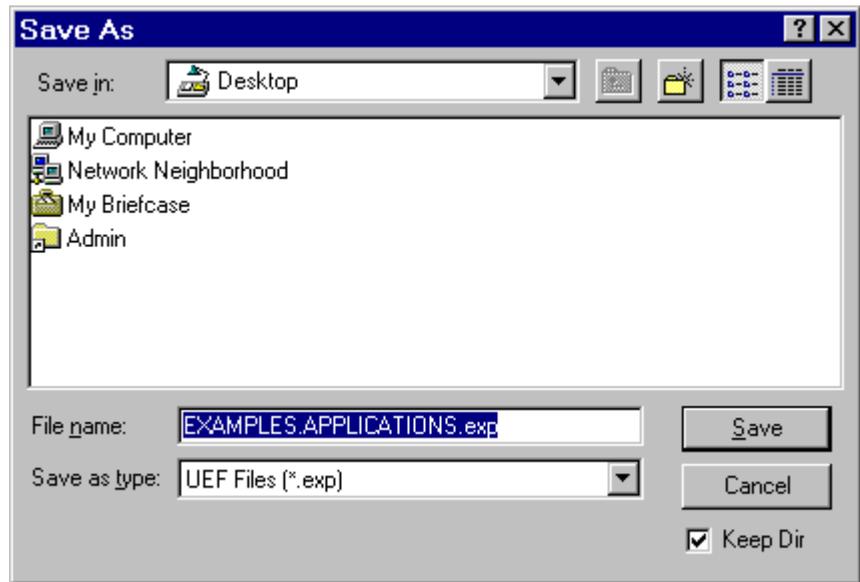


- Use this session to access the MANTIS Universal Export Facility (UEF). At the UEF menu, indicate that you want to export (that is, choose a Direction of EXP), set Directory to Y, and select SCREEN as the entity type. The following directory of screens then displays on a customized gOOi form:



5. Select the screen(s) for which you want a UEF image. Then set the Apply to desktop option from either the menu bar or the toolbar. Press ENTER to save the screens to your desktop.

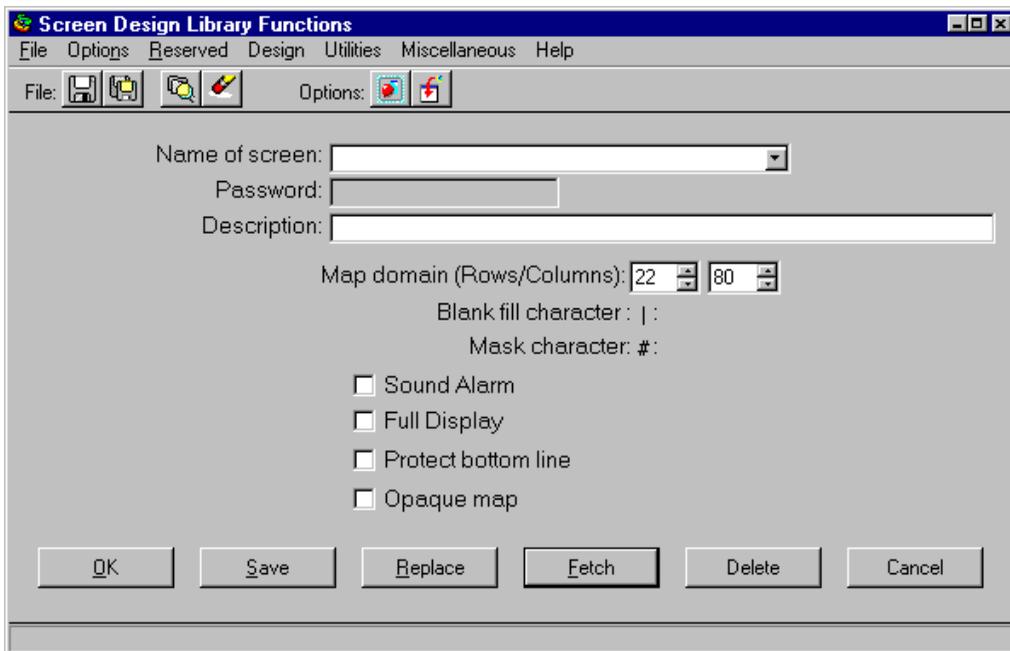
A Save As window, such as the following, displays for each screen selected:



6. Respond to each dialog with the name of the PC file where you want to save the UEF image. This approach allows you to select multiple screens at a time.

You can also download a UEF image to the PC from the MANTIS Screen Design Facility.

To do so, use the Screen Design Library Functions to fetch the screen for which you want a UEF image.



Choose Library Functions again, set the Apply to desktop option from either the menu bar or the toolbar, then select Replace.

A Save As dialog displays. Specify the name and location for a file, and a UEF image will be stored there.

---

## Using FTP

You can also transfer UEF information from the host to the PC via the FTP tool of your choice. To do so:

From the MANTIS Universal Export Facility, export all the screens for which gOOi forms are to be generated. The standard export process creates the UEF images on a VSAM ESDS cluster on an IBM host, or on a sequential file on OpenVMS/UNIX.

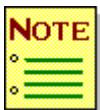
If you use an IBM host, run a REPRO job to copy the contents of this ESDS cluster to a sequential data set. (This sequential data set must be defined as variable, with a record length of 258.) No equivalent to REPRO is necessary for an OpenVMS/UNIX host since the export places the data in a sequential file. Use your FTP software to copy this sequential data set to the PC.



---

If you are an IBM mainframe MANTIS user, please see “[IBM mainframe considerations](#)” on page 293 for details concerning UEF processing.

---



---

Before downloading a single UEF file with multiple screens/prompters from the host, you should give it a .txt extension to distinguish it from the .exp files that will be generated from it. If your UEF file contains a single screen/prompter, give it a .exp extension before downloading.

---

## Creating .exp files with the File Splitter

After specifying host screen settings and gOOi form generation options, completing any pregeneration tasks appropriate for your environment, and creating any key map or visual item templates that may be required, you can do one of the following:

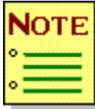
- ◆ Split your UEF file into .exp files for use by gOOi. Before doing so, survey the list of restrictions that apply to MANTIS applications to determine whether you need to make any adjustments to your MANTIS application (see “[gOOi restrictions for MANTIS applications](#)” on page 118).
- ◆ Split the .exp files that you converted from BMS or MFS source files in the previous section. Skip to the numbered steps in “[UEF file splitting steps](#)” on page 119.

### gOOi restrictions for MANTIS applications

The following restrictions currently apply to MANTIS applications:

- ◆ It is preferred, but not required, that every MANTIS application screen and pop-up screen must have a unique identifier at one of the specified screen ID profile locations. This screen ID must have at least 2 characters (0–9, A–Z). The ID must begin with a letter and cannot have embedded or trailing blanks. Using Dynamic CONVERSE to display host screen IDs at a position other than the one at which they were designed is not recommended. For more information, see “[Rules for screen IDs and MANTIS prompter IDs](#)” on page 307.
- ◆ Every MANTIS application prompter must have a unique identifier. This identifier must be at least two alphanumeric characters in length, begin with a letter, and be in parentheses at the end of the prompter description. For more information, see “[Prompter ID rules](#)” on page 310.
- ◆ gOOi does not support wrap-around capabilities for 22 x 80 domain screens. An example of such a screen would be one in which horizontal repeats cause a field to span multiple lines of the screen.
- ◆ Currently, gOOi does not support a MANTIS screen larger than 24 x 80.
- ◆ Two screen components cannot be placed side-by-side.

- ◆ gOOi handles vertical repeats of 255 as ending on line 22, rather than dynamically adjusting them according to the number of rows on the display.

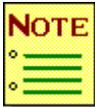


---

You do not have to define every screen to generate a gOOi application. If the gOOi HostMonitor encounters an undefined screen in your MANTIS application, it presents either the Just-In-Time window (if enabled) or the emulator host screen. However, you must define all pop-ups for generated gOOi forms. If an undefined pop-up is encountered while gOOi is processing a generated form, your application will become unsynchronized with the host.

---

## UEF file splitting steps



---

**MANTIS users:** It is recommended that you generate one UEF file from your MANTIS application, download it to the developer's PC, and run it through the File Splitter. Even if you have generated multiple UEF files, use the File Splitter. The File Splitter ensures that there is only one MANTIS screen or prompter *object* per file. If you are unsure whether you have one object per file, use the File Splitter on each file.

---

---

For gOOi to generate forms, there must be only one MANTIS screen or prompter definition per file on the PC. The File Splitter looks at your UEF file and generates one .exp file per MANTIS object. These files are placed in the same directory as your original UEF file.

---

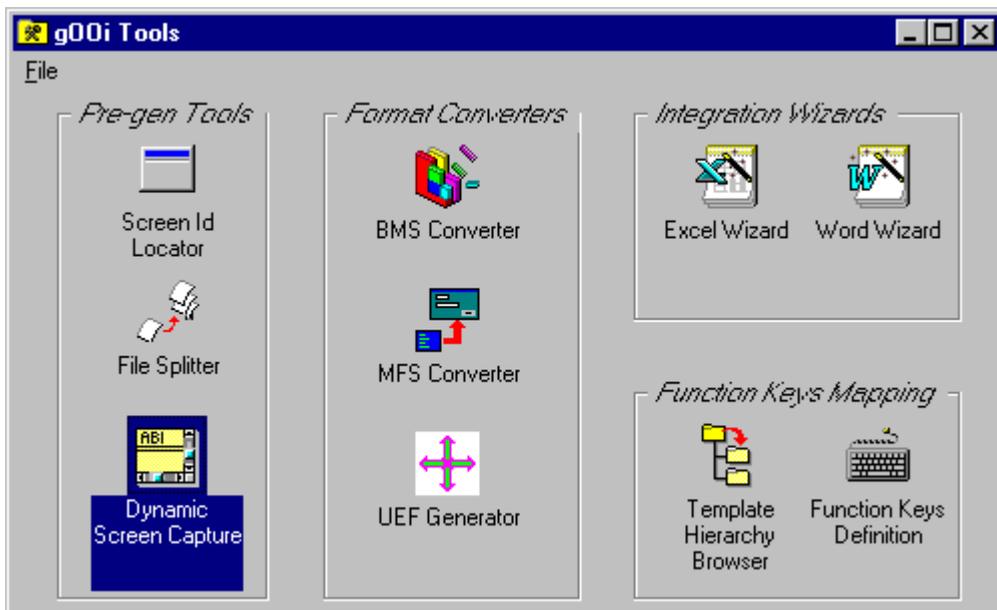
---

Before downloading your UEF file with multiple screens/prompters from the host, it is recommended that you give it a .txt extension to distinguish it from the .exp files to be generated from it. If your UEF file contains a single screen/prompter, give it a .exp extension before downloading.

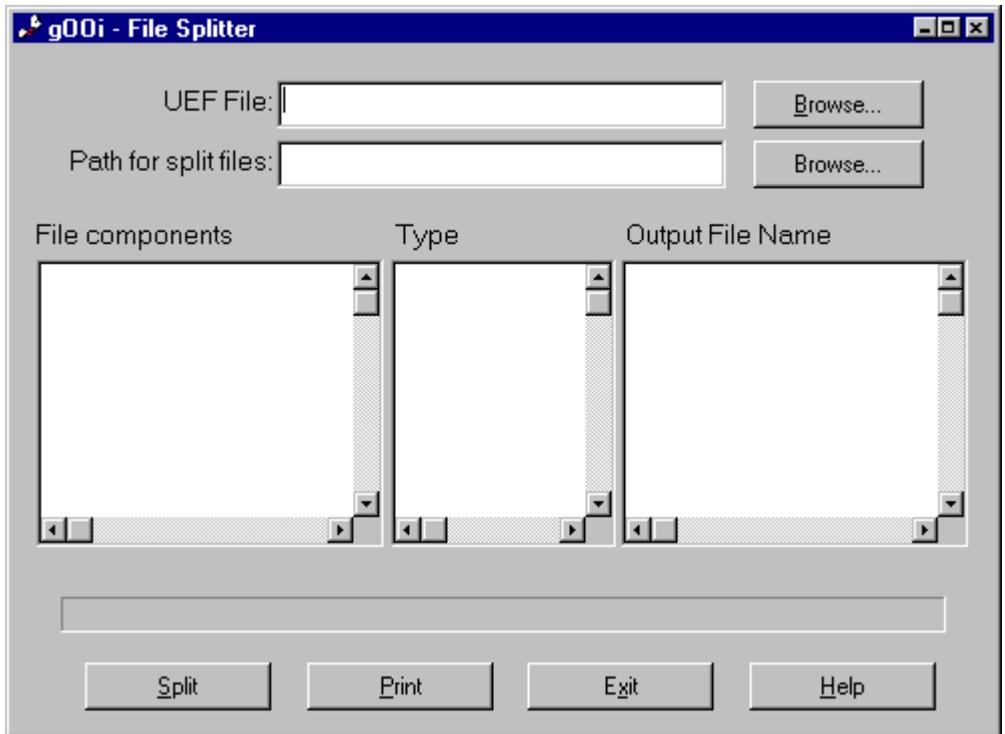
---

To split a MANTIS UEF file or an .exp file converted from BMS format, perform the following steps:

1. On the ObjectStudio Workplace Desktop, double-click the gOOi Developer icon. The following two windows are displayed:



2. Double-click the File Splitter icon on the gOOi Tools window. The following window displays:



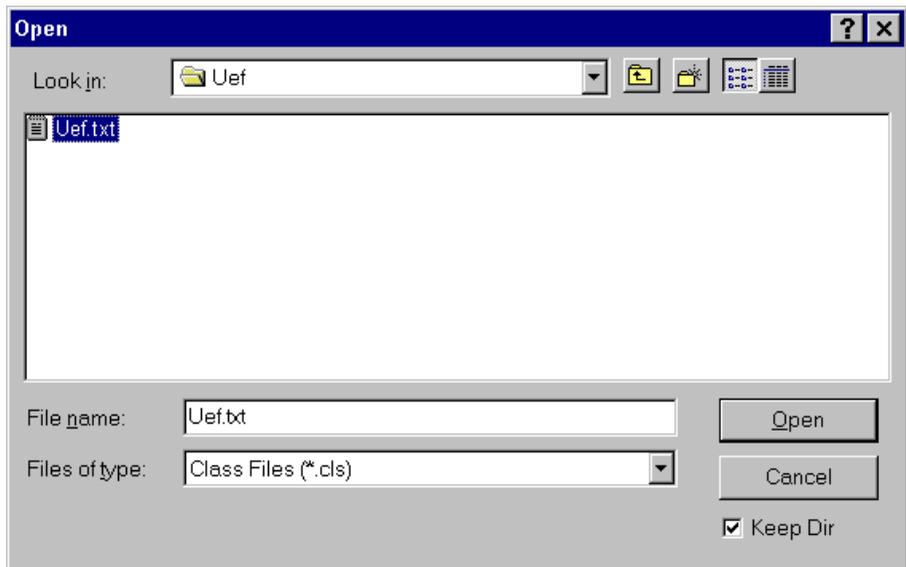
3. Click **Browse** to locate a downloaded UEF extract file (this example assumes one UEF file) and to choose a location for the .exp files that are output from the split.



**BMS users:** If you are splitting .exp files converted from BMS source files, perform the following steps as if each converted .exp file is the UEF file under discussion. In other words, perform all steps for each multiple-screen .exp file that was output from the BMS Converter. As soon as you bring a converted .exp file into the File Splitter, you can see whether it contains multiple screens. (The File Splitter does not split a file with only one screen.) The File Splitter splits multiple-screen files into new .exp files and places them back in the same directory. Later, you must be able to distinguish between single-screen and multiple-screen .exp files to use the gOOi Application Generator. The Print function is helpful for later reference on how the file was split.

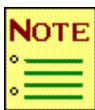
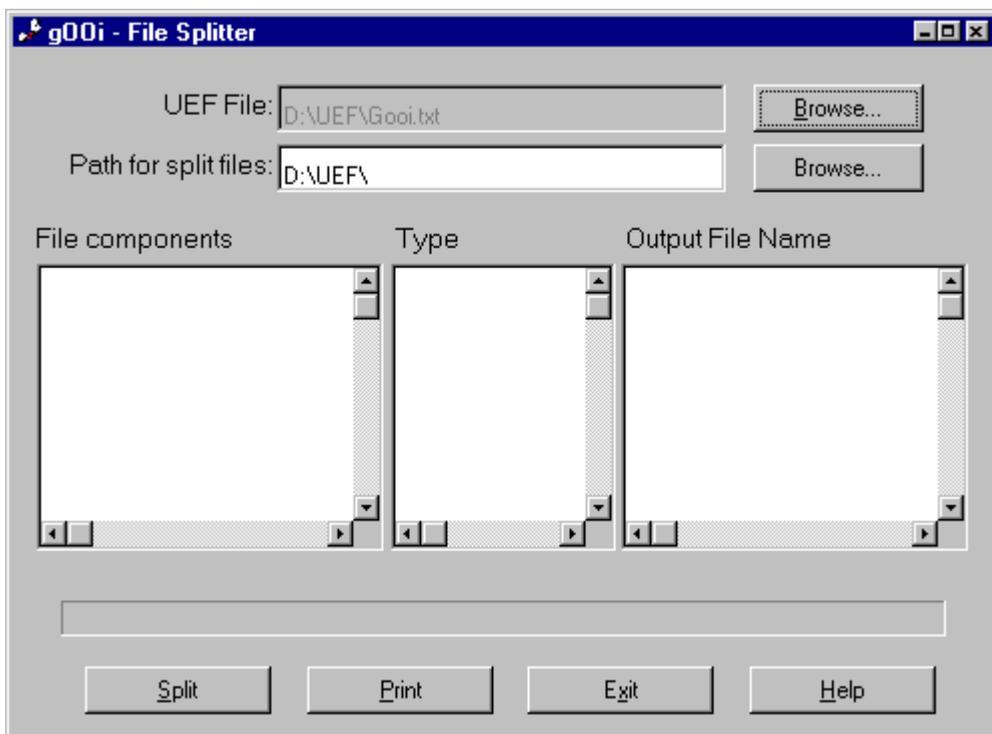
**Important!** Any existing files of the same name in the target directory will be overwritten without warning.

The following window displays:



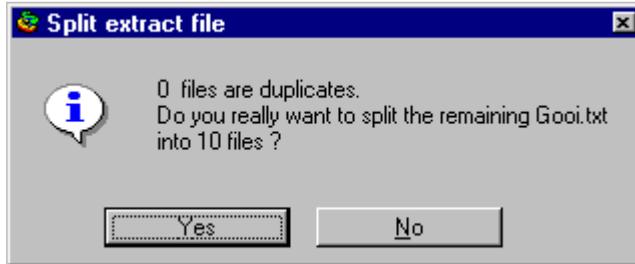
4. Select the target directory from the drop-down list. The resident files display. Click your UEF file name. The name displays in the File name field.

- Click **Open**. The File Splitter window displays your UEF input file selection and the path where the individual files will be placed following the split. By default, the Splitter will create the .exp files in the same folder where the input file is located:



If your UEF file includes multiple *instances* of the same screen or prompt, every occurrence except the last is designated “duplicate” in the Type column. In such a case, the File Splitter only generates a file for the last instance of the object encountered.

- When you click **S**plit, the input file is parsed to identify Screen and Prompter entities. These entities are listed under the 'File components', 'Type', and 'Output File' columns. You are prompted to confirm the operation, as shown in the following window:



The File Splitter generates as many files as there are unique objects in the list, and places them in the same target directory. Any existing files of the same name in the target directory will be overwritten without warning.

- Click **Y**es. A verification displays as follows:



- Click **O**K. You return to the File Splitter window.
- It is recommended that you print the results of the split for later reference.
- At the File Splitter window, click **E**xit. You return to the gOOi Tools window.
- Close the gOOi Tools window. You return to the gOOi Workplace window.

File splitting is complete. You are now ready to specify the host screen components of your gOOi application.

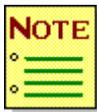
# 5

## Generating gOOi forms

### Overview of gOOi forms generation

After completing the following preparation tasks, you are ready to begin generating gOOi forms:

- ◆ Specifying host screen configuration settings and gOOi form generation options (see [“Specifying gOOi generation options and settings”](#) on page 45 for more information).
- ◆ (Optional) Perform pre-generation tasks, which are optional depending upon your environment (see [“Using environment-dependent tools”](#) on page 82 for more information).
- ◆ (Optional) Creating a template for function key mapping. Default templates are provided for all environments, and dynamic keyboard mapping is available for IBM mainframe users (see [“Creating a template for function key mapping”](#) on page 91 for more information).
- ◆ (Optional) Creating a template for visual items (see [“Creating an application template for visual items”](#) on page 103 for more information).
- ◆ (Optional) Creating .exp files using the File Splitter (see [“Creating .exp files with the File Splitter”](#) on page 118 for more information).



---

If you did not complete the preparation tasks, you should do so before continuing.

---

Generating gOOi forms in the default ObjectStudio format involves the following procedures:

1. Specify gOOi form components.
2. Generate gOOi application forms.

Both of these procedures are discussed in this chapter.

The generation process produces gOOi forms, each of which consists of:

- ◆ **An ObjectStudio controller.** A visual object that can be customized in ObjectStudio Designer.
- ◆ **A gOOi HostObject.** A non-visual object that contains host screen information. The host screen information is presented through the controller (the visual component of the gOOi form).

The separation of the host screen information into the host object allows you to regenerate the HostObject without losing any customizations that have been made to the controller.

After you have generated your forms, you can deploy them to end-user workstations (see “[Deploying a gOOi application](#)” on page 259).



---

The appearance of generated gOOi forms (interfaces) can be affected by moving them from finer to coarser display resolutions. Before gOOi application generation, set the resolution of your developer workstation to the coarsest resolution that will be displayed on any end-user workstation.

In other words, if your developer workstation has 1024 x 768 resolution, and you are going to deploy the gOOi application to an end-user workstation with 800 x 600 resolution, set your developer's resolution to 800 x 600.

The font size must also be considered. If your target end-user workstations use large fonts, ensure that your developer's workstation uses large fonts.

---

---

## Specifying gOOi form components

Once you have .exp files with one host screen object per file, you are ready to create gOOi forms for your host screens. To do this, you must specify each host screen component by extract file and type.



---

**MANTIS users:** Extract file types can consist of headers, footers, screens, pop-ups, menus, and prompts. When the File Splitter parsed your UEF file, it looked for only screen and prompter objects. For the file-splitting process, headers, footers, screens, pop-ups, and menus were classified as screens. Once the UEF file is split, each object file must be specified to gOOi by its exact type. Each of these types is explained later in this chapter.

---

### gOOi form generation restrictions

The following restrictions apply to gOOi form generation. None of these items prevents you from generating gOOi forms, but the results may be unsatisfactory:

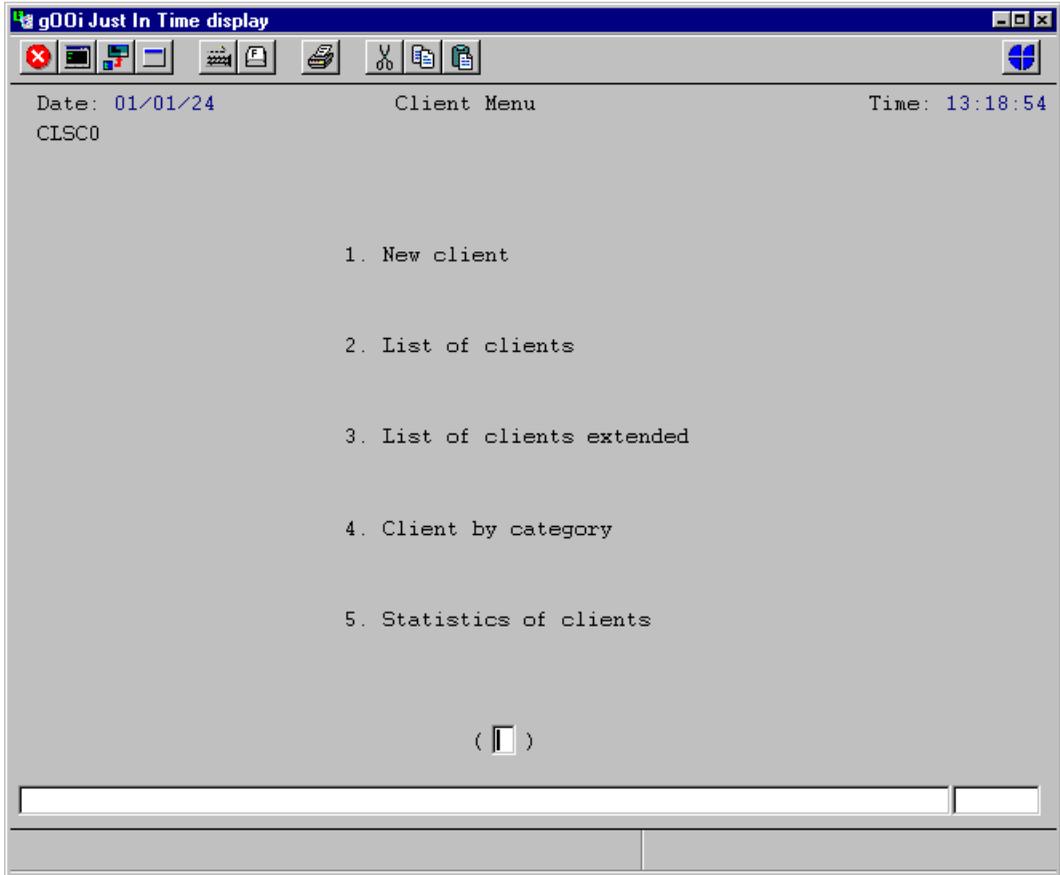
- ◆ gOOi does not support wraparound capabilities for 22 x 80 domain screens.
- ◆ gOOi does not support a host screen greater than 24 x 80.
- ◆ Two screen components cannot be placed side-by-side.
- ◆ If a screen serves as a menu, gOOi can generate the menu options in a list box if the screen design meets certain requirements. Select the menu type for your menu screen only if the following conditions are met:
  - The screen has an unprotected numeric field.
  - Menu options are in the following format:

*first-option-name*

*second-option-name*

[and so on]

Even if these conditions are met, you can elect to define the menu screen to gOOi as a screen type rather than as a menu type. The following Just In Time window illustrates a simple host screen that could be defined to gOOi as a menu or as a screen:



## **MANTIS users**

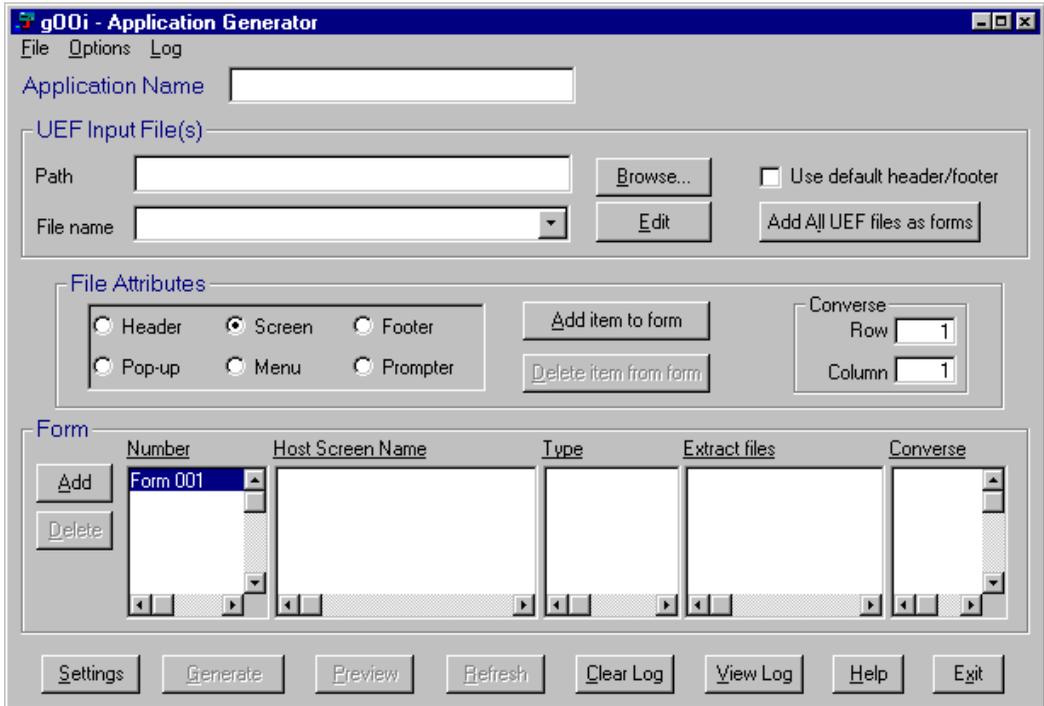
The following restrictions apply to MANTIS users:

- ◆ The status line on a gOOi form is generated using all the information from the 23rd line (message line) of the MANTIS screen if the map attribute is not FUL. If the map attribute is FUL, then no status line generates. You can get information to the status line by using SHOW with a semicolon in your MANTIS program.
- ◆ You cannot use the 24th line of a MANTIS screen for data input unless you authorize command line/key simulation generation (select Settings ⇒ Generation Options).
- ◆ We do not recommend using Dynamic CONVERSE to display host screen IDs at a position other than the one at which they were designed.

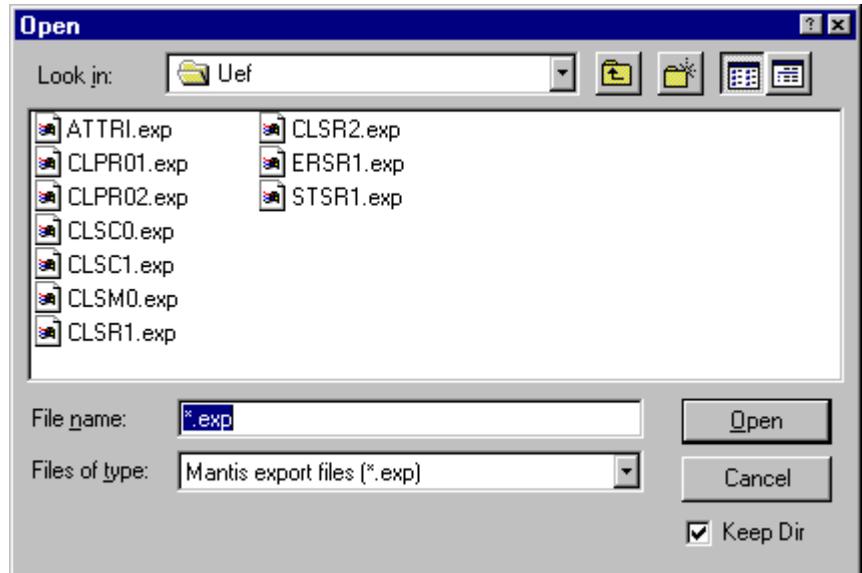
## Form component specification steps

To specify your host screen components for gOOi interface generation, perform the following steps:

1. From the gOOi Workplace window, double-click the gOOi Application Generator icon. The following window displays:

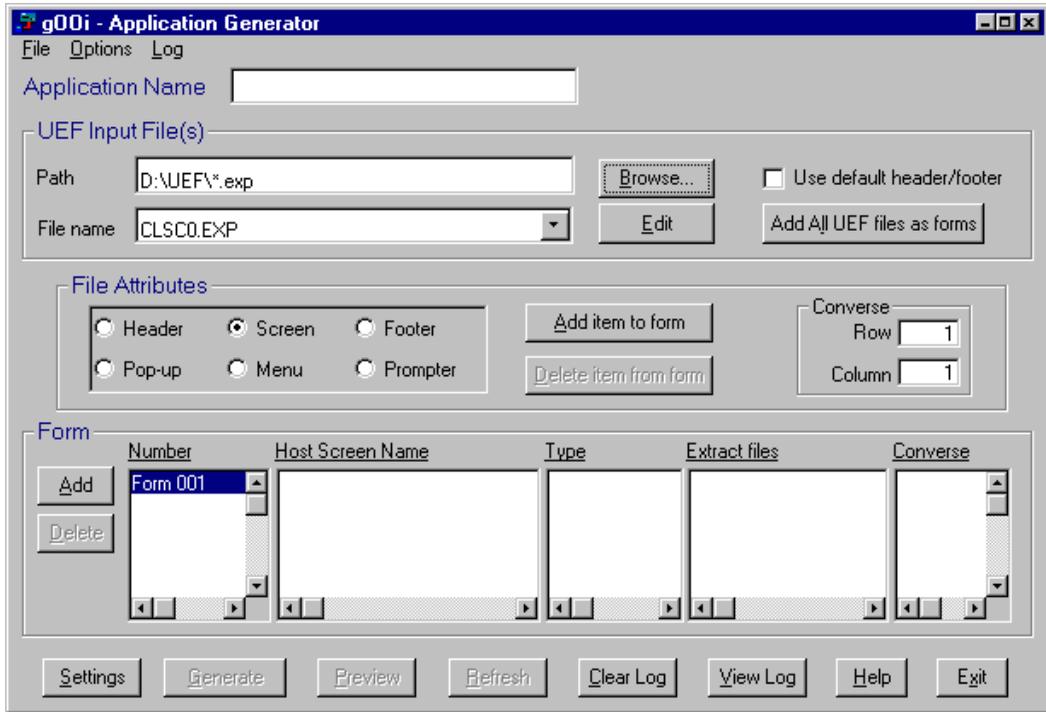


- Click **Browse**, or select the Browse extract path in the Options menu.  
The following window displays:



- Select the drive and directory containing the .exp files generated from your host application. All .exp files are in the same directory.
- Select a file and click **Open**. The Application Generator window displays. The Path field contains your selected path. The File name field contains the first listed file.

- To view the UEF file listed in the File name field, click **Edit**. The following window displays:



**AD/Advantage requirements:** Ensure that the header and footer information is in place and the AD/Advantage screen ID profile has been selected (see “[Specifying headers and footers](#)” on page 79). In the Application Generator, make sure that the Use default header/footer check box is selected.

- In the Application Name field, enter a name for the gOOi application to be generated for your host application. Use a maximum of 32 characters with no extension. This name is used to create a folder on the ObjectStudio desktop to store the icons for the interfaces you generate.

7. Determine the method by which you will generate your gOOi application. There are two possibilities:
  - If all the .exp files created from your host application are of file type Screen or Prompter, you can use the **Add All UEF files as forms** feature. Add All is designed to quickly generate a gOOi application when the source host application is a straightforward series of screens and prompters. Add All populates the application layout with one .exp file per gOOi form. If your .exp files meet this condition, proceed to step 8.
  - If the .exp files created from your host application are a mix of file types (for example, screens and prompters plus headers, footers, pop-ups, and menus), you must specify each gOOi form individually. If your .exp files meet this condition, proceed to step 9.
8. If you are going to use the Add All feature (see the first bulleted item in step 7), click **Add All UEF files as forms**. gOOi populates your application layout with one .exp file per gOOi form. Proceed to step 12.



---

When you click **Add All UEF files as forms**, you see as many Form IDs listed in the lower left as there are .exp files. However, you will see only one record in the remainder of the table (for Host Screen Name, Type, Extract files, and Converse). This is because you are viewing the details for Form001 only. To view details for other forms, click another Form ID.

---

9. If you are going to specify forms individually (see the second bullet in step 7), perform the following steps for every gOOi form you want to generate for your host application:

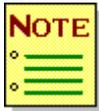


---

Although it may be logical to specify screens for gOOi in the order in which they appear in the host application, it is not mandatory to do so. The important thing is that each screen has its components specified correctly and that these components are associated with the correct .exp files.

---

- a. From the Form ID list, select Form001 to begin an application layout. (After completing Form001, click “+” to move to subsequent forms.)
- b. In the File Type box, select the type of .exp extract file that corresponds to the component you are specifying.



---

You can specify host screen components in any order, with one exception. If a host screen consists of multiple screen components (upper and lower screens), you must specify the screen components in order from top to bottom.

---

Specify the following options for File Type as appropriate:

Option	Description
Header	<p><i>MANTIS users:</i> Creates the first lines of the generated gOOi form. Every other component extract file generates lines following the header. A form can only have one header.</p> <p><i>AD/Advantage users:</i> You must select the Use default header/footer check box must be selected so that when you specify the UEF Input File(s), a File Type of Screen, then click either the <b>Add item to form</b> or <b>Add All UEF files as forms</b> button, gOOi automatically includes the necessary AD/Advantage header and footer with each screen.</p>
Screen	<p>Creates lines on the generated form. If a header is selected, the lines generated using the screen file are placed after it. If no header is defined, the lines display starting at the first line. There can be more than one screen extract file associated with a Form ID. The generated content of each component screen extract file displays in sequential order separated by one blank line.</p>
Footer	<p><i>MANTIS users:</i> Creates lines at the bottom of the generated form.</p> <p><i>AD/Advantage users:</i> You must select the Use default header/footer check box so that when you specify the UEF Input File(s), a File Type of Screen, then click <b>Add item to form</b> or <b>Add All UEF files as forms</b>, gOOi automatically includes the necessary AD/Advantage header and footer with each screen.</p>
Pop-up	<p><i>MANTIS users:</i> Creates a standard pop-up window corresponding to the pop-up currently in use in the host application. Pop-up forms cannot be generated alone. They must always be defined as a form specification component. For each gOOi Form ID, you must define each pop-up that displays in response to a keystroke. Several forms can call the same pop-up window. Shared pop-up windows are generated once and linked to the forms that call them.</p>



Be sure to specify all pop-ups that may be called by a screen. If an ungenerated pop-up is invoked, results will be unpredictable.

Specify the following options in separate Form IDs:

Option	Description
Menu	<p>Creates a menu interface with a special controller (which ignores all other controllers). If a screen serves as a menu, gOOi can generate the menu option in a list box if the screen design meets certain requirements. Select the menu type for your menu screen only if the following conditions are met:</p> <ul style="list-style-type: none"> <li>- The screen has an unprotected numeric field.</li> <li>- Menu options are in the following format:  <i>first-option-name</i>  <i>second-option-name</i>                      [and so on]</li> </ul> <p>Alternatively, you can define the menu screen as a screen type rather than as a menu type.</p>
Prompter	<p><i>MANTIS users:</i> Creates a special interface for a MANTIS help screen. The different pages of the prompter display in a scrollable list. A Next button is automatically provided to allow the user to open a chained prompter.</p>

- c. From the File Name drop-down list, select the extract file that contains the component you are currently specifying.
- d. **MANTIS users:** Perform this step only if your MANTIS application uses Dynamic CONVERSE to display your host screen at a position other than the one at which it was designed. Otherwise, proceed to step e.

The Converse field on the Application Generator window allows you to specify the row and column offset of a Dynamic CONVERSE. The default values are row 1, column 1 (1@1).

For example, a host application uses two fields displayed through:

```
CONVERSE MAP1 WAIT  
CONVERSE MAP2 UPDATE (20,1)
```

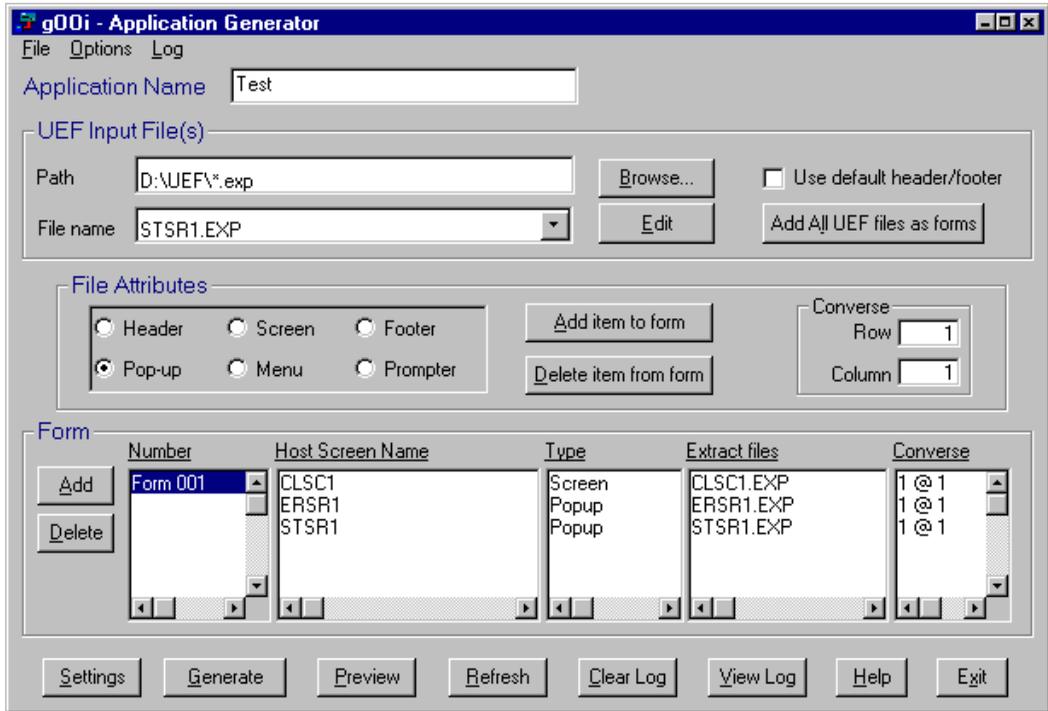
*Where:*

MAP1 =a screen designed at 1@1

MAP2 =a footer line designed at 1@1, but  
dynamically CONVERSEd at 20@1

To generate a gOOi form that duplicates the positioning of this mainframe display, define the gOOi form with two components: a screen with CONVERSE values of 1@1 and a footer with CONVERSE values of 20@1.

- e. Click **Add item to form** or double-click the file name. (If you make an error, select the extract file in the Form ID list and click **Delete**.) Your component displays in the list:



In the preceding example, note that the listed extract files are the components of the selected Form ID (Form 001). Once you have specified all your interfaces, selecting different Form IDs yields different extract file lists.

10. Repeat steps 9b through 9e for each Form ID component. After you have specified all the components for one Form ID, proceed to step 11.

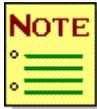
The following are two examples of gOOi form specifications. Form1 is composed of one header, one footer, two screens, plus two pop-up windows that may display:

Form 1	HEAD01.EXP	header
	PART1.EXP	screen
	PART2.EXP	screen
	FOOT01.EXP	footer
	LIST01.EXP	pop-up
	ERROR01.EXP	pop-up

Form2 uses a different header and footer than Form1, but shares two pop-up windows with Form1. Form2 also uses an additional pop-up:

Form 2	HEAD02.EXP	header
	CUSTOM.EXP	screen
	FOOT02.EXP	footer
	LIST01.EXP	pop-up
	ERROR01.EXP	pop-up
	LIST02.EXP	pop-up

11. After you have specified *all* components for *one* Form ID, select the next Form ID (with the “+” button) and repeat step 9. Proceed in this manner until all screens for your host application have been specified. Then proceed to step 12.



If you want to preview a form before generating the entire application, you can select the Form ID in the list and click **Preview**. Preview is handy for checking the effects of different font settings, or to see how your template controls display. You must still generate your forms to create a gOOi application.

Preview adds a temporary application to the ObjectStudio Work Area; this application is discarded when you terminate ObjectStudio. See “[Viewing an ungenerated form using Preview](#)” on page 146 for more information.

12. After you have specified *all* components for *all* Form IDs in your host application, read the following Note and make resolution adjustments, if necessary.



---

The appearance of generated gOOi interfaces can be affected by moving them from finer to coarser display resolutions. Before gOOi application generation, set the resolution of your developer workstation to the coarsest resolution at which it will ultimately be displayed on any end-user workstation.

In other words, if your developer's workstation has 1024 x 768 resolution, and you are going to deploy the gOOi application to an end-user workstation with 800 x 600 resolution, set your developer's resolution to 800 x 600.

---

The font size must also be considered. If your target end-user workstations use large fonts, ensure that your developer's workstation uses large fonts.

---

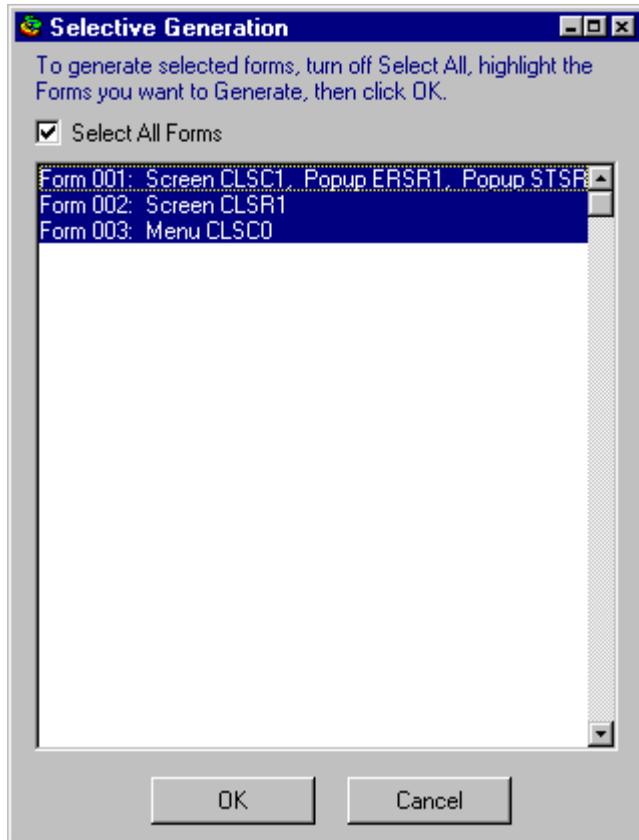
13. Ensure that the log file is enabled. The log file keeps a time-stamped record of your gOOi application generation session. It records the processing results of each phase of the generation process. If the Generator detects a warning or error condition, it notes the condition, but the Generator still continues. The log also records when a generated .cls file already exists (the existing file is renamed to .old).

To verify that the log is enabled, select ⇒ Log file options from the menu. The 'Enable log file' check box should be selected.

## Generating gOOi application forms

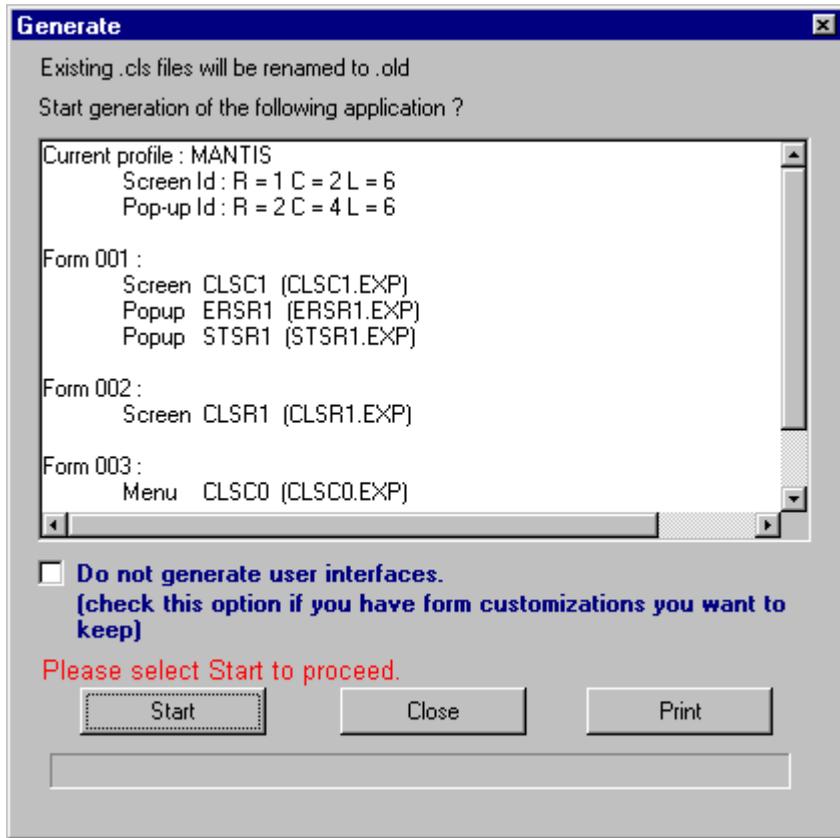
After specifying all components for all Form IDs (see “[Specifying gOOi form components](#)” on page 127), you are ready to generate the application forms as follows:

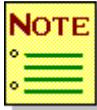
1. On the Application Generator window, click **Generate**. The first window to display allows you to generate only selected forms of the application:



This feature is useful if you want to regenerate part of an application that was previously generated, such as when a host screen changes. By default, all forms in the application will be generated. If you want to exclude forms, follow these steps:

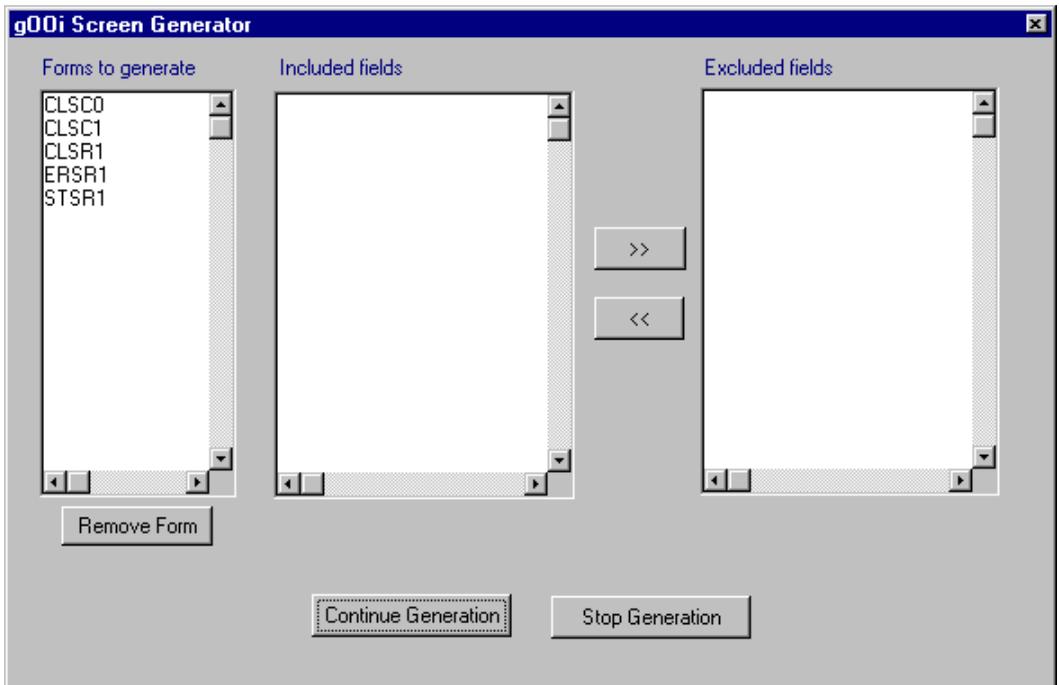
- a. Turn off the 'Select All Forms' check box.
  - b. Click on each form you want to generate. The form will be highlighted after you click it.
  - c. Click the **OK** button to continue generation.
2. The next window identifies the active profile and lists the forms that are included in the application. This window also includes a check box that can be selected to retain form customizations:





Selection of the 'Do not generate user interfaces' check box prevents gOOi from generating a controller (visual object) for the host screen. This preserves any previous customizations you may have made to the form generated for the host screen. gOOi will generate a new host screen object (non-visual object) that reflects all of the content of the host screen being processed. This allows you to add new content to a previously generated form from the newly generated host object without losing any form customizations.

3. Check the gOOi application forms list to verify that the profile and all forms are specified correctly. If the list is OK, click **Start**. The following window displays:



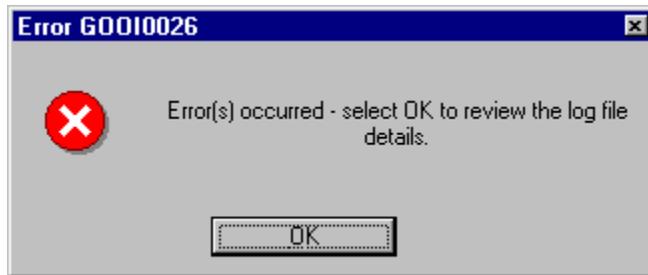
Use this window to exclude host fields from the generated gOOi forms. If you do not want to exclude any host fields, click **Continue Generation**.

If you want to exclude host fields, follow these steps:

- a. Select the form ID that contains the field(s) to be excluded.
- b. Select the generated screen name for this form ID.
- c. Select the generated field(s) to be excluded for this screen.
- d. Select the  button to move the field(s) into the Excluded fields list.

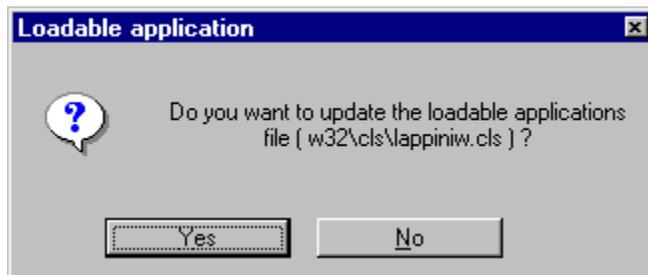
Repeat these steps as necessary, and then click **Continue Generation**.

4. If an error occurs during generation, the following message box is displayed:



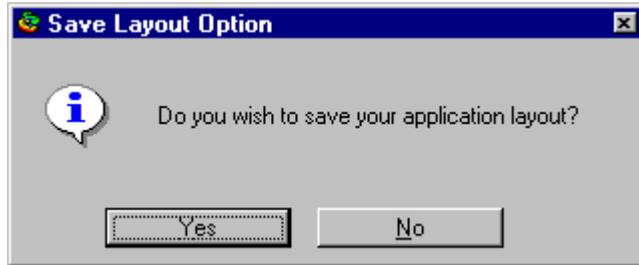
When you press **OK**, the contents of the gOOi log file are displayed in Notepad. The results of the most recent generation are at the end, which you can quickly jump to via the Ctrl/End keys.

If generation completes successfully, the following window displays:



You would normally click **Yes** to update the loadable applications file. If you choose **No**, you have to explicitly load your new classes the next time you start gOOi.

- When you respond to the preceding window, the Generate window again displays. Click **Close** to return to the Application Generator window. Click **Exit**; the following window displays:



We highly recommend that you save your application layout (definition). This is especially useful if you want to generate the same application several times (for example, to see the impact of a selected font).

You can save an application definition by selecting File ⇒ Save layout. Enter a valid file name. To avoid file overrides, use the default extension of .LAY. After responding to this window, you return to the gOOi Work Area.



The Application Generator automatically puts a new application into a folder. If you return to the ObjectStudio Desktop without exiting ObjectStudio first, you can open your application folder and your forms are arranged in a scrollable format. The next time you restart ObjectStudio, you have choices for loading and displaying your forms.

If you load your application using Load application, your forms appear as icons in the gOOi Work Area. If your application has many forms, the Work Area will be cluttered. You can create a new folder by selecting File ⇒ New ⇒ Folder. If you click mouse button 2 on the new folder and choose Load application into, you can load your application from the picking list. Once again, your forms are in a scrollable format. You also can put forms into a folder by clicking and dragging the form icons onto the folder icon.

## Viewing a gOOi form

If you have many gOOi forms to specify, you may want to view the appearance of your first form before you specify all forms. These are your options:

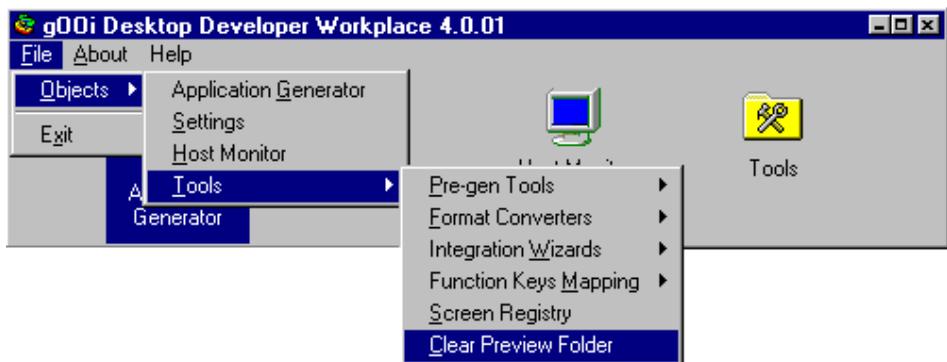
- ◆ (Recommended) You can view a gOOi form *before it is generated* by using the **Preview** feature on the Application Generator window.
- ◆ You can view a *generated* gOOi form *without* a host connection (with empty fields).
- ◆ You can view a *generated* gOOi form *with* a host connection (with real field values).

### Viewing an ungenerated form using Preview

When using the Application Generator to specify form components, you can use Preview to see a particular form before generating the entire application. At the Application Generator window, select the Form ID in the list and click **Preview**.

Preview is convenient for checking the effect of different font settings, or to see how your template controls display on the gOOi form. You must still generate your forms to create a gOOi application. Preview adds a temporary folder to the ObjectStudio Work Area, which is discarded when you terminate ObjectStudio.

The class files for the forms that you preview are created in the \gOOi\Preview folder under ObjectStudio. These files remain in this folder along with a Preview.txt file until you delete them. You can delete them via the Windows Explorer, or by using File ⇒ Objects ⇒ Tools ⇒ Clear Preview Folder from the gOOi Workplace menu as follows:



## Viewing a generated gOOi form without a host connection

There are two methods for viewing a generated gOOi form without a host connection (with empty fields) Viewing the generated form *outside* ObjectStudio Designer and Viewing the generated form inside ObjectStudio Designer.

### Viewing the generated form outside ObjectStudio Designer

Generate the form as described in this chapter. The generated form displays as an icon in the ObjectStudio Work Area. Double-click the icon to display the form. This method is adequate for viewing the placement of fields. What you see depends on what type of form it is. If the form is dependent on run-time data (for example, prompter text), the body of the form will be empty. If the form has many data fields, you can view their placement.

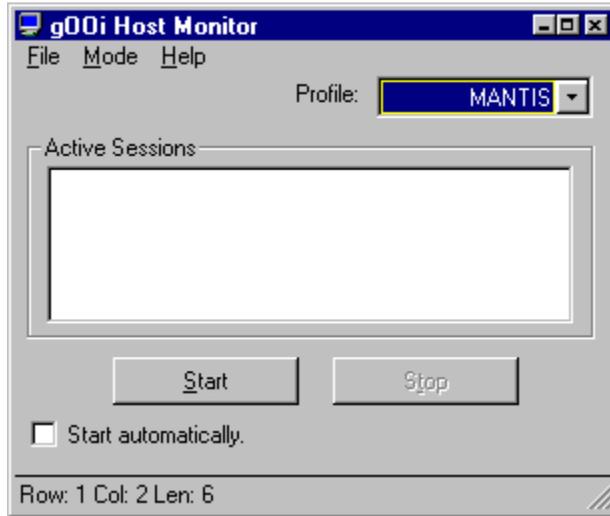
### Viewing the generated form *inside* ObjectStudio Designer.

Generate the form as described in this chapter. The generated form displays as an icon in the ObjectStudio Work Area. Open the form in ObjectStudio Designer. Click the form icon to select it, then right-click the form icon and select Edit. The form displays in Edit mode. This method is adequate for viewing the placement of fields. To ensure that you are viewing *all* fields on the form, select FormItem ⇒ Select all.

To verify the appearance of the form at runtime, select 'File ⇒ Test interface' to view the form in Test mode. After viewing, select File ⇒ Exit. To use ObjectStudio Designer to move fields and enhance the interface, see "[Basic customizations](#)" on page 164. For information about ObjectStudio Designer training for gOOi, contact your Cincom representative.

## Viewing a generated gOOi form with a host connection

To view a gOOi form with real values displaying in the fields, connect to a host by clicking on the HostMonitor icon on the gOOi Workplace window. Select your emulator and profile, and then click **Start**. The following window displays:



When a session becomes active, the GOOHostMonitor box is minimized. Run your host application. If a gOOi form exists for the active host screen, the gOOi form is displayed.

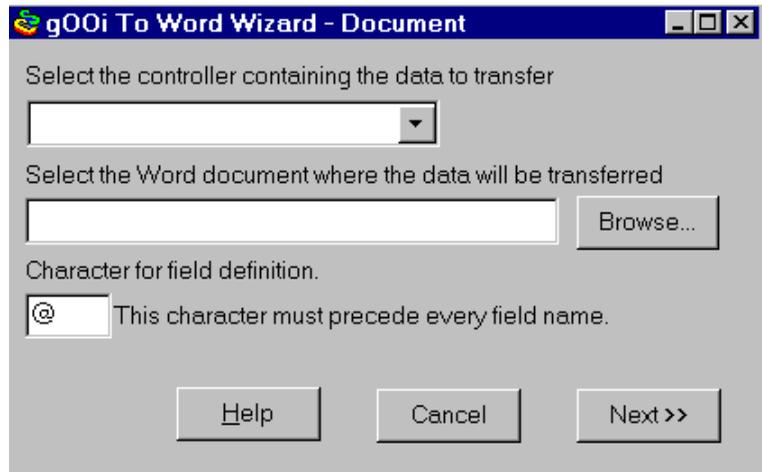
# 6

## Using integration wizards

### Word Wizard tool

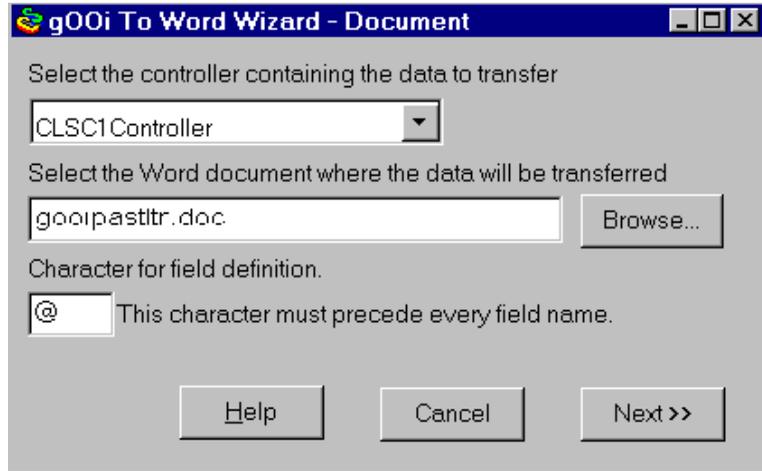
The Word Wizard tool provides an easy way of integrating a host application with Microsoft Word for Windows. The wizard allows you to pick the desired fields from a selected gOOi form for transfer to a Word document and adds a button to the form that executes the data transfer at run time. The code to perform the data transfer is generated by the wizard.

From the gOOi Workplace window, double-click the Tools icon, then double-click the Word Wizard icon. The gOOi To Word Wizard window displays:



To use the Word Wizard tool, perform the following steps:

1. Choose the gOOi form containing the data that you want to insert in a Word document, then select the target Word document:



2. Click **Next >>**.

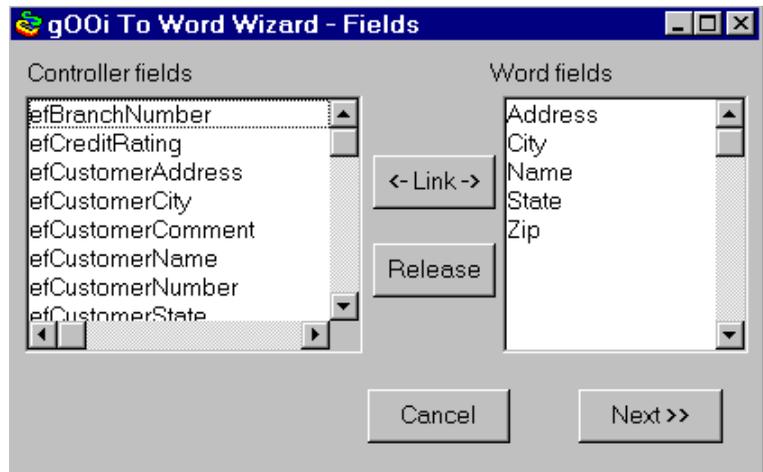


---

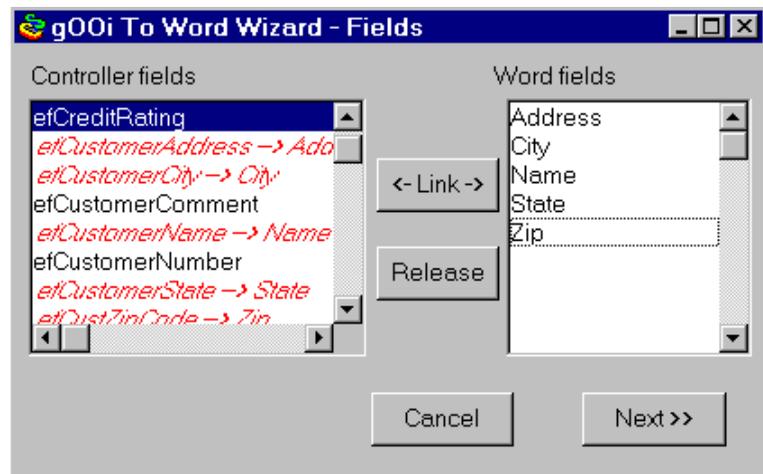
In the Word document, fields that are to receive data from the gOOi form must be preceded by an identifier character (@ by default).

---

- Match the fields on the gOOi form with fields on the Word document by selecting the desired controller field and then selecting the corresponding Word field:



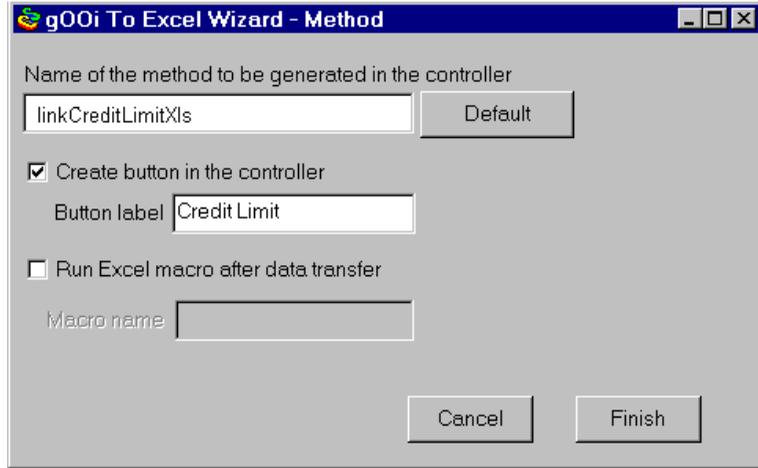
- Click **<-Link->** to establish the connection between these fields. The following window displays:



gOOi displays the relationship by appending an arrow and the name of the Word field after the name of the controller field. Linked fields display in italics.

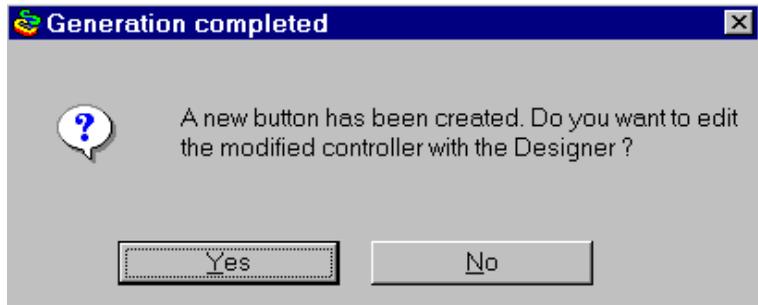
- Repeat steps 3 and 4 until all gOOi form/Word document field connections have been established, and then click **Next**.

- Name the Smalltalk method to be generated within the gOOi form. (It is a good idea to include the name of the Word document within the method name. This approach makes it easy to distinguish between methods if your gOOi form is integrated with multiple Word documents.) The method name is shown in the following window:



For example, if the name of your Word document is Past Due Letter, you might specify a method name of linkPastDueLetterDoc.

- Click **Finish**. The following window displays:



- Click **Yes** to create a button on the gOOi form. When the user selects this button at run time, the gOOi form to Word document data transfer will occur through the Smalltalk method.



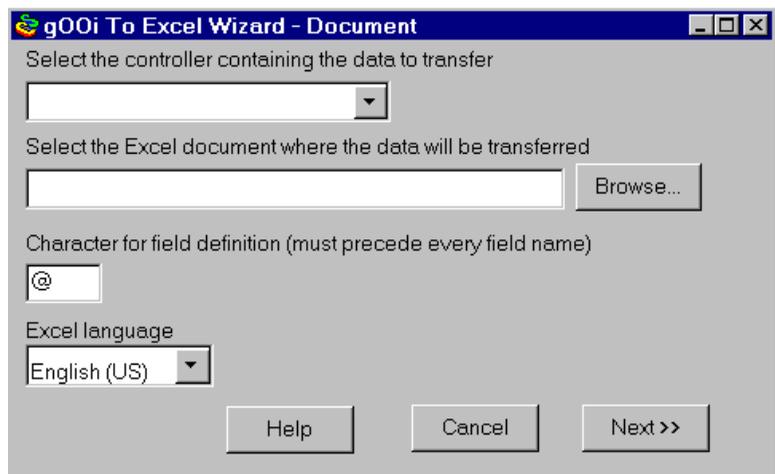
After creating a button with the wizard, ObjectStudio Designer displays the gOOi form so you can verify that the button is positioned where you want it.

---

## Excel Wizard tool

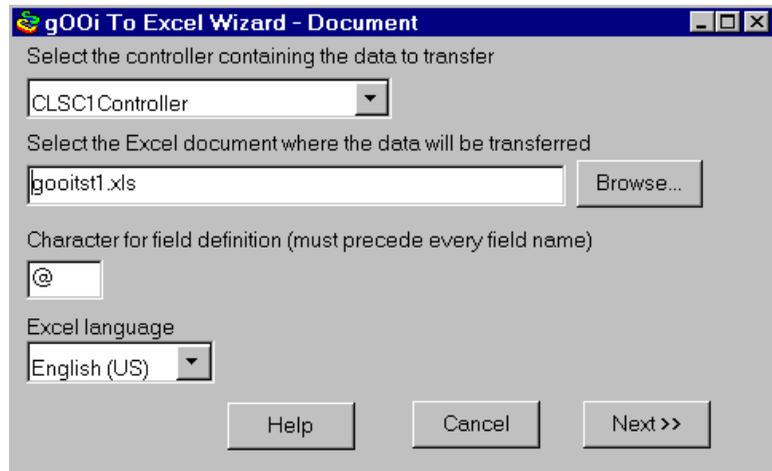
The Excel Wizard tool provides an easy way of integrating a host application with Microsoft Excel for Windows. The wizard allows you to pick the desired fields from a selected gOOi form for transfer to the desired location in an Excel spreadsheet. The wizard then adds a button to the form that executes the data transfer at run time and generates the code to perform the data transfer.

From the gOOi Workplace window, double-click the Tools icon, then double-click the Excel Wizard icon. The gOOi To Excel Wizard window displays:



To use the Excel Wizard tool, perform the following steps:

1. Select the gOOi form containing the data you want to transfer to an Excel spreadsheet, then select the target Excel spreadsheet. You can link gOOi form fields as a group to a sheet within the spreadsheet, to individual cells, or to individual fields. If you link to individual fields, the target fields in the Excel spreadsheet must be preceded by an identifier character (@ by default). To enter the @ character in a cell, enclose the @ character and the field name within quotes, such as "@Address". A sample window follows:

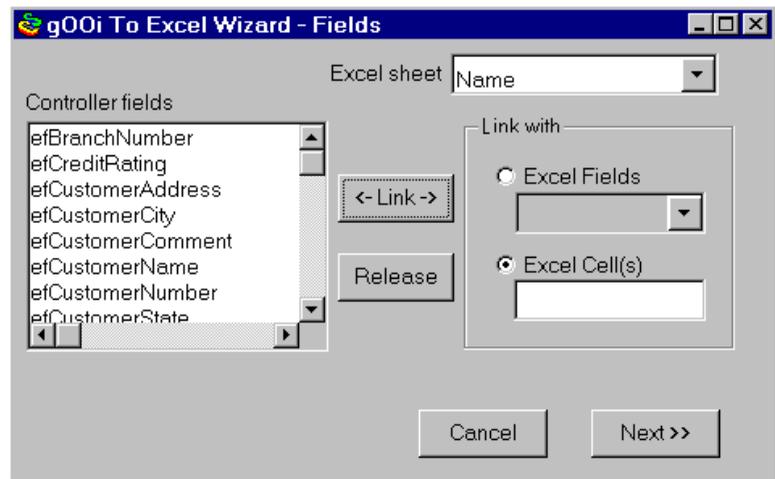


2. Click **Next**, then use one of the following methods to connect gOOi form fields to Excel spreadsheet cells:
  - Select a range of gOOi form fields and a sheet within the Excel spreadsheet, then click **Link**.
  - Select a gOOi form field and a spreadsheet cell, then click **Link**.
  - Select a gOOi form field and a named Excel field, then click **Link**.

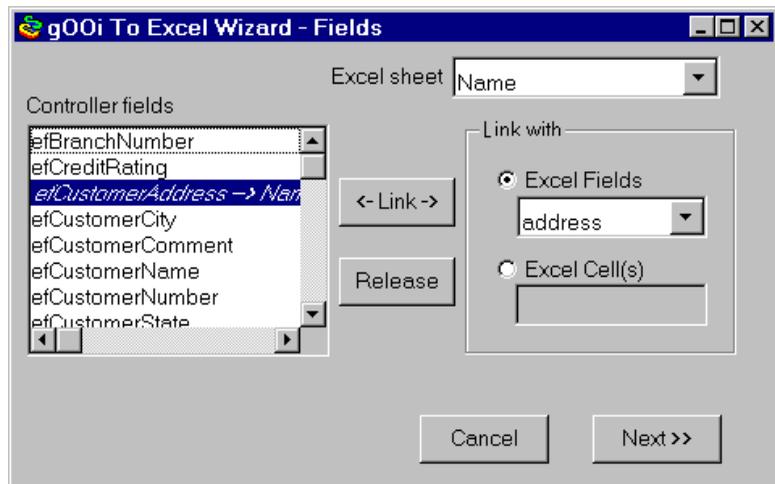


In all three cases, gOOi displays the relationship by appending an arrow and the name of the Excel cell or field after the name of the controller field.

The following example shows the controller fields prior to linking:



After linking, gOOi appends an arrow and the name of the Excel field after the name of the controller field. Linked fields display in italics as follows:



3. Establish connections for controller fields and Excel cells or fields as appropriate, and then click **Next**. The gOOi to Excel Wizard-Method window displays.

4. Name the Smalltalk method to be generated within the gOOi form. If your gOOi form is integrated with multiple Excel spreadsheets, including the name of the Excel spreadsheet within the method name helps to distinguish among methods.

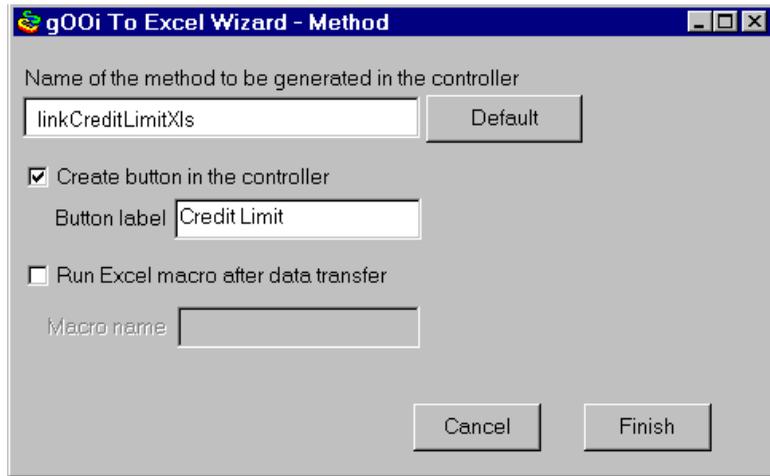


---

You can also specify an Excel macro name that will be executed within Excel after the data has been transferred.

---

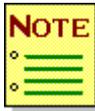
For example, if the Excel spreadsheet name is Credit Limit, you might specify a method name of linkCreditLimitXls as in the following window:



5. Click **Finish**. The following message box displays:



6. Click **Yes** to create a button on the gOOi form. Selecting this button at run time invokes the Smalltalk method for transferring data from gOOi to Excel.



---

When you create a button, ObjectStudio Designer gets focus after completion of the wizard. You can then see if the button is positioned where you want it on the gOOi form.

---



# 7

## Customizing gOOi forms

### Overview of customization

Generated gOOi forms can be modified and extended by using ObjectStudio's Designer tool. This section discusses how to perform the following basic customizations:

- ◆ Moving fields
- ◆ Changing the size and shape of fields
- ◆ Changing a field's color, font, and/or justification
- ◆ Changing the field's presentation type
- ◆ Grouping fields graphically
- ◆ Changing the form title and/or background



When you are customizing your gOOi screens, be sure to exit ObjectStudio Designer before you test them. If you do not, unexpected results will occur, even if your customization was done correctly.

Tabbing (item traversal sequence) between your gOOi form fields will by default follow the sequence of the corresponding host emulator screen. If you customize the gOOi form to use a different field ordering, you can also change the item traversal sequence by selecting Form ⇒ Item Traversal from ObjectStudio Designer.

The customizations explained in this chapter require no knowledge of ObjectStudio or Smalltalk, but such knowledge would increase the possibilities for gOOi extensions. To learn more about extending gOOi applications, refer to the following ObjectStudio manuals:

- ◆ *ObjectStudio Smalltalk User's Guide*, P40-3202, gives conceptual and procedural information about the Smalltalk language. It contains exercises that show you how to use Smalltalk.
- ◆ *ObjectStudio User's Guide*, P40-3201, describes how to use Smalltalk and ObjectStudio tools to build applications.
- ◆ *ObjectStudio User Interface Guide*, P40-3205, describes how to use ObjectStudio Designer to build interfaces for applications.

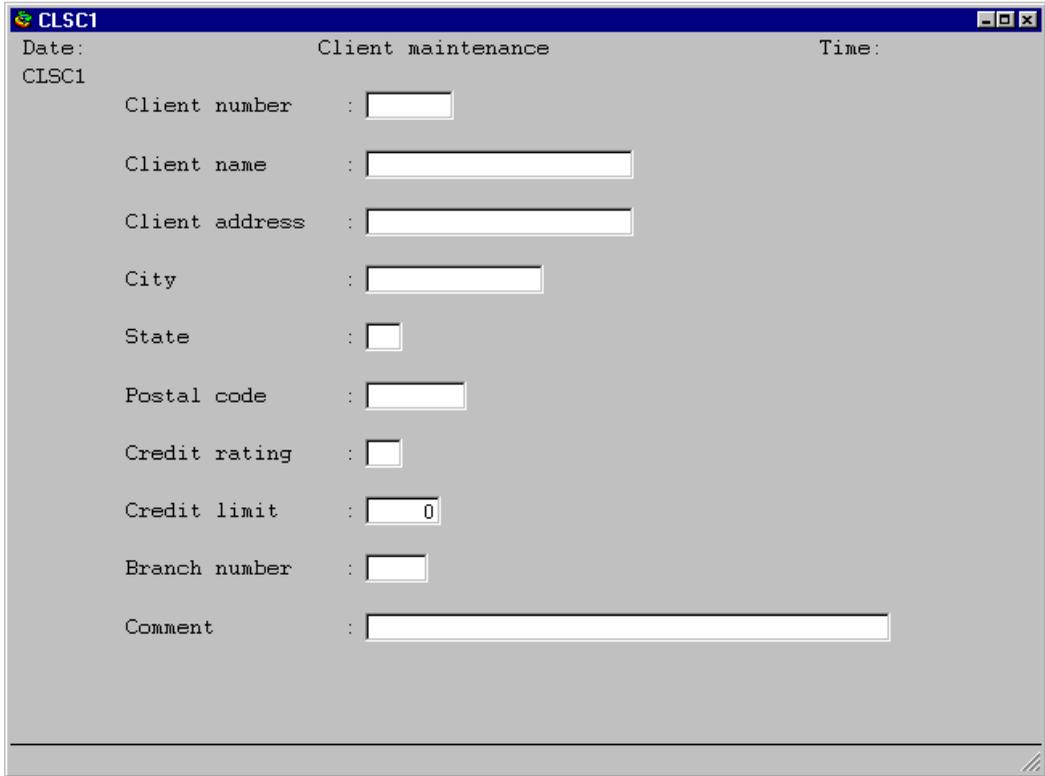
For training on using ObjectStudio to enhance gOOi applications, contact your Cincom representative.

## Bringing a gOOi form into ObjectStudio Designer

To bring a gOOi form into ObjectStudio Designer, perform the following steps:

1. Load the application you want to customize. If you updated the loadable applications file when you generated your application (see “[Editing your loadable application](#)” on page 102), your application displays in the application list when you start gOOi (see “[Loading gOOi](#)” on page 42). If you did not update the loadable applications file, you must explicitly load your new forms. When your application is loaded, each form displays as an icon in the ObjectStudio Work Area.

2. With mouse button 2, click the gOOi form icon you want to customize. Select Edit from the menu. The gOOi form displays in ObjectStudio Designer with empty fields as shown in the following example form:



The screenshot shows a window titled "CLSC1" with a subtitle "Client maintenance". The window contains a form with the following fields:

Date:	Client maintenance	Time:
CLSC1		
Client number	:	<input type="text"/>
Client name	:	<input type="text"/>
Client address	:	<input type="text"/>
City	:	<input type="text"/>
State	:	<input type="text"/>
Postal code	:	<input type="text"/>
Credit rating	:	<input type="text"/>
Credit limit	:	<input type="text" value="0"/>
Branch number	:	<input type="text"/>
Comment	:	<input type="text"/>

3. The ObjectStudio Designer toolbar is independent of the gOOi form you intend to customize. The toolbar may be behind the form window. If the toolbar is hidden, you can use Alt/Tab to move to it, or click on the Designer button on the Windows Taskbar.

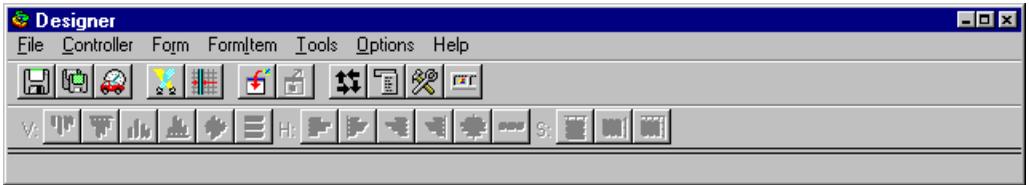


---

The display size of a gOOi form depends on the font size of the PC where the form is displayed. If you are working on a PC where the display size is set to small fonts, please keep in mind that the form may appear significantly larger on a PC where the display is large fonts. This can mean that the form will need scroll bars for all the data to be viewed on a large font PC.

---

A sample toolbar is illustrated in the following window:



4. Perform customizing tasks (see "[Basic customizations](#)" on page 164).

## Basic customizations

### Moving fields

After loading your gOOi application and moving a gOOi form into ObjectStudio Designer, you can move form fields to new locations one at a time, or in groups.

#### Moving fields one at a time

Perform the following steps to move a single field:

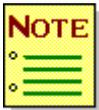
1. Select the field (by clicking over it).
2. Drag the field to a new location. You can activate Grid and Snap capability under the Form menu.
3. When you are satisfied with your changes, select File ⇒ Save.

#### Moving multiple fields

You can move multiple fields together by selecting them as a group. This allows you to align fields.

To move multiple fields together, perform the following steps:

1. Imagine a box that would enclose the fields you want to select as a group. Move your mouse pointer to the upper left of this imaginary box.
2. Click and drag (hold down mouse button 1) toward the lower right until all target fields are selected. During this process the selected fields will be outlined with a dashed box. This technique is referred to as *lassoing*.

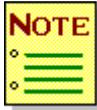


---

An alternative method to steps 1 and 2 is to hold down CTRL and click each field.

---

3. Click and drag your selected grouping to the desired location. You can activate Grid and Snap capability under the Form menu.

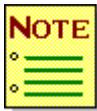


---

Notice that one of your selected fields has a thicker border. ObjectStudio Designer designates this bolded field as the selection head. Some ObjectStudio Designer alignment tools adjust the alignment of multiple fields relative to the selection head. For instance, all selected fields can be right aligned with the right side of the selection head. You can designate a different selection head by clicking one of the other selected fields (the last item clicked is the selection head).

---

4. You can adjust field alignment using the blue and red alignment tool buttons on the ObjectStudio Designer toolbar. If you cannot see the tool buttons, activate the toolbar.



---

To see alignment toolbar button descriptions, select the following topics: Help ⇒ Help index ⇒ Creating User Interfaces ⇒ The Designer Toolbars ⇒ The Designer Alignment Toolbar.

---

5. When you are satisfied with your changes, select File ⇒ Save. The following window displays fields moved to new locations:

The screenshot shows a window titled 'CLSC1' with a dark blue header bar. Below the header, the text 'Date: CLSC1' is on the left, 'Client maintenance' is in the center, and 'Time:' is on the right. The main area contains several input fields:

- Client number :
- Client name :
- Client address :
- City :  State :
- Postal code :
- Credit rating :  Credit limit :
- Branch number :
- Comment :

## Changing the size and shape of fields

After loading your gOOi application and moving a gOOi form into ObjectStudio Designer, you can change the size and shape of form fields. You can modify individual fields manually, or you can instruct ObjectStudio Designer to automatically resize selected groups of fields.

### Manually changing a field's size and shape

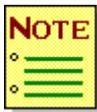
Perform the following steps to change the size and shape of a field:

1. Select the field. Selection handles (□) display along the field's perimeter.
2. Change the size and shape of the field by clicking and dragging the selection handles to resize it. You can activate Grid and Snap capability under the Form menu.
3. When you are satisfied with your changes, select File ⇒ Save.

### Automatically resizing selected fields

ObjectStudio Designer provides tools for automatically resizing a selected group of fields. To automatically resize fields, perform the following steps:

1. Imagine a box around the fields you want to select as a group. Move your mouse arrow to the upper left corner of this imaginary box.
2. Click and drag toward the lower right until all target fields are selected.



---

An alternative method to steps 1 and 2 is to hold down CTRL and click each field with the mouse.

---

3. Resize the selected fields using the blue and red resizing tool buttons on the ObjectStudio Designer toolbar.



Notice that one of your selected fields has a thicker border. ObjectStudio Designer designates this bolded field as the selection head. The ObjectStudio Designer resizing tools resize fields to one or more specifications of the selection head. For instance, all selected fields can be resized to the width and height of the selection head. You can designate a different selection head by clicking on one of the other selected fields.



To see resizing toolbar button descriptions, select the following topics: Help ⇒ Help index ⇒ Creating User Interfaces ⇒ The Designer Toolbars ⇒ The Designer Alignment Toolbar ⇒ Resize.

4. When you are satisfied with your changes, select File ⇒ Save.

The following window displays data fields lengthened and widened:

CLSC1

Date: Client maintenance Time:

CLSC1

Client number :

Client name :

Client address :

City :  State :

Postal code :

Credit rating :  Credit limit :

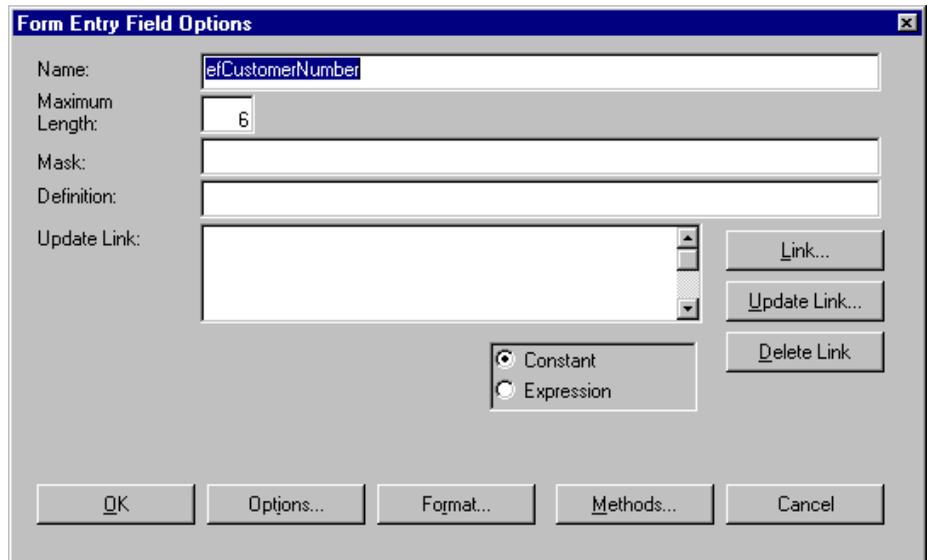
Branch number :

Comment :

## Changing a field's color, font, and/or justification

After loading your gOOi application and opening a gOOi form in ObjectStudio Designer, you can change a field's color and/or font as follows:

1. Double-click the field you want to change. A Form Entry Field Options window displays:



The image shows a dialog box titled "Form Entry Field Options" with a close button (X) in the top right corner. The dialog contains several input fields and buttons:

- Name:** A text box containing "efCustomerNumber".
- Maximum Length:** A text box containing "6".
- Mask:** An empty text box.
- Definition:** An empty text box.
- Update Link:** A large empty text box with a vertical scrollbar on the right side.
- Link Type:** A group box containing two radio buttons: "Constant" (selected) and "Expression".
- Buttons:** On the right side, there are three buttons: "Link...", "Update Link...", and "Delete Link". At the bottom of the dialog, there are five buttons: "OK", "Options...", "Format...", "Methods...", and "Cancel".

2. Click **Format**. The following window displays:

**String Display Format**

Prefix

Suffix

Font

Size

Foreground

Background

Vertical justification  
 Top  
 Center  
 Bottom

Horizontal justification  
 Left  
 Center  
 Right

Style  
 Italic  
 Bold  
 Underscore  
 StrikeOut

Example

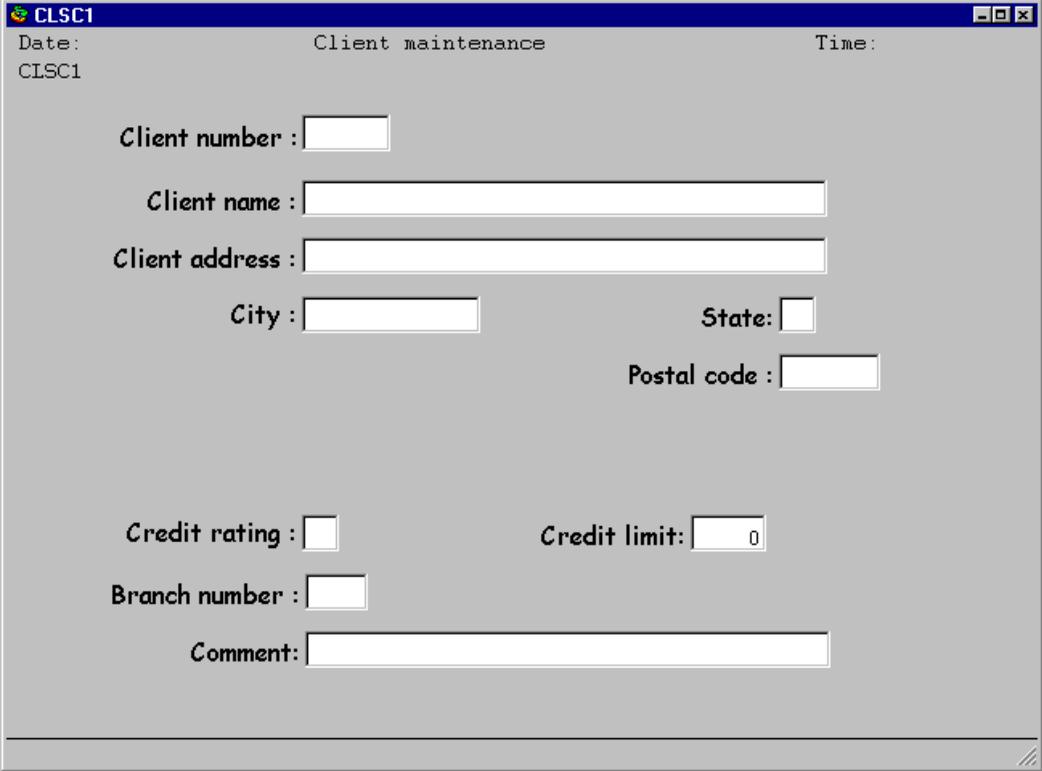
3. Select format options. The effects of your choices display in the Example area.



For information about maintaining host application field attributes (one of which is color), see [“Specifying host field attribute matching”](#) on page 59.

4. When you are satisfied with your format selections, click **OK**. You return to the Form Entry Field Options window.
5. Click **Cancel** or **OK**. You return to your gOOi form.

- When you are satisfied with your changes, select File ⇒ Save. The following window displays font and font size changes in static text fields:



The screenshot shows a window titled "CLSC1" with a blue header bar. Below the header, the text "Date: CLSC1" is on the left, "Client maintenance" is in the center, and "Time:" is on the right. The main area contains several input fields:

- Client number :**
- Client name :**
- Client address :**
- City :**  **State:**
- Postal code :**
- Credit rating :**  **Credit limit:**
- Branch number :**
- Comment:**

## Changing a field's presentation type

After loading your gOOi application and opening a gOOi form in ObjectStudio Designer, you can change a field's type. The example in this section shows how to change a blank data entry field into a drop-down list box, a set of radio buttons, or a check box.

In each case, the change provides choices for selecting values, reducing the chance of error that exists with an empty field. In effect, you can enhance an application independent of the host application.

Perform the following steps to change the presentation type for a field:

1. Select the field.



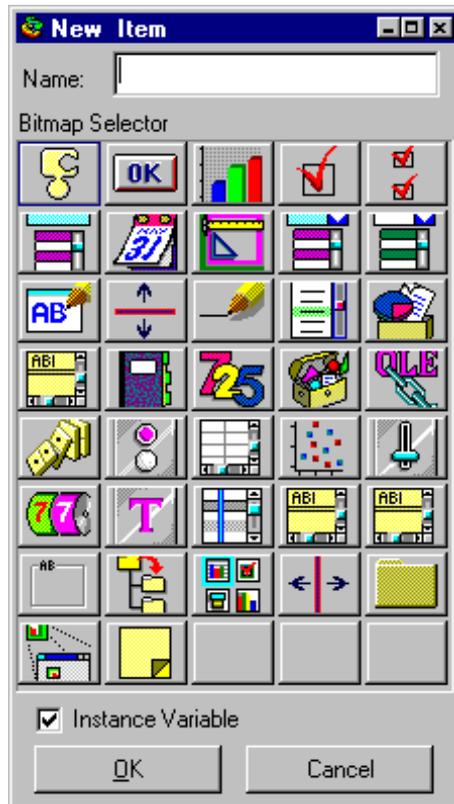
---

Note the exact field name for later use. To verify the field name, double-click the field; the name displays in the Form Entry Field Options window. (You can also click the field name; the name displays on the status line, if the status line is activated.) After noting the name (case-sensitive), click **Cancel** to return to the gOOi form.

---

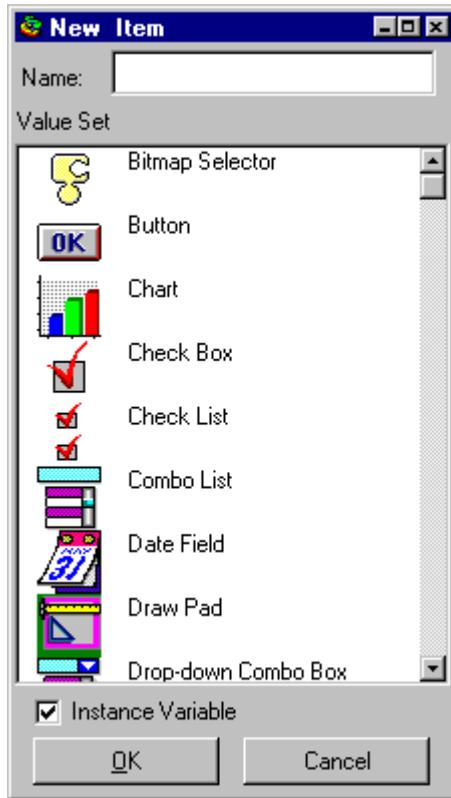
2. Select Controller ⇒ Subject. The selected field is listed after Item: at the bottom of the Subject window. The Mapped To: entry consists of the object name, a period, and the selected field name (for example, CustomerHostObject.efName). Note the mapped name for later reference.
3. Delete the selected field by either pressing Ctrl+D or selecting FormItem ⇒ Delete Item(s). You are prompted to confirm the deletion.

4. Create a new item either by pressing Ctrl+N or selecting FormItem ⇒ New Item. The following New Item window displays showing icons for available items:





When you click an icon, the type description—drop-down list box, radio button, or check box—displays near the top of the screen. If you click in the white space between the icons with mouse button 2, you receive an option box. If you select List, the icons are presented alphabetically in a scrollable format with their associated descriptions.



5. Click the field type you want. For this example, select Drop-Down List Box or Radio Button.
6. In the Name field, enter the field name you noted in step 1.

7. Click **OK**. For this example, the drop-down list box now displays on your gOOi form.
8. Double-click the drop-down list box field. The Form Drop-down List Box Options window or Form Radio Buttons Options displays. Both windows have the same content, but different titles.

As shown in the following example, enter the options you want displayed in the list box. Enter each option in the text box below the Labels/Values box and click **Add**. The option displays in the Labels/Values list:

The screenshot shows the 'Form Drop-down List Box Options' dialog box. The 'Name' field is 'efCreditRating'. There are two list boxes: 'Labels' and 'Values'. The 'Labels' list contains 'A1', 'B1', and 'C1', with 'C1' selected. The 'Values' list contains 'A1', 'B1', and 'C1', with 'C1' selected. Below the lists are two empty text boxes for adding new items. At the bottom, there is a 'Number of Visible Items' field set to '0'. On the right side, there are buttons for 'Add', 'Insert', 'Delete', 'Link...', 'Update Link...', and 'Delete Link'. At the bottom of the dialog are buttons for 'OK', 'Options...', 'Format...', 'Methods...', and 'Cancel'.



To retrieve values from a database table instead of hard coding them, you must use an ObjectStudio-compatible database. Click **Link** and select Table; the database table name and field that has the credit rating values displays. Once the link is established, the values from the field display on the gOOi form at run time.

9. To specify a distinction between the option name displayed in the list box and the value that is actually transmitted to the host application, change the option Label. For example, you could specify a Label of AAACredit for Value A1. To do this, double-click the entry in the Labels column. You are prompted for a new name.
10. New items receive default format settings (they are unaffected by format settings you have specified for existing fields). If you want the formatting of a new item to be consistent with the rest of the fields, you must click **Format** and explicitly set the values.
11. After clicking **OK** on the Form Options window for the form item type you are using, make sure that the new form item is selected. Select Controller ⇒ Subject. Verify that the Attribute radio button is selected with the name of the selected field. Click **Map**. The Mapped To: information from step 2 is restored. Click **Close**.
12. If you choose Drop-down List Box or Radio Buttons as your new form item, no Smalltalk coding additions are necessary; you can skip this step. For a check box, select Tools ⇒ Class Browser from the ObjectStudio Work Area. Under Classes, go to the class name that consists of your gOOi form name suffixed by Controller. Under Methods, we want to add a set method using the name of the check box that you added. The purpose of a set method is to set the value of an object (for example, a check box) based on data passed to the set method.

The name of a set method is the name of the object with a colon appended. For example, each host object has a set method corresponding to the entry fields on the host screen. In the case of the credit limit field, (nfCreditLimit, where nf indicates a numeric field), the set method of the host object is as follows:

```
nfCreditLimit: aValue
nfCreditLimit := aValue.
```

For this example, suppose you are changing the gOOi form to replace the credit limit field with a check box that is selected only if the customer credit limit exceeds 10000. To accomplish this, leave the set method of the host object intact, and add a simple Smalltalk set method to the controller as follows:

```
nfCreditLimit: aValue
(aValue asNumber > 10000) ifTrue: [
    nfCreditLimit display: true.
] ifFalse: [
    nfCreditLimit display: false.
].
```

The preceding code assumes that the original name was assigned to the check box in step 6 and this check box was linked with the corresponding host object variable in step 11. It is possible with gOOi to use a different name for the replacement form item. In this example, “cbCreditLimit” (where cb indicates a check box) could have been assigned as the check box name in step 6 and cbCreditLimit linked to nfCreditLimit of the host object in step 11. The set method would then be:

```
cbCreditLimit: aValue

(aValue asNumber > 10000) ifTrue: [
    cbCreditLimit display: true.
] ifFalse: [
    cbCreditLimit display: false.
].
```

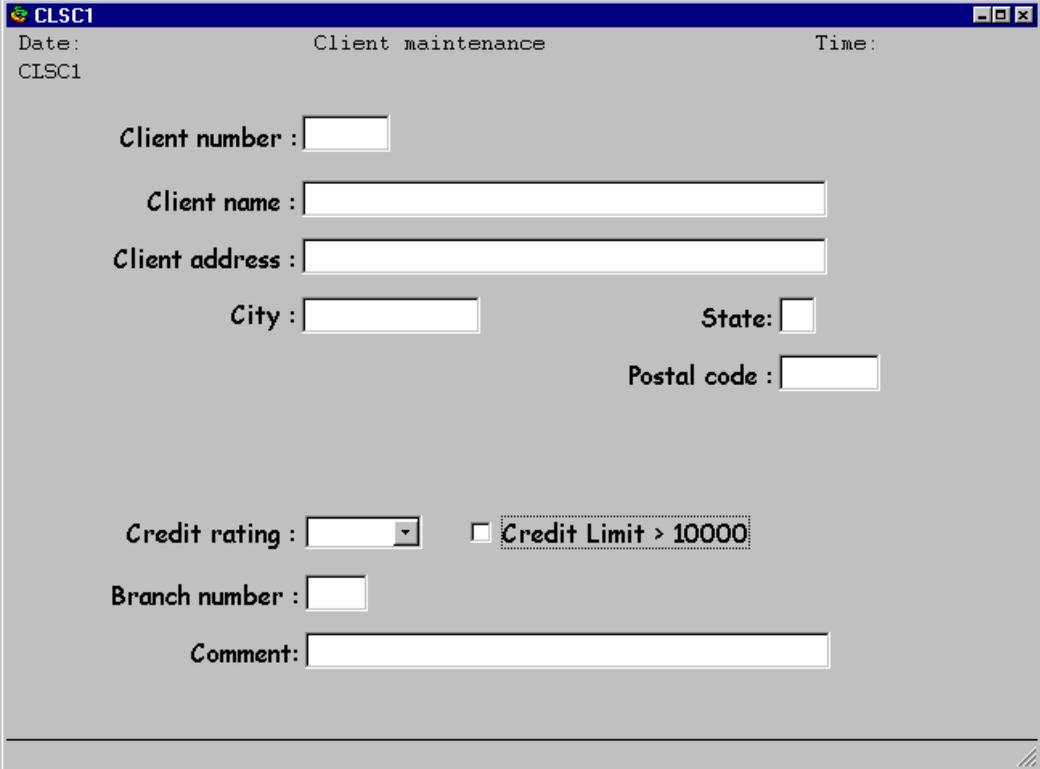
13. To test the new presentation type for the modified field, select File ⇒ Test interface. Your gOOi form is placed in test mode. For this example, clicking on the drop-down list box field provides the valid values you specified.
14. To leave test mode, select File ⇒ Edit interface on the ObjectStudio Designer window (behind the test interface). You return to the ObjectStudio Designer edit mode.

15. When you are satisfied with your changes, select File ⇒ Save. The following window displays the addition of a drop-down list box:

The screenshot shows a window titled "CLSC1" with a menu bar containing "MiscKeys" and "PFkeys". The window contains a form for "Client maintenance" with the following fields:

- Date: CLSC1
- Client maintenance
- Time:
- Client number:
- Client name:
- Client address:
- City:  State:
- Postal code:
- Credit rating:
- Credit limit:
- Branch number: 
  - A1
  - B1
  - C1
- Comment:

The following window shows the replacement of an entry field with a check box:



The screenshot shows a window titled "CLSC1" with a subtitle "Client maintenance". The window contains the following fields and controls:

- Date: CLSC1
- Client number:
- Client name:
- Client address:
- City:
- State:
- Postal code:
- Credit rating:
- Credit Limit > 10000
- Branch number:
- Comment:

## Grouping fields graphically

After loading your gOOi application and moving a gOOi form into ObjectStudio Designer, you can graphically group component fields in a topic box (for example, all address fields in an Address box). To graphically group fields, perform the following steps:

1. Imagine a box that would enclose the fields you want to select as a group. Move your mouse arrow to the upper left corner of this imaginary box.
2. Click and drag toward the lower right until all target fields are selected.

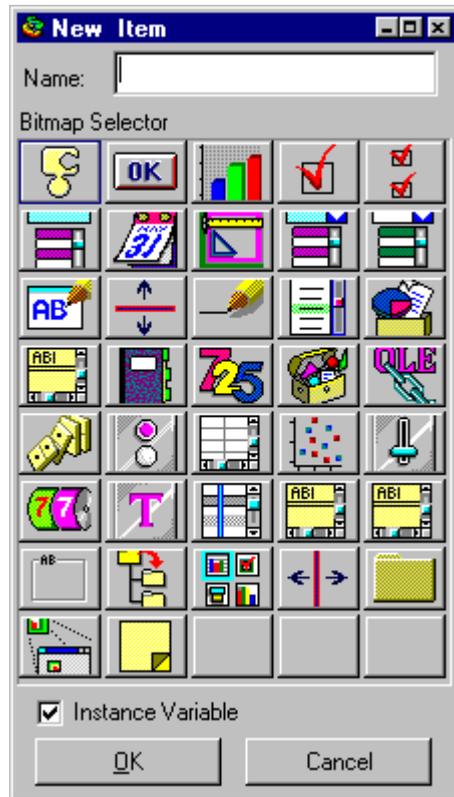


---

An alternative method to steps 1 and 2 is to hold down CTRL and click each field.

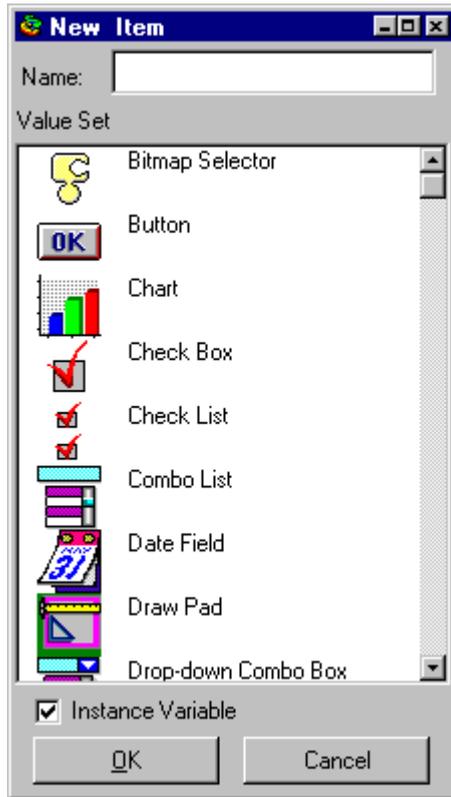
---

3. Create a new item by either pressing Ctrl+N or selecting FormItem ⇒ New Item. The following New Item window displays icons for available new items:





When you click an icon, the type description displays near the top of the window. If you click mouse button 2 in the white space between the icons and select List, the icons are presented alphabetically in a scrollable format with their associated descriptions.



4. Click the Topic Box icon.
5. In the Name field at the top of the window, enter the topic name you want. For example, address components could be grouped under "Address."
6. When you click **OK**, a topic box displays with your specified topic name at the top.

7. Double-click the topic box to receive the Form Topic Box Options window.
8. Click **Format**. The String Display Format window displays.
9. Make your format selections. You can see the effects of your choices in the Example window.
10. When you are satisfied with your format selections, click **OK**. The Form Entry Field Options window displays.
11. Click **Cancel**. You return to your gOOi form.
12. Change the size and shape of the topic box to accommodate the component fields you want to group inside the box. To do this, click and hold the appropriate “handle” along the box’s perimeter with mouse button 2. Drag the box to the size and shape you want. This requires a series of adjustments as you decide how to compose fields inside the box.

13. Move each field you want to group into the topic box. To do this, select each field and move it by clicking and dragging it to the new location. (For details and options for moving fields, see “Moving fields” on page 164.) The following window displays the example gOOi form reworked into two field groupings (Client and Credit):

The screenshot shows a window titled "CLSC1" with a blue header bar. Below the header, the text "Date: CLSC1" is on the left and "Client maintenance" is in the center, with "Time:" on the right. The main content area is divided into two sections, each with a blue title bar:

- Client:** This section contains several input fields: "Client number" (a small text box), "Client name" (a long text box), "Client address" (a long text box), "City" (a text box), "State" (a dropdown menu), and "Postal code" (a text box).
- Credit:** This section contains: "Credit rating" (a dropdown menu), a checkbox labeled "Credit Limit > 10000", "Branch number" (a text box), and "Comment" (a long text box).

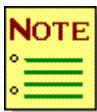
14. When you are satisfied with your changes, select File ⇒ Save.

## Changing the form title and/or background

The default form title is the form name, and the form name is usually the screen ID. You may want to change the form title to give the user a better description of the window contents. You may also want to change the form background to a different color, or to a bitmap. To change either of these items, perform the following steps:

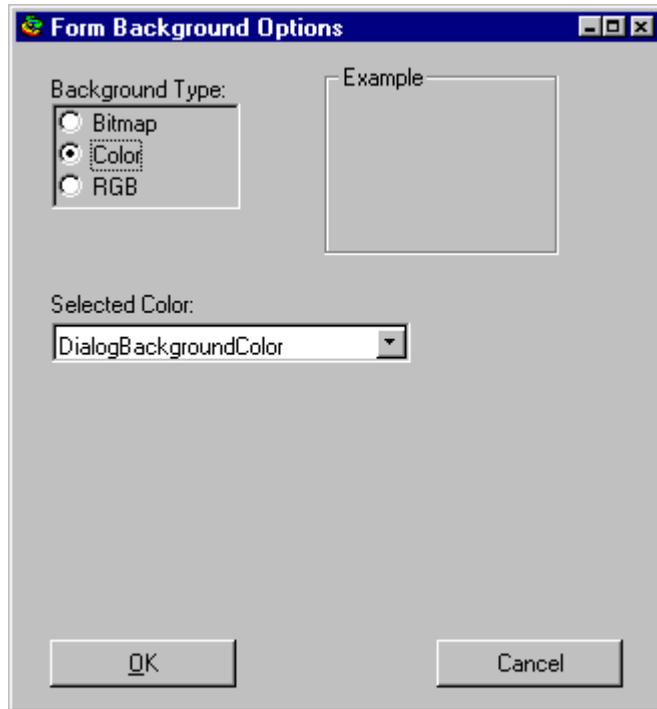
1. From the Designer menu, select Form ⇒ Change Form. The following window is displayed:

2. Overtyping the 'Form Title' to change the title.

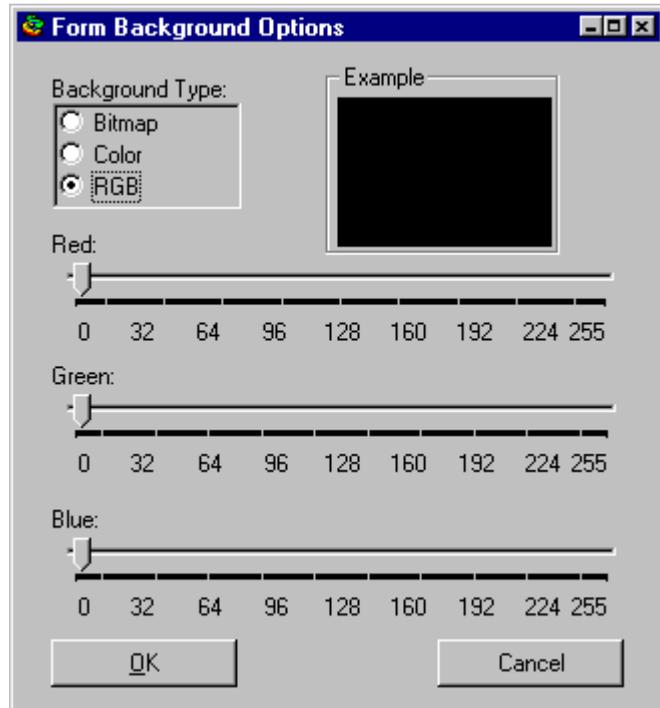


The form may include a static text item that serves as a host screen description. This item becomes redundant after changing the title. If you want to remove it, just click the item and press the Delete key. ObjectStudio will prompt you to confirm the item deletion.

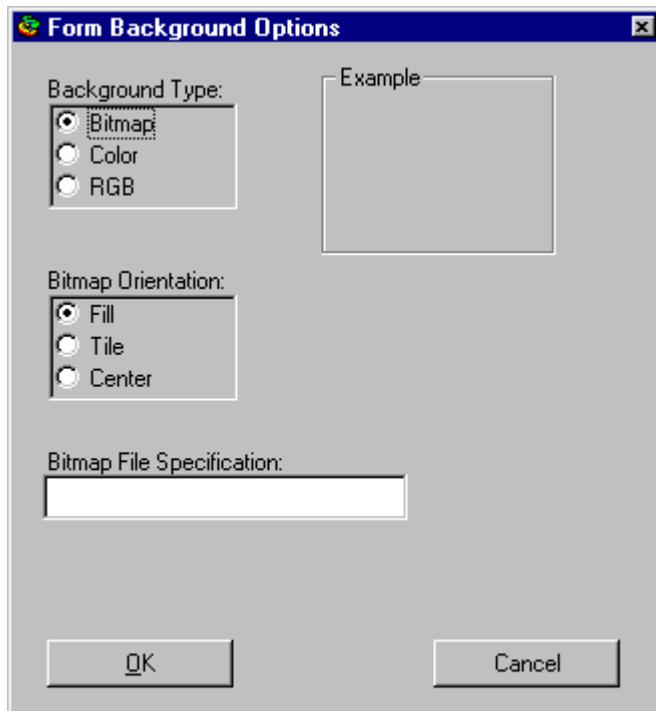
3. To modify the background, click the **Background** button, and the following window displays:



You can choose a physical color such as Red, or a logical color, such as DialogBackgroundColor (which may be different across PCs). You can create a custom color by switching to RGB mode, which presents the following window:



...or you can select Bitmap as the Background Type and designate a bitmap file in the following window:



4. When you are satisfied with your changes, click **OK**, then select File ⇒ Save.

The following window shows an example of a gOOi form after the title and background were changed:

The screenshot shows a window titled "Client Maintenance" with a yellow dotted background. At the top left, it displays "Date: CLSC1" and "Time:". The form is divided into two sections: "Client" and "Credit".

**Client Section:**

- Client number :
- Client name :
- Client address :
- City :  State:
- Postal code :

**Credit Section:**

- Credit rating :   Credit Limit > 10000
- Branch number :
- Comment:

## Additional customizations

There are many other possibilities for form customization beyond those discussed in the previous “Basic customizations” section. This section gives detailed instructions for some other customizations that you may want to implement:

### Using check boxes for Y/N fields

Another common use of a check box is to represent a single character entry field on the host that accepts only Y (yes) or N (no). The same steps listed under “[Changing a field's presentation type](#)” on page 172 can be followed to implement a check box for this purpose, except for the Smalltalk code shown in step 12. Two simple methods need to be added to the Controller:

- ◆ A *get* method to translate the check box setting to the Y or N recognized by the host. The name of a get method is the name of the object, for example *cbWantsMail*
- ◆ A *set* method to translate the Y/N value from the host to the corresponding checked/unchecked state for the check box. The name of a set method is the name of the object with a colon appended, for example *cbWantsMail:*

gOOi includes two methods that are inherited by forms (from GOOIGenericController) to allow easy coding of these get and set methods. For example, a host display has multiple fields of customer data. One of these fields is set to Y or N to indicate if the customer has requested to be included on a newsletter mailing list. After you generate the gOOi form (CustDataController and CustDataHostObject), you replace this entry field (name of *efWantsMail*) with a check box (name of *cbWantsMail*). This check box becomes functional with the following get method:

```
cbWantsMail
```

```
^ self getCheckBoxYorNfor: (cItemDict at: #cbWantsMail).
```

and set method:

```
cbWantsMail: aValue
```

```
self setCheckBoxYorNfor: (cItemDict at: #cbWantsMail) with:
  aValue.
```

Added to `CustDataController`. `cItemDict` in these two methods is a dictionary of controller items that is included in the Controller part of every `gOOi` form. This dictionary is keyed by controller item name, so (*cItemDict* at: *#cbWantsMail*) returns the controller item for the check box. The `get` method must have the same name as the check box, and the `set` method must be named the same as the check box name with a colon appended. *aValue* in the `set` method is a string of 'Y' or 'N' from the host entry field. `getCheckBoxYorNfor:` and `setCheckBoxYorNfor:` are methods inherited from `GOOIGenericController`.

## Using spin buttons for numeric fields

You may want to use a *spin button* to represent a numeric field that has a small range of valid values. For example, a host display with information about children includes a field to display a child's age from 0 to 21. After the `gOOi` form (`ChildDataController` and `ChildDataHostObject`) is generated, you replace this entry field (`efAge`) with a spin button (`sbAge`). For this example, to show an advanced technique, let's make the ages in descending sequence.

When defining the spin button in the ObjectStudio Designer, select the Only Numbers option. Also, enter the valid numbers from largest to smallest (21, 20, etc.). This spin button will become functional after you add to `ChildDataController` the following `get` method:

```
sbAge

^ (sbAge getValueAt: (sbAge getSelection)) asString.
and set method:
sbAge: aValue

sbAge setSelectionTo: (22 - (aValue trimBlanks asInteger)).
```

are added to `ChildDataController`. In the `get` method, `getSelection` returns an integer index of the current value displayed in the spin button. This integer is then passed to the `getValueAt:` method, which returns the value currently displayed at this index. Finally, the `asString` method converts the value to a string object. For example, if the spin button currently displays 21 (the first entry in the list), the index returned from `getSelection` is 1 and the value returned by `getValueAt:` at index 1 is 21. The `asString` method produces a string of '21'.

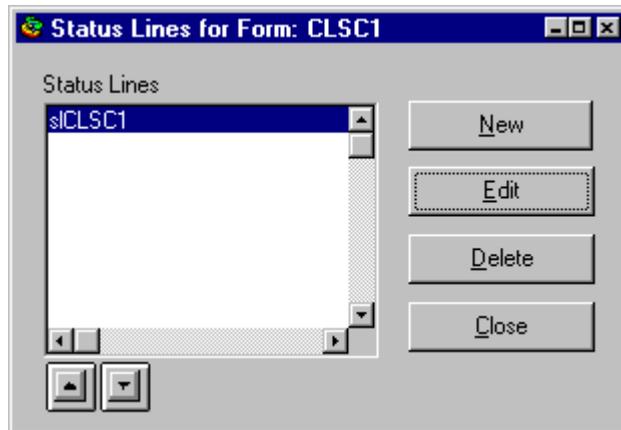
In the `set` method, the spin button must be positioned at the index associated with *aValue*, which is the age on the host display. If *aValue* is '21', this string is converted to an integer and subtracted from 22. The spin button is set to the resulting index of 1, and displays 21.

## Putting data in the status line

Every gOOi form is generated with a status line at the bottom. By default, this status line displays the contents of row 23 from the host screen. You may want the status line to display data from a different host screen location, or to display multiple data items.

For example, a host screen has a message line at row 22, and date and time fields on row 1. You want these 3 items to appear on the status line and not on the gOOi form. In the ObjectStudio Designer, select these items by clicking on one of the 3, then clicking on the other 2 while holding down the Ctrl key. Once all 3 are selected, press the Delete key to remove them from the form.

To create a multiple section status line, select Tools ⇒ Status Line from the Designer menu and the following window is displayed:



Click the **Edit** button to get the Status Line Options window:

**Status Line Options for Form: CLSC1**

Name:

Iop

Section Settings

Width (pixels):

Indicator

- Text
- Caps Lock
- Num Lock
- Scroll Lock
- Override
- Ext
- Record
- Kana

Display Effect

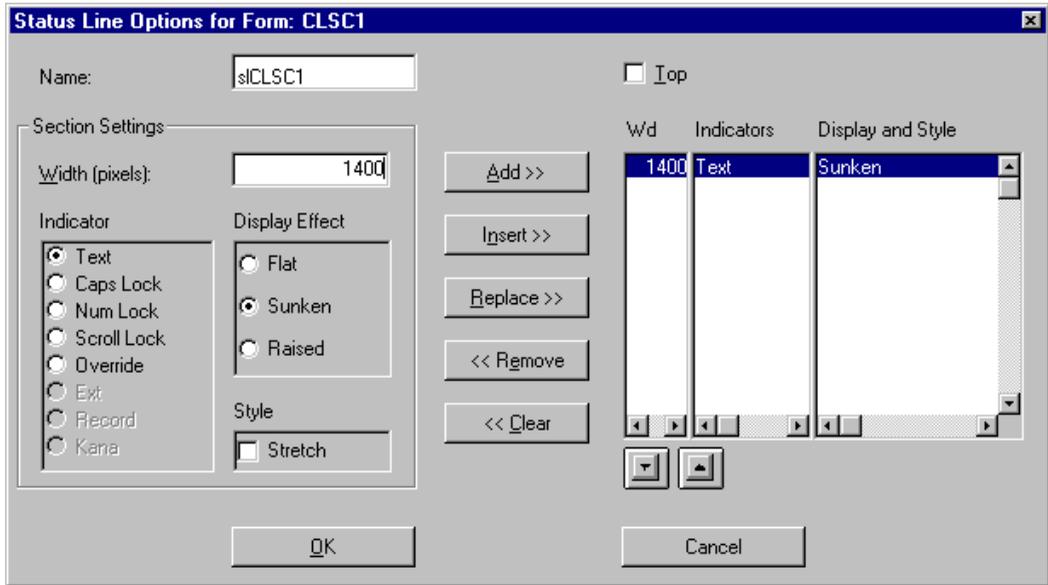
- Flat
- Sunken
- Raised

Style

- Stretch

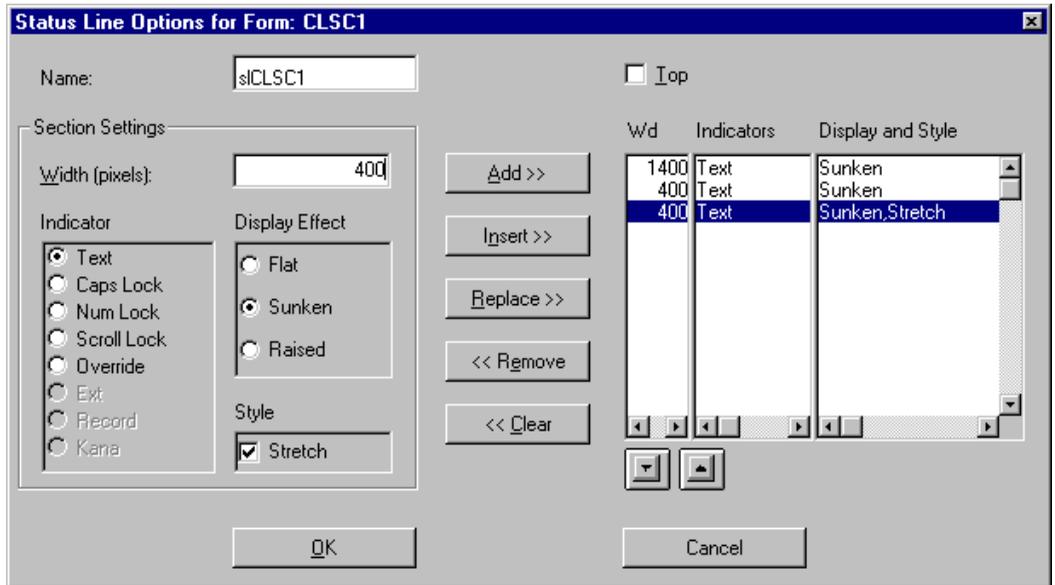
Wd	Indicators	Display and Style
100	Text	Stretch,Flat

For this example, the width will be increased to a larger value to accommodate the message field. In addition, the Display Effect will be changed to Sunken, and the Stretch style turned off. Click on **Replace>>**, and the window appears as follows:



The Stretch style is typically used for the last section of the status line so that the segment extends to the end of line. The width value may require experimentation to achieve the desired size for each section.

Set the width, Display Effect, and Style for a section to display date, then click **Add**. Repeat this process to create a section to display time, using the Stretch style since it is the final section. The following window shows a status line definition with three sections:



Click  to save the status line configuration. The status line on the form is displayed with 3 sections as in the below example:

The screenshot shows a window titled "Client Maintenance" with a standard Windows-style title bar. At the top, there are fields for "Date:" and "Time:". Below that, the text "CLSC1" is displayed. The form is divided into three sections:

- Client**: This section contains five input fields: "Client number:", "Client name:", "Client address:", "City:", and "State:". There is also a "Postal code:" field at the bottom right of this section.
- Credit**: This section contains a "Credit rating:" dropdown menu, a checkbox labeled "Credit Limit > 10000", a "Branch number:" input field, and a "Comment:" text area.
- Third Section**: This section is partially visible at the bottom of the form and contains two empty input fields.

A small amount of Smalltalk code is needed to put the three data items into the status line sections at runtime. This code is placed into *refreshFields*, a method in each controller that is executed each time the gOOi form is displayed. Go to the Class Browser and select the controller for the gOOi form to be updated (e.g. CLSC1Controller). Select the *refreshFields* method, which appears as follows:

```
refreshFields
super refreshFields.
```

By adding four lines of code to this method, message, date, and time will display in the three status line sections:

```
refreshFields

super refreshFields.
slCLSC1 at: 1 put: (self subject efMessage).
slCLSC1 at: 2 put: (self subject efDate).
slCLSC1 at: 3 put: (self subject efTime).
slCLSC1 formItem update refresh.
```

The name of the status line item is the name of the controller prefixed with 'sl' (slCLSC1 in this example). The first three of these additional four lines of code execute at: put: methods to retrieve field data from the host object (CLSC1HostObject) and put this data into the target status line sections. The final line of code updates and refreshes the status line display.

## Using property pages

If the host display is divided into sections, you may want to group the form items of each section of the generated gOOi form into a property page. To work with property pages, it is important to understand that the pages are contained within a property sheet. The following example implements two property pages on a generated gOOi form named Customer:

1. In the ObjectStudio Work Area, click mouse button 2 on the gOOi form, then select Edit to bring up this form in the Designer.
2. From the Designer menu, select Form ⇒ New property page. This brings up a prompt for the property page name. For this example, the property page will contain address data, so you assign it a title of "Address Information". The new property page now appears on your window.
3. Resize the original form so that you can see both this form and the Address Information property page, making sure that Address Information is outside the window borders of the original form.
4. Use lasso or Ctrl/click to select multiple form items to move to Address Information. Drag and drop the selected items onto Address Information.
5. From the Designer menu, again select Form ⇒ New property page. For this example, suppose you assign a title of "Credit Information" to this second property page.

6. Repeat the process described in step 4 to move the relevant form items to Credit Information.
7. From the Designer menu, select Form ⇒ Property sheet, and the Property Sheets dialog displays. Do the following:
  - a. Click the **New** button.
  - b. Enter a name for the property sheet (for example, “Customer Data”) and click **OK**.
  - c. Click the **Edit** button. The Property Sheet Options dialog displays.
  - d. Turn off the Modal check box and select the Modeless Buttons check box.
  - e. Add the two pages listed in the Available Pages list box to the Page Definition list box using the **Add>>** button.
  - f. Click **OK**, and the two property pages have been associated with the Customer property sheet.
8. Select File ⇒ Test interface from the Designer menu to see how the revised gOOi form looks at this point. The property pages are not displayed; you only see the original gOOi form, which is now empty because all the form items were moved to the property pages. The next step involves adding code that displays these property pages and hides the empty original form.
9. To add code that displays these property pages and hides the empty original form:
  - a. From the Designer menu, select Controller ⇒ Events, and the Event Editor dialog displays.
  - b. Under the Sender Object list box, select Customer (form), the original form object generated by gOOi.
  - c. Under the Event names list box, select activated.
  - d. Under the Specify event receiver list box, select Customer, the controller object for this gOOi form.

- e. A new method is needed for this event, so click the **Method** button and the Method Editor dialog displays for the `formCustomerActivated` method. Add the following method code:

```
| propertySheet |

propertySheet := (formDict at: #'Customer Data').
propertySheet isOpen ifTrue: [
    propertySheet activate.
] ifFalse: [
    propertySheet open.
].
self mainForm isVisible ifTrue: [
    self mainForm hide.
].
```

and click the **Save** button to exit the Method Editor.

- f. Click the **<<Add<<** button in the Event Editor to hook the activated event to this new `formCustomerActivated` method.
- g. Click the **OK** button to exit the Event Editor.

The first part of this method checks if the Customer Data property sheet is already open. If it is, the activate method ensures that the active property page is displayed. Otherwise, the property sheet is opened. Finally, the end of the method makes sure that the empty original form is not displayed.

All that remains is to implement the Cancel, Apply, and OK buttons that are automatically provided with every property sheet. Each button has a corresponding host behavior in this example:

- The Cancel button is equivalent to a PF3 key (exit without any change).
- The Apply button is the same as a PF5 key (update but don't exit).
- The OK button is the same as a PF5 key followed by a PF3 key (update and exit).

A simple method is needed to translate the clicked event for each button into the appropriate key(s).

10. To implement the Cancel button:

- a. Select Controller ⇒ Events from the Designer menu, and the Event Editor dialog displays.
- b. Under the Sender Object list box, select Customer Data(form), the property sheet object.
- c. Under the Event names list box, select onCancel.
- d. Under the Specify event receiver list box, select Customer, the controller object for this gOOi form.
- e. A new method is needed for this event, so click the **Method** button and the Method Editor dialog displays for the formCustomeronCancel method. Enter the following method code:

```
self PF3KeyPressed.
```

and click the **Save** button to exit the Method Editor.

- f. Click the **<<Add<<** button in the Event Editor to hook the onCancel event to this new formCustomeronCancel method. Do not yet exit from the Event Editor. The PF3KeyPressed method is inherited from the GOOIGenericController class.

11. Customer Data(form) is still selected as the Sender object. To implement the Apply button:

- a. Select onApply as the Event name. Customer is still selected as the event receiver.
- b. A new method is also needed for this event, so click the **Method** button and the Method Editor dialog displays for the formCustomeronApply method. Enter the following method code:

```
self PF5KeyPressed.
```

and click the **Save** button to exit the Method Editor.

- c. Click the **<<Add<<** button in the Event Editor to hook the onCancel event to this new formCustomeronApply method. Do not yet exit from the Event Editor. The PF5KeyPressed method is inherited from the GOOIGenericController class.

12. Customer Data(form) is still selected as the Sender object. To implement the **OK** button:
- Select onOK as the Event name. Customer is still selected as the event receiver.
  - A new method is also needed for this event, so click the **Method** button and the Method Editor dialog displays for the formCustomeronOK method. Enter the following method code:
 

```
GOOHostMonitor sleep.
self PF5KeyPressed.
self PF3KeyPressed.
GOOHostMonitor wakeUp.
```

 and click the **Save** button to exit the Method Editor.
  - Click the **<<Add<<** button in the Event Editor to hook the onOK event to this new formCustomeronOK method. This method uses the sleep and wakeup methods of host navigation to allow multiple host interactions to process without redisplay of this Customer gOOi form. Do not yet exit from the Event Editor.
13. The **Apply** button is initially disabled. Two methods are needed (one for each property page) to activate this button when data is entered into either property page. Perform the following steps to enable the **Apply** button:
- Select one of the entry fields on the Address Information property page as the Sender object.
  - Select wmCharPressed: as the Event name. Customer is still selected as the event receiver.
  - A new method is needed, so click the **Method** button and the Method Editor dialog displays for the formCustomerwmCharPressed: method. Change the method name as follows:
 

```
activateAddressApplyForWmCharPressed: arg
```

 then add the following method code:
 

```
(formDict at: #'Address Information') setModified: true.
```

- d. Click the **Save** button to exit the Method Editor. This code enables the Apply button for the Address Information property page. Click the **Method** button again and change the method name as follows:

```
activateCreditApplyForWmCharPressed: arg
```

then add the following method code:

```
(formDict at: #Credit Information') setModified: true.
```

- e. Click the **Save** button to exit the Method Editor. This code enables the Apply button for the Credit Information property page.
14. Now, every entry field must have its wmCharPressed: event hooked to either of these two methods, depending upon on which property page it was placed. For example, efCity is an entry field on the Address Information property page, and efCreditLimit is an entry field on the Credit Information property page. Perform the following steps:
    - a. Select efCity as the Sender object, wmCharPressed: as the event name, Customer as the receiver, and *activateAddressApplyForWmCharPressed: arg* as the message.
    - b. Click the **<<Add<<** button.
    - c. Select a different entry field on the Address Information property page and click **<<Add<<** until every entry field assigned to the Address Information property page has the wmCharPressed: event hooked.
    - d. Select efCreditLimit as the Sender object, wmCharPressed: as the event name, Customer as the receiver, and *activateCreditApplyForWmCharPressed: arg* as the message.
    - e. Click the **<<Add<<** button. Select a different entry field on the Credit Information property page and click **<<Add<<** until every entry field assigned to the Credit Information property page has the wmCharPressed: event hooked.
    - f. Click the **OK** button to exit from the Event Editor.
    - g. Save this form and exit from the Designer.

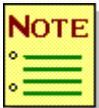
## Example of an event-driven customization

The button in this example allows the user to select some fields with the mouse and automatically paste the selected field values into an Excel spreadsheet. The following examples are described:

- ◆ The first version shows how to copy a Cincom-provided Smalltalk method, and how you can easily apply existing methods to gOOi forms.
- ◆ The second version shows how to create the same button if you create the Smalltalk method yourself. This version illustrates how a little knowledge of Smalltalk can vastly widen the possibilities for gOOi extensions.



These examples are provided to show the steps necessary to complete the task, and to show what can be done with a small amount of Smalltalk code. The end result is testable only if it is applied to the appropriate gOOi form and an English language version of Excel is in the path coded into the method (`\msoffice\excel\excel.exe`).



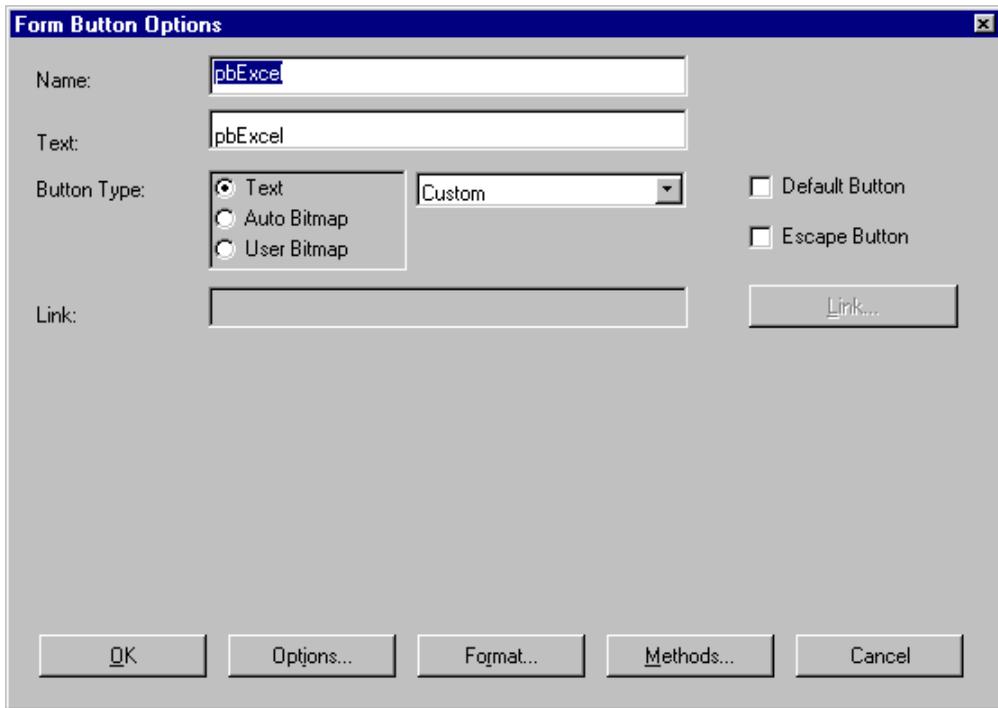
There are several important differences between the following examples and the Excel Wizard. The Excel Wizard automatically generates the Smalltalk code needed for integration. In addition, the wizard uses OLE for the data transfer, whereas this example uses DDE.

## Creating an event-driven button with an existing Smalltalk method

To add an event-driven **Excel** button using an existing Smalltalk method, perform the following steps (after loading your gOOi application and moving the appropriate form into ObjectStudio Designer):

1. Create a new button called pbExcel. To do this, either press Ctrl+N or select Formitem ⇒ New Item. The New Item window displays, showing icons for available new items. Click the Button icon.
2. Enter pbExcel in the Name field and click **OK**. You return to your gOOi form, which displays your new pbExcel button.

3. Double-click the pbExcel button. The following window displays:

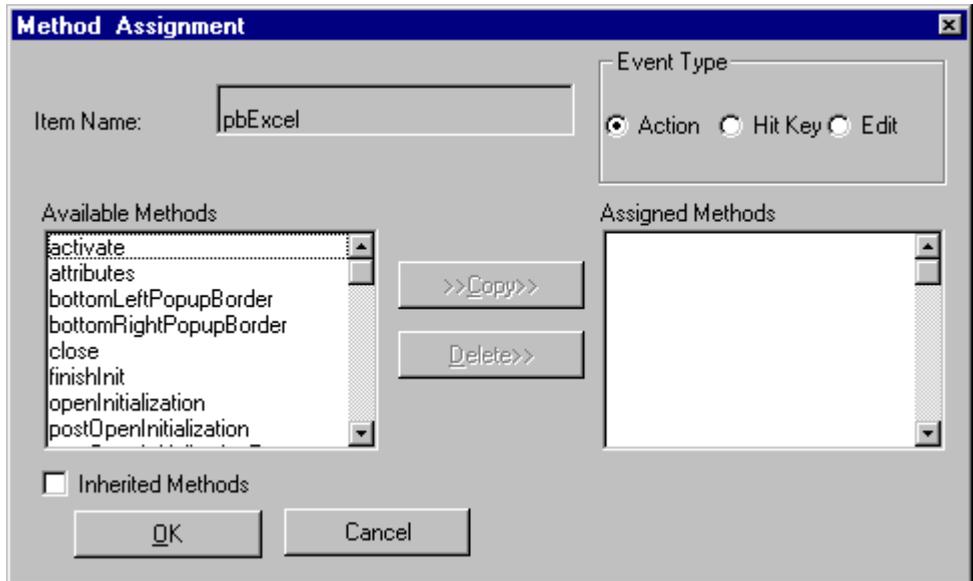


The image shows a dialog box titled "Form Button Options" with a standard Windows window border. The dialog contains several fields and controls:

- Name:** A text field containing "pbExcel".
- Text:** A text field containing "pbExcel".
- Button Type:** A group box containing three radio buttons: "Text" (selected), "Auto Bitmap", and "User Bitmap". To the right of this group is a dropdown menu currently set to "Custom".
- Default Button:** An unchecked checkbox.
- Escape Button:** An unchecked checkbox.
- Link:** An empty text field.
- Link...:** A button next to the Link field.
- Buttons:** At the bottom of the dialog are five buttons: "OK", "Options...", "Format...", "Methods...", and "Cancel".

4. In the Text field, delete the pb from pbExcel. The Text field represents the external button name. (The Name field is the internal name and must remain pbExcel.)

- Click **Methods**. The following window displays:



- Select the Inherited Methods check box at the lower-left of the window. This lists all the methods inherited by the gOOi class (form) that you are customizing.
- Select the Cincom-provided pbExcelClicked method from the Available Methods list.
- Click **>>Copy>>**. The pbExcelClicked method displays in the Assigned Methods box to the right.
- Click **OK**. You return to the Form Button Options window.
- Click **OK**. You return to your gOOi form.
- Select File ⇒ Save.
- Select File ⇒ Exit.

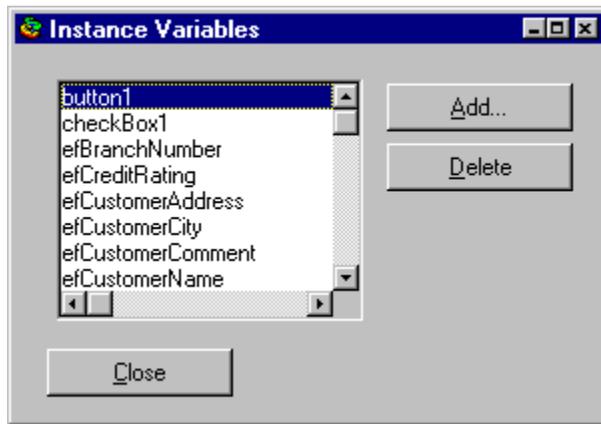
This sample Excel button is testable only if it is applied to the appropriate gOOi form, and an English language version of Excel is in the path coded in the method (`\\msoffice\excel\excel.exe`).

To use the button, you need to make a host connection and run the host application. At the appropriate window, click **Excel**. Select several data fields by lassoing them with the mouse. When you release the mouse button, the field values are inserted into Excel.

## Creating an event-driven button with Smalltalk

To create your own method for the example **Excel** button, perform the following steps (after loading your gOOi application and moving the appropriate form into ObjectStudio Designer):

1. Select **Tools** ⇒ **Variables**. The following window displays the variables in your current gOOi class (form):



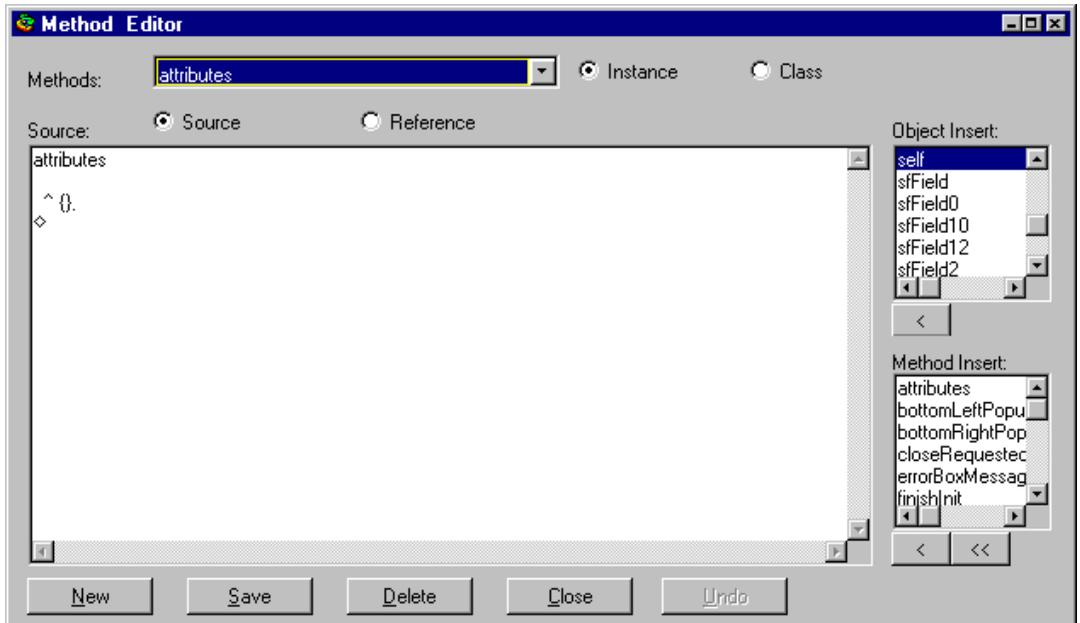
2. Click **Add**. You are prompted to provide a new instance variable name. Enter the following:

dde

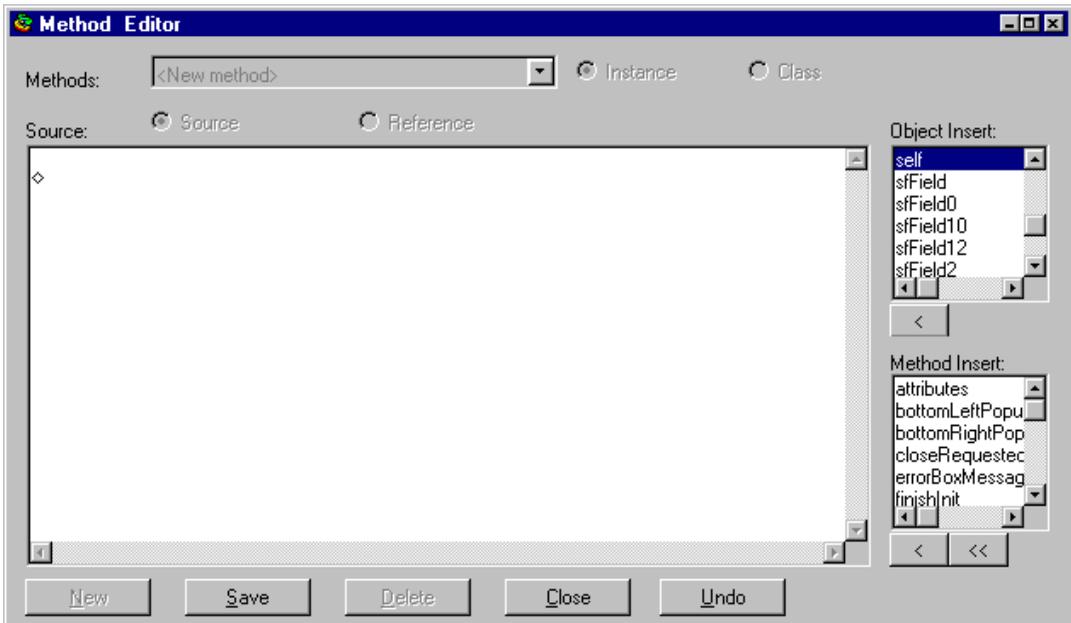
This variable holds the DDE conversation.

3. Click **OK**.
4. On the Instance Variables window, click **Close**.

5. Select Tools ⇒ Methods. The following window displays:



6. Click **New**. The Source field clears for source code entry as follows:



7. In the Source box, enter the following:

```
clipboardFilled
dde isNil ifFalse: [
    dde executeCommand: '[Paste]'.
].
```

This new method automatically pastes values held in the clipboard to Excel. These values go in the clipboard when you lasso them with the mouse.

8. Click **Save**. You are asked if you are creating a subclass method. Click **OK**.
9. Click **New**. The window clears for source code entry.

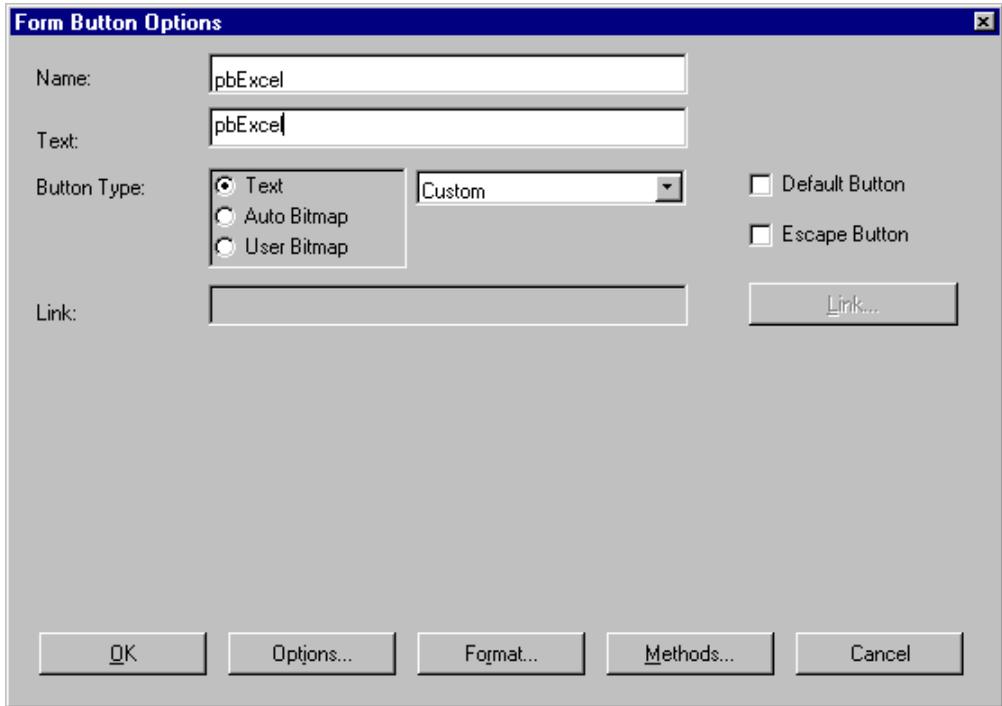
10. In the Source box, enter the following:

```
pbExcelClicked
dde isNil ifTrue: [
  'excel' startSessionProgram:
  '\msooffice\excel\excel.exe' inputs: ''.
  dde := DDEClientSession name: #DEMO
  application: 'Excel' topic: 'Sheet1' item: ''.
  dde initiate.
] ifFalse: [
  dde executeCommand: '[Close]'.
  dde terminate.
  dde := nil.
].
```

This code starts Excel when you click the button the first time; it closes Excel when you click it a second time.

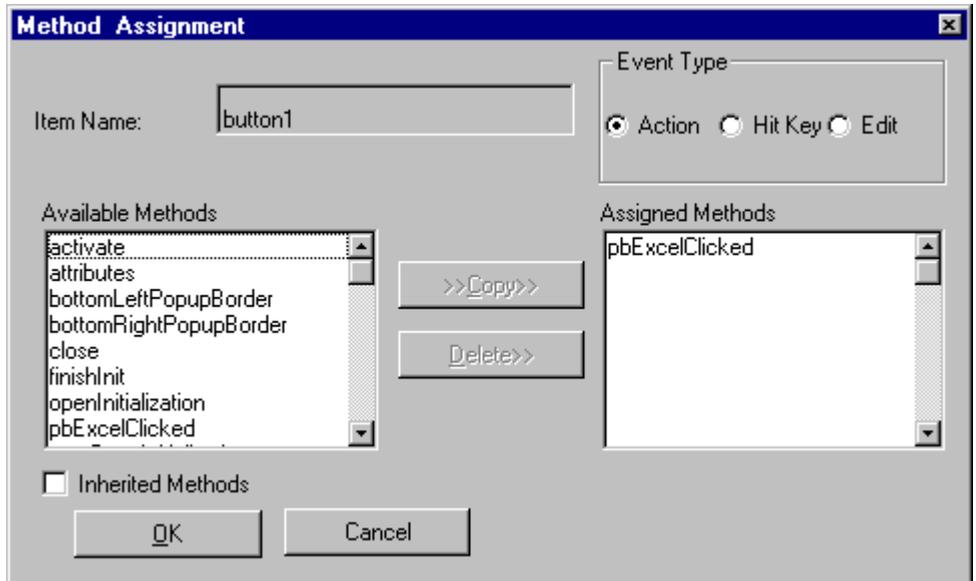
11. Click **Save**. You are asked if you are creating a subclass method. Click **OK**.
12. Click **Close**. You return to your gOOi form.
13. Create a new button called pbExcel. To do this, either press Ctrl+N, or select Formitem ⇒ New Item. The New Item window displays showing icons for available new items. Click the Button icon.
14. Type pbExcel in the Name field and click **OK**. You return to the gOOi form, which displays your new pbExcel button.

15. Double-click the pbExcel button. The following Form Button Options window displays:



16. In the Text field, delete the pb from pbExcel. The Text field represents the external button name. (The Name field is the internal name and must stay pbExcel.)
17. Click **Methods** and the Method Assignment window displays.
18. Select pbExcelClicked in the Available Methods box. You just created this method.

19. Click **>>Copy>>**. The pbExcelClicked method displays in the Assigned Methods box at the right:



20. Click **OK**. You return to the Form Button Options window.
21. Click **OK**. You return to your gOOi form.
22. Select File ⇒ Save.
23. Select File ⇒ Exit.

This sample **Excel** button is testable only if it is applied to the appropriate gOOi form, and an English language version of Excel is in the path coded in the method (`\\msoffice\excel\excel.exe`).

To use the button, you need to make a host connection and run the host application. At the appropriate window, click **Excel**. Select several data fields by lassoing them with the mouse. When you release the mouse button, the field values are inserted into Excel.



# 8

## Host Navigation

### Overview of Host Navigation

These sections describe how you can take control from the automation of gOOi run-time and customize your application even further. With Host Navigation methods and attributes, it is possible to use gOOi like an API and do all of your graphical presentation with any style of ObjectStudio controller.

Host Navigation methods lift any restrictions that the generality of gOOi generation imposes on graphical presentation, and let you take advantage of the flexibility of ObjectStudio.

Here are some examples of what you can do with Host Navigation:

- ◆ Design an MDI (Multiple Document Interface) controller for presentation of your host data
- ◆ Gather host data from multiple host screens for presentation in one gOOi form
- ◆ Automatically run cumbersome log-on procedures that require the same user and password to be entered multiple times

---

## Host Navigation methods and attributes

gOOi provides a set of methods (Smalltalk code) and attributes (variables) that you can use to navigate a host connection (that is, session). For example, if you want to automate the CICS logon process so that you don't have to repetitively enter keyboard responses to a series of logon screens, you can add code to the Just-In-Time custom controller (GOOIJITCustomController). The following code illustrates the methods and attributes used for this purpose. The line numbers shown at the left of the code are not valid in Smalltalk, but were added to this sample for use in the subsequent explanation.

```
1  startUp
2  | userID password userTran cicsTran acceptMsg welcomeMsg |
3  userID := 'USER'.
4  password := 'PASSWORD'.
5  userTran := 'TRAN'.
6  cicsTran := 'CESN'.
7  acceptMsg := 'been accepted'.
8  welcomeMsg := 'WELCOME TO CICS 4.1.0'.
9  (self checkForString: acceptMsg at: {3 42}) ifTrue: [
10     self mainForm hide.
11     GOOHostMonitor sleep.
12     (self waitForString: welcomeMsg at: {1 2}) ifTrue: [
13         self pressClear.
14         self enterString: cicsTran at: 1.
15         self pressEnter.
16         self enterString: userID at: 1.
17         self enterString: password at: 2.
18         self pressEnter.
19         self enterString: userTran at: 1.
20         self pressEnter.
21     ].
22     GOOHostMonitor wakeUp.
23 ].
```

The following table explains each line of the sample code.

Stmt	Code	Explanation
1	StartUp	The method name is startUp. This name follows the Smalltalk convention of using lower case for the first word, and beginning each subsequent word (Up) with an upper case letter. A stub startUp method is included in GOOIJITCustomerController for convenience. All navigation for JIT should be placed in GOOIJITCustomController.
2	userID password userTran cicsTran acceptMsg welcomeMsg	Six local variables are defined for data that might change between users or systems: userID, password, userTran, cicsTran, acceptMsg, and welcomeMsg.
3	userID := 'USER'	The userID local variable is set to the CICS user.
4	Password = 'PASSWORD'	The password local variable is set to the CICS user's password.
5	userTran := 'TRAN'	The userTran local variable is set to the transaction that this user wants to start.
6	cicsTran := 'CESN'.	The cicsTran local variable is set to the CICS sign-on transaction.
7	acceptMsg := 'been accepted'.	The acceptMsg local variable is set to the portion of the CICS acceptance message for which this method searches.
8	welcomeMsg := 'WELCOME TO CICS 4.1.0'.	The welcomeMsg local variable is set to the CICS welcome message.
9	(self checkForString: acceptMsg at: {3 42}) ifTrue: [	The first screen that appears during CICS logon has the message, "Your request has been accepted". The method looks for the "been accepted" part of this message at row 3, column 42. The method proceeds to the next line if this message is found, and continues at line 23 otherwise. The self object that receives the checkForString:at: message is the current class, GOIJITCustomController. This screen does not need a user response.

Stmt	Code	Explanation
10	self mainForm hide.	The current Just-In-Time form is hidden so that it does not appear while the navigation code is executing.
11	GOOHostMonitor sleep.	Host monitoring is turned off so that the default behavior of JIT (present a window with the user ID and password) will not occur. GOOHostMonitor is the name of the gOOi object that watches for host changes, and sleep is a method in this class which renders this object inactive.
12	(self waitForString: welcomeMsg at: {1 2}) ifTrue: [	The second screen that appears during CICS logon has the message, "WELCOME TO CICS 4.1.0". The method looks for this message at row 1, column 2. The method proceeds to the next line if this message is found, and continues at line 21 otherwise.
13	self pressClear.	The welcome message was found. Press the CLEAR key in response so the logon process can continue.
14	self enterString: cicsTran at: 1	The third screen that appears has a single entry field. The method places CESN into this field.
15	self pressEnter.	Press the ENTER key to start CESN.
16	self enterString: userID at: 1.	The first entry field CESN is the user ID. MOVE USER into this field.
17	self enterString: password at: 2.	The second entry field for CESN is the password. Move PASSWORD into this field.
18	self pressEnter.	Press the ENTER key to complete CESN.
19	self enterString: userTran at: 1.	The fourth screen that appears has an entry field for the user transaction. The method places TRAN into this field.
20	self pressEnter.	Press the ENTER key to start TRAN.
21	]	This closing bracket terminates the ifTrue block of code that was begun on line 12.
22	GOOHostMonitor wakeUp.	Host monitoring is turned back on so that gOOi will watch for host session changes and present a gOOi form or JIT window for each host screen.
23	]	This closing bracket terminates the ifTrue block of code that was begun on line 9.

The preceding example reflects the object-oriented structure of Smalltalk. As demonstrated in the example, Smalltalk code is built with objects (for example, self) and messages to objects (for example, pressEnter).

Even the local variables are instantiated as objects. For example, userID is an instance of the String object for which String methods can be used. The following sections detail messages (methods) that are available to gOOi objects in order to implement host navigation with gOOi.



---

A general rule to follow when writing code for host navigation logic is to put the code with the object it is associated with. Do not place code intended to navigate multiple objects within one object. For instance, do not put code into GOOIJITCustomController that tries to bypass both a host screen for which there is no generated gOOi form, and also a generated gOOi form.

---

The following table is a summary of the objects, attributes (variables), and methods (messages) available with gOOi Host Navigation:

Object	Attributes	Methods
gOOi form controller	currentSession	ENTERKeyPressed,.... hostObject pressCancel pressEnter pressKey: pressPFn refreshHostFor: searchForPopup subject
gOOi form host object	currentSession	refreshFields refreshHostFor:
GOOIHSTMonitor	ealSessions	setProfile: sleep sleeping wakeup
gOOi session	screenIDValue searchRowColumn searchString	flushGooiBufferToPs getStringAtRow:column:le ngth: gooiBuffering gooiBuffering: search searchAll searchNext wait

Object	Attributes	Methods
GOOIJITController	currentSession entryFields	checkForString: checkForString:at: enterString:at: functionKeyPressed: pressClear pressEnter refreshFields refreshHostFor: waitForString: waitForString:at:
GOOIJITCustomController		screenChangedOn: startup
GOOIVTEmulatorCustomController		checkForString: checkForString:at: enterString: eraseToEndOfField getStringAtRow:column:le ngth: pressEnter pressTab screenChangedOn: startUp waitForString: waitForString:at: windowDown windowHome windowLeft windowRight windowUp

Subsequent sections of this chapter provide details about how to use these objects, attributes, and messages for coding host navigation.

---

## **Object: Controllers generated by the Application Generator**

Each gOOi form consists of a controller for the visual display and a host object for the host screen description. The object for the methods and attributes in this section is the controller. Methods and controllers for this object are described in the following pages.

Note that there are two other objects of concern when coding host navigation for generated controller objects:

- ◆ The corresponding HostObject that was also generated by the Application Generator
  
- ◆ The host session

The data flow is through the Host Object. If you send a transmission key (for example, ENTER) to the host via host navigation, the data in the Host Object is transferred to the host session. You must ensure that the Host Object data is synchronized with the controller data to guarantee correct results.

---

## Methods

---

**ENTERKeyPressed (and any method with KeyPressed as the suffix...for example, PF1KeyPressed, PA3KeyPressed, ...)**



---

These ...KeyPressed methods are only available when running with an IBM mainframe host.

---

**Description:** Sends the appropriate key to the Host.

**Assumptions:** A session with the host exists.

**Return value:** The receiver.

**Receiver modified:** No.

**Example:**

```
self ENTERKeyPressed.
```

where self is the current controller.

---

### hostObject

**Description:** Instantiates the appropriate GOOIHostObject for self and establishes the subject for self. Should be used when navigating to gOOi Controllers not opened by gOOi.

**Assumptions:** A session with the host exists.

**Return value:** A subclass of GOOIHostObject (usually generated by gOOi) or nil.

**Receiver modified:** No.

**Example:**

```
self hostObject.
```

where self is the current controller.

## **pressCancel**

**Description:** Sends the CLEAR key to the Host.

**Assumptions:** A session with the host exists.

**Return value:** None.

**Receiver modified:** No.

**Example:**

```
self pressClear.
```

where self is the current controller.

---

## **pressEnter**

**Description:** Sends the ENTER key to the Host.

**Assumptions:** A session with the host exists.

**Return value:** None.

**Receiver modified:** No.

**Example:**

```
self pressEnter.
```

where self is the current controller.

---

**pressKey: aString**

---

This method is only available when using gOOi with an OpenVMS/UNIX host via the direct TCP/IP connection.

---

**Description:** Sends the specified key to the host as a String object. aString is a character, such as 'h', or a number, such as '1', from the keyboard, with the exception of the F1 through F4 keys, which are specified as 'F1', 'F2', 'F3', and 'F4'.

**Assumptions:** A session with the host exists.

**Return value:** The receiver.

**Receiver modified:** No.

**Example:**

```
pbHelpClicked
self pressKey: 'F1'.
self pressKey: 'h'.
```

where pbHelpClicked is the method name, and self is the current controller. In this example, the user clicks the Help button, and the F1 and h keys are sent to the host via pressKey:

---

**pressPF1 ... pressPF24**

**Description:** Sends the PF<sub>n</sub> key to the Host in an IBM mainframe environment. In an OpenVMS/UNIX environment, a host sequence is sent to the host of either the F1 or F2 key followed by one or more numeric keys. For example, the pressPF10 method sends an OpenVMS/UNIX host a PF2 key, then a 1 key, then a 0 key.

**Assumptions:** A session with the host exists.

**Return value:** None.

**Receiver modified:** No.

**Example:**

```
self pressPF4.
```

where self is the current controller.

## refreshHostFor:aFieldName

**Description:** Synchronize the host session for a given fieldName.

**Assumptions:** aFieldName is an object of class Symbol. A session with the host exists. aFieldName is included in the Host Object (it may have been deleted from the controller).

**Return value:** The receiver.

**Receiver modified:** No.

**Notes:** Use the *enterString: aString for: aFieldName* method instead if the controller data for aFieldName has changed and needs to be synchronized with the Host Object.

**Example:**

```
self refreshHostFor: #efCommand.
```

where self is the current controller, and efCommand is an instance variable in the Host Object.

---

## searchForPopup

**Description:** Search the controller for an associated popup. Should only be used after gOOi has opened the Controller.

**Assumptions:** A session with the host exists.

**Return value:** A popup controller if one is found, else nil.

**Receiver modified:** Yes.

**Example:**

```
popup := self searchForPopup.  
popup notNil ifTrue: [  
    ...  
].
```

where self is the current controller, and popup is defined as a local variable in the method.

---

**subject**

**Description:** Identifies the Host Object part of a gOOi form corresponding to a controller. Should only be used after gOOi has opened the Controller or after the hostObject method has been used.

**Assumptions:** A session with the host exists.

**Return value:** A subclass of GOOHostObject (usually generated by gOOi) or nil.

**Receiver modified:** No.

**Example:**

```
self subject efCommand: aCommand.
```

where self is the current controller, and efCommand: is a set method in the Host Object for the efCommand instance variable.

---

**Attributes**

---

**currentSession**

**Description:** Identifies the connection that gOOi has established with the host.

---

## Object: Host Objects generated by the Application Generator

Each gOOi form consists of a controller for the visual display and a host object for the host screen description. The object for the methods and attributes in this section is the host object. Methods and controllers for this object are described in the following pages. Host objects correspond one-for-one with the generated controller objects. Host object classes should only be generated by the gOOi Application Generator, and should not be customized.

---

### Methods

---

#### refreshFields

**Description:** Synchronize attributes of the HostObject with the gOOi session.

**Assumptions:** A session with the host exists.

**Return value:** The receiver.

**Receiver modified:** Yes.

**Example:**

```
self subject refreshFields.
```

where self is the current controller, and subject is the corresponding Host Object.

---

#### refreshHostFor:aFieldName

**Description:** Synchronize the host session for a given fieldName.

**Assumptions:** aFieldName is an object of class Symbol. A session with the host exists.

**Return value:** The receiver.

**Receiver modified:** No.

**Example:**

```
self subject refreshHostFor: #efOption.
```

where self is the current controller, subject is the corresponding Host Object, and efOption is an instance variable in the Host Object.

---

## Attributes

---

### currentSession

**Description:** Identifies the connection that gOOi has established with the host.

---

## Object: GOOHostMonitorController

The class for this object is the core of the gOOi system. The GOOHostMonitor manages all host application navigation automatically (unless you are using the Host Navigation features described here). There is always an instance of this class when gOOi is running, and the instance name is 'GOOHostMonitor'. Methods and controllers for this object are described in the following pages.

---

## Methods

---

### setProfile:aProfileName

**Description:** Set the active screen ID profile to the specified profile name.

**Assumptions:** The specified profile name is a valid screen ID profile.

**Return value:** Object of class Boolean: true if the active profile is set to the specified profile name, and false if the specified profile name is not found.

**Receiver modified:** Yes.

**Example:**

```
(GOOHostMonitor setProfile: 'MANTIS') ifFalse: [  
    self error: 'Severe error, MANTIS Profile missing!'.  
].
```

where self is the controller with this method code that calls setProfile:, and error: is a method inherited from the Object class.

## sleep

**Description:** Disable gOOi automatic controller navigation based on host session content.

**Assumptions:** None.

**Return value:** The receiver.

**Receiver modified:** Yes.

**Example:**

```
GOOHostMonitor sleep.
```

---

## sleeping

**Description:** Determine current mode of GOOHostMonitor.

**Assumptions:** None.

**Return value:** An object of class Boolean.

**Receiver modified:** No.

**Example:**

```
GOOHostMonitor sleeping ifTrue: [GOOHostMonitor wakeUp].
```

The *sleeping* method returns either true or false. In this example, the *wakeUp* method is called when true is returned.

---

## wakeUp

**Description:** Activate gOOi automatic controller navigation based on host session content.

**Assumptions:** None.

**Return value:** The receiver.

**Receiver modified:** Yes.

**Example:**

```
GOOHostMonitor wakeUp.
```

# Attributes

---

## realSessions

**Description:** An IdentityDictionary of sessions with the host, identified by EHLLAPI short name as the Dictionary key. For TCP/IP connected session, the Dictionary key is always #A. The value associated with the Dictionary keys will be a gOOi Session object.

## Object: gOOi Session

This object functions as the connection between gOOi and the host system. All communication between gOOi and the host occurs with this class/object. Methods and controllers for this object are described in the following pages.

---

## Methods

---

### flushGooiBufferToPs

**Description:** Updates the host with the contents of the internal gOOi buffer. This is used after the gooiBuffering option has been turned on, then host updates executed via the refreshHostFor: and/or refreshHostFor:item: methods.

**Assumptions:** None.

**Return value:** The receiver.

**Receiver modified:** No.



---

This option applies only to the gOOi TCP/IP host connection. It has no effect when using gOOi with an emulator

---

### Example:

```
currentSession flushGooiBufferToPs.
```

where currentSession is an inherited instance variable (from GOOIHostController) that addresses the session object.

---

**getStringAtRow:column:length:**

**Description:** Retrieves a string at the specified row and column for the specified length from the object's copy of the host data

**Assumptions:** None.

**Return value:** A String object.

**Receiver modified:** No.

**Example:**

```
str := currentSession getStringAtRow: 4 column: 1 length: 8.
```

where `currentSession` is an inherited instance variable (from `GOOIHostController`), and `str` is defined as a local variable in the method.

---

**gooiBuffering**

**Description:** Identifies whether the `gooiBuffering` option is on or off.

**Assumptions:** None.

**Return value:** Object of class `Boolean`: true if the option is on, and false if the option is off.

**Receiver modified:** No.




---

This option applies only to the `gOOi` TCP/IP host connection. It has no effect when using `gOOi` with an emulator.

---

**Example:**

```
(currentSession gooiBuffering) ifTrue: [
    count := 0.
].
```

where `currentSession` is an inherited instance variable (from `GOOIHostController`) that addresses the session object, and `count` is a local variable.

## gooiBuffering: aBoolean

**Description:** If gooiBuffering is turned on, host updates are queued to the internal gOOi buffer. This option improves performance when executing a large number of host updates before sending a transmit key to the host. These host updates are made via the refreshHostFor: and/or refreshHostFor:item: methods.

**Assumptions:** aBoolean is an object of class Boolean, either true (on) or false (off).

**Return value:** The receiver.

**Receiver modified:** No.



---

This option applies only to the gOOi TCP/IP host connection. It has no effect when using gOOi with an emulator.

---

### Example:

```
currentSession gooiBuffering: true.
```

where currentSession is an inherited instance variable (from GOOIHController) that addresses the session object.

---

## search

**Description:** Search the object's copy of the host data for a target string. This string and the starting row/column for the search are specified in object attributes.

**Assumptions:** The searchString attribute has been set to the target string (for example, "INV001"). The searchRowColumn attribute is an array with the starting row and column (for example, {2 5}).

**Return value:** Array with row/column where target string was found, or empty array if target string was not found.

**Receiver modified:** No.

### Example:

```
rowCol := currentSession search.  
rowCol isEmpty ifFalse: [  
    saveRow := rowCol at: 1.  
].
```

where currentSession is an inherited instance variable (from GOOIHostController) that addresses the session object, and rowCol is a local variable.

## searchAll

**Description:** Search the object's copy of the host data for a target string. The search begins from row 1, column 1 regardless of the value of the searchRowColumn attribute.

**Assumptions:** The searchString attribute has been set to the target string (for example, "INV001").

**Return value:** Array with row/column where target string was found, or empty array if target string was not found.

**Receiver modified:** No.

**Example:**

```
rowCol := currentSession searchAll.  
(rowCol = {23 2}) ifTrue: [  
    ^ true.  
].
```

where currentSession is an inherited instance variable (from GOOIHostController) that addresses the session object.

---

## searchNext

**Description:** Search the object's copy of the host data for a target string. This string and the starting row/column for the search are specified in object attributes that were set by a previous search. The search begins at one position past the starting row/column attribute.

**Assumptions:** The searchString and searchRowColumn attributes were set by a previous search method which found the string.

**Return value:** Array with row/column where target string was found, or empty array if target string was not found.

**Receiver modified:** No.

### Example:

```
rowCol := currentSession searchNext.  
(rowCol = {10 41}) ifTrue: [  
    ^ true.  
].
```

where currentSession is an inherited instance variable (from GOOIHController) that addresses the session object.

---

## wait

**Description:** For emulators, the wait method waits for the host to return to an idle state following a request (for example, refreshHostFor:item:). No wait occurs if this method is executed when the direct TCP/IP connection of gOOi is used.

**Assumptions:** A session with the host exists.

**Return value:** None.

**Receiver modified:** No.

### Example:

```
currentSession wait.
```

where currentSession is an inherited instance variable (from GOOIHController) that addresses the session object.

---

## Attributes

---

### screenIDValue

**Description:** A string containing the host data at the row/column position of the active screen ID profile.

---

### searchRowColumn

**Description:** An array with the row and column at which searching is to begin. Used by the search and searchNext methods.

---

### searchString

**Description:** A string to be located in the host data by one of the search methods.

---

## Object: GOOIJITController (Just-In-Time Controller)

This object/class presents a default Just In Time window for any host application screen that gOOi cannot identify.

In contrast to controllers generated by the Application Generator, there is no Host Object associated with GOOIJITController. Methods and controllers for this object are described in the following pages.



---

GOOIJITController is only available with hosts that support a 3270 data stream.

---

---

## Methods

### checkForString: aString

**Description:** Searches the session's host data for a target string.

**Assumptions:** aString is an Object of class String.

**Return value:** object of class Boolean: true if the string is found, and false if the string is not found.

**Receiver modified:** No.

**Example:**

```
(GOOIJIT checkForString: 'Login') ifTrue: [  
    ^ true.  
].
```

where GOOIJIT is an instance of the GOOIJITController class.

---

### checkForString:aString at:anArray

**Description:** Searches the session's host data for a target string at a target row/column location.

**Assumptions:** aString is an object of class String. anArray is a 2-element Array object consisting of the row and column of the target location (for example, {3 72} ).

**Return value:** Object of class Boolean: true if the string is found, and false if the string is not found.

**Receiver modified:** No.

**Example:**

```
(GOOIJIT checkForString: 'Login' at: {8 27}) ifTrue: [
    ^ true.
].
```

where GOOIJIT is an instance of the GOOIJITController class.

---

### enterString:aString at:location

**Description:** Copies a string to a target location in the session's host data.

**Assumptions:** aString is an object of class String. location is either a 2-element Array object consisting of the row and column of the target location, or a Number object with the number of the entry field where the string is to be copied.

**Return value:** The receiver.

**Receiver modified:** No.

**Example:**

```
GOOIJIT enterString: 'MASTER' at: {8 34}.
```

where GOOIJIT is an instance of the GOOIJITController class.

---

**functionKeyPressed: aKey**

**Description:** Sends the specified key to the Host.

**Assumptions:** aKey is an Object of class Symbol. The key name may either be an ELLAPI value or logical mnemonic. A session with the host exists.

**Return value:** The receiver.

**Receiver modified:** No.

**Example:**

```
GOOIJIT functionKeyPressed: #ENTER.
```

where GOOIJIT is an instance of the GOOIJITController class.

---

**pressClear**

**Description:** Sends the CLEAR key to the Host.

**Assumptions:** A session with the host exists.

**Return value:** None.

**Receiver modified:** No.

**Example:**

```
GOOIJIT pressClear.
```

where GOOIJIT is an instance of the GOOIJITController class.

---

**pressEnter**

**Description:** Sends the ENTER key to the Host.

**Assumptions:** A session with the host exists.

**Return value:** None.

**Receiver modified:** No.

**Example:**

```
GOOIJIT pressEnter.
```

where GOOIJIT is an instance of the GOOIJITController class.

---

## refreshFields

**Description:** Synchronizes the dynamic host attributes of GOOJIT with the gOOi session.

**Assumptions:** A session with the host exists.

**Return value:** The receiver.

**Receiver modified:** Yes.

**Example:**

```
GOOIJIT refreshFields.
```

where GOOIJIT is an instance of the GOOIJITController class.

---

## refreshHostFor: aFieldName

**Description:** Synchronizes the host session for a given fieldName from a given dynamic GOOIJIT controllerItem.

**Assumptions:** aFieldName is an object of class Symbol. A session with the host exists.

**Return value:** The receiver.

**Receiver modified:** No.

**Example:**

```
GOOIJIT refreshHostFor: #hostFieldR23C73.
```

where GOOIJIT is an instance of the GOOIJITController class, and hostFieldR23C73 is a host screen entry field at row 23, column 73.

---

**waitForString: aString**

**Description:** Searches the session's host data for a target string up to a maximum of 10 tries.

**Assumptions:** aString is an Object of class String.

**Return value:** Object of class Boolean: true if the string is found, and false if the string is not found.

**Receiver modified:** No.

**Example:**

```
(GOOIJIT waitForString: 'Password') ifTrue: [  
    ^ true.  
].
```

where GOOIJIT is an instance of the GOOIJITController class.

---

**waitForString:aString at:anArray**

**Description:** Searches the session's host data for a target string at a target row/column location, up to a maximum of 10 tries.

**Assumptions:** aString is an object of class String. anArray is a 2-element Array object consisting of the row and column of the target location (for example, {3 72}).

**Return value:** Object of class Boolean: true if the string is found, and false if the string is not found.

**Receiver modified:** No.

**Example:**

```
(GOOIJIT waitForString:'Pass' at: {9 27}) ifTrue: [  
    ^ true.  
].
```

where GOOIJIT is an instance of the GOOIJITController class.

---

## Attributes

---

### currentSession

**Description:** Identifies the connection that gOOi has established with the host.

---

### entryFields

**Description:** An array of updateable host session fields ordered by relative location (top to bottom, left to right).

## Object: GOOIJITCustomController (Just-In-Time Custom Controller)

This class inherits the methods and attributes of GOOIJIT, its superclass. Any host navigation logic for host screens that do not have matching gOOi forms should be placed in this class.

This class should also be used by forms created in ObjectStudio outside of the gOOi Application Generator that need to communicate with gOOi attached applications (when there are no gOOi Application Generated forms to be used with host navigation). Methods and controllers for this object are described in the following pages.



---

GOOIJITCustomController is only available with hosts that support a 3270 data stream.

---

## Methods

---

### screenChangedOn: aSession



The screenChangedOn: method is very powerful because it affords you the opportunity to examine the active host screen and identify it as having a matching gOOi form, even though this host screen lacks a screen ID. This method also has the potential to degrade gOOi performance because it is called every time that the host screen changes. Ensure that this method runs efficiently so that gOOi response time does not suffer.

---

**Description:** If implemented, called by the Host Monitor every time that a host screen changes. Provides a way to identify a gOOi form for a host screen that does not have a screen ID in a location matching one of the available screen ID profiles.

**Assumptions:** If the host screen is identified as having a matching gOOi form, the screenIDValue attribute of the session must be set with the gOOi form name.

**Return value:** True if host screen is identified for which there is a gOOi form; otherwise false.

**Receiver modified:** No.

**Example:**

```
screenChangedOn: aSession

(self checkForSignOn: aSession) ifTrue: [ ^ true.].
(self checkForPrintFacility: aSession) ifTrue: [ ^ true.].

^ false.

checkForSignOn: aSession
| rowCol |

aSession searchString: '5401.202'.
rowCol := aSession searchAll.
rowCol = {3 2} ifTrue: [
    aSession searchString: 'User'.
    rowCol := aSession searchAll.
    rowCol = {20 28} ifTrue: [
        aSession screenIDValue: 'SIGN_ON'.
        ^ true.
    ].
].
^ false.

checkForPrintFacility: aSession
| rowCol |

aSession searchString: 'Print Facility'.
rowCol := aSession searchAll.
rowCol = {1 33} ifTrue: [
    aSession screenIDValue: 'MPFMME'.
    ^ true.
].
^ false.
```

In this example, the *screenChangedOn:* method calls two other methods. First, *checkForSignOn:* is executed and looks for 2 strings at 2 different host screen row/column locations. If both strings are found, *checkForSignOn:* sets the screen ID to 'SIGN\_ON' and returns true to *screenChangedOn:*, which then returns true to the HostMonitor.

If *checkForSignOn:* returns false, *screenChangedOn:* calls *checkForPrintFacility:*. If *checkForPrintFacility:* finds the string 'Print Facility' at row 1, column 33, the screenIDValue is set to 'MPFMME' and true is returned to *screenChangedOn:*, which in turn returns true to the HostMonitor. If *checkForPrintFacility:* returns false, *screenChangedOn:* returns false to the HostMonitor and normal gOOi host screen identification processing continues.

---

## startUp

**Description:** Automatically called by GOOIJIT the first time this GOOIJIT subclass is opened. It is only called once per gOOi session.

**Assumptions:** This method would not normally be implemented if a start up file is used.

**Return value:** The receiver.

**Receiver modified:** No.

**Example:** The start up file sample, gOOiStartUpSampleMainframe.txt, contains sample code that could be included in this method.

---

## Attributes

This class inherits the attributes of GOOIJIT, its superclass.

## Object: GOOIVTEmulatorCustomController

This class is the OpenVMS/UNIX parallel to GOOIJITCustomController. Any host navigation logic for host screens that do not have matching gOOi forms should be placed in this class.

This class should also be used by forms created in ObjectStudio outside of the gOOi Application Generator that need to communicate with gOOi attached applications (when there are no gOOi Application Generated forms to be used with host navigation). Methods and controllers for this object are described in the following pages.



---

GOOIVTEmulatorCustomController is only available with OpenVMS/UNIX hosts.

---

---

## Methods

---

### checkForString: aString

**Description:** Searches the session's host data for a target string.

**Assumptions:** aString is an Object of class String.

**Return value:** object of class Boolean: true if the string is found, and false if the string is not found.

**Receiver modified:** No.

**Example:**

```
(currentSession emulator checkForString: 'Login') ifTrue: [  
    ^ true.  
].
```

where currentSession is an inherited instance variable (from GOOIHostController), and emulator is an instance variable in the session object.

---

**checkForString:aString at:anArray**

**Description:** Searches the session's host data for a target string at a target row/column location.

**Assumptions:** aString is an object of class String. anArray is a 2-element Array object consisting of the row and column of the target location (for example, {3 72} ).

**Return value:** Object of class Boolean: true if the string is found, and false if the string is not found.

**Receiver modified:** No.

**Example:**

```
(currentSession emulator checkForString: 'Login' at: {8 27})
  ifTrue: [
    ^ true.
  ].
```

where currentSession is an inherited instance variable (from GOOIHostController), and emulator is an instance variable in the session object.

---

**enterString: aString**

**Description:** Sends the specified string to the session's host data at the current cursor position.

**Assumptions:** A session with the host exists.

**Return value:** None.

**Receiver modified:** No.

**Example:**

```
currentSession emulator enterString: 'User'.
```

where currentSession is an inherited instance variable (from GOOIHostController), and emulator is an instance variable in the session object.

## eraseToEndOfField

**Description:** Sends the key sequences to MANTIS that clear the data in the MANTIS field from the current cursor position to the end of the field.

**Assumptions:** None.

**Return value:** None.

**Receiver modified:** No.

**Example:**

```
currentSession emulator eraseToEndOfField.
```

where `currentSession` is an inherited instance variable (from `GOOIHostController`), and `emulator` is an instance variable in the session object.

---

## getStringAtRow: aRow column: aColumn length: aLength

**Description:** Gets the string of specified length from the session's host data at the specified row and column.

**Assumptions:** A session with the host exists.

**Return value:** Object of class `String`.

**Receiver modified:** No.

**Example:**

```
x := currentSession emulator getStringAtRow: 3 column: 6  
length: 8.
```

where `currentSession` is an inherited instance variable (from `GOOIHostController`), `emulator` is an instance variable in the session object, and `x` is a local variable in the method.

---

## pressEnter

**Description:** Sends the ENTER key to the Host.

**Assumptions:** A session with the host exists.

**Return value:** None.

**Receiver modified:** No.

**Example:**

```
currentSession pressEnter.
```

where currentSession is an inherited instance variable (from GOOIHostController), and emulator is an instance variable in the session object.

---

## pressTab

**Description:** Sends the Tab key to the Host.

**Assumptions:** A session with the host exists.

**Return value:** None.

**Receiver modified:** No.

**Example:**

```
currentSession emulator pressTab.
```

where currentSession is an inherited instance variable (from GOOIHostController), and emulator is an instance variable in the session object.

---

**screenChangedOn: aSession**


---



The screenChangedOn: method is very powerful because it affords you the opportunity to examine the active host screen and identify it as having a matching gOOi form, even though this host screen lacks a screen ID. This method also has the potential to degrade gOOi performance because it is called every time that the host screen changes. Ensure that this method runs efficiently so that gOOi response time does not suffer.

---

**Description:** If implemented, called by the Host Monitor every time that a host screen changes. Provides a way to identify a gOOi form for a host screen that does not have a screen ID in a location matching one of the available screen ID profiles.

**Assumptions:** If the host screen is identified as having a matching gOOi form, the screenIDValue attribute of the session must be set with the gOOi form name.

**Return value:** True if host screen is identified for which there is a gOOi form; otherwise false.

**Receiver modified** No.

**Example:**

```
screenChangedOn: aSession

(self checkForSignOn: aSession) ifTrue: [ ^ true.].
(self checkForDirectoryFacility: aSession) ifTrue: [ ^
true.].

^ false.

checkForSignOn: aSession
| rowCol |

aSession searchString: '(C) Cincom Systems, Inc. 2000'.
rowCol := aSession searchAll.
rowCol = {4 43} ifTrue: [
    aSession screenIDValue: 'SIGNON'.
    ^ true.
].
```

```

checkForDirectoryFacility: aSession
| rowCol |

aSession searchString: 'Directory Facility'.
rowCol := aSession searchAll.
rowCol = {11 45} ifTrue: [
    aSession screenIDValue: 'DIRFAC'.
    ^ true.
].
^ false.

```

In this example, the *screenChangedOn:* method calls two other methods. First, *checkForSignOn:* is executed and looks for a string at row 4, column 43. If the string is found, *checkForSignOn:* sets the screen ID to 'SIGNON' and returns true to *screenChangedOn:*, which then returns true to the HostMonitor.

If *checkForSignOn:* returns false, *screenChangedOn:* calls *checkForDirectoryFacility:*. If *checkForDirectoryFacility:* finds the string 'Directory Facility' at row 11, column 45, the screenIDValue is set to 'DIRFAC' and true is returned to *screenChangedOn:*, which in turn returns true to the HostMonitor. If *checkForDirectoryFacility:* returns false, *screenChangedOn:* returns false to the HostMonitor and normal gOOi host screen identification processing continues.

---

## startUp

**Description:** Automatically called by GOOHostMonitor when the Host Monitor is started. It is only called once per gOOi session.

**Assumptions:** This method would not normally be implemented if a start up file is used.

**Return value:** The receiver.

**Receiver modified:** No.

**Example:** The start up file sample, gOOiStartUpSampleUNIXDigital.txt, contains sample code that could be included in this method.

### **waitForString: aString**

**Description:** Searches the session's host data for a target string up to a maximum of 10 tries.

**Assumptions:** aString is an Object of class String.

**Return value:** Object of class Boolean: true if the string is found, and false if the string is not found.

**Receiver modified:** No.

**Example:**

```
(currentSession emulator waitForString: 'Password') ifTrue: [  
    ^ true.  
].
```

where GOOIJIT is an instance of the GOOIJITController class.

---

### **waitForString:aString at:anArray**

**Description:** Searches the session's host data for a target string at a target row/column location, up to a maximum of 10 tries.

**Assumptions:** aString is an object of class String. anArray is a 2-element Array object consisting of the row and column of the target location (for example, {3 72}).

**Return value:** Object of class Boolean: true if the string is found, and false if the string is not found.

**Receiver modified:** No.

**Example:**

```
(currentSession emulator waitForString:'Pass' at: {9 27})  
    ifTrue: [  
        ^ true.  
    ].
```

where GOOIJIT is an instance of the GOOIJITController class.

---

---

## windowDown

**Description:** Sends the keypad 2 key to the host, which moves the display down if supported by the active MANTIS screen.

**Assumptions:** None.

**Return value:** None.

**Receiver modified:** No.

**Example:**

```
currentSession emulator windowDown.
```

where `currentSession` is an inherited instance variable (from `GOOIHostController`), and `emulator` is an instance variable in the session object.

---

## windowHome

**Description:** Sends the keypad 7 key to the host, which moves the display to the home position if supported by the active MANTIS screen.

**Assumptions:** None.

**Return value:** None.

**Receiver modified:** No.

**Example:**

```
currentSession emulator windowHome.
```

where `currentSession` is an inherited instance variable (from `GOOIHostController`), and `emulator` is an instance variable in the session object.

## windowLeft

**Description:** Sends the keypad 4 key to the host, which moves the display left if supported by the active MANTIS screen.

**Assumptions:** None.

**Return value:** None.

**Receiver modified:** No.

**Example:**

```
currentSession emulator windowLeft.
```

where `currentSession` is an inherited instance variable (from `GOOIHostController`), and `emulator` is an instance variable in the session object.

---

## windowRight

**Description:** Sends the keypad 6 key to the host, which moves the display right if supported by the active MANTIS screen.

**Assumptions:** None.

**Return value:** None.

**Receiver modified:** No.

**Example:**

```
currentSession emulator windowRight.
```

where `currentSession` is an inherited instance variable (from `GOOIHostController`), and `emulator` is an instance variable in the session object.

---

## windowUp

**Description:** Sends the keypad 8 key to the host, which moves the display up if supported by the active MANTIS screen.

**Assumptions:** None.

**Return value:** None.

**Receiver modified:** No.

**Example:**

```
currentSession emulator windowUp.
```

where `currentSession` is an inherited instance variable (from `GOOIHostController`), and `emulator` is an instance variable in the session object.

---

## Attributes

This class inherits the attributes of `GOOIVTEmulator`, its superclass.



# 9

## Deploying a gOOi application

This chapter provides the steps for deploying a completed gOOi application to a properly licensed end user (client) workstation. For a gOOi application to run, the client must have:

- ◆ An ObjectStudio executable (for example, ostudio.exe)
- ◆ An application image (.img file, which you create, see below)
- ◆ All ObjectStudio DLLs
- ◆ Any emulator-specific DLLs
- ◆ Any required prompters (.pmt file, which you create)

A sample directory list is provided in the README file.



---

The following process assumes that if you use an emulator, the emulator is licensed, installed, and operational on the client workstation.

---

The deployment process includes:

- ◆ Creating an ObjectStudio image (.img file) of your application in your development environment
- ◆ Creating a deployment subdirectory on your development workstation to hold your run-time/deployment entities
- ◆ Creating a subdirectory on the client workstation to accept the application.exe file, .img file, and DLLs from the developer
- ◆ Copying the contents of the developer's deployment subdirectory to the corresponding client directory
- ◆ Making a host connection and running the application on the client workstation

## gOOi application deployment steps

There are two ways to create an ObjectStudio image file (.img) for deployment of your completed gOOi application on end-user workstations:

- ◆ The standard Program Generator tool is available from the ObjectStudio menu. This tool is the simpler and more automated of the two choices, but creates a larger image file that may slow execution on smaller client workstations. To make this tool easier to use, all ObjectStudio classes that your application might need are included in the generated image.
- ◆ The Small Program Generator (SPgen) requires more detailed knowledge of the classes needed to run your application and will often require iterative work to complete the process. The benefit of SPgen is that it can produce a much smaller image that will speed execution of your application on all client workstations. It is initiated by starting ObjectStudio with the `-l` command line parameter. This `-l` parameter requires the name of a text file, as in the following example shortcut target:

```
C:\ostud630\ostudio.exe -l c:\ostud630\gooi\spgen\spgen63RunOnly.txt
```

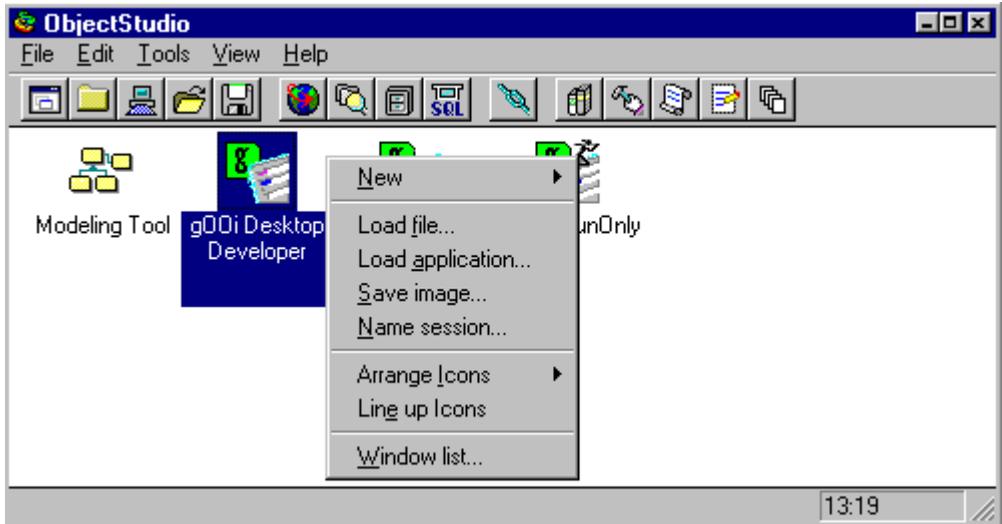
The specified text file, `spgen63RunOnly.txt`, contains a list of ObjectStudio classes that provide various functionality. It is installed as a working sample in the `spgen` subfolder of the `gOOi` folder. Many of the classes listed in this file are commented, and so are not included in the generated image file.

The exclusion of unused classes allows the Small Program Generator to produce a smaller image file than the standard Program Generator. The commented classes implement product options. Added functionality via form customizations may utilize some of these options and may require the inclusion of some commented classes.

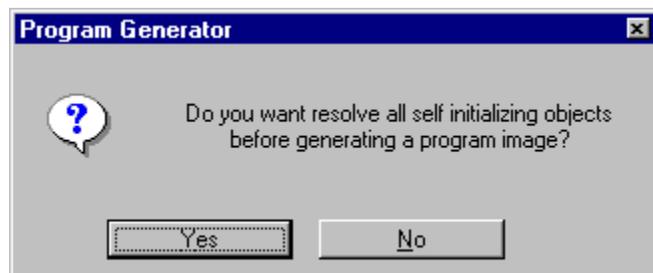
## Using the standard Program Generator

When using the standard Program Generator, perform the following steps *on your developer's workstation*:

1. Run ObjectStudio.
2. At the ObjectStudio Workplace Desktop, click mouse button 2 in the Work Area. A pop-up menu displays.

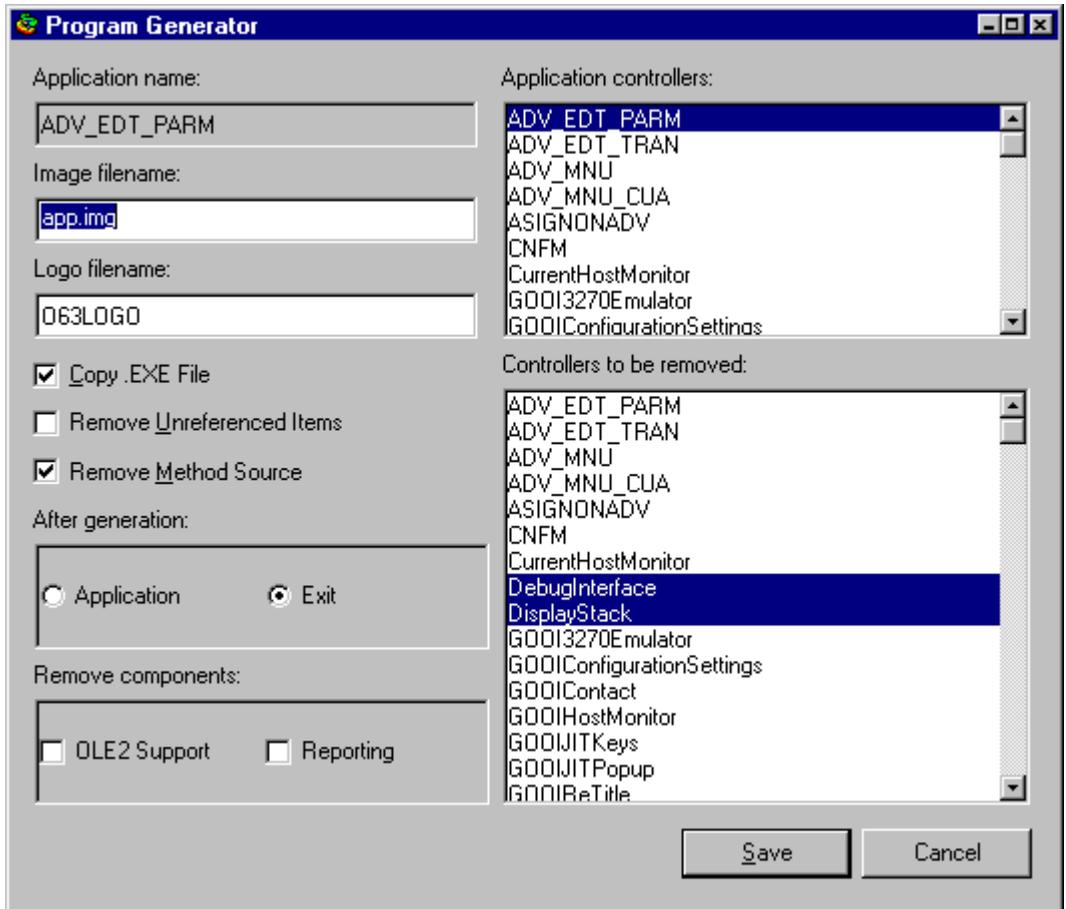


3. Select Load application. This opens the Applications picking list.
4. From the applications list, select gOOi Runtime and click **Load**.
5. From the Applications picking list, select your generated application, click **Load**, then click **Close**.
6. Select Tools ⇒ Program Generator. After gOOi loads the System Proxies, the following Program Generator message displays:





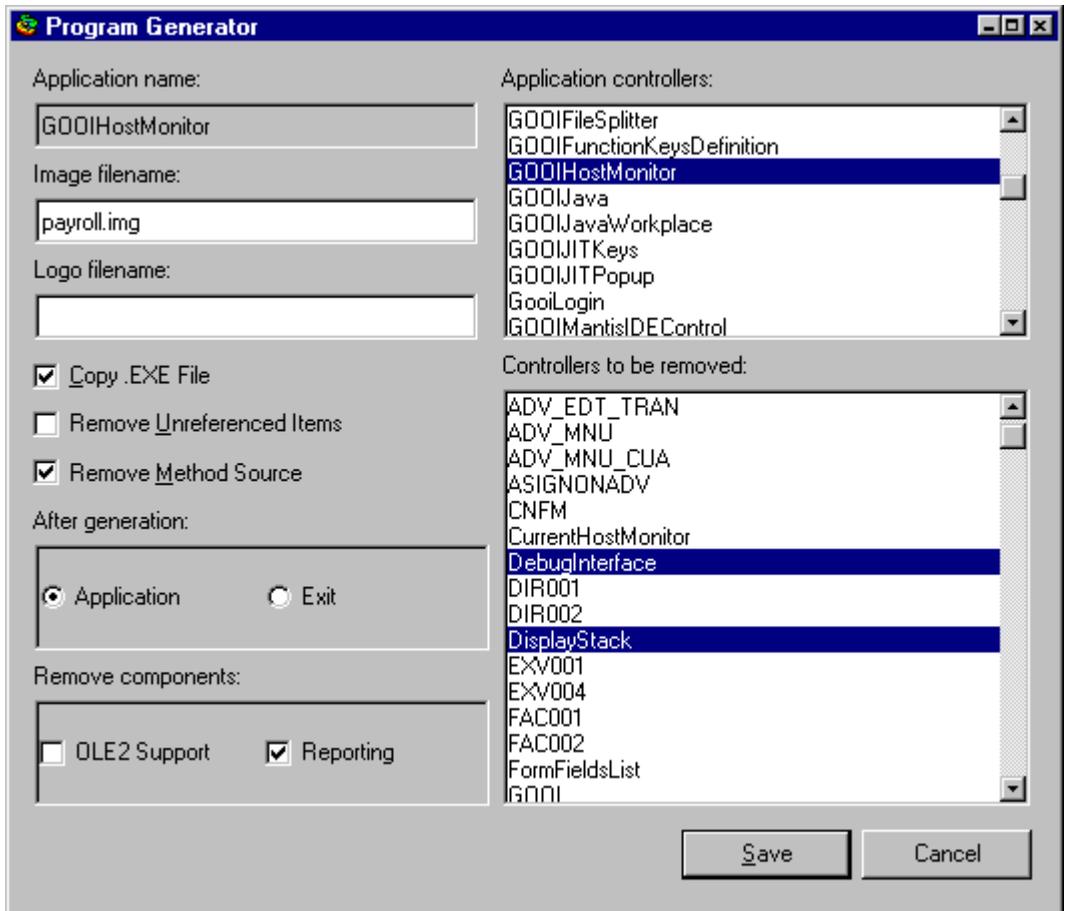
7. Reply **Yes**, and the System Proxies are loaded. After the loading completes, the Program Generator window appears:



8. Set the following fields:

Field	Description
Application name:	Indicate the first item to be executed when your deployed application opens. This must display GOOIHSTMonitor.
Application controllers:	Specify the first application. Highlight GOOIHSTMonitor.
Image filename:	Specify the name of the run-time image of your application. This name usually reflects the application this represents (for example, PAYROLL). An .img extension should follow the specified name.
Logo filename:	Specify the DLL that loads logo.bmp (a banner page for your application provided by ObjectStudio). If you do not want logo.bmp loaded, delete this value.
Copy.Exe file	Select this check box, so that the ObjectStudio executable is copied during the generation process. It receives the same file name prefix as the image (for example, for an image named PAYROLL.IMG, the executable is PAYROLL.EXE).
Remove Unreferenced Items	Do not select this check box.
Remove Method Source	Select this check box, so that the source code is not included in the created image file.
After generation:	Select Application so that ObjectStudio will display the gOOi Host Monitor window after image creation.
Controllers to be removed:	<p>Highlight any controllers that are unnecessary for execution so that they are excluded from the image file.</p> <p>Do <i>not</i> highlight:</p> <ul style="list-style-type: none"> <li>(1) Anything that names your generated screens and prompters</li> <li>(2) Anything that ends with the word <i>monitor</i></li> </ul>
Remove components:	Check Reporting. Check OLE2 Support only if you know you are not using OLE via the Word Wizard, Excel Wizard, or form customizations.

The following Program Generator window shows the effects of setting the fields per the preceding guidelines:



- Click **Save**, and you will be prompted for a confirmation. Reply **Yes**, and the generation process begins by displaying a Transcript window. The length of the process depends on the power of your workstation and the volume of generated entities. The gOOi Host Monitor window displays when the generation completes if Application was selected for 'After generation'. When the process is complete, the Generator places a .img file of your application in your ObjectStudio directory.

10. Create a subdirectory *on your development workstation* to hold all your run-time/deployment entities. This includes the following steps:
  - a. Place your application .img file and application .exe file in your deployment subdirectory.
  - b. Copy all ObjectStudio DLLs from the DLLW32 subdirectory of ObjectStudio.
  - c. Copy all .pmt files (prompters), if applicable, to your deployment subdirectory.
  - d. Copy the gooi.ini file from the ObjectStudio directory to your deployment subdirectory.
  - e. Copy the logo.bmp file from the ObjectStudio directory to your deployment subdirectory.
11. *On the client workstation*, create a subdirectory to receive the deployment entities from the developer.
12. Copy all entities from the developer's deployment subdirectory created in step 9 into the corresponding client directory created in step 10.
13. If you use an external emulator, check your PATH to make sure it includes the emulator. This should have been done automatically when the emulator was installed.
14. Create the appropriate program groups and items.
15. Specify the application.exe file in the command line for your program item.
16. On the client workstation, run your gOOi application (assuming an active emulator, if you are using one). The GOOIHostMonitor window displays; press **Start** if it has not been started automatically. The host monitor task is minimized after the task is started. As the application runs, your gOOi forms will display.

## Using the Small Program Generator



Before beginning to use the small Program Generator, review the section of the ObjectStudio documentation that explains the operation of this option. It is important to understand that creating a small image can be an iterative process of editing and testing the startup text file.

The advantage of the small program generator is that it creates a smaller image file than the standard program generator. It is initiated by executing ObjectStudio with a start up parameter (-l) that specifies an input text file. This text file contains a list of ObjectStudio classes that might be needed by the application(s) that are included in the image. Classes that are not needed by the application(s) can be commented, thus reducing the size of the created image file.

For example, in these 3 lines from an input text file:

```
cls\block.cls
w32\cls\block2.cls
"cls\bdecode.cls
```

The last line is commented (starts with a quote (")) because bdecode.cls is not needed in the image.

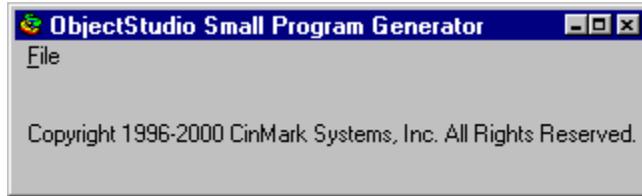
A number of working sample text files were installed with gOOi in the spgen subfolder of the gOOi folder. The spgen64RunOnly.txt file is for use with Release 6.4 of ObjectStudio for generating an image file when OLE support is not necessary. Another sample text file, spgen64RunOnlyOLE.txt, also works with Release 6.4 of ObjectStudio, but includes class files for OLE support. This spgen subfolder also includes files for ObjectStudio 6.3. Contact Cincom support before attempting to use the small Program Generator if you are using a release of ObjectStudio other than 6.4 or 6.3. The following steps assume that you are using the Spgen63RunOnly.txt file with Release 6.3:

1. Run ObjectStudio with the -l command line parameter. The following is an example of a shortcut target associated with a desktop icon:

```
C:\ostud630\ostudio.exe -lc:\ostud630\gooi\spgen\spgen63RunOnly.txt.
```

If any error occurs, edit spgen63RunOnly.txt to remove comments from required classes and retry this step.

2. The ObjectStudio Small Program Generator window displays:



3. Select File ⇒ Load Application from the menu. The Applications window appears.
4. From the list of applications, select gOOi Runtime and click **Load**. Also select and load all the gOOi applications you have generated that you want to deploy. If any error occurs, edit spgen63RunOnly.txt to remove comments from required classes and retry from step 1.
5. From the ObjectStudio Small Program Generator window, select File ⇒ Save Image from the menu to save the image file with the name of your choice, such as cust.img.
6. Test the image file by starting ObjectStudio with this file specified in the -i command line parameter, such as: -icust.img. If your application runs successfully, this image can be deployed to end-user workstations. If any error occurs, edit spgen63RunOnly.txt to remove comments from required classes and retry from step 1.
7. Continue at step 9 of the preceding procedure for the standard Program Generator. The application .exe file referred to in that step is just a copy of Ostudio.exe, renamed to match the image file name (for example, cust.exe).

# 10

## PC CONTACT file access

### Overview of PC CONTACT file access



PC CONTACT file access is only available to IBM mainframe MANTIS users with this feature authorized on their system. It can be run only with the gOOi TCP/IP connection, and not via an emulator connection.

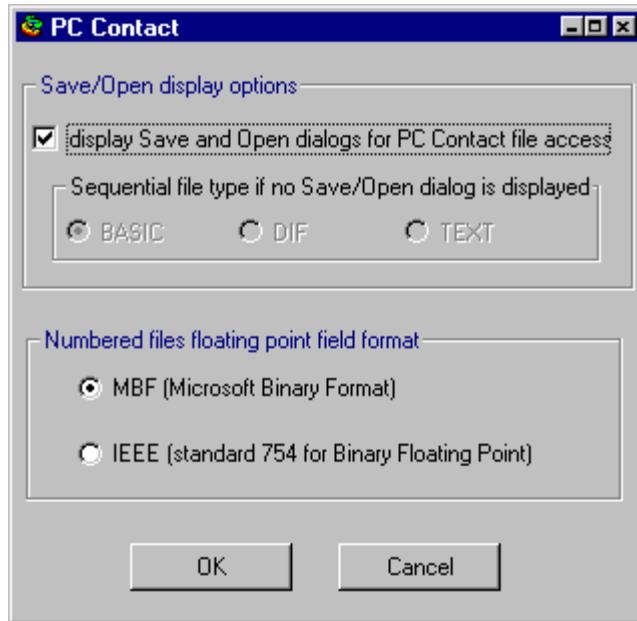
PC CONTACT provides upload/download file access between MANTIS for the mainframe and personal computers. PC CONTACT lets you extract data from files on the mainframe using MANTIS, and download this information to a file on your PC. PC CONTACT also allows you to read PC files from MANTIS programs.

PC CONTACT provides a direct connection between MANTIS and PC files. For example, a simple MANTIS program can read rows from DB2 tables and insert the data from these rows into a file on your PC. Conversely, it is easy to create a MANTIS program that directly reads data from a PC file and populates a DB2 table.

## Options for PC CONTACT

To access the options for PC CONTACT:

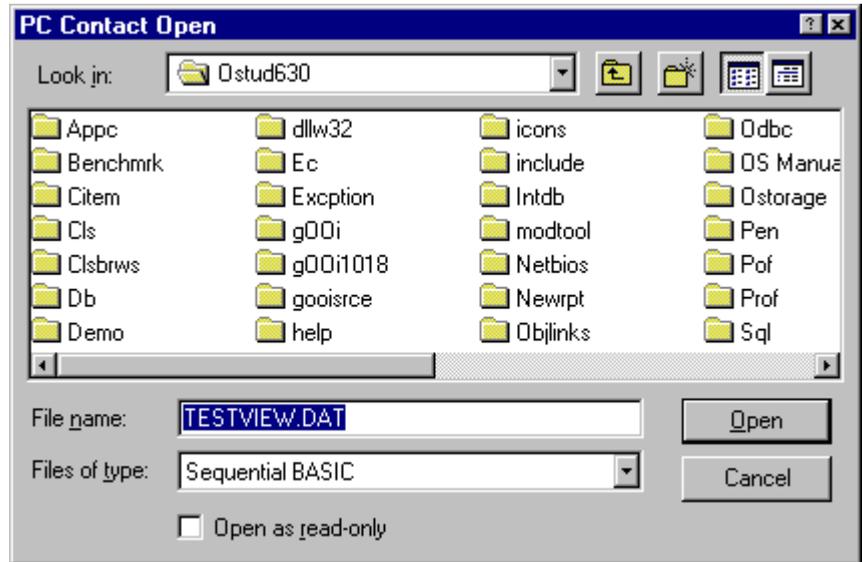
1. Double-click the icon for gOOi Desktop Developer.
2. Double-click the icon for Settings.
3. Click the **PC CONTACT** button. The following window displays:



You can display this same dialog at any time while running gOOi. Right-click near the border of the window to display the pop-up menu, then select PC CONTACT from the list of menu items.

## Save/Open display options

By default, the first file access during an upload displays the common Windows 'Open' window:



The first file access during a download displays the common Windows 'Save As' window:



To eliminate these common dialogs, turn off the check box for display Save and Open dialogs for PC CONTACT file access. If you do this, the three subsequent radio buttons are enabled in the topic box for Sequential file type if no Save/Open dialog is displayed. This file type selection is necessary because the ACCESS view in MANTIS that defines the PC file does not include a specification for the type of sequential file access (BASIC, DIF, or TEXT). The type that you choose will be used for all sequential file access when the common dialogs are turned off.



---

If you turn off the Open/Save As common dialogs, PC CONTACT will attempt to use the PC file name specified in the ACCESS view in MANTIS. Do not turn off these common dialogs if you want to be able to override the ACCESS view.

---

### Numbered files floating point field format

If you access numbered files via PC CONTACT, this option specifies the floating point field format. The choices are:

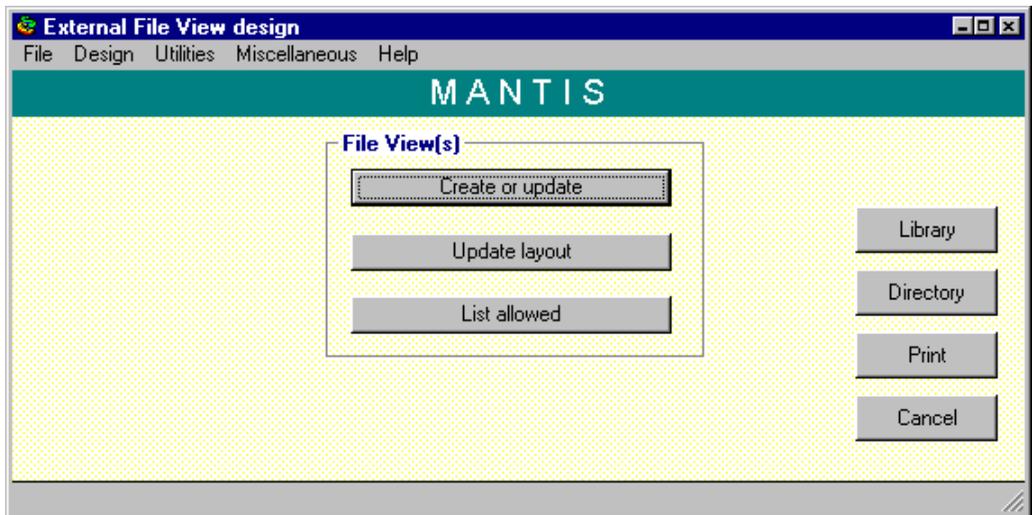
- ◆ **MBF (Microsoft Binary Format).** (Default) This is the floating point format supported by interpretive BASIC.
- ◆ **IEEE (IEEE standard 754 for Binary Floating Point).** This is the floating point format supported by the Intel 80 x 87 floating point co-processor, and also be used in PC MANTIS, Microsoft C, and Microsoft QuickBasic.

## Designing a MANTIS view for PC CONTACT

MANTIS views must be defined to access PC files. This section identifies the options of the External File View Design Facility of MANTIS that are relevant to PC CONTACT. It is intended as a supplement to the *Ad/Advantage MANTIS Facilities OS/390, VSE/ESA*, P39-5001, which covers the MANTIS External File View Design Facility in much greater detail.

For more information on the commands you will need to access and manipulate these views, refer to the *AD/Advantage MANTIS Language OpenVMS/UNIX*, P39-1310. For more information on the types of PC files that PC CONTACT can access, see “PC CONTACT supported file types” on page 286.

When you select the Design External File View option from the MANTIS Facility Selection menu, the following window displays:



The windows displayed in this section for working with the External File View Design facility of MANTIS are gOOi forms. If you instead see Just In Time windows when working with this facility, please contact Cincom about getting a copy of the gOOi Windows client for the MANTIS developer.

To create a new view, click the **Create or update** button. To update an existing view:

1. Click the **Library** button to fetch the view.
2. Once the view has been retrieved, click the **Create or update** button.

### Create or update file views

The Create or Update File Views option allows you to create a new PC file view or update the definition of an existing file view. When you select this option from the External File View Design Facility menu, the following window displays:

**Create or update external file view**

File Design Utilities Miscellaneous Help

Name:

Description:

External name:

**Passwords:** Viewing:

Altering:

Deleting/Inserting:

**VSAM:** Fixed or variable length:

Occurrence controlling element:

First occurring element:

**Attributes:** Status:

File type:

Access method:

Maximum record size:

Reference variable name:

**SAP:** SAP-Release:

Record-ID:

Type:

Compressed

Last profile update date: Last profile update time:

The following items on this window have particular considerations for PC CONTACT.

### **Name**

Specify the PC file name (including path information) in this field. You can override this name at run time if the common dialog option is turned on.

### **Password for viewing**

Specify a password in this field only if a view will be used for uploading data.



---

If a view includes only a password for viewing, PC CONTACT will assume that an upload is intended, and the PC file must already exist.

---

### **Password for deleting/inserting**

Specify a password in this field only if a view is to be used for a download.



---

If a view includes only a password for deleting/inserting, PC CONTACT will assume that a download is intended. If the target PC file already exists, a warning message is displayed that asks if you want to replace it.

---

### **Indexed, sequential, or numbered**

Specify either sequential or numbered for a PC file view. (The indexed choice is for VSAM files only, and does not pertain to PC CONTACT.)

### **Access method**

Specify PC for a PC CONTACT file view.

### **Maximum record size**

Specify the exact record length for numbered files. This field is not used for sequential files.

## Reference variable name

MANTIS requires a record identification for sequential or numbered files because there is no key defined for these files (as is the case for indexed files). The reference variable is a standard BIG numeric field. MANTIS allocates it (together with all the other variables defined in the file view layout) during the processing of the ACCESS statement. This variable has the same multiple buffer allocation and prefixing requirements as all other variables defined in this file view.

The reference variable name must be unique within the MANTIS program area. A sample name is FILENAME\_REFER.

The reference variable contains the relative byte address (RBA) for sequential files, or the relative record number (RRN) for numbered files. The maximum number of records allowed in numbered files is 32,767. For more information on using reference variables when executing MANTIS programs, refer to the descriptions of the GET, UPDATE, INSERT, and DELETE statements in the *AD/Advantage MANTIS Language OpenVMS/UNIX*, P39-1310.

## Update file view layout

The Update File View Layout selection allows you to create a new file view layout or modify an existing file view layout. When you select this option from the External File View Design Facility menu, the following window displays:

MANTIS		EXTERNAL FILE							
Name	Type	Position	Format	Length	Sign	Decimals	Dimension	Offset	Attribute

**Element**

MANTIS name:  Position:  Format:  Length:  Sign:  Decimals:  Dimension:  Offset:  Attribute:

Buttons:

Element count: 0 Page:

EXV002E:Access name must be given and must not start with a blank

The following items on this window have particular considerations for PC CONTACT.

**Position**

For numbered files, specify the position (relative to 1) of this field within the file record. (Position is not used for sequential files because the elements are positioned in the order that they are entered into the file design.)

**Format**

Specify one of the following options from the drop down list box:

- ◆ **ZONED.** Used for zoned (unpacked) decimal
- ◆ **BINARY.** Used for one- or two-byte fields
- ◆ **TEXT.** Used for a character string (maximum 254)
- ◆ **FLOAT.** Used for floating point (either four or eight bytes)

Numbered file fields can use any of the above four formats. Sequential file fields can be either ZONED or TEXT format.

Floating point can be either MBF (default) or IEEE. This can be controlled via the PC CONTACT dialog of Settings.

The following table lists equivalent MANTIS and PC data types:

MANTIS data type	PC data type
ZONED	NUMERIC CHARACTER
BINARY	BINARY or INTEGER
TEXT	CHARACTER
FLOAT (4 BYTES)	SINGLE PRECISION
FLOAT (8 BYTES)	DOUBLE PRECISION

## **Length**

If you specify ZONED format for your PC file data and indicate the number of decimal places (DEC), you must supply the length (LENGTH) of the field on the file (the value supplied for LENGTH will disappear when you press ENTER).

For PC SEQUENTIAL files, this field sets the length of the MANTIS TEXT variable.

## **Offset**

For PC NUMBERED files, if a field is part of a data structure that is also a part of an array, then the occurrence of that field in the block of data is positioned at an interval (offset) that is equal to the length of the data structure.

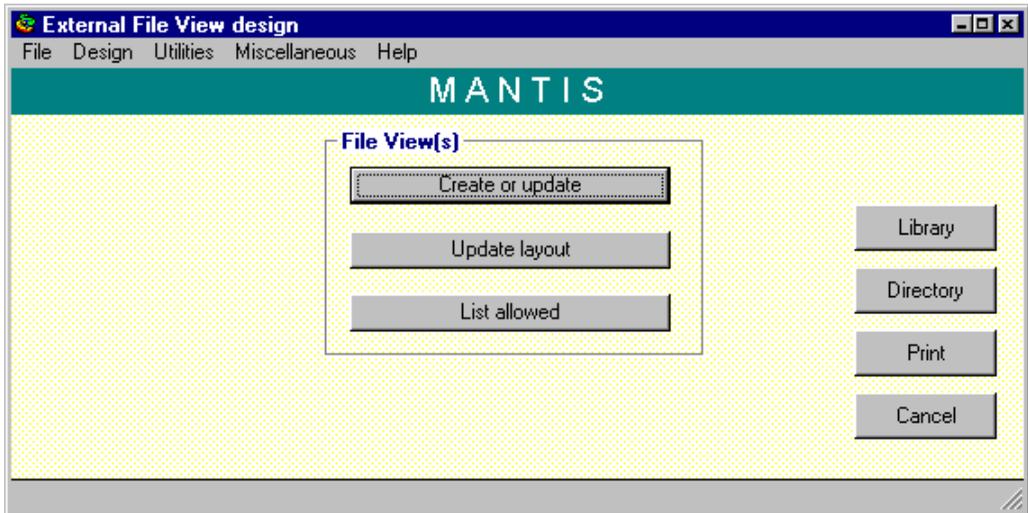
This field does not apply to PC SEQUENTIAL files.

## Sample PC file view design

In this example, assume that you want to create a file view for an Invoice master file. It is a numbered file and contains fixed-length records. Each record can accommodate up to 20 invoice items.

To create the file view:

1. Select the Design External File View option from the MANTIS Facility Selection menu. The External File View Design Facility menu displays:



- When you select the Create or Update File Views option from this menu, the File View Design window appears. Enter the sample data shown in the entry fields:

**Create or update external file view**

File Design Utilities Miscellaneous Help

Name:

Description:

External name:

**Passwords:** Viewing:

Altering:

Deleting/Inserting:

**VSAM:** Fixed or variable length:

Occurrence controlling element:

First occurring element:

**Attributes:** Status:

File type:

Access method:

Maximum record size:

Reference variable name:

**SAP:** SAP-Release:

Record-ID:

Type:

Compressed

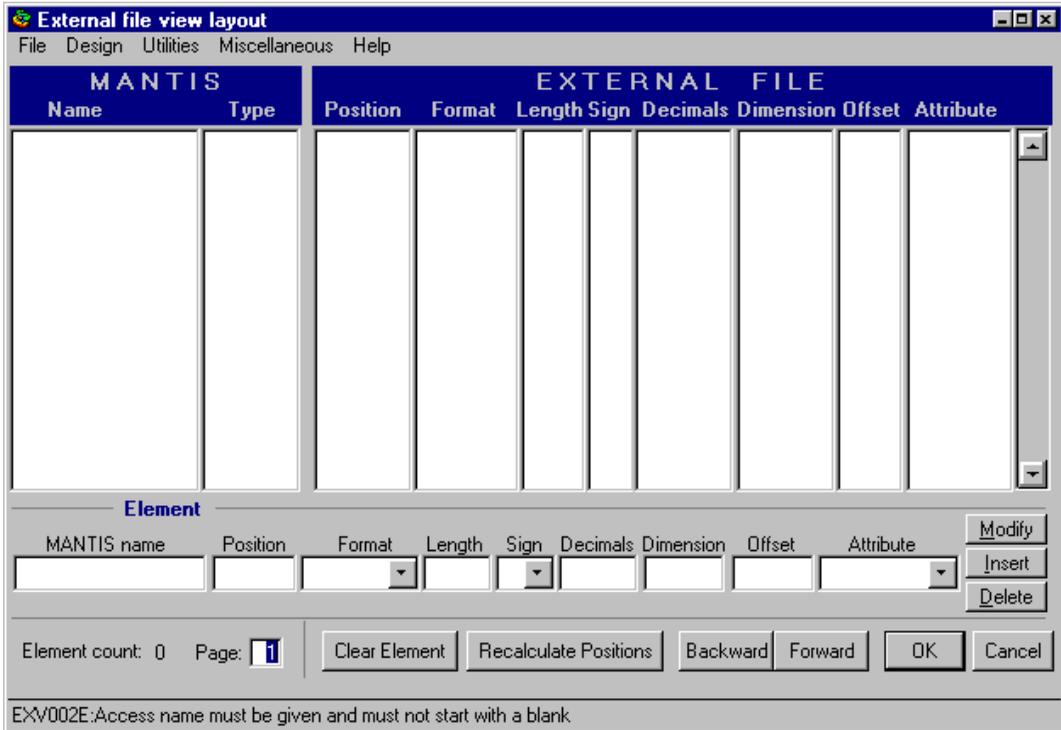
Last profile update date: Last profile update time:

Notice that only a Deleting/Inserting password is specified for this INVOICES\_PC file, since this file view is to be used for downloading records to the PC. The status is Active, enabling a MANTIS program using this file view to proceed with the intended access. The file type is Numbered. The maximum length of each record is 786 bytes, excluding the length of the field itself.

- Press ENTER or click  to store your data. MANTIS automatically returns to the External File View Design Facility menu.

To add elements to the file view layout:

1. Select the Update File View Layout option from the External File View Design Facility menu. The following window displays:



- Enter your data and click **Insert**, one field at a time, as indicated using the Element area of the window:

**External file view layout for INVOICES\_PC**

File Design Utilities Miscellaneous Help

MANTIS		EXTERNAL FILE							
Name	Type	Position	Format	Length	Sign	Decimals	Dimension	Offset	Attribute
CUSTOMER	TEXT	1	TEXT	6					
DEL_ADDRESS	TEXT	7	TEXT	30			3	30	
BILL_AMT	SMALL	97	FLOAT	4					
ITEM	TEXT	101	TEXT	6			20	16	
QUANTITY	SMALL	411	BINARY	2	YES		20	16	
PRICE	BIG	717	FLOAT	8			20	16	

**Element**

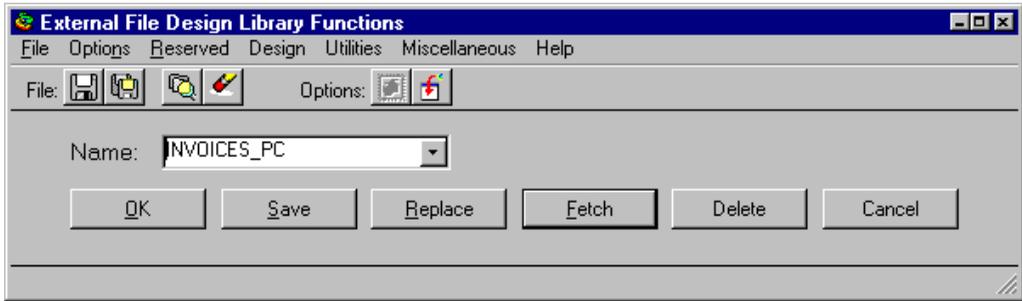
MANTIS name	Position	Format	Length	Sign	Decimals	Dimension	Offset	Attribute	Modify
<input type="text"/>	<input type="button" value="Insert"/>								
									<input type="button" value="Delete"/>

Element count: 6 Page: **1**

- After you have added all the elements that you want, click **Cancel** to return to the External File View Design menu.

To save this file view:

1. Click the **Library** button. The following window displays (note that the current view automatically appears in the Name field):



2. Click **Save** to save the file view.

MANTIS exits to the External File View Design menu when the view is saved. A confirmation message will appear in the lower left corner of the menu, indicating that the view has been saved.

## Sample program

This section provides sample MANTIS download and upload programs.

The following sample MANTIS program downloads data from the mainframe to the PC:

```

10 ENTRY PC_DOWNLOAD
20 .ACCESS V_REC( "VSAM_VIEW" )
30 .ACCESS PC_REC( "SAMPLE_PC_ONE", "WRITE" )
40 .GET V_REC
50 .WHILE V_REC<>"END"
60 ..INSERT PC_REC
70 ..GET V_REC
80 .END
90 SHOW "File Download Complete": WAIT
100 EXIT

```

In the program above, line 20 designates the VSAM external file view that the program reads, while line 30 defines the PC file where data is inserted. The WHILE loop in lines 50 through 80 reads the VSAM file and inserts the data into the PC file. This sample takes advantage of the automatic mapping feature of MANTIS to link the fields from the VSAM view and the PC view.

Use the following program to upload data from the PC to the mainframe:

```

10 ENTRY PC_UPLOAD
20 .VIEW_V_REC( "VSAM_VIEW" )
30 .ACCESS PC_REC( "SAMPLE_PC_ONE", "READ" )
40 .GET PC_REC
50 .WHILE PC_REC<>"END"
60 ..INSERT V_REC
70 ..GET PC_REC
80 .END
90 SHOW "File Upload Complete": WAIT
100 EXIT

```

Compare lines 60 and 70 of the upload program with those in the download program. Notice that to upload data, the PC file name and VSAM view name are reversed.

## PC CONTACT supported file types

PC CONTACT automatically reformats data to the required file type when it downloads data from mainframe MANTIS to the PC. This section contains supplemental information on each of these four file types.

### Sequential BASIC files

A sequential BASIC file stores data in ASCII format using lists (series) of expressions. These expressions (that is, data elements) can be of two types:

- ◆ **Numeric.** Numbers which can contain a decimal point and a sign (ZONED)
- ◆ **String.** Any text value or group of characters enclosed in quotes (TEXT)

Each field is delimited by a comma, while each record is terminated by a carriage return/line feed (shown as "cr/lf"). String expressions are enclosed within double quotes:

```
"J. Smith", "1234 Anywhere", 1234.45, 039458, "Widget", 3, 12.32, cr/lf  
"J. Smith", "2345 Anywhere", 2345.45, 039459, "Frames", 1, 13.67, cr/lf  
"J. Smith", "2334 Anywhere", 1534.45, 039460, "Widget", 4, 14.38, cr/lf  
"J. Smith", "1324 Anywhere", 1534.45, 039468, "Widget", 2, 16.62, cr/lf
```

Think of an ASCII expression as an element on a file. A group of expressions (a list, or line) is equivalent to a record, and a group of lists constitutes a file.

### Sequential TEXT files

A sequential TEXT file stores data in ASCII format using one ASCII line terminated with a carriage return/line feed for each data element defined in the external file view. Data elements may be either numeric (ZONED) or text (TEXT).

## Sequential DIF files

A Data Interchange Format (DIF) file (such as those used in spreadsheets) stores data in tables. Labels can describe numeric data in the tables, referring to all data in a table or to a specific column. Programs can use the numeric data and ignore titles and labels.

DIF uses *vectors* and *tuples* to refer to columns and rows. All vectors (columns) must be of equal length and all tuples (rows) in a table must be of equal length. The following Profit Table has three tuples with three data values in each.

Tuples:

500, 450, 50

300, 275, 25

100, 90, 10

Profit table:

Item	Price	Cost	Profit
Widget	500	450	50
Frames	300	275	25
Gizmo	100	90	10

This table uses four vectors and three tuples, as shown in the following table:

Tuple	Vector 1	Vector 2	Vector 3	Vector 4
	Item	Price	Cost	Profit
Tuple 1	Widget	500	450	50
Tuple 2	Frames	300	275	25
Tuple 3	Gizmo	100	90	10

The DIF file has a header section and a data section. The header section contains labels and information about the size of the table, while the data section contains data values. The column headings Item, Price, Cost, and Profit are labels.



For more detailed information on DIF files, refer to the article by Candace E. Kalish and Melinda F. Mayer, *DIF: A Format for Data Exchange between Application Programs*, published by Byte Publications, Inc., in November, 1981. This article was the source of the information presented in this section.

The following example shows a simple DIF file. The header section of the file begins with the name of the table, and ends with the DATA header item. The data section in this example consists of 3 tuples, each of which begin with the entry:

```
-1,0
BOT
```

The tuples and data section are terminated by the entry:

```
-1,0
EOD
```

#### Header Section:

```
TABLE \
0,1      Name of Table
"PROFITS" /
VECTORS \
0,4      Number of Vectors
" "      /
TUPLES \
0,3      Number of Tuples
" "      /
LABEL \
1,0      Label and Label Position
"ITEM"  /
LABEL
2,0
"PRICE"
LABEL
3,0
"COST"
LABEL
4,0
"PROFIT"
DATA \
0,0      Data Header Item
" "      /
```

**Data Section:**

```
-1,0
BOT
1,0
  "WIDGET "
0,500
V
0,450
V
0,50
V
-----1,0
BOT
1,0
  "FRAMES "
0,300
V
0,275
V
0,25
V
-----1,0
BOT
1,0
  "GIZMO "
0,100
V
0,90
V
0,10
V
-----
-1,0
EOD
```

## Header Items

The header section of a DIF file contains three-line header items that specify the name of the table, the number of vectors, the number of tuples, and the labels and their positions. Header values are in one of the following formats:

- ◆ TUPLES  
0, count  
" "
  
- ◆ LABEL  
Vector number, line number  
"label"

The DATA header item must be the last item in the DIF header. It indicates that all values that follow are data values.

When creating a DIF file, PC CONTACT writes four header items (TABLES, VECTORS, TUPLES, and DATA). When reading a DIF file, PC CONTACT requires that the VECTORS, TUPLES, and DATA header items be present. All other header items are ignored.

## Data Items

DIF data items are organized by tuples, and within the tuples by vector order. A single data item is of the form:

```
type indicator, numeric value
string value
```

Type indicators are as follows:

Type indicator	Description
-1	Specifies a special data value with number value of 0 and string value of either BOT (beginning of tuple) or EOD (end of data)
0	Specifies that the data is numeric and should be stored in the numeric value field
1	Specifies that the data is a string and the string value should be stored in the string value field. Possible string values are V (numeric value), NA (not available), ERROR (invalid calculation), TRUE (with number value of 1), and FALSE (with number value of 0)

BOT must be used to indicate the beginning of each tuple in the data section. EOD must be used to indicate the end of the last tuple in the file.

## Numbered files

Numbered files allow direct (random) access to specific records in a file. Since each record is assigned a number at the time it is inserted in the file, records can be retrieved from any position in the file by specifying the number of the desired record.

One advantage to numbered files is that, because records are accessed directly, it is not necessary to begin at the front of the file and read all the information to retrieve the desired record. Thus, information can be retrieved more quickly than when using sequential files.

In addition, numbered files often store numbers in binary format, instead of in the ASCII format (typically used in sequential files). Therefore, numbered files require less diskette storage space than sequential files.

When accessing a numbered file in BASIC, the file must be opened in random mode, the layout for the data defined in a field statement, and data accessed by GET or PUT statements.

# A

## IBM mainframe considerations

### Platform considerations

This appendix provides information specific to users of IBM mainframes. Please also see [“Downloading screens from MANTIS”](#) on page 111, for related information.

To use the Universal Export Facility (UEF) on the IBM mainframe (MVS), perform the following steps:

1. Ensure that the UEF files are defined.

UEF requires two ESDS VSAM data sets. The first DEFINE is for the UEF data file, and the second is for the UEF log file:

```
DEFINE CLUSTER (NAME(your.dataset.name)-
              VOLUMES(nnnnnn)-
              SHAREOPTIONS(2 3)-
              REUSE-
              NONINDEXED-
              RECORDSIZE(40 254)-
              FREESPACE(50 50))-
       DATA(NAME(your.dataset.name.DATA)-
              CONTROLINTERVALSIZE(4096)-
              RECORDS(20000 2000))

DEFINE CLUSTER (NAME(your.dataset.name)-
              VOLUMES(nnnnnn)-
              SHAREOPTIONS(2 3)-
              REUSE-
              NONINDEXED-
              RECORDSIZE(40 254)-
              FREESPACE(50 50))-
       DATA(NAME(your.dataset.name.DATA)-
              CONTROLINTERVALSIZE(4096)-
              RECORDS(2000 200))
```

## 2. Export the screens and prompters.

The following sample batch MANTIS job will export all screens that begin with the letter C for the gOOi user:

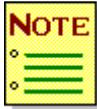
```
//JOBNAME JOB (parm,parm), 'gooi',CLASS=Q,MSGCLASS=T,
// MSGLEVEL=(1,1),USER=*UID,PASSWORD=*PSW,NOTIFY=*UID
//*
//STEP1 EXEC PGM=MANTISB
//STEPLIB DD DSN=your.batchmantis.linklib,DISP=SHR
//SYSPRINT DD SYSOUT=*,DCB=BLKSIZE=133
//SYSUDUMP DD SYSOUT=*,DCB=BLKSIZE=133
//TERMINAL DD SYSOUT=*
//PRINTER DD SYSOUT=*,DCB=BLKSIZE=133
//*
//SETPRAY DD DSN=your.mantis.cluster,DISP=SHR
//CSOT DD DSN=your.mantis.transfer,DISP=SHR
//EXPCLU DD DSN=your.uef.cluster,DISP=SHR
//EXPLOG DD DSN=your.uef.log,DISP=SHR
//KEYBOARD DD *
GOOI;GOOI;
<BLANK=ON>;<FAULT=ON>;<ECHO=ON>;<PAGESIZE=24X80>;
1;
CONTROL:EXP_MAIN_SCB;
EXP;;;C*;;;S;;;;;;;;;
<PA2>
<PA2>
<PA2>
/*
//
```

3. You can also export the screens and prompters into the data file through online UEF. If you use this approach, close this file to CICS after running the export in order to ensure that the VSAM buffers are flushed.

#### 4. REPRO to a sequential data set and transfer to the PC.

The UEF data file must then be REPROed to a sequential data set prior to moving it to the PC. This sequential data set must be defined as variable, with a record length of 258. The following example contains JCL for the REPRO on MVS:

```
//JOBNAME JOB (parm,parm), 'g00i',USER=*UID,PASSWORD=*PSW,
// MSGCLASS=T,NOTIFY=*UID,MSGLEVEL=(1,1),TIME=(,15),CLASS=Q
//BACKUP EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=T
//UEFINP DD DSN=your.UEF.cluster,DISP=SHR
//SEQOUT DD DSN=your.sequential.file,DISP=(,CATLG),
           SPACE=(TRK,(2,1),RLSE),UNIT=SYSDA,
           DCB=(BLKSIZE=2584,LRECL=258,RECFM=VB,DSORG=PS)
//SYSIN DD *
           REPRO INFILE(UEFINP) OUTFILE(SEQOUT)
/*
//
```



The sequential file can be transferred to the PC using your tool of choice, but you must ensure that the EBCDIC character set of the mainframe is translated to the ASCII equivalent on the PC.



# B

## Emulator considerations

### How gOOi recognizes emulators

This appendix provides the settings necessary for gOOi to recognize a specific emulator. In most cases, the short session name is being established. (The short session name is an EHLLAPI requirement.)

These discussions assume that the emulator is already functioning (that it has been successfully installed and a host session has been established). Make sure that the PATH contains your emulator. Consult the appropriate subsection for your emulator. Even though the emulator works properly, gOOi may not recognize it until you perform the steps outlined for the appropriate emulator.

### Emulators

#### PC3270

Check the following items to ensure that gOOi can recognize this emulator:

- ◆ Make sure the directory where the emulator is installed is in your path (default = PERSONAL COMMUNICATIONS).
- ◆ Under Appearance ⇒ Window Setup, make sure the check boxes are selected for each of the following: Short Session ID, Session Name (gOOi Session must be in the entry field), and Separator (-).
- ◆ When you change the keyboard mapping within the emulator, be sure to include between the braces any key command that is not in the list box. For instance, to map a key to the PA2 function, select Assist ⇒ Keyboard setup ⇒ Customize. Specify [PA2] in the window for changing current actions for the selected key.

## EXTRA!

Check the following items to ensure that gOOi can recognize this emulator:

- ◆ Make sure the directory where the emulator is installed is in your path (default = Program Files\E!PC).
- ◆ Under Options ⇒ Global preferences ⇒ Advanced ⇒ HLLAPI short name, place the path to your .EDP session file.

## RUMBA

Check the following items to ensure that gOOi can recognize this emulator:

- ◆ Make sure the directory where the emulator is installed is in your path (default = Program Files\walldata\RUMBA).
- ◆ When installing RUMBA, be sure to select the System Options bullet. If this bullet is not selected, API will not be available in the Options menu.
- ◆ Once the emulator is up, select Options ⇒ API ⇒ Identification and set the session short name. Under Configuration, check Convert null characters to spaces.

## KEA!

Check the following items to ensure that gOOi can recognize this emulator:

- ◆ Make sure you have KEA! 4.23 or later installed.
- ◆ The directory where KEA! is installed must be in the path ahead of any other emulator.
- ◆ The session must have a 1-character name (for example, A.KTC) in the USER subdirectory of KEA!
- ◆ Keyboard mapping within the emulator must be performed to enable PF key functions. To do this, select Options ⇒ Keyboard. This gives you the Keyboard Mapping Panel. Press Keyboard on the panel. A picture of the PC keyboard displays simultaneously with the Keyboard Mapping Panel.

The following describes how to map PC key F7 to PF7:

- a. On the Keyboard Mapping Panel, click the box labeled PC key. Press F7. F7 [76] should display in the box. The Map to: box should display Sequence.
- b. On the keyboard picture, click PF1 followed by the numeric 7. The box labeled Value should show <PF1><Keypad 7>. As you map keys, they are added to the Mapped Keys: box.

Repeat steps a and b to map the keys you want. When you are done, click **Done**. When you exit the emulator, save the changes to your session when prompted by KEA!

## Generic EHLLAPI

In addition to the other emulators listed in “[Emulators](#)” on page 297, gOOi also supports any emulator that is IBM EHLLAPI-compatible. Check the following items to enable gOOi support for such an emulator:

- ◆ gOOi must be installed using Generic EHLLAPI as the emulation option.
- ◆ The directory where the emulator is installed must be in the PATH statement.
- ◆ You must specify the name of your emulator’s DLL that supports EHLLAPI, along with the procedure name for EHLLAPI within this DLL.

Finding the correct DLL can require some experimentation. Many DLLs may be available, but not identified in the emulator documentation. Look for hll, hal, or hap in the file name. If you are unsure, contact your emulator vendor.



gOOi provides two generic EHLLAPI selections for the Host Connection because of inconsistencies in EHLLAPI implementations across emulators. The EHLLAPI selection works with some emulators that have a mixed implementation of the 16-bit and 32-bit EHLLAPI specifications. The EHLLAPI32 selection works with those emulators that have a complete 32-bit EHLLAPI interface. Contact Cincom if you are not sure which selection is appropriate for your emulator.

## Reflection

Check the following items to ensure that gOOi can recognize this emulator:

- ◆ The directory where the emulator is installed is in your path.
- ◆ The HLLAPI short session name is specified. You can verify this by selecting Setup ⇒ Terminal, or Setup ⇒ View Settings ⇒ HLLAPI Short Name.

## Reflection for UNIX and Digital

Check the following items to ensure that gOOi works with this emulator:

- ◆ The directory where the emulator is installed is in your path.
- ◆ The MFC42.DLL file is in your path. The REFLECT32.DLL module of gOOi for Reflection2 support requires this Microsoft DLL. Contact Cincom support if you do not have MFC42.DLL available on your system.

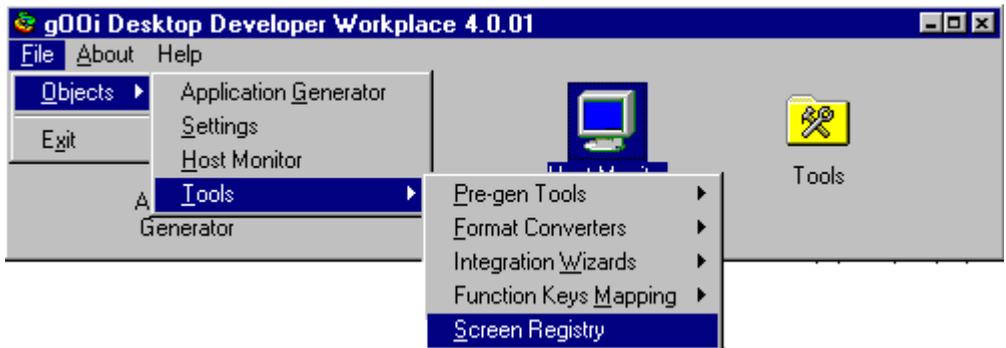
# C

## Screen Registry and AD/Advantage

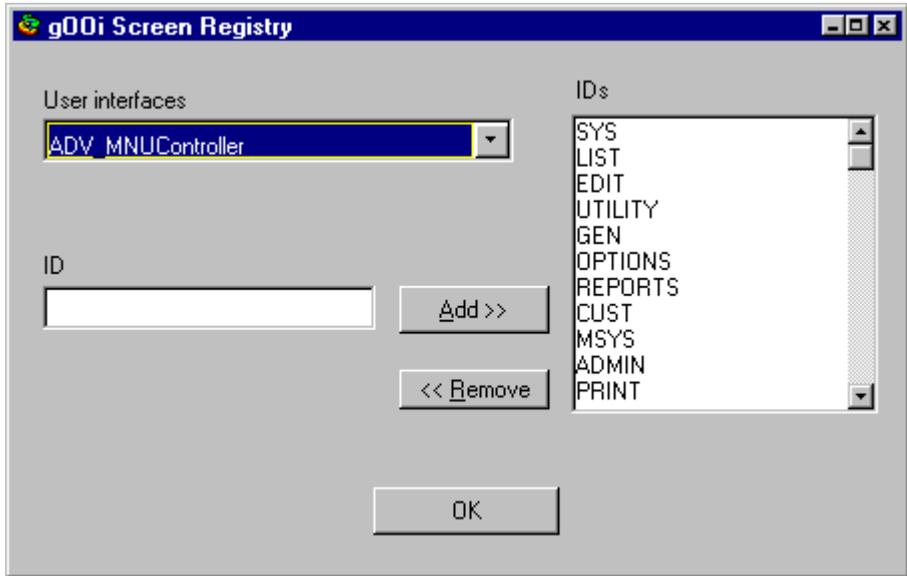
### Screen registry and AD/Advantage

The Screen Registry tool is necessary for AD/Advantage, and can also be used for other gOOi forms. By default, gOOi displays a form (ObjectStudio controller) with a name based on the screen ID. For example, screen ID CLSC1 is associated with the CLSC1Controller. Also by default, there is a one-to-one relationship between screen IDs and forms.

The registry allows you to override these default behaviors and associate one or more screen IDs with a user interface (ObjectStudio controller) of a name that does not match the screen ID. The Screen Registry is accessed from the cascaded menus available after choosing File from the gOOi Workplace menu:



After installing gOOi, you can see in the Screen Registry that the ADV\_MNUController that is included for AD/Advantage support is associated with many screen IDs:



This example shows how gOOi's ADV\_MNUController form is associated with multiple AD/Advantage system transactions such as SYS, LIST, EDIT, and UTILITY.

Outside AD/Advantage, the Screen Registry can be useful for gOOi forms generated via Dynamic Screen Capture. During the screen capture process, a screen ID must be specified. If an ID is entered that does not match the ID on the screen (or if a different ID is later added to the screen), you can associate the ID and form so that gOOi will recognize and display it. For example, say a screen ID of ABCDEF is assigned in Dynamic Screen Capture to a form with a screen ID of INV001. To associate INV001 with ABCDEFController, first choose ABCDEFController from the User Interfaces drop down list box. Specify INV001 in the Screen ID entry field and click the **Add>>** button. After the OK button is selected, gOOi registers the association and will then present the ABCDEFController when it encounters the INV001 screen ID.

# D

## gOOi class files and names

### gOOi classes

gOOi uses the following class files. The corresponding class names are shown in parentheses:

#### Emulator Communication class files

GOOIEALS.CLS	(GOOIEALSession)
GOOIKEAS.CLS	(GOOIKEASession)
GOOIXTRA.CLS	(GOOIXTRASession)
GOOIRUMB.CLS	(GOOIRUMBAsession)
GOOIHAPI.CLS	(GOOIEHLLAPISession)
GOOIRFLT.CLS	(GOOIREFLECTIONSession)
GOOIREFL.CLS	(GOOIREFLECTION2Session)
GOOIPC32.CLS	(GOOIPC3270Session)

#### Telnet class files

GOOITNPOOLDICTIONARIES.CLS	(GOOITelnetPoolDictionaries)
GOOITEMUSESS.CLS	(GOOInternalSession)
GOOISTRTRAN.CLS	(GOOStringTranslator)
GOOITELNETSESS.CLS	(GOOITelnetSession)

#### TN3270 class files

GOOIEMULATOR.CLS	GOOIAID.CLS	(GOOITN3270Aid)
(GOOI3270EmulatorController)		
GOOITELNETMGR.CLS	(GOOITN3270Manager)	
GOOITNCURSOR.CLS	(GOOITN3270Cursor)	
GOOITNDISPLAY.CLS	(GOOITN3270Display)	
GOOITNSTREAM.CLS	(GOOITN3270Stream)	

## TNVT class files

GOOIVTEMULATOR.CLS	(GOOIVTEmulatorController)
GOOIMUVTSESS.CLS	(GOOInternalVTSession)
GOOITELNETMGR.CLS	(GOOIVTEmulatorCustomController)
GOOITELNETVTSESS.CLS	(GOOITelnetVTSession)

## Generator class files

GOOIHSCR.CLS	(GOOIHostScreen)
GOOIHFLD.CLS	(GOOIHostField)
GOOIHAPP.CLS	(GOOIHostApplication)
GOOIHOST.CLS	(GOOIHostFile)
GOOIGNSC.CLS	(GOOIGenScreen)
GOOIGNAP.CLS	(GOOIGenApplication)
GOOIGNFD.CLS	(GOOIGenField)
GOOIBDAP.CLS	(GOOIHostInterfaceBuilder)

## Run-time class files

GOOISESS.CLS	(GOOISession)
GOOIHCTL.CLS	(GOOIHostController)
GOOIGENC.CLS	(GOOIGenericController)
GOOIGENCVT.CLS	(GOOIGenericVTController)
GOOIGENM.CLS	(GOOIGenericMenuController)
GOOIPOPP.CLS	(GOOIPopupController)
GOOIPRMP.CLS	(GOOIGenericPrompterController)

## gOOi user interfaces

GOOIBRWS.CLS	(GOOITemplateHierarchyBrowser)
GOOIFUNC.CLS	(GOOIFunctionKeysDefinition)
GOOIMONT.CLS	(GOOIHostMonitor)
GOOISPLT.CLS	(GOOIFileSplitter)
GOOICONF.CLS	(GOOIConfigurationSettings)
GOOIWORK.CLS	(GOOIWorkplace)
GOOI.CLS	(GOOIController)
GOOIMU.CLS	(GOOISelectEmulatorController)
GOOIFLDD.CLS	(GOOIFieldDetectorController)
GOOIPATDEF.CLS	(GOOIPatternsDefinitionController)
GOOIWIZA.CLS	(GOOIScreenIdLocatorController)

## Supporting classes

GOOIKEYPATTERN.CLS (GOOIKeyPattern)  
 GOOIWDWZ.CLS (GOOIWordWizardController)  
 EXCELOLE.CLS (GOOIExcelOLE)  
 GOOITBDEF.CLS (GOOIToolbarDefinitionController)  
 GOOIXLWZ.CLS (GOOIExcelWizardController)  
 GOOIADA.CLS (GOOIADATransactionsController)  
 GOOICONTACT.CLS (GOOIContactController)  
 GOOIRTLE.CLS (GOOIReTitleController)  
 GOOISERVERTIMEOUT.CLS (GOOIServerTimeoutController)  
 WORDOLE.CLS (GOOIWordOLE)

## Utility class

GOOISETP.CLS (GOOISetup)

## Just-In-Time classes

GOOIJIT.CLS (GOOIJITController)  
 GOOIJITC.CLS (GOOIJITCustomController)  
 GOOIJITM.CLS (GOOIJITMantisController)  
 GOOIJITP.CLS (GOOIJITPopupController)

## Messages

GOOIMSGS.CLS (Class GOOIMessages)

## BMS converter

BMS.CLS (Class GOOIBMS)  
 BMSFLD.CLS (Class GOOIBMSField)  
 BMSINPUT.CLS (Class GOOIBMSINPUTFILE)  
 BMSINT.CLS (Class GOOIBMSIntController)  
 BMSMAP.CLS (Class GOOIBMSMap)  
 BMSMSET.CLS (Class GOOIBMSMAPSET)  
 KEYWORD.CLS (Class GOOIBMKeyWord)  
 PARSER.CLS (Class GOOIBMParser)  
 STATEMT.CLS (Class GOOIBMStatement)  
 UEF.CLS (Class GOOIUEFObject)  
 UEFSCR.CLS (Class GOOIUEFScreen)  
 UEFSCR.F.CLS (Class GOOIUEFScreenField)

## MFS converter

MFS.CLS (Class GOOIMFS)  
MFSFLD.CLS (Class GOOIMFSField)  
MFSINPUT.CLS (Class GOOIMFSInputFile)  
MFSINT.CLS (Class GOOIMFSintController)  
MFSMAP.CLS (Class GOOIMFSMap)  
MFSMSET.CLS (Class GOOIMFSMapSet)  
MFSKEYWD.CLS (Class GOOIMFSKeyWord)  
MFSPARSE.CLS (Class GOOIMFSParser)  
MFSSTMT.CLS (Class GOOIMFSStatement)  
MFSUEF.CLS (Class GOOIMFSObject)  
MFSSCRN.CLS (Class GOOIMFSScreen)  
MFSSCRNF.CLS (Class GOOIMFSScreenField)

## UEF Generator

GOOIEFG.CLS (Class GOOIEFGeneratorController)  
GOOIEFL.CLS (Class GOOIEFFormFieldsListController)  
GOOIEFF.CLS (Class GOOIEFGenField)  
GOOIEFR.CLS (Class GOOIEFGenerator)  
GOOIEFP.CLS (Class GOOIEFGenApplication)  
GOOIEFS.CLS (Class GOOIEFGenScreen)  
GOOIEFC.CLS (Class GOOIEFControllerItemWrapper)

## Default templates

IBM3191.CLS (Class IBM3191Controller)  
IBM3191P.CLS (Class IBM3191PopupController)  
ASCIITRM.CLS (Class ASCIIterminalController)  
ASCIIPOP.CLS (Class ASCIIpopupController)  
GOOIVTTERMKEYS.CLS (Class GOOIVTterminalController)  
GOOIVTPOPUPKEYS.CLS (Class GOOIVTpopupController)  
REFL2TRM.CLS (Class Reflection2TerminalController)  
REFL2POP.CLS (Class Reflection2PopupController)

# E

## Rules for screen IDs and MANTIS prompter IDs

This appendix provides the rules and options for screen IDs and MANTIS prompter IDs for use with gOOi.

### Screen ID rules

A screen ID must have at least 2 characters (0–9, A–Z). The ID must begin with a letter and cannot have embedded or trailing blanks.

The following are examples of *valid* screen IDs:

- ◆ CLSC01
- ◆ CUSTOMERSCREEN

The following are examples of *invalid* screen IDs:

- ◆ 1 SCREEN (does not begin with a letter, and has an embedded blank)
- ◆ SCREEN\_TO\_DISPLAY (contains underscores)
- ◆ -SCREEN 1 (contains a hyphen, does not begin with a letter, and has an embedded blank)

## Screen ID options

gOOi attempts to recognize a screen ID at one of the profile locations specified in Settings. gOOi supports a number of options for identifying existing screen IDs. These options are illustrated in the following MANTIS examples:

- ◆ Your screen definition can be a field containing the text of the screen ID, having the exact position defined in Settings. For example:

```
FIELD "(
    POSITION(2,2)
    SIZE=5
    ATTRIBUTES(HED,MAS)
    MASK="CLSC1"
)
```

This is the definition of a screen ID at row 2, column 2, length 5, with the value CLSC1. In gOOi Settings, the screen ID row and column are each set to 2. gOOi looks for the string CLSC1 at location 2, 2.

- ◆ Your screen definition can be a field containing the text of the screen ID, *not* having exact position defined in Settings. This is considered an *embedded* screen ID:

```
FIELD "(
    POSITION(2,1)
    SIZE=30
    ATTRIBUTES(HED,MAS)
    MASK="*CLSC1***** Title Bar *****"
)
```

This example assumes that the screen ID row and column are each set to 2 in the gOOi Settings. The preceding field definition (row 2, col 1, length 30) overwrites the screen area where the screen ID is supposed to be. gOOi takes the part of this field that corresponds to the screen ID position (row 2, col 2, len 5). Thus, gOOi looks for the string CLSC1 at location 2, 2.

- ◆ Your screen definition can be a field that does not contain text because the value is set dynamically during run time. In this case, the value of the screen ID displayed must be the name of the screen object. For example:

```
SCREEN "CLSC1" (  
    LANGUAGE="ENGLISH"  
    DEVICE(24,80)  
    DESCRIPTION="Screen for new client"  
    DOMAIN(22,80)  
    ATTRIBUTES(BOT,NOF,NOA,WIN,AUW)  
    MASK_CHAR="#"  
    FILL_CHAR="|"  
    FIELD "SCREEN_ID" (  
        POSITION(2,2)  
        SIZE=5  
        ATTRIBUTES(HED,MAS)  
    )  
);
```

In this example, the screen ID field will be filled at run time. The value of this field must be set to CLSC1, which is the name of the screen. gOOi looks for the string CLSC1 to display at location 2, 2.

## Prompter ID rules

Since MANTIS prompters do not contain fields, the prompter ID must be displayed between parentheses at the end of the prompter description. The ID must be an alphanumeric string of at least 2 characters, starting with a letter. For example:

```
PROMPTER "CLPR01" (  
    LANGUAGE="ENGLISH"  
    DESCRIPTION="Information about Client (CLPR01)"  
    NEXT_PROMPTER="CLPR02"  
    ...
```

gOOi uses the description text displayed on the first line of the emulator to define which prompter to open. In this example, the screen ID of the prompter is CLPR01.

# F

## Host-PC translation tables

### Translation tables

The key mapping included in the default IBM3191Controller and default IBM3191PopupController is shown in the shaded portions of the following table:

Host key	ASCII key	Host key	ASCII key	Host key	ASCII key
Alt	@A	Cursor Down	@V	PF13	@d
Backtab	@B	Not used	@W	PF14	@e
<b>Clear</b>	<b>@C</b>	Reserved	@X	PF15	@f
Delete Char	@D	Caps Lock	@Y	PF16	@g
<b>Enter</b>	<b>@E</b>	Cursor Right	@Z	PF17	@h
<b>Erase EOF</b>	<b>@F</b>	Backspace	@\	PF18	@i
Help	@H	<b>Home</b>	@0	PF19	@j
Insert	@I	PF1	@1	PF20	@k
Jump	@J	PF2	@2	PF21	@l
Copy	@K	PF3	@3	PF22	@m
Cursor Left	@L	PF4	@4	PF23	@n
Enlarge	@M	PF5	@5	PF24	@o
New Line	@N	PF6	@6	PA1	@x
Print	@P	PF7	@7	PA2	@y
Finish (Quit)	@Q	PF8	@8	PA3	@z
<b>Reset</b>	<b>@R</b>	PF9	@9	SysReq	@A@H
Shift	@S	PF10	@a	Attn	@A@Q
Tab	@T	PF11	@b	CurSet	@A@J
Cursor Up	@U	PF12	@c	Erlnp	@A@F

The key mapping included in the default GOOIVTTerminalController and default GOOIVTPopupController is as follows:

Function key	Host sequence	Function key	Host sequence
F1	F1, 1	F14	F2,1,4
F2	F1, 2	F15	F2,1,5
F3	F1, 3	F16	F2,1,6
F4	F1, 4	F17	F2,1,7
F5	F1, 5	F18	F2,1,8
F6	F1, 6	F19	F2,1,9
F7	F1, 7	F20	F2,2,0
F8	F1, 8	F21	F2,2,1
F9	F1, 9	F22	F2,2,2
F10	F2,1,0	F23	F2,2, 3
F11	F2,1,1	F24	F2,2,4
F12	F2,1,2	Enter	Enter6
F13	F2,1,3	Cancel	Cancel,

The key mapping included in the default ASCII TerminalController is as follows:

Function key	Host sequence
F1	PF1, 1
F2	PF1, 2
F3	PF1, 3
F4	PF1, 4
F5	PF1, 5
F6	PF1, 6
F7	PF1, 7
F8	PF1, 8
F9	PF1, 9
F10	PF2, 1, 0
F11	PF2, 1, 1
F12	PF2, 1, 2
PAGEDOWN	^Z

The key mapping included in the default ASCII PopupController is as follows:

Function key	Host sequence
F7	PF1, 7
F8	PF1, 8
PAGEDOWN	^Z

The key mapping included in the default Reflection2TerminalController is as follows:

Function key	Host sequence
F1	870,831
F2	870,832
F3	870,833
F4	870,834
F5	870,835
F6	870,836
F7	870,837
F8	870,838
F9	870,839
F10	871,831,830
F11	871,831,831
F12	871,831,832
PAGEDOWN	870,867

The key mapping included in the default ASCIIPopupController is as follows:

Function key	Host sequence
F7	870,837
F8	870,838
PAGEDOWN	870,867

# G

## gOOi error messages

### Error messages

This appendix lists the gOOi error messages. These messages are in the GOOIMessages class as a set of Smalltalk pool dictionaries.

---

### GOOIMessages class

The first 4 characters of each message is the identifier (for example, HOST for HOST0003). It can be referenced against the gOOi subdirectory of ObjectStudio to find the class that issues the error.

**BDAP0004**     **Screen ID is missing or invalid. Please check screen definition <screen-name>.**

**Explanation** This is a warning message. gOOi encountered a screen during generation that is missing the screen ID at the specified location, or the screen ID at that location is invalid.

**Action** Add or correct the screen ID for the specified screen, or utilize the *screenChangedOn:* method of host navigation to recognize the screen.

**BRWS0001**     **You cannot edit keys for <controller-name>.**

**Explanation** When using the Template Hierarchy Browser, you tried to modify either the GOOIGenericController or the GOOIPopupController.

**Action** Create a subclass of these controllers and modify this new subclass.

**BRWS0002**      **The class <class-name> already exists.**

**Explanation** When using the Template Hierarchy Browser, you specified a new class name that already exists.

**Action** Choose a different class name that does not conflict.

**BRWS0006**      **<class-name> is not a valid class name.**

**Explanation** When using the Template Hierarchy Browser, you specified an invalid class name.

**Action** Check the class name spelling.

**CONF0001**      **Horizontal Repeat Spacing Factor must be greater than zero.**

**Explanation** In Generation Options, you entered 0 as the Horizontal Repeat Spacing Factor.

**Action** Change this value so that it is greater than 0.

**CONF0002**      **Please enter a valid row number. Row number range is 1 - 43.**

**Explanation** An invalid row number was specified for the screen ID row in Settings.

**Action** Specify a value between 1 and 43.

**CONF0004**      **Please enter a valid column number. Column number range is 1 - 132.**

**Explanation** An invalid column number was specified for the screen ID column in Settings.

**Action** Specify a value between 1 and 132.

**EMU0001**      **< item > must be specified to start the emulator.**

**Explanation** One of the emulator items under the Advanced dialog of HostConnections is missing. The item is either the Emulator .exe file, Emulator session file, or Emulator session ID.

**Action** Contact Cincom support for assistance.

- EMU0002**      **Emulator support unavailable.**
- Explanation** The class file for support of the selected emulator could not be loaded.
- Action** Contact Cincom support for assistance.
- FUNC0001**      **<syntax-error> at line <line-number>.**
- Explanation** When defining function keys, you entered Smalltalk method code that contained a syntax error.
- Action** Change the indicated line of Smalltalk code.
- FUNC0002**      **You must select a controller in the list.**
- Explanation** When defining function keys, you attempted a Generate without specifying a controller.
- Action** Enter a controller name.
- FUNC0003**      **No mapping is defined.**
- Explanation** When defining function keys, you attempted a Generate without specifying menu headers.
- Action** Enter at least one menu header.
- FUNC0004**      **This is not a valid import file.**
- Explanation** The import file that was specified for the menu option in Function Keys Definition does not contain a valid key mapping.
- Action** Check that the correct file was selected.
- GOOI0001**      **You must first select the location of your files.**
- Explanation** Before you can add form elements, you must define the directory where the extract files are located.
- Action** Click **Browse** to open the directory selection window or select the Browse extract path in the Options menu.

**GOOI0002      The application name must be entered.**

**Explanation** The Application Generator needs an application name to create a folder for the generated forms and an application file to load the generated forms.

**Action** On the Application Generator window, enter a name in the Application field (maximum eight characters).

**GOOI0003      Parent class <class name> is not loaded.**

**Explanation** The classes that are defined in the Settings interface (in the Generation Options window) are not loaded into the system.

**Action** Verify/perform both of the following tasks:

- ◆ Use the ObjectStudio Load file... or Load application... menu to load these classes.
- ◆ With the Settings interface, set the parent class to another class that is loaded (for IBM users, the default is IBM3191Controller for regular forms).

**GOOI0005      Popup parent class <class name> is not loaded.**

**Explanation** The classes that are defined in the Settings interface (in the Generation Options window) are not loaded into the system.

**Action** Verify/perform both of the following tasks:

- ◆ Use the ObjectStudio Load file... or Load application... menu to load these classes.
- ◆ With the Settings interface, set the pop-up parent class to another class that is loaded (for IBM users, the default is IBM3191PopupController for pop-up forms).

**GOOI0006      This is not a valid layout file!**

**Explanation** You have tried to load a layout file that does not have a valid structure (selecting File ⇒ Open layout).

**Action** You can edit the file and verify that the first line contains the following text: "gOOi LAYOUT DEFINITION".

- GOOI0007**     **A popup must be defined with a Screen.**
- Explanation** You have tried to add a pop-up element into an empty form definition. Pop-up screens can only be defined with one or more Screen elements.
- Action** You must add a Screen element before you can attach a pop-up element to it.
- GOOI0008**     **You cannot define two <Menu|Prompter>s with the same Form ID.**
- Explanation** You have tried to add a menu or a prompter to a Form ID that already contains one of these elements. Menus or prompters must have their own Form IDs and cannot be mixed with other screen elements.
- Action** Select another Form ID in the list and repeat the add action.
- GOOI0011**     **You cannot define a <Menu|Prompter> and other components with the same Form ID.**
- Explanation** You have tried to add a menu or a prompter to a Form ID that already contains another element. Menus or prompters must have their own Form ID and cannot be mixed with other screen elements.
- Action** Select another Form ID in the list and repeat the add action.
- GOOI0012**     **You cannot define a <Screen|Header|Footer|Popup> and a <Menu|Prompter> with the same Form ID.**
- Explanation** You have tried to add a screen, header, footer, or pop-up to a Form ID that already contains a menu or a prompter.
- Action** Select another Form ID in the list and repeat the add action.
- GOOI0014**     **This extract file does not contain a prompter definition.**
- Explanation** You have tried to add a file that does not contain a prompter definition.
- Action** Verify that your .exp file contains a “PROMPTER” string at the beginning.

**GOOI0016**      **Unable to open file <file-name>.**

**Explanation** gOOi could not open the loadable applications list file.

**Action** Check to see whether the loadable applications list file is installed. The file name is lappiniw.cls, which is located under the W32\CLS subdirectory of ObjectStudio.

**GOOI0017**      **Unable to update loadable application init file.**

**Explanation** gOOi could not open the loadable applications list file.

**Action** gOOi was unable to add the application to the loadable application init file lappiniw.cls. Ensure that the file in your environment is not read-only.

**GOOI0018**      **For a pop-up, the CONVERSE must be at (1,1).**

**Explanation** The dynamic CONVERSE feature is not available for pop-ups.

**Action** Use only the default of 1@1 for pop-ups.

**GOOI0024**      **Log file size is greater than <file size>.**

**Explanation** This is a warning message that the size of the Application Generator log file is greater than the size set in the log file options.

**Action** Clear the log file, first saving the contents if desired.

**GOOI0026**      **Error(s) occurred - select OK to review the log file details..**

**Explanation** This message box is displayed when the Application Generator stops because of an error..

**Action** Go to the most recent data at the end of the log file to locate the error.

- GOOI0027**      **Error(s) occurred - please enable log file via the Menu, then rerun the Generate..**
- Explanation** This message box is displayed when the Application Generator stops because of an error, and the log file is not enabled.
- Action** Ensure that the log file is enabled so that the error will be captured, then run the Application Generator again.
- GOOI0028**      **<file-name> is already included in Form <form number>.**
- Explanation** The file that you are trying to add to a form is already included in the form.
- Action** Verify the contents of the form.
- GOOI0033**      **Empty form already exists.**
- Explanation** You are trying to add another form to an application that already has an empty form available for adding items.
- Action** Use or delete the current empty form.
- GOOI0035**      **Cannot remove all forms - Stop Generation to exit.**
- Explanation** You cannot use selective generation to remove all the forms in the Application.
- Action** Run the Application Generator with at least one form selected for generation.
- GOOI0036**      **Warning(s) occurred - select OK to review the log file details.**
- Explanation** This message is displayed at the end of Application Generation if a warning condition occurs during the generation processing.
- Action** Go to the end of the log file to view the results from the last generation.

**HOST0002** File missing or path invalid.

**Explanation** gOOi could not open the loadable applications list file.

**Action** Check to see whether the loadable applications list file is installed. The file name is lappiniw.cls, which is located under the W32\CLS subdirectory of ObjectStudio.

**HOST0005** gOOi does not support HORIZONTAL 255 fields.

**Explanation** The extract file that you want to generate contains a field definition with a horizontal repeat of 255, which means a dynamic setting of this option at run time.

**Action** None. This feature is not supported by the gOOi Application Generator.

**MONT0001** EAL Error : Unable to retrieve sessions list.

**Explanation** *OpenVMS/UNIX and IBM emulators:* The HostMonitor has tried and failed to connect to the emulator in order to obtain a list of the running sessions.

**Action** Verify/perform all of the following:

- ◆ Verify that your session has a “short name” (A-Z).
  - Using KEA!, it means that your configuration file (.KTC) must have a name with 1 unique character.
  - Using EXTRA!, it means that you must attach a short name to your session (Options > Global Preferences > Advanced).

With other emulators, consult the emulator documentation.

- ◆ Verify that the directory of your emulator is in the path.
- ◆ Verify that your emulator is started and running. The HostMonitor will only retrieve opened sessions.
- ◆ If you are using an IBM host emulator, verify that you have selected the proper emulator in the HostMonitor.

**MONT0003**      **Unable to initiate conversation with session <session name>.**

**Explanation** *OpenVMS/UNIX only.* The HostMonitor has tried and failed to initiate a DDE conversation with KEA!

**Action** Verify both of the following items:

- ◆ Verify that your session has a “short name” (A–Z). Your configuration file (.KTC) must have a name with 1 unique character.
- ◆ Verify that your emulator is started and running. The HostMonitor will only retrieve opened sessions.

**MONT0005**      **EAL Error < n >. Unable to register client for <session-name>.**

**Explanation** You attempted to register the gOOi client with the emulator and failed.

**Action** Verify/perform all of the following:

- ◆ Verify that your session has a “short name” (A–Z).
  - Using KEA!, it means that your configuration file (.KTC) must have a name with 1 unique character.
  - Using EXTRA!, it means that you must attach a short name to your session (Options > Global Preferences > Advanced).

With other emulators, consult the emulator documentation.

- ◆ Verify that the directory of your emulator is in the path.
- ◆ Verify that your emulator is started and running. The HostMonitor will only retrieve opened sessions.
- ◆ If you are using an IBM emulator, verify that you have selected the proper emulator in the HostMonitor.

**MONT0006**      **EAL Error < n >. Unable to connect client for <session-name>.**

**Explanation** You attempted to establish a connection to an emulator session and failed.

**Action** Verify/perform all of the following:

- ◆ Verify that your session has a “short name” (A-Z).
  - Using KEA!, it means that your configuration file (.KTC) must have a name with 1 unique character.
  - Using EXTRA!, it means that you must attach a short name to your session (Options > Global Preferences > Advanced).

With other emulators, consult the emulator documentation.

- ◆ Verify that the directory of your emulator is in the path.
- ◆ Verify that your emulator is started and running. The HostMonitor will only retrieve opened sessions.
- ◆ If you are using an IBM emulator, verify that you have selected the proper emulator in the HostMonitor.

**SPLT0007**      **Loop condition detected! Please ensure the UEF file has the proper format.**

**Explanation** The File Splitter did not reach the end of file during parsing.

**Action** This error usually occurs because the UEF format is different between IBM mainframe MANTIS and OpenVMS/UNIX MANTIS. IBM mainframe MANTIS requires the line, ‘MANTIS UEF 110n’ at the beginning of the input file, where n is a number between 4 and 9.

**SPLT0008**      **File could not be created, error code: n.**

**Explanation** The File Splitter could not create individual files from the UEF input file. The error code is the return code received by ObjectStudio from its file creation request to Windows.

**Action** This error might be because you have no available space on your disk. Another possible cause is that the folder where ObjectStudio tries to create the file is write protected.

- SPLT0009**      **UEF File could not be accessed, error code: n.**
- Explanation** The File Splitter could not locate the specified input UEF file.
- Action** Check the spelling of the file name.
- SPLT0010**      **<file name> is not in UEF format. Splitting is not possible.**
- Explanation** The specified input file does not contain data in UEF format.
- Action** Check the contents of the specified file.
- WDWZ0001**      **You must enter a method name.**
- Explanation** The Word Wizard requires a name for the method to be generated in the specified controller.
- Action** Specify a method name.
- WDWZ0002**      **This is not a legal method name. Try: < name >.**
- Explanation** The specified method name contains is invalid. Ensure that the name does not includes only alphanumeric characters
- Action** Specify a different method name.
- WDWZ0003**      **You must enter a button label.**
- Explanation** If you request that the Word Wizard create a button in the controller, you must specify a label for this button.
- Action** Specify a label name for the button.
- WDWZ0004**      **This method already exists.**
- Explanation** The Word Wizard cannot create the specified method because it already exists.
- Action** Specify a different method name.

**WDWZ0005** You must select a Word field.

**Explanation** You selected a controller field and clicked the **Link** button without selecting a corresponding Word field.

**Action** Select a Word field from the list.

**WDWZ0006** You must select a controller field.

**Explanation** You selected a Word field and clicked the **Link** button without selecting a corresponding controller field.

**Action** Select a controller field from the list.

**.WDWZ0008** You must select a Word .doc file.

**Explanation** When using the Word Wizard, you must select a Word document before choosing the Next button.

**Action** Select a Word document.

**WDWZ0009** You must enter the character used for field identification.

**Explanation** When using the Word Wizard, you must select a character that precedes every field name before choosing the Next button.

**Action** Select a field definition character.

**WDWZ0012** <item> does not exist in the language dictionary.

**Explanation** When using the Word Wizard, a parameter could not be located in the language dictionary.

**Action** Contact Cincom support for assistance.

# H

## Using the UEF Generator

### Using the UEF Generator

The Universal Export Facility (UEF) Generator tool is for MANTIS users who wish to create a UEF file from their gOOi form(s), then upload this file to the host and import it into MANTIS.

**Generate UEF File from gOOi Form**

UEF File to be generated:

Type of Host System:

**Define Host Screen for gOOi Form**

Select from gOOi forms currently loaded:

gOOi Form Name	Screen Name	Screen ID	Description
----------------	-------------	-----------	-------------

Screen Name on Host:

Screen Description on Host:

Screen ID:

To use the UEF Generator, perform the following steps:

1. Specify the name of the UEF file to be generated.
2. Select the type of host system: IBM mainframe, OpenVMS, or UNIX.
3. Select the gOOi form from which the UEF file will be generated.
4. Specify the MANTIS screen name that will be used for the screen on the host.
5. Specify the MANTIS screen description that will be given to the screen on the host.
6. If the gOOi form does not already contain a screen ID, specify one to be added to the UEF file. This ID will be added at the location indicated by the current profile.
7. Click **Add** to insert this information into the generation list.
8. Repeat steps 3 through 7 for each form from which a UEF file is to be generated.
9. Click **Generate** to create the UEF file.

You can preview which fields are to be generated for a form by clicking **Fields**. The Form Fields List window shows all the fields on the form and whether a UEF field will be generated for each field. You can override the YES/NO setting in the Generate column by clicking **Set**.

# Glossary of terms

## **button**

An object that the user can press (click) to initiate an action. Contains either text or an image (bitmap).

## **check box**

Presents the user with a choice.

## **class**

An ObjectStudio object that determines the structure and behavior of objects called instances.

## **class browser**

A utility window in ObjectStudio that lets you browse the class hierarchy, view the list of methods for a class, and create and modify classes and methods.

## **class method**

A method associated with a class object, executed by sending a message to the class.

## **class variable**

A variable defined in a class, and shared by all instances of that class.

## **controller**

A class in ObjectStudio that manages communications between the user and the interface.

## **form**

A GUI window that is moveable and sizable. A form has a title bar and can have a menu bar, toolbar, and status line.

## **global variable**

A variable that is available to all objects in the Smalltalk environment.

## **inheritance**

A characteristic of an object-oriented language that allows a class to share all methods of all classes that reside above it in the class hierarchy.

## **instance**

An instance of a class is an object that was created by the class, and whose behavior and internal structure are determined by the class. For example, 7 is an instance of the Integer class.

## **instance variable**

An internal variable of an instance that is not shared with any other object. Instance variables are used to store data for the object.

## **instantiation**

Creation and allocation of memory for an object.

## **lassoing**

The technique of clicking and holding down mouse button 1 while dragging toward the lower right until all target fields are selected.

## **literal**

A general term for constants that are explicitly constructed in source code, such as numbers and strings.

## **local variable**

A temporary variable defined in a method, delimited by vertical bars.

## **message**

A request to an object to carry out an operation. All actions in Smalltalk are initiated via messages.

**method**

A named procedure that performs an operation. A method consists of temporary (local) variables and code.

**nil**

The only instance of UndefinedObject, the nil object represents an undefined value. All variables are initialized to nil.

**object**

An object is the fundamental building block of ObjectStudio. An object is a self-contained entity that has its own memory, data, and behavior.

**receiver**

The object to which a message is directed. In the code, self 'pressENTER', self is the receiver.

**self**

A pseudo-variable in Smalltalk that is used to send messages from an object to itself.

**super**

A pseudo-variable in Smalltalk that is used to send messages from an object to the superclass of the object.

**superclass**

A class from which subclasses inherit data and behavior. Also known as a parent class.

**symbol**

A symbol is a special kind of string that is read only. Unlike strings, symbols with the same value are identical (the same object).



# Index

## A

- AD/Advantage
  - considerations 301
  - Footer option 135
  - headers and footers 79, 135
  - hiding key descriptions 62
  - tool description 22
- ADV\_MNUController 302
- Application controllers field 264
- Application field 264
- Application Generator 21
- application layout, saving 145
- Application name field 264
- applications
  - deploying 259
  - development process 23
  - generating
    - forms 141
    - same several times 145
  - integrating
    - with Excel 153
    - with Word 149
  - testing 109
  - updating loadable 145
- ASCII
  - considerations 98
  - keys 311
- ASCIIPopupController 66
- ASCIITerminalController 63
- attributes
  - color 60
  - FUL 129
  - hidden 60
  - high intense 60
  - matching forms and host screens 59
  - protected 60
  - reverse 60
  - setting 60
  - underline 60

## B

- behaviors, inherited 93, 103
- bitmap, of logo 264
- BMS 24, 82
- BMS Converter 22
- buttons
  - example of event-driven customization
    - creating an event-driven button with an existing Smalltalk method 203
    - creating an event-driven button with Smalltalk 206
    - introduction 203
  - labels, pattern definition 75
  - spin buttons for numeric fields 191

## C

- cell coordinates, displaying 90
- character, height and width 62
- check box, assigning set method 176
- class files
  - and forms 25
  - BMS converter 305
  - communication 303
  - default templates 306
  - generator 304
  - messages 305
  - MFS converter 306
  - run-time 304
  - support 305
  - Telnet 303
  - TN3270 303
  - TNVT 304
  - UEF Generator 306
  - used by gOOi 303
  - user interface 304
  - utility 305
- colors
  - field
    - background and foreground 62
    - changing 169
    - set attribute 60
    - properties 68
    - window background 62
  - column, screen ID 46

- command line, including 62
- components
  - of forms 127
  - order specification 134
- configuration
  - procedure 41
  - screen settings 45
- considerations
  - AD/Advantage 301
  - emulators 297
  - IBM platform-specific 293
  - OpenVMS
    - Connect timeout value 53
    - Emulator Check Intervall 56
    - Host address 53
    - Just In Time GUI
      - Just In Time window 56
    - Pause value 55
    - Port number 53
    - Screen Tag option 58
- controllers
  - creating a pop-up subclass 96
  - deleting unnecessary 264
  - making parent class for
    - generation 108
  - new 100
  - saving a function key map
    - subclass 96
- Controllers to be removed field 264
- CONVERSE 49
- coordinates of a field, displaying 90
- Copy.Exe File field 264
- CUSTOM directory 96

- customization
  - changing
    - field color and font 169
    - field presentation type 172
    - field size automatically 167
    - field size manually 167
  - editing a form 162
  - example of event-driven customization
    - creating an event-driven button with an existing Smalltalk method 203
    - creating an event-driven button with Smalltalk 206
    - introduction 203
  - grouping fields 180
  - moving fields 164
  - overview 159

## D

- data, transferring
  - to Excel 153
  - to Word 149
- Delete field 264
- deployment of a gOOi application 259
- Designer
  - reference resource 25
  - toolbar, viewing 163
- development process 23
- directory, UEF files 131
- display resolution, recommended setting 126, 140
- down arrow key 33
- dynamic
  - fields, vertical repeat 119
  - inheritance 104
- Dynamic CONVERSE 129, 136
- Dynamic Screen Capture
  - control description 87
  - description 21
  - process description 86

**E**

- EHLLAPI emulation 299
- emulators
  - EXTRA! 298
  - Generic EHLLAPI 299
  - KEA! 298
  - PC3270 297
  - recognizing 297
  - Reflection 300
  - Reflection2 300
  - selecting 51
- ENFIN, how it works with gOOi
  - 24
- error messages 315
- Escape sequence, specifying 52
- ESDS cluster, UEF images on
  - 117
- event-driven customization
  - example
    - creating an event-driven button
      - with an existing Smalltalk
        - method 203
      - creating an event-driven button
        - with Smalltalk 206
    - introduction 203
- Excel Wizard
  - description 22, 153
  - matching gOOi form fields 154
  - using 154
- exp files
  - creating 118
  - for MANTIS objects 119
- exporting
  - key map text file 94
  - prompts 294
  - screens 294
- EXTRA! emulator 298
- extract file
  - selecting 136
  - types 127

**F**

- fields
  - aligning 165
  - attributes 59
  - changing
    - color and font 169
    - field size automatically 167
    - field size manually 167
  - excluding from forms 143
  - grouping graphically 180
  - identifying
    - in Excel spreadsheet 154
    - in Word document 150
  - item consistency 176
  - linking
    - to Excel 155
    - to Word 151
  - marking positions 89
  - matching gOOi form
    - to Excel spreadsheet 154
    - to Word document 151
  - moving 164
  - presentation type, changing
    - 172
  - tab order 159
  - viewing on forms 146
- File Splitter
  - creating .exp files with 118
  - description 21
  - usage rules 85
- files
  - directory for UEF 131
  - extract considerations 127
  - logo bitmap 264
  - overriding 145
  - overwriting with same name
    - 122, 124
  - pmt 80
  - saving to the CUSTOM
    - directory 96

- folder, temporary storage of
  - forms 146
- fonts
  - field, changing 169
  - properties 67
  - specifying 62
- Footer option 135
- footers, AD/Advantage 79
- format, of new item 176
- forms
  - and class files 25
  - changing
    - field color and font 169
    - field presentation type 172
    - field size automatically 167
    - field size manually 167
  - components, specifying 130
  - consistent format of items 176
  - customizing
    - bringing a form into ObjectStudio Designer 161
    - grouping fields graphically 180
    - overview 159
  - default format 24
  - effects of screen resolution 126, 140
  - example of event-driven customization
    - creating an event-driven button with an existing Smalltalk method 203
    - creating an event-driven button with Smalltalk 206
    - introduction 203
  - Footer option 135
  - for AD/Advantage transactions 302
  - generating
    - overview 125
    - procedure 141
    - restrictions 127
    - setting options 62
  - grouping fields graphically 180
  - headers and footers 135
  - inherited visual items 103
  - item traversal 159
  - keeping open 50
  - MANTIS, status line 129
  - Menu options
    - data items 291
    - description 136
    - equivalent MANTIS and PC data types 278
    - sequential DIF tables 287
  - moving fields 164, 166
  - Pop-up option 135
  - previewing 139
  - profile verification 143, 144
  - Prompter option 136
  - reducing display size 163
  - sample specifications 139
  - Screen option 135
  - setting configuration options 27, 41
  - setting field attributes 60
  - specifying components 127
  - viewing
    - generated, with host connection 148
    - generated, without host connection 147
    - overview 146
    - ungenerated, using Preview 146
- FTP, copying UEF images with 117
- FUL attribute 129
- function keys
  - accessing from menus 24
  - assigning to host commands 98
  - creating a template subclass 94
  - executing 93
  - mapping 91
  - specifying a map 63
  - translation tables 311
  - using hotspots 72
- Function Keys Definition 22
- function keys option 32

**G**

- gap, adding between toolbar items 79
- generation log 140
- generation options for forms 45, 62
- Generic EHLLAPI emulator 299
- gOOi
  - class files 303
    - BMS converter 305
    - communication 303
    - default templates 306
    - generator 304
    - messages 305
    - MFS converter 306
    - runtime 304
    - support 305
    - Telnet 303
    - TN3270 303
    - TNVT 304
    - UEF Generator 306
    - user interface 304
    - utility 305
  - component descriptions 21
  - customizing interfaces
    - changing field color and font 169
    - changing field presentation type 172
    - changing field size automatically 167
    - changing field size manually 167
    - editing a form 162
    - example of event driven customization, introduction 203
    - example of event-driven customization, creating an event-driven button with an existing Smalltalk method 203
    - example of event-driven customization, creating an event-driven button with Smalltalk 206
    - grouping fields graphically 180
    - moving fields 164, 166
    - overview 159
    - deploying an application 259
    - description 19
    - error messages 315
    - forms, generating 134
    - functionality description 24
    - generating application forms 141
    - loading 42
    - match form fields
      - to Excel spreadsheet 154
      - to Word document 151
    - relationship to ENFIN 24
    - sample form specifications 139
    - Tools
      - Excel Wizard 153
      - Word Wizard 149
    - Workplace 45
  - gooi.ini file
    - copying in order to deploy application 266
    - editing to remove, replace, or insert a pattern 36
    - prompter files path 80
    - saving gOOi generation options in 45
    - saving profile information in 46
  - GOOIGenericController 93
  - GOOIPopupController 93

**H**

- headers and footers
  - AD/Advantage 79, 135
  - AD/Advantage requirements 132
- hidden field attribute 60
- high intense field attribute 60
- home keys 33
- host
  - application
    - capturing screens 86
    - screen settings 45
  - attributes 59
  - commands, defining function keys 98
  - fields, excluding from generated forms 143
  - keys 311

**Host Monitor**

- connection to host 112
- description 21
- using 112

**HostObject 126**

host-PC translation tables 311

hotspots 32, 72

**I****IBM**

- capturing host screens 86
- considerations 98, 293

**IBM3191Controller**

- description 63
- function key inheritance 93

**IBM3191PopupController 66****identifiers**

- for prompters 118
- for screens 82, 118, 307

IDs, MANTIS rules 307

Image filename field 264

img file 264

import, key map text file 94

**inheritance**

- dynamic 104
- function key map 91
- static 104
- visual items 103

**initialization file**

- copying in order to deploy
  - application 266
- editing to remove, replace, or insert a pattern 36
- prompter files path 80
- saving gOOi generation options in 45
- saving profile information in 46

installation prerequisites 20

**integration wizards**

- Excel 153
- Word 149

**interfaces**

- generating 126
- inherited visual items 103
- setting up 41

item traversal sequence 159

**J**

JIT pop-up 34

**K**

KEA! emulator considerations 99, 298

**key map**

- creating 91
- creating a template subclass 94
- sample templates 63
- storing as text 94

key simulation 67

keys, host and ASCII 311

keystrokes, defining 98

**L**

layout, application, saving 145

length, screen ID 46

loadable application, updating 145

loading gOOi 42

log file 140

Logo filename field 264

**M****MANTIS**

command line/key simulation 67

Dynamic CONVERSE 49

rules for prompter IDs 307

Screen Design Facility 115

**Menu options**

data items 291

description 136

equivalent MANTIS and PC

data types 278

format 127

sequential DIF tables 287

**method name**

Excel Wizard controller 156

Word Wizard controller 152

Method source 264

## methods

- assigning to a button in an
    - example of event-driven customization
  - creating an event-driven button with an existing Smalltalk method 203
  - creating an event-driven button with Smalltalk 206
  - introduction 203
- MFS 22, 24, 82

**N**

## names

- prompters, MANTIS 310
  - screens 307
- new controllers 100
- nonproportional fonts 68

**O**

## objects, visual 104

## ObjectStudio

- copy executable during generation 264
  - required software 20
- OpenVMS/Alpha considerations
- Connect timeout value 53
  - Emulator Check Interval 56
  - Host address 53
  - Just In Time window 56
  - Pause value 55
  - Port number 53
  - Screen Tag option 58
- OpenVMS/VAX considerations
- connect timeout value 53
  - Emulator Check Interval 56
  - Host address 53
  - Just In Time window 56
  - Pause value 55
  - Port number 53
  - Screen Tag option 58
- options, form generation 45
- overwriting files 122, 124

**P**

## parent class

- for form generation 108
  - selecting 62
- Parent class name field 63
- pattern definition, for button label 75
- Pause, specifying 45
- pmt files 80
- pop-ups
- File Type options 135
  - identifiers 82
  - JIT 34
  - parent class name, setting 66
  - undefined 119
  - ungenerated 135
- prerequisites 20
- presentation
- properties, setting 71
  - type, changing for a field 172
- preview
- of form before generating 146
- preview of form before generating 139, 146
- product description 19
- profile
- default 46
  - for screen ID 45
  - naming guidelines 46
  - usage recommendations 47
- prompter
- file, pointing to 80
  - ID, MANTIS rules 307
- Prompter option 136
- prompters
- duplicates 123
  - ensuring one per file 119
  - exporting 294
  - identifiers 118
  - naming, MANTIS 310
  - set file path 45
- properties
- color 68
  - fonts 67
  - presentation, setting 71
- protected, field attribute 60

**R**

- Reflection emulator 300
- Reflection2 emulator 300
- requirements, for installation 20
- restrictions
  - for MANTIS applications 82, 118
  - form generation 127
- reverse, field attribute 60
- row, screen ID 46
- runtime
  - entities, storing 266
  - image of application 264

**S**

- samples
  - gOOi form specifications 139
  - templates 63
- saving in gOOi.ini file
  - gOOi generation options 45
  - profile information 46
- Screen Design Facility,
  - downloading UEF images with 115
- screen ID
  - displaying 129
  - locating 89
  - MANTIS rules 307
  - profile 45
  - setting 46
- Screen ID Locator
  - description 21
  - process description 89
- Screen option 135
- screens
  - as menus 127
  - capturing 86
  - components 127
  - configuration 45
  - duplicates 123
  - ensuring one per file 119
  - exporting 294
  - identifiers 82, 118, 307
  - location of identifier 49
  - MANTIS 129
  - maximum size 127
  - multiple components 134

- rules for naming 307
- side-by-side 82, 118, 127
- sizes supported 82, 118
- specifying components 134
- specifying order 134
- undefined 82, 119
- selection head 165
- Settings tool 21
- Settings window 45
- settings, screen configuration 45
- side-by-side screens 82, 118, 127
- size
  - host screen 127
  - of form 163
- Smalltalk
  - code implementation 25
  - creating an event-driven button with 206
- software, required 20
- starting gOOi 42
- static inheritance 104
- status line, MANTIS 129
- subclasses
  - changing 24
  - creating 94
  - function key map subclass naming 95
  - saving 96

**T**

- tab order 159
- Template Hierarchy Browser
  - description 21
  - screen 94
- templates
  - creating a key map 91
  - default key map 63
  - for visual items
    - application template inheritance 104
    - overview 103
    - steps for creating 105
  - inheritance 104
  - updating 93
- testing an application 109

- toolbar
  - definition 77
  - for hotspots 72
  - for user 76
  - items 78
  - spacing items 79
  - viewing
    - button descriptions 165
    - in Designer 163
- Tools window 83
- translation tables 311

## U

- UEF
  - definition 24
  - displaying screen and prompter
    - settings 123
  - extract files, locating 121
  - file
    - downloading 117
    - parsing 90
    - preparation 118
    - splitting 119
  - files directory 131
  - for IBM mainframe 293
  - images
    - copying from host to PC 111
    - downloading from MANTIS
      - Screen Design Facility 115
    - obtaining with Host Monitor 112
    - on ESDS cluster 117
    - saving 115
    - using FTP to copy to PC 117

- UEF Generator, using 327
- underline field attribute 60
- UNIX considerations
  - Connect timeout value 53
  - Emulator Check Interval 56
  - Host address 53
  - Just In Time window 56
  - pause Palue 55
  - Port number 53
  - Screen Tag Option 58
- up arrow key 33
- upgrades 96
- user toolbar 76

## V

- vertical repeat of dynamic fields 119
- VSAM ESDS cluster, UEF
  - images on 117

## W

- wizards
  - Excel tool 153
  - Word tool 149
- Word Wizard
  - description 22, 149
  - matching gOOi form fields 151
  - using 150
- wrap-around capabilities 82, 118, 127

