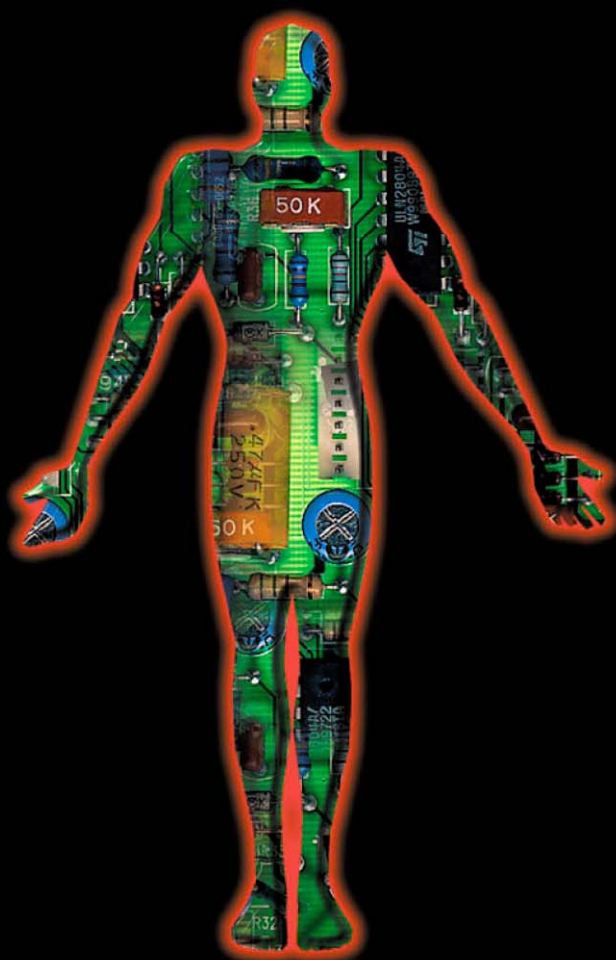


GURPS[®] CHARACTER BUILDER

Application Reference Manual



STEVE JACKSON GAMES

Version 3.0.0. Copyright © 1994-2002 Alter Ego Software, Inc., Inc. All Rights Reserved. **GURPS Character Builder** is a trademark of Steve Jackson Games Incorporated. The **GURPS Character Builder** application is licensed to Steve Jackson Games and is copyright © 1994-2002 Alter Ego Software, Inc. All Rights Reserved.

Table of Contents

<i>GURPS Character Builder Application</i>		
<i>Reference Manual</i>	1	
Menus.....	2	
File Menu	2	
New.....	2	
Game System.....	3	
Description.....	3	
Open.....	3	
Save	3	
Save As.....	3	
Close.....	4	
Save All	4	
Load Data Sheet.....	4	
Data Sheet Files	4	
Loaded Data Sheets.....	4	
Always Loaded	4	
Load Now >>	4	
Unload Now <<	5	
Load Always	5	
Remove	5	
Reload All Data Sheets	5	
Reload Character Sheet Script	5	
Set Copy/Print Filters.....	5	
Use Copy Filter or Print Template.....	5	
Description.....	5	
Associated Files	5	
Add	5	
Change	5	
Remove	5	
Open	6	
Print With Character Sheet.....	6	
Add Associated Files.....	6	
Print.....	6	
Through Filter	6	
Through Template	6	
Form Setup	6	
Print Dialog	6	
Print Using.....	6	
Print Preview.....	7	
Print Preview Using.....	7	
Print Preview	7	
Next Page	7	
Previous Page	7	
Print	7	
Page Setup.....	7	
Margins.....	7	
Measurement.....	7	
Text Format	7	
Text File Page Setup.....	7	
Left, Right, Top, Bottom.....	8	
Measurement	8	
Show Footers	8	
Recent Files	8	
Exit	8	
Edit Menu.....	9	
Cut.....	9	
Copy	9	
Paste	9	
Clear.....	9	
Duplicate	9	
Undo.....	9	
Filter Copy	9	
Edit Object.....	9	
Select All	9	
Search.....	9	
Search Again.....	9	
Bring to Front.....	10	
Send to Back.....	10	
Copy Formats.....	10	
Paste Formats.....	10	
Set Tab Order.....	10	
Search and Change.....	10	
Search Field	10	
Find What.....	10	
Found	10	
Target Field	10	
New Value	10	
Find Next.....	10	
Change All.....	11	
Regular Expressions.....	11	
Default Formats	12	
Copying Associated Items	12	
List Already Full.....	12	
Different Game System.....	12	
Different Game System Level	12	
Utilities Menu	13	
Notes Window.....	13	
Details Window	13	
Button Bar	13	
Context Menu	14	
Menu Commands.....	14	
File Print Details	14	
File Page Setup.....	14	
Format Menu	14	
Format Plain	14	
Format Bold.....	14	
Format Italic.....	14	
Format Underline.....	14	
Format Left	14	
Format Center	14	

Format Right.....	14	Filter	20
Format Bulleted	14	Output.....	20
Format Increase Indent	14	Copy to Clipboard	20
Format Decrease Indent.....	14	Single File.....	20
Format Paragraph.....	14	Append to This File	20
Format Font.....	14	Browse	20
Paragraph Format.....	14	Multiple	20
Alignment.....	14	Extension.....	20
Bulleted.....	14	Print through Template.....	20
Keep Next.....	15	Ask for Print Filter.....	20
Indentation.....	15	Print through Filter.....	20
Spacing.....	15	Print on Same Page	20
Game System Preferences.....	15	Preview.....	20
Preferences Dialog.....	15	Saving the Text of Many Character Sheets into a Single File	20
Selection Method.....	15	Saving Many Character Sheets as Text in Their Own Files	21
Check When Adding.....	15	Printing Multiple Character Sheets at Once	21
Disallow if Not Satisfied	15	Printing Multiple Character Sheets through a Filter	22
No Bad Value Dialog	15	Character List Files	22
Show Button Bar.....	15	Select Files for Filtering	23
Show Status Bar.....	15	Selecting More than One File.....	23
Ask for Filters.....	16	Selecting All Files.....	23
Show Hourglass	16	Save Filtered Text As	23
Binary Save	16	The Name Finder	23
Edit Added Options.....	16	Name Finder	23
Use Fractions	16	Name Search Examples	24
Show Rule Lines.....	16	Female Names	24
Confirm Mult. Deletion	16	Russian Male Names	24
Measure.....	16	Old French or Old German Male Names	24
Use Dialog Background Color	16	Names List.....	24
Data Sheet Loading Delay	16	Opening Different Name Files	24
Text Editor	16	Die Roller	24
Source Directory.....	17	Dice Window	25
Browse.....	17	Resizing the Die Roller.....	25
Show Tips Window.....	17	Other Topics.....	25
Update Character Sheets.....	17	Setting the Text of Buttons in the Die Roller...	26
Files to Update	17	Die Specifications	26
Files With Differences.....	17	Die Specifications.....	26
Select Files.....	18	Number of Dice	27
Open	18	Other Operations.....	27
Template.....	18	Best N of M Rolls	27
Script.....	18	Number of Independent Rolls	27
Destination.....	18	Choose Without Replacement	28
Browse.....	18	Counting Successes and Values Rolled	28
Update	18	Playing Cards	28
Compare After Update.....	18	Copying Die Rolls to the Clipboard.....	29
Close.....	18	Deleting Die Specifications	29
Selecting Character Sheets to Update.....	18	Die Roller Options.....	29
Selecting More than One File	18	Show Die Rolls.....	29
Selecting All Files	19	Always On Top.....	29
Choose Directory	19	Show Body Total	29
Multiple Character Sheet Filter	19	Draw Dice.....	29
Topics	19	Draw Pips on Six-Siders	29
Files	19	Play Sound Effects.....	29
Select Files	19		
Remove	19		
Include Associated Files.....	19		

Use Bitmap Images of Dice if Present.....	30	Default Check Expression and Message	42
Die Font Size	30	Font	42
Recalculate	30	Skin	42
Print Auxiliary Files.....	30	Use Dialog Background Color.....	42
Printing a File.....	31	Keyword	42
Files	31	Menu Character	42
Open	31	Modify List Information.....	42
Print	31	List Name	42
Data Menu.....	32	Var. Name	43
Tools Menu	33	Cat. Name	43
Character Sheet Tools Menu	33	Font	43
Check Requirements.....	34	Menu Character	43
Show	34	Cost Prompt	43
Satisfy	34	Item Format	43
Summary	34	Dialog	43
Satisfy All	34	Keyword	43
Satisfy Count of Items in a Category	34	Level Prompt	43
Required: #; Present: #.....	35	Cost Prompt	43
Add	35	Total Prompt.....	43
Show.....	35	Header.....	43
Requirements.....	35	Check Expression	43
Turning off Requirements.....	35	Check Message	44
Types of Requirements	35	Default Check Message	44
THEN and OR	35	Change Exp.....	44
Item Level Requirements	35	Alternate Formats.....	44
Required Value	35	Alternate Formatting Examples.....	44
Named Item	35	Modify Character Sheet Info	45
Excluded Item.....	35	Basic Character Sheet Info.....	45
Number of Items in List.....	36	Game System.....	45
Number of Items in a Category in a List.....	36	Sheet Type	45
Number of Items in a Number of Categories.....	36	Level.....	45
Expression.....	36	Template File.....	45
Option on an Item	36	Description	45
Buttons	36	Copyright.....	45
Satisfy	36	Calculations	46
Summary	36	Rounding	46
Satisfy All	36	Constant Cost.....	46
List Summary	36	Default Total Cost Formula	46
Looking at Other Lists.....	36	Max. Attempts	46
Alternate Format	37	Data Sheets	46
Name	37	Loaded Data Sheets	46
Header	37	Available Data Sheets	46
Format	37	Remove >>	46
Sort.....	37	<< Add.....	46
Sort By	37	Auxiliary Files.....	46
Reverse Order	37	Help File	46
Sublists	37	Load Script	47
Recursive sort.....	38	Shortcut File	47
Notes	38	Chargen Template	47
Data Sheet Tools Menu	39	Chargen Script	47
Print Template Tools Menu	40	Run On New.....	47
Modify Menu.....	41	Window Menu	48
Modify Data Menu	41	Pack Windows.....	48
Modify Dialog Information	41	Overlap Windows	48
Dialog Name.....	41	Arrange Icons	48
Check Expression and Message.....	41	Close All	48

Open All	48	Find	56
Window Names	48	Next	56
Debugger	48	Req.	56
Help Menu	49	Check All	57
Contents	49	Show Alphabetically	57
Help	49	Check Req.	57
Game System Contents	49	Random	57
Game System Info	49	Rules	57
Game System Tutorial	49	Apply Selection Rules	57
Item Info	49	Game System-Specific Help	57
Using Dialogs	50	Find	57
Opening Other Windows	50	Available Items Context Menu	57
Changing Values	50	Select Item	57
Pictures	50	Open Sublist	57
Display Adapter Note	51	Show All Items Alphabetically	57
Picture Search	51	Show Folder Hierarchy	57
Query	51	Find	58
Resizing the Dialog	51	Find Next	58
Search	52	Show Requirements	58
Or	52	Check All Requirements	58
And	52	Item Info	58
Find All	52	Select Random Item	58
Keywords	52	Item Selection Rules	58
Matching Pictures	52	Selection Rules	59
Preview	52	Selection Rule Sets	59
Previous	52	Working with the Rule List	59
Next	52	New	59
Browse	52	Change	59
Options	52	Delete	59
Browse Picture Library	52	Rule Set	59
Set Library Directory	53	Add	59
Picture Search Options	53	How Selection Rules Work	59
Allow Picture Description Changes	53	New Rule	60
Directory	53	Type	60
Search for Pictures	53	Category	60
Save Picture As	54	File	60
File Name	54	Browse	60
Save as type	54	Edit Rule	60
Notes	54	Category	60
Item Lists	55	Name	60
Adding Items	55	Value	60
Changing Items	55	Suppress Category	60
Highlighting Items	55	Expression	60
Deleting Items	55	Suppress Item	61
Moving Items	55	Suppress Item Prefix	61
List Tools Context Menu	55	Suppress Data Sheet ID	61
Getting More Help	55	Suppress Sublist	61
Editing Other Types of Items	55	Requirement	61
Sublists	55	Rules List	61
Excluded Items	55	Description	61
Selecting Items	56	NOT	62
To See All Available Items	56	Duplicates Only	62
Display Category	56	Active by Default	62
Select	56	Rule Set	62
Open	56	Select Rule File	62
Info	56	New Category Name	62

Add Rule Set	62	Var. Name	75
Button Bar	63	Original Name	75
Copyright Information	64	Data Sheet ID	76
Troubleshooting Printing Problems	65	Random Pips	76
The top (or bottom) of the character sheet is		Auto ID	76
chopped off	65	Parent	76
Rich Edit DLL	66	Priority	76
Shortcuts	67	See Also	76
Closing and Opening the Shortcuts Window	67	Edit Properties: Flags	76
Open and Close Shortcut Folders	67	Automatic Item	76
Widen the Shortcut Window	67	Exclude from Total	76
Editing Items	68	Generic Item	76
See also	68	Use Default	77
Editing Items	69	Baseable on Default	77
See also	69	Based on Default	77
Editing Items	69	Add As List	77
See also	70	Keep Cost	77
Editing Items	70	Create Unique Variable	77
See also	70	Don't Duplicate Automatic Items	77
Item Options	70	Delete Auto-added Items	77
Option Editing Buttons	71	Disallow Check Exp. Violation	77
Selecting Multiple Options	71	Duplicates Expected	77
Option Context Menu	71	Reference Item	77
Select Option	72	Check Req. only when adding	77
To Open or Close a Category of Options	72	Don't Set Adjustments and Automatic Items	77
To Select an Option	72	Don't Check Requirements	78
To Resize the Select Option Dialog	72	Disallow Satisfying Requirements	78
Select Category	72	Edit Properties: Sublist Flags	78
Edit Sublist Information	72	Charge for Automatic Items	78
Name	72	Don't Mark Automatic Items	78
Exclude	72	Add Sublist and Contents	78
Cost	72	Children Inherit Options	78
Total	72	List Has Own Cost	78
Children Inherit Options	72	Child Costs Are Separate	78
Open List	72	Edit Properties: Categories	78
Options	72	See Also	78
Notes	72	Edit Properties: Automatic Items	79
Hints	72	Setting the Level	79
Item Notes	73	Renaming Automatic Items	79
Edit Item Name	73	Adding Options	79
Special Editing Keys for Item Notes and Names		Duplicate Option	79
.....	73	No Clear Option	79
Choose Items	73	Select Option Value	80
Get Item Reference	73	Renaming Options	80
Choose Value	73	Remove Option from the Options to be Added	
Character Sheet File Name	74	80
Hints	74	Adding Adjustments	80
Details	74	Adding to Another List	80
Edit Properties	75	Specifying the Source Category	80
Property Groups	75	Clearing Automatic Item Modifiers	81
Basic Properties	75	Charging for Automatic Items	81
Name	75	Adding Item References	81
Value	75	Zeroing Item References	81
Cost	75	Clearing the Option List	81
Class	75	Conditional Automatic Items	81
Alias	75	Add a Category	82

Allow Duplicate	82	See Also.....	88
Default Item Name.....	82	List Editing Operations.....	88
Choose Items	82	Edit Properties: Requirements.....	88
Check Requirements	82	New	89
Select an Item at Random	82	Edit	89
Select a Number of Random Items	83	Delete	89
Selection Expression	83	Cut.....	89
Open and Close Sublist.....	83	Copy	89
Load Datasheet	83	Not.....	89
Set Default.....	83	Then	89
Set Value	83	Editing a Requirement	89
Setting a Qualifier	84	Expressions.....	89
Turning Off Requirements Checking	84	Value in a Dialog	90
See Also.....	84	Item (Single Required Item)	90
Edit Automatic Item	84	Option.....	90
<item>	84	Single Item Only	91
Add Adjustment	84	Number of Items.....	91
Add Category.....	84	Number of Items in List	91
Allow Duplicates	84	Number of Items in a Category	91
Charge	84	Number of Items in a Number of Categories.....	91
Choose Items	84	Message.....	92
To Add Entries to the List to Choose From	85	Edit Properties: Options.....	92
To Delete Entries	85	New Option	92
To Change the Text or Value of an Item.....	85	See Also.....	92
To Move the Items Up and Down in the List	85	Computing Costs of Items with Options	92
Choose Items by Cost.....	85	Expression Option	93
Choose Unique Items	85	Auxiliary Cost.....	93
Clear	85	Fraction Option	93
Clear Options.....	85	Multiplier Option	94
Close Sublist.....	85	Percentage Option	94
Conditional Else.....	85	Addition Option	95
Conditional End	85	Adjustment Option	95
Conditional If.....	85	Distributed Cost Adjustment	95
Default Item	86	Distributed Level Adjustment	96
Duplicate Option	86	Text Option	96
Item Reference	86	Editing Options.....	96
Load Data Sheet.....	86	Qualifier	96
No Clear Option.....	86	Modify.....	96
No Charge	86	To Bypass this Dialog and Directly Edit Option	97
Number of Random Items	86	Details	97
Open Sublist	86	Editing Options.....	97
Option	86	Type	97
Random Selection	86	Original.....	97
Remove Option.....	86	Alias.....	97
Selection Expression	86	Qualifier	97
Set Value	86	Display in Output.....	97
Source Category.....	86	Edit on Insertion	97
Target List.....	86	Keep Value on Conversion	97
Zero Item References.....	86	Inherited	98
Zero Options.....	86	Expression.....	98
<Generic Directive>	87	Token Pasting	98
Edit Properties: Adjustments.....	87	Disp. Exp.:	98
Reverse	88	Cat.	99
Important Note	88	Item Ref.....	99
		Item Reference Category	99
		Item Selection.....	99

Change	99	Forcing Objects to Be the Same Size	111
Types of Options	99	Edit Object	111
See Also	99	Special Keys	111
Change Option Value Type	99	Edit Rounded Rectangle	111
Check Exp.	99	Flowing Text from One Page to Another	111
Plain Text	99	Draw Menu	113
Item Reference	100	Snap to Grid	113
Toggle	100	Framed	113
List	100	Filled	113
Expression	100	Ruled	113
Cost Expression	101	Cell Lines	113
Check Expression	101	Line Style	113
Check Message	101	Line Color	113
Option Check Expression	101	Fill Color	113
Check Expression	101	Cell Lines	113
Reference Item	101	Removing Cell Lines	113
Check Message	101	Align Menu	114
Examples	101	Text Menu	114
Option Item References	102	Plain, Bold, Italic, Underline	114
Edit Properties: Format	102	Left, Center, Justified, Right	114
Format String	102	Font Size	114
Alt Fmts	103	Font Face	114
Edit Properties: Lookup Array	104	Font	114
Create an Array	104	Properties	114
Constant Values	104	Link	114
Dialog Type	104	Font Face	114
Standard	104	Text Properties	115
Name Only	104	Alignment	115
Options Only	104	Font	115
Level Only	104	Tab Stops	115
Cost Only	105	Default Tab Stop	115
No Cost	105	First Indent	115
Custom	105	Left Indent	115
Edit Properties: Expressions	105	Line Height	115
Cost Expressions	105	Wrap Tabs	115
Check Expressions	106	Num. Columns	115
Default Value	106	Column Spacing	115
Not Based on Default	106	Orientation	115
Based on Default	106	Edit Template Information	117
Total Cost Formula	107	Game System	117
Edit Properties: Automation	107	Sheet Type	117
Script	107	Game System Level	117
Editing Print Templates	109	Include File	117
Adding Graphic Objects	109	Inhibit Detail Printing	117
Polygon	110	Orientation	117
Number of Sides	110	Prompt for Margins	117
Angle of Rotation	110	Form Size	117
Editing Graphic Objects	110	Description	117
Selecting an Object for Editing	110	Number of Objects	117
Selecting More than One Object	110	Set Game System	117
Deleting Objects	110	Get Size	118
Changing the Size of Objects	110	Set Picture Reference	118
Changing the Colors of Objects	111	Character Sheet Picture Reference	118
Changing the Color of Text	111	File Name	118
Constraining Object Shapes	111	Expression	118
Lining Objects Up	111	Field Name Reference	118

Default File	119	Merging Data Sheets	127
Modify Print Template Pages	119	Save Data Sheet As	127
Add	119	Edit Option List	128
Delete	119	Merge Conflict	128
Edit	119	Keep Original	128
Up	119	Keep All Originals	128
Down	119	Overwrite	128
See Also	119	Overwrite All	128
Edit Page Info	120	Keep Both	128
Page Name	120	Keep All	128
Page Type	120	How To	129
<i>Normal</i>	120	How to Create a New Character	130
Optional	120	How to Create a New Game System	130
Continuation	120	Quick Links	130
Background	120	How to Convert Character Sheets	130
Expression	121	Single File Conversion	130
Background Page	121	Multiple File Conversion	130
See Also	121	How to Update Character Sheets to New Versions	131
Continuation Example	122	Character Sheet Update Tips	131
Edit Text Link	122	How to Change Print Templates	131
Tag	122	How to Create a Character Template	132
Continuation	123	How to Add Controls to a Dialog	132
See Also	123	Add Text Label	133
View Menu	124	Add Edit Variable	133
Ruler Window	125	Add Default Value	133
Adding Tabs	125	Add Check Expression and Message	133
Changing the type of tab to be added	125	Force Constant	133
Removing tabs	125	Add the Cost Formula	133
Setting the left indent	125	How to Create Data Sheets	134
Setting the first line indent	125	Topics	134
Replacing one tab with another	125	Old-Style Data Sheets	134
Editing Binary Data Sheets	126	How to Add Items	134
Adding a new category	126	Constant cost	134
Opening a List or Category	126	Fixed cost per level	134
Closing a List or Category	126	Arbitrary cost based on level	135
Changing the Name of a List or Category	126	Costs based on attributes	135
Adding a New Item	126	How to Create Print Templates	135
Changing an Item	126	How to Create Text Filters	135
Deleting Items	126	How to Add Campaign-Specific Items	137
Editing Options	126	Loading a Character Sheet as Data	137
Set the Copyright Notice	126	Creating the Character Sheet Data File	137
Data Sheet Info	126	Loading the Character Sheet as Data All the Time	137
Title	126	Locating Items in Sublists	137
Copyright Notice	126	Updating the Items in the Character Sheet	138
Level	127	Adding a Sublist and All Its Children	138
Logo File	127	How to Generate Random or Complex Characters	138
Help File	127	How to Create Custom Screen Layouts	139
Data Sheet Options	127	Group Files	140
Open	127	Topics	140
Edit	127	Creating Group Files	140
New	127	Working with the Group Window	140
Delete	127	Context Menu and Shortcuts	140
Cut	127	Adding Files to the Group	140
Copy	127		
Paste	127		
Data Sheet Sublists	127		

Opening Character Sheets	140	Deleting Items	149
Highlighting Files	141	Merge Data Sheets	150
Removing Entries	141	Trying Out the Data Sheet	150
Working with the Clipboard	141	Data Sheet Development Process	150
Renaming Character Sheets	141	Converting .cds Files to .mds File Format	150
Seeing the Full Path Name	141	Macro Define	151
Group File Info	141	Find	151
Game System	141	Go to Line	151
Template	141	Data Sheet Errors	151
Preferred Filter	141	Goto Error	151
Preferred Print Template	141	Save	151
Adding Character Sheets to the Group	141	Data Sheet Folder Properties	152
Adding Existing Files to the Group	141	Sort Order	152
Adding New Files to the Group	141	Random Pips	152
Renaming Character Sheets	142	Choose Macro Argument Value	152
Printing Character Sheets for Groups of		Choose Macro Argument List Entries	152
Characters	142	Value	152
Previewing Printing the Text of Character		Selected Values	152
Groups	142	Add	152
Printing the Text of a Character Group	142	Delete	152
Copying the Text of Character Sheet Groups to		Change	152
the Clipboard	143	Up, Down	153
Saving the Text of Group Characters	143	Macro Help	153
Save Group File As	143	Choose Predefined Macro Argument List Entries	
Data Sheets	144	153
Topics	144	Value	153
Creating a Data Sheet	144	Selected Values	153
The Data Sheet Window	145	Add	153
Editing Text Entries	145	Delete	153
Data Sheet Information	146	Up, Down	153
Tabs	146	Macro Help	153
Basic Info	146	Choose Free-Form Macro Argument List Entries	
Title	146	153
Copyright	146	Value	153
Game System	146	Selected Values	153
Level	146	Add	153
Logo File	146	Delete	153
Browse	146	Change	154
Help File	146	Up, Down	154
Include Files	146	Macro Help	154
Data Sheet Defines	147	Editing Dialogs	155
Data Sheets	147	Control Menu	155
Local Macros	147	Align Menu	156
Adding Categories	147	Edit Combo Box Control	157
Adding Sublists	147	Variable Name	157
Adding and Editing Items	148	Values	157
Adding and Editing Items	148	Sort	157
The Clipboard	148	Edit	157
Adding Options	149	New	157
Insert Option	149	Delete	157
Insert New Option	149	Script	157
Editing Options	149	Fonts	157
Adding Comments and Text Lines	149	Edit Combo Box Control Entry	157
Comments	149	Text	157
Text Lines	149	Value	157
Headers	149	Edit Picture Control	157

Edit Variable Control	157	Down	163
Variable Name	157	Edit Font List	163
Force Constant	158	Use Default Font	163
No Border	158	Font Index	163
Use Background Color	158	Font Expression	163
Fonts	158	Dialog Fonts	163
Default Value	158	Add	163
Transparent	158	Change	164
Check Expression	158	Delete	164
Check Message	158	Up	164
Format	158	Down	164
Formula	159	Edit Script	164
Edit Text Control	159	Set Tab Order	164
Name	159	Edit List Control	164
Script	159	Variable Name	164
No Border	159	List Name	164
Transparent	160	Maximum Items	164
Use Background Color	160	Category	164
Fonts	160	Level Prompt	164
Edit Group Box Control	160	Cost Prompt	165
Text	160	Item Format	165
Fonts	160	Dialog	165
Text Control	160	Check Expression	165
Text	160	Check Message	165
Transparent	160	Default Check Message	165
Fonts	160	Editing Defines	166
Justification	160	Adding Variables, Arrays or Functions	166
Edit Checkbox Control	160	Deleting an Entry	166
Text	160	Editing an Entry	166
Name	160	Editing Variable Definitions	166
Script	160	Name	166
Fonts	160	Initial Value	166
Edit Formula Control	161	Current Value	166
Variable Name	161	Script Added	166
Expression	161	Set	166
Transparent	161	Edit Function Definition	166
Fonts	161	Name	166
Justification	161	Formula	166
Edit Button Control	161	Editing Array Definitions	167
Variable Name	161	Name	167
Button Text	161	Values	167
Script	161	Value	167
Bitmap	161	Change	167
Depress	161	Add	167
Fonts	161	Delete	167
Edit Bitmap Control	162	Defines Dialog Clipboard Functions	167
Variable Name	162	Copy	167
Expression	162	Cut	167
<i>Number</i>	162	Paste	167
<i>String</i>	162	Define Clipboard Format	167
File	162	Expressions	169
Bitmap Files	162	Operators	169
Add	162	Bonuses	170
Delete	163	Predefined Functions	171
Change	163	Global Variables	181
Up	163	Predefined Global Variables	181

listiteminfo example.....	181	Checking for Dialog Fields	199
Operands	182	Filter @italic	199
Procedures.....	182	Filter @keepnext.....	199
While in Procedures	182	Example	200
If in Procedures	183	Filter @landscape.....	200
Local Variables in Procedures	183	Filter @leftindent.....	200
Assignment in Procedures	184	Filter @message.....	200
Return in Procedures	184	Filter @need	200
Creating Filter Files.....	185	Example	200
Formatting Commands	185	Filter @options.....	201
Macros	185	Example	201
Referencing Character Sheet Data.....	185	Filter @output	202
Referencing Variables	185	Filter @parinfo	202
Referencing Expressions	185	Examples.....	203
Special Variables.....	185	Filter @plain.....	203
Conditional References	186	Filter @samepage	203
Referencing Dialog Fields	186	Filter @sheettype.....	203
Referencing List Members	187	Filter @sortexp	204
Referencing Options for Items.....	187	Filter @sortorder.....	204
Referencing Game System Preferences....	187	Filter @sub.....	204
Escaping Characters	187	Filter @tabs.....	205
Special Characters	187	Example	205
Continuing Lines.....	187	Filter @textcolor.....	205
Comments	187	Filter @trans	206
Filtering for Blank Character Sheets	187	Filter @translate	206
Filter Commands.....	187	Filter @var	206
Commands	187	Filter @vline	207
Filter @array	188	Example	207
Filter @assign	188	Filter @while	207
Filter @bgcolor.....	189	Example	208
Example.....	189	Filter @wrtapab	208
Filter @bold.....	189	Example	208
Filter @bolditalic.....	189	@writepicture	208
Filter @clipformat	189	Examples.....	209
Filter @description	190	Data Sheet Text Format	211
Filter @dest, @footer, @header	190	Topics.....	211
@header.....	190	Structure and Header Information	211
@footer.....	190	Define Section of Data Sheets.....	213
@dest	190	Define Processing	213
Example.....	190	General Notes	213
Filter @exit.....	191	Variables	213
Filter @extension	191	Arrays	214
Filter @fail	192	Functions.....	214
Filter @firstindent.....	192	Text Data Sheet Example	215
Filter @font.....	192	Data Sheet Item Format	217
Example.....	192	Text Data Sheet Requirements	220
See Also	193	Expression.....	220
Filter @fontinfo.....	193	Item in a List.....	220
Filter @for	194	Option Set on an Item	220
Filter @foreach.....	194	Number of Items in a List	220
Example.....	197	Number of Items in a Category	220
Filter @gamesystem	198	Number of Items in a Number of Categories	220
Filter @gettext.....	198	220
Filter @hline	198	Message.....	221
Filter @if.....	199	Data Sheet Option Format.....	222
Example.....	199	Additions.....	222

Adjustments	222	Menus	243
Auxiliary Cost	222	Menu Items	243
Expression	222	Commands	244
Fraction	222	Scripts	244
Multiplier	222	Shortcut File BNF	244
Percentage	222	Game System Configuration Parameters	247
Text	222	How It Works	247
Check Expression and Item References	223	Referencing Game System Configuration Parameters	247
Stopping Option Printing	223	Setting Configuration Parameters Programmatically	248
Display Expressions	223	Predefined Configuration Parameters	248
Option Alias	223	Creating Name Files	249
Option Categories	223	Name File Example	249
Use Option Check Expression Message	223	Picture Description File Format	251
Keeping Old Values on Conversion	224	Synonyms	251
Item Selection	224	Ignored Words	251
No Inherit	224	Files	251
Computed Option Value	224	Selection Rule File Format	252
Option Value List Format	224	Configuration Section	252
Item Reference Option Value	224	DefaultValues Section	252
Evaluated Text Option Value	224	Rule Sections	252
Checkbox Option Format	225	Script Debugger	253
Data Sheet Macros	226	Script Source Window	253
Invoking Macros	226	Controlling Execution	253
Parentheses- and Comma-Delimited Macro Invocation	226	Stepping	254
Tab-Delimited Macro Invocation	226	Breakpoints	254
Omitted Arguments	227	Line Breakpoints	254
\$\$define	227	Subroutine Breakpoints	254
Conditional Macros	227	Script Breakpoints	254
\$\$ifdef	228	Removing All Breakpoints	254
\$\$ifnull	228	Viewing Variables	254
\$\$if	228	Direct Variable View	255
\$\$elseif	228	Context View	255
Expression Evaluation	228	Find	255
\$\$scan	228	Next	255
\$\$repeat	229	Goto	255
\$\$error	230	Open	255
\$\$macrodesc	230	Breakpoints	255
\$\$argdesc	230	Rmv All	255
\$\$argtype	231	Stack	255
\$\$hidemacro	232	Script	255
Limitations	232	Source Window Context Menu	255
Text Data Sheet Include	233	Limitations	256
Character Sheet Format	234	Breakpoints	256
Details	235	Subroutine	256
BNF for <i>GURPS Character Builder</i>	236	Set	256
Expression Syntax	236	Remove	256
Command Script BNF	237	Script	256
Adjustment Syntax	241	Set	256
Shortcut File Format	242	Remove	256
Finding Shortcuts	242	Line Breakpoints	256
Common Sections	242	Goto Line	256
Overriding Section Definitions	243	Find Text	257
Other Topics	243	Convert Character Sheet	258
Shortcut Formats	243	New Character Template	258
Sections	243		

Conversion Script	258	edittext	273
Game System	258	button, defbutton	274
Other Options	258	matchbox	274
Conversion Details	259	matchlist	274
Copied Information	259	keyword	275
Command Scripts	260	dialogsize	275
Including Command Scripts in the Data Menu	260	ctrlsize	275
Generate Character	261	keepnext	275
Conversion Script Format	262	rdonly	275
Topics	262	Die Roller	275
Commands	262	Examples	275
Comparing Character Sheet Totals	263	edititem	276
Conversion Expressions	263	exclude	276
Item Already Present in List	263	exec	276
Item Present in Source List	263	exit	276
Random Item Selection	263	fail	277
Item Available for Selection	264	for	277
Index of Selected Item	264	foreach	277
Count of Items	264	Example	278
Current Window Name	265	func	278
Current Field Name	265	gamesystem	279
Current Date	265	getfilename	279
Check Requirements Flag	265	<i>fileName</i>	279
User Canceled Script	265	<i>title</i>	279
Tick Count	265	<i>pattern</i>	279
Assignment	265	Example	279
add	266	getitem	279
addautoitems	267	if	280
addoption	267	include	281
appendProgress	267	insertitem	281
array	268	item	281
checkallreq	268	Example	282
checkreq	268	list	282
closeprogress	268	options	282
context	269	sublist	283
convlevels	269	category	283
copyall	269	destination	283
copydefines	269	samename	283
copydialog	270	default	283
copyitem	270	unconverted	283
copyleft	270	General Tips	283
copynotes	270	Special Symbols	283
copyoption	270	Example	284
datasheet	270	menucommand	284
deleteitem	271	notes	284
dialog	271	opensublist, closesublist	284
dialog	271	openwindow	285
text	271	option	285
checkbox	271	options	285
combobox	271	optqualifier	286
number	272	pauseProgress	286
dropdown	272	purgedatasheet	286
listbox	272	return	286
treelist	272	scriptmode	286
editcombo	273	selectitem	287
		set	287

Example.....	288
showProgress	288
showreq.....	289
showallreq	289
skip	289
sortlist	289
sub.....	290
submenu and menuitem.....	291
menuitem	291
Script File Execution	292
switch	292
title.....	293
var	293
warn.....	293
while	293
Conversion script variables	293
Converting Character Sheets from Other Applications.....	294
External Converter Setup File	294
The Process	294
The Converter Application	295
The <i>GURPS Character Builder</i> Output File.....	295
Progress Dialog.....	295
Text Editor	296
File Page Setup.....	296
Edit Copy (CTRL+C).....	296
Edit Paste (CTRL+V).....	296
Edit Delete Line (CTRL+D).....	296
Edit Search (CTRL+F).....	296
Edit Search Again (F3)	296
Edit Search and Replace (CTRL+R).	296
Edit Go to Line.....	296
Edit Font.....	296
Scroll (CTRL+UP and CTRL+DOWN arrow)	296
Save Text File	296
Search	296
Find Text.....	296
Match Case	296
Whole Word	296
Search	296
Search and Replace.....	297
Find Text.....	297
Replace with	297
Match Case	297
Whole Word	297
Search	297
Replace	297
Replace All	297
Terms	298
Character Sheet	298
Game System.....	298
Character Sheet Template	298
Data Sheet.....	298
Binary Data Sheet	298
Print Template	298

Groups.....	298
Filters.....	298

GURPS Character Builder Application Reference Manual

GURPS Character Builder™ creates characters for the **GURPS** roleplaying game system. It also prints character sheets, optimizes characters, edits data sheets, etc. You can also develop game system templates for game systems other than **GURPS**.

This application reference manual gives detailed information about the application's menus and dialogs, and how to program for it. For specific information about creating **GURPS** characters refer to the **GURPS Character Builder User's Guide**.

For the most current **GURPS Character Builder** news see the Steve Jackson Games website at <http://www.sjgames.com>. For support questions, see the Alter Ego Software, Inc. website, <http://www.alteregosoftware.com>.

GURPS Character Builder is a trademark of Steve Jackson Games Incorporated. The **GURPS Character Builder** application is copyright © 1994-2002 Alter Ego Software, Inc. All Rights Reserved.

You can obtain help anywhere in the application by pressing the F1 key, or by selecting the **Help** command on the **Help** menu.

Menus

The following menus are available in the application. Depending on which window is open, some menus may be absent.

- File (p. 2)
- Edit (p. 9)
- Utilities (p. 13)
- Data (p. 32)
- Tools (p. 33)
- Window (p. 48)
- Help (p. 49)
- Format (p. 14)
- Text Menu (p. 114)
- Draw Menu (p. 113)
- Align Menu (p. 114)
- Modify (p. 41)

File Menu

- New... (p. 2)
- Open... (p. 3)
- Save (p. 3)
- Save As... (p. 3)
- Save All (p. 4)
- Close (p. 4)
- Load Data Sheet... (p. 4)
- Reload All Data Sheets (p. 5)
- Reload Character Sheet Script (p. 5)
- Set Copy/Print Filters... (p. 5)
- Associated Files... (p. 5)
- Convert... (p. 258)
- Generate Character (p. 261)
- Merge Data Sheet (p. 150)
- Print (p. 6)
- Print Auxiliary Files... (p. 30)
- Print Using... (p. 6)
- Print Preview (p. 7)
- Print Preview Using... (p. 7)
- Page Setup... (p. 7)
- Exit (p. 8)

Recent files will appear on the **File** menu before the **Exit** command.

New...

The **New...** command creates new files, including character sheets (p. 298), character sheet templates (p. 298), print templates (p. 298), data sheets (p. 298), binary data sheets (p. 298), group files (p. 298) and text files.

- Click the radio button associated with the desired file type. To create a new character sheet or character template, click the name of the base character template. Click OK to create the new file.
- You can also double-click the character template to create a new character sheet based on the clicked template.

- To create a character template for a new game system, click the **Character Template** radio button, then click the template named `blank.cst`. This creates a completely blank character sheet template, which you can customize as you desire.
- To create a data sheet (p. 144), click the **Data Sheet** radio button and click the template for the desired game system.
- To create a group file (p. 140) for creating groups of characters, click the **Group** radio button. Select the game system and character sheet template that you wish to use when you add new files to the group.
- To create an empty text file (for macro include files, filters, conversion scripts, etc.), click **Text File**.

Game System

If you wish to display only those character sheet templates for a particular game system, click the Game System list. Then click the desired Game System.

To show all character sheet templates, select "(All)".

Description

When a character sheet template is highlighted, a short description of it will appear in the Description box (if one is available).

Open...

GURPS Character Builder recognizes four types of files: Character Sheets (p. 298), Character Templates (p. 298), Print Templates (p. 298) and Data Sheets (p. 298). You open each type of file through the Open command.

To select the type of file that is displayed in the File Name list, select one of the types in the List Files of Type list. Or you can simply type the pattern in the File Name edit box and press Enter. **GURPS Character Builder** automatically recognizes the type of file that you open.

If you select a type of file that **GURPS Character Builder** cannot recognize, it assumes that the file is a text file and opens it as such, either within **GURPS Character Builder**, or with the text editor that you specify in the Preferences (p. 15) dialog.

GURPS Character Builder categorizes files by examining their contents, then opens them according to what they appear to be. To force a file to be opened as a text file, enter the file name in the **File name** field, then click the **File of type** dropdown list and click Text File. Click **Open** to open the file as text.

If you have installed conversion scripts for other character generator applications, you can automatically convert them to **GURPS Character Builder** form by selecting them in the Open Character Sheet dialog. The conversion will be run automatically.

How to set up External Conversions (p. 294).

Save

Saves the file to disk. The file is not kept open while you are editing it.

When saving character sheet files, the following information is saved in addition to the explicit character sheet data:

- Current status of open windows (open/closed, location, size).
- Current status of lists (open/closed).
- The names of any data sheets currently open.

Save As...

The **Save As...** command lets you give the file you are editing a new name. If you are saving a character sheet or character sheet template, you may save the file as either of those types. See the Save command (p. 3) for more details.

The text of a character sheet can also be saved as a text file. The three types of commonly used text files are plain text, RTF files (or Rich Text Format, used by many word processing applications such as Microsoft Word), and HTML files (which are displayed on the web).

To save the character sheet as text:

- When the Save Character Sheet dialog is open, click the **Save as type** drop-down list.
- Click the type of file you wish to save as: Filtered text (a .txt file), Filtered HTML (.htm) or Filtered RTF (.rtf).
- Enter the name that you wish to save the file under. Make sure that the extension is not .chr and that you won't be overwriting the original character sheet.
- Click **Save**.
- The Use Copy Filter Dialog will appear (if it doesn't, escape from these dialogs and check the **Ask for Filters** checkbox in the **Utilities | Preferences...** (p. 15) dialog, then try again).
- Click the filter you wish to use to save the character sheet. Different filters present the character in different ways. Read the descriptions of the filters to get an idea of what the output will look like.
- Click **OK**.

The application will save the text of the character sheet as filtered through the selected filter.

You can also copy the text of character sheets to the clipboard with the **Edit | Filter Copy** (p. 9) command.

Close...

Close the file. If you have made changes, you will be asked whether you want to save them.

Save All

Save all changed files. Files that have never been saved will request a name.

Load Data Sheet...

In order for you to use items that are predefined for a game system, you must have the data sheet(s) for the game system loaded. You can specify that a character sheet automatically load a data sheet in the Edit Character Sheet Info (p. 45) dialog, or you can list the ones you want loaded each time **GURPS Character Builder** starts, or you can load and unload data sheets manually.

You can access items only from data sheets that use the same game system as the character sheet you are currently accessing.

Items from all data sheets for the same game system are merged into sublists of the same name.

Data Sheet Files

Lists the available (unloaded) data sheets. Click **Load Now >>** to load the highlighted data sheet. Click the **Load Always** button below this list to add the highlighted data sheet to the Always Loaded list. Character sheets may also be loaded as data.

Loaded Data Sheets

The data sheets in this list are currently loaded. Click the **<< Unload Now** button to unload the highlighted data sheet, which moves it to the **Data Sheet Files** list. Click the **Load Always** button below this list to add the highlighted data sheet to the Always Loaded list.

Always Loaded

The data sheets in this list are loaded each time **GURPS Character Builder** starts. To remove the highlighted data sheet from this list, click the **Remove** button.

Load Now >>

Loads the data sheet file highlighted in the Data Sheet Files list, moving it into the Loaded Data Sheets list.

Unload Now <<

Removes the data sheet highlighted in the Loaded Data Sheets list, deleting it from the loaded list and freeing all memory used for the data sheet. Once you unload the data sheet you will not be able to access items that appeared on it unless you reload it.

If you hold down the shift key when you unload a data sheet, it will be immediately reloaded.

Load Always

This button appears twice, once below the first two data sheet lists. Clicking it copies the highlighted data sheet to the Always Loaded list.

Remove

Removes the highlighted data sheet from the Always Loaded list. This does not unload the data sheet if it is currently loaded.

Reload All Data Sheets

This command reloads all loaded data sheets. This is most useful for developers of game systems who want a quick way to test their changes.

Reload Character Sheet Script

This command will reload the Load Script (p. 47) indicated in the character sheet info. This is useful for developers who are developing the load script and making frequent changes.

Set Copy/Print Filters...

When you print or preview your character sheet, you do so through a print template (p. 298) or a text filter (p. 298). These "filters" must be selected before you can use the Print (p. 6) or Print Preview (p. 7) commands. You can also copy (p. 9) text to the clipboard through a text filter.

Use Copy Filter or Print Template

Select the Copy Filter or Print Template for this operation. Only those files for the character sheet's game system are displayed.

When you select the "Ask for Filter" option in the Preferences (p. 15), you will be asked every time you print or use the Filter Copy command for the Copy Filter or Print Template you wish to use. If you use several different templates or filters frequently, this is more convenient than having to use the **File | Set Copy/Print Filters...** (p. 5) every time before you print.

Description

A short description of the file. If no description is available, the description area will be empty.

Associated Files

Associated files can be printed along with the character sheet. You might wish to associate a group of characters with the group leader so that when you print the leader, all characters are also printed, or a vehicle or a base with the character who owns it.

Associated files are added automatically when you select a file for a Character Sheet File option.

If the associated file no longer exists, it will be grayed out.

Add

Choose a file to be added to the list. In general, you should keep associated files in the same directory with the main file. You should save the character sheet before adding associated files.

Change

Change the associated file to one with another name. If you rename the associated files outside of **GURPS Character Builder** you should change the name here as well.

Remove

Remove the associated file from the list.

Open

Open the highlighted file. This will open the file in **GURPS Character Builder** and close the Associated Files dialog. Double-clicking the file in the list does the same thing.

Print With Character Sheet

This setting controls whether associated files are printed with character sheets. If checked, associated files will be printed when you select the **File | Print** commands.

Add Associated Files

Click the file you wish to add to the associated files (p. 5). To add more than one file at a time, hold down the CTRL key and click a file. Click **Open** when you have selected all the files you wish to add.

Print...

You can print your character sheet two different ways: through a print template (p. 298) or a text filter (p. 298). You should select the print template and text filter prior to printing with the Set Copy/Print Filters (p. 5) command, or if you wish to choose the Copy Filter or Print Template every time you print, check the Ask for Filters checkbox in the Preferences (p. 15) dialog.

Through Filter

Print your character sheet as text only, as filtered through the currently selected text filter.

Through Template

Print your character sheet graphically, using the currently selected print template.

Form Setup

This dialog appears when a print template has the **Prompt for Margins** checkbox checked. This is typically done for print templates defined for special-sized forms, such as printing character summaries on 3" x 5" index cards.

Enter the margins for the left, right, top and bottom. If the paper's orientation is landscape, check **Landscape**, or if its orientation is portrait, uncheck **Landscape**.

For example, if you're printing on special forms that have three 3" x 5" index cards vertically above one another centered in the middle of an 8.5" x 11" sheet, you would specify 1.75" for the **Left** and **Right** margins (half of 8.5" - 5") and 1" for the **Top** and **Bottom** margins (half of 11" - 3 cards x 3" per card).

If your forms have four 3" x 5" cards in a 2 x 2 layout centered on a sheet in landscape orientation, specify .5" for the **Left** and **Right** margins (half of 11" - 2 x 5") and 1.25" for the **Top** and **Bottom** margins (half of 8.5" - 2 x 3"), and check the **Landscape** checkbox.

If you're printing on sheets that are exactly the same size as the print template form size, specify 0 for all the margins.

For example, if you're actually printing on single index cards, specify 0" margins. Depending on your printer, it is likely that you would also specify Landscape in this dialog and possibly in your printer setup as well.

These settings are remembered and proposed whenever this print template is used.

Print Dialog

You should also be sure to select the proper paper size and orientation in the **Preferences** of the Print dialog. For example, if you're printing on index cards, select the appropriate paper size. If the printer doesn't have the paper size you need, choose the "Custom" or "User Defined" paper size and enter the necessary dimensions.

Print Using...

This print template command performs a Print (p. 6) using one of the open character sheets. At least one character sheet using the same game system as the print template must be open. If there is exactly one open character sheet, it will be previewed. If there is more than one, a dialog listing all compatible character sheets will be displayed.

Print Preview...

The **Print Preview...** command lets you see how your character sheet will appear when printed. You should select the print template and text filter prior to previewing with the Set Copy/Print Filters (p. 5) command, or select the Ask for Filters option in the Preferences (p. 15) dialog to be asked each time for the appropriate filter or template. There are two subcommands:

Through Filter...

Previews the character sheet as it will appear printed with a text-based filter (p. 298).

Through Template....

Previews the character sheet as it will appear with the graphical print template (p. 298).

The Print Preview dialog (p. 7) will then appear with a depiction of the printed character sheet.

Note: In this version of *GURPS Character Builder*, the print preview is only a rough guess at how the character sheet will appear when printed. Be aware that some things will not appear as they will on the printer.

Print Preview Using...

This print template command performs a Print Preview (p. 7) using one of the open character sheets. At least one character sheet using the same game system as the print template must be open. If there is exactly one open character sheet, it will be previewed. If there is more than one, a dialog listing all compatible character sheets will be displayed.

Print Preview

This shows a preview of what the character sheet will look like when it is printed. This is only a rough approximation of the printed page, because the resolution and the fonts used on the printer are not the same as used on the screen.

Click the scroll bars on the sides to see different parts of the preview.

Next Page

Shows the next page. Inactive if there is no next page.

Previous Page

Shows the previous page. Inactive if there is no previous page.

Print

Closes the Print Preview dialog and starts the printing process.

Page Setup...

The Page Setup dialog allows you to set the margins, text properties and preferred measurements used when printing the character sheet with the text filter. This modifies the currently selected filter (p. 298): **all** character sheets that use this filter will be affected. You should select the text filter prior to changing the page setup with the Set Copy/Print Filters (p. 5) command.

Margins

Sets the margins on the page. You can override the default measurement units by adding a unit after the number: for inches, indicate "in" and for centimeters indicate "cm".

Measurement

Indicates the unit of measurement, either inches or centimeters.

Text Format

Clicking this button to set the font, line height, paragraph formatting, etc. for the text filter.

Text File Page Setup

Set the margins for printing text files (p. 296) and character sheet Details (p. 13).

Left, Right, Top, Bottom

The amount of blank space on each side of the page.

Measurement

Click **Inches** to use English measurements, **Centimeters** to use Metric.

Show Footers

Check this to display the file name, date and page number at the bottom of each page. The footer will be printed a quarter-inch (six millimeters) below the bottom margin, right justified along the right margin. You may need to increase your bottom margin if your printer's printable area does not extend that far.

Recent Files

The **Recent Files** command brings up a menu of the eight most recent files you edited. Click on the file you wish to edit. The eight most recently used files will also be displayed on the **File** menu just before the **Exit** command.

Exit

Leaves ***GURPS Character Builder***. If any files have changes that have not been saved, you will be asked whether you wish to save them.

Edit Menu

Some of these commands will not appear on the **Edit** menu for every type of window.

Cut

Cut the selected object and put it on the clipboard.

Copy

Copy the selected object and put it on the clipboard.

Paste

Paste the object from the clipboard into the character sheet.

Clear

Delete the selected object.

Duplicate

Available only for print templates. Makes a duplicate of the selected objects; this is sort of short cut for copying and pasting, but it does not use the clipboard, so the contents of the clipboard will be untouched.

Undo

The Undo command undoes the effects of a previous command. Only certain actions can be undone:

- Deleting items from item lists.
- Moving controls when editing dialogs.
- Editing text in edit fields.

You can undo the last fifty actions performed in each character sheet file. If you perform another undoable action that changes the character sheet, the undo "stack" is cleared for that file, and undo is no longer available.

Undeleting items will open the list that the items were deleted from. Undeleting an item that has added other items automatically will restore all deleted items, including the associated items. Undoing text field changes will return the caret to the field.

Filter Copy

This copies the data in the character sheet onto the clipboard in a text format, filtering it through a text filter. You must first select the filter you want to use with the Set Copy/Print Filters (p. 5) command, or you can select the Ask for Filters option in the Preferences (p. 15) dialog, and you will be asked each time for the filter to use. The text generated can be pasted into any application that can accept text from the clipboard.

Different filters may produce different kind of text. Most common are RTF and HTML text. When a filter produces RTF, only applications that can handle RTF will be able to accept that text from the clipboard. When an HTML is used, the raw HTML commands will be placed on the clipboard as normal text, along with the standard Microsoft HTML clipboard format.

Edit Object

Allows you to directly set the size and location (p. 111) of the selected object.

Select All

Select all objects in the window.

Search...

Available only when editing data sheets and lists in character sheets. You may let's you search for and change (p. 10) strings in items with this command.

Search Again

Available only when editing data sheets and lists in character sheets. Does the previous search again from the currently highlighted item.

Bring to Front

Available only for print templates and when editing dialogs. Makes the selected objects come to the top. That is, they are plotted after all other objects in the window.

Send to Back

Available only for print templates and when editing dialogs. Makes the selected objects go to the back. That is, they are plotted first, before all unselected objects.

Copy Formats

Appears only for print templates. Copies the formats for the currently selected object to the default formats, so that they can be applied to other objects with the **Paste Formats** command.

Paste Formats

Appears only for print templates. Sets the formats for the currently selected objects to the default formats, which can be set by simply using commands on the **Text** and **Draw** menus when no objects are selected, or with the **Copy Formats** command.

Set Tab Order

Appears only when you are modifying a dialog. Sets the order (p. 164) in which the TAB key goes from one control to the next.

Search and Change

The Search and Change dialog allows you to search for strings in the fields of an item. You can also change the item based on the search, replacing the string found with another string, or even doing other actions such as setting different fields to other values, adding values to other fields, or deleting other fields.

Search Field

Indicates what field(s) to search. If you select **All Fields**, all fields in the item will be examined. If you select another value (Name, Class, Format, etc.) only that field will be searched.

Find What

The string to search for. If you have turned on regular expressions (p. 11), certain characters (., ?, *, +, etc.) will have special pattern-matching interpretations.

Flags	Meaning
Match Anything	The whole field will always match.
Ignore Case	Upper and lower case strings are treated the same. This is not available if regular expressions are used.
Current List Only	Only the current list is searched; that is, the list containing the highlighted item.
Regular Express.	Use regular expression pattern matching

Found

This box displays the item and field found every time you click **Find Next**. Item Name displays the name of the item that matched. Field shows the name of the field that matched, and the value that matched.

Target Field

This list indicates what field to change. If you have selected the **Replace** action, the target field must be **Same Field**.

New Value

Contains the new value that replaces the string that matched, or the value to be added or set if the those actions are indicated. If you have turned on regular expressions (p. 11), you may include \n (where n is a number from 0 to 9) values to indicate the part of the expression that matched is replaced into the new value.

Find Next

Starts the search. The first item that matches the search pattern is highlighted.

Change

Makes the change indicated by the action selected, then searches for the next match.

Action	Meaning
Replace	The string indicated in Find What is replaced with the value in New Value . The replacement is made for every instance found in the string. If you replace "a" with "x", "Kalamazoo" would become "Kxlxmzoo".
Set	The target field is set to the value in New Value .
Add	The value in New Value is added to the target field. If the target field is a single value (such as Format), the action is the same as Set . If the target field is a list (such as Category, Adjustment or Automatic Items), the indicated value is added to the end of the list.
Delete	Deletes the target field.

Change All

Searches and changes items according to the search pattern until no more matches are found.

Regular Expressions

"Regular expression" is a programming term used to describe a powerful pattern-matching syntax. Regular expressions allow you to specify "wild cards" in search patterns. The following characters have special meanings in regular expressions:

Character	Meaning
.	Period. Match any character.
[. . .]	Brackets. Match any of the characters inside the brackets. You may indicate ranges: "[a-zA-Z]" matches any alphabetic character.
[^ . . .]	Brackets complemented. Match any character <i>not</i> inside the brackets. The pattern "[^0-9]" matches any nonnumeric character.
*	Asterisk. Match zero or more of the previous pattern. "po*1" matches "pl", "pol", "pool", "poool", etc.
+	Plus. Match one or more of the previous pattern. "cr[aeiou]+m" matches "cram", "crom", "cream" and "craieaom", but not "crm".
?	Question mark. Match zero or one of the previous pattern. "ea?r" matches "er" or "ear".
()	Parentheses. Indicate grouping. Parentheses also can be used for the \n replacements. For example, if the Find What pattern is "^([^]+)(.+) \$" and the New Value is "\2, \1", the value "Hot Dog" would be changed to "Dog, Hot". Each number after the "\" indicates the corresponding set of parentheses, numbered from 1 for the first set.
	Or. Indicate alternation. "([0-9])([a-z]+)" would match "8" and "bcd", but not "@".
^	Caret. Anchor pattern to beginning of string.
\$	Dollar sign. Anchor pattern to end of string. "^.+ \$" matches all of any non-empty string.

When regular expressions are chosen, you can also specify "\0" in the New Value, which is replaced by the entire string that matched the regular expression. For example, if you want to put all strings that weren't already in double quotes into double quotes, specify:

```
^ [ ^" ] .+ [ ^" ] $
```

for Find What, and

```
"\0"
```

for New Value. Then click **Regular Express.**, **Find Next** and **Change** to change the first instance found.

You could also make costs negative by searching for "^.\+\$", and replacing the value with "-\0".

Default Formats

You can set the default formats to be used for print template objects with commands from the **Draw** and **Text** menus. You can also set them by selecting an object and then choosing the **Edit | Copy Formats** command.

The default formats are used whenever a new object is created, and when the **Edit | Paste Formats** command is selected.

Copying Associated Items

When you cut, copy or drag and drop items, there may be other items that are associated with them. These associated items are usually added automatically by another item or a process such as character generation.

For example, some game systems automatically add an associated skill when a certain advantage is added. Other game systems may add a large number of skills, advantages, disadvantages, etc., when a profession is chosen.

If you click **Yes**, the associated items will be copied along with the selected items. These items will be added to the end of the same list that contained them in the original character sheet. If that list doesn't exist in the destination character sheet, the associated items will be added to the same list as the selected items.

If you click **No**, only the selected items will be copied.

If you click **Cancel**, the action will be canceled and the character sheet will not be changed.

List Already Full

Some lists in dialogs can have only a single entry in them. You cannot drop or paste items into a list whose capacity will be exceeded by the operation. You may need to delete the existing entry (or entries) before attempting the operation.

Different Game System

You can't normally copy items between character sheets that use different game systems. This is because they may use different rules, and internally **GURPS Character Builder** uses a different set of formulas and variables to compute costs, etc. You should avoid this unless you are completely certain it's okay.

Different Game System Level

If you copy items from one version of a game system template to another you may run into similar problems encountered copying items between game systems. The variables and functions used to define the items often change between releases. You should convert (p. 258) old character sheets to the newest level before working with them.

Utilities Menu

Some of the commands on the Utilities menu will vary depending on the type of window that is open.

- Notes (p. 13)
- Details (p. 13)
- Preferences (p. 15)
- Game System Preferences (p. 15)
- Show Tips Window (p. 17)
- Show Shortcuts (p. 67)
- Update Character Sheets... (p. 17)
- Filter Character Sheets... (p. 19)
- Name Finder (p. 23)
- Die Roller (p. 24)
- Picture Library (p. 51)
- Recalculate (p. 30)

Notes Window

The Notes window allows you to enter any text you want about your character. Up to 5000 characters can be entered.

Details Window

The **Utilities | Details** command opens the Details window for the current character sheet. You can enter up to 1,000,000 characters of formatted information into the Details window. The contents of the Details window are printed after the character sheet is printed through the selected template. You can also print the details by themselves with the **File | Print Details...** command when you have the Details window open. Certain print templates also inhibit the printing (p. 117) of details

GURPS Character Builder uses the Rich Edit 2.0 DLL (p. 66) to provide this functionality.

Button Bar

The button bar gives quick access to formatting commands. Simply highlight the text and click the desired button bar to set that format. You can also use commands on the **Format** menu to achieve some of the same effects.

The two dropdown controls in the button bar select

- Font
- Point size.

The next group of three buttons select text formatting:

- Bold
- Italics
- Underlined

The next group of three buttons set paragraph alignment:

- Left-aligned
- Centered
- Right-aligned

The next button toggles the "bullet" on selected paragraphs.

The next four buttons control paragraph indentation.

- Reduce indentation of all lines
- Increase indentation ("exdent")
- Increase indentation of the first line of the paragraph
- Increase indentation of subsequent lines

Context Menu

Right-click the Details window to bring up a context menu.

Menu Commands

The following commands are unique to the Details window.

File | Print Details

Print the character Details window without printing the character sheet.

File | Page Setup...

The page setup dialog (p. 7) sets the margins for text files and Details. If the Show Footer checkbox is checked, a footer will be printed at the lower right of the page giving the file name, date printed and page number.

Format Menu

The following commands are available on the **Format** menu, which is displayed only when the Details window is open.

Format | Plain

Remove bold, italic and underlined formatting.

Format | Bold

Bold the selected text.

Format | Italic

Italicize the selected text.

Format | Underline

Underline the selected text.

Format | Left

Left-align the selected paragraphs.

Format | Center

Center the selected paragraphs.

Format | Right

Right-align the selected paragraphs.

Format | Bulleted

Toggle the bulleted state on the selected paragraphs.

Format | Increase Indent

Increase the indentation of the selected paragraphs.

Format | Decrease Indent

Decrease the indentation of the selected paragraphs.

Format | Paragraph...

Open the Paragraph Format dialog (p. 14) to set paragraph formatting (alignment, indentation, spacing, etc.).

Format | Font...

Open the Font dialog to choose the font name, style, size, etc.

Paragraph Format

This dialog sets the formatting of the selected paragraphs.

Alignment

Set the alignment (left-aligned, right-aligned or centered).

Bulleted

If checked, the selected paragraphs will be bulleted. You may wish to combine this with a First Line indentation that is less than the Left Indentation.

Keep Next

If checked, this paragraph will be kept with the following paragraph when the text is printed. This is useful for ensuring that a header is printed on the same page as the subsequent paragraph.

Indentation

Left: the distance of the second and subsequent lines from the left-hand margin. If this is greater than the **First Line** indentation, the first line will be "exdentented." To produce a bulleted list, also check **Bulleted**.

First Line: The distance of the first line from the left-hand margin.

Right: The amount of blank space to leave on the right-hand side of the paragraph.

Spacing

Before: The amount of blank space to leave above the paragraph.

After: The amount of blank space to leave below the paragraph.

Line Spacing: the values Double, One and a half and Single are available.

Game System Preferences...

This command displays a dialog for the current game system preferences. If no windows are open, then you will be asked to select the game system first.

Help is available for the Game System Preferences by clicking the Help button in the dialog itself.

The contents of this dialog are configured separately for each game system (p. 247).

Preferences Dialog

The Preferences dialog lets you customize some of ***GURPS Character Builder's*** behaviors.

Selection Method

Clicking Show Categories causes the hierarchical categories to be displayed in the Item Selection dialog. Clicking Show Names Only causes all items in the data sheet list to be displayed in alphabetic order.

Check When Adding

Makes ***GURPS Character Builder*** check the requirements when you add a new item or change a value in an edit box. If left unchecked, no checks are made on new items. You can still check requirements manually with the commands on the Tools menu.

Disallow if Not Satisfied

Causes ***GURPS Character Builder*** to reject any items you select that have unsatisfied requirements.

No Bad Value Dialog

Causes ***GURPS Character Builder*** to beep (instead of bringing up a dialog) when you attempt to enter a bad value for an edit field, or set the level of an item to an illegal value.

Make Backups

Causes ***GURPS Character Builder*** to make a .bak file each time you save a character sheet. If the files is a .bak file, no backup is made.

Show Button Bar

Causes ***GURPS Character Builder*** to display the button bar (p. 63), which you can use to perform menu commands with a single mouse click.

Show Status Bar

Displays a status bar at the bottom of the main window. This displays a single line of information about the currently highlighted menu command or depressed button on the button bar.

Ask for Filters

If you select this option, each time you perform an action that requires the use of a Print Template or Copy Filter, you will be asked which one you want to use. This can be more convenient than having to use the File | Set Copy/Print Filters... (p. 5) command each time you want to change the active filter, especially if you use many different filters.

Show Hourglass

Causes an hourglass to be shown any time lengthy operations will be taking place.

Binary Save

Checking this checkbox (the default) causes **GURPS Character Builder** character sheets and data sheets to be saved in "Binary Format." This is an internal format that can be read only by **GURPS Character Builder**. Unchecking this option causes these files to be saved in text format. You can edit such files with any text editor. You might wish to do this in order to use the macro features for creating text-format data sheets (p. 211). Binary files can be read faster than text files.

Edit Added Options

Checking this checkbox (the default) brings up the Edit Option when you add an Option. To cause the Option to be added to the Options list immediately, uncheck this checkbox.

Use Fractions

When this is checked, **GURPS Character Builder** uses fraction characters to display numbers such as 0.25, 4.5 and 5.75. This produces the following: ¼, 4½ and 5¾. If you change this, some values stored in character sheets may not be updated until you edit the item again.

Show Rule Lines

Most print templates display items in areas with rule lines, emulating character sheets that you fill in by hand. If you uncheck this checkbox, the lines will not be printed.

Confirm Mult. Deletion

If this is checked, you will be notified when you attempt to delete an item that has other items linked to it. This automatically added items are deleted when the parent item is deleted.

Measure

Select the system of measurement you wish to use within **GURPS Character Builder**: Metric or English. You should close all character sheets when changing this setting, and then reopen them.

Use Dialog Background Color

Checking this checkbox allows the use of background colors in dialogs and the character sheet windows. You may change the color by clicking the Set Color button.

Data Sheet Loading Delay

When a data sheet is loaded, the application makes sure that the splash dialog describing the data sheet has been displayed at least this many seconds (up to a maximum of 60). If you want no delay (that is, display the splash dialog only as long as it takes for the data sheet to load), set this to 0.

Text Editor

The name of the Windows application you wish to use to edit text files (such as filters, conversion scripts, etc.). Leave this empty to use the **GURPS Character Builder** text editor.

When you open a file that **GURPS Character Builder** doesn't recognize it will assume the file is text and attempt to use the text editor you specify here.

To use the Windows Notepad editor enter `Notepad.exe`. Notepad has a limit of 64K bytes; the **GURPS Character Builder** editor has a larger capacity.

Click the ... button after the editor edit field to browse for the desired application.

Source Directory

GURPS Character Builder obtains all character sheet templates, data sheets, print templates and text filters from this directory. By default it is the same as the directory where the application itself is installed. Note: you cannot save your preferences if this directory doesn't exist. Click the ... button after the directory edit field to get a directory browse dialog.

Note: be sure to include the driver designator when you change this value—certain functions in Windows require the drive designator to be present.

Browse...

This button brings up a directory browse dialog to search for the Source Directory.

Show Tips Window

Displays a window that tells you something about the commands that are of use in the current context.

To dismiss the Tips Window, click the **Close** button. The Tips Window will return as long as Show This Next Time checkbox is checked.

To turn off the Tips Window, uncheck the Show This Next Time checkbox.

Update Character Sheets

The **Utility | Update Character Sheets...** command updates character sheets from older versions to the most current versions loaded.

To update character sheets, follow these steps:

- Click the **Select Files...** button. The file selection dialog will appear.
- Choose the files you wish to update. Hold the CTRL key down and click a filename to select more than one file.
- Click the **Open** button to close the file selection dialog. The files you chose will be added to the Files to Update list.
- Select the template you wish to convert to. If you're simply updating to the most recent version of the templates, leave Template set to "(Original)".
- Select the script you wish to use for the conversion. If you're simply updating to the most recent version of the templates, leave Template set to "(Original)".
- If you want the new character sheets to be placed in a different directory from the original, click the **Browse...** button to select the destination directory for the updated files. This is suggested if you're converting from one game system to another, so that you don't lose your original character sheets.
- In the Choose Directory dialog double-click directories to switch to them. Click the **OK** button when you've selected the directory you want.
- Click the **Update** button. **GURPS Character Builder** will convert the files. If any errors occur (which might happen if the conversion script is for a different game system than the original character sheets), the update process will stop.

After the process is done, you can open files with any differences by highlighting them in the **Files with Differences** list and then clicking the **Open** button.

Files to Update

Lists the files you have selected (see **Select Files...**) for updating.

Files With Differences

Lists the files that have been updated, and have different character point totals from the original.

Select Files...

Click this button to choose the files you wish to update. A file selection dialog will appear. You can select multiple files by holding down the CTRL key when clicking file names. When you close the file selection dialog, the files you chose will be added to the **Files to Update** list.

Open

Highlight a file in the **Files with Differences** list and click the **Open** button to view the new file. You can also double-click the file you wish to edit. This dialog box hide itself until you open it again with the **Utilities | Update Character Sheets...** command.

Template

Click the drop-down list to select a template to convert to. Leave this set to "(Original)" if you want the character sheets updated to the original template.

Script

Click the drop-down list to select the conversion script (p. 262) to use with the conversion. Leave this set to "(None)" if you're updating character sheets to the most current version.

Destination

Indicates the destination folder (directory) where the updated files will be written. It is the original folder by default.

Browse...

Click the browse button (named ...) to browse for the directory where the updated files should be written. A dialog with the folder names will appear. The folder name you select will be placed in the **Destination** field.

Update

Click this button when you've selected the files you wish to update. The character sheets and items will be updated to the most current loaded versions. The original file will be renamed with a `.bak` extension and the updated version will take its place (except as noted below). Click the **Cancel** button to cancel the update while it is in progress.

If there are differences between the original file and the converted file, and the conversion script supports detection of such differences, the updated character sheet will be listed in the **Files with Differences** list.

Compare After Update

If this checkbox is checked, ***GURPS Character Builder*** compares the point totals of the old and new character sheets after the conversion. **Note:** *This will only work if the conversion script supports this feature.* If the point totals are not identical, the updated character sheet will be named with the original base file name plus "`_new`" and placed in the **Files with Differences** list. For example, the different version of the file named `George.chr` will be named `George_new.chr`.

Close

Dismisses the dialog box and forgets the list of files. If you wish to keep the list of files around while editing the files that have differences, just use the **Open** button -- don't close this dialog.

Selecting Character Sheets to Update

After you click the **Select Files...** button this dialog will appear.

Selecting More than One File

- Click the first file.
- SHIFT+click another file and Windows will select all files between the selected file and the one you click. (SHIFT+click means to press the SHIFT key, then, while you continue to hold it down, click the left button of mouse on the item you wish to select).
- CTRL+click files one at a time to add them to the selection.

Selecting All Files

- Scroll to the end of the list of files.
- SHIFT-click the last file in the list.

When you have selected all the files you want, click the **Open** button.

Choose Directory

Type the directory name in the Path edit field, or double click directory names in the Directories list.

Path

This is the name of the directory you will choose.

Directories

The list of directories. Double-click an entry in this list to set the directory to that entry.

Drives

Click the Drives drop-down list, then click the drive you wish to select. The Directories list will be updated to reflect your choice.

Create

Click this button to create the directory named in the Path edit field.

OK

Click this button when you are finished. The directory named in the Path edit field will be chosen.

Multiple Character Sheet Filter

The **Utilities | Filter Character Sheets...** command allows you to save many character sheets as text all at once.

This can be used to batch print many characters, and to print things such as combat record sheets, in which summary data from many characters is printed on the same page.

Topics

- Saving the Text of Many Character Sheets into a Single File (p. 20)
- Saving Many Character Sheets as Text in Their Own Files (p. 21)
- Printing Multiple Character Sheets at Once (p. 21)
- Printing Multiple Character Sheets through a Filter (p. 22)
- Creating Filter Files (p. 185)

Files

The list of files to process. Click the **Select Files...** button to add files to the list. When you add files to the list, the **Filter** list will change according to the game system of the selected files. If all files belong to the same game system, then only filters for that game system are displayed. Otherwise, all filters are displayed.

If any selected files are not compatible with the selected filter, they will not be processed. They will be left in the Files list so that you can try again with another filter.

Select Files

Opens a File Dialog in which you can select files to be filtered. To select multiple files, hold down the CTRL and SHIFT keys.

Remove

Remove the highlighted file from the Files list.

Include Associated Files

If checked, files associated with the files that you select are also added to the Files list.

Filter

The list of available filters. If all files in the Files list use the same game system, only filters for that game system will be displayed here. The next time this dialog is used, the game system will be used to decide which filters to display. If possible, **GURPS Character Builder** will highlight the last filter used.

When switching game systems, choose your files before selecting a filter. The appropriate filters will be displayed once you have chosen a file.

Output

Select a single radio button here to choose the type of output.

Copy to Clipboard

If checked, the selected files will be processed through the selected filter. The resulting text will be copied to the clipboard.

Single File

If checked, the selected files will be processed through the selected filter and written to the specified file name.

Append to This File

If checked, the text produced will be added to the end of the file specified for **Single File**.

Browse

Click this button to browse the file system to form the name of the name for single-file output. The resulting file name will be placed in Single File.

Multiple

If checked, each character sheet will be processed by the selected filter and written to separate files with the same name as the original character sheet, but with the extension specified by **Extension**.

Extension

The extension (the part of the file name after the dot) to be used for each character sheet produced by Multiple file filtering.

Print through Template

If checked, the files in the Files list will be printed with a graphical print template. If Ask for Template is unchecked, **GURPS Character Builder** will use the template specified in the Set Copy/Print Filter dialog.

Ask for Print Filter

If checked, **GURPS Character Builder** will ask for the name of the print template to use with each character sheet printed.

Print through Filter

If checked, **GURPS Character Builder** will print the selected files through the selected filter. If **Print on Same Page** is unchecked, these will be printed separately.

Print on Same Page

If checked, the selected files will be printed on the same page. This is often used for creating combat record sheets.

Preview

This is inactive until **Print on Same Page** is checked. If checked, the merged character sheet printing will be previewed rather than printed.

Saving the Text of Many Character Sheets into a Single File

- Click the **Select Files...** button.
- In the Select Files for Filtering dialog, CTRL+Click the files you wish to select (Character List files (p. 22) allow you to list commonly grouped characters together). You can also select all files by scrolling to the end of the file list and SHIFT+clicking the last file.
- Click **Open** to make the selection final. The files you chose will be displayed in the Files list box.

- You may click **Remove** to remove the highlighted file from the list (double-clicking the file also removes it).
- Click the name of the filter you wish to use in the Filters list box. When you click a filter name, a description of it will appear (if available) below the list.
- In the edit field after the Single File button, enter the name of the file you wish to create with the text of all the character sheets. You can click the **Browse** button to set the location for the file.
- If you want the text to be appended to the file you name (instead of overwriting it), click the **Append** button.
- Click the **Run** button. The button will be grayed out if you neglected to perform one of the necessary steps.
- As each file is successfully processed, it is removed from the File list.

If any character sheets could not be processed (because they use a different game system, for example), they will be left in the Files list. You can then choose a different filter in the Filters list, and run the filtering process again.

The files will be processed in the order specified in the filter file (p. 204), or alphabetically if no order is specified.

Saving Many Character Sheets as Text in Their Own Files

For all the files you choose, a new file will be created with the same base name as the original file, but a new extension. You might use this to create HTML versions of many character sheets at once.

- Click the **Select Files...** button.
- In the Select Files for Filtering dialog, CTRL+Click the files you wish to select (Character List files (p. 22) allow you to list commonly grouped characters together). You can also select all files by scrolling to the end of the file list and SHIFT+clicking the last file.
- Click **Open** to make the selection final. The files you chose will be displayed in the Files list box.
- You may click **Remove** to remove the highlighted file from the list (double-clicking the file also removes it).
- Click the name of the filter you wish to use in the Filters list box. When you click a filter name, a description of it will appear (if available) below the list.
- Click the **Multiple** button. The Extension edit field will become active.
- Enter the extension you wish to use for the new file. For example, if your original character sheet name is `Cuisinart.chr`, you would enter "html" for the extension to name the filtered file `Cuisinart.html`.
- Click the **Run** button. The button will be grayed out if you neglected to perform one of the necessary steps.
- As each file is successfully processed, it is removed from the File list.

When the filtering is complete, a new file with the extension you specified will exist in the directory with the original file.

Printing Multiple Character Sheets at Once

All the files you choose will be printed with a graphical character sheet template. This makes it quicker and easier to reprint a large number of character sheets because you don't have open each file separately and print it. You can also prevent any dialog boxes from popping up.

- Click the **Select Files...** button.
- In the Select Files for Filtering dialog, CTRL+Click the files you wish to select (Character List files (p. 22) allow you to list commonly grouped characters together). You can also select all files by scrolling to the end of the file list and SHIFT+clicking the last file.
- Click **Open** to make the selection final. The files you chose will be displayed in the Files list box.
- You may click **Remove** to remove the highlighted file from the list (double-clicking the file also removes it).

- Click the **Print Through Template** radio button.
- If you wish to be asked for a print template for each file, check the **Ask for Template** button. If this checkbox is unchecked, no further dialogs will be displayed. All character sheets will be printed with the print template selected in the file itself.
- Click the **Run** button.
- Each file will be removed from the Files list as it printed.

Printing Multiple Character Sheets through a Filter

All the selected files will be printed through the selected filter. The files can be printed separately or on the same page. The latter is especially useful for printing character summaries, where, for example, information on several characters is compiled on a single page for use during combat.

- Click the **Select Files...** button.
- In the Select Files for Filtering dialog, CTRL+Click the files you wish to select (Character List files (p. 22) allow you to list commonly grouped characters together). You can also select all files by scrolling to the end of the file list and SHIFT+clicking the last file.
- Click **Open** to make the selection final. The files you chose will be displayed in the Files list box.
- You may click **Remove** to remove the highlighted file from the list (double-clicking the file also removes it).
- Click the name of the filter you wish to use in the Filters list box. When you click a filter name, a description of it will appear (if available) below the list.
- Click the **Print through Filter** radio button.
- To filter all the files in the same print job (so that they appear on the same page), check the **Print on Same Page** checkbox. To print each file separately, leave that checkbox unchecked.
- To preview the appearance of the results of **Print on Same Page** check the **Preview** checkbox.
- Click the **Run** button.
- Each file will be processed and printed. If **Print on Same Page** is checked, the files will be printed after processing, or if you checked **Preview**, the result will be displayed on the screen.

Character List Files

Character list files are simple text files with the names of character sheets in them. These files are useful for remembering characters that are commonly printed together. Character list files have a `.clist` extension. To edit them, use any text editor that saves straight ASCII text files.

For example, you might put the names of the players' characters in a character list file called `PCs.clist`. The contents simply list the files, each file on a separate line, to be included in the group to be printed. For example:

```
Gorgon.chr
Stratus.chr
Diamond.chr
Aziz.chr
```

When you want to regenerate your combat sheets for the player characters, select `PCs.clist` instead of picking each individual character.

These character lists can also be used to list the non-player characters involved in a scenario, so that you can generate the combat sheet for them, or generate character summaries to include in your scenario writeup.

If one or more of the files doesn't exist, you'll be notified. The files that do exist will be placed in the list of files to be processed as normal.

Select Files for Filtering

Selecting More than One File

- Click the first file.
- SHIFT+click another file and Windows will select all files between the selected file and the one you click. (SHIFT+click means to press the SHIFT key, then, while you continue to hold it down, click the left button of mouse on the item you wish to select).
- CTRL+click files one at a time to add them to the selection.

Selecting All Files

- Scroll to the end of the list of files.
- SHIFT-click the last file in the list.

When you have selected all the files you want, click the **Open** button.

Save Filtered Text As

- Use this dialog to select or create the directory where you wish to save the filtered text.
- In the File name field, enter the name of the file you wish to save the filtered text info.
- Click the **Save** button.

The Name Finder

The Name Finder lets you search for names for your character. You can browse through all the names in the name database, or you can narrow your search by specifying attributes for the search. The currently selected name file is shown in the lower left hand corner of the dialog box.

Topics

Finding Names (p. 23)
Name Display (p. 24)
Search Examples (p. 24)
Creating New Name Files (p. 249)

To open a different name file click the Open... (p. 24) button.

Name Finder

- Click an entry (Male, Female) in the Primary Attribute list. The attribute will be highlighted.
- Click an entry in the Secondary Attribute list (Spanish, French, Russian, etc.). The attribute will be highlighted.
- Click the **Find** button.
- The names that match your criteria will be displayed in the Names list.

When you click the **Find** or **Random** button, *GURPS Character Builder* will search for all names that match any of the Primary Attributes **and** any of the Secondary Attributes. If you select no Primary or no Secondary Attributes, all names in the list with no selected attributes will match. If you select no attributes at all, every name in the database will be displayed.

Click... **To...**

Primary Attributes

Select an entry in the Primary Attributes list. Turn off a selected attribute by clicking it again.

Secondary Attributes

Select an entry in the Secondary Attributes list. Turn off a selected entry by clicking it again.

Find	Initiate the name search. This will find all names that match the selected attributes in the lists and place the matching names in the Names (p. 24) list.
Random	Initiate the name search (the same as Find) and pick one of the matching names at random.
Clear	Clear all the highlighted attributes and names.
Notes	Display any notes about the current name file.

Name Search Examples

Female Names

- Click Female in the Primary Attribute list.
- Click Find.
- A list of all female names in the open database will appear in the Names list.

Russian Male Names

- Click Male in the Primary Attribute list.
- Scroll the Secondary Attribute list to find Russian.
- Click Russian in the Secondary Attribute list.
- Click Find.
- The Names list will contain Russian male names.

Old French or Old German Male Names

- Click Male in the Primary Attribute list.
- Scroll the lower list box to find Old French. Click it.
- Click Old German.
- Click the **Find** button.
- The names list will contain the desired list. You can click **Random** to choose a name at random.

Names List

The **Names** list contains all the names that matched the search criteria.

Click...	To...
Info...	Get information on the name. The name, its attributes and the name's meaning, derivation or other information about the name will be displayed (if available).
Copy	Copy the name to the clipboard. You can paste the name into the other edit fields in the character sheet.
Paste	Paste the highlighted name into the window you started the Name Finder from. This button is active only in windows you can paste text into. The Name Finder dialog closes and pastes the selected name.

Opening Different Name Files

To search a different name database use the **Open...** button to select the database. A list of file names will appear. Select the one you want.

Die Roller

The die roller generates random numbers. When a random number is generated the actual die values rolled are displayed in the dice window on the right, with the total beneath.

In addition to rolling dice, you can flip coins (p. 26) and draw cards (p. 28).











To select the die specification:

- Click one of the die specification buttons on the left. This will generate a random number according to the specification on the button and place it in the **Die Roll** result box.
- Enter a die specification (p. 26) in the **Die Specification** edit box and click the **Roll** button. This die spec will be added to the list box if it isn't already present.
- Double-click one of the die specifications in the list box beneath the **Die Specification** edit box. This will replace the contents of the die spec edit box.

Previous die roll results are recorded in the **Die Roll** drop-down list.

Dice Window

The dice window displays the results of the roll. If you have opted to have the dice drawn, they will have the following appearance:

Image	Die Types
	Coin (d1). 0 indicates tails; 1 indicates heads
	d4
	d6
	d8
	d10
	d12
	d20
	d100. This is also used to display any other type of die.
	Fudge die
	Playing card

To display your own bitmaps for the dice or change the size, see the Die Roller Options (p. 29). The options also allow you to turn off the display of the die and just print the text.

Resizing the Die Roller

The Die Roller window can be resized. Move the cursor over the border of the window, click and drag.

Other Topics

- Setting the Text of Buttons (p. 26)
- Die Specifications (p. 26)
- Copying Die Results to the Clipboard (p. 29)
- Deleting Die Specifications from the List (p. 29)

Die Roller Options (p. 29)

Setting the Text of Buttons in the Die Roller

To set the die specification for a button:

- Type the die specification (p. 26) that you want in the Die Specification edit box.
- Hold down the shift key.
- Click the button that you want to set the text of.

The text in the **Die Specification** edit box will replace the text of the button.

The values of the buttons and the Die Specification list will be remembered between sessions.

Die Specifications

The following are some examples of common die specifications.

Specification	Description	Values
3d6	Roll three six-sided dice	3-18
d00	Roll percentile dice	0-99
2d6+1	Roll two six-siders and add 1	3-13
2d+1	As above	3-13
3d8-3	Roll three eight siders and subtract 3	0-21
4dF	Roll four Fudge dice	-4 - +4
3d1	Flip three coins and count the number of heads	0-3
3*d1	Flip three coins and display the results independently	Three 0-1 results
5p52	Draw five cards from a standard deck.	Five cards
5u10	Pick five unique numbers between 1 and 10.	Five numbers
7d10!6	Roll seven ten-siders and count the number of rolls of six or greater	0-7
6d6~1	Roll six six-siders and count the number of ones rolled	0-6

More details on the syntax of the die specifications follow.

Die Specifications

The simplest die specifications consist of the letter 'd' (or 'D') followed by a number. For example, "d6". This is like rolling a six-sided die; it will generate a number between 1 and 6. If you omit the number following the 'd', a six-sided die is assumed.

The following are examples of allowed specifications:

Die	Description	Value Range
d1	Two-sided coin	0 to 1
d2	Two-sided die	1 to 2
d3	Three-sided die	1 to 3
d4	Four-sided die	1 to 4
d6	Six-sided die	
d	Six-sided die (six assumed if number of sides omitted)	1 to 6
d9	"Nine-sided" die. Any positive integer number may follow the d.	1 to 9
d12	Twelve-side die	1 to 12
d20	Twenty-sided die	1 to 20
dF	Fudge die	-1 to +1
d0	Ten-sided die	0 to 9

d10	Ten-sided die	1 to 10
d00	Percentile die	0 to 99
d100	Hundred-sided die	1 to 100
d08	Eight-sided die minus one. A zero preceding the number generates a value between 0 and the number minus one.	0 to 7

Number of Dice

To roll a number of dice, specify a number before the 'd'. Examples, "3d6" (or "3d") will roll three six-sided dice, generating a number between 3 and 18.

Die Specification	Range of Results
3d6	3 to 18
2d	2 to 12 (six-siders assumed if the number of sides is omitted)
4dF	-4 to +4
3d10	3 to 30
3d0	0 to 27
5d1	0 to 5 (counts the number of times heads is rolled)

The maximum number of dice that may be specified is 32,767. The maximum number of sides is also 32,767.

Other Operations

You can perform addition (+), subtraction (-) and multiplication (x, the letter 'x') with die specifications and numbers. For example: 2d4+2 generates a number between 4 and 8, 2d-1 generates a number between 1 and 11, and 3d10x10+1 generates a number between 31 and 301. Note that there is no parenthesization.

Logical operators are also provided. This can be useful for making a roll and deciding whether it "succeeded." For example, "3d6>=10" will result in 1 if the value rolled is greater than or equal to 10.

Operator	Description
+	Addition
-	Subtraction
x	Multiplication
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<> or #	Not equal to

To make multiple rolls, separate each specification with a semicolon (";").

Best N of M Rolls

To specify that only so many highest-valued dice be summed use the ':' operator. For example, "3:4d6" would roll four dice and sum the three highest values. If the rolls are 6, 4, 3, and 1 the total is 13. The maximum number of dice (*m*) that may be rolled is 1000.

Number of Independent Rolls

The '*' operator specifies the number of independent rolls to be made. Each roll is separated from the previous one with a comma. Everything between the * and the end of the die specification is repeated.

Die Specification	Sample Results
3*3d	12,8,15
5*d1	1,1,0,1,0

Choose Without Replacement

Normal die rolls generate random numbers with replacement; that is, the same random number can occur more than once. Also see Playing cards (p. 28) below. For example, rolling 4d6 can roll 4 sixes. Choosing numbers without replacement prevents the same number from being generated more than once. This kind of random number generation is used for drawing cards or generating lottery numbers.

To specify that n random numbers be chosen from a range of r and then summed, specify ncr . To specify n unique random numbers be chosen and listed separated by commas, specify nur .

Die Specification	Result
3c6	Choose 3 unique values in the range from 1 to 6 and sum them.
6c0	Choose 6 unique values in the range from 0 to 9 and sum them.
10u00	Choose 10 unique values in the range 0 to 100 and list them separately.
10u10	Randomly order the numbers between 1 and 10.
5u52	Choose 5 unique values in the range 1 to 52 and list them separately.

It is an error for n to be greater than r . The maximum value for r is 1000.

Counting Successes and Values Rolled

To count the number "successful" die rolls add a "!" and a number after the die specification: " $nds!v$ ", where n is the number of dice, s is the number of sides and v is the value of a successful roll. The result will be the number of dice that are equal to or greater than v .

To count the number of times a specific number is rolled, add a "~" and a number after the die specification: " $nds~v$ ", with the values as above, except that only dice that roll exactly v will be counted.

Die Specification	Result
7d10!6	The number of 10-sided dice that rolled 6 or more.
10d2~2	The number of 2s rolled on 10d2. This is equivalent to counting the number of "heads" on 10 coin flips.
6d6~1	The number of 1s rolled on 6d6.

Playing Cards

The Die Roller also draws cards. This is similar to choosing random numbers without replacement, except that instead of generating numbers, cards are drawn. The specification is: npd , where n is the number of cards to draw and d is the deck size.

The deck size must be between 4 and 54. When a deck size of less than 52 is specified, only the top end of the suit is used. If the deck size is not a multiple of four, the extra cards are assumed to be jokers. For example, if you specify a deck size of 44, there are 11 cards in each suit, with the 2s and 3s omitted.

Some examples:

Specification	Cards
5p54	Draw five cards from the standard deck with two jokers
7p53	Draw seven cards from the standard deck with one joker
5p52	Draw five cards from the standard deck with no jokers
10p45	Draw 10 cards from the fours through aces and one joker
4p16	Draw four face cards
52p52	Randomly order the standard deck.

The following symbols are used to indicate the cards.

Symbol	Card
C	Clubs (♣)
S	Spades (♠)

D	Diamonds (♦)
H	Hearts (♥)
2x to 10x	The card's value, where x is one of the above suits
Jx	Jack
Qx	Queen
Kx	King
Ax	Ace
J	Joker
J1	First Joker
J2	Second Joker

Playing card specifications do not result in numbers; you cannot do arithmetic on them. Any other expressions appearing after a playing card specification will be treated as a separate entity.

Copying Die Rolls to the Clipboard

To copy the contents of the latest die roll to the clipboard, click the **Copy** button. If you have rolled dice, the **Copy** button will be active.

Deleting Die Specifications

To remove a die spec from the die spec list:

- Click the die spec you want to delete.
- Click the **Delete** button.

When you close the Die Roller, the changes you have made to the list will be saved away.

Die Roller Options

Show Die Rolls

If checked, the die rolls are displayed on the right side of the Die Roller window. If unchecked, the size of the Die Roller dialog is reduced and only the totals are displayed.

Always On Top

If checked, the Die Roller window is always on top of other applications.

Show Body Total

If checked, a "BODY" total is calculated and displayed for any d6 values rolled. The BODY is defined as 0 for one pip, 1 for two to five pips and 2 for six pips.

Draw Dice

If checked, representations of the dice are drawn beneath the die values, to show what type of die generated which number. If unchecked, only the values of the die rolls are displayed.

Draw Pips on Six-Siders

If checked, pips (dots) are drawn on the six-sided dice. If unchecked, the values are displayed in Arabic numerals.

Play Sound Effects

If this button is checked, die-rolling sound effects are played when die rolls are performed. To provide your own die-rolling sound effects, replace the following files in the `dice` subdirectory of the source directory (p. 17).

File Name	Description
1die.wav	A single die rolling
2dice.wav	Two dice rolling
3dice.wav	Three dice rolling
severaldice.wav	Four-six dice rolling
manydice.wav	More than six dice rolling

coin.wav	The sound of a coin
cards.wav	The sound of cards

If replace these files, reinstalling the application will overwrite your custom files. Be sure your original sound files are backed up.

Use Bitmap Images of Dice if Present

If this button is checked, the application checks for the existence of JPEG image files in the `dice` subdirectory of the source directory (p. 17). If these files are present, the images are used to render the dice in the dice window.

The files are named with the following convention: the letter 'd' followed by the number of sides, followed by a tilde ('~') followed by the value, followed by '.jpg'. For die specs such as d0, d06, etc., the name is the letter 'd', followed by the number of sides, followed by '-1', followed by the value (0 through n-1), followed by '.jpg'

To provide your own images of six-sided dice for display in the dice window, create the following JPEG image files in the `dice` directory:

```
d6~1.jpg
d6~2.jpg
d6~3.jpg
d6~4.jpg
d6~5.jpg
d6~6.jpg
```

The names for other images are:

File Name	Description
dF~- .jpg	Fudge die: -1
dF~0 .jpg	Fudge die: 0
dF~+ .jpg	Fudge die: +1
d10~1 .jpg	Ten-sider 1-10: 1
d10~1~0 .jpg	Ten-sider 0-9: 0
2C.jpg	Two of Clubs
d2~1~0 .jpg	Tails of a coin
d2~1~1 .jpg	Heads of a coin
JH.jpg	Jack of Hearts
QS.jpg	Queen of Spades
AD.jpg	Ace of Diamonds
J.jpg	Joker
J1.jpg	First Joker
J2.jpg	Second Joker
...	And so on

Die Font Size

Enter the point size for the font used to display the rolls. This sets the size of the dice drawn behind the numbers as well. Values between 8 and 128 are allowed. The default value is 16.

Recalculate

Recalculates the values of all the items in the character sheet.

Print Auxiliary Files

This dialog provides a way to print auxiliary files supplied with the application. These files include the manuals, blank character sheets, and so on.

The text in the dialog describes special features of the files. Scroll through it for answers to your questions specific to the files listed in the **Files** list.

Printing a File

- Click the file you wish to print in the **Files** list.
- To print the file immediately, click the **Print** button.
- To open the file in the printing application (usually Adobe Acrobat Reader), click the **Open** button. If you wish to select printing options (specify the printer, only print certain pages, etc.), you should select this.

Files

The list of files that can be printed. Double-click a file in this list to open it for printing, or click it and click **Print** to print it immediately.

Open

Opens the file highlighted in the **Files** list in the reading application (usually Adobe Acrobat Reader). Clicking **Open** will start up the reading application and load the file. You'll need to manually print the file and close the application.

This is useful when you want to specify printing options, select the printer, etc.

Print

Prints the highlighted file through the reading application. Clicking **Print** will usually print the file immediately, without providing any printing options, and close the printing application once the file has been printed. The exact behavior depends on the reader application, however.

To gain more control over the printing process, use the **Open** button instead.

If you use the **Print** button, make sure you have selected a default printer in Printer Settings. To select the default printer, click the Windows **Start** button, then select the **Settings | Printers** command. Right-click the printer you wish to use, then select **Set as Default**.

Data Menu

The Data menu contains the dialogs (p. 50) and lists (p. 55) that contain the data in the character sheet.

The first entry in the data menu is the main dialog (p. 50) for the character sheet. The other dialogs, follow, then the lists.

To open a dialog or list window, select its name from the **Data** menu.

Command Scripts (p. 260) may also appear in the **Data** menu. Command Scripts perform automated sequences of commands on the character sheet. When you select a command script from the **Data** menu, its script will be executed.

Tools Menu

The commands listed on the Tools menu will differ based on the type of window that is open, and the operations you are conducting.

Character Sheet Tools Menu (p. 33)

Data Sheet Tools Menu (p. 39)



Print Template Tools Menu (p. 40)

Character Sheet Tools Menu

Some commands on this menu are active only when a list is active, and others are active only in a dialog.

Select Item... Lists only: Bring up a list of items (or categories of items) to select from. Choose this command to add a new item to the list. The Available Items (p. 56) dialog will appear. Double-click items in the Available Items dialog to select them.

Insert Sublist...

Lists only: Create a sublist, which is indicated by a  icon. You can add items to this sublist by clicking on the  icon to open it (or by selecting the **Open/Close Sublist...** command), then highlighting the sublist or any of its members. Finally, use the **Select Item...** command to add the new item.

Select From Master List...

Lists only: Choose items from any data sheet item list, not just the default item list associated with the current list. You select the data sheet category (p. 72) you're interested in and proceed as normal from there.

Modify... Lists only: Change the information (name, value, option list, etc.) about an item or sublist.

Open/Close Sublist...

Lists only: Open a sublist so that its members appear. If the sublist is already open, it is closed. The command opens an item for editing the same way the **Modify...** command does.

Delete Item Lists only: Delete the highlighted item. It is the same as the **Clear** command on the Edit menu. You can also use the **Copy**, **Cut** and **Paste** commands on the Edit menu in the item lists.

Edit Properties...

Lists only: Bring up the edit properties (p. 75) dialog.

Notes... Lists only: Edit the notes (p. 73) for the highlighted item.

Show Requirements...



Lists only: Show the list of requirements (p. 35) for the highlighted item. This command is dimmed if the item has no requirements.

Check Requirements...

Check requirements (p. 34) for all items in all lists, and presents a list of those items which have requirements that are not met.

Add 1 Add 1 to the value of the currently selected item in a list, or to the variable for the edit field that contains the caret. The keyboard equivalent of this is CTRL-UP. If the SHIFT key is also depressed, the value will be increased by 5.

Subtract 1 Subtract 1 from the value of the currently selected item in a list, or from the variable for the edit field that contains the caret. The keyboard equivalent of this is CTRL-DOWN. If the SHIFT key is also depressed, the value will be decreased by 5.

Move **Move | Up** moves the highlighted item up. **Move | Down** moves the highlighted item down. More useful are the buttons  and , and the keyboard equivalents, SHIFT+UP and SHIFT+DOWN.

Moving up an item that is beneath an open sublist moves it into the sublist. Moving down the last item in a sublist moves it out of the sublist.

List Summary

Lists only: Present a list of the categories of items in the current list (p. 36), and the number of items in each category.

Reset Field

Dialogs only: "Reset" the edit field which the cursor is currently to the default value. If there is no default value defined for the edit field, this has no effect. In some game systems a character's initial attributes are determined by a die roll. In this case, the default value would be a random number, and selecting **Reset Field** would "reroll" the value.

When an edit field is set to the default value, it is displayed in bold. If you set the field to an explicit value it will be displayed in the normal font.

Reset All Fields

Dialogs only: "Reset" all edit fields in the dialog to the default value, as above. In a character sheet with randomly generated attributes, this "rerolls" the character.

Sort...

Lists only: Sort the items in the list.

List Format

Lists only: Select the list display format. Some game systems have more than one format that items are displayed in. To select an alternate display format, select the submenu command corresponding to the desired format. The currently select format is checked. Select the first entry, **Default**, to restore the default list format. If the submenu is grayed, then there are no alternate display formats for the list.

Check Requirements

Checks all items to make sure their requirements (sometimes called prerequisites) are satisfied. Any unsatisfied requirements are displayed in a list in a dialog box called Unsatisfied Requirements, with a count of the number of unsatisfied requirements.

The summary shows the name of the list to which the item with unsatisfied requirements belongs, followed by the name of the item.

Show

Display the highlighted item's requirements (p. 35).

Satisfy

Attempt to satisfy the highlighted item's requirements. Required items are added to the appropriate list at the specified level, required values in dialogs are set accordingly. Some types of requirements cannot be satisfied automatically; these are left unsatisfied (for example, NOT requirements are not deleted if they are found -- it's up to the user to decide to delete something).

Summary

Display a summary of the items in the list the highlighted item belongs to.

Satisfy All

Attempt to satisfy all item requirements. As above, for all items with unsatisfied requirements. Note that if you use this command, some types of requirements (for the number of items in a category, total number of items, etc.) may be erroneously satisfied in the future.

Satisfy Count of Items in a Category

This dialog displays a list of the items when you tell **GURPS Character Builder** to satisfy a requirement for a number of items in a category. You choose the items you wish to add, and **GURPS Character Builder** will satisfy the requirements for them (if possible).

Required: #; Present: #

This gives the number of items required in the category, and the number currently present. As you add items, the number present will increase. When the number present equals or exceeds the number required, this dialog automatically dismisses itself.

Add

Adds the highlighted item to the item list, plus any other items it requires.

Show

Shows the requirements for the highlighted item.

Requirements

All requirements are listed below the item name. Unsatisfied requirements are indicated with an asterisk (*). Remember that if you rename an item, other items that require it will not be able to find it under the new name.

This dialog may be resized by clicking and dragging on the border.

Turning off Requirements

See the **Disallow If Not Satisfied** checkbox in the Preferences (p. 15) dialog for information on controlling how requirements are enforced.

Types of Requirements

There are several types of requirements:

THEN and OR

Requirements may be grouped together with THEN and OR. To satisfy an OR grouping, all the items between two ORs must be satisfied. To satisfy a THEN grouping, all the items in it must be satisfied, or if an OR is present, at least one OR grouping must be satisfied.

For example:

```
Spells: Acute Vision
--- OR ---
5 Light Spells
--- THEN ---
NOT Disadvantages:Blindness
NOT Disadvantages:Poor Vision
```

This indicates that you must have either Acute Vision or 5 Light spells, **and** you may not have Blindness or Poor Vision Disadvantages.

Item Level Requirements

If the item's value is not allowed, a message will be displayed that states this.

Required Value

Checks that the named value is in the proper range. For example, "Attributes: Strength>=12" means that the Strength value in the "Attributes" dialog must be at least 12.

Named Item

Checks that the named item is present in the named list. For example, "Spells: Wish" means that the item "Wish" is required in the "Spells" list.

If an item is renamed, the requirement will still be satisfied as long as the item's original name is left intact. The original name is set in the item properties (p. 75).

Excluded Item

If another item has a requirement that excludes the current item, you will be notified. For example, "Advantages:Wealth excludes this item" would be displayed if you chose the Poverty Disadvantage. This allows you to specify exclusions on only one item, yet still be notified of the conflict when either item is chosen.

Number of Items in List

Makes sure you have at least the specified number of items in the indicated list. For example, "6 Spells" means that you must have 6 items in the Spells list.

A minimum level can also be specified for the items in the list. For example, "6 Spells>=12" indicates that there must be at least six spells at level 12 or greater.

Number of Items in a Category in a List

Checks that you have at least the specified number of items in the indicated category in the indicated list. For example, "5 Combat Skills" means that you must have at least 5 items belonging to the "Combat" category in the "Skills" list.

If a value is specified, it means that the specified category of item must have the specified level. For example, "1 Magery Advantages>=2" means that you must have at least one item of the Magery category in the Advantages list with a value of 2 or greater. If you already have an item of that category at a lower level, reselecting that same item from this list will set the item's level to the required value.

Number of Items in a Number of Categories

Checks that you have at least the specified number of items in the specified number of categories in the indicated list. For example, "3 item(s) in 4 categories in Skills" means that you must have at least three different items in four different categories in the Skills list.

A level can also be specified. For example, "1 item(s) in 10 categories in Spells>=12" means that you must have at least one spell in 10 different categories that have a level of 12 or more.

Expression

Checks that a particular condition exists in the character sheet. For example, "Magecraft>2".

Option on an Item

Checks that the named item is present in the named list and that it has an option, possibly with a particular value. For example, "Skills: Science (Option: Specialty=Astronomy)" means that there must be an item named Science in the Skills list with an option named Specialty that has a value of "Astronomy".

Buttons

Satisfy

Attempt to satisfy (p. 34) the highlighted requirement. Required items are added to the specified list at the required level (if specified), and required values in dialogs are set accordingly.

Summary

Display a summary of the items in the list.

Satisfy All

Perform the Satisfy operation on all unsatisfied requirements listed. Some types of requirements cannot be satisfied automatically; these are left unsatisfied.

List Summary

All the items in the current list are counted for all the categories referenced, and the results are tallied here. Some game systems require a certain number of items in various categories to satisfy requirements for certain other items. This provides a handy place to look for this information.

The item count is shown for each category, followed by the category name and the total cost of items in that category in parentheses. At the bottom, the grand total of the number of items and the number of categories is shown. An item can belong to more than one category.

Items in "hidden" categories (those categories whose names start with "**") are not enumerated in the list summary.

Looking at Other Lists

To switch to another list, click the **Lists** dropdown list and click the desired list. The information for that list will be displayed in the listbox.

Alternate Format

This dialog is used to edit the alternate formats for lists and items. More details can be found in the description of the alternate formats for lists (p. 44).

Name

The name of the format. This can be left empty for an item format, in which case it is the default format. The format string (below) is inserted into the list format string wherever the \$! sequence occurs.

For a list, this name will be displayed in the **Tools | List Format** submenu, under the Default entry.

Header

The header for the format. This header will replace the default header. For example:

```
%01!Name%-25r!BC%-20r!Add%-15r!Mult%-7r!Cost
```

This is ignored for items and should be left empty.

Format

Lists: the format to be used for items in the list. If you wish to allow items to specify their own "subformat," place the \$! sequence where you wish the item-specific formatting to be inserted.

For example:

```
%01!$!%-25rx|@bc@|%-20rx|@a@|%-15rx|@m@|%-7rc
```

Items: if the format name matches the selected format, this format string overrides the list format. In this case, the format should be complete with field specifiers. For example:

```
%01n%-7rc
```

If the format name is empty, this is the "(Default)" format. It will be inserted into the list format wherever the \$! sequence occurs. In this case, there should be no field specifiers (no % signs). For example, to display the item name and level use

```
^n-^v
```

Sort

The Sort dialog sorts the items in the list. The options you select here will be remembered the next time you sort.

Sort By

This indicates the main sort criterion.

Name	Sort alphabetically by name.
Category	Sort alphabetically by category, and then by name within category.
Class	Sort alphabetically by the Class property of the items. Exactly what this represents is dependent on the item.
Level	Sort by level, lowest level first.
Cost	Sort by cost, lowest cost first.

Reverse Order

If this is checked, the items are sorted in the reverse order (i.e., if sorted by name, then the sort is reverse alphabetical).

Sublists

The options here indicate how sublists are treated.

Intersperse	Sublists are interspersed among all the items.
Beginning	Sublists are sorted at the beginning of the list, sorted among themselves by the main criterion.

End Sublists are sorted at the end of the list, sorted among themselves by the main criterion.

Recursive sort

If this is checked, then all sublists included in the sort are also sorted.

Notes

If you have automatic items you may wish to add the parent item to a sublist so that you can sort the list and keep the automatic items near the parent.

Data Sheet Tools Menu

These commands are available only in *GURPS Character Builder*.

New Item... Create a new item in the data sheet. This opens up a dialog box in which you may edit (p. 75) the new item. The item is added after the currently highlighted item in the list. If you are creating a large number of items, it is extremely useful to create "template" items that you Copy and then Paste into the data sheet. You may not create a new item at the highest level of the data sheet. You must insert a sublist first, and then you may insert new items into the sublist.

Insert Sublist... Creates a sublist in the data sheet after the currently highlighted item. You must create a sublist before you can add other items.

Modify Item... Allows you to modify the currently highlighted item. You can change the name of sublists with this command, as well as edit (p. 75) the properties of items.

Delete Item Deletes the highlighted item. Use this command with care: if you delete a sublist, everything inside it will also be deleted.

Reload Data Sheet If the data sheet is currently loaded (p. 4), this command causes it to be reloaded, making its contents available to character sheets.

Unload Data Sheet Removes the contents of the data sheet from memory, so that they cannot be used by character sheets.

Print Template Tools Menu

The commands on the Tools menu add new objects to the print template. These commands are available only in ***GURPS Character Builder***.

Cursor Turns off object addition and allows you select objects once more by clicking and dragging to form a selection rectangle.

Rectangle Add a rectangle.

Rounded Rectangle
Add a rounded rectangle.

Ellipse Add an ellipse.

Line Add a line.

Horizontal Line
Add a horizontal line. This line can only be lengthened and shortened; its orientation cannot be changed.

Vertical Line Add a vertical line. This line can only be lengthened and shortened; its orientation cannot be changed.

Text Add text. This text will be displayed exactly as entered.

Text Box Add a text box. Any filter references (p. 185) will be expanded according to the data in the data sheet being printed.

Picture Add a picture reference (p. 118).

Modify Menu

The Modify menu has commands that let you change the structure of the character sheet. These commands are available only in **GURPS Character Builder**.

Dialog Item Properties

When this command is selected, it becomes checked or unchecked. When checked, you are editing the dialog box (p. 155) itself instead of the values in it. When checked, a different set of menus are available.

Window Information...

Lets you change information about the current dialog (p. 41) or list (p. 42) window.

Defines

Allows you to edit the defines (p. 166).

Character Sheet Info...

Allows you change information about the character sheet (p. 45).

Data Menu

Allows you to modify the data menu (p. 41). You can add new dialogs and lists, set the order of the window names on the **Data** menu and delete unwanted dialogs and lists.

Modify Data Menu

This dialog lets you change the appearance of the **Data** menu. You can add and delete dialogs and lists, change the order that they appear in the **Data** menu, and add menu item separators. Note that you cannot cancel any changes made here; once you have made the change it is permanent, unless you revert to the original file.

Up

Move the currently highlighted menu item up.

Down

Move the currently highlighted menu item down.

Edit

Allows you to change the currently highlighted dialog (p. 41) or list (p. 42).

Delete

Deletes the currently highlighted separator, list or dialog. If you answer Yes to the delete query, the window will be deleted permanently. To get it back, you must open the original file again without saving the changed character sheet.

New Dialog

Creates a new dialog (p. 41), adding it to the menu after the highlighted item. You may have up to 12 dialogs.

New List

Creates a new list (p. 42), adding it to the menu after the highlighted item. You may have up to 12 lists.

Separator

Adds a menu separator after the highlighted item. You may have up to 5 separators.

Modify Dialog Information

This dialog allows you to create and change information about a dialog.

Dialog Name

The name of the dialog. This will appear in the caption of the dialog window, and also in the **Data** menu for the character sheet.

Check Expression and Message

This expression is evaluated every time one of the edit fields in the dialog is changed. If the expression evaluates to false, the Check Message is displayed. You can use this to warn the user if an illegal setting has been made. For example, some games systems disallow more than one computed attribute to have a negative cost. The following expression would make that check (given that the costs of the attributes are c_attr1, c_attr2, etc.)

```
(c_attr1<0)+(c_attr2<0)+(c_attr3)<2
```

If the check message (p. 158) begins with a "\$" the rest of the message string will be evaluated as an expression in the context of the character sheet. If it begins with any other character it will be presented to the user verbatim.

Default Check Expression and Message

Each edit field in a dialog can have its own check expression, but if all the edit fields share the same restrictions (must be greater than 0, for example), you can set a default check expression. If you set a check expression on the edit field directly, it is evaluated instead of the default check expression.

The default Check Message will be displayed if the default check expression evaluates to false (0). As with the dialog check message, if the default check message (p. 158) begins with a "\$" the rest of the message string will be evaluated as an expression.

Font

To select the font used in the dialog click the **Change** button. This font is the "default font" for the dialog, and will be used for any controls that don't specify their own font (p. 163).

Skin

A dialog window "skin" is a bitmap that serves as a background for the dialog. The **Skin** field contains the name of the bitmap file to use as the skin, relative to the **GURPS Character Builder** source directory (p. 17). Click the **Browse...** button to browse the file system for bitmap files to use as skins. If the skin is less than three-fourths the width or height of the dialog box, it will be "tiled" -- displayed in a square array as many times as needed to cover the entire face of the dialog.

Controls can be placed over the skin, allowing you to have any design you choose as the background for the dialog. The skin can be a BMP, JPEG or PNG file. BMP files are most efficient in terms of memory usage, because other file formats must also be rendered as BMP images internally to be displayed under Windows. One copy of the skin file will be loaded and shared among all the dialog windows using that skin. You will probably want text and formula controls to be transparent (p. 160).

The skin file must be located in the **GURPS Character Builder** source directory (p. 17), or in one of its subdirectories.

If the skin name begins with a \$ the value is interpreted as an expression that is evaluated in the context of the character sheet. This allows different skins to be displayed based on values in the character sheet. The character sheet file must be closed and reopened for the new skin to take effect.

Use Dialog Background Color

Check this checkbox to specify a background color for the dialog. By default, the Windows dialog background color is used, or the Dialog Background color set in the Preferences dialog (p. 15). Check the **Set...** button to change the dialog background color.

The dialog background color can be used in conjunction with a skin to ensure that text and formula controls have a background color that matches the skin.

Keyword

The Help Keyword that is used to display help for this window. When you create a help file for a game system, the link between the dialog and the help file is usually a *keyword* in the form "Dialog: Dialog Name". If the keyword is present, it overrides the dialog name. For example, if you have a keyword of "Vehicle Characteristics", then the help keyword would be "Dialog: Vehicle Characteristics".

Menu Character

You can indicate the character to be used to select the dialog from the **Data** menu by specifying it here. The first instance of that character in the dialog name is underlined in the menu.

Modify List Information

You can add and change list window definitions with this dialog.

List Name

This is the name of the list. It will appear in the caption of the list window, as well as in the **Data** menu.

Var. Name

This is the name of the variable that will be set to the sum of all the costs of all the items that appear in the list. You might access this variable in a formula field in a dialog when summarizing character point costs.

Cat. Name

The name of the category of items that is you can select from when you insert items into this list. If you leave this blank, you must select the category each time you insert an item into the list.

Font

You can select the font to be used in the list window by clicking the **Change** button.

Menu Character

The character to be used in the **Data** menu to select the list from the menu. The first instance of that character in the list name is underlined in the menu.

Cost Prompt

Replaces the "cost" prompt with the specified string in the item editing dialog for members of the list. This is useful for game systems that use different terminology for the point cost, or for labeling the prompt when a different unit is being used for the cost. For example, you might wish to enter the item's weight instead of a cost in a list of possessions.

Item Format

The format (p. 102) used for all items in the list that have no format (p. 102) of their own.

Dialog

Selects the type of dialog (p. 104) to use when editing members of this list. If an item specifies its own dialog type, that type will have precedence over the type specified for the list.

Keyword

The Help Keyword that is used to display help for this window. When you create a help file for a game system, the link between the dialog and the help file is usually a *keyword* in the form "List: List Name". If the keyword is present, it overrides the list name. For example, if you have a keyword of "Vehicle Powers", then the help keyword would be "List: Vehicle Powers".

Level Prompt

Replaces the "level" prompt with the specified string in the item editing dialog for members of this list. This is useful for game systems that use different terminology (such as "score"), or for labeling non-numerical values for items.

Cost Prompt

The text that will be used as the prompt for the cost in the item edit dialog (p. 68). You might want this to read "Weight" instead of "Cost" for a list of possessions, where you define the "cost" of a possession as its weight.

Total Prompt

The prompt that appears at the bottom of the list window by the total of all the costs in the list. You can reference the values of expressions by specifying a format (p. 102) string with embedded expression references (^x'expression').

Enter a single space into this field to indicate that there should be no total prompt. The list will fill the entire window.

Header

The prompt that appears at the top of the list window. Use a format (p. 102) string to achieve the desired alignment with the items appearing in the list.

Check Expression

This expression is evaluated whenever an item is changed or added to the list. If the expression evaluates to FALSE (zero), the Check Message is displayed. If you cancel, the action that caused the expression to fail is rescinded -- an item's level is reset to its original value, or a pasted item is deleted. This expression is also evaluated when you check requirements.

You can use this alert the user of an infraction of rules. For example, if the game system allows only one skill to be above 15, the following expression could be used:

```
countItems (p. 179) ("Skills", 15, 1, "level")<=1
```

Check Message

This message is displayed when the Check Expression (p. 158) fails. If the check message begins with a "\$" the rest of the message string will be evaluated as an expression in the context of the character sheet. For example:

```
$format("Only %d points may be spent on skills.", maxSkills)
```

If the message begins with any other character it will be displayed verbatim.

Default Check Message

This message is displayed when the check expression for an item in the list fails and the item has no check message defined. If the default check message begins with a "\$" the rest of the message string will be evaluated as an expression in the context of the character sheet. If the message begins with any other character it will be displayed verbatim.

Change Exp.

The change expression is used to detect changes to variables that are referenced in the Total Prompt so that it is properly updated. You would use this only in the case where variables referenced in the total prompt aren't affected by the variable defined for the list total. For example, if your total prompt displays the character's wealth in addition to the total weight of the equipment purchased, nothing would cause the total to be replotted if the wealth changed.

The change expression is any arithmetic expression that involves the variables you wish to check for changes. For example, if the total prompt displays `availableMoney` and `totalWealth`, your expression would be `"availableMoney+totalWealth"`.

The variable associated with the change expression is a unique variable generated from the list variable name. You can explicitly specify the variable to use (in case you wish to reference it elsewhere in the character sheet) by indicating the variable, an equals sign and the expression. For example, `"equipChange=availableMoney+totalWealth"`.

Alternate Formats

The alternate formats list defines alternate list display formats. This can be used to define a "simple" default item display format, and one or more detailed formats. This might be used to define an alternate format the displays details such as modifier totals, costs before modifiers are applied, after modifiers are applied, etc.

Each alternate format has a name. The user selects the format to be displayed by choosing one of the command on the **Tools | List Format** submenu. When an item is displayed, the standard format is used if the current list format is the Default. If an alternate format is chosen, the format on the item with the same name is used if it is present. If there is no matching alternate format on the item, the format with the empty name (displayed as "(Default)" when editing the item's properties) is substituted for the occurrence of the sequence "\$!" in the selected alternate format. If there is matching format on the item, the format "^n" (the item name) is used.

Click **Add** to add a new format to the list. Click **Delete** to delete the highlighted format. Click **Edit** to change the highlighted format.

Alternate Formatting Examples

The following alternate format would display the item's detailed information: the name (or special item formatting, indicated by \$!), the base cost, the total additions, the total multipliers and the total cost:

```
%01!$!%-25rx|@bc@|%-20rx|@a@|%-15rx|@m@|%-7rc
```

The corresponding header for this format would be:

```
%01!Name%-25r!BC%-20r!Add%-15r!Mult%-7r!Cost
```


To override the default formatting for the item (p. 103) (which is just the name), add an alternate format for the item that has an empty name (this will be displayed in the alternate formats list as "(Default)"). The following displays name and level:

```
^n-^v
```

To completely override the alternate formatting, add an alternate format that has the same name as the alternate format for the list. Assuming that the name of the example alternate list format above is "Details," add an alternate format named Details. This could be used to prevent a sublist having the additional details. Add an alternate format named "Details" that has the value:

```
%0ln%-7rc
```

The above examples would be displayed this way when Details is the selected format:

Name	BC	Add	Mult	Cost
 Sublist				25
Fireball-6	30	10	0.275	10
Lightning Bolt-2	10	5	0.85	15

Modify Character Sheet Info

This dialog lets you set basic information about the character sheet.

Topics

- Basic (p. 45)
- Calculations (p. 46)
- Data Sheets (p. 46)
- Auxiliary Files (p. 46)

Basic Character Sheet Info

Game System

Sets the game system (p. 298) type. This determines which data sheets will be used for this character sheet. You may not leave the Game System blank. Only data sheets that use the same game system can be included in the data sheet list. If the data sheets (p. 46) don't match this game system you will not be able to save your changes.

Sheet Type

The character sheet type. This determines which print templates and copy filters may be used with this character sheet. For example, some game systems have different character sheet templates for Characters and Vehicles, but both types use the same data sheet. If the sheet type for the print template or copy filter is left blank, any character sheet type may be used with it.

Level

The "release level" for the character sheet. This should match the levels of the data sheet and print templates for the same game system. If significant changes are made to the definitions of functions in the character sheet (which are used in the data sheet), this level should be increased, and the levels of the data sheet and print template should be changed to match. Then, if an old version of a character sheet is loaded with the new data sheet, you will be notified of the mismatch.

Template File

The original character sheet template file used to create this character sheet. This is used when a character sheet is converted (p. 258) to a newer version of the character sheet template.

Description

A short description of the character sheet or template. This is displayed when a character sheet template is highlighted in the New File (p. 2) dialog.

Copyright

The copyright message for the character sheet.

Calculations

Rounding

The type of rounding that is done for this character sheet. This is used when computing costs (p. 92) with options. **Floor** means that the fractional part of the cost will be thrown away, reducing the cost to the next lowest whole number if there is a fractional part. **Ceiling** means that the cost will be increased to the next whole number, if there is a fractional part. **Up** means the cost will be rounded up if the fractional part is 1/2 or greater. **Down** means the cost will be rounded down if the fractional part is 1/2 or less.

Constant Cost

If the Constant Cost checkbox is checked, then *GURPS Character Builder* will attempt to keep the costs of items in lists constant when you change the values they depend on. If you do not check the Constant Cost checkbox, *GURPS Character Builder* will attempt to keep the item levels constant, allowing the costs to "float." Different game systems use different defaults for this, and some game systems may use both: allowing the cost to float during initial character creation, and fixing the cost during later updates.

Default Total Cost Formula

This formula is applied against all items when the total cost (p. 107) is computed. It is applied to the cost of an item when the item has no total cost formula of its own. If the default total cost formula is omitted, then the cost is computed as described in Total Cost Formula (p. 107).

Max. Attempts

The maximum number of attempts made when setting the level of an item that you're adding, or changing with the **Tools | Add 1** and **Subtract 1** commands. Lowering this value can speed up the search when you enter incorrect values; increasing this value allows you to broaden the automatic search, at the expense of slower evaluation.

GURPS Character Builder searches for a value that meets the check expression for the item. The search starts at the last value the user set directly (the *desired level*). Then values up the desired level plus Max. Attempts are searched. Finally, values down to the desired level minus Max. Attempts are search. For example, if the last level the user set directly was 12 and Max. Attempts is 50, values up to 62 are first tried, then values down to -38.

If no check expression is present this search is not made.

Data Sheets

Loaded Data Sheets

The list of data sheets (p. 298) that will be automatically loaded when this character sheet is opened. Click **Remove >>** to remove the highlighted data sheet from the list.

Available Data Sheets

The data sheets (p. 298) that can be loaded. Click **<< Add** to add a data sheet to the list of data sheets that will be loaded when the character sheet is opened. Character sheets may also be loaded as data.

The data sheet's game system must match the game system of the character sheet.

Remove >>

Remove the highlighted entry from the **Loaded Data Sheets** list. The entry will be moved to the **Available Data Sheets** list.

<< Add

Add the highlighted entry to the **Loaded Data Sheets** list. The entry will be removed from the **Available Data Sheets** list.

Auxiliary Files

Help File

The name of the game-system specific help file to use for this character sheet.

Load Script

This command script (p. 260) is loaded and executed whenever a character sheet is opened. The context (defined variables, arrays, subroutines, etc.) will be retained as long as the file is open. The subroutines defined here may be called from the scripts associated with dialog controls.

The Load Scripts must be stored in the **GURPS Character Builder** source directory (p. 17). The default extension is `.script`.

Note that the context for the scripts is **not** the same context as the character sheet. Therefore, any references to character sheet variables from the script must be prefixed with `$@` (or `$*`, though the latter is preferred), just as with Conversion and Command Scripts.

Command scripts executed from the **Data** menu do not have access to the context and subroutines of the Load Script.

Shortcut File

The name of the shortcut file to search for shortcuts (p. 242) in.

Chargen Template

The name of the target character sheet template to use during character generation with the **File | Generate Character** (p. 261) command.

Chargen Script

The name of the conversion script to use during character generation (p. 261). The default extension is `.cnv`, but any extension may be used.

Run On New

If a Chargen script is specified, it will be executed whenever a character sheet based on this template is created. This is useful for game systems that have "scripted" character generation.

Window Menu

The Window menu contains commands that control the windows.

Pack Windows

Pack all open windows of the currently active character sheet into a rectangular array. All windows are moved to the left and then up, packing them as closely as possible to the windows on those sides.

If there is no overlap, list windows are then widened until they run into the nearest window on the right, or even with the right-most window edge, or the right side of the application window. Finally, list windows are grown vertically until they run into a window below, the bottom-most window edge or the bottom of the application window.

For best results, arrange the windows manually leaving a small amount of space between the windows, then select the **Windows | Pack Windows** command to have **GURPS Character Builder** make the final alignment.

Overlap Windows

When this command is checked, character sheet windows being opened the first time are overlapped on top of already open windows. Windows decides the exact position.

When this command is unchecked, **GURPS Character Builder** attempts to open new windows for the first time in a location that does not overlap open windows of the active character sheet. Spots immediately to the right or directly below currently open windows are examined first. If the window cannot be opened without exceeding the bounds of the application window, it is opened in an overlapping position.

Arrange Icons

Line up all iconized windows along the bottom of the screen.

Close All

Close all open windows.

Open All

Character sheet files only: Open all data windows for the character sheet.

Window Names

The rest of the entries on the Window menu bring the named window to the top.

Debugger

Open the Script Debugger (p. 253) window.

Help Menu

Contents

Presents the contents page for *GURPS Character Builder* help. This is a good place to start.

Help

Presents general help for the active window.

Game System Contents

Presents the contents page of the Game System (p. 298)-specific help.

Game System Info

Presents Game System (p. 298)-specific help for the active window. If there is no help for the game system, this command is inactive.

Game System Tutorial

Presents step-by-step instructions on building a character using the current game system.

Item Info

Searches the Game System help file for more information about the selected item in the data list. Some game systems may not have a help file, in which case you will not be able to use this command.

Using Dialogs

For game system-specific information, use the Game System Info command from the Help menu, or press SHIFT+F1. The help provided here is generic across all game systems. For a list of commands common to all windows, see Menus (p. 2).

Opening Other Windows

The Data menu shows all dialogs and lists associated with the main dialog of the character sheet. To open another list or dialog, just select its name from the Data menu.

Changing Values

When you change a value that other fields in the dialog depend, all changes will be automatically propagated. Character point totals, for example, will be automatically refigured. Your game system may one or more of the following ways to enter data:

- | | |
|------------|--|
| Edit Boxes | Click in the box and type your new information, or use Add 1 (p. 33) or Subtract 1 (p. 33) commands to increase or decrease numbers. When an edit box field defines a default value, the value is displayed in bold when set to the default (p. 34). When an explicit value has been entered for the field the value is displayed in the normal font.

Some edit boxes will display an error message if you enter an unacceptable value. If you click OK in the dialog (and Disallow if Not Satisfied is not checked in the Preferences), the error message will not be displayed again until you move to another edit box. |
| Combobox | Click the combobox with the mouse, and select the option you want. |
| Checkbox | Click the checkbox to toggle the value. |
| Pictures | Click the picture for picture (p. 50) commands. |
| List Box | Click in the list and press INS to insert an item, or select list-oriented commands from the Tools (p. 33) menu. You can also right-click the list box to bring up a menu of appropriate commands. |

The Tools menu (p. 33) contains commands to manipulate dialog fields.

Pictures

You can include pictures in your character sheet. They can be referenced in print templates (p. 109) and printed. Pictures are added by modifying the Dialog Item Properties (p. 155). To set the picture to be used, click inside the picture frame. A menu will pop up with the following commands:

- | | |
|---------------------------|---|
| Paste Picture | Copies a the current picture (bitmap) on the clipboard into the picture frame in the character sheet. If there is no bitmap image on the clipboard, this command will be grayed out. |
| Delete Picture | Deletes the picture. |
| Copy Picture | Copies the picture to the clipboard. (Not implemented yet) |
| Read File... | You may browse through directories for pictures that you want to include in your character sheet. A variety of picture formats are supported: bitmap (. BMP), JPEG (. JPG), and PNG (. PNG). JPEG and PNG format files usually result in the smallest files. Once you have read the picture into the character sheet it is independent of the original file. |
| Picture Library... | Brings up the Picture Search (p. 51) dialog. |
| Save Picture As... | Save the picture to a file on disk (p. 54). |

Display Adapter Note

Some display adapters do not display bitmaps properly because they don't have enough colors. For example, if you have a 16 million color bitmap image of your character and you're running on a computer with a 256-color display adapter, the image won't look right. This is to be expected. The image should still print fine (as long as your printer software can deal with the color image, of course!).

Picture Search

The Picture Search dialog appears when you select the Picture Library command from the Picture menu, or select the **Utilities | Picture Library...** command.

To search for pictures in the picture library to insert into your character sheet:

- Open the Picture dialog for the character sheet.
- Right click the picture area (or click the **Utilities** menu).
- Select the **Picture Library...** command.
- The Picture Search dialog will appear.
- Scroll through the **Keywords** list.
- Double-click a keyword of interest.
- A list of pictures that contains that keyword will be displayed in the Matching Pictures list.
- Click a picture with a file name or keywords that sound right.
- The picture will be displayed in the **Preview** window.
- Click the **Next** and **Previous** buttons to go through the list of matching pictures.
- Click **OK** when you have found a suitable picture.

The picture will be added to your character sheet.

To perform more complicated queries, enter query keywords and click the **Search** button. To add more pictures to the library (p. 53) or edit picture descriptions (to add more keywords), see the Picture Search Options (p. 53).

Query

More complicated picture searches can be performed by specifying multiple search keywords:

- Enter the keywords for your search in the **Query** field, separated by blanks.
- To see all pictures that include *any one* of the keywords you entered, click the **Or** button. To see only those pictures that include *all* the keywords, click **And**.
- You can also click the **Search** button when you have entered the keywords, or press ENTER.

For example, to search for pictures of men with a shield and axe:

- Click the **And** button
- Enter "man shield axe" in the **Query** field.
- Click **Search**.

Pictures with the highest number of keyword "hits" are listed first.

Synonyms are defined for many keywords. For example, "man," "men" and "male" are all treated the same. Descriptions will display the actual keyword defined for the picture file, not the synonym that matched. Additional synonyms can be defined in the picture files (p. 251).

Certain common words are ignored in searches, including "and," "or," "with," "in," "on," "a," "an," etc. The query "women from the old west" will be treated as "women old west." If your query isn't matching what you expect, remove any superfluous words and punctuation.

Additional ignored words can be defined in the picture files (p. 251).

Resizing the Dialog

The Picture Search dialog can be resized to enlarge the **Preview** window. To resize the dialog:

- Move the mouse cursor over one of the edges of the dialog.
- When the cursor changes to a double-headed arrow, click and drag.

Search

Click the **Search** button to perform a search for the keywords entered in the **Query** field.

If you leave the **Query** empty and click **Search**, pictures with no descriptions will be listed. This is useful when you have added new pictures in the Picture Search Options (p. 53) and you wish to find them to add keyword descriptions.

Or

If this is checked, picture descriptions containing any of the keywords in the **Query** will be listed in the **Matching Pictures** list.

And

If this is checked, only picture descriptions containing all the keywords in the **Query** will be listed in the **Matching Pictures** list.

Find All

If this is checked, all pictures in the library will be listed in the **Matching Pictures** list. The **Query** field will be disabled to indicate that no keywords are used for this search.

Keywords

The **Keywords** list displays all the terms that have been used to describe pictures in the library. Double-click a keyword to display all the pictures that contain that keyword.

Matching Pictures

This list contains all the pictures that match the most recently performed query. Click an entry to display the picture in the **Preview** window.

Preview

When a picture has been selected in the **Matching Pictures** list, it is displayed in this window. The name of the picture file is displayed directly below the picture and the keyword description is below that.

To edit the keyword description, click the **Options** button and check the **Allow Picture Description Changes** checkbox. When the Picture Search dialog is closed you will be asked if you wish to save the changed picture descriptions.

When editing picture descriptions, pressing the ENTER key will save the current picture's description and then proceed to the next picture matching the query. This allows you to quickly go through pictures and add descriptions.

Caution: if you change picture descriptions in the picture library installed with GURPS Character Builder, those changes will be lost if you reinstall the application or update to a new version. If you add your own pictures, you should place their descriptions in a new picture description file.

Previous

View the previous picture in the **Matching Pictures** list.

Next

View the next picture in the **Matching Pictures** list.

Browse...

Close this dialog and bring up the Browse Picture Library (p. 52) dialog, which allows you to preview pictures in the file system.

Options...

The Picture Search Options dialog (p. 53) allows you to change picture descriptions, set the picture library directory and search for picture files.

Browse Picture Library

The Browse Picture Library dialog appears when you click the **Browse...** button in the Picture Search dialog. When you click a picture in the file list, a preview of it will appear in the Preview Pane on the right.

To make your picture selection final and insert the picture into the character sheet, click **Open**. The standard Open File dialog controls in the dialog work the same as in the standard Open File dialog. You can also double-click the desired image file to select it.

Set Library Directory

If you install a library of pictures the Picture Library directory should be automatically set. To change the directory, click this button and select the directory where you wish to begin searching when you select the Picture Library command.

Picture Search Options

Allow Picture Description Changes

The picture description beneath the picture preview window in the Picture Search dialog is inactive by default. To activate the window to allow changes to the description to be made, check the **Allow Picture Description Changes** checkbox.

If any changes have been made when the Picture Search dialog is closed, you will be notified and asked if the changes should be saved. If you click **Yes** the description changes will be saved to the file where the pictures were defined (see Search for Pictures). If you click **No** the changes will be lost.

Directory

This displays the directory where the picture library is located. To change this, click the **Browse...** button and select a new directory. If any changes have been made to picture descriptions the application will ask if they should be saved.

When a new directory has been selected, all picture description files in it will be read, creating a new list of keywords in the Picture Search dialog.

Search for Pictures

To add pictures to the picture library:

- Copy the picture files to the picture search directory (p. 53). Make sure that they have unique names. You may place them in subdirectories of the Picture Library directory for organizational purposes.
- If you're not already in the Picture Search Options dialog, select the **Utilities | Picture Library...** command and click the **Options** button.
- To add the picture descriptions to an existing picture description file, click that file in the list of `.pdsc` files. **Note:** you should avoid changing description files installed with the application. If you reinstall or update the software, your changes will be lost.
- To add the picture descriptions to a new description file, click "(New Description File)".
- Click **Search**.
- If you're creating a new description file, enter the name in the Picture Description File Name dialog and click **Save**.

The application will search for picture files in the picture library directory and its subdirectories. Files with the following extensions will be found:

`.jpg`
`.jpeg`
`.bmp`
`.png`
`.gif`

Any files not already listed in a file will be added to the indicated description file. A dialog will report on the number of new files found.

To add descriptions to the new files:

- Check the **Allow Picture Description Changes** checkbox in the Picture Search Options dialog.
- Close the Picture Search Options dialog.
- Erase all text in the **Query** field of the Picture Search dialog.

- Click the **Search** button.
- Any pictures with no keyword descriptions will be listed in the **Matching Files** list, including those just found by Search for Pictures. The first picture without a description will be displayed in the **Preview**.
- Click the mouse in the empty keyword description area beneath the file name under the picture preview.
- Enter keywords for the picture and press ENTER.
- The next picture without keywords will be displayed.

When you're finished editing descriptions, click the **OK** button. Click **Yes** to confirm that you want the descriptions saved, or **No** to discard the changes.

To add new synonyms and ignored words:

- Open or create a picture description file in the picture library directory. The file should have an extension of `.pdsc`.
- To add synonyms, edit the `[Synonyms]` section according the format description in the Picture Description format (p. 251).
- To add ignored words, edit the `[Ignore]` section.

Save Picture As...

File Name

Enter the name of the file which you save the picture under. The extension (see below) indicates the format which the picture is saved under.

Save as type

The following formats are available:

JPEG	The JPEG format (Joint Photographic Expert Group) is the most compact image format. It is more suitable for “realistic” images, rather than line drawings or computer graphics, because it is “lossy.” GURPS Character Builder saves JPEGs at 75% quality. The extensions for this format are <code>.JPEG</code> and <code>.JPG</code>
PNG	The PNG format (Portable Network Graphics) is a lossless format that compresses well. It is intended to replace GIF images. It compresses better than GIF, but not as well as JPEG. It should be used for line-drawn and “blocky” computer graphics. The extension for this format is <code>.PNG</code> .
BMP	The BMP format is the standard Windows bitmap format. It is the least efficient in terms of space (these images can be 10 times larger than the equivalent JPEG or PNG images), but displays quickly. The extension for this format is <code>.BMP</code> .

Notes


GURPS Character Builder maintains the original format of the image files read. If the character sheets containing pictures are too large, you may wish to save the pictures with a more compact representation, then read that image back into the picture.

Item Lists

For game system-specific information, use the Game System Info command from the Help menu, or press SHIFT+F1. The help provided here is generic across all game systems.

From this window you can add, delete and modify one of the categories of items that your game system provides for. The Tools (p. 33) menu contains most of the commands that manipulate lists. For the commands common to all windows, see Menus (p. 2).

Adding Items

Press the Ins key, or select the **Select Item...** command from the Tools menu, or click the  button on the button bar.


Changing Items

Double-click the item, or highlight it and press the Enter key. You can also select the **Tools | Modify...** command.

Highlighting Items

To highlight an item, click it once. To highlight multiple items, click an item, then drag the selection up or down to include more items. To add an item to the highlighted items, hold down the CONTROL key and click the item. Hold down the SHIFT key to include all items between the highlighted items and the clicked items.

Deleting Items

Highlight the item(s) and press the Del key, or select the **Tools | Delete Item** command, or click the  button.

Moving Items

Select the items you want to move, then click one of the highlighted items and drag it to the new location. Then release the mouse button ("drop" the items). The items will be added to the list after the item they are dropped on. To move the items to the beginning of the list, drop them on the title bar of the list window. To move items to the end of the list, drop them on the Totals area at the bottom of the list window. Hold down the CTRL key to copy the items instead of moving them.

List Tools Context Menu

To bring up a menu of the list commands on the Tools (p. 33) menu, click the right mouse button in the list. The item in the list that you click will be the target of the command you select from the menu.




Getting More Help


Press Shift+F1 for game system-specific help.

Editing Other Types of Items

Select the desired category of items from the **Data** menu.

Sublists

A sublist is indicated by a  icon. To open the sublist double-click the  icon (or select the **Open/Close Sublist...** command). To close the open sublist  double-click it again. You can add a sublist with the **Tools | Insert Sublist...** command.





Members of the sublist are indicated by light gray lines drawn from the  down to the member items.

Excluded Items

Items that are excluded from the list total are "grayed out." Items that were automatically added by the selection of another item ("automatic items (p. 79)") are indented and preceded by a "+" sign.

Selecting Items

For information about specific items that may be available for this game system press F1 in the Available Items dialog.

To select an item, double-click on its name in the Available Items list. It will be added to the main list of items in the window. Names in the selection list that start with a  are sublists. Names in the selection list that start with a  are sublists that can be added to a character sheet with all their included items. To open them double-click the  or  icon, and close them by doing so again.

The Available Items dialog is *modeless*. You can click on a character sheet item list without dismissing the Available Items dialog, perform actions on the item list (move, delete or edit items), then click on the Available Items dialog (or press INS) to continue adding items. Clicking on the Available Items dialog brings to the top the last list you were adding to.

If Item Selection Rules (p. 57) are applied, or if you have selected a single category to display, some items may not be displayed in the Available Items list.

Many of the button commands are available on the context menu (p. 57). Click the right mouse button in the list area of the Available Items dialog to bring up the menu.

To See All Available Items

To view all available items alphabetically, outside of their sublists, set the Display Category to (All) and check the **Show Alphabetically** checkbox.

Display Category



By default, items from all categories are displayed. To display items from a single category:

- Click the Display Category dropdown listbox.
- Scroll to find the desired category.
- Click the category.

Only items from the selected category will be displayed in the list. The hierarchical sublist structure is suppressed when displaying a single category.

To display all categories of items again, follow the steps above, but select "(All)" in the dropdown listbox.

Select

For items that can be selected, adds the currently highlighted entry in the Available Items dialog to the current list. For sublists that can be added () , the sublist will be added to the list. A normal sublist () will be opened.

Open

Opens the highlighted sublist.

Info

Display information from the help file about the highlighted item. This information is usually specific to the use of the item with the application.

Items with no application-specific information may not have entries in the help file.

Find

Find an item in the list (p. 57) by searching for text in the item name. Only items not suppressed by item selection rules (p. 57) will be searched.

Next

Find the next item that matches the text entered in the Find dialog.

Req.

Show the unsatisfied requirements for the highlighted item. This button is inactive unless the highlighted item has requirements.

Check All

Check all unsatisfied requirements and present a list of those items with unsatisfied requirements (p. 34).

Show Alphabetically

If this checkbox is checked, then the hierarchical structure of the list is ignored and all item names are displayed in alphabetical order. If there are any duplicate items, the duplicates will not be displayed if they are identical. Items that are different but have the same name will still be displayed.

Check Req.

If the checkbox is checked, then requirements for items are checked when they are selected. If you have the Disallow If Not Satisfied checkbox is checked in the Preferences (p. 15) dialog, you will not be allowed to select items whose requirements are not satisfied.

Random

Select an item at random. If an open folder is currently highlighted, an item will be selected from within the sublist. Otherwise **GURPS Character Builder** chooses an item at the same “level” as the highlighted item. If a folder is chosen, items will be chosen randomly from it. This is useful for generating characters with “random” skills or powers. Only items not suppressed by item rules (p. 57) will be chosen. Items already present in the list will not be chosen again.

Rules

Click the **Rules** button to bring up the Selection Rules dialog (p. 59). Item selection rules control which items are displayed in the Available Items list. Rules can be defined to suppress the display of items based on their category, name, source data sheet and arbitrary expressions based on the data in the item. They can also be applied only to items that have the same names as other items. The rules are in effect only if the Apply Selection Rules checkbox is checked.

Apply Selection Rules

If this is checked, the item selection rules are applied. Any items that are suppressed by the rules are not displayed in the Available Items list. If there are no active rules, the **Apply Selection Rules** checkbox is grayed.

Game System-Specific Help

Press SHIFT+F1 for help specific to the game system you are working with. Some game systems may not be prepared for this, and may not have help.

Find

Search the Available items for text in the item name. The next item in the Available items list that contains the text you enter will be highlighted. If the item is in a sublist, **GURPS Character Builder** will open the sublists to present the item found. The search begins at the item following the currently highlighted item.

Available Items Context Menu

This context menu appears when you right click an item in the Available Items list.

Select Item

Add the item clicked to the list window.

Open Sublist

Open or close the sublist clicked. This command is active only when a sublist folder is clicked. It is checked if the sublist is already open.

Show All Items Alphabetically

Remove the folder hierarchy in the Available Items list and display all items alphabetically. This is useful for searching for an item directly by name, rather than browsing by category. This command is checked when the items are already displayed alphabetically.

Show Folder Hierarchy

Display the folder hierarchy in the Available Items list. This command is checked when the folder hierarchy is displayed.

Find...

Search for an item with certain text in its name. The search begins after the item clicked.

Find Next

Finds the next item (after the clicked item) that has the text specified in the Find dialog in its name.

Show Requirements...

Show the requirements for the selected item. This command is inactive if the item has no requirements.

Check All Requirements...

Check all unsatisfied requirements and present a list of those items with unsatisfied requirements (p. 34).

Item Info

Display help with any game-specific information about the highlighted item (note that this may not be implemented for every item in the list).

Select Random Item

Select an item at random (p. 57).

Item Selection Rules...

Bring up the selection rules dialog (p. 57).

Selection Rules

The Selection Rules dialog indicates which rules will be applied when the Available Items dialog is populated, and the **Apply Selection Rules** checkbox is checked. If an item meets the criteria for any of the rules, it will be omitted from the Available Items dialog. If a sublist has no items in it, it will also be omitted.

Rules at the top level apply to all categories. Rules in the category sublists apply only to the Available Items list when drawing from that category.

For example, a rule can be defined to suppress the display of items that have an option greater than a specific value: you could use this suppress items that have Technological Level greater than the default for a character sheet. Or you could define a rule that suppresses the display of items from a particular data sheet if there is another item in the same category with the same name, allowing you to choose which set of items is in effect.

Selection Rule Sets

You can define different sets of selection rules by creating *rule sets*. Settings for the different rules are shared among all character sheets using that rule set. This is useful if, for example, you run both a Space campaign and a Fantasy campaign, and you don't want to display Fantasy Equipment for characters in the Space campaign, or alien languages for your Fantasy campaign.

To select a rule set, click the **Rule Set** drop-down list and click the desired rule set. Whenever a rule set is selected and you apply or unapply a rule, that setting will be used for all character sheets that share that same rule set.

To add a new rule set, click the **Add...** button. To delete a rule set that is no longer needed, select the rule set and click the **Delete...** button. If any character sheets still use that rule set, it will reappear the next time it is opened for a character sheet that uses it. To remove it completely, you must remove all references to it in all character sheets.

Working with the Rule List

- To apply a rule, click the rule's checkbox. If the rule is checked, it will be applied.
- To open or close a category sublist, double-click the sublist entry.
- Click **OK** to make the changes permanent.
- Click **Cancel** to undo the changes in the rule activation and any rule deletions.

New

Create a new rule (p. 60) or category.

Change

Change (p. 60) the parameters for an existing rule. Double-clicking a rule also brings up the Edit Rule dialog.

Delete

Delete the highlighted rule. Clicking **OK** will make the deletion permanent. Clicking **Cancel** will undo the deletion.

Rule Set

The name of the rule set for the current character sheet (if a character sheet is being edited). "(None)" is the default rule set.

Add...

Add a new rule set (p. 62).

How Selection Rules Work

The selection rules are stored in `.rules` files (p. 252) in the *GURPS Character Builder* source directory. When the application starts, it reads all `.rules` files and remembers the rules for each game system.

When the Available Items list is populated, all rule-set active file-wide rules and rules specific to the category being displayed are evaluated for each item. If the item satisfies all rules, it is displayed, otherwise it is suppressed. Any sublists that have no items in them will also be suppressed.

New Rule

Create a new item selection rule.

Type

The rule type (p. 60) describes the behavior of the rule. The types are: Suppress Category (p. 60), Expression (p. 60), Suppress Item (p. 61), Suppress Item Prefix (p. 61), Suppress Data Sheet ID (p. 61), Suppress Sublist (p. 61), Requirement (p. 61) and Rules List (p. 61). If you select **Category**, a new category folder is created.

Category

The list of available categories. The new rule is placed in the category you select. See above for creating categories.

File

The name of the `.rules` file where the rule is stored. By default this is the game system name with the `.rules` extension. This file must be stored in the **GURPS Character Builder** source directory. Click the **Browse...** button to find files.

If you're writing custom rules that you want to give to other users, you should create a new `.rules` file for them, rather than adding them the standard rules file for that game system, so that you can avoid overwriting the standard rules file.

Browse...

Search the file system for a `.rules` file. The name of the file is placed in the **File** field.

Edit Rule

This dialog allows you to modify the parameters that define an item selection rule.

Category

The file and category for the rule.

Name

The name of the rule. This name is used to identify the rule internally, and must be unique within the category. The name must not be empty.

Value

The rule value is different for each type of rule. The value must be not be empty.

Suppress Category

The value is a list of semicolon-delimited category names. If the item belongs to any of the categories, the item will be suppressed. If the first character of the category name is "!" the rule matches if the item is not in that category.

You might want to use this rule to suppress a category of items that are not appropriate to a specific genre; for example, to prevent firearms from being displayed.

Expression

The value is an item-specific expression. If the expression evaluates to true (non-zero), the rule matches and the item is displayed. If the expression evaluates to false (zero), the item is not displayed.

For example, the following would suppress all items that have a Tech Level option greater than the current TechLevel:

```
`Tech Level`<=TechLevel
```

References to item information keywords used with `@foreach` (p. 194) (such as `@level`, `@cost`, `@dsid`, etc.) are also allowed. For example, to check for a blank datasheet ID use:

@dsid=""

This rule is the most flexible because you can check for a number of different attributes for an item. For example, to suppress the standard languages (items in the language category from the standard data sheet -- with the empty data sheet ID):

!inCategory('Language') or @dsid!=''

If an item is not a language, or is not in the standard data sheet, it satisfies the rule.

Suppress Item

The value is a list of semicolon-delimited item names. If the item's display name or the original name matches any of the names in the list, the item is suppressed. If the item name is preceded by "!" the item is suppressed if it is different.

This rule can be used to suppress items that aren't used for a particular campaign.

Suppress Item Prefix

The value is a list of semicolon-delimited item name prefixes. If the item's display name or the original name begins with any of the prefixes in the list, the item is suppressed. If the prefix is preceded by "!" the item is suppressed if the name or original name does not begin with the prefix.

This rule can be used to suppress the display of duplicate items. For example, let's say we have a custom data sheet that is constructed so that all items have a plug-in name of "Custom" that prefixes the original item names. We want the custom items to override the items in the standard data sheets. To specify this:

- Enter "Custom " for the **Value** (or whatever prefix you use).
- Check **NOT**.
- Check **Duplicates Only**.

Suppress Data Sheet ID

The value is a list of semicolon-delimited data sheet IDs. These are one to four characters in length. Comparisons are case-sensitive. Data sheet IDs are another method for specifying the data sheet of origin for an item, and can be used in a similar fashion to item name prefixes.

If the ID is preceded by "!" items with IDs that do not match are suppressed.

Suppress Sublist

The value is a semicolon-delimited list of sublist names. Any item appearing in one of these sublists, or in any child sublist of these sublists, will be suppressed.

Requirement

The rule is satisfied if the item's requirements are satisfied. The value is the name of the list which the item would be added to.

Rules List

The Rules List rule is special. The value for a Rules List rule is a list of other rule names, separated by semicolons. These rule names are used only when the Rules List rule is checked or unchecked in the list of rules. An Rules List rule is always satisfied.

Rules in a category are qualified by the category name and a colon. When the Rules List rule is checked, all the rules listed in the value are also checked. Thus, you can apply several rules at once by checking a Rules List rule. After a Rules List rule is checked, the rules it (un)checks can be changed manually to modify their state. They are not forced to obey the Rules List rule.

If the name of a rule is preceded by a "!" that rule is unchecked if the Rules List rule is checked, and checked if the base rule is unchecked.

Rules List rules cannot modify other Rules List rules.

Description

The description of the rule. This description is displayed to the user in the Selection Rules dialog. It must not be empty.

NOT

If checked, the sense of the rule is negated. For example, if a Suppress Category rule is negated, then only items in those categories are displayed (rather than suppressing items in those categories).

Duplicates Only

If checked, the rule applies only for items that have the same name as another item in the same sublist. This is useful for suppressing items that are present in two different data sheets. Items that do not have the same name as another item in the same list always satisfy the rule and are not suppressed.

Active by Default

If checked, the initial state of the rule is active (checked). If not checked, the rule will be inactive by default. This is useful for creating rule files that are given to other users to allow installation of a package of a data sheet and rule set with default settings for duplicate items.

A different default value is stored for each rule set. This allows you to define rules that have different initial settings based on the character sheet (which can have a rule set setting). If a rule set doesn't have a default value set for it explicitly, it uses the default setting for the (None) rule set.

Rule Set

The rule set that is in effect. Changing the default value affects the default for the indicated rule set only.

Select Rule File

Use the file dialog to select a rule file name. The rule file must be kept in the **GURPS Character Builder** source directory, so only the name part of the file path you select will be used.

New Category Name

Enter the name of the item selection rule category. This will create an entry in the Selection Rule list, but if you don't actually add a rule to that category, it will not be permanent.

If there are no data sheets that define items in the category, the rule can never be applied. To avoid having to create the categories manually, you should edit the selection rules after you have loaded the data sheets that you wish to apply the rules to.

Add Rule Set

Enter the name of the selection rule set to add. The new rule set will automatically be selected for the current character sheet (if there is one).

Rule set names may not begin with the "~" and "(" characters, and may not include the ";", "\", and "=" characters. You may not add a rule set name that already is in the list.




















Button Bar

The Button Bar is a window just below the menu bar. Clicking a button on the button bar executes the corresponding command in the menu.

To dismiss or display the button bar right-click the button bar, status bar or application background. A context menu will appear with check marks by the windows that are open: button bar, status bar and shortcuts window. Click the desired command to change the state of the desired window.



Button	Action
--------	--------

	File New (p. 2)
	File Open (p. 3)
	File Recent Files (p. 8)
	File Save (p. 3)
	File Print Through Template (p. 6)
	File Print Preview Through Template (p. 7)
	Edit Cut (p. 9)
	Edit Copy (p. 9)
	Edit Paste (p. 9)
	Edit Clear (p. 9)
	Tools Select Item... (p. 33)
	Tools Move Up (p. 33)
	Tools Move Down (p. 33)
	Tools Add 1 (p. 33)
	Tools Subtract 1 (p. 33)
	Tools Check Requirements... (p. 33)
	Utilities Die Roller... (p. 24)
	Modify Dialog Item Properties (p. 155)
	Help Help: brings up help on the current topic.

Copyright Information

This application and its associated files are copyright © 1994-2002 by Alter Ego Software, Inc. All Rights Reserved.

The copyrights of the game systems supported by this application belong to the individuals and corporations to whom they are credited. Any redistribution of those materials without permission of the copyright holder is illegal.

If you, the user of this application, wish to sell or even give away anything that contains copyrighted material, you must first obtain the permission of those copyright holders.

The following is not “real” legal advice. It does not pretend to be real advice. I’m putting this here so that if you do violate someone’s copyright I can say I told you not to do it. If you’re really concerned about violating someone’s copyright, talk to your lawyer.

You can...

- ... do anything you like for **your own personal use**.
- ... create a data sheet, print template, character template, etc., containing information from the “ABC” supplement for the (fictitious) game system XYZ **for your own personal use**.
- ... give away character sheets you create for your own characters using game system XYZ (they’re original).
- ... create new filters for the game system XYZ and give them away (ditto).
- ... create new print templates that are based on **original** character sheet layouts for game system XYZ and give them away (ditto again).
- ... create new data sheets using **non-copyrighted** materials based on game system XYZ and give them away. This includes new, **original** lists of equipment, advantages, skills, etc.
- ... create data sheets, character templates, print templates, etc., for your own game system and give it away. Or sell it. *You own the copyright!*

In general, you can give away anything **original** that you create, but you can’t give away anything that is copyrighted *without permission from the holder of that copyright*.

You **can not**...

- ... give away or sell that “ABC” data sheet for game system XYZ unless you first obtain permission from the holder of the copyright for XYZ.
- ... redistribute any copyrighted material without permission of the copyright holder.

If some of this seems redundant to you, you’re correct.

Troubleshooting Printing Problems

If you have problems printing, check here for possible solutions.

The top (or bottom) of the character sheet is chopped off

The print template is too large or misaligned for your printer.

- Find out which print template is causing the problem (choose the **File | Set Copy/Print Filters...** command).
- Make sure that the printer you're having the problem with is the default printer (by selecting it in the **Start | Settings | Printers** command under Windows 95)
- Edit the print template you found above by selecting the **File | Open...** command. Go to the **GURPS Character Builder** source directory (p. 15) (usually `c:\Creator`). List files of Print Template by clicking on the drop down list for listing file types. Choose the desired print template from the list of files.
- In the opened print template, a light gray rectangle shows the extent of the printable area of the page.
- Press CTRL+A to select all graphic objects in the print template.
- Click and drag the objects in the print template so that all of the objects appear within the bounds of printable area.
- If you can't move the template so that all objects fit within the printable area, you may need to change the print template so that it will fit—this may require making some objects smaller or rearranging the print template.
- Save the print template and try printing again. You should save the print template under a new name so that when you change printers or update to a new version of **GURPS Character Builder** you have both your customized and the original versions of the print template.

Rich Edit DLL

The Rich Edit 2.0 DLL is required for using the formatted text capabilities in ***GURPS Character Builder***. This DLL is normally provided with your operating system. It comes with Windows 98, Windows NT 4.0, Windows ME and Windows 2000. It is not provided with Windows 95, but many applications install it.

Under Windows 95, ***GURPS Character Builder*** will attempt to use the Rich Edit 1.0 DLL (RichEd32.dll) if it is present. RichEd 1.0 lacks some of the paragraph formatting capabilities that Rich Edit 2.0 provides.

You can obtain Rich Edit 2.0 from Microsoft. It is provided with many applications, but it is dependent on language. You should make sure that you obtain the correct version for your operating system.

Shortcuts

The Shortcuts window provides a flat, always-on, context-sensitive user interface to **GURPS Character Builder** commands and features.

To select a command click on it and release the mouse button.

The commands that appear on the Shortcuts window can be customized (p. 242).





Closing and Opening the Shortcuts Window

To close the shortcuts window click the "x" close box in the upper right corner.

To open the shortcuts window select the **Utilities | Show Shortcuts** command.

You can also right click the application background, status bar or button bar to bring up a context menu. Checked commands indicate open windows. Click the Shortcuts command to open/close the shortcuts window.

Open and Close Shortcut Folders

Shortcut folders appear with  and  icons. To close a shortcut folder and hide the shortcuts inside it, click the . To open a shortcut folder and display the shortcuts contained within, click the .

GURPS Character Builder will remember which folders are open each time you restart the application.

Widen the Shortcut Window

Move the mouse cursor over the right-hand border of the shortcut window. When it changes to a two-headed arrow, click and drag to the right.

Editing Items

For information about specific items that may be available for this game system press SHIFT+F1.

In the Edit Item dialog you can change information about items.

Name	The text you have for the name will be displayed everywhere the name for the item is used. In particular, it is displayed in the item lists. It is also used for looking up requirements and summaries, so if you change the name of an item you select from a list, other items that may reference it (for example, to check requirements) will not be able to find it.
Level	<p>This is either a numerical value representing the level of the item, or a selection from a drop-down list of items. In the former case, simply enter the value you want (or click the up and down arrows to increase or decrease the currently displayed value by 1. If you hold down the SHIFT key, the value will be changed by 5). When you click the OK or Try It buttons, the values in the character sheet totals and any other items that depend on this item will be updated (unless, of course, you click the Exclude from Total checkbox). If you enter a value in the Points field and click Try It, <i>GURPS Character Builder</i> will attempt to find a Level that corresponds to that value. If one cannot be found, you will be notified.</p> <p>Sometimes the Level field will be inactive: when you have an item that has no level (i.e., it only has a point cost), or when you have clicked the Default checkbox.</p>
Try It	After you enter a value in either the Level or Points fields, you can click Try It to see how the change will affect the totals in the rest of the character sheet. The other fields in the dialog box will be updated according to your changes. Note: this button will not be present for items that have a drop-down list to select their values.
Default	If the item has a default value, you can click this checkbox to use the default value. Frequently default values have zero cost, so the Points field will often be zero and the Level field will be the "default" value. When you have selected the default value you cannot change the Level and Points fields.
Points	<p>This value reflects the point cost of the item. If you have changed the Level, you must click Try It or OK for the value in Points to be updated. You can also change Points to be the cost you desire. Click OK or Try It to have <i>GURPS Character Builder</i> automatically update the Level.</p> <p>The number following the points value reflects the total cost of the item, including the level and all options you have selected.</p>
OK	Click this button to save the changes to the item.
Cancel	Click this button to throw away all changes that have been made to the item.
Exclude	If you check this checkbox, the cost of this item will not be included in the list. However, any external variables or conditions changed by options for this item will still be in effect. The item will be "grayed out" in the list to indicate that it is not included in the total.
Options	Opens the options part of the item dialog if there are no options to begin with.
Notes	Opens a window in which you can enter notes (p. 73) about the item.

The item dialog can be resized by clicking and dragging when the cursor changes to a two-headed arrow over the dialog window border.

The first time an error condition occurs during item editing the message will be displayed. If you click the **OK** button the message will not be displayed again until you edit another item, or click the **Try It...** button.

See also...
Item Options (p. 70)

Editing Items

This dialog appears when there are options only and no editable level or cost. For information about specific items that may be available for this game system press SHIFT+F1.

In the Edit Item dialog you can change information about items.

Name	The text you have for the name will be displayed everywhere the name for the item is used. In particular, it is displayed in the item lists. It is also used for looking up requirements and summaries, so if you change the name of an item you select from a list, other items that may reference it (for example, to check requirements) will not be able to find it.
Points	This value reflects the point cost of the item.
OK	Click this button to save the changes to the item.
Cancel	Click this button to throw away all changes that have been made to the item.
Exclude	If you check this checkbox, the cost of this item will not be included in the list. However, any external variables or conditions changed by options for this item will still be in effect. The item will be "grayed out" in the list to indicate that it is not included in the total.
Options	Opens the options (p. 70) part of the item dialog if there are no options to begin with.
Notes	Opens a window in which you can enter notes (p. 73) about the item.

The item dialog can be resized by clicking and dragging when the cursor changes to a two-headed arrow over the dialog window border.

See also...
Item Options (p. 70)

Editing Items

This dialog appears when there is only an editable level and no editable cost. For information about specific items that may be available for this game system press SHIFT+F1.

In the Edit Item dialog you can change information about items.

Name	The text you have for the name will be displayed everywhere the name for the item is used. In particular, it is displayed in the item lists. It is also used for looking up requirements and summaries, so if you change the name of an item you select from a list, other items that may reference it (for example, to check requirements) will not be able to find it.
Level	This is either a numerical value representing the level of the item, or a selection from a drop-down list of items. In the former case, simply enter the value you want (or click the up and down arrows to increase or decrease the currently displayed value by 1. If you hold down the SHIFT key, the value will be changed by 5). When you click the OK or Try It buttons, any other items that depend on this item will be updated.
Try It	After you enter a value in the Level field, you can click Try It to see how the change will affect the rest of the character sheet.
Default	If the item has a default value, you can click this checkbox to use the default value.
OK	Click this button to save the changes to the item.
Cancel	Click this button to throw away all changes that have been made to the item.
Exclude	If you check this checkbox, the cost of this item will not be included in the list. However, any external variables or conditions changed by options for this item will still be in effect. The item will be "grayed out" in the list to indicate that it is not included in the total.
Options	Opens the options part of the item dialog if there are no options to begin with.

Notes Opens a window in which you can enter notes (p. 73) about the item.

The item dialog can be resized by clicking and dragging when the cursor changes to a two-headed arrow over the dialog window border.

The first time an error condition occurs during item editing the message will be displayed. If you click the **OK** button the message will not be displayed again until you edit another item, or click the **Try It...** button.

See also...
Item Options (p. 70)

Editing Items

This dialog appears when there is an item cost only, and no editable level. For information about specific items that may be available for this game system press SHIFT+F1.

In the Edit Item dialog you can change information about items.

- | | |
|----------------|--|
| Name | The text you have for the name will be displayed everywhere the name for the item is used. In particular, it is displayed in the item lists. It is also used for looking up requirements and summaries, so if you change the name of an item you select from a list, other items that may reference it (for example, to check requirements) will not be able to find it. |
| Try It | After you enter a value in the Points fields, you can click Try It to see how the change will affect the totals in the rest of the character sheet. The other fields in the dialog box will be updated according to your changes. |
| Points | <p>This value reflects the point cost of the item. You can change Points to be the cost you desire.</p> <p>The number following the points value reflects the total cost of the item, including all options you have selected.</p> |
| OK | Click this button to save the changes to the item. |
| Cancel | Click this button to throw away all changes that have been made to the item. |
| Exclude | If you check this checkbox, the cost of this item will not be included in the list. However, any external variables or conditions changed by options for this item will still be in effect. The item will be "grayed out" in the list to indicate that it is not included in the total. |
| Options | Opens the options part of the item dialog if there are no options to begin with. |
| Notes | Opens a window in which you can enter notes (p. 73) about the item. |

The item dialog can be resized by clicking and dragging when the cursor changes to a two-headed arrow over the dialog window border.

The first time an error condition occurs during item editing the message will be displayed. If you click the **OK** button the message will not be displayed again until you edit another item, or click the **Try It...** button.

See also...
Item Options (p. 70)

Item Options

Item Options allow you to add more information to item. You can use the options to...


- Add short explanatory notes. For example, to state the casting time for a spell.
- Add additional costs. For example, if you have a skill that has an extra ability that incurs additional costs, you can use an option to obtain that.
- Set other conditions that are reflected elsewhere in the character sheet. For example, if the item is a suit of armor, you can set defensive adjustments in the options, or the monetary cost of the item, or its weight.

The exact options that will be available depend on the game system that your character sheet uses. Not all game systems will use options.

Most options will simply have a single value. Two values may be displayed for computed value options and option value list. For a computed value option, the level is displayed first, following by the actual cost. For an option value list the label is displayed, followed by the value for that label.


The item dialog can be resized by clicking and dragging when the cursor changes to a two-headed arrow over the dialog window border. The option list will grow, allowing more items to be displayed at once without scrolling.

Option Editing Buttons

New	Brings up a list of available options to choose from. The option is added after the currently highlighted option, or at the end of the list if no option is highlighted.
Edit	Allows you to change the value (p. 96) associated with the highlighted option (this is not active if more than one option is highlighted). You can also double-click the option for the same effect. To edit the specific details of the option (the expression, display formats, etc.), hold down the SHIFT key when you click the Edit button.
Delete	Delete the highlighted options.
Cut	Cut the highlighted options and put them on the clipboard. To select all items and cut them, hold down the SHIFT key and click the Cut button.
Copy	Copy the highlighted options to the clipboard. To select all items and copy them, hold down the SHIFT key and click the Copy button.
Paste	Paste an item from the clipboard into the option list after the highlighted option, or at the end of the list if no option is highlighted.
	<p>Increments or decrements the highlighted option. If you click the up arrow, 1 is added to the option's value. If you click the down arrow, 1 is subtracted from the value. If you hold down the SHIFT key, 5 is added to or subtracted from the value.</p> <p>If the option value is a list of labels and values, each time you click the up/down button the previous or subsequent value will be selected.</p> <p>If the option value is computed, each time you click the up/down button for the option the check expression will be evaluated.</p>
<input type="checkbox"/> and <input checked="" type="checkbox"/>	The check boxes indicate the selected state of the option. If the box is checked, the option is selected. If the option is not checked, it will not be printed on the character sheet and will not be included in any computations of the item's cost.
...	Selects the name of an item.

Selecting Multiple Options

You can select multiple options by...

- Clicking on an option and dragging to the last option you wish to select.
- Hold down the CTRL key and click on another option (do not click on the ). Each option clicked will be highlighted. If you click an option that's already highlighted, it will become unhighlighted.
- Highlight an option. Then hold down the SHIFT key and click another option. All options between the highlighted option and the option you click will be highlighted.

Option Context Menu


Right click an option to bring up the option context menu. The commands on the menu are the same as the option editing buttons, with the following additions:

Open File...	Present only for a file option. This dismisses the current item dialog and opens the file specified for the option. If the file is already opened, the main window for the character sheet is brought to the top.
---------------------	---

Select Option

From this dialog you may select options for the item you are currently editing. Options can modify the cost of the item, provide explanatory notes, and make adjustments to other values in the character sheet.

To Open or Close a Category of Options

Double click the  for the category, or highlight it and click **Select**.

To Select an Option

Double click the option, or highlight it and click **Select**. The option will be added to the item list and you can change it.

To Resize the Select Option Dialog

Click on an edge of the window and drag it.

Select Category

Click the category from which you wish to choose items, then click **Select**.

You may place items from any category in any list, though it may not always make sense. For example, if you add some equipment to your list of skills, it may add the weight of the equipment to your skill cost.

If the category you're looking for is not present in the list, use the **File | Load Data Sheet...** (p. 4) command to load the necessary data sheets.

Edit Sublist Information

Sublists are used to group items together under a common title. This can be used to effect such things as backpacks, banks, storage places at home, etc.

Name

The name of the sublist.

Exclude

If this is checked, the sublist's total cost is not included in the cost for the parent list.

Cost

The "cost" of the sublist. This is the sum of the costs of all the items in the sublist.

Total

The total cost of the sublist modified by the Options.

Children Inherit Options

If this checkbox is checked, the options for the sublist are applied against each child item in the sublist. If it is unchecked, the options apply to the total of the child items. The two values computed will differ depending on how the game system defines roundoff.

This is useful for game systems that apply the same modifier to many items that have something in common -- a cost break, for example. You can add the option to the sublist instead of every item, allowing you to easily change the options on many items at once.

Open List

If this box is checked, the sublist is open. You can open and close the sublist by checking and unchecking here. This is useful if you've just added a sublist and you wish to immediately add items to it.

Options

The options (p. 70) can be used to modify the cost of the sublist to produce the Total Cost.

Notes...

Opens a window in which you can enter notes (p. 73) about the item.

Hints

In certain game systems it is useful to place items in a sublist called "Backpack" and then exclude the cost of the item when the "cost" of items in the list are their weights. This allows the values in the

character sheet due to encumbrance to be computed without the contents of the backpack, which would normally be dropped in combat.

Item Notes

Enter any notes about the item. Up to 5000 characters can be entered.

If you click **Cancel** or press the ESC key, the window will be dismissed, all changes you have made will be discarded and the original text will be restored.

Special Keys (p. 73)

Edit Item Name

Change the name of the item. Up to 5000 characters can be entered.

If you click **Cancel** or press the ESC key, the window will be dismissed, all changes you have made will be discarded and the original text will be restored.

Special Keys (p. 73)

Special Editing Keys for Item Notes and Names

Key	Function
CTRL+TAB	Inserts a tab into the text.
CTRL+ENTER	Inserts a paragraph into the text.
ALT+BACKSPACE	Undo the last editing change.
ESC	Close the window and abandon all changes

Choose Items

GURPS Character Builder is adding auxiliary items for the item you selected. You must choose which auxiliary items to add.

The prompt indicates how many items to select:

Choose N items.

Choose from N to M items.

Choose up to N items.

Choose at least N items.

Choose any number of items, or none.

The **OK** button will be grayed out and inactive until you have selected the correct number of items. In the last case, you may pick no items if you don't want any of them.

Select an item by clicking it. To deselect a selected item, click it again.

Get Item Reference

One of the items that is being added automatically for the item you selected references another item. You must choose which item is referenced.

Click the name of the item that you wish the item to reference. Click **OK** when you are finished.

The **OK** button will be grayed out and disabled until you make a selection.

Choose Value

You are selecting an alternative for the item you chose. This choice will direct **GURPS Character Builder** in the addition of other items.

Click (or double-click) the entry that you wish to select. Then click **OK**. Click **Cancel** to cancel everything. The item you chose and other automatically added items in the main list will be removed.

Character Sheet File Name

The value of this option is a character sheet file name. Click a character sheet to choose the file that will be referenced by this option, then click **Open**. You may also double-click the desired file to select it.

Hints

- You should save the referencing character sheet before making a reference to another character sheet. This provides the character sheet with a known location.
- It is best to place character sheets that reference one another in the same directory (see below for details).
- If character sheet A references character sheet B, and you have both open in **GURPS Character Builder** at the same time, changes in B will be reflected in A only when you save B.
- If a file does not exist, all the values referenced in it will be zero.
- You may specify a non-existent file by entering the name directly in the **File name** field. This way you can reference a file, then create and save it, which will update the referencing character sheet.
- If you "lose" a reference, you should reselect the referenced character sheet, preferably moving the files into the same directory to maintain referential integrity.
- If you change a referenced character sheet outside of **GURPS Character Builder**, you must close and reopen all character sheets that reference it in order to get the changes.

Details

If the referencing character sheet has already been saved and the referenced character sheet is in the same directory, only the base name of the referenced file will be saved in the option. This allows you to move the character sheets together without losing the reference. If the referenced character sheet is another directory, the entire path name to the file is saved.

If the referencing character sheet has not yet been saved, it has no known directory. The entire path name will be saved.

Edit Properties

The different groups of item properties are edited within this dialog. To select a group, click its radio button.

Property Groups

- Basic (p. 75)
- Flags (p. 76)
- Sublist Flags (p. 78)
- Expressions (p. 105)
- Automation (p. 107)
- Requirements (p. 88)
- Adjustments (p. 87)
- Options (p. 92)
- Categories (p. 78)
- Automatic Items (p. 79)

Basic Properties

Name

The Name field is the name of the item. This name is displayed in the lists and can be edited in the Item Editing dialog.

Value

The value associated with the item. For example, the level of a skill. Some types of items will not have a value and may only have a cost. If an item has a cost that is computed based on the value, then a cost expression (p. 105) should also be specified to indicate the relationship between the value and the cost.

Cost

The cost field is assigned as the item's cost if there is no cost expression. A cost will automatically be computed if the cost expression (p. 105) is filled in, based on the value chosen. If an item has a fixed cost, then the cost expression should be left empty and the cost value should be assigned.

Class

The class field can be accessed through formatting, and be used to give more information about an item to be selected (i.e., whether it is hard or easy, etc.).

Alias

An alternate name for the item, to be used when the UseAliases game system configuration (p. 248) parameter is set. When an item is converted without a script, the alias on the original item is retained.

Var. Name

The variable name field must be set if you want the value of an item to track another value. For example, if you have a skill that is dependent on the intelligence attribute, you must assign the skill a variable name so that it will automatically be updated when intelligence changes.

Original Name

This value is set to the original name of the item as it was selected from the data sheet item list. This is used when you update a character sheet to a new version of the template. If the item name cannot be found in the data sheet item list during conversion, the Original Name is searched for. If found, the item with that original name is used. Therefore, if you have the wrong original name, the wrong item will be used to determine the item type at conversion time. You can prevent an incorrect conversion by setting this to nothing. The Generic Item (p. 76) flag will prevent any conversion at all from being done on an item.

The original name can also be used to identify items that receive adjustments.

Data Sheet ID

The data sheet ID is a four-character identifier used to indicate which data sheet an item was loaded from. The ID may be between zero and four characters in length. Items from the "main" data sheet for a game system have no data sheet id by convention.

The data sheet ID is used during character sheet updates to ensure that the same item is used as a pattern for the update. It is also used to determine which item should be used in requirements and automatic items, should multiple items with the same name be loaded.

Random Pips

This value specifies the weight of an item during random selection. This is the number of "pips" associated with it. A value of 0 is the default, and is treated the same as 1. A value of -1 removes the item from consideration. If you wish to preclude an entire sublist of items, you should have set random pips on the sublist to -1.

When items are selected at random, the random pips are totaled for all items at the current level. A random number is generated in that range, and the item corresponding to that number is chosen. If that item is a sublist, then the process is repeated for its members until an item is found. If a sublist is chosen that has no eligible members, no item can be selected.

For example, if there are 10 items in a sublist, and you wish the first two to be selected 50% of the time, you would assign a value of 4 for those two items and 1 for the other eight.

Auto ID

Whenever an item is added that adds automatic items, it is assigned an auto ID. This is used to determine which items should be deleted when the Delete Auto-added Items (p. 77) checkbox is set. All items with Auto Parent equal to the Auto ID of the item deleted will also be deleted.

To break the link between the parent item and all child automatic items (to prevent their deletion), set this value to 0.

Parent

The auto ID of the parent item that added this item. This value is 0 if item has no parent. When the parent item is deleted and it has the Delete Auto-added Items flag set, all items with Parent IDs that match the Auto ID of the parent will be deleted as well.

To break the link between a single item and its parent, set the Parent value to 0.

Priority

This value ranges from 0 to 15. It is used to "weed out" duplicate items, allowing you to specify which items have priority over others. When a data sheet is loaded each item's name is compared to the other item names in the same sublist. If two items have the same name, the item with the higher priority is retained and the item with the lower priority is discarded. If two items have the same priority, both are retained (unless an item with higher priority is loaded).

See Also...

- Formats (p. 102)
- Lookup Array (p. 104)
- Dialog (p. 104)
- Flags (p. 76)

Edit Properties: Flags

Automatic Item

Checked if the item was automatically added (p. 79) by the addition of another item.

Exclude from Total

Checked if the item's cost is excluded from the list's total.

Generic Item

Checked if the item is "generic." Items that you build from scratch in a character sheet can be marked as generic, which will prevent them from being converted to the new form of the original item when the

character sheet is converted to the new template. This allows you to create items that will not be changed when you convert the character sheet.

If you want everything but the cost converted, use the Keep Cost flag (see below).

Use Default

Checked if the "default value" for the item should be used.

Baseable on Default

If this is checked, the **Use Default** flag is not set by the **Default** checkbox in the item edit dialog. Instead, the **Based on Default** flag is reflects the value of the checkbox.

Based on Default

If this is checked, the interpretation of the **Default Value** and **Default Formula** values are interpreted differently.

Add As List

Applies only to items in data sheets. When this item is added, it is added as a sublist, and any automatic items will be added within the sublist.

Keep Cost

The current cost will be kept when you convert this character sheet. This allows you to change the cost of the item, but get all the features of the new item.

Create Unique Variable

If the variable specified by the variable name already exists, a new variable name is generated and saved permanently. This is useful for declaring items that other items reference with item reference options.

Don't Duplicate Automatic Items

When adding automatic items, this checkbox indicates that a second item should not be added if an instance of the automatic item is already present. The existing item will be "adopted" by the parent item, setting the newly-adopted child's auto Parent field to the auto ID (p. 76) of this item.

Delete Auto-added Items

If this checkbox is checked, any child items (whose Parent field is equal to the auto ID field of this item) will also be deleted. This is useful for items that always require a secondary item to be added. For example, some game systems require a Skill for each Power. When the Power is added, the Skill must also be added, but if the Power is deleted, the Skill is useless without it, so it should be deleted too.

Disallow Check Exp. Violation

If this is checked for an item, **GURPS Character Builder** will not allow you to set values on that item that violate the check expression.

Duplicates Expected

If this is checked on an item that is being added, no check is made in the target list for an item by the same name. This is useful for items that differ by the options selected, rather than by the actual item name.

Reference Item

If this is checked, the item is a *reference item*. Reference items basically point to other items. When they are selected in the Available items dialog, the item named in the Requirement is added, instead of the reference item. If no Requirement is specified, the sublist in the Category is opened and selected. Only one item or category should be named in the Requirement or Category.

Check Req. only when adding

When checked, the requirements for this item are only checked when the item is being added for the first time (from the Available Items list). The requirements are not checked when the item is pasted or undeleted. You might use this when an item automatically adds items to another list which is limited to a single item.

Don't Set Adjustments and Automatic Items

When checked, adjustments on the item or any child items are not set, and any automatic items are not added. That is, if the item adds bonuses to other items, those bonuses will not be added. Note that if

other items reference the variable defined for the item, that variable reference will still work as normal. Additionally, the automatic item list is not evaluated and no automatic items are added.

This is useful for adding items to sublists that represent conditional selection of items, or character sheets acting as data sheets.

Don't Check Requirements

When checked on an item or an item's parent, the requirements are not checked when the item is added to the character sheet.

This is useful for adding items to sublists that represent conditional selection of items, or character sheets acting as data sheets.

This flag is retained when an item is converted by the default item conversion routine.

Disallow Satisfying Requirements

When checked on an item, the user cannot use the Requirements dialog to satisfy requirements automatically -- the user must manually select the required items or set the required values. This is useful for game systems in which the addition of items and setting of values are moderated by scripts. For example, some game systems might require a character be of a particular race, and the race is indicated by the presence of a race item in a single-entry listbox. Trying to satisfy the requirement through the Requirements dialog can't work if a race is already present in the listbox. This turns off the **Satisfy** and **Satisfy All** buttons in the Requirements dialog.

Edit Properties: Sublist Flags

Charge for Automatic Items

When checked, automatic items associated with the item will be included in the total for the list when added. By default, automatic items are excluded from the list's total.

Don't Mark Automatic Items

Any automatic items added by this item will not be marked as automatic -- they will be treated as normal items.

Add Sublist and Contents

Allows a sublist item to be selected from the Available Items list. The sublist and any items in it will be added to the list. This is useful for adding "package deals" to data sheets.

Children Inherit Options

Child items of the sublist inherit the sublist's options. An inherited option is displayed in the child item's list of options, and is included in any adjustments to the item's cost.

List Has Own Cost

The cost of the list can be set explicitly. This is useful for setting the cost of a list of items, regardless of the actual total cost of the items in the list. This is useful for fixed-cost "package deals." This can be used in conjunction with the next flag to get different effects.

Child Costs Are Separate

The costs of items in the sublist are added to the list total, in addition to any cost that the list itself may have. A useful combination is to set List Has Own Cost and Child Costs Are Separate, which allows the sublist to have a set cost, but still have the child item costs be added to the total list cost.

Edit Properties: Categories

The list of categories associated with an item can be displayed in the list window. It can also be used to determine bonuses for a set of items, as well as to specify certain kinds of requirements (p. 88).

Categories that begin with a "*" are not listed in the List Summary (p. 36), but can still be used for fulfilling category count requirements.

See Also...

List Editing Operations (p. 88)

Edit Properties: Automatic Items

Automatic items are automatically added to the item list when the item that includes them is added. By default they are marked as automatic in the item list (p. 55) by a "+" sign. They are excluded from the list total by default, though you can change that in the item editing dialog (p. 68), and with the `?charge` (p. 81) and `?nocharge` (p. 81) directives.

You can specify the data sheet list where the item can be found by prefixing the item name with the data sheet category and a colon (include no spaces). The item will then be included in the current list, and all adjustments and options for the item will be in effect. If you omit the data sheet list name, or if the item cannot be found in any of the loaded data sheets and no `?defitem` (p. 82) is specified, an item will be added that simply has the specified text and zero cost.

Requirements are not checked for automatic items.

Setting the Level

To set the level of an automatically added item, follow the name with "=" and the desired level:

```
Skills:Lockpicking=10
```

Renaming Automatic Items

Include the new name after the name as it appears in the data sheet, enclosed in brackets:

```
Combat Sense[Combat Training]
```

If both are included, the level goes after the new name:

```
Talents:Combat Sense[Combat Training]=9
```

Adding Options

To add a named option to subsequent automatic items in the list, include an `?opt:` entry:

```
?opt:Option Name
```

where "Option Name" is the name of the option. The named option will be added to every automatic item.

If a value is specified after the option will be set to that value. The expression will be evaluated, so if you want a string to be assigned you must use quotes around it. For example,

```
?opt:Base Level='DX+1'
```

If the option is already present in the list of options to add, the new value will override the current value (see below to override this behavior).

If the automatic item that the option is being set on was already in the character sheet, the option value will not be changed if the new value is less than the existing value.

Duplicate Option

The `?dupopt` is the same as `?opt`, except that the option will be added with the new value, even if another option with the same name already exists. You would use this to add the same option to an item with different values.

No Clear Option

The `?ncopt` is the same as `?opt`, except that the option is not cleared when a `?clearopt` is processed, though it is cleared when `?clear` is processed. Use this when you have an option that should be added to all items, but you're adding a other options to single items. For example:

```
?ncopt:Racial Disadvantage
?opt:Frequency='Daily'
Dependency[Dependency (Popcorn)]
?clearopt
Vulnerability[Vulnerability (White Toast)]
```

This adds the Racial Disadvantage option to both the Dependency and the Vulnerability, but the Frequency option is added only to the Dependency.

Select Option Value

Adds an option that presents the user with a list of choices:

```
?seloptval:Enemy=Choose Favored Enemy,*Enemies
```

The value after the : is the name of the option. An = follows, then the prompt that will be displayed in the dialog displayed to the user. After the prompt the choices appear. Choices can be drawn from lists in the data sheets by indicating an asterisk before the list name. This can be further qualified by adding a colon and a category:

```
?seloptval:Skill=Choose Craft Skill,None,*Skills:Craft
```

Renaming Options

An option can be renamed when it is added. That is, you can give an option a different name than it had in the list of options. For example:

```
?opt:Generic Addition[Cost Break]=-5;Invisibility
```

This adds the Generic Addition option to the Invisibility and renames it "Cost Break."

Remove Option from the Options to be Added

The ?rmopt keyword removes the named option from the list of options to be added. You would specify this to remove an option that you want added to a single item.

```
?opt:Base Level='DX+1'  
Tracking  
?rmopt:Base Level
```

Adding Adjustments

To add adjustments to subsequent automatic items in the list, include an ?adj: entry:

```
?adj:totalLevels+x
```

The above example would add the level of each automatic item to the variable totalLevels.

Adding to Another List

To add the automatic items to another list, include a ?list: entry:

```
?list:Skills  
Skills:Concealment  
Skills:Disguise  
Skills:Stealth
```

The above would add the named items to the Skills list. If the items to be added are defined in the category for the destination list, you must also place the name of the destination list before each item. Otherwise the items are drawn from the category for the current list (though see the ?cat entry below).

Note that an open sublist (see ?open) overrides the ?list directive. Any items enclosed by an ?open;...?close sequence will be added to the sublist.

Specifying the Source Category

To name the category that subsequent items should be drawn from, include a ?cat: modifier:

```
?cat:Skills  
?list:Skills  
Concealment  
Disguise  
Stealth
```

The above would add the named items to the Skills list, drawing from the Skills category.

If you have a ?cat directive and the item you wish to add includes a colon in its name, you must precede the item name with the list name and a colon to prevent the part of the name before the colon from being interpreted as a list name.

For example, if you want to add an item named "Magic Use: Single School" chosen from the Benefits list, you would need specify "Benefits:" before the item name even if you have a ?cat:Benefits directive :

```
?list:Benefits
?cat:Benefits
Magic Use=1
Benefits:Magic Use: Single School=2
```

Clearing Automatic Item Modifiers

The ?clear keyword clears the automatic item modifiers (?charge, ?cat, ?list, etc.), allowing to continue adding more automatic items without the previously indicated modifiers.

```
?list:Skills
Skills:Musical Instrument
?clear
Acute Hearing
```

Charging for Automatic Items

The ?charge keyword begins charging for subsequent automatic items. The cost of automatic items is excluded unless the Charge for Automatic Items (p. 78) flag is set. The ?nocharge keyword turns off charging for subsequent automatic items.

Adding Item References

The ?itemref keyword adds an item reference option to the subsequent items in the list. The following example adds the Throwing skill, then adds the Feint skill with an item reference to the Throwing skill just added.

```
Throwing
?itemref:Skill Reference=Throwing
Hit Location
Feint
?clear
```

To indicate that the user must choose the item to be referenced from a category, specify the category of the items, preceded by a *, followed by a comma and a prompt:

```
?itemref:Skill Reference=*Combat/Weapon,Disarming
Disarming
```

The above will add an item named Disarming. All the items in the Combat/Weapon category will be displayed in a list box and the user must choose the one to be referenced.

Zeroing Item References

The ?zref keyword removes all ?itemref entries.

Clearing the Option List

The ?clearopt command (or ?zopt) will remove all options currently slated to be added to automatic items (indicated by ?opt), except those options created with ?ncopt. There are no arguments.

Conditional Automatic Items

The ?if: keyword allows conditional specification of automatic items. An expression appears after the colon. When the expression for the if is true, all keywords and items that occur between the ?if and the subsequent ?else or ?endif will be processed. If the ?if: expression is false, then keywords and items between the optional ?else and ?endif are processed, and the items between the ?if: and the ?else are not processed.

```
?if:ConfigParam('AutoAddFoci')
?list:Equipment
Equipment:Branch of Holly
?endif
```

The `?elseif:` keyword may also appear within the automatic item list, with an expression delimited with a colon. If clauses may not be nested in the current version.

Add a Category

The `?addcat` keyword adds a category to the list of categories. Note that this category will be lost when the item is converted, so it may make more sense to add an option that specifies a category to obtain this effect.

Allow Duplicate

The `?allowdup` keyword turns off the prevention of duplicates for the next item added.

Default Item Name

The `?defitem` keyword specifies the default item to use if a named item is not present in the data sheet. If the named item is present it will be used, but if it is not currently loaded the default item will be added and given the name of the item that cannot be found.

For example, if the Hunting and Tracking skills are defined, but the Rifle skill is not, Hunting and Tracking and Generic Skill will be added, with Generic Skill being renamed to Rifle.

```
?defitem:Generic Skill;Hunting;Tracking;Rifle
```

Choose Items

The `?choose` keyword specifies a list of items that the user may choose from. A dialog will be presented that indicates the number of items that may be selected. The general format is:

```
?choose:Title,[count,]Item 1[=level], Item 2[=level], ...
```

Title will be taken as the title of the dialog box that lists the available items. *Count* is the number of items that may be selected. This is either a single number, or an allowable range: *min-max*. For example, "2" indicates that exactly two items must be selected. "1-3" indicates one to three items. "0-5" means that zero to five items may be selected. If the count is omitted, any number of items (including zero) may be selected.

If the optional `=level` is present, the item will be set to the specified level. If a level is specified for a category (see the note on `*` below), all items drawn from that category will have the same level.

After the optional count the items to be chosen are listed, separated by commas. A category of items may be specified by preceding the category name with a `*`. For example, the following will display the named items, plus all items in the Athletic category.

```
?choose:Secondary,1,Broadsword,Flail,Shortsword,Spear,*Athletic
```

See `?selexp` below for ways to limit the number of items selected from the category.

The `?uchoose` keyword works exactly the same, except that any items that already appear in the target list will not be displayed. The `?cchoose` keyword is identical to `?uchoose`, except that the costs of the items are displayed and a running total of the costs of the selected items is displayed.

If item selection rules (p. 57) are active, items that do not satisfy the active rules will not be included in the selection list.

Check Requirements

The `?checkreq` keyword indicates that the requirements for categories of items in subsequent `?choose` directives will be checked, and if the requirements are not met the items will not be available for selection by the user. Items named explicitly are not affected.

Select an Item at Random

The `?rand` keyword specifies item selection at random from a specified sublist. After a `?rand` the subsequent items are assumed to be the names of sublists from which items are drawn. Only items that aren't already present in the character sheet will be selected.

For example, the following would select three items at random from the Combat, Covert and Athletic sublists.

```
?rand
Combat
Covert
Athletic
```

The effects of `?rand` persist until the end of the automatic items list, or a `?clear` is encountered.

Select a Number of Random Items

The `?nrand` keyword is similar to the `?rand` keyword, except that it select a number of items at random from the specified sublist.

For example, the following will select five items at random from the Combat sublist.

```
?nrand:5
Combat
```

Items already present in the character sheet will not be selected again. Once the items have been selected, the effects of `?nrand` are terminated; that is, it affects only the next item in the automatic items list. If available items are exhausted an error will be reported, but the subsequent automatic items will be processed normally.

Selection Expression

A large number of items may be selected in a category. To limit this, you may specify a selection expression. For example, to limit the items displayed to those that have a Tech Level option with a value of less than 10:

```
?selexp:optPresent('Tech Level') and optValue('Tech Level')<10
```

Open and Close Sublist

The `?open` keyword "opens" the last item for further item addition, if it was a sublist. The `?close` keyword closes the currently open sublist. For example, the following adds a sublist, renames it Hunting Skills, and adds three skills to it.

```
Sublist[Hunting Skills]
?open
Survival
Tracking
Orienteering
?close
```

Note that an open sublist overrides the `?list` directive (p. 80). Any items enclosed by an `?open;...?close` sequence will be added to the sublist.

Load Datasheet

The `?datasheet` keyword loads the specified data sheet if it's not already loaded.

```
?datasheet:Magic.cds
```

Set Default

The `?default` keyword sets the default checkbox on items subsequently added.

Set Value

The `?setval` keyword sets a temporary variable in the character sheet. This variable can be examined by `?if` keywords. The value will not be saved when you save the character sheet. You should not attempt to set a variable that already exists in the character sheet.

The general form is:

```
?setval:prompt,varName,Choice 1, Choice 2, ...
```

The choices are strings separated by commas. Semicolons and commas not appear in the choices. Empty choices will be ignored. Blanks at the beginning and end of each choice will be stripped.

The variable named `varName` will be assigned a string equal to the choice selected. You should select a unique variable name that is not already used in the character sheet. One convention frequently used is to begin such temporary variables with an underscore ("`_`").

For example, the following sequence sets the variable named `_ccweap` to either "Knife" or "Shortsword," then adds either the Close Combat or Close Combat (Long Weapon) item, setting the item reference option accordingly.

```
?setval:Choose close combat weapon.,_ccweap,Knife,Shortsword
?if:_ccweap='Knife'
?itemref:Skill Reference=Knife
Close Combat
?else
?itemref:Skill Reference=Shortsword
Close Combat (Long Weapon)
?endif
```

Setting a Qualifier

Sets the qualifier for the last option added. For example,

```
?opt:Works Against Hardened Defenses;?qualifer:Force Fields
```

Turning Off Requirements Checking

Turns off requirements checking on the item (sets the Don't Check Requirements flag (p. 78)).

```
?noreq;Spells:Darkness
```

The `?noreq` directive can be used to turn off items that have requirements that are satisfied by some other aspect of the character (such as being a member of a race).

See Also...

List Editing Operations (p. 88)

Edit Automatic Item

Automatic items and directives can be added to the Automatic Item list through this dialog. Click an entry in the Automatic Item Type list to select the type of entry. Select **<item>** to add an item (skill, advantage, etc.).

<item>

Indicates an item to be added to the destination list.

If a category is specified, the item is sought in that category. All items must specify an item name. To select from the lists in the loaded data sheets, click the ... button. If no category is specified, you will be asked for the category.

To specify a new name for the item when it is added, enter the desired name in the **Add As** field.

To specify a value for the item, enter a value in the **Value** field. An expression may be specified.

Add Adjustment

The `?adj` keyword takes arguments.

Add Category

The `?addcat` keyword takes arguments.

Allow Duplicates

The `?allowdup` keyword takes no arguments.

Charge

The `?charge` keyword takes no arguments.

Choose Items

The `?choose` keyword is used to present the user with a list of choices from several items.

Enter the title to be displayed for the selection dialog in the **Title** field.

If a specific number of items must be chosen, enter the minimum number of items in the **Count** field and the maximum number of items in the **to** field. If an exact number of items is to be chosen (say, exactly one) enter the number in the **Count** field and leave the **to** field empty. If any number of items may be chosen, leave both count fields empty.

Enter the name of the item to add in the **Item** field. If the item should be assigned a level, also enter that level in the **=** field. Then click the **Add** button to add the item to the list.

To choose from items in the loaded data sheets, click the ... button. Double-click the desired category, then double-click items from the Available Items dialog. Each item you select will be added to the list. When you're done selecting items, click the **Cancel** button.

To delete an item, click it and then click **Delete**.

To change the item (either its spelling or the level), click it. The item and its value will be placed in the **Item** and **=** fields. Change the values as desired and click **Change** to cause the list entry to match the values in the **Item** fields.

To reorder items, click an item and click the **Up** or **Down** buttons.

To Add Entries to the List to Choose From

Enter the name of the item in the **Item** field. If the item should be assigned a value, enter that value in the **=** field. You can search for items by clicking the ... button. Click the **Add** button to add the item to the list. It will be inserted after the currently highlighted entry.

To Delete Entries

Click the item to be deleted and click the **Delete** button.

To Change the Text or Value of an Item

Click the item in the list. It will be copied to the **Item** fields. Make the desired changes in the **Item** field. Click the **Change** button to transfer those changes to the list.

To Move the Items Up and Down in the List

Click the **Up** and **Down** buttons.

Choose Items by Cost

The `?cchoose` keyword is the same as the `?uchoose` keyword, except that items already present in the character sheet are not listed, and the costs of items are displayed along with the total cost of the selected items.

Choose Unique Items

The `?uchoose` keyword is the same as the `?choose` keyword, except that items already present in the character sheet are not listed.

Clear

The `?clear` keyword takes no arguments.

Clear Options

The `?clearopt` keyword takes no arguments.

Close Sublist

The `?close` keyword takes no arguments.

Conditional Else

The `?else` keyword takes no arguments.

Conditional End

The `?end` keyword takes no arguments.

Conditional If

The `?if` keyword takes an expression as its argument.

Default Item

The `?defitem` keyword takes an item name as its single argument.

Duplicate Option

The `?dupopt` keyword takes an optional name and a value as arguments. If no value is desired, leave the value field empty.

Item Reference

The `?itemref` keyword takes a reference item as an argument.

Load Data Sheet

The `?datasheet` keyword takes a data sheet name as an argument.

No Clear Option

The `?ncopt` keyword takes an option name and a value as arguments.

No Charge

The `?nocharge` keyword takes no arguments.

Number of Random Items

The `?nrand` keyword takes a number as an argument.

Open Sublist

The `?open` keyword takes no arguments.

Option

The `?opt` keyword takes an option name and a value as arguments. Omit the value if no value is desired.

Random Selection

The `?rand` keyword takes no arguments.

Remove Option

The `?rmopt` keyword takes the name of an option that has been indicated with `?opt` or `?dupopt`.

Selection Expression

The `?selexp` keyword takes an expression as an argument.

Set Value

The `?setvalue` keyword displays a dialog and allows the user to select one of the values, which can be examined in subsequent `?if` directives in the automatic item list.

Enter the prompt to be displayed to the user in the **Prompt** field. Enter the name of the variable in the **Variable** field.

To add items to the list, highlight the item you wish to add the item after. Then enter the text for an item in the **Item** field. Click the **Add** button to add the item to the list.

To change an existing entry in the list, click the list entry. The text will appear in the **Item** field. Edit the text in the **Item** field as desired. Click the **Change** button to set the highlighted entry to the value of the **Item** field.

Source Category

The `?cat` keyword takes the name of a category as an argument.

Target List

The `?list` keyword takes the name of the list where the items should be added.

Zero Item References

The `?zref` keyword takes no arguments.

Zero Options

The `?zopt` clears the options. It is the same as `?clearopt`.

<Generic Directive>

Enter a generic directive in its entirety, including any initial question mark and keyword.

Edit Properties: Adjustments

Adjustments change the states of other variables: they can add (or subtract) bonuses to other variables, multiply a variable or members of a particular category in a list by a particular value, divide it by a value, and they can set variables to specific values to indicate that you possess a particular ability.

There are three parts to each adjustment: the name of the variable (or category of item) being modified, the operator for the adjustment, and the expression indicating the appropriate adjustment. This expression may contain the following special variables:

<code>x</code>	Replaced by the current value of the item to which the adjustment belongs.
<code>c</code>	Replaced by the current total cost of the item.
<code>`option`</code>	Replaced by the value of the option named "option". If the option value contains any characters other than letters and digits, it will be enclosed with by parentheses so that it will be correctly evaluated in a formulaic context. Note that the quoting character is the "back tick" ("`"), not the single quote.
<code>`~option`</code>	Replaced by the exact value of the option, without any parentheses.
<code>@...@</code>	Replaced by a value associated with the item (p. 102).
<code>\$adding\$</code>	Replaced by 1 if the adjustment is being set, or 0 if the adjustment is being reversed.

The exclamation point (!) may not be used in an adjustment (it is used to separate reverse adjustment from the initial adjustments -- see below). If you need to use a logical not, use the NOT operator instead.

If a member of a category is being modified, you must specify the name of the list that the category is found in, followed by a colon and then the name of the category.

The types of adjustments are as follows, along with examples.

Addition	<p>defense+2: Adds 2 to the variable indicated.</p> <p>Skills:Craft+4: Adds 4 to all items of the Craft category in the Skills list.</p> <p>Skills:Combat+x: Adds the value of the item to all skills in the "Combat" category of the Skills list.</p> <p>"Skills:Alien Customs"+1: Gives a +1 bonus to the skill with the name (or original name (p. 75)) of Alien Customs in the Skills list. Note the use of quotes around names that contain blanks or other special characters.</p> <p>Skills:longsword+2: Gives a +2 bonus to all items in the Skills list that have the "longsword" variable name.</p> <p>longsword+2: Gives a +2 bonus to the first item added that has the "longsword" variable name. If you add subsequent items with the same variable name, they are internally renamed "longsword0", "longsword1", etc. If you want all items with the longsword variable to have the bonus, you should include the name of the list as indicated above.</p> <p>longsword+`Bonus`: Adds a bonus to the longsword variable that is equal to the "Bonus" option in the item.</p>
Subtraction	<p>vision-2: subtracts 2 from the vision variable.</p> <p>Skills:Combat-2: puts a -2 penalty on the skills in the Combat category.</p>
Multiplication	IQM*2: multiplies the IQM variable's current value by 2.
Division	speed/2: divides the speed variable by 2.
Assignment	combatreflexes=1: sets the variable to 1.

The adjustments are made when the item is added, and reversed when it is deleted (the inverse operation is used, and an assigned variable is set to 0).

Reverse

The reverse operations can be specified explicitly. To add the reverse operations, click the **Reverse** button. A `--REVERSE--` will be added to the entry. For each adjustment specified before the `--REVERSE--`, specify a corresponding "reverse" adjustment after the `--REVERSE--`. Failure to do so will cause the adjustments to be made without ever being reversed.

For example, in the following the `gifted` variable is increased by 1 when the item is added and the `addAge()` function is called, adding its result to the `age` variable. When the item is removed, the `gifted` variable is decreased by 1 and the `removeAge()` function is called, subtracting its value from the `age` variable.

```
gifted+1
age+addAge(20)
--REVERSE--
gifted-1
age-removeAge(20)
```

The `addAge()` and `removeAge()` functions might, for example, be defined to add and remove ages to an array of ages, in characters that have several different advantages that require different minimum ages.

In the text of an adjustment the `--REVERSE--` is indicated by an exclamation point and adjustments are separated by semicolons. The above example would be represented this way:

```
gifted+1;age+addAge(20)!gifted-1;age-removeAge(20)
```

Finally, if you use the `$adding$` keyword, you can do away with the `--REVERSE--` completely by changing the definition of your function that adjusts age. For example:

```
gifted+1;age+adjustAge($adding$, 20)
```

In this case the `adjustAge` function would examine the first argument and add the additional age when it's non-zero, and remove the age when the first argument is zero.

Important Note

The form "Skill:Item Name" can also utilize the item's original name (p. 75), in addition to the current name. This allows you to change the name of the item and still have the adjustments apply. If you don't want the adjustments to apply to an item you have renamed, you must also change the item's original name.

See Also...

- Adjustment Syntax (p. 241)
- List Editing Operations (p. 88)

List Editing Operations

When editing the Automatic Items, Categories and Adjustments lists, the following buttons are available.

- | | |
|---------------|--|
| Add | Add the string in the edit field at the top of the dialog to the list after the highlighted entry. |
| Clear | Remove the highlighted entry from the list. |
| Change | Change the highlighted entry to the value in the edit field at the top of the dialog. |
| Up | Move the highlighted entry up in the list. |
| Down | Move the highlighted entry down in the list. |

Edit Properties: Requirements

Requirements are checks that can be made when items are added to a character sheet. For example, you can require that a spell have a particular level of magical ability, or that if you are Brave that you not be Cowardly.

If you have requirements checking turned on, all requirements in the list must be satisfied when the item is selected; otherwise, **GURPS Character Builder** will notify you of those requirements that have not been met.

New

Create (p. 89) a new requirement. This adds a requirement to the list after the highlighted requirement.

Edit

Edit (p. 89) the highlighted requirement. You may also double-click a requirement in the list.

Delete

Delete the highlighted requirement.

Cut

Cut the highlighted requirement to the clipboard, removing it from the list.

Copy

Copy the highlighted requirement to the clipboard.

Paste

Paste a requirement from the clipboard into the list after the highlighted requirement.

Not

Negate the highlighted requirement. For example, if the requirement is that you may not be blind (i.e., you have the Blindness disadvantage), you would click **New** and create "Disadvantages:Blindness" requirement, then click the **Not** button to negate it and form "NOT Disadvantages:Blindness."

Or

Adds an "---OR---" after the highlighted item. The requirement is then all the requirements listed before the "---OR---", or all the requirements after the "---OR---". For example, if the requirement is that you have "Gifts:Clerical Investment" or "Gifts:Holy Status", you would add the first requirement, then click **Or**, then add the second requirement. You cannot leave this dialog when you have unbalanced ORs.

Then

Adds a "---THEN---" after the highlighted item. A sequence of items and ORs follows. The entire requirement is satisfied if the items before the THEN and at least one of the sets of items between the ORs is satisfied. Only one THEN may appear in the requirements, and it must be before all ORs.

For example, if the requirement is that you are a wizard and have the Fire Spell or the Brimstone spell, you would enter this:

```
Class:Wizard
---THEN---
Spells:Fire
---OR---
Spells:Brimstone
```

Editing a Requirement

You can check for two basic conditions with requirements: whether a value (variable) is greater than a certain number; or whether an item (or number items) is a member of a list. *Note:* if you change the name of an item, other item requirements will not be able to find that item.

To select a requirement, click the radio button by the type of requirement you need, then enter the data in the fields beside it.

Expressions

You can check that a variable has a particular value. For example, if you are creating a sword that requires the user have a total strength and dexterity of 24 or greater (represented by the variables STR and DEX, say), click the **Expression** radio button, then enter "STR+DEX>=24" in the text box beside it.

You may wish to specify a Message (p. 92) to explain in terms the user may easier understand why the expression requirement is not satisfied.

Value in a Dialog

You can require a particular value for a field in a dialog. For example, let's say you have a dialog named "Attributes" with a field named "INT." Let's say you're setting the requirements for the Calculus skill, which requires INT of at least 55. Click the **Item** radio button, enter "Attributes" in the **Window** drop-down list box and enter "INT" in the edit field after the **Item** radio button. Then click the **Value** checkbox, select ">=" from the drop-down box and finally enter "55" for the value.

Any expression may be specified for the value. If compilation errors occur when evaluating the expression you will be allowed to retain the value. This will be normal if you're editing the data sheet and referencing a function in the character sheet. When this occurs, double-check to make sure that the expression is correct.

Item (Single Required Item)

You can also require that an item be in a list. The simplest case is checking to see whether an item is in a list. For example, the "Scribe Scroll" Spell might require the "Literate" Advantage. When you create Scribe Scroll, add a requirement for Literate by clicking the **Item** radio button, entering "Literate" for the item name, and selecting "Advantage" from the **Window** drop-down list box. The current list is the default. When you select (p. 56) Scribe Scroll from the Available Spells dialog, *GURPS Character Builder* will check to see if you also have "Literate" in the "Advantage" list.

To search for items in the Available items list, click the ... button after the **Item** field. A list of the items from the list specified in the **Window** combobox will be displayed. Highlight an item and click **Select** to choose an item.

To require that an item in a list have a particular value follow the instructions above, then click the **Value** checkbox, select the appropriate relationship and enter the required expression. For example, if the "Scribe Scroll" spell also required the "Rune Mastery" skill at level 12 or greater, you would select "Skills: Rune Mastery>=12".

Option

If the **Option** checkbox is checked, the presence (and perhaps value) of an option is tested for the requirement. If no value is specified (i.e., the **Value** checkbox is unchecked), the requirement is satisfied if an option with the name specified in **Option** is present on the named item. If a value is specified, the requirement is satisfied if the option's value matches the condition specified by the **Value**.

For example, to check for the presence of an item named "Maximize Damage" in a list named "Feats" with an option named "Spell" and a value of "Fireball":

- Select the Feats name in the **Window** field.
- Click the **Item** button.
- Enter the name Maximize Damage in the item name field. You can also click ... to choose from a list.
- Click the **Option** checkbox.
- Enter the name of the option, Spell, in the option name field.
- Check the **Value** checkbox.
- Click the comparison drop-down field and select the = relationship.
- Enter an expression that returns the name of the spell in the expression field. In this case, you would enter

"Fireball"

Note that since this is an expression the quotes are required (you can use either single or double). If you don't specify the quotes, and just enter Fireball, it will be interpreted as a variable name, which won't have the intended effect.

Any type of expression can be specified. For example, to indicate that the Weapon Specialization Feat require that a Weapon Focus Feat be present with the same value as the Weapon option for Weapons Specialization, enter something like:

optValue('Weapon')

for the expression. Note that if the option isn't set to anything when the item is first added, the requirement won't make any sense, so you may need to indicate that the item not check requirements when it is added, and handle the initial requirements checking yourself in a script executed after the item is initially added.

Single Item Only

Check this if you expect that there should only be one of the named item present in the character sheet at any time for an option requirement.

If **Single Item Only** is set and the user automatically satisfies this requirement when the named item is already present on the character sheet, the existing item will be used and the option will be changed to the required value.

If **Single Item Only** is not set, another item with that name will be added if one doesn't already exist that satisfies the option requirement.

Do not check **Single Item Only** if you intend that there be several items with the same name and the same option but different option values for that option.

Number of Items

You can also require that at least so many items occur in a particular list, or that so many items belonging to a certain category belong to a list, or that so many categories are represented by items in a list.

When the number of items is computed, the item you are checking the requirements of is not counted, even if it is already part of the list.

Number of Items in List

You can require that at least so many items belong to the named list. For example, say the requirement for the "Dispel Magic" spell is that you know 32 other spells. That is, in order to add the Dispel Magic spell, you must have 32 other entries in the Spells list. To do this, click the **Number of Items** radio button, enter 32 in the edit box beside it, and select Spells from the **Window** drop-down list.

Number of Items in a Category

You can also require that at least so many items in the list belong to a certain category. For example, the Petrify spell is an Earth spell that requires four other Earth spells. First, for all your Earth spells you must add "Earth" the category (p. 78) list. Then, for the Petrify spell requirement you would click **Number of Items**, fill in the value "4" in the edit box, then click the **In Category** button and fill in the word "Earth" in the corresponding edit box, and finally you would select the "Spells" list from the **Window** drop-down listbox.

If the items in the category must have a particular level, click the **Value** checkbox, select the comparison operator from the drop down list, and enter the expression. For example, if the game system requires the character to be a Level 11 Wizard to cast a particular spell, it could be implemented the following way: Let us assume that there several wizard categories (Illusionist, Necromancer, etc.) of character classes. The class is added to the Class list. To implement the requirement:

- Click **Number of Items**
- Enter "1" for the number.
- Click **In Category**.
- Enter "Wizard" for the category.
- Click the **Value** checkbox.
- Click on drop down list for the operator and select ">=".
- Enter "11" for the expression.

This will be reported in the requirements list as "1 Wizard Class>=11".

Number of Items in a Number of Categories

Finally, you can require that you have at least so many items in at least so many different categories. For example, let us say that the Remove Curse spell requires you to have at least 3 spells in at least 10 different categories. To indicate this, you would click **Number of Items** and enter "3" in the edit box, then

click **Number of Categories** and enter "10" in the edit box. Finally, select "Spells" from the **Window** drop-down listbox.

Message

If a message is specified, then it is displayed when the requirement is not satisfied. This allows you to explain in specific terms what the requirement is.

If the message begins with a "\$" the expression will be evaluated as an expression (p. 169). For example,

```
$format('Must have STR of 12 or greater. STR is %d.', STR)
```

Edit Properties: Options

Options are used for many things:

- To modify the cost of an item (increasing it or decreasing it to reflect different modifiers on the item).
- To modify other variables in the character sheet (increasing or decreasing armor effectiveness, or other bonuses or penalties).
- To provide textual notes (noting a page number, special ability, casting time, material component, etc.).

The following buttons are available:

New	Create (p. 92) a new option.
Select	Select an option from the list of options.
Edit	Edit (p. 97) the highlighted option.
Delete	Delete the highlighted option.
Cut	Remove the highlighted option and put it on the clipboard.
Copy	Copy the highlighted option to the clipboard.
Paste	Paste the option on the clipboard into the list after the highlighted option.

New Option

To create a new option, click on the radio button for the type of option you want and click OK. You will then be able to edit (p. 97) the option.

See Also...

- Expression (p. 93)
- Auxiliary Cost (p. 93)
- Fraction (p. 93)
- Multiplier (p. 94)
- Addition (p. 95)
- Percentage (p. 94)
- Adjustment (p. 95)
- Text (p. 96)
- Computing Costs of Items with Options (p. 92)

Computing Costs of Items with Options

When cost modifier options are used with item, they either add to the cost or multiply the cost by some value. When computing the cost, any addition (p. 95) options are first added to the base cost of the item. Then all multiplier options (percentage (p. 94), fraction (p. 93) and multiplier (p. 94)) are multiplied together to form a single multiplier. The total cost is then computed by multiplying this value by the sum of all additions and the base cost. The final step is to round the value according to the rounding specified in the character sheet info (p. 45). You can control this computation by setting the Total Cost Formula (p. 107).

If any auxiliary cost options exist, all cost modifiers affect the auxiliary costs instead of the base cost. The total cost is then computed by summing the base cost and the modified auxiliary costs.

For example, if you have an item that costs 5 points per level (has a Cost expression of "x*5"), has an Addition option of +10, and a fraction option of +1/2, it would cost 45 points if the level is 4: $(4 * 5 + 10) * (1 + 1/2)$.

Expression Option

An expression (p. 169) whose value will be reflected in the options. This is useful for displaying things such as skill levels which are adjusted by a bonus on a magic item, for example. The variable "x" in the expression is the current value (level) of the item, and "c" is the cost. You can reference the values of other options in an expression by enclosing the option name in "back quotes."

For example, if you have a magic sword that has a bonus to the longsword skill, you might list the skill level for the sword as an option, with name "Skill" and the expression "longsword+1". You could also add another text option, "Skill Bonus", and reference that value in the Skill expression in the following manner: "longsword+`Damage Bonus`". The value of the Damage Bonus will replace the reference to it.

You might also use this to specify the range of a missile weapon if the game system's range depends on things such as strength, with name "Range" and the expression "st*2", if the range of the weapon is twice the "st" variable. Spell ranges might also be applicable, with name "Range" and expression "3*wizardlevel".

Auxiliary Cost

An additional cost that will be added to the cost and reflected in the total cost for the item. It consists of a name, value and expression.

The name will be displayed in the list, along with the value and the actual cost.

The value is the basis of the additional cost.

The expression is used to compute the auxiliary cost. Special variables in the cost expression are:

x The value in the auxiliary cost.

The cost expression can be any valid expression. For example, "x*2" or "supcost (x, 2)" where `supcost` is a function defined in the character sheet.

You might use an auxiliary cost on a item that has both a skill level and a power level with independent costs. The item's level would be the skill level, and the option's value would be the basis for the auxiliary cost.

If you have an auxiliary cost, any other options (cost multipliers) will affect the auxiliary cost instead of the item cost.

For example, you have a super power called Control Weather that has a skill level and a power level. The power level affects range and area of control. You could set up the skill level as the item's primary value and the power level as an auxiliary cost option.

Fraction Option

Fractions modify the cost of the item. A fraction may be positive or negative. Positive fractions increase the cost of the item; negative fractions decrease the cost. Fractions should always start with a '+' or '-' to indicate which type of fraction they are. If omitted, a '+' is assumed.

The total cost of the item is computed by the following formula:

```
total cost = base cost * (1 + positive fractions) / (1 + negative
fractions).
```

For example, with a base cost of 10 and rounding up you get the following total costs:

Total Cost	Fraction
------------	----------

7	-1/2
8	-1/4
6	-1/2-1/2+1/4
20	+1
5	-1

You may freely mix positive and negative fractions. Fractions will interact with other cost multipliers (p. 92), though game systems do not usually mix cost modifiers.

Fractions are often used in game systems to reflect extra advantages or limitations that an item may have. For example, a super power with doubled range might cost twice as much (+1), or a psionic ability that was limited to work randomly (activates on a die roll), might cost two-thirds as much (-1/2).

Multiplier Option

Multiplier options modify the cost of the item. A multiplier may be a whole number or a fraction of the form "x/y". Multipliers should always start with a "*" to indicate that they are multipliers.

The total cost of the item is computed by multiplying the base cost by all the multiplier options to determine the total cost. Other modifier options (p. 92) may also affect the total cost.

For example, with a base cost of ten and rounding up, the following total costs would be produced:

Total Cost	Multiplier(s)
20	*2
5	*1/2
7	*2, *1/3

Multipliers are often used in game systems to reflect extra advantages or limitations that an item may have. For example, a super power with doubled range might cost double (*2), or a psionic ability that was limited to work randomly (activates on a die roll), might cost one third as much (*1/3).

Percentage Option

Percentage options modify the cost of the item. A percentage may be a negative number (indicated by "-") or a positive number (indicated by "+"). A percent sign should always be present at the end. If the sign is omitted, it is assumed to be "+".

The total cost of the item is computed by the following formula:

$$\text{total cost} = \text{base cost} * (100 + \text{sum of all percentages}) / 100$$

Other modifier options (p. 92) may also affect the total cost.

For example, with a base cost of ten and rounding up, the following total costs would be produced:

Total Cost	Percentage(s)
20	+100%
5	-50%
8	+10%, +10%, -40%

Percentages are often used in game systems to reflect extra advantages or limitations that an item may have. For example, a super power with doubled range might cost 50% more (+50%), or a psionic ability that was limited to work randomly (activates on a die roll), might cost 30% percent less (-30%).

Addition Option

Addition options modify the cost of the item. An addition may be negative or positive. The addition will be added to the base cost of the item before any multiplicative options (p. 92) are applied.

Addition options are used in game systems that allow extra abilities to be added to items at fixed costs. For example, a super power of illusion might cost 5 points per level, plus a fixed additional 10 points if the illusions can do physical damage to the victim.

Adjustment Option

An adjustment option is very similar to the adjustments (p. 87) that can be made in the Adjustment Option. Exactly the same sorts of adjustments can be made in the options.

The "x" variable in the expression for the adjustment is replaced by the adjustment value.

The following examples show some of the uses of adjustments.

Adjustment	Purpose
armor+x	On a suit of armor, to increase armor effectiveness.
dex+x	On pair of Gloves of Dexterity, to increase dexterity (which would be a variable defined in a dialog as the value of editable field)
posscost+x	On a sword, to indicate that the item cost 550 units of currency.

Distributed Cost Adjustment

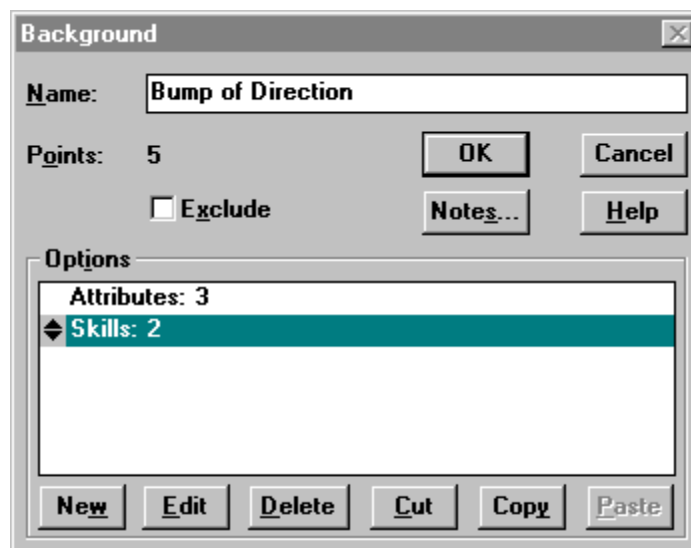
The distributed cost adjustment is similar to an ordinary adjustment (p. 95), except that it is linked with other distributed cost adjustments in the item. The sum of all the distributed cost adjustments will always be the total cost of the item.

When you change the value of one of the adjustments, the other distributed cost adjustments in the item will be proportionally changed so that the sum of all the distributed cost adjustment values is still the total cost of the item.

The distributed cost adjustment expression indicates how the value of the distributed cost adjustment is applied. You might use this in a game system that has two pools of character points to draw from. For example, let's say that your game system allows 100 points of Attributes and 100 points for Skills, and Advantages may draw points from either pool. A typical Advantage would be defined with two distributed cost adjustments:

Option Name	Adjustment Expression
Skill	skills+x
Attributes	attributes+x

where `skills` is the total number of points spent on Skills and `attributes` is the total number of points spent on Attributes.



In this example, the cost of Bump of Direction is 5. If you increase Advantages, the points in Skills goes down, and vice versa.

Distributed Level Adjustment

The distributed level adjustment is the same as the distributed cost adjustment (p. 95), except that the sum of all the distributed level adjustments is always equal to the level.

Text Option

A text option provides a way to include extra information for an item. It can be displayed in printed character sheets and referenced in copy filters.

You might include such things as page number references, casting times, fixed spell ranges, material component lists, etc.

Editing Options

This dialog allows you to edit the value of the option.

Value

The value of the option. If this is a text option, edit the text in the box, or click the up and down arrows to change the value.

If this is a computed value option, you can change the level in the **Value** window and the computed value of the option will be displayed below the level.

If this is an option value list, a drop-down list box will be present. Click the down arrow to display the possible values for the option, then click the desired value.

If this option is intended to be the name of an item, an item browse button (...) is available. Click the ... button to see the list of item names that can be chosen for the value of this option.

Qualifier

Enter text that qualifies the option. This allows you to qualify the option without changing its name.

For example, if the option is named "Linked," which means that it is linked to another power, you could enter the name of the linked power. The text entered in **Qualifier** is enclosed in parentheses and displayed after the name of the option. This would be displayed this way:

Linked (Lightning Bolt): -1/2

Modify

Click this button to edit the details of the option (p. 97).

To Bypass this Dialog and Directly Edit Option Details

Hold down the SHIFT key when clicking the **Edit** button in the item dialog, or hold down SHIFT and double-click the option.

Editing Options

This dialog allows you to edit all the details of the option. To edit the value alone, just double-click the option, or click the **Edit** button. To edit the details of the option, also hold down the SHIFT key and double-click the option (or click **Edit**).

Type

This indicates the type of the option. The other fields you can change will depend on the type of option you are editing.

Name

The name of the option. This is usually displayed first in the option list, separated from the value by value.

Original

The original name of the option. If this is empty, the original name is taken to be the same as **Name**. The original name is used when converting character sheets to get the original option when an option is renamed.

Alias

An alternate name for the option. When the UseAliases (p. 248) game system configuration parameter is checked, this is displayed in filter output in place of the name when printing or copying to the clipboard. It's used for abbreviations in most game systems, but can be used for other purposes as well. When an option is converted without a script, the original alias is retained.

Value

The value of the option. If this is a text option, edit the text in the box, or click the up and down arrows to change the value.

If this is a computed value option, you can change the level in the Value window and the computed value of the option will be displayed below the level.

If this is an option value list, a drop-down list box will be present. Click the down arrow to display the possible values for the option, then click the desired value.

If this option is intended to be the name of an item, an item browse button (...) is available. Click the button to see the list of item names that can be chosen for the value of this option.

Qualifier

Enter text that qualifies the option. This allows you to qualify the option without changing its name.

For example, if the option is named "Linked," which means that it is linked to another power, you could enter the name of the linked power. The text entered in **Qualifier** is enclosed in parentheses and displayed after the name of the option.

Display in Output

If this checkbox is checked, the option is included in the output for character sheets, Filter Copy, etc.

Edit on Insertion

If this is not checked, the Edit Option dialog will not appear when the option is initially inserted. This allows options that have no user-changeable values to be directly inserted without bothering the user with an extra dialog box.

Keep Value on Conversion

If this is checked, the value of the option and the category are retained on conversion. Option values are generally not retained on conversion, unless they have a structured value.

Inherited

If this is checked, the option is "inherited" by items that are contained in a sublist that the **Children Inherit Options** checkbox checked (p. 78) -- the option will be listed along with the options on the item itself when printed or otherwise evaluated. If this checkbox is not checked, the option is not inherited.

Expression

The expression for the option. This is required for adjustment and auxiliary cost options, and will typically be omitted for others. The value (see above) of the option is referenced in the expression with the variable "x". The total value for the option is computed by evaluating the expression. The total cost of the item is referenced with "c". The level of the item itself is referenced with "@v@"

If the expression is omitted for addition, fraction, multiplier and percentage options, the value is used as the total value. The value of the option is referenced by the variable 'o'. References to other options can be made in the expression, allowing options to affect each other's costs. For example, if you had a fraction option that you wanted to be affected by the presence of another option, totaling all related options into a single fraction, you would define the expression for the "main" option with:

```
o+`Option 2`+`Option 3`
```

and defining Option 2 and Option 3 as text or toggle options to avoid adding them in twice.

Token Pasting

The "token pasting operator" ## allows you to place two items next to each other which otherwise cannot be adjacent and be properly resolved. After all other text replacements are resolved, the token pasting operator is simply discarded.

For example, in an adjustment option the variable names to receive the adjustment can be constructed from the actual value of the adjustment:

```
x##PD+`PD`;x##DR+`DR`
```

If the value of the adjustment is "Head" the variable HeadPD is adjusted by the value of the PD option and the variable HeadDR is adjusted by the value of the DR option.

Disp. Exp.:

The expression that is used to display the option. You can reference several values in the expression and format it as desired. The available symbols are:

- x The level of the item.
- c The cost of the item.
- o The option's cost.
- t If the option has a text description in addition to the cost, that text.
- l The label for the option.
- q The qualifier for the option.

For example, the expression below formats a simple option with the label and the cost (note that the quotes are required):

```
"l: o"
```

If you have an expression option that has a cost of 5 per level, and you wish to have each 5 points double the displayed value, the following expression will do it:

```
format('l: x%d, +o', 2^(o/5))
```

This would produce the following output:

```
Multiplier: x8, +15
```

Cat.

An item category. This category is considered in addition to the categories on the item itself when the item is tested for being a member of a category. It is retained on conversion if the Keep Value on Conversion is checked.

Item Ref.

Reference to another item (p. 99). You can refer to other items to have their values affect the costs of an item. Click the list to choose the referenced item. This will be active only if the check expression (p. 101) or option value references an item.

Item Reference Category

The category of items that should be displayed when selecting an item reference. If this is empty, then all items are displayed.

Item Selection

If this field contains text, the **Value** field is assumed to be the name of an item. A browse button (with the label ...) is present beside the **Value** field. The contents of this field indicate how that item is selected. The basic format is:

```
[*]List Name[:Category]
```

If the asterisk (*) is present, the user can select from the list of available items for the specified list name. If not, the user selects from the items currently present in the specified list name. If the category is present, the list of items is limited to those items in that category. If the category is absent, all items are displayed.

Change

Allows you to change the Value type (p. 99).

Types of Options

- Expression (p. 93)
- Auxiliary Cost (p. 93)
- Fraction (p. 93)
- Multiplier (p. 94)
- Addition (p. 95)
- Percentage (p. 94)
- Adjustment (p. 95)
- Text (p. 96)
- Distributed Cost Adjustment (p. 95)
- Distributed Level Adjustment (p. 96)

See Also...

- Computing Costs of Items with Options (p. 92)

Change Option Value Type

There are four Option Value types. To set the type, click the radio button associated with that type.

- Plain Text (p. 99)
- Item Reference (p. 100)
- Toggle (p. 100)
- List (p. 100)
- Expression (p. 100)

Check Exp...

Set the check expression (p. 101) for this option. This allows additional checks on the value of the item.

Plain Text

The option simply has the value of the text. Enter the desired value in the edit box.

Item Reference

This option value refers to another item, allowing you to have one item depend on another item for its cost.

For example, if your game system allows you to subtract the level of another item from the cost of a second item, you can use an item reference. In this instance, you would enter the following value for the item reference text of an Add option:

```
-@v@
```

The item references (p. 102) are codes surrounded by @@. References may also be made to the current item, by preceding the reference code with the "*" character. For example, the following item reference returns 10 if the basic cost of the current item is greater than the basic cost of the referenced item:

```
@*bc@>@bc@ ? 10 : 5
```

References to the referenced item can be made with @! . . .@ to make the references explicit: the above is equivalent to @*bc@>@!bc@ ? 10 : 5.

The item references are inserted directly into the text, and the result is evaluated as an expression. For example, to reference the category of the reference item use the following

```
"@cat@"
```

You must also set the item referenced (p. 97).

Toggle

Check the checkbox to turn the option on. If the checkbox is checked, the option will be applied. If it is unchecked, it will be ignored.

Enter the text of the value in the edit box.

List

The option value consists of a list of choices, one of which is the current choice. You may select among only these choices. Each choice consists of a label and a value. If you omit the value, the label is assumed to be the value, and vice versa.

To add a label and value pair to the list, click the label/value pair in the list that you wish to add after. Enter the desired value for the label, then the value. If you omit one or the other, the value and the label will be the same. Click the **Add** button.

Note: the ";" (semicolon) character may not be used in the label or value.

Button	Use
Add	Adds the Label/Value pair to the list after the highlighted item.
Delete	Deletes the highlighted label/value pair in the list.
Change	Changes the highlighted label/value pair in the list to the values currently in Label and Value.
Up	Moves the highlighted label/value pair up.
Down	Moves the highlighted label/value pair down.

Expression

The option value consists of a level and computed value. Associated with the value may be a cost expression, a check expression and a check message. This allows you to control the possible values for an option and compute a "cost" based on a level that you choose.

Note: the ";" (semicolon) character may not be used in any of these fields.

Cost Expression

The computed value is determined by replacing the "x" variable in the cost expression with the level. The computed value is used for the actual value of the option.

For example, if you want an option to cost 2 per level, the cost expression would be " $x * 2$ ".

Check Expression

Whenever you change the level, the check expression is evaluated to determine whether the value is acceptable. If the check expression is false (zero), the check message is displayed. If the check expression is true (non-zero), the value is accepted. When evaluating the check expression, the "x" variable is replaced with the level and the "c" variable is replaced with the cost value, as computed from the cost expression.

For example, if the level must be between 1 and 10, the following check expression could be used: " $x >= 1$ and $x <= 10$ ".

Check Message

When the check expression is not satisfied, the check message is displayed. The check message (p. 158) may be preceded with a "\$" to indicate an expression that is parsed in the context of the current character sheet and this item. If the message begins with any other character it will be displayed to the user verbatim. If no check message is specified, a standard warning is displayed along the lines of "The value 'z' is not allowed." The variable x is replaced with the proposed value of the item, while the variable o is replaced with the name of the option.

Option Check Expression

This specifies a check expression that is evaluated whenever the item's value changes. If the expression is false, the message is displayed. This is useful for telling the user when a game system constraint is violated. For example,

Check Expression

An expression (p. 169) that is evaluated to determinate the acceptability of chosen values. If this expression evaluates to 0, the check message is displayed.

The following special symbols can be used in the check expression:

c	The basic cost of the item, excluding all multipliers.
x	The level of the item.
o	The value of the option.
`option`	Reference to an option.
@...@	References to other item values (p. 102), such as multipliers, additions, etc.
@!...@	References to values (p. 102) in other items (p. 99).

Reference Item

Check this box if there are references to another item (p. 99) in your check expression. This is required for @!...@ references to be evaluated properly. References to entities in the current item can also be specified with @*...@.

Check Message

The message displayed if the check expression evaluates to false (0). The check message (p. 158) may begin with a "\$" to indicate an expression that will be parsed on the context of the item and character sheet. If the message begins with any other character it will be displayed to the user verbatim. The special symbols in the Check Expression can also be used in the Check Message.

Examples

Limit the value of the item to less than the specified value. Add a text option named "Limit" with the following check expression:

```
x <= `Limit`
```

Now when the user tries to increase the level of the item above the value of "Limit," the check message will be displayed.

Require that the basic cost of an item be double that of another item:

```
@bc@>=2*@!bc@
```

You must also set an item reference for this to work.

Option Item References

Option item references allow options to refer to values in the main item and in other items. The values available are:

@a@	Additions: The sum of all the add options.
@ao@	Additions Only: The sum of all positive add options.
@bc@	Basic Cost: the basic cost of the item, disregarding any option modifiers.
@c@	Cost: the total cost of the item.
@cat@	Category: the category of the item.
@d@	Default: the text of the default value.
@fd@	Fractional Divides: The sum of the fractional divides. That is, all the negative fractions.
@fm@	Fractional Multiplies: The sum of the fractional multiplies. That is, all the positive fractions.
@m@	Multipliers: The product of all the multipliers, fractions and percents.
@mo@	Multipliers Only: The product of just the multipliers.
@n@	Item name.
@nc@	Number of children. For sublists, the number of items that are contained within the sublist. A sublist and all its children count as a single child.
@p@	Percents: The sum of all the percents (i.e., if there is a 20% and 50% option, the @p@ reference will return 70).
@so@	Subtracts Only: The sum of the negative adds.
@v@	Level

You can also use the longer names defines for the @foreach (p. 194). For example, @origName@.

Note: In Option Check Expressions that refer to other items, @...@ or @*...@ accesses values in the main item, while @!...@ access values in the referenced item. For example, if the check expression is

```
x>=2*@!v@
```

The check expression will require that the cost of main item be double that of the item referenced.

Edit Properties: Format

The **Format** string is used to display the item name and other properties of the item in the list. The **Sel Format** string is used to display the item when it is in the Available Items list. The formats in the **Alt Fmts** list is used to display the item when an alternate display format (p. 34) is in effect.

Format String

Each format string consists of string with references to properties:

```
%0ln%-5rc%-10rv%-16l1
```

Each property reference consists of a '%', followed by a number, which is the distance from the edge of the list (in character widths). If the number is positive, the number indicates the distance from the left side of the list. If negative, the distance is from the right side of the list.

The type of justification for the field is indicated by the next character:

- l Left
- r Right
- c Centered

The last character indicates which property of the item to display.

- n Name
- v Value (level)
- c Cost
- d Default value.
- e Item expression. The text of the expression for an item. This is not necessarily the same as its value, though it is if the expression is a constant.
- l Class
- o Option. This must be followed by a quoting character, the name of the option to display, followed by the closing quote (must be the same as the first quote). For example,


```
%0ln%-12ro'Price'%-5rv
```

 would display the item name, followed by the Price option, followed by the item level. If the option indicated in the quotes does not exist, "???" will be displayed.
- t Category list
- x Arbitrary expression. This must be followed by a quoting character, the expression to display, and the closing quote (must be the same as the first quote). For example,


```
%0!!Total Cost/Weight%-18r!$^x'posscost'%-9rx'possweight'
```

 displays some text, the value of the variable "posscost" and the value of the variable "possweight". If the expression is bad, "???" will be displayed.
Note: Expressions will not automatically update when a variable in the expression changes unless the item defines a variable name and has a cost expression that includes that variable. A common trick is to include the variable in the cost expression by adding "0*variable" to whatever the desired cost expression is. Then, whenever variable changes, the text displayed in the list will be updated.
 References to other options can be made in the expression by indicating the option name in backquotes. References to option item references (p. 102) can also be made.
Selection Formats: Since the selection format operates in the context of the data sheet, any variables and functions defined in character sheets are not available. Only built-in functions can be used in the expressions for the selection format.
- ! No property — this just allows you to position the text following the position indicator.

You can also make the same references these same things without specifying a location by using "^". For example, the format

```
%0ln-^v%-5rc
```

will display the name of the item, followed by a hyphen, followed by the level, with the cost right-justified at the far right.

If you want to include a '%' symbol in the output, simply double it.

Alt Fmts

The alternate formats are used to display the item when an alternate list display format (p. 44) has been selected. Normally the format in the **Format** field is used. If the list is being displayed with an alternate format (p. 34), the format with the same name is used, if present. This format should be complete, specifying complete field locations for the name, cost, etc. For example:

```
%0ln%-10rc
```

If there is no alternate format with that name, the (Default) alternate format is used (this has an empty name field). This format is partial: it should not be a complete specification because it will be inserted into the list alternate format string where the \$! sequence occurs. Typically it will look something like "^n-^v", or whatever is required to display something meaningful.

Click **Add** to add a new format to the list. Click **Delete** to delete the highlighted format. Click **Edit** to change the highlighted format.

Edit Properties: Lookup Array

A lookup array is useful when you want the level of a item in a list to be displayed as a string of text instead of a number. For example, associating the following strings with the corresponding values,

Wonderful	5
Great	4
Okay	3
Crummy	2
Rotten	1

can be done two ways. Either way, when the level of the item is equal to a value in the array, the text for that value is displayed. For example, if the level of the item is 3, "Okay" will be displayed for the item's level.

Create an Array

If many items share the list of values, create an array (p. 167) with the following values: "Wonderful", 5, "Great", 4, "Okay", 3, "Crummy", 2, "Rotten", 1. Then, in the definition of the item, you must name the array you created in the Lookup Array edit box. When the item is edited (p. 68) normally, the list of textual values will be displayed instead of the numerical values. When the costs are computed, however, the numerical values will be used.

Constant Values

If the item is unique, then you should specify the lookup array as a constant. This is indicated by starting the lookup array with a dollar sign: \$. The values that would appear in the array are separated by semicolons:

```
$Wonderful;5;Great;4;Okay;3;Crummy;2;Rotten;1
```

If a *value entry* begins with \$, then the value is evaluated as an expression. For example:

```
$Attractive;1;$sex=2?'Beautiful':'Handsome';2
```

will display Attractive when the value is 1. When the value is 2, Beautiful will be displayed if `sex` is 2, and Handsome will be displayed if `sex` is not 2.

Dialog Type

Selects the type of dialog to use when editing this item.

Standard

Allows you to change the level, cost and options for the item. This is the dialog used by default.

Name Only

Allows you to change only the item name. This is useful for items that have a fixed (hidden) cost or for which only the name is really relevant. For example: text-only, constant-cost "quirks"; possessions in game systems that don't account for weight or other aspects, etc.

Options Only

Allows you to change the item name and add options. The level and cost are not directly changeable, though by adding options you can change the cost.

Level Only

Allows you to change the item name and level, as well as add options. The cost is not displayed at all. This is intended for items that have only a value associated with them, and no point cost.

Cost Only

Allows you to change the item name and cost, as well as add options. No level is associated with the item. This is for items that have fixed cost and no level.

No Cost

Allows you to change the item name and options, while display no cost. Used for game systems that have items that have options, but no real cost or level associated with them.

Custom

Allows a script to handle all the editing for the item. You must set the script in the **Script** field of the Automation tab of the Properties dialog (or the `script` keyword of your item macro definition).

The script should be as short as possible. Typically it should be a single call to subroutine defined in the load script (p. 47) indicated in the character sheet info. For example, if you have a subroutine named `EditMagicItem` in the load script that edits a magic item, your automation script would be:

```
EditMagicItem();
```

Edit Properties: Expressions

An item's expressions are used to calculate the costs and default values, and check the validity of the value entered.

- Cost Expression (p. 105)
- Check Expression (p. 106)
- Default Value (p. 106)
- Total Cost Formula (p. 107)

Cost Expressions

The cost expression indicates how the cost is computed based on the value for the item. The expression (p. 169) is a normal expression, with the special symbol "x" standing for the current value of the item.

If you wish the cost for the item to change based on another variable, or for the item's display in the list to update when another variable changes, you must:

- Assign a variable name to the item.
- Reference the variable in the cost expression for the item. If the cost doesn't derive from the variable directly, any kind of reference will do (adding a `+0*variable` is typically done).

The following examples illustrate some of the possible cost expressions. If you have a cost expression that is used commonly, you should define a function (p. 166) in the base character template.

Example	Explanation
<code>x*2</code>	The item costs two points per level.
<code>~x*2</code>	The item costs two points per level, reduced or increased by any bonuses or penalties on the item.
<code>~x>dex? (~x-dex) *5:0</code>	The item costs 5 points for each additional level over the variable "dex" (which is assumed in this example to be set in a dialog). <i>In order for the cost to be automatically updated when dex is changed, you must also give the item a variable name.</i> The unary operator "~" is also used in case any bonuses are applied to the variable.
<code>x=1?10:25</code>	If the level is 1, the cost is 10, otherwise the cost is 25. If you were to use this type of cost expression, you would probably also set a check expression (p. 106) to make sure that you can enter only two different values.
<code>skcost (~dex)+0*str</code>	The cost of the item is based on the unadjusted value of <code>dex</code> . Furthermore, whenever <code>dex</code> or <code>str</code> changes, the item's cost will be reevaluated and its entry in the list will be updated. This would be useful, for example, if options that are referenced in the displayed

format for the item reflect the current value of `str`. Without the `+0*str` term in the cost expression, changes in `str` would not affect the item and the entry in the list would not be updated.

Two special unary operators are useful in cost expressions:

- `~` The base value of a variable, disregarding any bonuses applied to the variable. This should generally be used in cost expressions that are applied to variables that may have bonuses.
- `&` The value of the bonuses for the variable.

If you reference bonuses you must define a variable for the item.

Check Expressions

A check expression allows you to create an item that verifies that you have selected an allowable value for an item in the Item Edit (p. 68) dialog. A check expression is a normal **GURPS Character Builder** expression with two special variables:

- `x` The value (level) of the item.
- `c` The cost of the item.

When you set the level or cost for an item, **GURPS Character Builder** checks the value against the check expression. If the expression is true, then the new value is silently accepted. If it is false, a dialog comes up informing you that the new value is not allowed. You may click OK to take the value anyway, or Cancel to go back to the original value.

Typical check expressions might be:

`x>=1 and x<=3`

Ensures that the value that you select is between 1 and 3

`c >= 0.5`

Ensures that the cost is greater than or equal to one-half point.

`x>=&x and x>0`

Ensures that the level of the item is always at least as large as the bonus on the item (and always positive). If you reference bonuses, you must define a variable for the item.

Default Value

The default value has a different interpretation based on whether the **Based on Default** flag is set.

Not Based on Default

If you specify a default value for an item, the **Default** checkbox in the Item Edit (p. 68) dialog will be active. If you click that checkbox, the level of the item is set to the default value. This can be a constant, or it can be an expression (p. 169). Typical default values might be:

`int-5` The value of the "int" variable minus 5.

`max(iq-1, shortsword-2, forcesword-2)`

The maximum of the "iq" variable, the shortsword skill - 2 and the forcesword skill - 2.

If you have a named variable, the cost for selecting the default value is 0 by default. If you specify a value in **Default Formula**, that expression is used to compute the cost.

Based on Default

If the Based on Default flag is set, the Default Value and the Default Formula are used to compute the cost of the item. The **Default Formula** is used to compute the cost of the item.

The **Default Value** is a list of expressions separated by semicolons. The first expression is the expression used as a pattern for the rest of the expressions.

The subsequent expressions should be of the form `Variable+value`: that is, a variable must be the first thing in the expression.

When the `defaultValue` function is called, its value is computed with the following procedure:

- Each subsequent expression is evaluated.
- The variable `v` in the pattern is replaced with the result of the expression that follows the variable.
- The variable `x` in the pattern is replaced with the level of the item.
- The newly formed expression is evaluated with the item associated with the variable as the "current" item. This allows functions such as `itemInfo` to return information about the item associated with the variable referenced in the default value. If the variable has no such association, the item info functions will return zero.

The maximum result of all the evaluated expressions is returned for `defaultValue`.

In order for **GURPS Character Builder** be able to recognize when the variables referenced in the **Default Value** are changed, the variable names must be define on items in the same list as the referring item. Another way of detecting changes is to reference a variable in the Default Formula.

Total Cost Formula

The total cost formula is used to compute the total cost of an item (p. 92), using the base cost, using any auxiliary costs, multipliers or additions indicated in the options. The built-in total cost formula, if no auxiliary costs are included in the options, is:

```
round((c + a)*m)
```

If auxiliary costs are included in the options, the formula is

```
round((aux + a)*m)+c
```

If no total cost formula exists for the item, the Default Total Cost Formula (p. 46), if specified, is for the character sheet is used.

Variable	Meaning
<code>c</code>	Base cost of the item
<code>a</code>	Sum of all additions and subtractions
<code>m</code>	Product of all multipliers, percentages, and fractions
<code>aux</code>	Sum of all auxiliary cost options
<code>round</code>	Rounding set for the character sheet in the character sheet options.
<code>@ref@</code>	References to other item values (p. 102): keywords surrounded by @ signs.
<code>`option`</code>	References to option values.

If you set your own cost formulas, you can use the above variables to access cost information for the item. For example, if you wanted to set a lower limit of 10% of the original cost (always rounded up), you could use the following total cost formula:

```
aux > 0 ? round(max((aux+a)*m,aux/10))+c : round(max((c+a)*m,c/10))
```

Edit Properties: Automation

Script

The item script is executed only once: the first time item has been added. It will not be executed when an item is pasted, or dragged and dropped. Variables, functions and subroutines in the load script (p. 47) may be referenced. This script is similar to the script that is executed when a control changes.

When the script starts, the current item is considered to be the last item added. This means that script variables such as `$l` and `$c` reference the level and cost of the current item.

The script can contain any script commands (p. 260) except `insertItem`, which will cause problems if you insert an item into a different list from the list currently being inserted into. The script can also contain references to the special symbols "x", "~x", "&x", "c" and ``Option``.

The item script can be used for automating tasks that can't be accomplished by other means. For example, automatic items are added before the user sets the level for the item. If you wanted to base automatic item selection on the level selected for the item, you could use a script to add the items.

Let's say that you have a Sword Master advantage that grants the character a +1 bonus on sword skills for each level, plus one skill from the Sword category for each level. The item definition could be written this way, with the script referencing the `addItem` subroutine defined in the character sheet's load script:

```
item "Sword Master"
  level 1
  adj swords+x
  formula x*5
  script "addItem(x, 'Weapon', 'Skills', '*Sword')"
```

The load script could be defined this way:

```
sub addItem(num, type, listName, items)
{
  addAutoItems "?list:$$listName;?cat:$$listName;?choose:Choose $$type
Skills,$$num,$$items",
    retval, noautodup, dontmarkauto, autochargecost;
}
```

Editing Print Templates

Print templates are used to print the contents of character sheets. You can use several different print templates to print information from the same character sheet (p. 298) in a different format.

How to...

- Add graphic objects (p. 109)
- Edit graphic objects (p. 110)
- Use multiple pages (p. 119)
- Flow text from one page to another (p. 111)

Menu Commands

- File (p. 2)
- Edit (p. 9)
- Utilities (p. 13)
- View Menu (p. 124)
- Text Menu (p. 114)
- Draw Menu (p. 113)
- Align Menu (p. 114)
- Window (p. 48)
- Help (p. 49)

Adding Graphic Objects

To add a graphic object, you must first select the type of object in the object palette window by clicking the icon for that type of object, or choosing the corresponding command from the **Tools** menu. Then click the left mouse button in the print template window where you want the object to start, and drag the mouse to draw the object. Let go of the mouse button to create the object. The object will have the default formats (p. 12).

The types of objects available are:



Cursor. Allows you to select and resize objects. You can also directly set the dimensions of objects by choosing the **Edit | Edit Object** (p. 111) command.



Rectangle.



Rounded Rectangle. You can set the degree of roundness of the rectangle by clicking the object and choosing the **Edit | Edit Object** (p. 111) command.



Circle (or ellipse).



Line. The orientation of this type of line is not constrained horizontally or vertically.



Horizontal line.



Vertical line.



Text box. The text in this kind of object can contain character sheet references (p. 185) and commands (p. 187) that access information about the character sheet. They may also be used to flow text (p. 122) onto other pages.



Text. This text object should be used for titles, labels and so forth.



Picture. This creates a window in which you can reference pictures (bitmaps) defined in the character sheet. To set the reference, click the picture window and enter the name of the dialog, separated from the name of the picture with a colon (Dialog:Picture).



Bitmap file. The area you trace out will be filled with a bitmap read from a file that you choose. The bitmap may be a Windows BMP file, a JPEG file, or a PNG file. The image's original aspect ratio will always be maintained.



Polygon. The rectangle you trace out will be filled with a regular polygon inscribed inside the circle (ellipse) defined by the rectangle. If the polygon is wider than it is tall, the polygon will be stretched horizontally. Hold down the SHIFT key to force the area to be a square. When you let up the mouse button, a dialog will prompt (p. 110) for the number of the polygon's sides and its angle of rotation.

You can set the border width and fill color for the objects with commands on the **Draw** menu.

Shortcuts

Once you have selected an object type, you can click and drag with the right mouse button to create another object of the same type.

When a single object is selected, you can move the selection to another object with the TAB key. To select the next object (an object in front of the currently selected object), press TAB. To select the previous object, press SHIFT+TAB.

Polygon

This dialog appears after you outline the rectangle that indicates the circle (ellipse) inside which the polygon is inscribed.

Number of Sides

Enter the number of sides for the polygon. The default value is 6 (for a hexagon). You may enter any number greater than two. This number is remembered and reused as the default for as long as the application runs.

Angle of Rotation

The angle in degrees by which the polygon will be rotated. By default, all polygons with even numbers of sides (except four-sided diamonds) are drawn with a flat edge on the top and bottom. Polygons with odd numbers of sides are drawn with a vertex at the top. This angle is remembered and reused as the default for as long as the application runs.

Editing Graphic Objects


Selecting an Object for Editing

Click on the object with the mouse. Selected objects are indicated by the small black squares at the corners of the object. These black squares are called *handles*.

Selecting More than One Object

Hold down the shift key and then click on all the objects you want selected. Alternately, you can click and drag on the screen (on no object) and a *selection rectangle* will appear. Any objects completely contained within the rectangle will be selected when you let the mouse button up.

Deleting Objects

Select the object(s) and press the Del key, or click the  button on the button bar. You can also select the **Edit | Clear** command.

Changing the Size of Objects

Select the object. Click and drag on one of the handles to change the object size.

Changing the Colors of Objects

Select the object and choose the **Draw | Fill Color...** or **Draw | Line Color...** command.

Changing the Color of Text

Select the text object(s) and choose the **Text | Font...** command. Select the color you need from the dialog box. The background color of text objects is set with the **Draw | Fill Color...** command.

Constraining Object Shapes

Hold down the shift key before you click the mouse button when adding an object. Rectangles will be constrained to squares and ellipses to circles. To force an existing object to a square or circle, select the object, then hold down the shift key and click on the handle. Drag until the object is the desired size. When you constrain a metafile picture, it is forced to its original aspect ratio (if it had one).

Lining Objects Up

Select the objects you want to align and select the corresponding command from the **Align** (p. 114) menu.

Forcing Objects to Be the Same Size

Select the objects you want to be the same size and select the desired command from the **Align** (p. 114) menu.

Edit Object

You may set the size and location of the selected object directly, using the following units: in (inches), cm (centimeters) or pt (points). If no units are specified, the default units are assumed. This can be set in the Windows Control Panel.

The **Original Size** button is active only for objects pasted from other applications. It returns the object to its original size.

For text objects, the text is present so that you can edit it. This is useful if you use a symbol font or have a very small text object.

Special Keys

When editing the text of an object in this dialog, you must press special key combinations to get certain functions.

Character	Keypress
TAB	CTRL+TAB
Undo	ALT+BACKSPACE

Edit Rounded Rectangle

You may set the size and location of the selected rectangle directly, using the following units: in (inches), cm (centimeters) or pt (points). If no units are specified, the default units are assumed. This can be set in the Windows Control Panel.

The Corner Sizes indicate the radius of an ellipse that is drawn for the corner. The larger the value, the more rounded the rectangle will be.

Flowing Text from One Page to Another

The text boxes in a print template are of a fixed size. If character sheets may have more items than will fit in the available space, you should create Continuation pages in your print template so that the overflowing text is not simply lost.

To flow text from one page to the next:

- Select the text box object that you wish to continue.
- Select the **Text | Link...** command (p. 122) (or press CTRL+K key).
- Enter the name of the text link tag you wish to use in the **Tag** field. (Be sure to leave the **Continuation** checkbox unchecked.)

- Click the **OK** button.
- Select the **File | New Page...** (p. 120) command (or press CTRL+SHIFT+N).
- In the Edit Page Info dialog type the page name.
- Set the **Page Type** to Continuation.
- If you have a common background page, select the Background.
- Click **OK**.
- Create the text box that the text should flow into with the **Tools | Text Box** command.
- Select the **Text | Link...** command (or press CTRL+K).
- Click the arrow on the **Tags** field to bring down the list of available tags.
- Click the tag that you want to continue.
- Click the **Continuation** checkbox.
- Click **OK**.
- Save the print template and try it out.

Draw Menu

If an object is selected, the commands on this menu reflect the attributes of that object. If no object is selected, then the commands reflect the default formats (p. 12).

Snap to Grid

When this is checked all objects are constrained to an invisible grid. This makes it easier to line things up quickly.

Framed

When this is checked, rectangles, ellipses, text and text boxes will have frames. The line style and line colors control the frame style and color.

Filled

When this is checked, rectangles, ellipses, text and text boxes will be filled in with the color specified by the fill color.

Ruled

When this is checked, text and text boxes will be "ruled," which means that horizontal lines will be drawn beneath all the text. The line height (p. 115) indicates the distance between the lines.

Cell Lines

Draws a horizontal and vertical lines inside a text box or rectangle to form a specified number of "cells." (p. 113)

Line Style

This sets the thickness for lines and the frames on rectangles, ellipses, text and text boxes.

Line Color

This sets the color for lines and the frames on rectangles, ellipses, text and text boxes.

Fill Color

This sets the color for rectangles, ellipses, text and text boxes.

Cell Lines

It is common in character sheets to have a "grid" of small boxes. The easiest way to do this is to use the **Cell Lines...** command, which divides up a rectangle or text object into a specified number of equally sized rectangles, or cells.

For example, to draw a rectangle that is divided into four smaller, equal-sized, rectangles, set the Horizontals Cells to 4 and the Vertical Cells to 1 (or leave it at 0). For a rectangle that is divided

If you wish to have rules in a text object (that is, underline all text lines for the entire width of the text object), you should use the Ruled (p. 113) attribute instead. The rules depend on the font size, whereas the cell count is independent.

The cell lines are the thickness specified by the line style (p. 113).

Removing Cell Lines

Set the horizontal or vertical cell count to 0. The object will contain a single "cell."

Align Menu

The following commands appear on the **Align** menu:

- | | |
|-------------------|---|
| Left | Align the left sides of all highlighted objects to the left-most point of the left-most highlighted object. |
| Right | Align the right sides of all highlighted objects to the right-most point of the right-most highlighted object. |
| Top | Align the top sides of all highlighted objects to the top-most point of the top-most highlighted object. |
| Bottom | Align the bottom sides of all highlighted objects to the bottom-most point of the bottom-most highlighted object. |
| Horizontal Center | Align the vertical centers of all highlighted objects in a horizontal line. The left-most highlighted object defines the position of the center line. To use right-most object as the reference instead, hold down the SHIFT key. |
| Vertical Center | Align the horizontal centers of all highlighted objects in a vertical line. The top-most highlighted object defines the position of the center line. To use the bottom-most object as the reference instead, hold down the SHIFT key. |
| Smallest Height | Make all highlighted objects the same height as the shortest highlighted object. |
| Largest Height | Make all highlighted objects the same height as the tallest highlighted object. |
| Smallest Width | Make all highlighted objects the same width as the narrowest highlighted object. |
| Largest Width | Make all highlighted objects the same width as the widest highlighted object. |

Text Menu

Commands selected on the Text menu affect the highlighted object(s). When no objects are highlighted, they affect the default properties.

Plain, Bold, Italic, Underline

Change the text style. Bold, Italic and Underline are additive; selecting Plain cancels any other styles.

Left, Center, Justified, Right

Change the paragraph formatting.

Font Size

Set the font size. The size is in points.

Font Face...

Sets the font name (p. 114) only.

Font...

Set the font name, font style, color and point size.

Properties...

Set the text properties (p. 115).

Link...

Set the text link tag (p. 122) for a text box.

Font Face

Sets the font "face" without changing any other attributes of a text object. This is useful for setting the font for multiple text objects to the same font at one time, without changing the size or style.

Type the name of the font in the edit box, or click the name in the list.

Text Properties

Set the properties of the text. You may follow any distances with the following units: "cm" for centimeters, "in" for inches and "pt" for points.

Alignment

Sets the alignment of the paragraph. **Left** is left-justified with a ragged right margin. **Right** is right-justified with a ragged left margin. **Justified** is justified on both the right and left. **Centered** centers each line of the paragraph.

Font

Allows you to set the font name, style and size.

Tab Stops

Displays the list of tab stops set for the paragraph. To add a tab stop, enter the distance (optionally followed by the unit, cm or in), select the tab stop alignment (left, right, centered or decimal) and click the **Set** button. If you omit the unit, the current measurement unit is assumed. To clear a tab stop, select it in the list and click the **Clear** button. To remove all tab stops, click the **Clear All** button.

All tab stops apply to the current column.

It is easier to set tab stops by eye if you turn on rulers (p. 124) and change them in the ruler window (p. 125).

Default Tab Stop

If no tab stops are indicated, tab stops occur at this default distance. The default tab stops are always left-aligned.

First Indent

The first line of the paragraph is indented by this distance. If this is less than the left indent, it is a "hanging indent." If the first indent is less than the left indent, the first tab in the line will place the text following it to the left indent position. You can also change this in the ruler window (p. 125).

Left Indent

All lines in the paragraph but the first are indented by this distance. You can also change this in the ruler window (p. 125).

Line Height

The height of each line in the paragraph. The value "Auto" causes the line spacing to depend on the font being used.

Wrap Tabs

Checked to make text wrap to the next line when text would cross over a tab. If unchecked, the text is truncated and an ellipsis is used so that the text all stays on the same line.

Num. Columns

The number of columns in the text object. If you specify so many columns that the column spacing alone would be larger than the object that contains the text, you'll get an error message: "Zero column width", because the width of the columns themselves would be reduced to zero.

Column Spacing

The distance between each column. If the text object is ruled, the rules will break between columns.

Orientation

The orientation of the text. This is expressed in degrees. You can select the following orientations:

- | | |
|-----|---|
| 0 | Normal orientation (text is plotted right to left, upright characters). |
| 90 | Rotated 90 degrees (text is plotted bottom to top, characters rotated 90 degrees to the left) |
| 180 | Rotated 180 degrees (text is plotted right to left, characters upside down). |

270 Rotated 270 degrees (text is plotted top to bottom, characters rotated 90 degrees to the right)

Edit Template Information

Game System

Select an existing system from the list by clicking on the down arrow, or simply type in the name of the new game system.

Sheet Type

The character sheet type that may be used with this print template. For example, there may be different print templates for Characters and for Vehicles. To indicate that this print template can be used with any type of character sheet for this game system, leave the sheet type blank.

Game System Level

The "release level" for the print template. This should match the levels of the character sheet and data sheets for the same game system. If significant changes are made to the definitions of functions in the character sheet, this level should be changed to match the character sheet. Then, if you attempt to print an old character sheet with this print template, you will be notified of the mismatch.

Include File

The name of an "include file" that contains macro definitions (p. 226). The include file will be searched for in the ***GURPS Character Builder*** source directory (p. 15).

Include files and macros are useful for placing shared logic into a common location, so that you need change the definition only once and it can be used in multiple places.

Inhibit Detail Printing

If this is checked, the contents of the Details window (p. 13) are not printed after the template. This is useful for print templates that only print certain information about the character, such as spell books.

Orientation

Click the **Portrait** button to select portrait mode for the print template. Click **Landscape** to select landscape mode. In landscape mode the character sheet will be printed on the page with the longer dimension of the paper in the horizontal direction. In portrait mode the the longer dimension of the page will be oriented vertically.

Prompt for Margins

If this checkbox is checked, the user will be prompted for the margins and the paper orientation just before character sheets are printed with this template. This should be checked when you set the form size to non-standard paper sizes. For example, a template intended to be printed on 3" x 5" index cards should have this checked so that the user can specify the margins for the particular forms used to print: some forms have three cards per sheet while others have four, which would require different margins. Printing directly on index cards would require zero margins.

Form Size

If the print template is for a non-standard size, the height and width of the required form is specified here. For example, print templates intended to print on 3" x 5" cards would specify those dimensions in the **Form Height** and **Width** edit fields. If a form size is indicated, the gray rectangle is displayed in the print template to indicate the approximate printable area for the form. If no form size is indicated, the printable area is indicated for the currently selected printer.

Description

The description of this print template. This text is displayed when the print template is highlighted in the list of available print templates.

Number of Objects

The number of objects in the print template is also reported.

Set Game System

Set the name of the game system. Select an existing system from the list by double-clicking on an entry in the list, or simply type in the name of the new game system.

Get Size

Enter the font size or line thickness you wish to use for this object. You must keep the size within the allowed range.

Set Picture Reference

Picture references display graphics in the character sheet, or in the file system. Graphics files in the file system must be .BMP files (Windows bitmap files). If the files cannot be found, a rectangle is displayed where the graphics should be.

Character Sheet Picture Reference

The reference is in the form "Dialog:Picture", where Dialog is the name of the dialog which contains the picture you wish to reference, and Picture is the name of the picture control (p. 157) in that dialog. If there is no picture in the reference, the default file will be displayed (if specified).

File Name

This is useful for keeping the graphics outside the print template, which allows users to change the graphics that are displayed on the character sheet without having to edit the print template directly.

The reference is a file name reference. This can be relative to the Creator source directory (usually Creator), or an absolute path name. The resolution of the file reference is first attempted relative to Creator, then absolute. Absolute path names are discouraged because they will not work on all computers.

For example, if the reference is:

```
gamesys\logo.bmp
```

GURPS Character Builder would attempt to open C:\Program Files\GURPS\gamesys\logo.bmp and display it in the picture reference. For a real game system, you'd replace `gamesys` with the name appropriate to your game system.

The recommended procedure for this and the other reference types is to place a subdirectory in the Creator source directory for related graphics files.

Expression

To display different graphics based on values in the character sheet, choose the Expression reference type.

The reference is an expression that resolves to a string that is a file name. That file name is then resolved relative to the Creator source directory (or an absolute path name if that fails).

For example, to display different icons on the character sheet for male and female characters, create an "gamesys\icons" subdirectory in the Creator source directory. Create `male.bmp` and `female.bmp` graphics files and place them in the `gamesys\icons` directory. If your character sheet is set up so that the variable "sex" is 1 for male and 2 for female, the following reference could be used:

```
format('gamesys\icons\%s.bmp', sex=1?'male':'female')
```

Field Name Reference

To allow users to specify any graphics files without having to edit the print template, use a Field Name Reference.

The value of the reference is of the form "Dialog:Edit Text Field" where Dialog is the name of the dialog where an edit text control is placed, whose name is "Edit Text Field". Whatever value is placed in that edit text field is used as the pathname to the graphics file. For example, if you add an edit text control named LogoFile to the Configuration dialog, the picture reference would be

```
Configuration:LogoFile
```

By default the value of that edit text control would be, hypothetically, `gamesys\logo.bmp`. The user could then change that to another file name in the Configuration dialog to display a different logo, without having to change the print template itself.

Default File

The path name, relative to the Creator source directory, of the graphics file that is displayed if the primary picture reference doesn't resolve to a file that exists.

Modify Print Template Pages

The **View | Pages...** command allows you to modify the structure of your print template, adding new pages or deleting old ones. The commands available in the dialog are described below.

The names of the pages in the print template are displayed in the list. The number of objects in each page is also shown.

To view or edit another page:

- Select the **View | Next Page** or **View | Previous Page** command, or
- Select the **View | Pages...** command, click the page you want and click the **Close** button.

Add

Adds a page to the print template.

Delete

Deletes the highlighted page. You cannot delete all pages in the template. If you attempt to delete a background page that other pages reference, you will be asked if you want to delete those references as well.

Edit

Allows you to edit (p. 120) the highlighted page.

Up

Moves the highlighted page up one. Pages are printed in the order they appear in this list, from top to bottom.

Down

Moves the highlighted page down one.

See Also...

Flowing text from one page to another (p. 111)

Edit Page Info

The **Edit | Edit Page...** command brings up the Edit Page Info dialog. This dialog allows you to set the page name, type, display expression, and the background for the page.

Page Name

Enter the name of the print template page. This name will be displayed in the print template window so that you know which page you are currently editing. If there is only one page in the print template, its name is not shown. No two pages may have the same name.

When a page is a background page, the name is used by other pages to reference that page.

Page Type

The available page types are:

Normal

Normal print template pages are always printed. The initial page of the print template will usually be a normal page. Normal pages often contain text boxes that have links (p. 122) to other text boxes on continuation pages, allowing text to overflow onto another page.

Optional

An optional page is printed only if its expression evaluates to true for the character sheet being printed. For example, if you have an Equipment page that you only want printed if there are entries in the Equipment list, you could set the expression to:

```
@Equipment > 0
```

Continuation

A continuation page is printed only if a text box on another page that is printed overflows into a continuation text box on the continuation page. You must set a text link (p. 122) tag on the text box that overflows, and set the same link tag in the text box on the continuation page that the text should flow into.

When **GURPS Character Builder** prints a page, it finds all the text boxes in the template that have overflowed and remembers the link tags for them. Then it searches all continuation pages in the template for link tags that match the list just generated, counting the number of "hits" for each continuation page. The continuation page with the greatest number of hits is printed next.

If more than one page has the same number of hits, the first page found will be printed. You should therefore order the pages in the template with that in mind. To achieve minimum page count you should order continuation pages with the fewest link tags first, to give the maximum amount of space to each of the tags that are present. Click here for an example (p. 122).

Continuation pages may also specify an expression. The page is printed if the expression evaluates to true, as for an Optional page. If a continuation page has an expression, it will be printed only once, whether it was printed because the expression was true or because text overflowed on to it. The expression must always be true for the page to print. If the expression is false, the continuation page will not be considered for text overflow.

Background

A background page is printed on every page that references it as a background page. The background objects are printed first, then the objects for the referencing page are printed.

Background pages are useful for providing a common border for all printed pages, or for locating the page number in a consistent place.

Background pages may reference other background pages. If the references are circular, each background page will be printed only once. Each background page is printed before the page that references it.

If you change the name or type of a background page, you will be asked if you want to also change all the references to that page. You may not change a background page name without changing the references.

Expression

This field is active only for Continuation and Optional page types. If the expression evaluates to true for the character sheet being processed, the page will be printed.

The expression can be something as straightforward as:

```
@Powers > 0
```

which would print the page if there are any entries in the Powers list, or as complex as:

```
@notes or @`History.History` or @`Appearance.Description` or  
CountItems("Merits:Derangement", 0, 0, "cost")
```

which would print the page if there are notes in the **Utility | Notes** window, or if the History field of the History dialog contains text, or if the Description field of the Appearance dialog contains text, or if there are any items in the Merits list of category "Derangement" that have a cost of 0.

Background Page

The name of the background page to print for this page. Click the drop-down list to see the list of available background pages. Click the desired page, or (None) to have no background page.

The objects on the background page will be printed before any objects on the referencing page. Objects directly above background objects will overwrite the background objects.

If you have a common border or background for some or all pages, you should put those objects on a background page and reference it in all the pages that share that background.

You can also use background pages to reduce the amount of work needed to define a set of pages. For example, you could define a character sheet with a first page that displays a character picture if a picture is present, or extends the Powers & Skills section if there is no picture.

To do this, start with a "Main Background" page that defines all the standard objects with nothing in the area that is "shared" between the picture and Powers list extension. This should be a background page. Then define two optional pages, both of which reference the Main Background page as a background: one has the picture in the free space, and the second has a continuation text box for the Powers & Skills. If the character picture is named Picture in the Picture dialog, the expression for page that displays the picture is @`Picture.Picture`, while the expression for the Powers extension page is !@`Picture.Picture`.

This is advantageous because the complex part of the character sheet is on the Main Background: only one or two objects are defined on the optional pages. This decreases the template size and reduces the amount of work you have to do if you change the main character sheet display.

See Also...

Flowing text from one page to another (p. 111)

Setting text links (p. 122)

Continuation Example

To achieve minimum page count, you can define a number of different continuation pages to handle the common cases where text on the first page of the character sheet overflows.

Let's say the print template has the following pages:

1. Normal page with all character attributes and text boxes with text link tags for Skills & Spells, Benefits, Failings and Equipment.
2. Continuation page with a three-column Skills & Spells text box.
3. Continuation page with a half-column Benefits text box, half-column Failings and a two-column Skills & Spells text box.
4. Continuation page with a single-column Equipment text box and a two-column Skills & Spells text box.
5. Continuation page with a single-column Powers & Skills text box, half-column Benefits text box, half-column Failings and single-column Equipment text box.
6. A background page that the other pages all reference.

The following table describes which pages are printed, given the text boxes that overflow on the first page.

Overflowing Text Boxes on First Page	Pages Printed
None	1 (6)
Skills & Spells	1, 2 (6)
Skills & Spells, Equipment	1, 4 (6)
Skills & Spells, Benefits	1, 3 (6)
Skills & Spells, Failings	1, 3 (6)
Equipment	1, 4 (6)
Skills & Spells, Equipment, Benefits	1, 5 (6)
Equipment, Benefits	1, 5 (6)
Skills & Spells, Benefits, Equipment (which also overflows second page)	1, 5, 4 (6)

The choice of the particular combinations is informed by the designer's knowledge of typical characters: here, the most common text box that overflows is assumed to be Skills & Spells.

It isn't necessary to produce all possible combinations: cover the common cases, and make sure that every text box that can overflow has a continuation text box on at least one continuation page.

Edit Text Link

The **Text | Link...** command edits the text link tag for a text box. The link tag specifies a tag that is used to identify the text boxes that text may flow from and flow into.

To use text links:

- Define a text box that is **not** a continuation text box and has a particular tag.
- Define one or more text boxes that **are** continuation text boxes and have that same tag.

The continuation text boxes are usually on a different Continuation (p. 120) page (to provide a place for the overflow text to go if the current page is filled).

Tag

The tag used to identify the text box link. Text is continued from a non-continuation text box to a continuation text box with the same tag.

If a text box has no text link tag, it is not continued. If the text does not fit within the confines of the text box, it is simply discarded.

Continuation

If the **Continuation** checkbox is checked, the text box is a continuation text box. It can receive text that has overflowed from a non-continuation text box. Normally you would place a continuation text box on a Continuation (p. 120) page, giving it the same tag as a non-continuation tag on a previous Normal page.

A continuation text box (CTB) may be placed on the same page as a non-continuation text box (NCTB) with the same tag. In this case, the text will flow from the NCTB to the CTB *if* the NCTB is "below" the CTB. That is, if the NCTB is added after the CTB, or has been brought to the front (with the **Edit | Bring to Front** command).

See Also...

Flowing text from one page to another (p. 111)

Edit Page Info (p. 120)

View Menu

Pages... Allows you to select, add and delete pages (p. 119) in the print template.

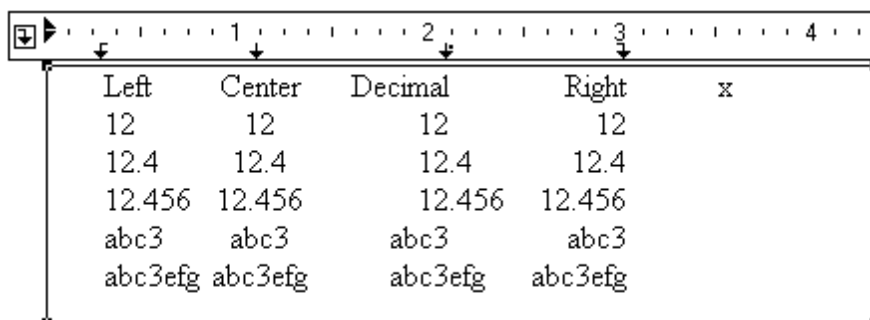
Show Rulers... When this command is checked and a text object is highlighted, a ruler window (p. 125) is displayed above it. Only one text object at a time will display a ruler.

Next Page Edit or view the next page in the print template. When the last page is displayed, the first page is considered the "next" page.

Previous Page Edit or view the previous page in the print template. When the first page is displayed, the last page is considered the "previous" page.

Ruler Window

When the **Show Rulers** command is checked in the View menu (p. 124), a ruler window will be displayed just above the highlighted text object.



The ruler window lets you set the tab stops and indentation for text objects. The **Text | Properties...** (p. 115) command allows you set some of the same properties of the text object.

Note: when you edit the text beneath the ruler, the text will not line up with the tabs until you have stopped editing the text.

The icons on the ruler have the following meanings:

- ↵ A left-aligned tab. The text after the tab is set to start at the specified tab position.
- ⇨ A right-aligned tab. The text after the tab is set to end at the specified tab position.
- ⬇ A centered tab. The text after the tab is centered about the specified tab position.
- ⇩ A decimal tab. The last digit of a number before the decimal point is set to end at the specified position. This is useful for aligning columns of numbers.
- The left indent. All lines after the first start at this position. If the first indent is less than the left indent, the first tab in the line will place the text following it to the left indent position.
- ⬄ The first line indent. The first line starts at this position.

The following are some hints to accomplish common tasks:

Adding Tabs

Click the ruler below the numbers.

Changing the type of tab to be added

Click the tab in the box on the left side of the ruler. It will change to the next tab type.

Removing tabs

Click on the tab and drag the mouse out of the ruler window.

Setting the left indent

Click and drag on the left indent (➤).

Setting the first line indent

Click and drag on the first line indent (⬄).

Replacing one tab with another

Click and drag one tab and drop it on the tab to be replaced.

Editing Binary Data Sheets

Binary data sheets store all the items that are available for selection in the character sheets. Binary data sheets are an older format; data sheets are currently recommended for data input because of the simpler, table-oriented format.

Data sheets are arranged hierarchically: at the topmost level are the Categories. These are lists of items that are categorized according to function. Categories might include Skills, Advantages, Possessions, Disadvantages, Powers, Spells, etc. Categories can in turn contain sublists, breaking down Skills into Combat Skills, Social Skills, Thief Skills, for example.


You must save a data sheet before you can use the items that you have added. When the data sheet is saved, all items currently in the master item selection originally from the data sheet will be removed and the new set of items will be loaded.

Items and lists in data sheets are always kept in alphabetical order for easy reference.

Adding a new category

Select the **Tools | Insert Sublist** command. In an empty data sheet this will add a new category. If categories are already defined, make sure all categories are closed and then select a category at the highest level, then choose **Tools | Insert Sublist...**

Opening a List or Category

Double-click the  icon for the list, or select **Tools | Open/Close Sublist**.

Closing a List or Category

Double-click the  icon, or select **Tools | Open/Close Sublist**.

Changing the Name of a List or Category

Double click the name of the list or category, or select **Tools | Modify Item**.

Adding a New Item

Open the category or sublist you wish the item to be a member of, and select the **Tools | New Item** command (or press the Ins key).

Changing an Item

Double-click the item, or select **Tools | Modify Item...** (p. 75).

Deleting Items

Select the item and select the **Tools | Delete Item** command, or press the Del key. You can also use the clipboard to cut, copy and paste.

Editing Options

Select **Modify | Options....** Options can be used to modify the cost of items, provide extra information, etc.

Set the Copyright Notice

Select **Modify | Info...** (p. 126) to change copyright and other information.

Data Sheet Info

This gives information about the data sheet.

Title

The title displayed in the dialog box that is opened while the data sheet is being loaded. If omitted, no title is displayed.

Copyright Notice

The copyright notice for the data in the data sheet. You should make sure that you properly credit the copyright holder for any game system data you enter in new data sheets. If no copyright message is specified, none will appear.

Level

The "release level" for the data sheet. This should match the levels of the character sheet and print template for the same game system. If significant changes are made to the definitions of functions in the character sheet, this level should be changed to match the character sheet. Then, if you attempt to load an old character sheet with this data sheet loaded, you will be notified of the mismatch.

Logo File



The name of the logo file to display when loading the data sheet. These files normally have a .LGO extension. They are normal BMP files as saved by the Paintbrush application.

Help File

The name of the help file to search when getting Item Info.

Data Sheet Options

The Data Sheet Options are available for selection in the Item Edit dialog (p. 68) via the New button (p. 70). You should add commonly used options to this list so that they are readily available in the Item Edit dialog.

To open a sublist (indicated by , double-click it or highlight it and click Open). To close the sublist, double-click the .

Open

Highlight an entry in the list and click Open to open or close the option or sublist. Double-clicking the entry also opens it.

Edit

Highlight an entry in the list and click Edit to change its name or other information.

New

Creates (p. 92) a new option or sublist. The new entry is added after the highlighted entry.

Delete

Deletes the highlighted entry.

Cut

Cuts the highlighted entry to the clipboard.

Copy

Copies the highlighted entry to the clipboard.

Paste

Pastes the contents of the clipboard into the option list after the highlighted entry.

Data Sheet Sublists

Enter the new name of the sublist. Sublists are always maintained in alphabetical order.

When a data sheet is read in, all items from sublists with the same name are read into a single sublist with that name.

Merging Data Sheets

Items and options from different data sheets can be combined into another character sheet by merging them.

First open the character sheet into which you wish to merge another character sheet. Select the **File | Merge Data Sheet** command. Then select the data sheet you wish to merge and click OK.

If there are any conflicts (p. 128) (that is, items or options with the same name and different values in one or more fields), you will be notified and allowed to choose which one (or both) item or option to keep.

Save Data Sheet As

Allows you to save the data sheet under a different file name.

Edit Option List

You can change the name of the option list by simply entering the new name. When the options are read in from the data sheet, all the options in sublists with the same name will be combined into one sublist.

Merge Conflict

When you merge two data sheets, there may be conflicts if an item in the original data sheet has the same name as an item in the data sheet you are merging. You have six options:

Keep Original

Keeps the item in the original data sheet and discards the item being merged.

Keep All Originals

Does the same as **Keep Original** for the current item, and will do the same thing for all other merge conflicts without asking you again.

Overwrite

Discards the item in the original data sheet and adds the item being merged to the data sheet.

Overwrite All

Does the same as **Overwrite**, and will do the same thing for all other merge conflicts without asking you again.

Keep Both

Adds the item that caused the conflict and retains the original item.


Keep All

Does the same as **Keep Both**, and will add the rest of the items in the merged data sheet without asking you.

How To...

This section contains information on how to accomplish a wide variety of tasks using ***GURPS Character Builder***.

How to Create a New Character

- Select the **New...** command from the **File** menu. You can also just click  on the button bar.
- In the New File dialog, select the game system for the character sheet by clicking the Game System drop-down list and clicking the desired system.
- Make sure the File Type is Character Sheet.
- Click the character sheet template that you want to use to create your character.
- Click the **OK** button.
- The main window of your new character will soon appear. If the game system offers help specific to it, you can press SHIFT+F1 to get more help on that game system. Otherwise, pressing F1 will give you generic help for using **GURPS Character Builder**.

How to Create a New Game System

For a complete implementation of a game system in **GURPS Character Builder**, you must create the character sheet template (p. 132), the data sheet (p. 134), at least one print template (p. 135) and one or more Text Filters (p. 135).

The best way to learn how to implement a new game system is to use an existing one as an example. An example character sheet and support files are contained in `Example.cst`, `Example.prt`, etc. To examine the example, create a new character sheet and select `Example.cst` as the template. Once in the character sheet, press SHIFT+F1 for more information about how the example is constructed.

Quick Links

[Creating Character Sheet Templates \(p. 132\)](#)
[Conversion Script Format \(p. 262\)](#)
[Expression Syntax \(p. 169\)](#)
[Filter Files \(p. 185\)](#)
[Creating Data Sheets \(p. 134\)](#)
[Data Sheet Text Format \(p. 211\)](#)
[Print Templates \(p. 109\)](#)

How to Convert Character Sheets

There are two ways to accomplish character sheet conversion. Both require that a conversion script (p. 262) be written, if you are converting between game systems.

Single File Conversion

- Open the character sheet to be converted.
- Select the **File | Convert...** command.
- Click the new character sheet template to use for the converted character.
- Click the conversion script to use for the conversion.
- Click **OK**.

Multiple File Conversion

- Select the **Utilities | Update Character Sheets...** command.
- Click the **Select Files...** button to choose the files to convert.
- Click the **Template** dropdown list, then choose the template to use for the converted character sheets.
- Click the **Script** dropdown list to select the conversion script to use.
- Click the **Browse** button to select the destination directory for the converted character sheets.
- Check the **Compare After Update** button if the conversion script has the ability to compare the character sheet totals.
- Click **Update**.

How to Update Character Sheets to New Versions

When you get a new version of **GURPS Character Builder**, you may wish to update old character sheets to the new version. This has several advantages: the items are updated to the versions in the newer data sheets; you can take advantage of any new features of the new character sheet template. Click here for some tips (p. 131) before you update your character sheet.

To update the character sheet:

- 1) Open the original character sheet.
- 2) Make sure that you have loaded all data sheets that you drew items from. A good way to do this is to use the **Modify | Character Sheet Info...** command to indicate which data sheets should be loaded.
- 2) Select the **File | Convert...** command. The Convert Character Sheet dialog will come up.
- 3) The Template selected should be the original template. Make sure the Conversion Script is (None).
- 4) Click OK. A new character sheet will be opened (named Untitled) and the values will bounce around while the character sheet is being updated.
- 5) Examine everything to make sure that it converted according to your desires.

You can also use the **Utilities | Update Character Sheets...** (p. 17) command to automatically update many character sheets at once.

Character Sheet Update Tips



When converting character sheets, it is important that all items in your character sheet have the correct "Original Name (p. 75)." If you don't want the item to be converted, set the Original Name to nothing.

You can also prevent items from being converted by setting the Generic Item (p. 76) flag

If you have changed something intrinsic about your character sheet (basic variables, arrays, formulas, positioning of edit fields in dialogs, added new dialogs, etc.), you will lose those changes when you update to the new version of the character sheet template.



How to Change Print Templates

If you want to change the appearance of the printed character sheet (for example, to remove the character's picture to make room for another list), you need to change the Print Template for the game system. It's probably best to make a new version of the Template and keep the original around for reference. Follow these steps.

1. Open the Print Template. Select **File | Open**, then choose Print Template in the List Files of Type list. Pick the template you want to modify.
2. Save the file under a new name. Select **File | Save As** and save the template with the new name.
3. To delete an object, click on it and select the **Edit | Clear** command (or click , or press the Del key).
4. To add a new object, select the appropriate command from the **Tools** menu. For example, if you want to add a text box that lists all items in the Skills list, you would select **Tools | Text Box** (or click ). More detailed help is available on editing print templates (p. 109).
5. Size the text box by clicking the handles (the black squares on the corners of the box) and dragging.
6. Add text to it by double-clicking the text box. The following will show the contents of the Skills list:

```
@foreach(skill)
  %@name%: %@level%      %@cost%
@endfor
```

Note that a Text object will *not* be processed for commands, while a Text Box will be. If the commands appear in the final printed character sheet, you probably have selected a Text object instead of a Text Box object.

7. Add other objects (p. 109) as desired.
8. Save the Print Template. You cannot use the modified template until it has been saved.
9. Create a character sheet (with **File | New...**) and set its print template to the one you just saved with the **File | Set Copy/Print Filters...** command.
10. Preview the character sheet on the screen with **File | Print Preview | Through Template...** (or click ). Continue the cycle above (changing, saving, previewing) until the character sheet looks the way you want. Then print with the **File | Print | Through Template...** command (or click .

How to Create a Character Template

Creating a character sheet (p. 298) is the first step in implementing a new game system in **GURPS Character Builder**. All the other files will depend on how you have set up the character sheet template (p. 298). You can base a new character template on an existing one, but be aware that if you change the original, the new template will not track any changes -- you must explicitly update character sheets (p. 130) to the new format.

The following steps show how to create the template:


1. Select the **File | New** command. Click the Character Template radio button and then click the character template that you wish to base the new template on. If you want to start from scratch, click on BLANK.CST. Click OK to create the new character template.
2. You now have a blank main dialog. You can change the title of the dialog, the font used in the dialog and other information by selecting the **Window Information...** (p. 41) command from the **Modify** menu. A good font to use is 8-point MS Sans Serif.
3. Add controls (p. 155) to the main dialog. Select the **Modify | Dialog Item Properties** command to enter the mode which allows you to add controls. Select it again to exit that mode and edit the character sheet normally. Click here for more information (p. 132).
4. If you need to define variables, functions or arrays to support your definitions in your dialogs, use the **Modify | Defines...** command to bring up the Modify Defines (p. 166) dialog.
5. Add any other dialogs you need: the **Modify | Data Menu...** command brings up a dialog that lets you modify the Data menu (p. 41). A common dialog is the Information dialog, which has such things as your name, physical appearance, height, weight, etc. You can add a picture dialog, or a dialog summarizing combat values, etc.
6. Add lists. Use the Modify Data Menu (p. 41) dialog to add lists.
7. Set the Game System: select **Modify | Character Sheet Info...** (p. 45) and the Edit Character Sheet Info dialog will appear. Type the name of the game system you are creating the template for (or select it from the list). All the files you're creating will depend on the game system being set to the same value. After you've created the data sheet for the game system, you'll need to come back to this dialog to add any data sheets that will be automatically loaded each time you open the character template, or create a character sheet based on the template.

How to Add Controls to a Dialog


Most main dialogs contain the character's major attributes (statistics), totals of character points, etc. To add text labels to the dialog, use the **Control | Text** (p. 155) command. To add editable variables, use

Control | Edit Variable (p. 155). This creates a text edit box into which you can enter numbers. The variable associated with the edit box is assigned the number the user enters. Use the **Control | Formula** (p. 155) to add a computed value.

Add Text Label

For example, to add a Strength attribute, add a text object and set the text to "Strength". Do this by selecting **Control | Text** (or the  button) and clicking somewhere within the dialog. A gray box with the string "Text" inside it will be deposited at the place you clicked. Now double-click that to bring up the Edit Text Item dialog. Change the text and alignment as suits your purposes. You can change the size of the label by clicking and dragging on the little blank squares on the corners of the text label.

Add Edit Variable

Now add the Edit Variable that will contain your strength value. Select **Control | Edit Variable** (or click ) and click within the dialog. Double-click the edit box created. In the Edit Variable dialog change the name of the variable to STRENGTH. Whenever the user changes the value in this edit box, the variable STRENGTH will be set to the value the user entered.

Add Default Value

You can set a default value for the variable, which will be the value that the variable is set to whenever the variable is reset (p. 33). This default value can be computed from other attributes (which might be the case for a game system such as Champions®, which computes attributes such as PD from STR), or you can use a random number function (p. 171) to generate attributes randomly. In that case, you can "reroll" the character by selecting the **Reset All Fields** command.

Add Check Expression and Message

You can also set a minimum or maximum value by putting an expression in the Check expression field. The name "x" takes the place of the variable you're checking in the check expression. For example, if you wanted to limit the strength to be greater than 2 and less than 19, you could enter

$$x > 2 \text{ and } x < 19$$

or, alternately,

$$x \geq 3 \text{ and } x \leq 18$$


You can also enter a message to be displayed if the check expression fails. If you omit the message, a generic message is displayed ("The value you entered is not in the acceptable range").

Force Constant

If you randomly generate values for an attribute, you will want those values to be constant after the initial generation. Check Force Constant in this case. Otherwise, every time you open the character sheet a new value will be generated.

Add the Cost Formula

If you're creating a template for a game system that charges character points for attributes, select the

Control | Formula (or click ) command and add a formula. Double-click the new text. Set the name of the variable to c_strength and write the formula. If the cost formula is at all complex, you should create a function that computes it. If each point of strength over 10 costs 1 point (and each point below gives you back 1 point), then your cost formula would be

$$\text{strength}-10$$

You can also use the conditional operator (p. 169) to prevent costs for illegal values, if strength below 1 is not possible:

$$\text{strength}>0?\text{strength}-10:0$$

Now switch back to normal mode (select **Modify | Dialog Item Properties** to uncheck the command) and enter a new value in the edit field: the number in the cost formula will automatically to reflect the new cost.

How to Create Data Sheets

GURPS Character Builder provides two ways to create data sheets: data sheets (p. 144), and the old-style binary data sheets that require you to edit items one-by-one.

New-style (macro) data sheets have the following advantages:

- The macros provide a pre-built framework that allows you to enter new items easily.
- A table-oriented interface, similar to spreadsheets, greatly speeds data entry.
- When the definition of an item changes, all the items using that definition change the next time the data sheet is loaded.



Topics

Creating and editing data sheets (p. 144)

Creating and editing old-style binary data sheets (p. 134)

Old-Style Data Sheets

Follow these steps to create an old-style binary data sheet (p. 298).

1. Choose the **File | New** command. In the New File dialog (p. 2) click the **Binary Data Sheet** radio button. Select the name of the game system (p. 298) to be used with the data sheet, or if it is a new type, enter the name with the keyboard.
2. Add categories to the empty data sheet. Select the **Tools | Insert Sublist...** (p. 126) command. Enter the name of your first category. Category names that are commonly used include Skills, Advantages, Possessions, etc. Add as many main categories as your game system requires.
3. Add sublists (subcategories). If your item categories contain sublists, open a category (double-click on the  icon) and choose the **Tools | Insert Sublist...** command again. Name your subcategories. In Skills you might have Combat, Animal, Scientific, etc. In your possessions category you might have Armor, Pistols, etc. In a Spells category you might have the different colleges of magic be the subcategories.
4. Add items to the categories. Open the desired sublist (double-click the  icon) and press the Ins key. This brings up the Edit Properties dialog (p. 75). Click here for more details on creating items (p. 134).
5. To reduce the amount of work, use **Edit | Copy** and **Edit | Paste** to make copies of items that you make as a template, and after pasting select the pasted item and change it as required. This can be less work than doing everything from scratch for every item.

You can also create data sheets as text files (p. 211), which can be edited in any standard text editor. This allows you to do large amounts of data entry. **Important Note:** If you create a data sheet with a text editor, you should *not* edit it with **GURPS Character Builder**. All your macros and text will be lost when you save the data sheet from **GURPS Character Builder**.

How to Add Items

The first thing to do is set the item name. Then you need to consider how the cost will be set. Generally, item costs are figured in three basic different ways:

Constant cost

These items have a fixed cost, and no "level" associated with them. Benefits such as Night Vision, Bump of Direction, Combat Reflexes/Combat Sense, etc. have a constant cost. Disadvantages often have a constant cost. Possessions might have a constant cost, depending on how you define them (their cost might be their weight or monetary price).

For constant cost items, simply set the required cost in the **Cost** field.

Fixed cost per level

These items have a fixed cost per level. Benefits are often defined this way. For example, you might define Flight as costing 2 points per level, where each level allows your character to fly 2 yards per turn.

Heightened Alertness is another example: each level might cost 8 points and adds 1 to all your sense rolls.

For fixed cost per level items, define a cost expression of the form " $x*2$ " to indicate the cost is 2 per level, or " $x*5$ " for 5 points per level, etc.

Arbitrary cost based on level

Some game systems use more complex, non-linear functions to determine the cost of a level-based item. The best way to handle this is to define a function in the character template, and then reference that function. In this case, the Cost Expression will be a reference to the function with whatever arguments are necessary.

For example, if the first level costs 3 points and all subsequent levels cost 2 points, the cost formula would be " $3+(x-1)*2$ ". If the minimum cost is 1, you would also want to add a Check Expression of " $x>0$ ".

For items that have a non-linear but fixed number of levels (say, Magic Ability that costs 10, 20 and 40 points and is available in only those three levels), you might define an array (p. 167), `magicCost`, via the **Modify | Defines...** command. Set the appropriate values and set the Cost Expression for the item to "`magicCost[x]`".

Costs based on attributes


Some game systems base skills on other attributes, such as your intelligence or dexterity. For the same cost, a higher underlying attribute results in a higher skill level.

To make the skill level track the underlying attribute, its cost must be defined in terms of the attribute. Furthermore, a variable name must be defined for the item so that it will track the changes in the attribute it depends on.

For example, if the cost of a skill is 2 for every point above intelligence, the cost expression would be " $(x-intelligence)*2$ ". To make the skill level track changes in the underlying attribute, you can set the Check Expression to " $c>=0$ ".

How to Create Print Templates

If you have an existing template that could be modified, it's easiest to start by changing that print template (p. 131). If no print template is an appropriate starting point, follow these steps.

1. Select the **File | New...** command (or click ,).
2. Click the Print Template radio button and click OK.
3. Set the game system by selecting one from the list, or type in the name if the one you want isn't there.
4. Add objects (p. 109) by clicking the appropriate buttons in the button bar, or selecting the commands from the **Tools** menu. The instructions for modifying a print template (p. 131) will apply now.
5. If you're making a print template that you intend to be used by others, be careful that you don't draw things too close to the edge of the paper. The gray lines around the edge of the window indicate the paper size of *the currently selected printer* -- if you draw too close to the edge the character sheet may not print in its entirety on printers that are smaller.

Similarly, if you live in the United States and intend for the print template to be used in other countries, realize that paper in Europe (known as A4), for example, is taller and narrower than paper in the US.

How to Create Text Filters

Text Filters (p. 298) are used to print text-only versions of the character sheet and to copy the character sheet to the clipboard. For more detailed information on creating filter files, click here (p. 185)

Follow these steps to create your own text filter.

1. Copy the example below by selecting the **Edit | Copy** command in Windows Help. In the Copy dialog, scroll to the end and highlight all the text from the line "@gamesystem Example" to "@endfor".

```
@gamesystem Example
Name: %@info.name%    Point Total: %total%
Story: %@info.story%
Appearance: %@info.appearance%

STRN %strength%  SIZE %size%
DEXT %dexterity% AGIL %agility%
INTL %intelligence%  WILL %will%
HEAL %health%    HITS %hits%

@if @notes
Notes: %@notes%
@end
Height: %@info.height%    Weight: %@info.weight%
@if @skills
Skills: (%skills%)
\
@foreach(Skill,, )
%@name%: %@level% [%@cost%]\
@endfor
```

2. Create a new text file: Select the **File | New...** command, click the **Text File** button and click **OK**.
3. Change "Example" following @gamesystem to the name of the game system you're creating the filter for.
4. Reference variables (p. 185) defined in your character sheet by enclosing the variable name in percent signs: "%strength%", for example. These can be defined as Edit Variables in dialogs.
5. Reference text fields in dialogs (p. 186) by enclosing in percent signs the name of the dialog, preceded by an '@' sign and following by a '.', then the name of the edit field you wish to display: for example, "%@info.name%" displays the "name" edit field located in the "Information" dialog. You can abbreviate dialog names by simple shortening of the name.
6. Reference the contents of lists (p. 194) in the following fashion:

```
@foreach(Skill,, )
%@name%: %@level% [%@cost%]\
@endfor
```

This example lists all skills together on the same line, separated by a comma and a space. If you want each skill to be indented on a line by itself, you could use the following:

```
@foreach(Skill)
    %@name%: %@level% [%@cost%]
@endfor
```

7. When the filter is complete, save the file (use an .flt extension) in the directory where you installed **GURPS Character Builder** (usually C:\Program Files\GURPS).
8. Create a new character sheet and use the **File | Set Copy/Print Filters...** (p. 5) command to set the Copy Filter to the newly created filter file.
9. Use the **Edit | Filter Copy** (p. 9) command to copy the character sheet to the clipboard. If you made mistakes in writing the filter, you'll get a warning dialog telling what was wrong. Go back to the editor and fix the problem, and try again.

10. If you wish to use the filter file for printing, you can set the text parameters (font, font size, tab stops, etc.) with the **File | Page Setup...** (p. 7) command. *Make sure* you close the filter file in your editor before you do this. Page Setup will modify the filter file, writing some extra commands at the top of the file that record the selections you made.

How to Add Campaign-Specific Items

For your own campaigns you may have a set of items that you wish to have available for selection at all times. The best way to do this is to:

1. Create your own data sheet (p. 134) or load a character sheet as data (p. 137). Add all the items for your campaign to it. Save it in the source directory (p. 17).
2. Make a new character sheet template (p. 2) of your own by selecting **File | New...**, click Character Template and click OK.
3. Select **Modify | Character Sheet Info...** and add your data sheet to the list of data sheets (p. 45) for the template.
4. Save the template with a new name.

Whenever you create a new character sheet for this campaign, use this new template for it. Your campaign data sheet will be automatically loaded.

You can also load the data sheet manually if you don't want to create a new character template. You could change the original character template to load the data sheet, but when you upgrade to the next version of the game system module your changes would be overwritten and you would lose them.

Loading a Character Sheet as Data

Customized items can be loaded from character sheets and be made available for selection in other character sheets. You can do this by loading character sheets as data.

Creating the Character Sheet Data File

- Add the desired items to a character sheet.
- Save that character sheet in the source directory. This is usually the `C:\Program Files\GURPS` directory, but if you installed **GURPS Character Builder** in another location it will be different. To see what the source directory is set to, select the **Utilities | Preferences...** command.
- If you have several custom items you should add them all to the same character sheet. Use copy and paste to collect the items from other character sheets and consolidate them into a single character sheet that is loaded as data.

Loading the Character Sheet as Data All the Time

- After the character sheet has been saved in the source directory, select the **File | Load Data Sheet...** command.
- Find the character sheet you saved in the list of Data Sheet Files.
- Highlight the file by clicking on it.
- Click the **Load Now >>** button. The character sheet will move to the Loaded Data Sheets list.
- Click the **Load Always** button. The character sheet will also appear in the **Always Loaded** list.

The items in that character sheet will not be available for selection in other character sheets. Henceforth that character sheet's data will be loaded each time you start **GURPS Character Builder**. If you omit the final step, the items will be loaded only until you leave **GURPS Character Builder**, and will not be loaded the next time the application starts.

Locating Items in Sublists

Items can be located in sublists. For example, you can make a new handgun appear in the Handguns sublist under Weapons:

- Open the appropriate list window. For example, Equipment.
- Select the **Tools | Insert Sublist...** command.

- In the Edit Sublist Information dialog enter the name of the sublist. For example, Weapons.
- Click the Open List checkbox to cause the sublist to be open.
- To add a sublist to that sublist, select the **Tools | Insert Sublist...** command again.
- In the Edit Sublist Information dialog enter the name of the sublist. For example, Handguns.
- Click the Open List checkbox to cause the sublist to be open.
- When you've created the desired nested sublists, insert the desired item into the sublist.
- Save the character sheet in the source directory as above and load the character sheet as data.

Updating the Items in the Character Sheet

If you update the game system to a new level (p. 258), you should also convert the character sheet you're loading as data.

- Open the character sheet in the source directory.
- Select the **File | Convert...** command. Follow these instructions (p. 258) for conversion. The conversion will require the data loaded from the character sheet to be unloaded.
- Close the original character sheet.
- Save the converted file in the source directory over the old file.
- Close the new character sheet.
- Select the **File | Load Data Sheet...** command.
- Highlight the character sheet.
- Click the **Load Now** button.
- Close the dialog.



The updated items will now be available to all other character sheets using that game system.

Adding a Sublist and All Its Children

Some game systems have the idea of "Frameworks," "Package Deals" or "Occupations" that provide a list of preselected items that are added within a sublist. Frequently this can be handled by defining an item with a list of automatic items (p. 79), but sometimes it's easier to build such as list by example. The problem is that a sublist appearing in the available items list cannot be added -- that is, unless it is marked properly.

To mark a sublist so that it and its children can be added when selected in the Available Items list:

- Highlight the desired item.
- Select the **Tools | Edit Properties...** command (or press ALT+ENTER).
- Click the **Flags** tab.
- Check the **Add Sublist and Contents** checkbox.
- Click **OK** to close the dialog.

If you load the character sheet containing the item as data, it will appear in the Available Items list with  instead of .

How to Generate Random or Complex Characters

Sometimes a game system requires a complex series of decisions or random selections to create a new character. **GURPS Character Builder** provides a "scripting" capability to accommodate this.

This process requires three files: the "initial" character sheet template, the "genchar" script, and the "genchar" template. If you don't have all these files, see below for directions on creating them.

1. Choose the **File | New...** command and select the initial character template.
2. Fill in the required information in the initial character sheet.
3. Choose the **File | Generate Character** (p. 261) command to generate the character. The new character sheet will be created based on the Genchar script specified in the initial character sheet.

The following are instructions for creating the necessary files for the Character Generation process.

1. Create (p. 132) the Genchar Template, which will be the final character sheet produced. The data entered in the initial character template will be processed by the genchar script and placed in a new character sheet.
2. Create the Genchar Script. This file has exactly the same format as a conversion script (p. 262). Make sure the correct game systems are specified in the source and destination.
3. Create the initial character template (p. 132) for the initial character sheet. This should contain the basic parameters for your character generation process. For example, you might specify your character class, sex, level, etc.
 - Choose the **Modify | Character Sheet Info...** (p. 45) command and set the Genchar Template and Genchar Script to the names of those files you just created.

How to Create Custom Screen Layouts

Creator remembers the size and locations of character sheet windows. If a character sheet template has open windows, it will use that layout when new character sheets are created.

To make a character sheet template with a custom screen layout:

- Make a copy of the base character sheet: for example, open `Charsheet.cst` and save it as `MyCharsheet.cst`.
- Open the windows and position them as desired.
- Save the character sheet template.
- When creating character sheets that use this layout, use `MyCharsheet.cst`.

When you get a new release of the game system, update your custom layout character sheet template to the new version.

Group Files

A group file contains a list of character sheets that you work together with as a group. Group files usually have either a `.group` or `.clist` extension.

Group files are typically used to list the character sheets for the player characters, or for the non-player characters used in an adventure or scenario. Group files can be used for the following:

- Print all the character sheets for the characters in the group.
- Print character combat summaries.
- Copy the text of the characters to the clipboard using one of the text filters, so that you can paste that text into a word processing document.
- Save the text of character sheets in the group as text into a text file, for display on the web, or to create special data sheets for character templates with certain game systems such as **GURPS**.

Topics

Creating Group Files (p. 140)
Working with the Group File Window (p. 140)
Group File Information (p. 141)
Adding Character Sheets to the Group (p. 141)
Printing Character Groups (p. 142)
Previewing Text Printing of Character Groups (p. 142)
Printing the Text of Character Groups (p. 142)
Saving Groups as Text Files (p. 143)
Copying the Text of Character Groups to the Clipboard (p. 143)
Saving the Text of Character Groups to Files (p. 143)

Creating Group Files

To create a group file:

- Select the **File | New...** command.
- Click the **Group** radio button.
- Click the **Game System** dropdown list to select the game system that you wish to use for this group. If you leave the **Game System** set to (All) no game system will be selected. Since most groups of character must have the same game system to use the group functions (copying text, printing through print templates), you will probably always want to select a specific game system.
- Select the character sheet template you wish to use when creating new character sheets in this group. You won't be limited to characters using this template; this will just be the "default" template used when using the **File | Add New File...** command.
- Click the **OK** button to create the group file.

Working with the Group Window

Character sheets added to the group will be sorted by file name. Only the name part of the file path is normally displayed.

Context Menu and Shortcuts

To bring up a context menu for the group, right click in the group window. If the Shortcuts window (p. 67) is open, commands for working with groups are available under the Group heading.

Adding Files to the Group

Select the **Tools | Add Files...** (p. 141) command to add files that already exist (pressing the INS key also works), or **File | Add New File...** (p. 141) to create a new character sheet and add it to the group.

Opening Character Sheets

Highlight one or more entries and select the **Tools | Open Files** command. The selected character sheets will be opened in **GURPS Character Builder**.

You can also double-click an entry to open that character sheet.

If the character sheet is already open, the main window will be brought to the front.

Highlighting Files

Click an entry to highlight it. Hold down the SHIFT key to click and extend the selection to all entries between the highlighted entry and the new entry. Hold down the CTRL key and click an entry to add that entry to the selection. You can also click and drag to select multiple entries.

Removing Entries

Highlight the desired entries and select **Edit | Clear**. Pressing the DEL key also works. This will not delete the character sheet file, it will only remove the reference to it from the group.

Working with the Clipboard

Highlight the desired entries. You can then use **Edit | Cut** and **Edit | Copy** to place those entries on the clipboard. They can be pasted into other groups with **Edit | Paste**.

Renaming Character Sheets

To rename an entry and the character sheet it refers to, use the **Tools | Rename** (p. 142) command.

Seeing the Full Path Name

Only the file name part of the character sheet is displayed. To see the full path name of the character sheet file, hold the cursor motionless above the file name for a second. A tip window with the full path name will appear.

Group File Info

The group file has several pieces of information associated with it to indicate your preferences when working with the group. Select the **Modify | Info...** command to open the Info dialog.

Game System

The game system indicates which game system character sheets in this group will use. Set this to (None) to indicate no preference.

Template

The character sheet template selected here is used as the template when you create a character sheet with the **File | Add New File...** command. Set the character template to (None) to indicate that you wish to choose it when adding a new file.

Preferred Filter

Choosing a filter here preselects the filter when you select the **Edit | Filter Copy...**, **File | Save Group Through Filter...**, **File | Print | Through Filter...** and **File | Print Preview** commands. Select (None) if you wish to choose the filter when you select these commands.

Preferred Print Template

Choosing a print template preselects the print template when you select the **File | Print | Through Template...** command. Select (None) if you wish to choose the print template when you select this command.

Adding Character Sheets to the Group

There are two ways to add character sheets to the group:

Adding Existing Files to the Group

- Select the **Tools | Add Files...** command. You can also press INS.
- Find the files you wish to add to the group with the Browser dialog.
- To select multiple files click file names and hold down the CTRL or SHIFT keys.
- Click **OK** when the desired files have been selected.

An entry for the character sheet will be added to the group in alphabetical order.

Adding New Files to the Group

- Select the **File | Add New File...** command.

- If no character template is selected for the group, select the desired template.
- Enter the name of the new character sheet in the dialog.
- Click **Save**.

The file will be created in the folder with the specified name. An entry for the new character sheet will also be added to the group in alphabetical order.

Renaming Character Sheets

To rename a character sheet listed in a group:

- Highlight the entry that you wish to rename.
- Select the **Tools | Rename...** command.
- When the Rename File document opens, use the keyboard to change the file name. You may change the folder where the file is located; the file will be moved.
- Click the **Save** button once you have set the file's name and location.

Files may not be renamed on to other drives -- renaming must take place on the same device.

Printing Character Sheets for Groups of Characters

To print the characters in a group file through a print template:

- Open the group file.
- Select the **File | Print | Through Template...** command.
- The Group Print dialog will appear.
- Click the desired print template. Click the first entry (Use Default Print Templates) to use the print template associated with each character sheet.
- Use the **Select Files...** and **Remove** buttons to add or remove character sheets.
- Click the **Print** button.
- Use the Print dialog to select printer.
- Click the **OK** button.
- The character sheets will be printed, reading each character sheet in turn.

Previewing Printing the Text of Character Groups

To preview the text of characters in a group file:

- Open the group file.
- Select the **File | Print Preview...** command.
- The Group Print Preview dialog will appear.
- Click the filter to use to produce the desired output.
- Use the **Select Files...** and **Remove** buttons to add or remove character sheets.
- Click the **Preview** button.
- The text of the characters will be generated, reading each character sheet in turn, in the order indicated in the filter.
- The Print Preview dialog (p. 7) will appear, displaying the text of the character sheets.

Printing the Text of a Character Group

To print the text of characters in a group file:

- Open the group file.
- Select the **File | Print | Through Filter...** command.
- The Group Print dialog will appear.
- Click the filter to use to produce the desired output.
- Use the **Select Files...** and **Remove** buttons to add or remove character sheets.
- Click the **Print** button.
- Use the Print dialog to set parameters for printing (select printer, etc).

- Click the **OK** button.
- The text of the characters will be printed, reading each character sheet in turn, in the order indicated in the filter.

Copying the Text of Character Sheet Groups to the Clipboard

To copy the text of characters in a group file to the clipboard:

- Open the group file.
- Select the **Edit | Filter Copy...** command.
- The Group Filter Coyp dialog will appear.
- Click the filter to use to produce the desired output.
- Use the **Select Files...** and **Remove** buttons to add or remove character sheets.
- Click the **Copy** button.
- The text of the character sheets will be copied to the clipboard, reading each character sheet in turn, in the order indicated in the filter.

Saving the Text of Group Characters

To save the text of characters in a group file to a file on disk:

- Open the group file.
- Select the **File | Save Group through Filter...** command.
- The Group Filter Save dialog will appear.
- Click the filter to use to produce the desired output.
- Use the **Select Files...** and **Remove** buttons to add or remove character sheets.
- Click the **Save** button.
- Use the Save Filtered Text As dialog to indicate the location where the text of the characters will be saved.
- Click the **Save** button.
- The text file will be produced, reading each character sheet in turn, in the order indicated in the filter.

Save Group File As

Enter the name that you wish to save the group file as.

If you save the group file in another folder, the references to the character sheets in the group will be made relative to the group file's new location. Generally, it is best to keep the files in a group in the same folder as the group file, or in a subfolder of the folder where the group file is located.

Data Sheets

A data sheet (also called a macro data sheet) is a text file that contains macro invocations that define items. ***GURPS Character Builder*** provides a table-oriented graphical user interface for performing item data entry.

A "macro" defines a set of instructions for building an item. The arguments to macros provide the data that customizes each item, specifying such things as the item name, level, category, etc.

Macros are the best way to define items, because if the underlying definition of the item changes, only the macro needs to change -- all items defined with the macro will change along with the macro definition. If you edit data sheets directly, you must manually change each item.

Macro data sheets implement a limited subset of data sheet commands. The major difference is that macros are not expanded when a macro data sheet is read for editing -- the macro definitions are read verbatim as text. When the macro data sheet is written back out, the macro invocations are preserved. When macro data sheets are loaded as data, they are treated the same way as standard data sheets.

Macro data sheets are text files with specific formatting. You should always use ***GURPS Character Builder*** to edit macro data sheets, unless you really know what you're doing. If you make the wrong change, ***GURPS Character Builder*** will no longer be able to read the macro data sheet for editing, and you will have to edit the file with a text editor.

Once you have created a macro data sheet you can load it as data (with the **File | Load Data Sheet...** (p. 4) command) to make the items defined in it available to character sheets.

You cannot edit a standard data sheet (.cds file) as a macro data sheet. The .cds format is a superset of the macro data sheet format.

Topics

- Creating a Data Sheet (p. 144)
- The Data Sheet Window (p. 145)
- Data Sheet Information (p. 146)
- Adding Categories (p. 147)
- Adding Sublists (p. 147)
- Adding and Editing Items (p. 148)
- Adding Options (p. 149)
- Adding Comments and Text Lines (p. 149)
- Headers (p. 149)
- Deleting Items (p. 149)
- Merging Data Sheets (p. 150)
- Trying Out the Data Sheet (p. 150)
- Converting .cds files to .mds files (p. 150)

Creating a Data Sheet

Following these steps to create a data sheet:

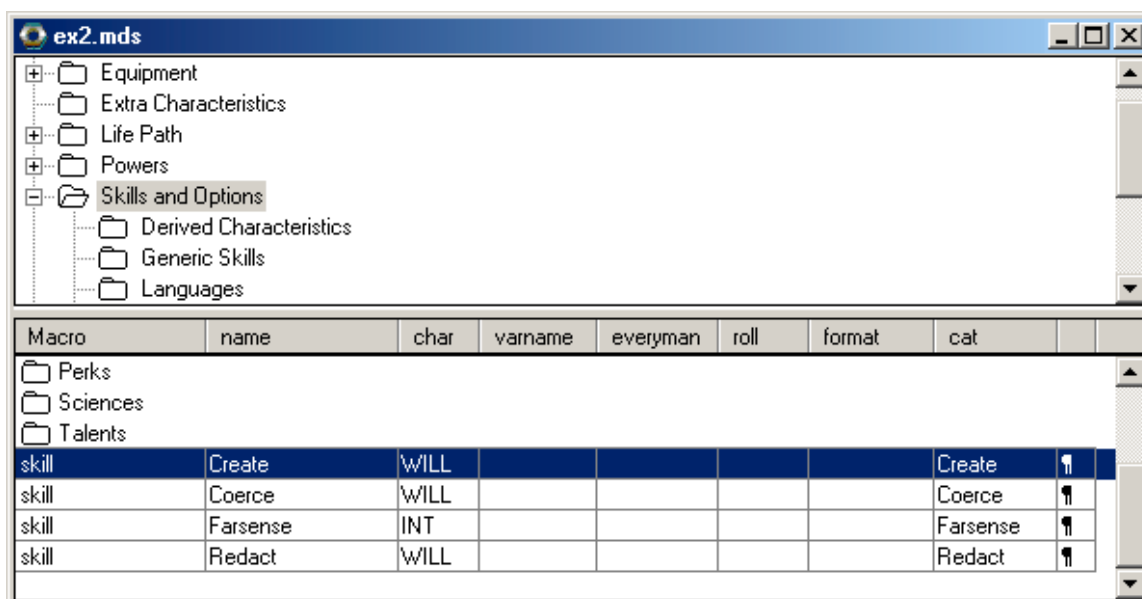
- Select the **File | New...** command.
- Click the **Data Sheet** button.
- Click the Template that you wish to base your macro data sheet on. If you want to start from scratch, click (None) and select a game system. ***GURPS Character Builder*** will fill in the data sheet information from the loaded data sheets for the selected game system (including categories, sublists, character sheet level, etc.).
- Click **OK**.
- Select the **Modify | Info...** command.
- Fill in the information about the character sheet: title and copyright (which are displayed when the data sheet is loaded), game system, level, logo file and help file (if not already set).

- If your game system has a "plugin" identifier, double-click it in the Defines list. Set the value to a unique plugin name for your data sheet.
- Ensure that you have at least one include file specified. Data sheets must get their macro definitions from an include file. Be sure to select include files only for the game system set for the data sheet.
- Click **OK**.

Select **File | Save** to save the data sheet. Data sheets must be saved in the *GURPS Character Builder* source directory if they are to be loaded as data.

The Data Sheet Window

The data sheet window is divided into halves:



The upper pane displays a tree view of the folders defined in the data sheet. The contents of the category or sublist highlighted in the tree view are displayed in the list view at the bottom.

The header for the list view displays the names of the macro arguments for the highlighted items. If there are no arguments, or the type of the highlighted item has no arguments, the header columns have no names.

Each macro has a defined number of arguments. The ⚡ symbol shows the last argument.

The lower pane provides a table-oriented interface for editing items. Double-click an argument to begin editing it.

The macro argument cells are edited either as text or with special dialogs (for requirements, adjustments, categories and automatic items). The argument definitions in the header file indicate which special argument editor is used, if any. To edit all cells as text, select the **Modify | Edit Cells as Text** command.

Editing Text Entries

Once you have begun editing an argument cell as text, the following keypresses are in effect:

TAB Move to the next argument.

ENTER Finish editing this item.

SHIFT+ENTER Finish editing this item, create another item of the same type and begin editing it.

UP, DOWN, LEFT, RIGHT
 Edit arguments above, and below and to the right and left.

ESC Restore the cell to the original value and end editing.

Press SHIFT+F1 to get help on the currently highlighted macro.

Click the right mouse button in the list or tree view area to bring up a context menu for commands that are applicable to the item you click on.

Data Sheet Information

The Data Sheet Information dialog contains configuration information for the data sheet.

Tabs

- Basic Info (p. 146)
- Include Files (p. 146)
- Defines (p. 147)
- Data Sheets (p. 147)
- Local Macros (p. 147)

Basic Info

Title

The title of the data sheet. This is displayed when the data sheet is loaded as data. For data sheet templates, it is also displayed in the New dialog box when the data sheet is highlighted.

Copyright

The copyright message for the data sheet. This is displayed when the data sheet is loaded as data.

Game System

The game system for this data sheet.

Level

The "level" for the data sheet. This is how **GURPS Character Builder** tracks whether the various files define data consistently. Character sheets, print template and data sheets all define the level (or version) of their data. When files are read that have a different level, the user is notified of potential problems.

Logo File

The logo file displayed when the data sheet is loaded as data. Any bitmap file may be specified. Click the **Browse...** button to use the standard open file dialog to search for files.

The logo file should be in the source directory (p. 17), or one of its subdirectories, in which case the subdirectory name should be specified, followed by a backslash, then the file name. For example, if the file is named `MyLogo.lgo` in the `GameSystem` subdirectory of the source directory, enter

```
GameSystem\MyLogo.lgo
```

Browse...

Click this button to search for logo files. The file selected will be placed in **Logo File**. The file selected should be in the source directory or one of its subdirectories.

Help File

The help file to use when displaying help for the macros.

Include Files

The list of the include files for the data sheet. These files define the macros. There should always be at least one file specified.

The include files are assumed to be in the source directory (p. 17).

Add: Add an include file.

Edit: Change the highlighted include file.

Delete: Delete the highlighted include file.

Data Sheet Defines

The defined symbols for the data sheet. Some macros require symbols to be defined for proper operation. For example, most item definition macros have a PLUGIN define that should be set to properly identify the source of the items defined in the data sheet.

Add: Add a define.

Edit: Edit the highlighted define. Double-clicking the define also edits it.

Delete: Delete the highlighted define.

When the Data Sheet Information dialog is closed, the include files are reread and the macros redefined.

Data Sheets

These data sheets are loaded when the data sheet is loaded (if they are not already loaded). This allows you to use items from these data sheets in the automatic item lists for any items in the data sheets.

All data sheets are assumed to be in the source directory (p. 17).

Add: Add a data sheet.

Edit: Change the highlighted data sheet.

Delete: Delete the highlighted data sheet.

Local Macros

The macros entered here are defined only in the context of this data sheet. Any number of macros may be defined here. Only macro definitions should be placed here, and they must be complete. They should be of the form:

```
$$macro mac(arg1, arg2)
{{
    # Macro body
}}
```

Only blank lines should appear between the macros. Comments are not allowed outside the macro definitions.

Macros intended to be used by other data sheets should be placed in include files.

See Data Sheet Macros (p. 226) for more details on defining macros.

Adding Categories

Unless you are developing your own game system, you will probably not need to add your own categories. You should use the categories predefined in the data sheet template.

Categories are the top-level folder in a data sheet. You must create a category before you can add any sublists or items.

- Select the **Insert | New Category** command.
- An Untitled category will be added to the top-most level of the data sheet.
- Edit the name of the category and press ENTER.
- If another category already has the new name, the name will be set back to the previous name.

To change a category name, click the category when it is highlighted.

Category names must match up with the categories that are defined for the lists in the character sheets for the game system. If you define categories which no list references, you will never be able to find those items.

Adding Sublists

Sublists are the child folders in a data sheet. You must create a category before you can add any sublists.

- Select the **Insert | New Sublist** command.
- An Untitled sublist will be added to the highlighted folder.
- Edit the name of the sublist and press ENTER.
- If another sublist already has the new name, the name will be set back to the previous name.

To change a sublist name, click the sublist when it is highlighted.

Adding and Editing Items

Items may be added to a category or a sublist -- they cannot be added at the topmost level. You must therefore create a category or sublist before you can add items.

Adding and Editing Items

- Click the folder in the tree view list to select where the new item should go.
- Select the **Insert | New Item** command, or press the INS key.
- An edit field will open where you can enter the name of the macro that defines the item. If any macros are defined in the include files, a list will appear with all the macros defined. If there is no macro list, you probably have not specified any include files (p. 146). If a description has been defined for the highlighted macro it will be displayed in the status bar.
- Choose a macro, then press ENTER or TAB. *GURPS Character Builder* will attempt to match what you have typed so far to one of the macro names. Any time an entry in the macro list is highlighted, pressing ENTER or TAB will select that macro.
- The first macro argument may now be entered. This is usually the name of the item. If a description has been defined for the argument, it will appear in the status bar.
- To get detailed information on the macro press the SHIFT+F1 key. If it is present, more help will be displayed.
- Press TAB to enter the next argument.
- If the text you're entering is too large for the cell, the edit window will be enlarged. The text in the cell must always be a single physical line, though when the cell is expanded it may display across several "logical" lines. Pasting multiple lines into the cell will cause errors when the data sheet is read.
- Press ENTER to conclude data entry. Press SHIFT+ENTER to complete this item and begin editing a new one of the same type. This greatly speeds up data entry.

GURPS Character Builder will sound an alarm if the macro name or the first argument is empty. Leaving these fields empty will cause errors when the data sheet is loaded.

Macro arguments left empty often have default values. Some arguments may be required. Omitted arguments may cause errors when the data sheet is loaded as data. The line number where the error occurred is usually displayed. Use the **Edit | Go To Line...** command to find the line in the macro data sheet that caused the error.

The `define` macro is a special case: it always has two arguments, a name and a value. It defines a symbol that can be read by macros, in the same fashion as the data sheet defines (p. 147). In general, you should make defines for the entire data sheet, rather than scattering them around in the data sheet. If you do place defines within the data sheet sublists, care must be taken: if you change an existing define you should do so only in a local context, and then reset the define to the original value within that same context.

The Clipboard

The **Edit | Cut** and **Copy** commands put data on the clipboard, while the **Paste** command copies data from the clipboard into the data sheet. The format for data sheet data is simple text. You can therefore copy properly formatted text in a text file and paste it into a data sheet.

Macro invocations are tab-delimited text. For example:

```
$skill  Acrobatics  Agility      High;Artisan
```

defines a skill using a macro. You can discover the text equivalences of all items by copying item in a data sheet and pasting the results into a text editor.

If you paste a category into a data sheet, it will always be inserted at the highest level of the data sheet. If the items you paste include a hierarchy of categories and/or sublists, the sublists and items will be pasted into the hierarchy that already exists in the data sheet, rather than duplicating the hierarchy.

Adding Options

Options provide additional information to items. Options may be added only after an item.

Two commands are available for adding options:

Insert | Option

Displays a list of the loaded options and allows you to select one. These options are the same ones that are available when you create characters. To load the options, use the **File | Load Data Sheet...**

command, or edit a character sheet using the desired game system (which will automatically load the data sheet).

Insert | New Option

Creates an option from scratch and adds it to the data sheet. A dialog will appear that allows you to specify the type of option to create. Once the option type has been selected, the option can be edited.

Editing Options

To change the value of an option, double-click it. The option editing dialog will appear.

Adding Comments and Text Lines

Comments

The **Insert | Comment** command adds a comment to the selected folder, after the highlighted item in the view list. Comments are ignored when the data sheet is loaded as data.

Text Lines

Text lines should be added only if you are very familiar with the syntax of data sheets (p. 217). If you enter the wrong data, the application may not be able to load the data sheet as data (with the **File | Load Data Sheet...** command, or automatically loaded with a character sheet). If this happens, change the text line to correct the problem.

Basically, you can enter any data sheet keywords and arguments in a text line. For example, most item macros don't have an argument to specify the alias. To specify the alias for an item:

- Highlight the item.
- Select **Insert | Text Line**.
- Type the text of the alias. Let's say that the power we're creating an alias for is Density Increase, and we want "DI" to be the alias:

alias DI

- Press ENTER.

The data sheet text format (p. 217) describes the keywords available for creating data sheets.

Headers

Macro arguments are displayed in columns. To change the width of a column:

- Move the mouse to the header window for the item list.
- When the cursor changes to left-right arrow, click and drag to the right or left. When you let the mouse button up, the new column width will be set
- You may not make a column less than 10 pixels wide.

Column widths are remembered for each folder.

Deleting Items

To delete items in the list window:

- Click the item.
- Select the **Edit | Clear** command, or press DEL.

You can also cut items with the **Edit | Cut** command (CTRL+X).

To delete folders in the tree view window follow the same process after highlighted the category or sublist to delete. You will be asked if you're sure when deleting folders.

Merge Data Sheets

The **File | Merge Data Sheet...** command merges another (source) data sheet into the open (destination) data sheet. With this you can combine any number of data sheets into a single data sheet. You might wish to do this to combine automatically generated data sheets with hand-edited data sheets in a final data sheet to be distributed.

All items in the source data sheet will be copied into the corresponding locations in the destination data sheet. Any categories and sublists will be created in the destination data sheet if they do not already exist.

The source data sheet must use the same game system and be at the same level as the destination data sheet. Any macro defines in the source not already present in the destination will be copied to the destination. Any include files not already present will be added to the destination. Any macros defined in the body of the data sheet will be copied. Any variable, array and function defines will be copied to the destination.

Trying Out the Data Sheet

Creating a data sheet is a cyclical process: items are added, the data sheet is loaded, errors are corrected, the data sheet is reloaded, the items are tested, the items are modified, etc.

Data Sheet Development Process

- Open the data sheet.
- Make the desired changes.
- Save the data sheet.
- Select the **File | Load This Data Sheet** command *.
- Try out the items by creating a character and selecting the items into it.
- Change the data sheet.
- Save the data sheet. If the data sheet is already loaded as data, it will automatically be reloaded *.
- Continue the process.

* If errors occur when the data sheet is loaded, you can find the line responsible with the **Edit | Go To Line...** command:

- Select **Edit | Go To Line...**
- Enter the line number that the error was reported at.
- Click **OK**.

GURPS Character Builder will highlight the item closest to that line number. Due to the way macros are expanded, errors are sometimes reported for the line *after* the item that actually caused the error. If the line specified looks okay, also look at the line before.

The most common errors are due to omitting required arguments.

Converting .cds Files to .mds File Format

To convert existing macro-based text `.cds` files to `.mds` files follow these steps:

- Make a copy of the file with an `.mds` extension.
- Change the identifying string at the beginning of the file from

```
"Data Sheet file v. 1"
```

to

"Macro Data Sheet file v. 1"

- Move all macro definitions to a separate include file, or to the top of the file, before any categories are defined.
- Move all `$$define` and `$$include` commands to the beginning of the file.
- Make sure all item macros start on their own lines.
- Place `textline` directives before other keywords you may have after macro invocations. In particular, be sure that any `cat` keywords intended to be within an item do not come at the beginning of a line.
- Make sure there are no macro or define references in option definitions.

Once you've done the preliminary visual scan of the file and fixed the more obvious problems, read the file with **GURPS Character Builder** and fix the problems as described in Trying Out the Data Sheet (p. 150).

Macro Define

Macro defines are symbols defined for use with the macros in the header files. When the defined symbol is referenced in a macro, the reference is replaced with its value.

Most data sheets require a `PLUGIN` define in order to identify the source data sheet for items. The value can usually be any string, though some game systems have more stringent requirements.

Find

Enter the text to find. If the text is found, the item or folder containing it will be highlighted.

The search begins after the currently highlighted item. If the text is found in a folder name, the focus will move to the tree view window and the folder will be highlighted.

If the text is found in any of the arguments of a macro item,

Go to Line

This is used to locate items that cause errors during data sheet loading.

- Enter the line number where the reported error occurred.
- Click **OK**. The specified line will be highlighted.
- If no item or sublist corresponds to that line number, the closest match will be highlighted, if possible.

Due to the way macros are expanded, the line number reported by an error can be off sometimes. If the suspected line looks fine, take a look at the previous line.

Data Sheet Errors

Before a data sheet is saved, **GURPS Character Builder** checks the validity of the items. If the data sheet contains errors, you may not be able to load it.

When the dialog appears, the first item with an error is highlighted.

This dialog displays the errors that have been detected. Click an error to display the associated item.

Goto Error

Go to the highlighted error. Make the necessary correction and try saving again.

Save

Save the data sheet anyway. Beware: if you save the data sheet with errors, you may not be able to load it as data later. You'll probably have to edit it by hand in a text editor to repair it.

Cancel

Cancel saving the file.

Data Sheet Folder Properties

This dialog allows you to modify the properties of the folders in data sheet.

Sort Order

The sort order controls how items in the Available Items dialog are sorted. The default is Name. Category, Cost and Class order items in the list by those attributes of the items. The Constant Cost First order sorts items by cost, placing items with a constant cost first, and then placing all items after that in alphabetical order.

Random Pips

When items are selected at random from the Available Items list, the random pips for all the items are summed. A random number is then generated and the item whose range of pips that random number falls in is selected. The random pips is therefore the number of "chances" that the sublist will be selected.

A value of 0 is treated the same as 1. A value of -1 prevents the item from being chosen randomly. If a sublist contains items that you would never want selected randomly, its random pips should be -1.

When a sublist is chosen, the random selection process is performed on that sublist.

Categories do not have random pips, since they are at the top level.

Choose Macro Argument Value

This dialog appears when you edit a macro argument that has a menu of possible choices. To make your choice, click the desired entry in the list under the prompt and click **OK** (you can also double-click the entry).

If other text can be entered in addition to the choices offered in the list, the edit field at the bottom of the dialog will be active. Use the keyboard to enter the desired value and click **OK**.

Choose Macro Argument List Entries

This dialog appears when you edit a macro argument that is a list that has a menu of choices, plus any other text you care to enter.

To add your choices to the **Selected Values** list click the desired entry in the list of choices under the **Value** edit field, then click the **Add** button (double-clicking the entries has the same effect).

If the value you want is not in the list, use the keyboard to enter any text in the **Value** edit field. Click **Add** to add the contents of the field to the **Selected Values** list.

Value

Text in this field will be added to the **Selected Values** list when the **Add** button is clicked.

Selected Values

The values listed here are the contents of the macro argument you are editing.

If the list of **Selected Values** does not allow duplicate entries, then entries from the list of choices will be removed as they are added to the list of selected values.

Add

Click this button to add the contents of the **Value** field to the **Selected Values** list. If the list doesn't allow duplicates, the value will also be removed from the list of choices if it is present.

Delete

To delete an entry in the **Selected Values** list, click the value to highlight it, then click the **Delete** button. If the list doesn't allow duplicates, the entry you delete will be added back to the list of choices so that you can select it again later.

Change

To change the contents of an entry in the **Selected Values** list:

- Click the entry.
- The text will be copied to the **Value** edit field.

- Use the keyboard to change the **Value** as desired.
- Click the **Change** button.
- The contents of the **Value** edit field will replace the currently highlighted entry in **Selected Values**.

Up, Down

To change the order of entries in the **Selected Values** list, click the **Up** and **Down** buttons.

Macro Help

Click the **Macro Help** button to get help specific to the macro whose arguments is being edited.

Choose Predefined Macro Argument List Entries

This dialog appears when you edit a macro argument that is a list of items, and the members of that list can be a predefined set of choices.

To add your choices to the **Selected Values** list click the desired entry in the list of **Value** choices, then click the **Add** button (double-clicking the choices has the same effect).

Value

The highlighted entry in this list will be added to the **Selected Values** list when the **Add** button is clicked. Double-clicking an entry also adds it to the list.

If duplicate entries are not allowed in the **Selected Values**, the choices you select will be removed from the **Value** list.

Selected Values

The values listed here are the contents of the macro argument you are editing.

If the list of **Selected Values** does not allow duplicate entries, then entries from the list of choices will be removed as they are added to the list of selected values.

Add

Click this button to add the highlighted entry in the **Value** list to **Selected Values**.

Delete

To delete an entry in the **Selected Values** list, click the value to highlight it, then click the **Delete** button. If the list doesn't allow duplicates, the entry you delete will be added back to the **Value** list so that you can select it again later.

Up, Down

To change the order of entries in the **Selected Values** list, click the **Up** and **Down** buttons.

Macro Help

Click the **Macro Help** button to get help specific to the macro whose arguments is being edited.

Choose Free-Form Macro Argument List Entries

This dialog appears when you edit a macro argument that allows you to enter any values.

To add entries to the **Selected Values** list use the keyboard to enter any text in the **Value** edit field. Click **Add** to add the contents of the field to the **Selected Values** list.

Value

Text in this field will be added to the **Selected Values** list when the **Add** button is clicked.

Selected Values

The values listed here are the contents of the macro argument you are editing.

Add

Click this button to add the contents of the **Value** field to the **Selected Values** list. If the list doesn't allow duplicates, you can't add the same value twice.

Delete

To delete an entry in the **Selected Values** list, click the value to highlight it, then click the **Delete** button.

Change

To change the contents of an entry in the **Selected Values** list:

- Click the entry.
- The text will be copied to the **Value** edit field.
- Use the keyboard to change the **Value** as desired.
- Click the **Change** button.
- The contents of the **Value** edit field will replace the currently highlighted entry in **Selected Values**.

Up, Down

To change the order of entries in the **Selected Values** list, click the **Up** and **Down** buttons.

Macro Help

Click the **Macro Help** button to get help specific to the macro whose arguments is being edited.

Editing Dialogs

When the **Dialog Item Properties** command on the **Modify** menu is checked, you may change the properties of the dialog.

How To...

Stop Editing Dialog Properties and Change Values

Select the **Modify | Dialog Item Properties** command, or click the  button.

Add a control Select a control type from the **Control** menu (p. 155).

Select a control

Click on it. Most commands require that one or more controls be selected before an operation can be performed.

Delete a control

Select it, then select the **Clear** command from the **Edit** menu (or press the Del key).

Move a control

Click on it and drag. When you click a control, "handles" (small black squares) will appear on its four corners.

Resize a control

Click on a control and drag one of the handles.

Select more than one control

Hold down the Shift key and click the next control. You may also click inside the dialog (on no control) and drag the mouse to form a selection rectangle. All controls that are partially within the selection rectangle will be selected.

Change the properties of a control

Double-click anywhere inside it. You may also select it and then choose the **Edit Object...** command from the **Edit** menu.

Line up controls with each other

Select the desired controls and choose a command from the **Align** menu (p. 156).

Resize the dialog window




Click on a border and drag until the window is the desired size.









Order the plotting of Controls

Use the **Edit | Set Tab Order** (p. 164) command, or select the controls and select the **Bring To Front** or **Send To Back** commands on the **Edit** menu.

Control Menu









The commands on this menu add controls to the dialog. The buttons on the Dialog Modification button bar also perform the same tasks. After you select the command, position the cursor (it will become a crosshair) in the dialog window and click the left mouse button. The control will be added. The size can be altered by clicking and dragging on the handles.



- | | | |
|---------------|---|--|
| Text |  | Add a text object. This text is simply displayed in the dialog. |
| Edit Text |  | Add a named edit text field. This field can be referenced in print templates. This can be used to hold things such as the character name, description, height, weight, etc. |
| Edit Variable |  | Add an editable variable. You must name the variable. When you change the value in the edit box (outside of Modify Dialog Properties mode), the value of the variable will change, as well as any other items that depend on it. |

Formula		Add a formula. You must name a variable and associate an expression with it. Every time a value that this formula depends on is changed, the value displayed here will be updated automatically.
List		Add a list box. A list box is a list of items, functionally equivalent to an independent list window. You can perform all the list window commands on the items in a list in the dialog box.
Combo Box		Add a combo box. A combo box is a drop down list of text strings associated with values that set a variable. Every time you select a new value, the variable will be set and any other values that depend on it will automatically be updated.
Checkbox		Add a check box. Check boxes are either on or off. You must associate a variable with the checkbox. Whenever the box is checked, the variable will have a value of 1. If the checkbox is unchecked, the variable will be 0. All values that depend on the variable will automatically be updated.
Group Box		Add a group box. This draws a titled box around a group of controls.
Picture		Add a picture. To set the picture, leave the Modify Dialog Item Properties mode and click inside the picture frame (p. 50).
Button		Add a button. Buttons execute command scripts (p. 260) to provide character sheet automation.
Bitmap		Add a Bitmap control. Bitmap controls display pictures in the character sheet. The bitmap defines a variable and an expression. These can be used to display different bitmap files. This allows you to display different character class logos, graphic indicators of the character stats, etc.

Align Menu

The commands on the Align menu are used to line up the controls in dialogs and change their sizes. Most of the commands require that one or more controls be selected (to select more than one, shift-click subsequent controls).

Left		Line up all the selected controls on the left side of the left-most control.
Right		Line up selected controls on the right side of the right-most control.
Top		Line up selected controls on the top of the top-most control.
Bottom		Line up selected controls on the bottom of the bottom-most control.
Smallest Height		Make all selected controls as short as the shortest selected control.
Largest Height		Make all selected controls as tall as the tallest selected control.
Smallest Width		Make all selected controls as wide as the narrowest selected control.
Largest Width		Make all selected controls as wide as the widest selected control.

- Distribute Vertically  Distribute all the selected controls evenly within the area indicated by the highest control and the lowest control in the dialog.
- Distribute Horizontally  Distribute all the selected controls evenly within the area indicated by the left-most control and the right-most control in the dialog.

Edit Combo Box Control

Combo box controls allow you to select values by self-explanatory textual names rather than numbers.

Variable Name

The name of the variable associated with the combo box control. Whenever you select one of the entries in the combo box, the variable is set to the value associated with the selected text.

Values

This list shows the text (on the right) and the numeric value associated with it (on the left). To change a value, double-click it, or highlight it and click the Edit button.

Sort

If the sort checkbox is checked, the text values will be sorted in the combo box.

Edit

Changes the highlighted entry (p. 157) in the Values list.

New

Add a combo box entry in the list after the highlighted entry.

Delete

Delete the highlighted entry from the Values list.

Script...

This opens a dialog that allows you to edit the command script (p. 260) for this combo box. The script is executed when the user changes the combo box selection. You can also call subroutines from the script loaded for the character sheet (p. 47).

Fonts...

Set the fonts (p. 163) used to display this control.

Edit Combo Box Control Entry

Here you can change the values of an entry in the combo box.

Text

The text displayed in the combo box when the entry is selected.

Value

The value assigned to the combo box's variable. More than one entry can have the same value, though you should do this only when those values are essentially the same.

Edit Picture Control

Enter the name which you wish to give this picture. This is the name by which you will reference this picture in the print template (p. 118).

Edit Variable Control

When you create an edit variable control in a dialog, you are creating a link between the value displayed in the dialog and a variable in the character sheet.

Variable Name

This is the name of the variable that is associated with this edit field. When the user changes this value in the dialog, this variable will be changed as well. Then all other values that depend on this variable will also be changed.

Force Constant

If you make the formula for this edit variable random, or you want it computed just once and then fixed at that value, check the Force Constant checkbox. In the case of random values, the value will be recomputed every time the user selects the **Tools | Reset Field** or **Tools | Reset All Fields** commands and left at that value until the next time one of those commands is used.

No Border

If this is checked, the edit field window will have no border. This can be useful if you have defined a skin (p. 42) that has custom borders drawn for the edit controls.

Use Background Color

If this is checked, the control has its own background color which is set by clicking the **Set...** button. If the **Transparent** checkbox is checked, it overrides the background color.


Fonts...

Set the fonts (p. 163) used to display this control.

Default Value

The variable is set to this value when the **Tools | Reset Field** (for the current field only) or the **Tools | Reset All** command is used. This is useful for setting the field to a random number, or to a expression (p. 169) that depends on other values. The field will retain that value until you explicitly change it, or use one of the above commands again.

To set the variable's initial value (the value originally in the edit field when the template or character sheet is initially opened), stop editing Dialog Item Properties (select the **Modify | Dialog Item Properties**

command, or click the  button).

Transparent

If this is checked, the edit control is "transparent." Because of limitations in Windows, this isn't true transparency: the background color of the window is used for the background color of the control. If a skin is used beneath the control, it cannot be seen beneath an edit control.

Check Expression

In the check expression the new value that the user enters will be substituted for the "x" variable. If that expression (p. 169) evaluates to true (non-zero), the value is accepted. If the expression evaluates to false (zero), the text in the Check Message is displayed. If no Check Message is provided, the default message is "The value you entered is not in the acceptable range".

Check Message

The message that is displayed if the check expression fails. This can be either a normal text string, which is simply displayed, or an expression that is parsed in the context of the character sheet. This is indicated by making the first character in the message a "\$" character.

For example, if your game system has a maximum Intelligence that is configured differently for each race, you could reference that maximum value in the following check message:

```
$format("Int must not be greater than racial maximum Int: %d", IntLimit)
```

It's often useful to write a function for the check message, passing any necessary arguments. The above might be recast as:

```
$charmsg("Int", IntLimit)
```

in which case the same basic function could be used to display similar messages for other characteristics that had similar checks.

Format

The expression that is used to render the value when it is placed in the edit text field. It usually requires a Formula to be set as well, though not always. This is useful for exercising complete control over what is displayed, perhaps representing height in terms of inches internally, but formatting it as feet and inches when it is displayed.

For example, the following format:

```
x=0?'':format("%d'%d'", integer(x/12), x mod 12)
```

causes nothing to be displayed (' ') if the value of the field is 0, otherwise the value is rendered as feet and inches.

Value	Display
69	5'9''
1	0'1''
0	
60	5'0''

Another example is game systems that customarily place a + or - in front of every stat. The following format will always display a sign in front of the value, be it positive or negative.

```
format('%+d', x)
```

Formula

The formula is used to compute the value of field, allowing a display representation that is different from the internal representation.

When the displayed value is changed, all numeric values in it are assigned to an array of numbers (up to 10), in order of their appearance in the input text. Any other characters are ignored. Those numbers are then run through the formula, replacing the \$1 with the first number, \$2 with the second number, etc. References to numbers that were not defined are set to 0.

For example, the corresponding formula to the feet/inches format above would be

```
$1*12+$2
```

which means that the first number found in the input text is multiplied by 12, and the second number in the input is added to it. A third number would be ignored. The following are examples of converted values:

Display	Value
5'9''	69
6 1	73
	0
5	60
5'0"	60

Edit Text Control

Edit text controls are used to enter general textual information about the character.

Name

The name by which the edit text control will be referenced in filters (p. 186).

Script

This command script (p. 260) is executed when the user changes the contents of the control and finalizes it. That is, types some text and presses ENTER (or TAB), clicks on another control, etc. You can also call subroutines from the script loaded for the character sheet (p. 47).

No Border

If this is checked, the edit text field window will have no border. This can be useful if you have defined a skin (p. 42) that has custom borders drawn for the edit controls.

Transparent

If this is checked, the edit control is "transparent." Because of limitations in Windows, this isn't true transparency: the background color of the window is used for the background color of the control. If a skin is used beneath the control, it cannot be seen beneath an edit control.

Use Background Color

If this is checked, the control has its own background color which is set by clicking the **Set...** button. If the **Transparent** checkbox is checked, it overrides the background color.

Fonts...

Set the font(s) (p. 163) used to display this control.

Edit Group Box Control

Group box controls are used to draw boxes around other controls to group them.

Text

The label for the group box.

Fonts...

Set the font(s) (p. 163) used to display this control.

Text Control

Text controls serve as labels for other controls.

Text

The text displayed. If you include a '%' character in these controls, the text is processed by the filter (p. 185) processor. If you want a '%' to stand by itself without referencing a variable, you should "escape" '%' characters with '\' (that is, instead of just '%', enter '\%').

If you reference variables that change when other parts of the character sheet are modified, the text will not be updated automatically: it will be reprocessed only when the window is opened. If you wish to have the text update automatically, you should use an edit formula control (p. 161) instead.

Transparent

When this is checked, the background of the text control is left unchanged when the text of the control is plotted. This allows the dialog skin (p. 42) to "show through" the text control.

Fonts...

Set the font(s) (p. 163) used to display this control.

Justification

Click the button for the justification you need: Left, Right or Centered.

Edit Checkbox Control

The checkbox control is used to define a variable that has only two possible values: 0 or 1. If the checkbox is checked the variable is 1. If it is not, the variable is 0.

Text

The text of the checkbox.

Name

The name of the variable associated with the control. The variable will be set to 1 when the checkbox is checked, and 0 when it is not. All other variables that depend on the checkbox control's variable will be changed when the checkbox state changes.

Script

This command script (p. 260) is executed when the user clicks the checkbox. You can also call subroutines from the script loaded for the character sheet (p. 47).

Fonts...

Set the font(s) (p. 163) used to display this control.

Edit Formula Control

A formula control is used to display a value that is computed from other values. A common use is for costs of attributes.

Variable Name

The name of the variable associated with the control. Every time the variable changes (when the variable(s) it depends on change), the value in the dialog box will be updated.

Expression

The expression (p. 169) that defines the formula.

Transparent

When this is checked, the background of the text control is left unchanged when the text of the control is plotted. This allows the dialog skin (p. 42) to "show through" the text control.

Fonts...

Set the font(s) (p. 163) used to display this control.

Justification

Click the button for the justification you need: Left, Right or Centered. If the formula appears in a column of other formulas, Right justification is usually preferred.

Edit Button Control

A button control is used for initiating command scripts. Buttons can have text labels or bitmaps.

Variable Name

Each button defines a variable. This variable is incremented every time the button is clicked. If other expressions reference it (for example, a formula control), they will be reevaluated and redisplayed whenever the button is clicked.

Button Text

The text of the button. For bitmap buttons, this text is displayed if the bitmap file is not present.

Script

This command script (p. 260) is executed when the user clicks the button. You can also call subroutines from the script loaded for the character sheet (p. 47).

Bitmap

The name of a graphics file relative to the *GURPS Character Builder* source directory (p. 17). This will be displayed when the button is in the normal (unclicked) state. If the bounding rectangle of the button is sufficiently large, a dotted rectangle will be displayed on the button when it has the focus. To prevent the focus rectangle from appearing, make the button only slightly larger than the bitmap. If the bounding rectangle is too small for the bitmap, it will be reduced proportionally to fit.

Click the **Browse...** button to scan the file system for graphics files. You may use BMP, JPEG and PNG files.

Depress

The name of a graphics file relative to the *GURPS Character Builder* source directory (p. 17). This will be displayed when the button is in the depressed (clicked) state. If the bounding rectangle of the button is sufficiently large, a dotted rectangle will be displayed on the button. To prevent the focus rectangle from appearing, make the button only slightly larger than the bitmap. If the bounding rectangle is too small for the bitmap, it will be reduced proportionally to fit.

Click the **Browse...** button to scan the file system for graphics files. You may use BMP, JPEG and PNG files.

Fonts...

Set the font(s) (p. 163) used to display this control.

Edit Bitmap Control

A bitmap control is used for displaying graphic files in the character sheet dialogs. They can also be printed through the print template.

Variable Name

The name of variable associated with this control. When the variable changes (because other variables referenced in the expression change) the appropriate bitmap is displayed.

Expression

The expression (p. 169) controls what bitmap is displayed in the control. Bitmaps are displayed in two different ways, depending on whether the expression evaluates to a number or a string. If the file that the expression resolves to does not exist, the name of the Bitmap control will be displayed.

Number

When the expression evaluates to a number, the value is used as an index into the **Bitmap Files** list. If the value is 1 or less, the first bitmap file is displayed. If the value is 2, the second bitmap is displayed, and so on. If the value is greater than the index of the last bitmap, the last bitmap is displayed.

Floating point numbers are truncated (rounded down) when they are used as indexes into the **Bitmap Files** list.

String

When the expression evaluates to a string, it is used verbatim as the name of a graphics file in the **GURPS Character Builder** source directory (p. 17). The bitmap file may be in a subdirectory of the source directory, but that subdirectory name must be included as part of the string representing the file name. When the expression evaluates to a string, the **Bitmap File** list is ignored.

A constant string may be specified. For example, if you have an image called `logo.jpg` in a subdirectory of the source directory called `cwrps` you can reference it directly with:

```
"cwrps\logo.jpg"
```

This is the same as adding the above file name to the **Bitmap Files** list and specifying 1 in the **Expression**.

To display an arbitrary image from a subdirectory based on a the text of a variable set by another control (say, a listbox with clan name strings as the values), do the following:

```
format("cwrps\%s.jpg", clanName)
```

Note that it's possible to use the ternary operator (`? :`) and return either an integer or a string in the same expression. For example:

```
fileExists(getSourceDir(0), logoFile) ? logoFile : 1
```

If the file named in `logoFile` exists, that file name is used. Otherwise the first file in the **Bitmap Files** list is used. This allows display of a default bitmap if the named bitmap doesn't exist.

File

Enter into this field the name of the bitmap file you wish to add to the **Bitmap Files** list below. Click the **Browse...** button to scan the file system. When you click the **Add** button the contents of the **File** edit field will be added to the **Bitmap Files** list.

Bitmap Files

The names of the graphic files to display. This is used only when the **Expression** evaluates to an integer.

Add

Add the graphic file named in the **File** field to the **Bitmap Files** list after the highlighted entry in the list.

To add a file to the list:

- Highlight the file you wish to add after in the **Bitmap Files** list.
- Click the **Browse...** button.

- Highlight the graphic file you wish to add in the Select Skin File dialog.
- Click **Open**.
- Click the **Add** button.
- Repeat for multiple files.

Delete

Delete the highlighted file in the **Bitmap Files** list.

Change

Change the highlighted entry in the **Bitmap Files** list to the value in the **File** field.

Up

Move the highlighted entry in the **Bitmap Files** list up.

Down

Move the highlighted entry in the **Bitmap Files** list down.

Edit Font List

The Font List is common to all controls in the same dialog. A control indicates rendered by the font indicated by the control's "font index." The default value is 0, which is the dialog's font.

Use Default Font

If checked, the control is displayed with the default font, which is the font specified in the enclosing dialog.

Font Index

If checked, the font highlighted in the Dialog Fonts list. This is a value from 1 to the number of fonts listed.

Font Expression

If checked, the expression (p. 169) in the edit field is evaluated every time the control's variable is reevaluated. The result is used as an index into the Dialog Fonts list. The value 1 is the first entry, 2 is the second entry, etc. The value 0 indicates the font specified for the dialog.

Font expressions are most frequently used for Formula and Edit Variable controls. Some controls do not have variables and are not reevaluated predictably, so their fonts cannot be reliably changed. Text, Group Box, List Box, Combo Box, Checkbox, Button and Edit Text controls should therefore not specify font expressions.

A typical use for the font expression is to render a formula in another color, for example to display a total in red when there is a character point deficit. To get this effect, define the first font in the Dialog Fonts list as the font to be displayed when there is a deficit. Click the Font Expression button and set the expression to

```
total>availPoints
```

where `total` is the name of the formula variable and `availPoints` is the variable that contains the number of available points. This works because the greater than comparison returns 0 when the expression is false, and 1 when the expression is true. Thus, the default dialog font is used when the total is less than or equal to the available points, and the first font in the dialog fonts list is used when the total is greater than the available points.

You should also include `availPoints` in the computation of `total` so that whenever `availPoints` changes `total` will be reevaluated. This is most easily done by adding `" +availPoints*0"` to the expression for `total`.

Dialog Fonts

The list box contains the fonts defined in the dialog. Note that as you add, delete and move the entries in the font list, the font indexes for the controls will remain unchanged. Be sure that any changes in the font order is consistent with all your control font indexes.

Add

To add a font to the Dialog Fonts list:

- Highlight the font that you wish the new font to appear after.
- Click the **Add** button.
- Choose the desired font from the Font dialog. Be sure to select both a Font and Size. If you choose Black for the color, the parent dialog's color will be used.
- Click the **OK** button.

Change

Change the highlighted entry. The Font dialog will appear to let you change the font. Double-clicking and entry is the same as highlighting it and clicking the **Change** button.

Delete

Delete the highlighted entry.

Up

Move the highlighted font up, decreasing its font index by 1.

Down

Move the highlighted font down, increasing its font index by 1.

Edit Script

Edit the command script (p. 260). The script is executed when the user changes the control associated with the script.

In addition to all the standard scripting commands, you can also call subroutines from the script loaded for the character sheet (p. 47).

Set Tab Order

The "tab order" is the order in which the focus moves from one control to another when you press the TAB key. When you create dialogs, you should set this order to make the movement of the focus from control to control logical.

Selecting this command causes the tab order number for each control to be displayed in white on a green background. The outline of the control is also displayed in gray.

To set the first control, click within the outline of the control (not the number). The tab order number will be set to 1, the number's background will change to black and the outline will go away, indicating that the order for that control has been set. Clicking on subsequent controls will change their order.

To exit this mode, select another command or press ESC.

Edit List Control

A list control has the complete functionality of a list window, except that it occurs within a dialog. All the list commands on the Tools menu will function when this control has the focus.

Variable Name

The name of the variable that is the sum of all item costs for the list.

List Name

The name of the list, which can be referenced in requirements, conversion scripts, etc.

Maximum Items

The maximum number of items allowed in the list. If this value is 0, any number of items is allowed.

Category

The category of items (from the categories defined in the data sheet) that can be selected from in this list.

Level Prompt

Replaces the "level" prompt with the specified string in the item editing dialog for members of this list. This is useful for game systems that use different terminology (such as "score"), or for labeling non-numerical values for items.

Cost Prompt

Replaces the "cost" prompt with the specified string in the item editing dialog for members of the list. This is useful for game systems that use different terminology for the point cost, or for labeling the prompt when a different unit is being used for the cost. For example, you might wish to enter the item's weight instead of a cost in a list of possessions.

Item Format

The format (p. 102) used for all items in the list that have no format (p. 102) of their own.

Dialog

Selects the type of dialog (p. 104) to use when editing members of this list. If an item specifies its own dialog type, that type will have precedence over the type specified for the list.

Check Expression

The expression checked every time you add or change an item in the list. This is the same as the check expression in the list window (p. 42).

Check Message

The message displayed if the Check Expression fails. The same as the check message (p. 158) in the list window (p. 42).

Default Check Message

The default check message for any items in the list that don't have their own check message. The same as the default check message (p. 158) in the list window (p. 42).

Editing Defines

You can define variables, arrays and functions for character sheets in this dialog. These defines can be used in the definitions of cost formulas for items in lists, formula fields in dialogs, etc.

The order in which the defines appear in the list is the order in which they are created when the character sheet is opened. You must define arrays and functions before they can be used. If you attempt to leave this dialog with the defines in an inappropriate order you will be warned. If you ignore the warning, you'll get errors when you load the character sheet the next time.

Adding Variables, Arrays or Functions

Click the appropriate button and the indicated definition will be added to the list following the currently highlighted entry in the list.

Deleting an Entry

Click it in the list and then click the Delete button.

Editing an Entry

Click it in the list, and then click the Edit button.

Editing Variables (p. 166)

Editing Functions (p. 166)

Editing Arrays (p. 167)

Clipboard Functions (p. 167)

Editing Variable Definitions

Name

Enter the name of the variable in the Name box.

Initial Value

The value assigned to the variable when the character sheet is opened. This may be any expression (p. 169). Click **OK** to create the variable.

Current Value

If the variable already exists, its current value will be displayed here. Variables' values may change after the character sheet is opened by the addition of items that adjust the values of variables, so the Current Value may not equal the Initial Value.

Script Added

This checkbox is checked if the variable was added or set by a conversion or command script (p. 265). The values of script-added variables are retained when character sheets are converted without conversion scripts.

Set

To reset the variable's Current Value to the Initial Value click the **Set** button.

Edit Function Definition

Name

Enter the name for the function, followed by a left parenthesis, followed by a list of the arguments for the function (separated by commas), and finally a right parenthesis.

Formula

Enter the formula for the function definition. A standard expression (p. 169) can be used.

The arguments named in the Name box can be used in the expression, as can any previously defined array. Variables can be referenced before they are defined (initially they will be assumed to be 0).

Editing Array Definitions

Name

Enter the name of the array.

Values

The list of values in the array.

Value

To add the first entry to the array, click in the **Value** box (the edit field beneath the **Values** list) and type in a value. Then click the **Add** button to add it.

Change

To change a value, click on the entry in the list. The value will appear in the **Value** box. Edit the value, then click the **Change** button.

Add

To add subsequent array entries, click on an entry in the list. Click in the **Value** box and type the value you wish to add. Then click the **Add** button. The value will be inserted in the array after the highlighted entry.

Delete

To delete a value from the array, click on the entry in the list you wish to delete, then click the **Delete** button.

When you are done, click the OK button.

When arrays are referenced, the value in the [] is used to index into the array. Indexes of 1 or less return the value of the first entry in the array. 2 returns the value of the second entry, and so on. If there are N entries in the array, indexes of N or greater return the Nth entry.

Defines Dialog Clipboard Functions

The Cut, Copy and Paste buttons move defines to and from the clipboard.

Copy

Copies the highlighted define to the clipboard.

Cut

Copies the highlighted define to the clipboard and removes it from the list.

Paste

Pastes the contents of the clipboard into the defines list after the highlighted item.

Define Clipboard Format

The defines are stored on the clipboard in standard text format.

For example, an array is stored on the clipboard in the following way:

```
skillCost[5]=0,0.5,1,2,4
```

The array name is followed by the number of elements in square brackets, followed by '=', followed by a comma-separated list of elements.

A variable is even simpler:

```
b_speed=0
```

The variable name is followed by '=', followed by the value (which may be an expression).

Finally, a function is defined in the following way:

```
mvh(sk,att)=sk-att<=-2?skillcost[sk-att+6]:(sk-att+2)*4
```

The function name is followed by a comma-delimited list of the argument names in parentheses, followed by '=', followed by the function definition. If a multi-line function is defined, you must surround it with braces:

```
max(x,y)={  
    if (x > y)  
        return x;  
    endif  
    return y;  
}
```


Expressions

Expressions are used frequently in **GURPS Character Builder**. They can express the cost of an item relative to its value (level), or to compute new values when character sheets are converted to other game systems, etc.

Expressions in **GURPS Character Builder** follow the normal rules for parenthesized arithmetic expressions. Expressions use the standard "binary" operators (p. 169) (operators with two operands (p. 182), such as plus and times), and select set of unary operators (such as negation). You can also call functions which you can define in character templates, and set of predefined functions (p. 171). Variables and functions can also be defined in terms of procedures (p. 182).

When one of the two operands is a floating point number, the other is promoted to a floating point number as well.

Operators

The expression operators in **GURPS Character Builder** are for the most part standard arithmetic and boolean (true/false) operators. Each operator takes one or more arguments (operands (p. 182)) and produces a value. Numeric operands may be floating point or integer. A boolean value is false (zero) or true (non-zero). Boolean operators (equals, not equals, etc.) always produce 0 for false and 1 for true.

The following operators are defined in **GURPS Character Builder** expressions:

Operator	Meaning
+	Addition. "1 + 2" produces 3. This can also be used as a unary operator (i.e., "+5"), in which case it has no effect.
-	Subtraction. "2 - 2" produces 0. This can also be used as a unary operator (i.e., "-5") to negate a value.
*	Multiplication. "3 * 2" produces 6.
/	Division. Integer values are promoted to floating point values to prevent truncation during division. "3 / 2" produces 1.5. If you want integer division you must use the integer (p. 174) function: <code>integer(3/2)</code> will produce 1.
MOD	Modulo. The expression "x MOD y" produces the remainder of x divided by y. For example, "5 MOD 3" is 2.
^	Super. "3 ^ 3" produces 81. If the operands are integers, the maximum value of the expression is 32,767, so if you know that values will be larger than that, you must make sure that one of them is a floating point number.
=	Equality. Produces a boolean value (0 or 1). "2 = 2" produces 1.
<>	Inequality (boolean). "2 <> 2" produces 0. The operator != is syntactically identical.
<	Less than (boolean). "2 < 3" produces 1.
<=	Less than or equal to (boolean). "2 <= 3" produces 1.
>	Greater than (boolean). "2 > 3" produces 0.
>=	Greater than or equal to (boolean). "2 >= 3" produces 0.
OR	Boolean "or". "2 < 3 OR 2 > 3" produces 1.
AND	Boolean "and". "2 < 3 AND 2 > 3" produces 0.
! (or NOT)	Logical "not". "!0" produces 1, and "!1" produces 0. NOT is exactly equivalent to !.

?:	Conditional operator, which allow "if" checks inside expressions. The expression " $x > st ? (x - st) * 5 : 0$ " would produce a value of 0 if "x" is less than or equal to "st", and $(x - st) * 5$ if "x" is greater than "st".
~	Unary operator that produces the base value of a variable, disregarding any bonuses (p. 170) applied to the variable. This should generally be used in cost expressions that are applied to variables that may have bonuses.
&	Unary operator that produces the value of the bonuses for a variable.
[]	Array index operator. Arrays may be indexed with the square brackets.
%	Percent. The preceding number constant is divided by 100.

See also

Predefined Functions (p. 171)

Bonuses

Bonuses are set by adjustments (p. 87). They allow you to add a "hidden" value to a variable, reducing the cost of skill, or increasing (decreasing) the value of a variable.

Note that the values of the variable and the bonus are kept track of separately. When the value of a variable is 0, its bonus value is returned when it is referenced. When the value is non-zero, the base value is returned.

If you want your cost formulas to reflect the value of the variable discounting the bonus, you must refer to the variable with "~varname (p. 169)". You can obtain the value of the bonus with "&varname".

If you want to get both the base value and the bonus value of a variable, you must reference them separately: total value = var + &var.

Predefined Functions

The following functions may be used in **GURPS Character Builder** expressions (p. 169). No variables may be named with these reserved words.

abs(*x*)

The absolute value of a number. "**abs**(-5)" produces 5.

addelement(*array*, *value*)

Adds *value* as the last element of *array*. Returns the number of elements in *array* after the addition.

categoryBonus(*list*, *category*)

Returns the total of category bonuses for items that are in *category* in *list*. Multiple categories may be specified, separated by semicolons.

ceil(*x*)

The "ceiling" of a floating point number. It return the next integer away from zero. For example, **ceil**(1.2)" returns 2, "**ceil**(1)" returns 1, "**ceil**(-1)" returns -1 and "**ceil**(-1.2)" returns -2.

This does not implement the definition of a true mathematical ceiling, because for most cases in RPG construction the mathematical definition is counterintuitive and less convenient. The mathematical version is defined by the following: " $x > 0 ? \text{ceil}(x) : \text{integer}(x)$ ".

checkRequirements(*listName*, *checkExclusions*, *isInsert*)

This can be called only in the context of an item. Checks the requirements of the current item, assumed to be in the list named *listName*. If the requirements are satisfied, returns a non-zero value.

If *checkExclusions* is non-zero, other items are checked to see if their requirements would exclude the addition of this item. If *isInsert* is non-zero, the item is assumed to not already be present in the list (this would be an insertion, which has an effect on requirements for number of items).

childiteminfo(*child*, *reference*)

This can be used only in expressions related to sublists. For a sublist, returns a reference to the referenced item information for the child item. The first child item is referred to as 1, etc. The reference is formatted as a option reference in an adjustment option (p. 87). See *iteminfo* (p. 174) for more details.

chrval(*number*)

Converts the specified number into the corresponding ASCII character. For example, **chrval**(65) returns "A".

clearMap(*mapName*)

Removes all entries from the map named *mapName*. See **putMap** (p. 177).

concat(*s1*, *s2*)

concat(*s1*, *delim*, *s2*)

Two-argument version: concatenate *s1* and *s2*, returning the result. **concat**("ab", "cd") returns "abcd".

Three-argument version: concatenate *s1*, *delim* and *s2* if *s1* has a value other than the empty string. If *s1* is the empty string, return *s2*. **concat**("a", " ", "b") return "a, b", while **concat**("", " ", "b") returns "b".

ConfigParam(*parameterName*)

Reads a game system configuration parameter (p. 247) from the preferences for the current user. If the parameter is not defined by the preferences file for the current game system, the user will not be able to change the parameter explicitly; it can only be set by **setConfigParam**.

countItems(*list*, *value*, *operation*, *attribute*)

The count of items in *list* that satisfy the condition specified by *value*, *operation* and *attribute*. If *list* is qualified by a category (specified by placing a colon between the list and category, i.e., "Skills:Combat"), only items in that category in that list are examined. To indicate items that have a specific option value, specify "Skills:@Option Name='Option Value'".

An attribute of each item, specified by *attribute* (a text string specified by the same name as for **foreach** (p. 194), excluding the '@'), is compared against *value* with the operation specified by *operation*.

The operations are: -1, less than; 0, equal; and 1, greater than. For example, "**countItems**('Skills', 0, 0, 'cost')" returns a count of all the items in the Skills list that have a cost of 0. "**countItems**('Skills:Melee', 19, 1, 'level')" returns a count of all Combat skills that have level greater than 19.

countItems('Skills:@Secondary Skill="Karate", 1, -1, 'cost') returns the number of entries in the Skill list that have the Secondary Skill option with a value of Karate and have a cost of less than 1.

To count all items in the specified list (and category, if specified), specify 0 for the *attribute*:

`countItems('Skills:Combat/Weapon', 0, 0, 0)` returns the number of items in the Combat/Weapon category of the skills list.

dialogFieldValue(*dialogName*, *fieldName*)

Return the text of the dialog field in the specified dialog. Variables referencing this function will not be automatically updated when the field changes.

dround(*x*)

Round the floating point value down to the nearest integer. "**dround**(1.6)" will produce 2. "**dround**(1.5)" will produce 1.

dupstr(*string*, *number*)

Duplicates *string* *number* times and returns that string. For example, **dupstr**("x", 5) will return "xxxxx".

eval(*expression*)

Evaluates the expression in the string and returns the value. This is useful in @foreach (p. 194) loops when an item names another variable explicitly. For example, if you wanted to iterate through the list of all skills in the data sheet, displaying the default value of each skill (which happens to be the underlying attribute, which is stored in the category), you could use the following code:

```
@foreach(*Skills)
  %@name%: %eval(@category)%
@endfor
```

evaluateDefault(*arg*)

Evaluates the default value for the current item. The *arg* parameter should be zero.

fileExists([*directory*,] *fileName*)

Return true if the file specified by *fileName* exists. The optional *directory* argument, if present, indicates the name of the directory to look in.

findItemValue(*list*, *operation*, *attribute*)

Search for an item in *list* using *operation* to find an extreme of *attribute*, similar to **countItems**. *List* is the name of a character sheet list. A category may also be specified, separated by a colon, as well as an @Option Name=value expression. The *operation* is either `max` or `min`.

An attribute of each item, specified by *attribute* (a text string specified by the same name as for `foreach` (p. 194), excluding the '@'), is compared against the attributes of the other items and the minimum or maximum value is returned.

For example, `findItemValue('Skills:Electronics', 'max', 'basevalue')` will find the maximum level (ignoring any bonus) of an item in the Electronics category of Skills.

format(*formatString*, *arg1*, ...)

Format a string with arguments, sort of like the C "sprintf" function. The "format string" is returned, with each instance of "%d" (for numbers) or "%s" (for strings) in the string replaced by the corresponding argument. For example, "**format**("%dd%d", 2, 6)" would return "2d6". If you wish to include a "%" in the format string, use "%%". If you have more "%d"s in the format string than arguments, the return value is undefined.

The "%n" format returns a number formatted with separators each three sets of digits. The decimal point and group separator are defined in the Windows Control Panel. A number after the '%' indicates the number of positions after the decimal point. For example, "**format**('\$%2n', money)" where money = 100000.45 will return "\$100,000.45". Note that floating point numbers are prone to roundoff and may not always produce the expected value.

A + after the % will print the sign of the number. For example, `'format("%+d", 10)'` will return "+10" and `'format("%+2n", -4)'` will return "-4.00". A - after the % will also print the sign, but the sign will be - for zero (it is + for zero if the + indicator is used). If a number is specified ("%2d") that many decimal places will be displayed. If the number of decimal places is not indicated, fractions may be included in the rendered number (if the Preferences indicate that fractions may be used).

The "%x" format assumes the value is a number and displays it in hex using lower-case letters. The "%X" format displays a hex number in upper-case letters. The "%o" format displays an octal number, and the "%b" format displays a binary number.

For example, `'format("%x, %2X, %o, %8b", 10, 10, 10, 10)'` would produce "a, 0A, 12, 00001010".

The limit on the number of format specifications in the format string is 10.

float(x)

Convert an integer number to floating point. `"float(2)"` produces 2.0.

getAssoc(array, key)

Get the value of the "association" for the value *key* from *array*. Associations are added to an array with **putAssoc** (p. 176). This allows for easy lookup of dynamically created data.

getCategoryList(listName, catArray [, costArray [, countArray]])

Get a list of categories in the specified list. The *catArray*, *costArray* and *countArray* arguments must be arrays that are already defined. The last two arguments are optional. The names of all the different primary categories in the named list will be placed in *catArray*. The subtotals of items costs in each category will be placed in *costArray*. The number of items in each category will be placed in *countArray*.

getFileValue(fileName, expression)

Evaluate an expression based on the values found in another character sheet. When this function is evaluated, the file specified in *fileName* is opened and *expression* is evaluated in the context of that file. For example, if you wanted to obtain the value of the `total` variable, *expression* would be `"total"`. Any arbitrary expression can be specified. This function is most useful in conjunction with options that are file names. For example,

```
getFileValue(optValue('File Name'), 'avail+exp')
```

will return the value of the `avail + exp` variables from the file named in the File Name option associated with an item.

The values are determined when the file is first referenced. If the file is open in **GURPS Character Builder**, the values are updated when the file is saved.

getGlobal(name)

Get the value of the global variable (p. 181) named in the string *name*. For example, `getGlobal("currentFile")` gets the index of the current file being processed in a multiple file filter or update.

getMap(mapName, entryName)

Retrieves the value associated with *entryName* in the map named *mapName*. See **putMap** (p. 177).

getSourceDir (reserved)

Get the name of the **GURPS Character Builder** source directory (p. 17). You should pass the value 0 in for *reserved*.

inCategory(category)

Return true if the current item is in the specified *category*. A list of categories separated by semicolons may also be specified for *category*, in which case the item must be in all the specified categories. An item is considered to be in a category if that category appears in its category list. Categories beginning with "*" indicate "hidden" categories; thus, category names may not begin with "*".

Return false if there is no current item, or *category* is not among the categories listed for the item. Options may contain additional categories, so **inCategory** is recommended over searching the category string directly by obtaining it through **itemInfo**.

If the item's list of categories is "Language;Archaic", the following values result:

<code>inCategory("Language")</code>	true
<code>inCategory("Archaic")</code>	true
<code>inCategory("Language;Archaic")</code>	true
<code>inCategory("Language;World")</code>	false
<code>inCategory("Combat")</code>	false

index(array, value)

Produce the index into the *array* of the *value*, a value from 1 to the number of elements in the array. If the value is not found in the array, index produces -1.

integer(x)

Produce the integer part of a floating point number, truncating toward zero. "**integer**(1.2)" returns 1, while "**integer**(-1.2)" returns -1. This is similar to the mathematical "floor" function, but works slightly differently for negative numbers. A "true" mathematical floor function can be defined with the following: "**x**>0?**integer**(x):**ceil**(x)".

itemCheckExpression(level)

This function only works in expressions in conjunction with items. Return true (non-zero) if the current item's check expression is satisfied with a value of *level*. This is intended to be used within the `item ... enditem` construct in command scripts, in item options and similar contexts. Items with no check expression always return true.

itemCostFunction(level)

This function only works in expressions in conjunction with items. Return the cost of the item for the specified *level*. This is intended to be used within the `item ... enditem` construct in command scripts, in item options and similar contexts. Items with no cost formula always return the current cost. This is the base cost of the item, excluding any additions or multipliers from options.

iteminfo(reference)

This function works only in expressions in conjunction with items. Return the named information about the item. The reference is formatted in the same fashion as text in adjustment options (p. 87). For example, **iteminfo**("@c@") returns the cost of the item, and **iteminfo**("Damage") returns the value of the "Damage" option. For numeric expressions to be correctly evaluated, the first character of the reference must be recognizable as a part of an expression (for example, a number or a left parenthesis).

listDirectory(array, dir, patterns)

List a directory and place the file names found in an array. *Array* will receive the names of the files. *Dir* is the directory to be listed. *Patterns* is a semicolon-delimited list of patterns. For example,

```
listDirectory(files, getSourceDir(0), "*.mds;*.cds")
```

will place the names of all files in the **GURPS Character Builder** source directory that end in `.mds` and `.cds` into the array `files`. The files are not sorted; the **sortElements** function can be used to sort arrays.

listentries(listName)

The number of entries in the named list. This is most useful in conjunction with **listiteminfo**.

listiteminfo(listName, itemIdentifier, reference)

If *itemIdentifier* is a number, return information about the *itemIdentifier*'th item in the list named *listName*. If *itemIdentifier* is a string, return information about the item with that name in *listName*. The *reference* is formatted in the same fashion as text in adjustment options (p. 87). This can be used in conjunction with the **listentries** (p. 181) function to loop through all items in a particular list to find items that meet certain criteria. If *listName* begins with an asterisk ("*") the list in the data sheet is accessed, allowing you to obtain information about items that aren't selected in the current character sheet. In this case, the *itemIdentifier* should be the item name, not an index.

listMap(mapName, delimiter)

List the names of all entries in the map named *mapName*, separating each entry from the next with *delimiter*. Entries are added to the map with **putMap**. The entry names are returned sorted in alphabetic order.

Given the following sequence of **putMap** calls:

```
n = putMap("attack", "Longsword", 1);
n = putMap("attack", "Knife", 1);
value = listMap("attack", "; ");
```

The **listMap** call would return "Knife; Longsword".

listOptions(*type*, *name*, *delimiter*)

listOptions(*type*, *name*, *array*)

The first form returns a string containing the options for the current item, separated by the indicated delimiter. The second form places the specified option information into *array*.

The value of *type* indicates how the options should be listed:

optName	Display names of options.
optActive	The value "1" if the option is "active" (it is not a toggle option, or it is a toggle option and the checkbox is on), or "0" if the option is "inactive" (it is a toggle option with the checkbox off).
optDisplay	The value "1" if the option is displayed on the output, and "0" if not.
optExp	The value of the expression. For adjustments the values are expanded out.
optKeepValue	The value "1" if the option has the flag Keep Value on Conversion set, otherwise "0".
optOrigName	The original name of the option. This can be used to determine whether the option was renamed and what the original option name was.
optValue	Values of options (costs).
optTextValue	Text part of values (descriptive part of cost).
optQualifier	The qualifier on the option.
option	Names and values of options, including costs.
optRawType	The raw option types.
optRawValue	The raw value of the option (not generally useful, as it must be parsed to extract information).
optType	The option types (p. 201).

The *name* is the name of the option to display. If the empty string (""), then all options will be displayed. For example, **listOptions**("optTextValue", "Sense", ", ") would display "Normal Sight, Normal Hearing, Smell" if you had three Sense options with those values.

log(*x*)

The natural logarithm (base e) of *x*. **log**(2) produces 0.6931.

log10(*x*)

The logarithm (base 10) of *x*. **log**(100) produces 2.

lookup(*array*, *key*)

Array is an array (p. 167) defined in the character template, which contains pairs of values. The first value in each pair must be numeric, while the second value can be a number or a string. The values must be sorted in the array (lowest first). Lookup will examine the first value of each pair, and if *key* is less than

or equal to that value, the function will produce the second value of the pair as the result. This is useful for converting numbers into string values which can be used for display purposes in dialogs. For example, let's say you have an array named "dmgdice" with the values "5, '1d', 10, '2d', 15, '3d', 20, '4d'". The expression "lookup(dmgdice, 5)" produces the string '1d'. "lookup(dmgdice,2)" also produces '1d'. "**lookup**(dmgdice, 8)" produces '2d'. If *key* is larger than the last pair in the array, the last value is returned, so in this example "**lookup**(dmgdice,400)" would produce '4d'.

The elements of the array may be numbers or strings, so you can also use **lookup** to search for string translations. Make sure that the key values are sorted in alphabetic order.

max(x, ...)

The maximum of a list of values. "**max**(2, 4, 3)" produces 4.

min(x, ...)

The minimum of a list of values. "**min**(2, 4, 3)" produces 2.

optAvailable(*name*)

Returns non-zero if an option called *name* is available in the list of options.

optCount(*name*)

Usable only in the context of an item, this returns the number of options on the item named *name*, including any inherited options. If *name* is the empty string ("" or ' ') all options are counted.

optNumberValue(*optname*)

This can be used only with expressions in conjunction with items. Returns the number part of an option whose value consists of a text label and a numeric value.

optpresent(*option*)

Usable only in option expressions, expressions in item format strings and sublist cost formulas. Returns TRUE (non-zero) if the named option is present in the item, and FALSE (zero) if it is absent.

optTextValue(*optname*)

This can be used only with expressions in conjunction with items. Returns the text part of an option whose value consists of a text label and a numeric value.

optvalue(*optName*)

optvalue(*optName*, *index*)

optvalue(*listName*, *itemIdentifier*, *optName*)

Return the value of an option. The one- and two-argument versions can be used only with expressions in conjunction with items. The one-argument version returns the text of the value of the first option in the current item named *optName*. The two-argument version returns the text of the value of the *index*'th option named *optName* in the current item. The three-argument version returns the option specified by *optName* for the item identified by *itemIdentifier* in the list *listName*. *ItemIdentifier* can be either the item name (a string), or an index into *listName* (a number).

An option reference recursion error occurs if the an expression option refers to its own value.

parentiteminfo(*reference*)

This can be used only conjunction with items. Returns the referenced item information for the parent item. The reference is formatted as a option reference in an adjustment option (p. 87). See iteminfo (p. 174) for more details.

putAssoc(*array*, *key*, *value*)

Puts an "association" for the value *key* in *array*. This allows for easy lookup of dynamically created data. When you use the **getAssoc** function it will return the *value* that was associated with *key*. There must be an even number of elements in *array* at all times. If you add an association for a *key* that's already present, it will replace the existing association. The value returned is the number of elements in the array (which is two times the number of associations present).

For example,

```
n = putAssoc(counts, "Skills", getAssoc(counts, "Skills")+1)
```


adds 1 to the number associated with the value "Skills" in the array.

putMap(*mapName*, *entryName*, *value*)

Associates an *entryName* with *value* in the context of the map named *mapName*. Any number of maps may be defined. If no entry exists for the named entry in the map, **putMap** returns 0. The case of the *entryName* is ignored. Calling `n = putMap('map', 'Value', 1);` is the same as `n = putMap('MAP', 'value', 1);`.

A map is a list of name/value pairs. Storing and retrieving values in maps is relatively efficient. Maps are useful for associating extra information with items or other named entities. For example, a weapon might have several different kinds of bonuses bestowed by other items in the game system: attack, damage and defense bonuses from several different skills and advantages. To store these bonuses you could define three different maps: "attack", "damage" and "defense". To add the attack bonus, the following function could be used:

```
func addBonus(adding, map, weapon, bonus)
{
  return putMap(map, weapon, getMap(map, weapon)+(adding ? bonus : -bonus));
}
```

This could be used in an adjustment for the items that provide bonuses for the weapons. The adjustment might be:

```
attackMap+addBonus($adding$, "attack", "Knife", 1)
```

For large numbers of entries, maps are more efficient than associative arrays populated with **putAssoc**. Any number of maps may be created dynamically, while associative arrays must be explicitly declared. If you have no need to sequentially search through an array, a map is preferable to an associative array.

qindex(*index*, *v1*, *v2*, *v3*, ...)

"Quick index" into the argument list, which consists of an *index* and a list of values to index into. If *index* is 1, the value *v1* (the second argument) is returned. If *index* is 2, *v2* (the third argument) is returned, etc. If the *index* is less than 1, the first value (second argument) is returned. If the *index* is greater than the number of values, the last value is returned. You should avoid more than 20 arguments to this function -- more may fill up the expression evaluation stack (which is currently at 40 variables). If you need more, you should create an array variable instead.

rand(*x*)

Generate a random number in the range 1 to *x*, where *x* is an integer value. "rand(6)" will produce a number between 1 and 6.

removeallelements(*array*)

Removes all the elements from *array*. The number of elements (which will always be 0) is returned.

removeelement(*array*, *index*)

Removes the *index*'th element of *array*. *Index* is a value from 1 to the number of elements in the array. Returns the number of elements in the array after the removal. If an attempt to remove a non-existent element is made (an *index* less than 1 or greater than the number of elements), -1 is returned.

replaceString(*source*, *pattern*, *replacement*)

Replaces all occurrences of the string *pattern* with the string *replacement* in the string *source*. The comparison ignores case. For example,

```
replaceString("Hunted by CIA", "Hunted by", "Enemy:")
```

would return the string "Enemy: CIA".

If the *pattern* string is surrounded by "/" characters, the pattern is a *regular expression* (p. 11). For example,

```
replaceString("This/Is-A Test.", "/[^a-zA-Z]/", '')
```

returns the string "ThisIsATest". To include matched pieces of the string in the replacement, surround the parts in the regular expression pattern with parentheses. Put `\digit` in the replacement, where *digit* is the number (1-9) that matches the number of the pair of parentheses. For example, the following

```
replaceString("Yours,Mine,Yours\Mine and Ours", "/([^\s]),/", "\1!!!")
```

will return the value "Yours!!!Mine!!!Yours\Mine and Ours". This works by matching the `,` character preceded by any character but `\`, then replacing that string with that character and `!!!`. Since the `,` in `Yours\, Mine and Ours` is preceded by a `\` it isn't matched and is therefore not changed.

To return the entire string that matched in the destination, use `\0`. Also note that if you're using backslashes in a filter script, you will need to double them up.

replaceToken(string, token, value)

Replace all instances of *token* occurring in *string* with *value*. A token is essentially a "word." If the token occurs within the confines of another "word" it is not replaced. Case is ignored. For example,

```
replaceToken("This was a waste of time.", "was", "is")
```

would return "This is a waste of time". The sequence of characters "was" is the word "waste" will not be changed by the replacement.

round(x)

Round the floating point value up to the nearest integer. "round(1.2)" will produce 1. "round(1.5)" will produce 2.

rolldice(diespec)

rolldice(b, n, s)

The one-argument version returns a random value (or set of values) based on *diespec*. For example, "**rolldice**('2d10-2')" will produce a value between 0 and 18. The call "**rolldice**('3*4d4')" will produce a list of numbers: "11,12,9". The *diespec* has the same syntax that die specifications (p. 26) for the die roller have.

The value returned by the one-argument version is a string (because a *diespec* can specify a list of random numbers be generated). To convert a single random value to a number use the **integer** or **round** function.

The three-argument version rolls *n* dice with *s* sides and sums ("keeps") the top *b* dice. "**rolldice**(3, 9, 6)" will roll 9 six-sided dice and pick the top 3 values, producing a number between 3 and 18, with a much higher average value than would be produced by just rolling three dice. "**rolldice**(3, 3, 4)" would produce a value between 3 and 12, with an average value of 7.5. Zero is returned if the number of dice or sides is zero or less. The three-argument version always returns a number.

setConfigParam(name, value)

Set the configuration parameter (p. 247) for the current game system called *name* for the current user to *value*. Zero is returned if there is an error (no such parameter named, etc.), and non-zero is returned on success. If the configuration parameter is not defined in the preferences for the game system associated with the current character sheet, it can only be read and set by **configParam** and **setConfigParam**.

setGlobal(name, value)

Set the global variable (p. 181) named in the string *name* to *value*. Zero is returned on error, and one on success.

Return the number of elements in *array*.

setElement(array, index, value)

Set the *index*'th element of *array* to *value*. The array must previously exist. If a "hole" is left in the array, it will be filled with zeros. The return value is the index of the value. A zero value is returned if you attempt to set an illegal element (with a zero or negative index, or an index greater than 65536).

setVariable(varname, value)

Set the variable named in the string *varname* to the specified *value*. For example,

`setVariable("ParryWeapon", listItemInfo("Equipment", i, "@name@"))` sets the variable `ParryWeapon` to the name of the *i*'th item in the Equipment list. If the assignment was successful, TRUE is returned. If the assignment could not be made FALSE is returned.

sizeof(array)

sortelements(array, ascending)

Sort the elements of *array* in ascending order if *ascending* is true (non-zero). If *ascending* is false the elements are sorted in descending order. The number of elements in the array is returned.

split(string, pattern, "varname1", "varname2", ...)

Split the "fields" in *string* up into the variables named. The *pattern* is the string that separates string into fields. The number of fields found is returned. For example,

```
@assign _nfields=split(@item, " ", "_cost", "_name")
```

when used in a `@foreach` loop in a script will split up the item string (referenced by `@item`), into the variables named `_cost` and `_name`. The `@item` string returns the fields displayed in the list, separated by tabs. The " " pattern is a single tab character. Note that the variable names *must* be surrounded by quotes ("_name").

If the third argument is an array, the elements of the array are set to the result of the **split**.

If the *pattern* is surround by "/" characters, it is a regular expression (p. 11). For example,

```
@assign _nfields=split("a, b,c, d", "/", */", _array)
```

would put the values "a", "b", "c" and "d" into the four elements of `_array`.

sqrt(x)

Returns the square root of the argument. For example, "sqrt(4)" returns 2.

strindex(s1, s2)

Returns the index of string *s2* in string *s1*. The indexes start at 1. If *s2* is not found in *s1*, **strindex** will return 0. For example, '**strindex**("abcde", "bcd")' will return 2.

If *s2* is surrounded by "/" characters, it is assumed to be a regular expression (p. 11). For example, `strindex("abc123", "[0-9]/")` will return 4.

strlen(string)

Return the length of the named string.

substr(string, index, length)

Returns a "substring" of the first argument, which is a string. The index is the offset into the string. The first character is numbered 1. The length is the number of characters to copy. For example, **substr**("abcd", 2, 2) will return "bc". If the index is less than 1 or greater than the length of the string, the empty string will be returned. If the indicated substring is longer than the actual string, only as much of the actual string that exists will be returned.

tolower(string)

Return the lower-case equivalent of *string*.

totalItems(list, value, operation, attribute)

The total cost of items in *list* that satisfy the condition specified by *value*, *operation* and *attribute*. These arguments are interpreted exactly as for **countItems**. For example,

```
totalItems('Skills:Combat', 0, 1, 'level')
```

returns the total cost of all items in the Combat category of the Skills list whose level is greater than 0.

The list can also specify an option and associated value of the form `Skills:@Option`

`Name=expression`.

totalItemValues(list, attribute)

Return the sum of the specified *attributes* on the items in *list*. The list can be either a list name, or a list name followed by a colon and a category. For example,

```
totalItemValues('Skills:Combat', 'level')
```

returns the sum of all skill levels in the Combat category. This is different from the **totalItems** function because that function will only return the sum of the *costs* of the items that qualify.

toupper(*string*)

toupper(*string*, *initCap*)

One-argument version: return the upper-case equivalent of *string*. Two-argument version: if *initCap* is non-zero, return *string* with the first letter of each word made upper case. If *initCap* is zero, return the upper-case equivalent of *string*. A word is defined as a sequence of characters starting with a letter and ending with a blank or the end of the string.

For example, `toupper("Bug-eyed monster")` returns "BUG-EYED MONSTER". The call `toupper("Bug-eyed monster", 1)` returns "Bug-eyed Monster".

variteminfo(*varname*, *reference*)

Return information about the item associated with the variable *varname*. The reference is formatted as a option reference in an adjustment option (p. 87). See `iteminfo` (p. 174) for more details. The name of the variable is specified in the **Variable Name** field (p. 75) on the Basic tab of Edit Properties dialog. If no item is associated with a variable, zero is returned.

Global Variables

Global variables are defined in the "game system context," which means that all character sheets using that same game system can access those global variables. This context is also used when no other context is available (for example, when displaying entries in the Available items dialog).

The **getGlobal** (p. 173) and **setGlobal** (p. 178) functions can be used to read and modify the values of global variables.

Global variables persist only as long as the current session; their values are lost when **GURPS Character Builder** exits.

Global variables are intended to be used to remember state across the processing of different character sheets. For example, if information must be shared during the filtering of multiple character sheets, global variables are a natural repository.

Predefined Global Variables

currentFile When processing multiple filtered files, this variable contains the index of the current file, starting from zero. That is, **currentFile** is 0 for the first file, 1 for the second, etc. For example, the following script fragment emits a gray background color command for the third, fourth, seventh and eighth, etc., files, and a blank **@bgcolor** command for the first, second, fifth and sixth, etc., files.

```
@if integer(getGlobal("currentFile") / 2) mod 2 = 1
@bgcolor 224 224 224
@else
@bgcolor
@endif
```

outputFileName

When processing filters, this variable contains the name of the output file name if the target of the filter is a file being saved to disk. This value is set to 0 if no output file is being written.

listiteminfo example

This example determines the total cost of all items in the named list and category.

```
catTotal(list, category) = {
  local i, n, cost;

  n = listentries(list);
  i = 1;
  cost = 0;

  while (i <= n)
    if (strindex(listiteminfo(list, i, "@cat@"), category))
      cost = cost + listiteminfo(list, i, "@c@");
    endif
    i = i + 1;
  endwhile

  return cost;
}
```

Note that if you use a function such as this, it will not automatically be called when an item is added to the list. If you wish automatic update, you need to reference the list's variable. For example, if you wanted **attackTotal** to update every time the Powers list was changed, you could define it this way:

```
attackTotal = catTotal("Powers", "Attack") + powers*0
```

Operands

Operands are the arguments for operators (p. 169) in **GURPS Character Builder** expressions. There are three basic types of operands:

Numerical values

Numbers, such as 2, 3.14159. These can be integers or floating point numbers. Integers must be in the range -32768 to 32767. Floating point numbers may be in the range from approximately 1E-38 to 1E38.

Strings

String values are specified with quotes around them: "d5" or 'Red'. You may use either single or double quotes. Use strings with care: numerical operators cannot deal with strings and may cause problems.

Variables

Variables may be defined in a number of different ways. Variable names must start with a letter and may include letters, numbers and the '_' character.

You can define variables to be the values associated with fields that the user can edit in a dialog box, or predefine them in the character sheet template. If you reference a variable that has never been set, its value will initially be zero.

Certain predefined functions (such as lookup) will require an array as an argument.

Procedures

Instead of defining a variable or function as a simple expression, you can define them as procedures. A procedure is a sequence of programming-language like statements, including if (p. 183), elseif (p. 183), else (p. 183), while (p. 182) return (p. 184) and assignment (p. 184) statements. Local variables (p. 183) may also be declared for use within the body of the procedure.

A procedure must start with a '{' and end with a '}'. The last statement in a procedure must be a return (p. 184). Procedures defined within function bodies may access the function arguments. Statements must be separated by semicolons.

For example, let's say you have a non-linear function for computing the cost of a particular attribute, -5 per point below 10, 10 per point between 10 and 15, 20 points between 15 and 20, and 5 per point above 20, you could write the following procedure:

```
attrCost(attr)
{
    local cost;
    if (attr < 10)
        cost = (attr - 10) * 5;
    elseif (attr < 15)
        cost = (attr - 10) * 10;
    elseif (attr < 20)
        cost = 50 + (attr - 15) * 20;
    else
        cost = 150 + (attr - 20) * 5;
    endif
    return cost;
}
```

If you code an "infinite loop" (where the procedure never ends) you can press CTRL+BREAK to break out of the loop.

While in Procedures

The general structure of a while statement is:

```

while (expression)
    ...
endwhile

```

The parentheses around the expression is required. The expression is evaluated and if found true (non-zero), the statements between the while and the endwhile are executed.

Care must be exercised to ensure that the while loop terminates. Don't forget to increment your control expression -- if you do, your loop could run forever. If you happen to write an "infinite loop" (where the while never ends) you can press CTRL+BREAK to break out of the loop.

The following is an example of a procedure that uses a while statement to determine the value of the child item with the greatest cost.

```

{
    local maxCost, i, children, childCost;
    children = iteminfo("@nc@");
    maxCost = 0;
    i = 1;
    while (i <= children)
        childCost = childiteminfo(i, "@c@");
        if (childcost > maxCost)
            maxCost = childCost;
        endif
        i = i + 1;
    endwhile
    return maxCost;
}

```

If in Procedures

The general structure of an if statement is:

```

if (expression)
    ...
elseif (expression)
    ...
else
    ...
endif

```

The parentheses around the expressions are required. The expressions are evaluated top to bottom. If the (expression) evaluates to true (non-zero), the statements after the if and before the next elseif, else or endif are executed. When the elseif, else or endif is reached, execution continues at the end of the if statement (after the endif). If the expression is false (zero), the expression in the subsequent elseif is evaluated, and its statements executed if that expression is true. If none of the expressions is true, the statements in the else (if present) are executed.

See the procedures (p. 182) introduction for an example.

Local Variables in Procedures

Variables defined by items or associated with edit fields are "global." That is, they are accessible to all functions and other items. Local variables are available only to the current procedure. Only local variables can be changed in procedures, to prevent "side effects." One procedure might "accidentally" change a global variable.

Local variables are declared by the following statement, which must immediately follow the opening '{' of the procedure:

```

local var1, var2;

```

The variables may be named anything you like, though they must start with a letter and must contain only letters, numbers and the '_' symbol.

Local variables are always set to zero whenever a procedure starts. The value disappears every time the procedure exits. For an example, see the description of the assignment (p. 184) statement.

Assignment in Procedures

Assignments have the following form:

```
localVar = expression;
```

The semicolon is required after the expression. Creator procedures are "pure" functions: they may have no side effects. That means only local variables may be assigned values.

For example, let's say you wanted to define a rounding function that rounded up if the fractional part of the number was greater than 0.4:

```
myround(arg)
{
  local fracPart, intPart;
  intPart = integer(arg);
  fracPart = arg - intPart;
  if (fracPart > 0.4)
    return intPart+ 1;
  endif
  return intPart;
}
```

You can use local variables to store the temporary variables you created during the execution of the function.

Return in Procedures

A return indicates the value of the procedure. It may appear anywhere in a procedure, but the last statement in a procedure must be a return. The format is:

```
return expression;
```

The semicolon must follow the expression.

Creating Filter Files

A text filter file is a normal text file, which contains text and a way to reference (p. 185) character sheet data. You can create a filter file with **GURPS Character Builder (File | New...** and click **Text File**), Windows Notepad or any Windows word processor (such as Write) that saves files out as normal Windows text files. You must make sure that no special document formatting codes for the word processor are stored in the filter file.

Filter files are used to print (p. 6) and save (p. 3) the contents of a character sheet as text, as well as copy (p. 9) the contents of the character sheet to the clipboard in a text format.

Filter files use an `.flt` file extension. The first command in the filter file must be `@gamesystem` (p. 198). Other commands (p. 187) iterate through item and option lists, perform calculations, etc.

Formatting Commands

Formatting commands (`@font`, `@bgcolor`, `@tabs`, etc.) change the format of the rendered text when printed or displayed by **GURPS Character Builder**. However, such formatting is lost when text is copied to the clipboard, unless you use a special filter that produces formatted text such as RTF or HTML.

Macros

Filter files may use data sheet macros (p. 226). For example, you may create macros that define a common way of printing options, placing those macros in an include file. You can then include that file with the `$$include` (p. 233) macro command. Thereafter in the filter you can invoke macros defined in the include file.

Referencing Character Sheet Data

Any normal text in a filter is simply output to the destination, be it a file, the clipboard or the printer. You can reference character sheet data in the filter by using special commands and characters in the filter. Output can be turned off and on with the `@output` (p. 202) command.

You can reference character sheet data in text boxes of print templates and in copy filters. These references are indicated with special characters ("`%`" and "`?`"). When these references occur embedded in normal text, the references are replaced with the values they represent. Filter commands (p. 187) are also useful in filters and print templates.

Referencing Variables

A variable name within percent symbols (`%total%`, for example) displays the value of that variable. Within a `@foreach` (p. 194) there are some additional variables available that reference values for the items in a list. There are also "global variables" that are accessible via the `setglobal` (p. 178) and `getglobal` (p. 173) functions.

Referencing Expressions

You can put any valid **GURPS Character Builder** expression (p. 169) within a `%%` reference. If you wish to include a `%` within a pair of percent signs, escape it with `\`. For example,

```
%format('\%ln', weight)%
```

will display the value of `weight` to one decimal place. Without the escaping `"\"`, the `%` in the format string would terminate the expression reference. You can also include carriage return ("`\r`") and new line line ("`\n`") values in the expression reference. Similarly, if you wish to include a `"\"` you must enter `"\"` to tell the filter processor that a single backslash is required. Any other characters will be left verbatim.

When printing blank character sheets (p. 187) the results of `%%` references are ignored.

Special Variables

There are special variables that are automatically available in all character sheets.

`@blankCharsheet`

True (non-zero) if the user checked the **Print Blank Character Sheet** checkbox when asked for a print template. Otherwise false (0). This value should be checked in print

templates that have conditional display of text to make sure that the correct text is output when the user wants to print a blank character sheet (p. 187).

@datasheets	A comma-delimited list of the data sheets used by this character sheet.
@date	The current date.
@datecreated	The date the character sheet was created.
@details	This can be used in filter expressions (such as in @if commands). It is true if details are present in the character sheet.
@detailstext, @detailsrtf, @detailhtml	The contents of the Details window (p. 13), rendered as plain text, RTF and HTML. When writing a filter for RTF and HTML, you should use the corresponding variable and surround it with @translate (p. 206) commands to turn off escaping of the special formatting characters.
@filename	The name of the character sheet file.
@gamesystem	The name of the game system for the current character sheet.
@listname	The name of a list preceded by "@" returns the number of entries in that list.
@`dialog.field`	The "numerical" value of the specified field in the dialog. Text-only values are treated as 0 if empty, otherwise their value is 1. If you are simply referencing a field in %, don't bother with the @`...` -- see below.
@notes	The contents of the Notes (p. 13) window.
@pageNumber	(Only valid while printing through print templates). The number of the current page of the print template.
@ruleset	The name of the selection rule set for the current character sheet. The empty string, if no rule set is selected.
@sheettype	The character sheet type.

Conditional References

You can examine variables and conditionally output different text based on those variables. Four question marks set off the condition. Between the first two is the expression (p. 169) to check. Between the second and third ?'s is the text to output if the expression is true (a non-zero value). Between the third and fourth ?'s is the text to output if the expression is false (a zero value). For example,

```
ST ?x_st=st?%st?%st?/%x_st?
```

For x_st = 10 and st = 10, this would be output

```
ST 10
```

For x_st = 11 and st = 10, this would be output

```
ST 10/11
```

Referencing Dialog Fields

You can reference the values of fields in dialogs in the following manner:

```
%@dialog.field%
```

where `dialog` is the name of the dialog and `field` is the name of the field. The name of the dialog can be abbreviated. If you have a dialog named "Information" and a field named "appearance" you could abbreviate it to "%@info.appearance%".

Referencing List Members

The `@foreach` (p. 194) command can be used to reference list members.

Referencing Options for Items

The `@options` (p. 201) command iterates through all options for an item.

Referencing Game System Preferences

You can use the `ConfigParam` (p. 171) function to reference game system-specific user preferences (p. 247).

Escaping Characters

If you actually want to use one of the special symbols used in filters into the output, you can "escape" it by preceding it with a backslash. For example, if you wanted to include a "?" in the output text, you would type: "Are you there\?".

Special Characters

You can force a new line with the "\n" sequence.

Continuing Lines

You can continue a line (prevent a new line from being started) by ending the line with a "\" character.

Comments

The "#" character indicates the beginning of a comment. All text after the "#" is ignored.

Filtering for Blank Character Sheets

When a blank character sheet is printed through a print template the results of any %% references are discarded.

When you design the filters for text blocks in print templates, check the value of the `@blankCharsheet` variable to ensure proper display of text blocks when blank character sheets are being printed. Most frequently the best thing to do is output nothing, but in some cases you may wish to display underlines or space characters to leave space for handwritten entries on the blank character sheet.

For example:

```
Sex: ?@blankCharsheet?_____?%@Information.sex%?
Height: ?@blankCharsheet?_____?%@Information.Height%?
Weight: ?@blankCharsheet?_____?%@Information.bodyWeight%?
```

When a character sheet is printed normally you'll get a display similar to this:

```
Sex: Male
Height: 5'9"
Weight: 145lbs
```

When a blank character is printed you'll get something similar to this:

```
Sex: _____
Height: _____
Weight: _____
```

Filter Commands

Filter commands must stand alone on a line, preceded by optional whitespace. They consist of a keyword that starts with an "@" symbol (such as `@if`).

Commands

<code>@array</code> (p. 188)	<code>@landscape</code> (p. 200)
<code>@assign</code> (p. 188)	<code>@leftindent</code> (p. 200)

@bgcolor (p. 189)	@message (p. 200)
@bold (p. 189)	@need (p. 200)
@bolditalic (p. 189)	@options (p. 201)
@clipformat (p. 189)	@output (p. 202)
@description (p. 190)	@parinfo (p. 202)
@dest (p. 190)	@plain (p. 203)
@exit (p. 191)	@samepage (p. 203)
@extension (p. 191)	@sheettype (p. 203)
@fail (p. 192)	@sortexp (p. 204)
@firstindent (p. 192)	@sortorder (p. 204)
@font (p. 192)	@sub (p. 204)
@fontinfo (p. 193)	@tabs (p. 205)
@footer (p. 190)	@textcolor (p. 205)
@for (p. 194)	@trans (p. 206)
@foreach (p. 194)	@translate (p. 206)
@gamesystem (p. 198)	@var (p. 206)
@gettext (p. 198)	@vline (p. 207)
@header (p. 190)	@while (p. 207)
@hline (p. 198)	@wraptab (p. 208)
@if (p. 199)	@writepicture (p. 208)
@italic (p. 199)	
@keepnext (p. 199)	

Filter @array

```
@array _numbers[4]="One", "Two", "Three", "Four"
@array _skills[0]
```

The @array command defines an array in the context of the character sheet. If the array already exists, it is not redefined. Since these arrays are created in the context of the current character sheet, you must take care to name them with names that will not conflict with existing arrays.

This is intended for use with the addElement, removeElement, etc., predefined functions.

Filter @assign

The @assign command lets to create and modify variables during filter processing. You should be careful not to change variables that are part of the character, because those changes cannot be made permanent.

The basic format of the @assign command is

```
@assign variable=expression
```

where variable is a variable name and expression (p. 169) is a normal **GURPS Character Builder** expression.

The assign can be omitted. The following is the same as the above.

```
@variable=expression
```

You might want to use an expression to compute a value that is used more than once. For example,

```
@assign combatbonus=reflexes+armor
Block: %block%/%block+combatbonus%, \
```

```
Parry: %parry%/%parry+combatbonus%, \
Dodge: %dodge%/%dodge+combatbonus%.
```

Filter @bgcolor

This command sets the background color for subsequent lines in the text output:

```
@bgcolor red green blue
@bgcolor
```

Three numbers follow the command, indicating the color intensities of red, green and blue. 255, 255, 255 is white. 0, 0, 0 is black. Light gray is 224, 224, 244. Other colors can be used as desired, but many printers may not be able to print in color. This color overrides the color that was specified in the text object.

If the arguments are omitted, the background color is restored to the background color of the text object, which is nothing if the object is not filled.

This command is ignored when text is copied to the clipboard.

Example

The following script will emit lines alternately colored white and gray (assuming the enclosing text object's background color is white or is unfilled).

```
%str%      STR
@bgcolor 224 224 224
%dex%      DEX
@bgcolor
%hits%     HITS
@bgcolor 224 224 224
%con%      CON
@bgcolor
%int%      INT
...
```

Filter @bold

The @bold command causes the following text in the filter to be displayed in a **bold** font.

```
@bold
```

All subsequent text in the current text area will be displayed with this format. If you intend to change the formatting of a single line, you need to set it back to the plain font after that line with a @plain command.

See also: @plain (p. 203), @italic (p. 199), @font (p. 192).

Filter @bolditalic

The @bolditalic command causes the following text in the filter to be displayed in a ***bold italic*** font.

```
@bolditalic
```

All subsequent text in the current text area will be displayed with this format. If you intend to change the formatting of a single line, you need to set it back to the plain font after that line with a @plain command.

See also: @plain (p. 203), @italic (p. 199), @font (p. 192).

Filter @clipformat

If you specify the @clipformat command in a filter, when the text is copied to the clipboard that format is specified. If omitted, the output of the filter is written to the clipboard as text. For example, to specify Rich Text Format (RTF), use the following command:

```
@clipformat Rich Text Format
```

This command must appear at the beginning of the file, after the `@gamesystem` command but before any display commands.

Filter @description

`@description` HTML filter for SRPS.

A one-line description of the filter. This is displayed to the user when the filter is highlighted in the list of available filters.

This command must appear immediately after the `@gamesystem` command.

Filter @dest, @footer, @header

These commands work together to reorder filter output.

```
@header
...
@dest body1
...
@header header2
...
@dest body2
...
@footer footer2
...
@footer
...
```

Filter destinations are used to redirect the output of filters to named "buckets." When the filter output is generated, the text following each destination command is output in the order that the destinations were declared. This allows text output at different times in the filter to be placed adjacently.

This is especially useful for creating summaries of multiple character sheets (p. 19) in a single file or for printing.

Each time text the destination is set to a header or a footer, the current text in that header or footer is replaced with the new text. When a normal destination is indicated (`@dest`), text is appended to the end of that destination.

@header

The `@header` command indicates a "header" destination. If no name is specified, this is the header for the entire output file. Named headers appear within the body of the output where they were first encountered. Only the last text destined for a header is retained. For this reason the text in a header should be static.

@footer

The `@footer` command indicates a "footer" destination. If no name is specified, this is the footer for the entire output file. Named footers appear within the body of the output where they were first encountered. Only the last text destined for a footer is retained. For this reason the text in a footer should be static.

@dest

The `@dest` command indicates a normal destination. All text falling within the destination appears together in the final output, in the order that it appeared.

Example

The following filter makes a summary of several character's characteristics and combat information in two separate sections.

```
@header
Characteristics
```

```

Name      STR  DEX  CON  APP  SIZ  EDU  INT  SAN  POW  HITS  Magic
  Idea  Luck  Know
@dest stats
%@Char.name% \
%STR%    %DEX% %CON% %APP% %SIZ% %EDU% %INT% %SAN% %POW% %HITS%
  %Magic%    %Idea%\    %Luck%\    %Know%\
@header combat

Combat
Name      Dmg Bonus and Weapons
@dest combat
%@Char.name% \
%dmgbonus%\
@assign _comma="; "
@foreach(Equipment)
@if strindex(@category, "Weapon")
%_comma%%@name%: %@Skill%, %@Damage%\
@endif
@endfor

```

Sample output when run with the **Utilities | Filter Character Sheets...** command:

Characteristics															
Name	STR	DEX	CON	APP	SIZ	EDU	INT	SAN	POW	HITS	Magic	Idea	Luck	Know	
Anna Vixen	8	14	14	17	11	13	10	75	15	12	15	50%	75%	65%	
Artie Gumshoe	15	14	16	12	13	14	11	60	12	14	12	55%	60%	70%	
Dr. Warren Bedford	10	7	9	9	10	23	17	80	16	10	16	85%	80%	99%	
Harvey Walters	4	12	14	17	16	16	17	34	9	15	9	85%	45%	80%	
Rachel Hemingway	8	12	11	14	9	17	16	65	13	10	13	80%	65%	85%	

Combat	
Name	Dmg Bonus and Weapons
Anna Vixen	none
Artie Gumshoe	+1D4; .45 Automatic: 65%, 1D10+2
Dr. Warren Bedford	none; .30 Carbine: 40%, 2D6
Harvey Walters	none
Rachel Hemingway	none; .38 Revolver: 40%, 1D10

Filter @exit

Exits the filter immediately. You should avoid using this without first warning the user with a @message command, unless of course all output has been correctly emitted.

Filter @extension

If you specify the @extension command in a filter, a separate application is launched to handle the printing and viewing of the file. For example,

```
@extension .rtf
```

would cause the application that deals with RTF files to be executed when you print through the filter. If you have Microsoft Word installed, this would cause Word to print the file in the RTF format generated by the filter.

If no application is configured on your system to handle this file extension, then **GURPS Character Builder** will attempt to print the file (the result will likely be unsatisfactory).

A temporary file is created in the system TEMP directory (Windows\temp by default). This is normally deleted when you exit **GURPS Character Builder**, but if the application is holding on to the temporary

print file, **GURPS Character Builder** will be unable to delete it. In this case, you'll have to delete the file yourself.

This command must appear at the beginning of the file, after the `@gamesystem` command but before any display commands.

Filter **@fail**

```
@fail This message is displayed and filter processing is stopped.
```

This command is used to stop filter processing and indicate an error. You can use, for example, to stop processing if the character sheet is the wrong type:

```
@if @SheetType != "Vehicle"
@fail This copy filter may be used only with Vehicles.
@endif
```

Filter **@firstindent**

This command sets the indentation of the first line of the paragraph:

```
@firstindent .2"
```

A single argument specifying the indentation distance and the unit is required (units are inches [in or "], centimeters [cm] and points [pt]).

The `@leftindent` (p. 200) command specifies the indentation of the following lines of the paragraph. To make a hanging indent, make the first indent value be less than the left indent value.

The `@parinfo` command set the first indent distance and the paragraph justification as well.

The `@firstindent` should appear after the `@leftindent`.

Filter **@font**

This command sets the font for subsequent lines in the text object. It overrides the font specified in the text object containing the text.

```
@font "fontName" fontSize textStyle
```

The *fontName* is the name of the font. It must be enclosed in double quotes.

The *fontSize* is the size of the font. Generally it should be specified in points (for example, 10pt).

The *textStyle* is a concatenation of characters indicating the style:

Style	Description
b	Bold: bold
i	Italic: <i>italic</i>
s	Strikeout: strikeout
u	Underlined: <u>underlined</u>

If the style is omitted, plain is assumed.

Font style changes can take place only across lines. The line height remains constant throughout the text object. If you change font sizes, the text object should be defined with the largest font.

Example

The following shows how to change fonts:

```
@font "Times New Roman" 12pt bi
Header
@font "Times New Roman" 12pt
```


Text
...

This will rendered as:

Header
Text
...

See Also

@bold (p. 189)
@italic (p. 199)

Filter @fontinfo

The @fontinfo commands selects the font information for the font. This is a low-level command that provides direct access to the parameters used to create fonts under Windows.

```
@fontinfo pointSize charStyle pitchAndFamily charset
@fontinfo 12 3 2 2
```

The point size is a decimal number that gives the number of points.

The charStyle is a hexadecimal number that gives the character style, with each of the styles represented by a bit in the value. The following values may be ORed (or added) together to indicate a combined style:

0	Plain
1	Bold
2	Italics
4	Underlined
8	Overstrike

For example, bold italic text is indicated by 3.

The pitchAndFamily parameter indicates the pitch (fixed or variable) and the font family. These are also ored together to form the pitch and family.

The values for pitch:

0	Default pitch
1	Fixed pitch (all characters are the same width).
2	Variable (character widths vary)

The values for family:

0000	Don't care
0100	Roman (serif font)
0200	Swiss (sans-serif font)
0300	Modern (fixed-pitch font)
0400	Script font
0500	Decorative font

For example, the value for Times New Roman is 0102.

The charset value indicates the character set to use:

0	ANSI
---	------

- 1 Default character set
- 2 Symbol character set.

Other character sets indicate foreign language character sets such as Japanese, Hangeul, Chinese, etc. See the Windows documentation for further information.

For example, the following commands can be used to select the Wingdings font. If the `@fontinfo` command isn't specified, Windows won't find the proper font because the character set won't be set to Symbol (2).

```
@fontinfo 16 0 2 2
@font "Wingdings" 16pt
```

Filter **@for**

The basic format for the `@for` command is:

```
@for var = 1; var <= 10; var = var + 1
...
@endfor
```

The `@for` command consists of an assignment statement, followed by a condition, followed by an increment statement, all separated by semicolons. All commands between the `@for` and the `@endfor` are repeated until the condition evaluates to false (zero). The assignment and the increment parts may be omitted:

```
@var i
@for ; i < 10;
i = %i%
    @i=i+1
@endfor
```

If the increment is omitted you must be sure to increment the variable in the condition, or change something else to make sure that the loop terminates. The assignment and increment are normal assignments, requiring a variable, an equals sign and an expression.

If the condition is initially false, no commands within the loop will be executed. For example, nothing will be emitted from the following sequence of commands:

```
@var i
@for i = 1; i < 0; i = i + 1
i = %i%
@endfor
```

The above `@for` command is identical to the following `@while` construct:

```
@var i
@i=1
@while i < 0
i = %i%
@i=i+1
@endwhile
```

You may place parentheses around the `@for`'s arguments.

Filter **@foreach**

The basic format of the `@foreach` command is

```
@foreach (List, delim)
text
...
@endfor
```

The `@foreach` repeats once for each item in the list you name as its first argument. The text you specify for `delim` is placed between each member of the list.

If item selection rules (p. 57) are active, items that do not satisfy the active rules will not be processed.

Within the `@foreach`, there are several special variable names that you can use to reference information about the current item in each iteration through the list:

Name	Purpose
<code>@adj</code>	The item's adjustment list.
<code>@autoadded</code>	True if item was automatically added (only used in <code>@if</code> commands inside <code>@foreach</code>).
<code>@autochargecost</code>	True if automatic items added by this item should have their cost included in the total.
<code>@autoID</code>	The automatic item parent ID. Each time an automatic item is added, it's assigned an <code>autoID</code> (p. 76), which is an integer. The <code>autoParent</code> value for each child is set to this <code>autoID</code> . The <code>autoID</code> is 0 if the item added no automatic items.
<code>@autoParent</code>	The <code>autoID</code> of the item's "parent" (the item that added it). This value is 0 if the item has no parent (was not added automatically).
<code>@basecost</code>	The item's base cost (before any options that may modify the cost are applied).
<code>@basevalue</code>	The base value of the item's level before any bonuses are applied.
<code>@bonus</code>	The value of the item's bonuses.
<code>@category</code>	The item's category.
<code>@checkexp</code>	The item's check expression.
<code>@checkfailed</code>	TRUE if the item's check expression has failed. This can be used to display an indication that an item has an illegal cost.
<code>@childcost</code>	The sum of the costs of a sublist's children.
<code>@class</code>	The items' class.
<code>@cost</code>	The item's cost.
<code>@defaultvalue</code> or <code>@default</code>	The item's default value.
<code>@deleteauto</code>	True if automatic items added by the item should be deleted along with the item.
<code>@dsid</code>	The data sheet ID for this item. If no data sheet ID was specified, it is the empty string ("").
<code>@editdlg</code>	The type of edit dialog used for the item. This is useful for determining whether an item is level-based to decide whether to display the value of an item with <code>@level</code> or <code>@cost</code> or both. The possible values are 0: standard level-based item; 1: name only (no cost displayed), 2: Options only, 3: Level only; 4: Cost only.
<code>@excluded</code>	True if the item's cost is excluded from the list total.
<code>@exp</code>	The item's level expression.
<code>@filename</code>	The name of the character sheet file.
<code>@formula</code>	The item's formula.

@format	The item's format string (p. 102) (used to display items in the list window), with the appropriate values substituted. When referenced in an expression, this contains the raw format.
@generic	1 if the item's "generic" flag is set, and 0 if it is not.
@indent	The "indentation level" of the item. Unindented items, those at the top-most level of the item list, have a value of 0. Indented items -- those that are members of a sublist -- have an @indent value equal to the number of ancestors. A sublist that is a member of a top-most sublist has an @indent of 1, while any items in that sublist have an @indent of 2.
@item	The text of the item as it is displayed in the list window. Tab characters are inserted between the "fields" of the item display. Unlike @format, this will render the text of the item even in expressions.
@level	The item's level (value) -- the skill or power level.
@listindex	<p>The "number" of the item in the list. The first item in a sublist has a listindex of 1. This can be used to number or letter (using the chrval function (p. 171)) items. For example, the following sequence</p> <pre>@foreach(Powers) %dupstr(" ", @indent)%chrval(@listindex+96)%: %@name% @endfor</pre> <p>produces this output:</p> <pre>a: Clairsentience b: Elemental Control a: Ego Attack b: Duplication c: Partially Limited Power a: Energy Blast b: Energy Blast c: Armor</pre>
@lookuparray	The value of the lookup array for the item, if one is present. If not present, the empty string is returned.
@name	The item's name.
@noautodup	If true, any automatic items that already are present in the character sheet are not added again.
@notes	The notes for the item.
@options	The number of options for this item.
@optrawtype	The "raw" type of the option in an @option ... @endoption construct. The possible values are: frac, add, auxcost, exp, mult, percent, text, adj, adjdistcost, adjdistlevel.
@optrawvalue	The "raw" value of the option. That is, the string of the option verbatim, without any processing. For example, with an expression option of "2+2", @optvalue returns "4", while @optrawvalue returns "2+2".
@origList	The original list where the item came from. If the item is in its original list, this will usually be the empty string, though if the item has been renamed or moved, it may have the list name.
@origName	The original name of the item as it appeared in the selection list.

@parent	The name of the parent item.
@prereq	The list of requirements for the item.
@rand	Return the random pips value. Zero should be treated equivalently to 1. The value -1 should be treated as an indicator of exclusion from random selection.
@see	Return the value of the "see" flag. This is set if the item is a reference or "pointer" to another item. This is useful when adding items to lists in conversion scripts. Any item with the see flag set is not a "real" item.
@sformat	The item's selection format string.
@sublist	TRUE (1) if the item is a sublist, else FALSE (0) if not.
@varname	The variable name for the item.
@XXX	The option XXX. For example, if you defined an option (p. 70) named "Armor", you could reference its value with @armor. You can reference options that contain blanks by surrounding the option name with backquotes: @`Option With Blanks`. Partial matches are allowed (that is, @Skill will match "Skill Bonus"). Options are searched from first to last for a partial match. Exact matches have precedence over partial matches: @Skill will match "Skill" even if "Skill Bonus" appears first in the list.

You may abbreviate references to the predefined variables, using "lev" instead of "level". You may simply reference the text of the items, or actually compare them numerically in expressions. Comparisons of the variable names are case-insensitive.

To access the members of all the options for an item, use the @options (p. 201) command.

To iterate through the members of a data sheet list, prepend a "*" to the category list name.

Example

Given this list of skills: Swimming at 12, Skydiving at 14, Nuclear Physics at 16, Golf at 8, the filter

```
Skills: \
@foreach(Skills,, )
%@name%-%@level%\
@endfor
.
```

would output the following:

```
Swimming-12, Skydiving-14, Nuclear Physics-16, Golf-8.
```

The above example uses the "\" character at the end of the line to keep text on the same line. The following example references several options that may be defined for item in the spell list:

```
@foreach(Spell)
%@name% %@level%    ?@time?%@time%?1? ?@Duration?%@Duration%?--?
    ?@energy?%@energy%?1?    ?@page?%@page%?--?    @notes
@endfor
```

If the "time" option is not defined for the item, then a default value of 1 is used. If no duration is specified, then "--" is used. The others work similarly.

The following would print a list of all the skills in the data sheet skills list:

```
@foreach(*Skills)
%@name%
@endfor
```

Filter @gamesystem

The first command in a filter file should be `@gamesystem`. This specifies the game system to be used for the filter. You can only filter character sheets with filters for the same game system.

The `@gamesystem` command should only be used in filter files. It should not be used in a print template text object.

Filter @gettext

This command gets text from the user in a dialog box and stores it in a variable in the character sheet's context. This allows input

```
@gettext _variable, Dialog Box Title
```

This command displays a dialog box which has the title specified after the comma. A text edit field is present that contains the text in the named variable. If the user clicks the **OK** button, the variable is set to the text entered in the edit field. If the **Cancel** button is clicked, the filter will terminate immediately.

The following filter looks for the value of a configuration parameter and proposes that as a value in a `@gettext` dialog for the first file that uses a filter. The value obtained from the user is that output by the filter.

```
@output off

@assign _title = ConfigParam("filterTitle")

@if getGlobal("CurrentFile") = 0
  @if (_title = 0)
    @assign _title="Enter a title here"
  @endif
  @gettext _title, Enter the Title
  @assign _n=SetConfigParam("filterTitle", _title)
@endif

@output on
%_title%
```

Filter @hline

This command emits a horizontal line in the text.

```
@hline type width [ length ]
```

The type of line displayed is one of the following values:

Value	Description
0	Solid line.
1	Tabbed line: wherever left-adjusted tabs appear, a small break will be made in the line.
2	Centered line.

The width of the line is specified in points (pt), inches (in or "), centimeters (cm) or no units, which is assumed to be TWIPS (twentieth of a point).

The `length`, if present, is the length of the line (using the same units as the width. If omitted, the line is drawn all the way across the text object. For a solid line, a negative value indicates the line will be drawn the absolute value of the length from the left border of the text object to the right border.

The `@vline` (p. 207) command has an example of `@hline`.

Filter @if

The basic format of the @if command is

```
@if expression
...
@elseif expression
...
(More @elseif's are allowed)
...
@else
...
@endif
```

This evaluates each of the expressions (p. 169) in turn, and if any are true (non-zero) then the text for that clause is output. If none of the expressions is true, the text after the @else is output.

Example

The following example outputs a list of advantages if there are any items in the Advantages list:

```
@if @advantages
Advantages: \
@foreach(adv,; )
?@level?%@name%-%@lev%?%@name%?\
@endfor
.
@endif
```

In the line that outputs the text, note that if the level is 0, only the name of the item is output, and if it is non-zero, the name and the level are output.

Checking for Dialog Fields

You can check for dialog fields in @if by including the dialog and field names in @`. For example, if you want to print something only if the "Name" field in the "Information" dialog has something in it, you can do the following:

```
@if @`Information.Name`
%@Information.Name%
@endif
```

For text fields, the value of @`. is true if there is any text and false if the field is empty. For fields that have variables associated with them, the value of the field is the value of the variable.

Filter @italic

The @italic command causes the following text in the filter to be displayed in an *italic* font.

```
@italic
```

All subsequent text in the current text area will be displayed with this format. If you intend to change the formatting of a single line, you need to set it back to the plain font after that line with a @plain command.

See also: @plain (p. 203), @bold (p. 189), @font (p. 192).

Filter @keepnext

This command keeps the next printable line with the subsequent printable line. This is useful for ensuring that a header line stays on the same page or in the same column as the text that it heads.

```
@keepnext
```

Formatting commands do not count as lines in this context. This works by skipping to the next page or column if there aren't at least two lines worth of space remaining in the current display area. It's essentially the same as @need 2ln.

Example

The following ensures that a bold header stays with the following text:

```
@keepnext
@bold
Skills
@plain
@foreach(Skills)
%item%
@endforeach
```

Filter @landscape

This command specifies that the filter print in landscape mode. The text will be printed on the page with the wider dimension of the paper in the horizontal direction.

```
@landscape 1
```

To select portrait mode, omit the @landscape command.

Filter @leftindent

This command specifies the left indentation of subsequent paragraphs.

```
@leftindent indent
```

The *indent* parameter is the distance from the left border that the text will be printed. The first line indentation is specified separately, with the @firstindent (p. 192) and @parinfo (p. 202) commands. The @leftindent command should always appear before the @firstindent or @parinfo (p. 202).

Filter @message

Display a message box to the user. The argument is an expression that should evaluate to a string. For example,

```
@message format("An error was found processing %s", @filename);
```

Filter @need

This command informs the text formatter that a certain amount of space is needed to display the subsequent text. This is used to keep several printable lines of text together.

```
@need space
```

Space is the amount of space needed. *Space* must be a number followed immediately by a unit. Available units are inches [in, "], centimeters [cm], points [pt], lines [ln] and none, which are assumed to be TWIPS (twentieth of a point).

If space specified is not available, the formatter moves to the next column or page and displays the text. Formatting commands are not included in the distance when the ln measurement is specified.

The @keepnext (p. 199) command is a special case of @need, equivalent to @need 2ln.

Example

The following example keeps the the four lines describing the character point costs together on the page or column.

```
@need 4ln
@tabs 0 .5" 3 1.5" 0 2" 3 3"
Costs: Char.: %t_char% Disad.: %disadvantages%
Powers: + %t_powers% Base: + %basePoints%
@if experience Exp.: + %experience%
@endif
```


Total: = %total% Total: = %ptsavail%

Filter @options

The @options command goes through each option for an item and allows access to the name and value for each option in turn. The @options command can be used only inside a @foreach command. The basic format for the @options command is

```
@foreach (xxx)
...
@options(; )
text
...
@endoptions
...
@endfor
```

The parentheses are optional. If present, the string within them is used to delimit the options found.

If options have the Display Option in Output (p. 97) attribute unchecked, they will not be displayed by the @options command.

The following special variables are available within the @options command:

Name	Purpose
@option	The name and value of the option. This is normally formatted with the name of the option, followed by a colon and the value of the option.
@optexp	The expression for the option. For adjustments this is expanded with the values for the item.
@optname	The name of the option.
@optnumbervalue	For options that have a separate cost and textual description, the numeric value of the option.
@optqualifier	The qualifier on the option.
@opttextvalue	For options that have a separate cost and textual description, the text value of the option.
@opttype	The type of the option. The possible types are: "Fraction", "Addition", "AuxCost", "Expression", "Multiplier", "Percentage", "Text", and "Adjustment".
@optvalue	The value of the option.

Example

The following is an example of how the @options command can be used. The name of each item in the Equipment list is output on its own line, and if there are any options for the item, a parenthesized, semicolon-delimited list of the options is output after it. If the name of an option is "Note" the name is not output.

```
Equipment
@foreach (Equipment)
\
%@name%\
@if @options > 0
(\
@options(; )
@if @optname != "Note"
%@optname%: \
@endif
```

```
%@optvalue%\
@endoptions
)
@else

@endif
@endfor
```

The following is less complex, but more typical. It prints the list of skills, using the format from the list. If there are any options, they are printed on the next line, indented.

```
Powers
@foreach(Powers)
  %@format%
  @if @options > 0
    \
    @options(; )
    %@option%\
  @endoptions

@endif
@endfor
```

Filter @output

The @output command suppresses output from the filter, or outputs its argument.

```
@output off
@output on
@output The value of x = %x%.
```

The @output command useful if you have a long sequence of computational commands (@if, @assign, etc.) that you would like to intersperse with blank lines to make the code more readable, but you don't want those blank lines to appear in the output.

If the argument for the @output command is off, output from the filter is suppressed. If the argument is on, output is turned on again. If the argument is anything else, that text is output, even if output has been turned off.

You should not leave output turned off at the end of the filter. That is, you should always balance an @output off with an @output on.

If output is turned off at the end of a filter it is considered an error. The first time **GURPS Character Builder** encounters such errors it will inform the user. Subsequent failures during the same run of the application will not be reported. This prevents the error from appearing every time an erroneous filter is used, which would inundate a naïve user who cannot fix the problem.

When using @output to output text, all white space between the command and the text is discarded. To output leading white space, place a backslash before the white space:

```
@output \           Two tabs precede this.
```

Filter @parinfo

The @parinfo command sets the indentation of the first line of the paragraph and the type of paragraph.

```
@parinfo firstIndent justType
```

The firstIndent parameter is a measurement (2", .5in, 2cm, 20pt, or TWIPS [1/20th of a point] if no unit is specified). It is the indentation from the left border of the first line. In combination with the @leftindent command it can be used to create "hanging indents."

The @parinfo should appear after the @leftindent.

The `justType` parameter is the type of justification used for the paragraph. It is a number with the following values:

Value	Description
0	Left-aligned
1	Right-aligned
2	Centered
3	Justified

This command is ignored when text is copied to the clipboard.

Examples

The following will cause a hanging indent of a quarter inch:

```
@leftindent .25"
@parinfo 0" 0
Some example text. Some example text. Some example text. Some example
text. Some example text. Some example text.
```

This will yield:

```
Some example text. Some example text. Some example text. Some example
text. Some example text. Some example text.
```

The following will indent following paragraphs a quarter of an inch.

```
@leftindent 0"
@parinfo .25" 0
```

This will yield:

```
Some example text. Some example text. Some example text. Some example
text. Some example text. Some example text.
```

Filter @plain

The `@plain` command causes the following text in the filter to be displayed in a plain font. It removes the effects of previous `@bold` and `@italic` commands.

```
@plain
```

See also: `@bold` (p. 189), `@italic` (p. 199), `@font` (p. 192).

Filter @samepage

This command indicates that the filter should print multiple files on the same page. It should be used for combat summary filters where several characters are printed on the same page.

```
@samepage 1
```

This command frees the user from having to remember to click the **Print on Same Page** button in the multiple file filter dialog.

Filter @sheettype

```
@sheettype Character
```

Specifies the type of character sheets that may be used with this filter. If the filter sheet type doesn't match the character sheet type, the filter may not be used with the character sheet.

To allow a filter to be used with any character sheet type for a particular game system, omit the `@sheettype` command.

Filter @sortexp

If you specify the `@sortexp` command in a filter that is used with the multiple character sheet filter (p. 19) the files will be output in the order specified by the sort expression and sort order (`@sortorder` (p. 204)). This is useful when you want to display a combat list in DEX order, for example. The `@sortorder` command controls the order – ascending or descending.

The value of the sort expression is evaluated in the context of each character sheet processed. The result is used to order the files when they are output. The following would output the results in Descending order of `total_dex`:

```
@sortexp total_dex
@sortorder 1
```

This command must appear after the `@gamesystem` command and before the first displayable command.

Filter @sortorder

The `@sortorder` command is specified in a filter used with the multiple character sheet filter (p. 19) to order the output in the order specified by the sort expression.

This command takes an argument to indicate the sort order:

```
0 Ascending
1 Descending
```

This command must appear after the `@gamesystem` command and before the first displayable command.

Filter @sub

The `@sub` command defines a filter subroutine:

```
@sub subName(arg1, arg2)
# Commands.
@endsub
```

A subroutine can be invoked with a `@call` command:

```
@call subName("param1", "param2")
```

or the name of the subroutine can be placed immediately after the `@`.

```
@subName("param1", "param2")
```

When a subroutine is called, the arguments are evaluated and the local variables with the parameter names are defined in the context of the subroutine. If an array is passed to a subroutine, any references to that variable in the subroutine must be an array reference.

A subroutine can be defined with no parameters:

```
@sub paramLess
...
@endsub
@call paramLess
@paramLess()
```

If the `call` is omitted the parentheses must be present for the call to be recognized as a call.

A subroutine name may not be one of the predefined commands (for example, `for`, `foreach`, `sub`, etc.).

When a subroutine is invoked, all text within between the `@sub` and `@endsub` is emitted, or lines containing commands are executed. When the parameter variable names are referenced within the subroutine the values passed into the subroutine are returned.

When the `@var` (p. 206) command appears within a subroutine, the variable is defined only within the context of that subroutine call. When the subroutine exits (reaches the `@endsub`) all local variables (including parameters) are destroyed.

Subroutines may not be called recursively, either directly or indirectly.

Filter **@tabs**

The `@tabs` command specifies the tabs in the current text object. This changes the tabs from the initial value set in the text object. If the text is continued to another object that object will inherit these tabs.

```
@tabs type position [ type position ] ...
```

The `type` may have the following values:

Value	Description
0	Left-aligned tab.
1	Right-aligned tab.
2	Centered tab.
3	Decimal tab. The end of the number, or the decimal point, is placed at the tab if the text is a number. Otherwise it treats the text as a right-aligned tab.

The `position` is specified in inches (in or "), centimeters (cm), points (pt) or TWIPS (twentieth of a point) if no unit is specified. The tabs must be specified in order, from left to right.

If the position is a negative value it is taken to be relative to the right border of the enclosing text object. This is especially important if you're using a right-justified tab near the right border and your text flows from one text object to another text object with a different width.

If the tab exceeds the right bound of the enclosing object it is moved into the object.

Example

The following can be used to specify a hanging indent with a right-justified number leading the paragraph.

```
@tabs 1 .4" 0 .5"
@leftindent .5"
@parinfo 0" 0
3      The number to the left should be right justified and the rest of
this text should be displayed with a hanging indent.
```

This will be rendered as:

```
3 The number to the left should be right justified and the rest of
  this text should be displayed with a hanging indent.
```

The following will display the contents of the Skills list with the cost of the skill centered at half-an-inch from the right border, and the level of the skill centered at a quarter-of-an-inch from the right.

```
@tabs 2 -.5" 2 -.25"
@foreach(Skills)
  %@name%  %@cost%      %@level%
@endforeach
```

Filter **@textcolor**

This command sets the color of the subsequent text to the specified value:

```
@textcolor red green blue
```

The values are the color intensities, ranging from 0 to 255. For example, 255 255 255 is white, 0 0 0 is black and 0 0 255 is blue.

This color overrides the text color specified in the parent text object. This color is retained until another text color command is found, even if the text continues to another text object.

Filter @trans

The `@trans` command specifies character translations to take place on the characters that appear in the text of the character sheet.

Following the `@trans` command must be a single space, followed by a single character (which may be a special escaped character -- see below), followed by a single space, followed by a string.

The escaped characters are:

<code>\n</code>	New line
<code>\r</code>	Carriage return
<code>\t</code>	Tab
<code>\\</code>	Backslash
<code>\b</code>	Backspace

For example, if you wish to write a filter that outputs RTF (Rich Text Format), you can specify the following list of translations to prevent any text in the character sheet from being misinterpreted:

```
@trans \t \\tab
@trans \r \r\n
@trans \n \\par
@trans { \\{
@trans } \\}
@trans \\ \\\\
@trans - \\_
```

This command must appear after the `@gamesystem` command but before any display commands.

Filter @translate

Controls whether the text obtained from character sheet objects is processed by the `@trans` directives (p. 206). If the argument is "true", the translations are performed. If it is "false" the translations are suspended until a `@translate true` is encountered.

For example, the following except from a script emits the RTF version of the details into the filter output without any processing. This allows the formatting specified in the details to come through, rather than be escaped by the `@trans` directives.

```
@if @details
@translate false
%@detailsrtf%
@translate true
@endif
```

Filter @var

The `@var` command defines variables or arrays that are local to the filter or the subroutine (p. 204) in which the command appears:

```
@var i
@var a, b, c
@var true=1
@var false=0
@var costArray[6]=.5,1,2,4,8,16
```

When `@var` appears in a filter outside a subroutine, the variable is defined within the context of the filter and not within the character sheet: the character sheet will be unchanged. If a `@var` has the same name as a character sheet variable, the `@var` will have precedence.

It is recommended that all "working" variables used in a filter be declared to avoid accidentally changing the character sheet's values.

When a variable is declared inside a subroutine it is destroyed when the subroutine exits.

More than one variable can be declared in a `@var` command. All variables not explicitly initialized are set to zero.

Variables can be initialized to a value by indicating the value after an equals sign. Only one initialization may be performed per `@var`.

Arrays defined with `@var` are defined within the context of the filter (or subroutine).

References to a variable before it is defined in a `@var` will be to the character sheet variable by that name, not the the filter variable. It is good practice to declare all filter variables at the beginning of the filter.

Filter @vline

The `@vline` command plots vertical lines in the current text object.

```
@vline width position [ width position ] ...
```

As many pairs of width and position may be specified as desired.

The `width` argument indicates the width of the line (.1", .1in, .254cm, 7pt, 144, where no unit indicates TWIPS [1/20th of a point]).

The `position` argument indicates the position of the line from the left border. If the value is negative, the position relative to the right border. That is, `-.5"` indicates a line drawn half an inch from the right border of the current text object. You should specify right-relative lines if your text object continues to a text object with a different width.

If another `@vline` command occurs, the current line drawing is stopped and new lines are defined. A `@vline` command with no parameters stops vertical line drawing. Lines are drawn to the end of the current text object if the text ends.

When text is copied to the clipboard, nothing is emitted for this command.

Example

The following command draws two boxes around the stats displayed, with the header line in bold.

```
@font "Times New Roman" 10pt b
Primary Characteristics
@font "Times New Roman" 10pt
@hline 0 1pt 1.5"
@vline 1pt 0" 1pt .45" 1pt 1.5"
@tabs 1 .4" 0 .5"
    14    STR
    12    DEX
    11    HITS
    13    CON
    15    INT
@vline
@hline 0 1pt 1.5"
```

Filter @while

The basic format of the `@while` command is

```
@while expression
...
@endwhile
```

As long as the expressions (p. 169) is true (non-zero) then the text between the @while and the @endwhile is output. If expression is initially false, nothing will be output.

Note: you use @assign commands to set the value of a control variable before and within the @while loop, otherwise it will loop infinitely.

Example

The following filter commands output a table of numbers, 10 per line, enclosing the number that is equal to the value of the variable `hits` in square brackets.

```
@assign i = 1
@while i <= 30
    ?i=hits?[%i%]?%i%?\
@if i mod 10 = 0

@endif
@assign i = i + 1
@endwhile
```

The following is the output, if the value of `hits` is 12.

1	2	3	4	5	6	7	8	9	10
11	[12]	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30

Filter @wraptab

This command controls whether text will be wrapped to the line when a tab is encountered, or if it is ellided.

```
@wraptab setting
```

If *setting* is 1, text is wrapped at tabs. If it is 0, text is ellided at tabs.

Example

If you have a right-aligned tab on the right side of a text object (to display a skill value, for example) and you want text to display completely rather than be ellided, specify @wraptab 1.

Wrap tab on:

```
Theology (Specialty: Judeo-Christian demon
names)                                     12
Broadsword (Specialty: Parry)              14
```

Wrap tab off:

```
Theology (Specialty: Judeo-Christian d... 12
Broadsword (Specialty: Parry)              14
```

@writepicture

This command writes out the specified picture to a file, or to the output.

The basic formats for the @writepicture command are:

```
@writepicture Picture.Picture, jpeg, binary, "fileName.jpg"
@writepicture Picture.Picture, png, hex
```


The syntax is:

```
@writepicture pictureRef, format, type [, fileName]
```

The final argument is optional.

Argument	Description
<i>pictureRef</i>	A picture reference, in the form "Dialog.Field". Usually this is <code>Picture.Picture</code> , but it may differ in the particular character sheet you're using.
<i>format</i>	The image format. Allowable values are: <ul style="list-style-type: none"> <code>bmp</code> Standard Windows bitmap. The "bulkiest" format. <code>jpeg</code> JPEG (Joint Photographic Expert Group) format. The most compact format, though it is lossless. <code>png</code> PNG (Portable Network Graphics) format. A compact, lossless format.
<i>type</i>	The type of output: either <code>binary</code> or <code>hex</code> . If you're writing to a file, you'll almost certainly want to specify <code>binary</code> . If you're placing the picture in the output, you may need to use <code>hex</code> . For example, you can use <code>hex</code> when writing out JPEG and PNG format images to an RTF file (see below).
<i>fileName</i>	An optional argument containing the name of the file to write to. This is an expression. Check out the example below for suggestions on forming the name to avoid overwriting the original character sheet.

Examples

The following is an example of how `@writepicture` can be used to write an image out to an RTF file. If a picture is present, the dimensions are obtained. If the image is wider than three inches (assuming 96 pixels per inch under Windows), it is scaled to make sure it is no wider than three inches. Finally the picture is written to the output in hex.

```
@if @`Picture.Picture`
@assign _n = split(dialogFieldValue("Picture", "picture"), "/[x ]+/",
    "_width", "_height")
\\pard\\plain\\s8\\keepn\\fi0\\li0\\f2\\fs32\\sb120
{\\*\\shppict
{\\pict\\jpegblip
@assign _dwidth=3*96
@if _width > _dwidth
@assign _scale=integer(100*_dwidth/_width)
\\picscalex%_scale%\\picscaley%_scale%
@endif
@writepicture Picture.Picture, jpeg, hex
}}
\\par
@endif
```

In the next example, the image is written out as a JPEG file. This is intended to be used with the **File | Save As...** command through a text filter. The name of the file is formed by taking the name of the character sheet and substituting `.jpg` for the `.chr` extension. A check is made to ensure that the result is different from the original, to avoid overwriting the original character sheet. Finally the picture is written out as a binary file in JPEG format, in the current directory. When **File | Save As...** through a text filter is used, this is the same directory as the saved file.

```
@if @`Picture.Picture`
@assign _fname=replaceString(@filename, '/\\.chr$/', '.jpg')
# Ensure output file name is different from the source file
```

```
# name to avoid overwriting the character sheet.
@if (_fname = @filename)
    @assign _fname=concat(@fileName, ".jpg")
@endif
@writepicture Picture.Picture, jpg, binary, _fname
@endif
```

Data Sheet Text Format

If you need to do a lot of data entry for a data sheet, it may be more convenient to use a text-based format. Be warned that this takes some skills akin to programming!

You can save a data sheet in text format by holding down the Shift and Control keys when you select the Save or **Save As...** commands when editing a data sheet.

Important Note: if you create a data sheet with a text editor, you should *not* edit it with **GURPS Character Builder**. When **GURPS Character Builder** saves files, it does not preserve macro definitions and invocations.

Topics

- Structure and Header Information (p. 211)
- Data Sheet Item Format (p. 217)
- Data Sheet Macros (p. 226)
- Include Directive (p. 233)
- Text Data Sheet Example (p. 215)
- Character Sheet Format (p. 234)

Structure and Header Information

Generally, the structure of the text data sheet is a sequence of keyword/value pairs. The values are usually text strings. No quotes are required around the string value if it contains no blanks, tabs, new lines or '#', and doesn't start with '\$'. You can continue a string on the next line by ending the line with a \ (backslash).

Comments are introduced by the '#' character. All other text after the '#' to the end of the line is ignored.

The first line of the text data sheet must be:

```
"Data Sheet file v. 1" gamesystem "Example" version 240
```

where "Example" is the name of the game system. Following the header line is more header information. These fields are all optional and may be omitted:

Keyword	Value
title	A string containing the title of the data sheet, displayed in a window when the data sheet is opened.
copyright	A string containing the copyright message displayed in the opening window.
level	The "release level" of the data sheet. If this mismatches the level of the currently loaded data sheets of the same game system, GURPS Character Builder will notify you. This helps ensure that the character sheet and the data sheet agree on the functions that are to be used.
logo	The name of BMP format file that is displayed in the opening window.
datafile	A list of data sheet names, separated by white space and terminated by <code>end</code> . The data sheets listed will be loaded (if necessary) whenever this data sheet is loaded. This allows requirements and automatic item references to items in other data sheets.
define	The define section. Variables, arrays and functions may be defined here (in this format (p. 213)). They are accessible when items in the data sheet are displayed. These definitions are also added to all character sheets that use the same game system, to allow additional functionality to be added to the character sheet without requiring explicit changes to the character sheet file.

Following the header information are the categories. Any number of categories may be included. Each category has the following basic format:

```
cat "Category Title"
  sublist "Sublist 1" memlist
    item "Item 1" cost 1
    item "Item 2" cost 2
    # ...
  end # Sublist 1
  sublist "Sublist 2" memlist
    item "Item 3" cost 3
    item "Item 4" cost 4
    # ...
  end # Sublist 2
end # Category
```

Each category can contain items or sublists. Sublists may also contain sublists. Items (p. 217) are defined in terms of keyword/value pairs.

Following the categories are the options. They have the following general format:

```
options
  list List
    text Text Value
    # ...
  end
end
end
```

Any number of options and lists may be contained in the Options list. Lists may also include lists beneath them. Options (p. 222) generally consist of a keyword and two values.

Finally, the data sheet must end with an end keyword.

Define Section of Data Sheets

The `define` section of the data sheet defines variables, functions and arrays that can be used in computations in the character sheet and in the display formats of items displayed in the data sheet window and the available items dialog. These defines are also automatically added to all character sheets using the same game system.

This allows you to define new functions in a data sheet that is used by items defined in that data sheet, obviating the need to modify the core character sheet template every time you need functions to support new functionality required for items in a new data sheet.

For example:

```
define
var varname 0

func "function(arg1, arg2)" "{\r\n\
    return arg1 * arg2;\r\n\
}"

array arr[10] "'a','\b','c','d','e','f','g','h','i','j'"

end
```

Define Processing

When a data sheet is loaded, the definitions in the defines section are created in a context for the game system. Variable, function and array names defined in data sheets should be unique across all data sheets for the same game system. If a name is defined in two different data sheets for the same game system a duplicate definition error will occur.

When a data sheet is loaded its defines will be added to any open character sheets for the same game system that don't already have a definition by that name.

When a character sheet is loaded, all definitions for open data sheets for the same game system will be added to the character sheet, if a definition by that name is not already present in that character sheet.

When a character sheet is saved, the defines added to the character sheet by the data sheets will be saved along with the character sheets original defines. Those defines will have the values specified by the data sheet at that time. If the defines change in the data sheet, they are **not** updated automatically in the character sheet -- the character sheet must be manually updated (by **File | Convert...** or **Utilities | Update Character Sheets...**) to get the latest defines.

However, when character sheet *templates* are saved, the definitions from data sheets are **not** saved. They are temporarily defined in the context of the template so that items added to the template using those definitions will compute properly, but they are not stored in the template. Each time the template is used the defines are reread from the game system defines created by loading the data sheets.

General Notes

The names and values in the define section are strings. If they contain spaces or double quotes, they must be surrounded by double quotes. Within a quoted string a double quote must be preceded by a backslash ("`\`"). All backslashes must similarly be preceded by a backslash ("`\\`").

To indicate a new line within the string, end it with the sequence "`\r\n`", which is a carriage return line, line feed and string continuation character.

Variables

A variable is defined with the `var` keyword, followed by the name of the variable (a string as specified above), followed by the value (another string). For example:

```
var int int+a_int
var acost 10
var jump round((str+a_str)/5)
```

Arrays

An array is defined by the keyword `array`, followed by the name and size of the array, followed by the list of elements in the array separated by commas. The size follows the array name in brackets. For example, `"charnames[28]"`. The value is a single string, which must be quoted as specified above if there are embedded blanks or quotes.

For example:

```
array charnames[28] "\"STR\",1,\"DEX\",2,\"CON\",3,\r\n\
\"BODY\",4,\"INT\",5,\"EGO\",6,\"PRE\",7,\"COM\",8,\"PD\",9,\r\n\
\"ED\",10,\"SPD\",11,\"REC\",12,'END',13,'STUN',14"
```

Functions

A function is defined by the `func` keyword, following by the name and argument list for the function, followed by the definition of the function. If the name or definition include blanks or quotes, they should be quoted as specified above.

For example:

```
func rnd2(x) round(x/2)
func cmToEnglish(cm) "{\r\n\
    local inches;\r\n\
\r\n\
    if (cm < 10)\r\n\
        inches = cm / 2.54;\r\n\
    else\r\n\
        inches = round(cm / 2.54);\r\n\
    endif\r\n\
    if (inches < 12)\r\n\
        if (inches = integer(inches))\r\n\
            return format('%d"', inches);\r\n\
        else\r\n\
            return format('%2n"', inches);\r\n\
        endif\r\n\
    endif\r\n\
    return format(\"%d'%d%s\", \r\n\
        integer(inches/12), inches mod 12, '\"');\r\n\
}\r\n\
"
```

Text Data Sheet Example

You can copy this example and paste it into a file to use as a template for your own text-based data sheets.

```
"Data Sheet file v. 1" gamesystem "Example" version 240
title "Text Data Sheet Example"
copyright "Portions Copyright © 1995 by Bruce Kvam. All Rights Reserved."
logo charcon.lgo

# Macro definitions.

# Dex-based skills. If the requirement is omitted, none is output.

$$macro skdex(name, varname, cat, prereq)
{{
    item "$(name)" cat "$(cat)" class Dex
    formula skillcost(~x,dexterity)
    default dexterity-2
    format %0ln%-5rc%-10rv%-16ll sformat %0ln%-5ll
    varname $(varname) level dexterity
    $$ifdef(prereq) prereq $(prereq) $$endif
}}

# Intelligence-based skills.

$$macro skint(name, varname, cat, prereq)
{{
    item "$(name)" cat "$(cat)" class Int
    formula skillcost(~x,intelligence)
    default intelligence-2
    format %0ln%-5rc%-10rv%-16ll sformat %0ln%-5ll
    varname $(varname) level intelligence
    $$ifdef(prereq) prereq $(prereq) $$endif
}}

# Constant-cost benefit.

$$macro cben(name, cost, adj)
{{
    item "$(name)" cost $(cost)
    $$ifdef(adj) adj $(adj) $$endif
}}

# Level-based benefit.

$$macro lben(name, costlev, adj)
{{
    item $(name) formula x*$(costlev) checkexp x>0
    sformat "%0ln%-10r!$(costlev) per +1" level 1
    $$ifdef(adj) adj $(adj) $$endif
}}

# Lists of items.

cat "Skills"
```

```
sublist "Animal" memlist
  $$skdex(Riding, riding, Animal)
  $$skint(Animal Handling, animalhandling, Animal)
  $$skint(Teamster, teamster, Animal, "Skills:Animal Handling")
end # Animal

sublist "Combat" memlist
  $$skdex(Sword, sword, Combat)
  $$skdex(Brawling, brawling, Combat)
end # combat

sublist "Psionic" memlist
  $$skint(Telepathy, telepathy, Psionic, "Benefits:Telepath")
  $$skint(Mind Search, mindsearch, Psionic,
"Benefits:Telepath;Skills:Telepath")
end # Psionic

sublist "Social" memlist
  $$skint(Leadership, leadership, Social)
end # social

end # Skills

cat "Benefits"
  $$cben(Combat Master, 15, "Skills:Combat+1;armor+1")
  $$lben(Charisma, 4, "leadership+x")
end # Benefits

end # of Datasheet
```


Data Sheet Item Format

Item list entries have the following general format:

```
item Intimidation class MA formula MA(x,iq-b_so)*IQm
default max(st-5,acting-3)
format %0ln%-5rc%-10rv varname intimidation level 17
cost 1
```

They consist of the keyword `item` followed by the name of the item. The rest of the attributes of the item may appear after that, in no particular order. Each of these fields corresponds to the fields in the Edit Properties (p. 75) dialog.

<code>addaslist</code>	Adds an item as a list. This is useful for items with lists of automatic items. There are no arguments.
<code>addwholelist</code>	Allows a sublist item to be selected from the Available Items list. The sublist and any items in it will be added to the list. This is useful for adding "package deals" to data sheets.
<code>adj</code>	The adjustments (p. 87) are a semicolon-delimited list of adjustments.
<code>alias</code>	The alias (p. 75) for the item.
<code>altformat</code>	The arguments are three strings: the name (the empty string, "", defines the default format), the header (should be the empty string for items: ""), and the format string for the item display (in the same format as <code>format</code>). Multiple alternate formats may appear, each with different names corresponding to the alternate formats defined for the list window. The default format specified here is inserted into the list format string wherever the \$! sequence occurs.
<code>autochargecost</code>	Normal costs are charged for the automatic items added for this item.
<code>autoid</code>	The auto ID (p. 76) of this item.
<code>autoitems</code>	A list of semicolon-delimited automatic item (p. 79) names follows. If you wish to include delimiter characters in an argument of a automatic item directive, escape the character with a backslash (\). Since backslash is also used as an escape for data sheet files, you will need to use a double backslash in data sheets.
<code>autoparent</code>	The auto ID of the parent of this item.
<code>cat</code>	The category list (p. 78) is semicolon-delimited list of category names.
<code>childcostsseparate</code>	If present on a sublist, the cost of a sublist is not determined by the children: it is independent and the children's costs are added separately to the sublist's parent's cost. Used only in conjunction with <code>setlistcost</code> .
<code>class</code>	Class (p. 75).
<code>cost</code>	Cost (p. 75).
<code>checkexp</code>	Check Expression (p. 106).
<code>checkoninsert</code>	If present, the requirements are checked only when the item is inserted.
<code>datafile</code>	Specify data sheets to load. An <code>end</code> keyword must appear after the last data sheet file name specified. File names are separated by spaces.

<code>default</code>	Default Value (p. 106).
<code>defformula</code>	Default Formula (p. 106).
<code>deleteauto</code>	Sets the Delete Auto-added Items (p. 77) flag. If indicated, all child items of this item will be deleted along with it.
<code>dontmarkauto</code>	The automatic items added by this item are not marked as automatic items.
<code>dsid</code>	The argument is a string which is the data sheet ID for the item, a string between one and four characters in length.
<code>dupexpected</code>	The item is expected to be duplicated. Don't display a warning if an item by this name already exists.
<code>editdlg</code>	<p>Edit Dialog: allows you to set the type of edit dialog (p. 104). The following values are available (the default is 0 if not specified).</p> <ul style="list-style-type: none"> 0 Standard edit dialog (allows setting of level and cost, options, etc.). 1 Name-only dialog -- no options, no cost, no level. 2 Name and options only -- no cost or level edit fields, though the total cost is displayed. 3 Name and level -- no cost fields, though options are available. 4 Name and cost -- no level edit field, though options are available.
<code>format</code>	Format (p. 102).
<code>formula</code>	Cost expression (p. 105). If present in a sublist, this has precedence over the default action, which is to sum the totals of all child items.
<code>inheritopts</code>	If specified for a sublist, the children of the sublist will inherit the sublist's options.
<code>intotal</code>	Indicates whether the item should be counted in the total for the list. A value of 1 (the default) indicates that the cost should be included. A value of 0 indicates the cost should be excluded.
<code>keepcost</code>	Copies the cost of the original item to the new item when an item is converted. This is useful for items that have no fixed cost that you wish to retain the cost the user assigned.
<code>level</code>	The level or actual value (p. 75).
<code>lookuparray</code>	Lookup array (p. 104).
<code>noadj</code>	No adjustments (p. 77) are made for this item and its immediate children.
<code>noautodup</code>	Don't add automatic items if they're already present in the character sheet.
<code>noprereq</code>	Do not check requirements (p. 78).
<code>notes</code>	A string consisting of the notes for the item.
<code>open</code>	If present in a sublist, the sublist is open.
<code>opt</code>	The option list for the item. The <code>opt</code> keyword is followed by the options (p. 222). The list is terminated by an <code>end</code> .
<code>original</code>	Original name of item as it was selected from the list.
<code>noautodup</code>	If present, sets the Don't Duplicate Automatic Items (p. 77) flag.
<code>nocheckoninsert</code>	If present, indicates that no requirements checking should be done when items are inserted, but afterwards they are checked normally.

<code>noviolation</code>	If present, indicates that the check expression can never be violated, regardless of the setting of the Disallow if Not Satisfied setting in the preferences.
<code>prereq</code>	A semicolon-delimited list of prerequisites (p. 220).
<code>priority</code>	Specify the priority (p. 76) of the item.
<code>rand</code>	The number of "points" or "pips" associated with this item, for purposes of random item selection. For example, if you have 7 items in a sublist, and you want two of them to be selected with a 1 of 10 chance, and the other five to be selected with a 2 of 10 chance, specify <code>rand 2</code> for the five items. If <code>rand</code> is not specified, the value 1 is assumed.
<code>script</code>	Specifies a command script (p. 260) to be executed when the item is first added by a user. This command is executed only in that instance: it is not executed if the item is pasted. Variables, functions and subroutines in the load script (p. 47) may be referenced. This script is similar to the script that is executed when a control changes.
<code>see</code>	<p>If this flag is present, the item is a <i>reference</i> to another item. If selected in the available items list, the item is not added. Instead, the item named in the <code>prereq</code> is added. If the <code>prereq</code> is not set, the sublist named in the <code>cat</code> is opened. These references must be to another item in the same list. Items with the <code>see</code> bit set are not displayed when all items are displayed in the Available items list.</p> <p>The intent of the <code>see</code> bit is to allow "duplicate" items to occur in different sublists, while maintaining a single actual item that all the duplicates reference. These items should have unique names, different from what they reference. To show a name that's identical to the target item, give the item a that name plus a unique suffix, for example, "!!", and then define the selection format for the item to contain the target name instead of a reference to the item's name. That is, use <code>sformat "%01!Target item"</code> instead of <code>"%01n"</code>.</p>
<code>setlistcost</code>	If present for a sublist, allows you to set the cost of the sublist directly, overriding the cost of the sublist's children. If used in conjunction with <code>childcostsseparate</code> it allows you to have a sublist that has its own cost that is independent of the children's costs.
<code>sformat</code>	Selection format (p. 102).
<code>sort</code>	<p>Sorting order for sublists and categories. This controls the order in which items are listed in data sheets and the item selection dialog. The following values are available (0 is the default):</p> <ul style="list-style-type: none"> 0 Sort by name. 1 Sort by cost. 2 Sort by class. 3 Sort by category. 4 Sort constant cost items first by cost, then non-constant items after alphabetically.
<code>textline</code>	This keyword is inserted by the macro data sheet editor. Everything after this keyword until the end of the line is taken to be a text line for the macro data sheet. When the data sheet is loaded, the <code>textline</code> keyword is simply ignored. It should not be inserted between keywords and their arguments.
<code>totalcostformula</code>	Total cost formula (p. 107).
<code>uniquevar</code>	If the variable specified by <code>varname</code> already exists, a new variable name is generated and saved permanently. This is useful for declaring items that other items reference with item reference options.

usedefvarname

If this is set for an item, the prefix "_default_" is added to the variable name when the item is set to the default value.

varname Variable name (p. 75).

Text Data Sheet Requirements

The requirement string lists all the requirements (p. 89) for the item. All items separated by semicolons are required. A ' | ' indicates an "or". If the first character is ' ! ' the requirement is reversed -- for example, "!Disadvantages:Deafness" means the deafness is disallowed as a disadvantage.

The "~" operator indicates "then." For example, if an item required Advantages of Magic Use-3 and Clerical Powers, and then either the Fire Spell or the Brimstone Spell, it would be written:

```
prereq "Advantages:Magic Use>=3;Advantages:Clerical
Powers~Spells:Fire|Spells:Brimstone"
```

Expression

A dollar sign followed by a standard expression (p. 169):

```
$Int>=10
```

Item in a List

A list name, followed by a colon, followed by the name of the item. For example, the first line below indicates that either the Mind Reading spell or the Telepath benefit is required. The second line indicates that the Detect Magic spell is required and the Mage level must be 2 or greater.

```
Spells:Mind Reading|Benefits:Telepath
Spells:Detect Magic;$Mage>=2
```

If the list name is omitted, it is the same as the item in which the item whose requirements are being tested occurs.

Option Set on an Item

An optional list name and colon, followed by the name of the item, followed by "@@", followed by the option name. If a comparison is required, the comparison operator follows, then the expression. Since an expression is required, any text strings must be surrounded by quotes (either type). If only one item should exist with the specified name, a "*" should follow the "@@".

For example:

```
Feats:Weapon Focus@@Weapon='Longsword'
Feats:Weapon Specialization@@Weapon=optValue("Weapon")
Spells:Height Spell@@Spell='Fireball'
Benefit:Magic Use@@*Specialty="Fire"
```

The single item form is used when the user automatically satisfies requirements: if the item already exists, only the option is set.

Number of Items in a List

The number of items required in the specified list. The following indicates three spells are required.

```
#10,Spells
```

Number of Items in a Category

The number of items in a specified category in a list. The following indicates that 10 Earth spells are required:

```
#10,Spells:Earth
```

Number of Items in a Number of Categories

The following indicates that at least 2 spells from at least six categories are required.

```
#2,6,Spells
```

Message

Any requirement can be followed by ":" and a message that will be displayed if the requirement is not satisfied. This allows explicit descriptions of requirements that are not self-explanatory. For example,

```
$Int>=10::Intelligence must be 10 or greater
```

If the message begins with a "\$" it will be parsed as an expression in the context of the character sheet. For example,

```
$Int>=10::$format('Intelligence must be 10 or greater. It is %d.',Int)
```

Data Sheet Option Format

Options consist of a keyword, a name and one or two value strings. Value strings may be simple text, or they may be prefixed by special character sequences to accomplish more complex functions:

- ? : Option value lists (p. 224): select a value/cost from a list of possibilities.
- ?? Computed option (p. 224): compute the cost of an option with a formula from a level, with optional check expression and check message.
- ?+ Checkbox option (p. 225): a checkbox on the option controls whether it is included in the computation of the cost.
- ?! Item references (p. 224): reference values from other items.
- ?@ Evaluated text (p. 224): generate random initial values for options, or perform other functions to compute the initial value.

Additions

Adds (p. 95) to the cost of the item. Values consist of a name and a value. The number can be preceded by a '+' or a '-'. If no sign is indicated, it's assumed to be an addition rather than a subtraction.

```
add Addition +10
add Subtraction -10
```

Adjustments

Makes an adjustment (p. 95) to a variable or set of items. Consists of a name, an adjustment expression and a value that's substituted for "x" in the adjustment expression

```
adj Adjustment Var+x 1
```

Auxiliary Cost

Auxiliary cost (p. 93) options consist of a name, an expression and a value. The 'x' in the expression is replaced with the value when the auxiliary cost is computed.

```
auxcost "Auxiliary Cost" 5*x 1
```

Expression

An expression (p. 93) option consists of a name and an expression (p. 169). The variable 'x' is replaced with the current value of the item, and the variable 'c' is replaced with the cost.

```
exp Expression x
```

Fraction

A fraction (p. 93) consists of a name and a value, which is a fraction. The sign is optional, and positive is assumed if omitted. The value may have a whole number followed by a fraction (separated by a space).

```
frac "Attack vs. Limited Defense" "+1 1/2"
```

Multiplier

A multiplier (p. 94) consists of a name and multiplier value. The first character of the value can be either '*' or 'x'. It can also be omitted. You can multiply by fractions by using a slash between the numerator and the denominator.

```
mult Multiplier *1/2
```

Percentage

A percentage (p. 94) consists of a name and a value. The first character is an optional sign, which is assumed to be '+' if omitted. The '%' at the end is also optional, but should be used as a hint to the user.

```
percent Percent +10%
```

Text

A text (p. 96) option consists of a name and a string of text.

```
text Note "Some Text"
```

Check Expression and Item References

Option check expressions and item references must be specified in an option `begin ... end` block after the option. For example,

```
begin text Limit "20" checkexp "x<=`Limit`" end
```

declares a text option with a check expression that requires the level of the item not exceed the value in the text option.

```
checkexp "expression;message"
```

Expression is the check expression (p. 101) for the option, while message is the message displayed if it evaluates to false. References to "x" in the expression are replaced by the level of the item. To refer to the option value, place the name of the option in backquotes: ``Option Name``.

```
itemref "varname"
```

Varname is the name of the variable associated with the item being referenced.

```
itemrefcat "Category"
```

If indicated, only items belonging to this category will be listed in the dropdown list for the item reference. Useful for limiting the items displayed to only those that qualify.

Stopping Option Printing

To prevent an option from appearing in the list of options that `@option ... @endoption` produces, place "noprint" in a `begin ... end` block for the option. For example,

```
begin exp Damage "kadmig(2)" noprint end
```

You're likely to use this in cases where the option is referenced in the format, making it redundant and unnecessary to be displayed again.

Display Expressions

A powerful method for formatting options any way you desire is available through display expressions. For example,

```
opt begin add "Non-Combat Multiplier" "??1;0;(x-1)*5;x>0"  
  dispexp "format('1: %d, +o', 2^((o)/5+1))"  
end
```

Displays "Non-Combat Multiplier: ×2, +0" initially .

Option Alias

The `alias` keyword indicates the alias for the option name, usually an abbreviation. It must be contained in a `begin ... end` block. For example,

```
begin frac "No Normal Defense" +1/2 alias NND end
```

Option Categories

Categories can be specified on options, in addition to items. The expression `inCategory("Primary Skill")` would return true for an item that contains the following option:

```
begin text "Primary Skill" cat "Primary Skill" end
```

Use Option Check Expression Message

If no check expression message is specified for an item, the `usemsg` tag indicates that the check expression message on the option will be used instead.

```
begin text "Cinematic Skill" ""  
  cat "Cinematic Skill" usemsg noprint  
  checkexp "machk(x);Cinematic skill level is incorrect." end
```

Keeping Old Values on Conversion

When an option appears on an item and you intend that its value be changed by the user and retained, you should place the `keepold` keyword on the item:

```
begin text Weapon Choose keepold end
```

If you omit `keepold`, the changed option will be added in addition to the original option when the character sheet is converted, resulting in two options with two different values.

Item Selection

If the value of the option is an item name, the `selitem` keyword specifies how the item should be selected:

```
begin text Ability "(Choose)" selitem "*Abilities:Combat" keepold end
```

This would make a browse button available that displays the items in the Combat category in the available Abilities list.

No Inherit

If no `inherit` is set on an option, children of the item do not inherit the option.

```
begin text Number "??1" noinherit end
```

Computed Option Value

A computed option value is a two-part value consisting of a level and a computed value. The format of the string is:

```
??level;cost;cost expression;check expression;check message
```

The purposes of these fields are explained in the change option value type dialog (p. 99) in more detail. The user changes the level directly, which sets the cost based on the cost expression. The check expression is used to verify that the level is valid, and the check message is displayed if it is not.

Option Value List Format

The option value list allows you to specify a list of labels and values from which the user can choose. The format of the string is:

```
?:index;label1,value1;label2,value2;...;labeln,valuen
```

The index indicates which label/value pair is selected. The range is from 1 to n, where n is the number of label/value pairs in the list. If a label or value is omitted, the value and the label are the same. The purposes of these fields are explained in the change option value type dialog (p. 99) in more detail.

Item Reference Option Value

The option value is an expression that can reference another item. The referenced item must be specified with an `itemref` tag in the basic option.

The item reference option value starts with "?!". For example,

```
?!-@v@
```

returns the value of the referenced item's level, negated.

The values in the option are referenced with `@...@` references (p. 102).

Evaluated Text Option Value

This might also be called the "random option" value. The text of the option is evaluated the first time the item is added. This is especially useful if you wish to generate random values for the option.

```
?@rand(6)
```

This generates a random number from 1 to 6, which becomes the option value. A more complex example combines two option value types:


```
text Sex ?@format('?:%d;Female;Male',rand(2))
```

This creates a text option that is a list with two values, Female and Male, and randomly selects one when the item is added.

Any expression (p. 169) is valid after the ?@.

Checkbox Option Format

To make an option's value be a "toggle," use the ?+ prefix for the option value. For example:

```
opt
  add "Immune to Mental Powers" "?+0;+20"
  begin text "Affect Totals" "?+1;" dispexp '1' noprint end
end
```

The first option will cause 20 to be added to the total cost of the item when checked, while the second option will return 1 for its value when checked, and 0 when not checked.

The value after the ?+ is 0 if the checkbox is not checked. It is 1 if the checkbox is checked. The value after the semicolon (if present) will be the value of the option when it is checked. If that value is omitted, the value of the option will be the value of the check (0 or 1).

When checked, the option will perform its normal function (performing an addition, multiplying by a fraction, etc.). When unchecked, it will not be included in the computation of the cost and it will not be listed in @options ... @endoptions. You can always directly reference the value through the optValue function and other mechanisms.

Data Sheet Macros

You can define macros in data sheets to reduce the amount of typing you have to do. A macro is defined in the following way:

```
$$macro mac(arg1, arg2)
{{
    # Macro body
}}
```

The macro body can contain any valid text. You can reference the values of the arguments in the macro body in the following manner: \$(arg1). Conditional expansion (with \$\$ifdef and \$\$ifnull) allow you to ignore null arguments. There is a limit of 30 arguments for a macro.

Invoking Macros

Macros are invoked in two ways: parentheses- and comma-delimited, and tab-delimited. The method of invocation is up to the implementer. Both methods have their uses.

Tab-delimited invocation is most useful for macros that fit on a single line: generally items that have few parameters with short values. Comma-delimited invocation is useful when there are many optional parameters, long parameters, or if the macro cannot fit on a single line.

Macro arguments may be passed to other macros. If the argument is to be passed directly to the macro, it appears by itself, with the \$() decoration. If any other text appears with the argument reference, the whole argument must be double-quoted. For example:

```
$$macro Skill(name, cost)
{{
    $$item$(name)
    $$formula("x*$(cost)")
}}
```

In the call to \$\$item the name argument is passed directly. In the call to \$\$formula other text appears with the argument reference, so the entire value must be quoted.

Parentheses- and Comma-Delimited Macro Invocation

```
$$mac("Value for arg1", Value for arg2)
```

Arguments to the macro are parsed differently depending on whether they are enclosed in quotes. If no quote is present, the argument is ended when a comma or right parenthesis is encountered. If a quote is present, the argument is considered to be all text until a matching quote is found. If you wish to include a quote in a string, you can escape it with ". If you wish to include a comma or right parenthesis in an argument value, you must enclose the argument value in quotes.

If the argument reference within a macro body is included within quotes, any quotes surrounding the argument value are discarded when the argument is expanded within the macro body. This allows the following macro to work properly:

```
$$macro quotes(value) {{ item "$(value)" }}
$$quotes(String with blanks and no comma)
$$quotes("(String with blanks, a comma and parentheses.)")
```

Generally speaking, it's probably safest to enclose argument references in quotes so that the value is expanded into a delimited string.

Tab-Delimited Macro Invocation

For certain kinds of data it's much easier to use tabs to separate the arguments, rather than commas. When you use tabs you don't need to put double quotes around arguments that contain commas or parentheses. However, all macros must fit on a single line for tabbed macro invocations.

To use tabs, simply type tabs in place of the initial left parenthesis and the commas. The end of the line terminates the macro invocation. For example,

```
$$macro skill(name, diff) {{ ... }}

$$skill Typing Average
$$skill Teaching Average
$$skill Computer Use Hard
```

Note that when you use tabs the columns will not always line up with each other, if the text before the tab goes beyond the "tab stop." An editor that shows "hidden characters" can be very useful for tabbed data entry.

Omitted Arguments

If you omit an argument, its value is the null string. You can test to see whether arguments are null or defined within the body of the macro. This allows for "conditional expansion" of the macro. With this, you can create "default values" for omitted parameters, or simply omit the keyword/value pairs for the omitted argument.

For example, the following will define an item and an optional requirement:

```
$$macro item(name, req)
{{
    item "$(name)" $$ifdef(req) prereq $(req) $$endif
}}
```

The following are some examples of how macros with omitted arguments are expanded:

Macro Call	Expanded Value
item(I1,"Int>=10")	item "I1" prereq "\$Int>=10"
item(I1)	item "I1"
item()	item ""
item(,"Int>=10")	item "" prereq "\$Int>=10"
item("I1")	item "I1"

The same rules apply for omitted arguments when the macro invocation is tab-delimited, but it may be difficult to see where arguments were omitted unless you use an editor that displays "invisible" characters.

\$\$define

Global macro variables may be defined with the \$\$define macro command:

```
$$define defcostform ceil((c+a)*max(m,0.2))
```

The value of the variable is all text to the end of the line after the macro name, excluding initial white space.

Global macro variables may be referenced anywhere in the data sheet, not just within in the body of a macro. They are referenced by surrounding the name with \$(). For example,

```
totalcostformula "$(defcostform)"
```

Conditional Macros

There are three conditional macro commands: \$\$if, \$\$ifdef and \$\$ifnull. They have the following general syntax:

```
$$ifdef(arg) Expanded if arg is non-null
$$elseif(arg = 'Value') Expanded if value of arg is 'Value'
$$else Expanded if null
$$endif
```

The conditional macro commands may appear only inside macro bodies (between the {{ and }} braces).

\$\$ifdef

If the macro argument (or global macro variable) named in the `$$ifdef` is defined (non-null), the text between the `$$ifdef` and the corresponding `$$else` (or `$$endif` if no else is defined) is emitted. The macro argument must be defined, either through a `$$define` macro command, or as the result of argument name declaration in a macro definition.

If the argument is null (the empty string) the text between the `$$else` and the `$$endif` is emitted.

\$\$ifnull

The text in the body of the `$$ifnull` is emitted if the argument is null. If the argument has a non-null value, the text between the `$$else` and `$$endif` is emitted, if present.

\$\$if

If the expression in the parentheses is true, the text between the `$$if` and the `$$else` (or subsequent `$$elseif`) is emitted. If the expression is false, the next `$$elseif` is evaluated, otherwise the text between the `$$else` and the `$$endif` is emitted, if present.

Macro arguments and `$$defines` must appear within the parentheses without the `$()` decoration.

For example, if `def` is a macro parameter:

```

    $$if(strindex(def, ","))
        default "max($ (def)) "
    $$else
        default "$ (def) "
    $$endif

```

the following values of `def` will produce the following text:

Value of <code>def</code>	Emitted Text
INT-4	INT-4
DEX-4,Sword-2	max(DEX-4,Sword-2)

\$\$elseif

If the expression in the parentheses is true, the text between the `$$elseif` and the next `$$elseif`, `$$else` or `$$endif` encountered is expanded.

Expression Evaluation

The `$$exp` macro command evaluates the expression and emits the value of the expression. For example:

```
varname $$exp("replaceString(name, \"/[^a-zA-Z]/\", '')")
```

will strip all non-alphabetic characters from `name` and emit them.

Quotes are required around the expression. Arguments of the current macro and any global defines may be referenced within the expression unadorned. For example, if you had a macro with numeric arguments of `lev` and `number` you could evaluate them into a number with the following:

```
adj limit+$$exp("lev*number")
```

\$\$scan

The scan macro command scans the value of a macro argument or global macro variable according to a regular expression (p. 11) pattern and assigns the parts that match to global macro variables. Parts of the regular expression must be enclosed in parentheses to indicate which part of the match is assigned to the variables.

For example,

```

$$macro dmgDice(dmg) {{
    $$scan(dmg, "([0-9])d([-+][0-9])*", dice, adds)
}}

```

produces the following values.

Invocation	dice	adds	Note
dmgDice(4d)	4		The empty string matched adds (the * allows no add to be present).
dmgDice(4d-2)	4	-2	Both matched.
dmgDice(3d+1)	3	1	Both matched.
dmgDice(4)			Nothing matched because the "d" was missing.

If there is no match, the target variables are set to null (the empty string).

\$\$repeat

Repeat a sequence of text. The basic syntax is:

```

$$repeat($(variable), iterator, "delimiter", "separator")
...
$$endrepeat
or
$$repeat("string", iterator, "delimiter", "separator")
...
$$endrepeat
or
$$repeat($(variable), iterator, "delimiter")
...
$$endrepeat

```

The first argument of the repeat structure is the text that serves as the source for the *iterator* argument. It may be a variable reference (surrounded by `$()`: i.e., `$(variable)`) or a string constant (surrounded by double quotes). The text between the `$$repeat(...)` and `$$endrepeat` is emitted once for each substring of the first argument's value, as delimited by the *delimiter* string. The *delimiter* string is a regular expression (p. 11) pattern.

Within the body of the `$$repeat` references to *iterator* are replaced with the current value from the source. The value of *separator* (if specified) is emitted after all iterations but the last.

Repeats may be nested. The iterator variable may be referenced for the repeat by surrounding the iterator name with `$()`. Iterators for repeats that include other repeats may also be referenced within the inner repeats, as can global defines and arguments of an enclosing macro.

For example, the following sequence:

```

adj "$$repeat("Head, Body, Arms", a, " ", "*", ";")\
$(a)+`DR`\
$$endrepeat"

```

produces the text:

```
adj "Head+`DR`;Body+`DR`;Arms+`DR`"
```

The following sequence:

```

$$macro item(name) {{ item "$(name)" }}
$$define numbers One,Two,Three,Four,Five,Six,Seven,Eight,Nine,Ten
$$repeat($(numbers), num, ",")\
    $$item($(num))
$$endrepeat

```

emits this text:

```

item "One"
item "Two"
item "Three"
item "Four"
item "Five"
item "Six"
item "Seven"
item "Eight"
item "Nine"
item "Ten"

```

\$\$error

Report an error in macro expansion. Reading of the data sheet will be terminated. The `$$error` macro command may be invoked only within macros.

```

$$macro skill(name, cost) {{
    $$ifnull(name)
        $$error(Name for skill may not be omitted)
    $$endif
    ...
}}

```

\$\$macrodesc

Describe a macro. This must appear at the beginning of the macro body (after the `{{`). When **GURPS Character Builder** edits data sheets (p. 144) this description is displayed in the status bar when you select the macro name.

```

$$macro skill(name, type)
{{
    $$macrodesc("Skill Macro")
    $$argdesc(name, The name of the skill)
    $$argdesc(type, "The type of the skill: \"Hard\", \"Average\" or \"Easy\"
(optional, default is Average)")
    ...
}}

```

The argument to `$$macrodesc` is a string. If any right parentheses appear within the string, you must enclose the string with double quotes. If double quotes appear, they must be escaped with backslashes. Backslashes must be similarly escaped.

Macro and argument descriptions should be brief, as they are displayed in the status bar of **GURPS Character Builder**. No absolute maximum width can be given, though eighty characters would probably be displayable on most users' monitors.

\$\$argdesc

Describe a macro argument. This must appear after the `{{` of a macro definition.

```

$$argdesc(argname [ : type ] , argdesc)

```

argname The argument name. This must be one of the argument names defined in the `$$macro` definition.

type The argument type. If the colon and *type* are omitted, then the argument is assumed to be plain text. Other types indicate that a special argument editing dialog is used. The following predefined tags can be specified:

adjustment or adj

Adjustments. The adjustments dialog will be invoked.

autoitems or auto

Automatic items. The automatic items dialog will be invoked.

category, categories or cat

Categories. The category dialog will be invoked.

requirement or req

Requirements. The requirements dialog will be invoked to edit this argument.

text

The normal text editor will be invoked.

Other user-defined argument types (see `$$argtype` (p. 231)).

argdesc The argument description. This string must be enclosed in double quotes if it includes double quotes or right parentheses (see `$$macrodesc` (p. 230) for an example).

Argument descriptions are displayed in the status bar when editing an argument in the data sheet (p. 144) window.

\$\$argtype

Define an argument type, which is used in an `$$argdesc` to identify the type of an argument. This is used to indicate how the argument is edited in a data sheet.

The general syntax is:

```
$$argtype(name, class, "Dialog Title", "User Prompt", ...)
```

where ... indicates additional arguments for the particular class of argument type.

The values for *class* can be the following:

list Indicates the argument is a list of entries separated by a delimiter. The selection of the values to be included in the list can be "free-form" (p. 153) (the user types anything), or from a list of choices (p. 153), or a combination of the two (p. 152). The rest of the arguments are:

```
$$argtype(name, class, "Dialog Title", "User Prompt",
"delimiter" [ , "choices" ] )
```

The *delimiter* is a string that contains a single non-blank delimiter character, plus any extra blanks that you might wish to include for legibility. For example, " ; " or " , ".

The *choices* is a list of strings separated by the | character. If one of the choices is "*", the editing dialog will allow the user to enter any text desired, in addition to the other listed choices. If the *choices* argument is omitted completely, no list of choices will be presented, but an edit field will be present.

For example, the following would display a list of four choices (Fire, Air, Water and Earth) and an edit field into which the user could enter any text to be added to the list.

```
$$argtype(spellcat, list, "Spell Categories",
"Select one or more spell categories.", ";",
"Fire|Air|Water|Earth|*")
```

After the editing dialog closes, the value of the argument will be something like "Fire;Water".

Directives can be included in the choices to further modify the behavior of the interaction. A directive begins with a question mark, is followed by a colon, and then its arguments.

?list List items in the choices from the specified list and optional list of categories. The argument is the list name, optionally followed by a colon and the category names separated by commas. If the category name is preceded by an exclamation point, all categories are included except the named one.

For example,

```
$$argtype(abilities, list, "Abilities",
          "Choose Abilities", ";",
          "?list:Abilities:!Note|*")
```

?choose Display a button (...) after the **Value** field that allows the user to choose from a list and optional category. This automatically indicates that any text can be entered.

For example:

```
$$argtype(abilities, list, "Abilities",
          "Choose Abilities", ";",
          "?choose:Abilities")
```

?checkexp Define a regular expression (p. 11) after the colon. If the text the user enters doesn't match that regular expression when the **Add** button is clicked, the text specified in the **?checkmsg** is displayed, and if no text was specified, a general message is used. If any special characters are used (\ or |), they must be preceded by an escape sequence (\\).

For example:

```
$$argtype(choicetype, list, "Choices",
          "Enter the name, a comma, and the cost", ";",
          "?checkexp:., *[-+]?[0-9]+|?checkmsg:The choice must
          include a name and cost, separated by a comma. The
          cost should be a + or - followed by a number.|*")
```

duplist The same as **list**, except that duplicates are allowed in the list of choices.

choose Indicates the argument is a single entry chosen from a list of choices (p. 152). The choices can be selected from a list, or if allowed, entered in an edit field.

```
$$argtype(name, class, "Dialog Title", "User Prompt",
          "choices" )
```

The choices are as above for a list. If "*" is listed as one of the choices, the edit field in the dialog is enabled.

For example, the following defines an argument type that can have the values Easy, Medium, Hard and Very Hard.

```
$$argtype(skilldiff, choose, "Choose Difficulty",
          "Choose the difficulty of the skill.",
          "Easy|Medium|Hard|Very Hard")
```

The **? directives** for **list** can also be specified for **choose**.

\$\$hidemacro

Prevent the display of the macro in the list of available macros in the data sheet (p. 144) window.

Limitations

Macros may be nested to a maximum depth of 10. If statements also have a maximum nesting of 10.

Text Data Sheet Include

You can include other files in a data sheet with the `$$include` directive:

```
$$include filename.inc
```

It is most useful to place commonly used macros in include files so that you can keep a single copy of the macros in one place.

Include files may be nested, but recursion is not allowed. That is, include file A may not include itself or any other file that includes it directly or indirectly. Files may be included within the body of a macro. If this is done, you should ensure that the macro is only used in one place in a data sheet, otherwise any definitions in the include file will be duplicated.

Character Sheet Format

The following illustrates the text format of a character sheet. Items in the item lists have the same format as data sheet items (p. 217). Lines beginning with "#" are comments. If a string doesn't have any blanks, you don't need to surround it with quotes, but if, for example, the name of dialog had blanks in it, you must enclose it with double quotes.

```
# The first line of the character sheet must be the following:
"Character Sheet file v. 1"

# The name of the game system.

gamesystem "Game System"

# The version of this format.
version 240

# The "level" of the character sheet. Used to identify differing
# versions of the same character sheet.
level 1

# The number of pixels per inch used to create the character sheet.
pixinx 96 pixiny 96

# The list of datasheets to load with this character sheet.
datafile gamesys.cds end

# The template that this character sheet is based on.
template gamesys.cst

# The name of the dialog is followed by the width, height of the dialog.

dialog Dialog 236 244
  # The location on the screen. -1 -1 gives the default positioning
  # determined by Windows.
  loc -1 -1

  # The font, font size and character set used for the dialog.
  font "MS Sans Serif" 8 charset 0

  # A text control specifies the text of the label, x, y,
  # width, height and alignment.

  text Name: 6 12 34 13 left

  # An edit text control has the text value, x, y, width,
  # height and the name of the control.

  edittext "" 45 11 178 20 Name

  # An editvar control has the value, x, y, width, height and
  # name of the variable that the value is assigned to.

  editvar 10 4 44 30 20 str
  text STR 44 47 40 13 left
end
```

```
# A list of items.

list Skills
  # The category in the data sheet from which the selection is
  # made for this list.
  catname Skills

  # The name of the variable that is assigned to the sum of the
  # costs for this list.
  var Skills 0.000000

  # The prompt at the bottom of the list window.
  listprompt Total

  # The font name, font size and character set used to display
  # this list.
  font "MS Sans Serif" 8 charset 0

  # The location and size of the list.
  loc -1 -1 275 251

  # The items in the list follow. The have the same format as
  # items in a data sheet.

  item "Basket Weaving"
    level 5
    formula x
    format %0ln%-5rc
  end
end
```

Details

pixinx	The number of pixels per inch horizontally for this character sheet. This allows the reader to scale the dialog coordinates for the display that the character sheet is loaded on. Whenever the character sheet is written, this information is updated for the current machine. If omitted, the value of 96 pixels per inch is assumed.
pixiny	The number of pixels per inch vertically.

BNF for GURPS Character Builder

BNF (Backus-Naur Form) is a formal notation for describing the syntax of computer languages. The definition of the metasymbols used are:

Keyword	A literal keyword
[]	Zero or more instances.
()	Bracketing symbols.
...	Repeat the last item any number of times.
	Separates alternatives.
<construct>	Identifies a syntactic construct.
::=	A syntactic production. The construct named on the left is defined by the definition on the right.
'x'	Literal character.

Expression Syntax

The following section describes the syntax of the language that defines variables and functions.

```

<function definition> ::= <expression> | <procedure definition>
<procedure definition> ::= '{' <local definition> <statement> ... <return>
    '}'
<local definition> ::= local <variable name> [ , <variable name> ... ';'
<statement> ::= <return> | <while> | <if> | <assignment> | ';'
<return> ::= return <expression> ';'
<while> ::= while '(' <expression> ')' <statement> ... endwhile
<if> ::= if '(' <expression> ')' <statement> ...
    [ elseif '(' <expression> ')' <statement> ... ]
    [ else <statement> ... ]
    endif
<assignment> ::= <local variable> '=' <expression> ';'
<expression> ::= <ternary expression> | <logical expression>
<term> ::= [ <unary operator> ] ( <constant> | <function call> | <symbol> |
    '(' <expression> ')' ) | <symbol> '[' <expression> ']'
<binary operator> ::= '+' | '-' | '*' | 'x' | '/' | '^' | mod
<unary operator> ::= '-' | not
<boolean operator> ::= '=' | '!=' | '<>' | '<' | '>' | '<=' | '>='
<logical expression> ::= <boolean expression>
    [ <logical operator> <boolean expression> ] ...
<boolean expression> ::= <arithmetic expression>
    [ <boolean operator> <arithmetic expression> ]
<ternary expression> ::= <boolean expression> '?'
    <boolean expression> : <boolean expression>
<logical operator> ::= or | and

```

```

<function call> ::= <symbol>
    '(' <expression> [ ',' <expression> ] ... ')'
<arithmetic expression> ::= <term> [ <arithmetic operator> <term> ] ...
<constant> ::= <number> | <string>
<symbol> ::= [ '~' | '&' ] ( <letter> | '_' ) [ <character> | <digit> | '_'
    ] ...
<number> ::= [ <digit> ... ]
    ( '.' <digit> ... | '¼' | '½' | '¾' ) [ '%' ]
<string> ::= ''' <character> ... ''' | ''' <character> ... '''

```

Command Script BNF

The command script language is interpreted text, as opposed to a interpreted compiled form. It is an extension of the expression syntax. Any constructs not defined in this section use the definition from the previous section.

Comments may appear anywhere. A '#' character starts the comment, which continues till the end of the line.

```

<script> ::= <script ID>
    [ <game system> | <title> | <submenu> | <menuitem> ] ...
    <command> ...
    [ end ]
<script ID> ::= "Conversion script v. 1"
<gamesystem> ::= gamesystem <string> <string>
<title> ::= title <string> ;
<submenu> ::= submenu <menu name> ';' ( <submenu> | <menu item> ) ...
endsubmenu
<menu item> ::= menuitem <menu text> [ ',' <key equivalent> ]
    '{' <command> ... '}' ';'
<command> ::= <add> | <assignment> | <array> | <call> | <checkallreq> |
    <checkreq> | <context> | <copyall> | <copydialog> | <copyitem> |
    <copylist> | <copynotes> | <copyoption> | <copyright> | <datasheet> |
    <deleteitem> | <dialog> | <edititem> | <end> | <exec> | <exclude> |
    <exit> | <fail> | <for> | <foreach> | <func> | <getfilename> |
    <getitem> | <include> | <insertitem> | <item> | <list> | <notes> |
    <opensublist> | <openwindow> | <option> | <options> | <return> |
    <script mode> | <selectitem> | <set> | <showallreq> | <showreq> |
    <sortlist> | <sub> | <switch> | <unconverted> | <var> | <while>
<add> ::= add [ <list name> '.' ] <item name> [ '[' <item name> ']' ]
    [ '=' <expression> ] ';'
<array> ::= array <symbol> '[' [ <integer> ] ']'
    [ '=' <constant> [ ',' <constant> ] ... ] ';'
<assignment> ::= (
    [ '$@' ] <symbol> |
    <dialog name> '.' <field name> |
    <list name> '.' <item name>
) '=' (
    <expression> |
    <sub name> '(' [ <expression>
        [ ',' <expression> ] ... ] ')' |

```

```

        '$$' <dialog name> \. ' <field name>
    ) ';'

<call> ::= [ call ] <sub name> '(' [ <expression>
    [ \, ' <expression> ] ... ] ')' ';'

<checkallreq> ::= checkallreq <symbol> ';'

<checkreq> ::= checkreq <symbol> ';'

<copyall> ::= copyall

<context> ::= context '(' [ destination | source ] ')' <command> ... endcontext

<copydialog> ::= copydialog <dialog name> <dialog name>

<copyitem> ::= copyitem <list name>

<copylist> ::= copylist <list name> <list name>

<copynotes> ::= copynotes ';'

<copyoption> ::= copyoption ';'

<copyright> ::= copyright <string>

<datasheet> ::= datasheet [ '@' ] <file name>
    [ \, ' [ '@' ] <file name> ] ... ';'

<deleteitem> ::= deleteitem ';'

<dialog> ::= dialog <symbol> \, ' <dialog text> ';'
    <dialog command> ...
end

<dialog command> ::= <text> | <checkbox> | <combobox> | <number> | <dropdown>
    | <listbox> | <treelist> | <editcombo> | <edittext> | <button> |
    <defbutton> | <keyword>

<text> ::= text <dialog text> ';'

<dialog text> ::= <string> | <any text but semicolon>

<checkbox> ::= <symbol> \, ' ( <any text> | <string> ) ';'

<combobox> ::= combobox <symbol> \, ' <prompt text>
    ( \, ' <option text> \, ' <integer> ) ... ';'

<prompt text> ::= <any text but comma> | <string>

<number> ::= number <symbol> \, ' <dialog text> ';'

<dropdown> ::= dropdown <symbol> \, ' <prompt text>
    ( \, ' ( <prompt text> | '$$' <symbol> ) ) ... ';'

<listbox> ::= listbox <symbol> \, ' <prompt text>
    ( \, ' ( <prompt text> | '$$' <symbol> ) ) ... ';'

<treelist> ::= treelist <symbol> \, ' <prompt text>
    ( \, ' <treelist argument> ) ... ';'

<treelist argument> ::= <prompt text> | '$$' <symbol> |
    \{ ' <treelist argument> [ \, ' <treelist argument> ] ... \}'

<editcombo> ::= editcombo <symbol> \, ' <prompt text>
    ( \, ' ( <prompt text> | '$$' <symbol> ) ) ... ';'

```

```

<edittext> ::= edittext <symbol> ',' <dialog text> ';'
<button> ::= button <expression> ',' <dialog text> ';'
<defbutton> ::= defbutton <expression> ',' <dialog text> ';'
<keyword> ::= keyword <dialog text> ';'
<edititem> ::= edititem <symbol> ';'
<matchbox> ::= matchbox <array name> ',' <array name> ',' <dialog text> ';'
<end> ::= end
<exec> ::= exec <file name> ';'
<exclude> ::= exclude ';'
<exit> ::= exit ';'
<fail> ::= ( fail | warn ) <any text but semicolon> | <string> ';'
<for> ::= for '(' <symbol> '=' <expression> ';' <logical expression> ';'
    [ <symbol> '=' <expression> ] ')'
    <command> ...
endfor
<foreach> ::= foreach '(' [ '*' ] <list name> [ ':' <category name> ] ')'
    <command> ...
endforeach
<func> ::= func <symbol> '(' <symbol> [ ',' <symbol> ] ... ')'
    <procedure definition>
<getfilename> ::= getfilename <symbol> ','
    ( <string> | <any text but comma and semicolon> ) ','
    ''' <wildcard pattern> [ '|' <wildcard pattern> ] ... ''' ';'
<wildcard pattern> ::= <prompt> '|' <filename pattern> [ ';' <file name
    pattern> ] ...
<getitem> ::= getitem <list name> [ ':' <category name> ] ','
    <variable name> ';'
<include> ::= include ( <file name> | '$$' <symbol> ) ';'
<insertitem> ::= insertitem <list name> [ ':' <category name> ] ';'
<item> ::= item '(' <list name> ','
    ( <string expression> | <numeric expression> ) ')'
    <command> ...
enditem
<list> ::= list <list name>
    [ destination <list name> ';' ]
    [ <options> ]
    ( <list targets> '~' <command> ... end ) ...
    [ skip ]
    [ unconverted <list name> ]
endlist
<list targets> ::=
    ( sublist | <item name> ) [ ',' ( sublist | <item name> ) ] ... |

```

```

category <category name> [ ',' <category name> ] ... |
default | samename

<notes> ::= notes ( <text> | <string> ) ';'

<opensublist> ::= opensublist <list name> '.' <sublist name>

<openwindow> ::= openwindow <window name>

<option> ::= option <option name> [ '[' <name> ']' ] [ = <expression> ]

<options> ::= options [
    ( default | <option name> [ ',' <option name> ] ... ) '~'
    <command> ... end
] ... endoptions

<return> ::= return [ <expression> ]

<script mode> ::= scriptmode

<selectitem> ::= selectitem '(' <list name> ','
    ( <integer expression> | <string expression> ) ')'

<set> ::= set <set keyword> '=' <expression>

<set keyword> ::= level | cost | constcost |
    ( ( '$*' | '$@' ) ( default | level | name | origname | autoParent | autoID |
        varname | notes ) )

<showallreq> ::= showallreq ';'

<showreq> ::= showreq ';'

<sortlist> ::= sort <list name>
    [ ',' <sort key> [ ',' <sort direction> ] ]

<sort key> ::= name | category | cost | level

<sort direction> ::= forward | reverse

<sub> ::= sub <subroutine name>
    '(' <parameter name> [ ',' <parameter name> ] ... ] ')'
    '{' <command> ... '}'

<switch> ::= switch '(' <expression> ')'
    [ [ <expression> [ ',' <expression> ] ... '~'
        <statement> ...
        end
    ]
    [ default '~' <statement> ... end ]
endswitch

<unconverted> ::= unconverted <string>

<var> ::= var <variable name> '=' <expression>

<while> ::= while '(' <expression> ')' <command> ... endwhile

```


Adjustment Syntax

The BNF syntax for adjustments is as follows:

```

<adjustments> ::= <adjustment list> [ '!' <adjustment list> ]
<adjustment list> ::= <adjustment> [ ';' <adjustment> ]
<adjustment> ::= <target> <operator> <expression>
<target> ::= <variable name> | <category> | <original name>
<operator> ::= '=' | '+' | '-' | '*' | '/'
<expression> ::= Any legal expression, also including <special symbol>
<special symbol> ::= 'x' | 'c' | '`' <option name> '`' |
    '~' <option name> '`' | '@' <option item reference> '@'
<option item reference> ::= option item reference (p. 102)
<variable name> ::= Any legal variable name
<category> ::= '"' <simple list name> ':' <simple category name> |
    '"' <list name> ':' <category name> '"'
<original name> ::= '"' <simple list name> ':' <simple item name> |
    '"' <list name> ':' <item name> '"'
<simple category name> ::= <simple name>
<simple list name> ::= <simple name>
<simple item name> ::= <simple name>
<category name> ::= <name>
<item name> ::= <name>
<option name> ::= <name>
<simple name> ::= Name with only letters and numbers
<name> ::= Name with any character but colon and double quote

```

Shortcut File Format

Shortcut files define the items that appear in the Shortcuts window. All shortcut files are kept in the GURPS Character Builder Source Directory. Shortcut files are text-only files with a specific structure described below. By convention, shortcut files have an `.sct` extension.

The default shortcut file is called `Main.sct`. You should not modify `Main.sct` because it will be overwritten when updates are installed. Instead you should add your custom shortcuts to a file named `Custom.sct`.

The following is an example of a shortcut file.

```
[Main]
menu Help;
    menuitem GURPS Character Builder Help { HelpHelp }
    menuItem Game System Help Contents { HelpGameSystemContents }
    menuitem Game System Tutorial { HelpGameSystemTutorial }
endmenu
menu Character Sheets;
    menuitem Open Character Sheet { FileOpen }
    menuitem Create New Character Sheet { FileNew }
endmenu
```

Finding Shortcuts

When GURPS Character Builder starts up, it looks for a file named `Main.sct`. When no character sheet is open, the application displays the shortcuts in the `[Main]` section of `Main.sct`. It will also display any shortcuts listed in the `[Common]` sections of `Custom.sct` and `Main.sct`.

When a character sheet is open, the shortcuts for the currently open window are displayed. Each window is either a dialog or a list. When a dialog window is opened, the following files are searched for the following sections. When a match is found, the search stops.

File Name	Section Name
<code>Custom.sct</code>	[Game System:Dialog:Name] [Game System:Dialog] [Default]
The Shortcut File (p. 47) specified in the Character Sheet Info dialog.	[Dialog:Name] [Dialog] [Default]
The shortcut file for the game system, which is formed by removing all illegal file system characters and appending an <code>.sct</code> extension.	[Dialog:Name] [Dialog] [Default]
<code>Main.sct</code>	[Dialog:Name] [Dialog] [Default]

The logic is the same for list windows, except that the section `[List:Name]` is sought.

Common Sections

All common sections found in these files are also added to the shortcuts window, including `[Dialog:Common]` (or `[List:Common]`), `[Common]` and in `Custom.sct`, `[Game System:Dialog:Common]`.

Overriding Section Definitions

To override the definition of the shortcuts for a particular game system, add a specific section to the `Custom.sct` file. For example, to override the Fuzion shortcuts for the Skills and Options list, add a section named `[Fuzion:List:Skills and Options]`. To add an override for all dialogs in Fuzion, add a section named `[Fuzion:Dialog]` to `Custom.sct`.

Other Topics

Format of the shortcuts (p. 243)

Shortcut BNF (p. 244)

Shortcut Formats

The following is a general description of the shortcut formats. Click [here](#) for the technical BNF (p. 244).

Sections

Each section begins with a left bracket, followed by the section name, followed by a right bracket.

Sections may be qualified with colons for game system and section type: `[Fuzion:List:Skills and Options]`, `[Dialog:Attributes]`.

Menus

The general format for a menu is:

```
menu Menu Name;
  # Command list
endmenu
```

You may place other shortcuts within the menu as desired, including other menus.

The menu name is displayed in the shortcuts window with a plus or minus box in front of it to indicate that it can be expanded and contracted. Shortcuts in the menu are indented below.

For example, the following menu defines a number of shortcuts that open specific dialogs and windows using a variety of methods:

```
menu Change...;
  command Characteristics { openWindow Characteristics; }
  command Configuration { openWindow Configuration; }
  command Information { openWindow Information; }
  menuitem Notes {UtilitiesNotes}
  menuitem Details {UtilitiesDetails}
  command Picture {openWindow Picture; }
  command Life Path { openwindow Life Path; if ($+listcount(Life Path) =
0) exec LifePath.scp; endif }
endmenu
```

Menu Items

The general format for a menu item is:

```
menuitem Menu Item Name { MenuCommandKeyword }
```

or if the menu item has an argument:

```
menuitem Menu Item Name { MenuCommandKeyword : Argument}
```

The text of the menu item name is displayed in the shortcuts window. When the menu item is clicked, the menu command indicated by the `MenuCommandKeyword` is executed. The names of the menu commands are formed by removing all non-alphanumeric characters from the menu command name, and adding that to the menu name. For example, the **File | Open...** command is named `FileOpen`. The complete menu command keyword list (p. 245) is in the BNF section.

The first two example shortcuts execute the menu commands for opening the Notes and Details windows.

```
menuitem Notes {UtilitiesNotes}
```

```
menuItem Details {UtilitiesDetails}
```

The following prints a combat summary using the specified filter. When a filter is specified, that filter is selected and the Print on Same Page checkbox is set in the Multiple Character Sheet Filter dialog (p. 19).

```
menuItem Print Combat Summary {UtilitiesFilterCharacterSheets :
CombatSummary.flt}
```

Commands

The general format for a command is:

```
command Shortcut Name { command list; }
```

The command list is executed in the context of the current character sheet. Commands may appear only for Dialog and List sections. The command list may be the commands that appear in command scripts (p. 260).

For example, the commands below:

- Open the Picture dialog
- Open the Life Path list and execute the `LifePath.scp` script if there are no items in the Life Path list
- Open the Skills list and bring up the Available Skills dialog.

```
command Picture {openWindow Picture; }
command Life Path { openwindow Life Path; if ($+listcount(Life Path) = 0)
exec LifePath.scp; endif }
command Skills { insertItem Skills; }
```

Scripts

The general format for a script is:

```
script Script Name : ScriptFile.scp;
script Script Name : ScriptFile.scp { command list }
```

The first form simply executes `ScriptFile.scp` when the shortcut is selected. The second form loads `ScriptFile.scp`, executes it, then executes the command list in the brackets. The second form is intended for scripts that define common subroutines and variables, and then execute a specific subroutine within that script.

The following script shortcut executes the `perform` subroutine in the `Advance.scp` script file.

```
script Practice Skill: Advance.scp { perform('Practice'); }
```

This shortcut executes the `LifePath.scp` script.

```
script Generate Life Path and Origin : Lifepath.scp;
```

Shortcut File BNF

```
<shortcut file> ::= [ <comment> | <section> ] ...
<comment> ::= '#' Any text <eol>
<eol> ::= End of line character
<section> ::= '[' <section name> ']' <shortcut> ...
<shortcut> ::= <command> | <menu item> | <script> | <menu>
<command> ::= command <name> '{' <command list> '}'
<menu item> ::= menuItem <name> '{' <menu command keyword> '}'
<script> ::= script <name> ':' <command file name>
[ '{' <command list> '}' | ';' ]
```

```

<menu> ::= menu <name> ';' [ <shortcut> ] ... endmenu

<command list> ::= Command script command list (p. 260).

<name> ::= Any text excluding the following characters: { (left brace), ; (semicolon), [ (left
    bracket), : (colon).

<menu command keyword> ::=
    EditFilterCopy |
    EditSearch |
    EditSearchAgain |
    FileAddNewFile |
    FileAssociatedFiles |
    FileClose |
    FileConvert |
    FileExit |
    FileGenerateCharacter |
    FileLoadDataSheet |
    FileNew [ ':'
        <file type> [ ',' <game system> [ ',' [ <template name> ] ] ] |
    FileOpen |
    FilePageSetup |
    FilePrint |
    FilePrintAuxiliary [ <auxiliary file name> ] |
    FilePrintPreview |
    FilePrintPreviewThroughFilter |
    FilePrintPreviewThroughTemplate |
    FilePrintThroughFilter |
    FilePrintThroughTemplate |
    FileSave |
    FileSaveAs |
    FileSaveGroupThroughFilter [ <filter name> [ ',' <directory>
        [ ',' <default extension> ] ] ] |
    FileSetCopyPrintFilters |
    HelpAboutCreator |
    HelpContents |
    HelpGameSystemContents |
    HelpGameSystemInfo |
    HelpGameSystemTutorial |
    HelpHelp |
    HelpItemInfo |
    ToolsAdd1 |
    ToolsAddFiles |
    ToolsCheckRequirements |
    ToolsDeleteItem |
    ToolsEditProperties |
    ToolsInsertSublist |
    ToolsListSummary |
    ToolsModify |
    ToolsMoveDown |
    ToolsMoveUp |
    ToolsNotes |
    ToolsOpenCloseSublist |
    ToolsOpenFiles |
    ToolsRenameFile |
    ToolsResetAllFields |
    ToolsResetField |

```

ToolsSelectFromMasterList |
ToolsSelectItem |
ToolsShowRequirements |
ToolsSort |
ToolsSubtract1 |
UtilitiesDetails |
UtilitiesFilterCharacterSheets [':' <filter file name>] |
UtilitiesGameSystemPreferences |
UtilitiesNotes |
UtilitiesPreferences |
UtilitiesRecalculate |
UtilitiesSelectionRules |
UtilitiesShowShortcuts |
UtilitiesShowTipsWindow |
UtilitiesUpdateCharacterSheets |
WindowArrangelcons |
WindowCloseAll |
WindowOpenAll |
WindowOverlapWindows |
WindowPackWindows

<filter file name> ::= Name of a filter file in the source directory

Game System Configuration Parameters

Game system configuration parameters are defined in files that end with a `.pref` extension, and are stored in the source directory (usually called `C:\Program Files\GURPS`). These files are essentially Windows `.ini` files.

The `.pref` file must have a `[Configuration]` section, with a `GameSystem` parameter. The other sections in the file define parameter names that are referenced in character sheets, print templates and filters with the `ConfigParam` (p. 171) function. These options are in effect for all character sheets using the specified game system.

For example:

```
[Configuration]
GameSystem=Hero

[PrintOptions]
Prompt=Print Options
Type=checkbox
DefaultValue=1
```

The section name for a configuration parameter defines the name that is passed in a string to the `ConfigParam` function. The following parameters are defined for each configuration parameter:

Prompt	The prompt to display to the user in the Game System Preferences dialog (p. 15).
Type	The type of input control for the configuration parameter. The following values are supported: <ul style="list-style-type: none"> <code>prompt</code>: A simple text prompt. No input is allowed. <code>checkbox</code>: A checkbox. If the checkbox is checked, the value of the parameter is 1. If it is unchecked, the value is 0. <code>edittext</code>: any text string. <code>number</code>: A number. <code>dropdown</code>: A drop-down list. <code>listbox</code>: A listbox.
DefaultValue	The initial value of the parameter before the user changes it.
Values	For dropdowns and listboxes only. A semicolon-delimited list of the values that are used to populate the list. The <code>DefaultValue</code> should be included in that list.

There is no limit on the number of parameters that may be defined, though the dialog is limited to the size of the screen.

How It Works

When **GURPS Character Builder** starts, it reads all the `.pref` files in the source directory. Then, when the user invokes the **Utilities | Game System Preferences...** command, **GURPS Character Builder** builds a custom dialog from the parameters described in the `.pref` files for the current game system. **GURPS Character Builder** remembers the settings that the user changes in the Windows Registry.

Changing the value of a configuration parameter does not automatically update any character sheet variables that depend on it. References in filters, however, are evaluated when the filter is executed.

Referencing Game System Configuration Parameters

The parameters are referenced through the `ConfigParam` function. For example, you could write the following in a filter:

```
@gamesystem Hero
@description Basic Hero System Copy Filter
@foreach (Power)
%@item%
@if ConfigParam("PrintOptions")
@options(; )
%@option%\
@endoptions

@endif
@endfor
```

This would print each Power, followed by the options separated by semicolons.

Setting Configuration Parameters Programmatically

The SetConfigParam (p. 178) function sets a configuration parameter from within a script or character sheet. The parameter value is set for the current user for the current game system.

Predefined Configuration Parameters

The following predefined configuration parameters control internal operations of **GURPS Character Builder**. If you define them in your .pref files, the user can change the settings for that game system. The parameter must be present for every game system that needs to use them.

PrintInheritedOptions

If this parameter is 0, options inherited from a parent list will not be printed for members of that list. They will be printed for the parent list. This is useful for producing more compact output.

UseAliases

If this parameter is non-zero, the aliases on items and options will be used when print and copy filters are used. Depending on the game system, this allows for abbreviations or alternate names for items when they're being printed.

UseMetric

This parameter reflects the setting of the Measure setting in the Preferences Dialog (p. 15). It is non-zero if Metric is indicated, and zero if English is indicated. This parameter is independent of game system.

Creating Name Files

To create your own name files, you must first create a name list text file. This must be a standard text file, saved in the Windows character set. To create such a file, select the **File | New...** command, click the **Text File** button and click **OK**.

The input name text file consists of two parts. The first part is introduced by a line with the word “@notes” on it. All text following that line until another line beginning with a “@” directive will be included in the notes for the name file.

To specify the attributes that will be listed in the primary attribute list, place a @primary command before the @names list:

```
@primary Male,Female
```

Comments are indicated by beginning the line with the # character. Blank lines may occur anywhere within the name list, and are ignored.

Each name in the @names part must be on a separate line. The format for each line is

```
name,attribute[,attribute]...[:notes]
```

For example:

```
Aaron,Male,Hebrew:light, enlightened, brother of Moses
```

declares the name “Aaron” to be a male, Hebrew name with the meaning “light, enlightened, brother of Moses.” The name and the attributes should be 20 characters or less.

You can declare attributes to be used for all following names with the @attr directive:

```
@attr Female,Russian
```

indicates that all following names are Female Russian names.

To “compile” the text file, run the following command line in DOS:

```
compname names.txt names.nam
```

where names.txt is the input text file and names.nam is the output name file. The name file should always have the .nam extension. The compname program will be in the source directory if you installed the name files.

Part of the Anglo-Saxon name file (p. 249) is included in this help file. You can copy it to the clipboard and paste it into a text file as created above, then save it to a file and compile it to produce a .nam file.

Name File Example

Copy from the line starting with @notes.

```
@notes
```

```
These names are Anglo-Saxon and Danish names from the period 800-1000 AD.
They were taken from the Anglo-Saxon chronicle and other sources from the
period.
```

```
Pronunciation Key
```

```
æ : 'a' in sad.
```

```
ð : 'th' in the.
```

```
þ : 'th' in thing.
```

```
cg: 'j' in just.
```

```
sc: 'sh' in ship.
```

```
h: at the end of a word, or after a consonant like 'l', as the 'ch' in Bach.
```

```
@primary Male,Female
```

```
@names
```

```
@attr Male,Anglo-Saxon
```

Æa scferð

Æ lfnoð

Æ lfred

Æ llan

Æ lmær

@attr Female,Anglo-Saxon

Æ lfgifu

Alhðryð

Eadburg

Picture Description File Format

Picture description files have three sections: [Synonyms], [Ignore] and [Files]. The picture description file can be edited as a normal text file with GURPS Character Builder. Descriptions can also be added through the Picture Search dialog (p. 51).

Comments can be included with the # character. Everything after the # is ignored. Blank lines are also ignored.

Picture description files have a .pdsc extension. When the Picture Search dialog opens, all .pdsc files in the Picture Library directory are read and their synonyms, ignored words and files defined for searching.

To install new picture description libraries, simply copy the description file and associated picture files into the picture library directory. Uniquely named files and subdirectories are recommended to avoid overwriting existing images.

When the application writes out a description file, all extant synonyms and ignored words are written out as well, along with the files. Files with only synonym and ignored word definitions may be created to separate those definitions from the file descriptions.

Synonyms

The synonym section describes words that are treated the same during a picture search. The root word is listed, followed by a colon and any synonyms for that word, separated by blanks. For example, the following entry causes the words "man" and "men" to be treated the same as "male", and also defines synonyms for "female" and "mage:"

```
[Synonyms]
male: man men
female: woman women
mage: sorcerer wizard enchanter
```

Case is ignored in all keywords.

Ignored Words

The [Ignore] section lists words that are ignored during picture search queries. Simply list the words to be ignored on lines after the section header, separated by blanks. For example,

```
[Ignore]
at and for or in with the a an on
```

Files

The [Files] section lists the picture file names and the keywords that describe them. The file names must be relative to the picture library directory. That is, if the file is named AntMan.jpg in the Supers subdirectory of the picture library directory, it should be specified as:

```
Supers\AntMan.jpg
```

The backslash character ("\") should be used as a directory separator.

Each file description is entered on a separate line. The file name (including any relative directory) is followed by a colon and then the list of blank-separated keywords. For example:

```
[Files]
Supers\Leopard.jpg: female super modern feline
Supers\AntMan.jpg: male super modern antennae cape
```

Case is ignored in all keywords.

Selection Rule File Format

The `.rules` files in the source directory define the selection rules.

Configuration Section

Each rule file has a Configuration section that indicates the game system for which these rules are defined. For example:

```
[Configuration]
GameSystem=GURPS
Rulesets=Space;Fantasy
```

`GameSystem` The name of the game system.

`Rulesets` This parameter is optional. It is a list of the rule sets that should be predefined. These rule set names may be used in the default values. The list is semicolon-delimited.

DefaultValues Section

A rule file may have an optional `[DefaultValues]` section. It provides default active values for a rule defined in another rules file for rulesets defined in the current rules file. You might use this when you want to override the default active value for a rule in the standard rules file.

Each line in the section gives the name of the rule, preceded by the category and a colon (if there is a category), followed by an equals (=). A list of ruleset names and active values follows, delimited by semicolons. These are composed of a ruleset name, followed by an equals, followed by "true" or "false".

For example, if the rule for suppressing fantasy languages is inactive by default, and you want to define a ruleset named "Space" that has that rule active by default, you would specify the following:

```
[DefaultValues]
Skills:SuppFantLang=Space=true
```

The rule `Skills:SuppFantLang` must be defined in a rules file for this game section for this to have any effect.

Rule Sections

The rest of the sections in the `.rules` file define the names of the rules. Each section name is the name of the rule.

```
[TL]
Type=Expression
Value=`TL`<=tl+&tl and integer(optvalue('TL Range'))<=tl+&tl
Description=Suppress items above current Tech Level
DuplicatesOnly=false
DefaultValue=true
Not=false
```

If the rule is defined for a category of items, the list name precedes the rule name, separated by a colon. For example:

```
[Equipment:SuppressFantasy]
Type=Suppress Sublist
Value=Fantasy/Medieval Equipment
Description=Suppress Fantasy Equipment
DuplicatesOnly=false
DefaultValue=false;Space=true
Not=false
```

Each rule has the following parameters:

`Type` The type of the rule. The following rule types are defined:

 Suppress Category: suppress items in this category

	Suppress Item: suppress named items
	Suppress Item Prefix: suppress items whose names begin the specified prefixes.
	Suppress Datasheet ID: suppress items with the specified data sheet IDs.
	Suppress Sublist: suppress items that are below the named sublists.
	Rules List: a list of rules to apply
	Expression: an expression rule.
	Requirement: if the item's requirements are satisfied, the rule is satisfied.
Value	The value of the rule. A semicolon-delimited list of items, category names, prefixes, data sheet IDs, sublist names, or rules, or an expression. For a requirement rule, the name of the list against which the item is applied.
Description	The description of the rule, which is displayed to the user in the rules list.
DuplicatesOnly	The value "true" if the rule applies only to duplicate items, otherwise "false".
DefaultValue	The default value for the rule. If "true", the rule is applied (turned on) by default. If "false", it is not applied. The default values for other rule sets are indicated by the rule set name, and equals sign and a true or false value.

By creating `.rules` files with appropriately set default values for specific rule sets, you can have **GURPS Character Builder** automatically suppress undesirable items for specific classes of character sheets whose rule sets you have assigned. See Selection Rule Sets (p. 59).

Script Debugger

The debugger is a script and filter development tool. It makes writing scripts and filters much easier because it shows you exactly what the script is doing. Without the debugger window you must change the script to add dialogs to show the values of variables.

The script debugger allows you to step through the lines of a command script or text filter one by one. While debugging a script or filter, you can examine the contents of variables, stop execution at specific lines or subroutines, step over subroutine calls, step into subroutines, and step out of subroutines.

To open the debugger window, select the **Debugger** command on the **Window** menu. After you open the debugger window the debugger will stop execution on the first executable line of the next script that is executed.

If you no longer wish to debug a script, close the debugger window (click the close box) and the script will resume normal execution.

To ensure that all scripts are defined for the debugger, you should open the debugger window before any character sheets you wish to debug.

Script Source Window

The source text for scripts that are being debugged is displayed in the script source window. To enlarge the debugger window, click and drag the edges.

To enlarge the script source window and decrease the size of the context window, where the variables are displayed, move the cursor to the area between the two windows. The cursor will change to a left- and right-pointing arrow. Click and drag to indicate where the split between the two windows should be.

Controlling Execution

The current point of execution in the script is indicated by a green arrow (➤) to the left of a line number. The following buttons control execution of the script:



Go (F5). Resume execution of the script. Execution will continue until the next breakpoint is encountered. If there are no breakpoints, the script will run until completion.



Step in (F7). Step into the subroutine at the current line in the debugger. If there is no subroutine at the current line, the line is simply executed.



Step over (F8). Step over the current line. If a subroutine is called in the current line execution is continued after the call (unless a breakpoint is encountered elsewhere).



Step out (SHIFT+F7). Step out of the current subroutine, returning to the caller.

Stepping

If the debugger does not have control, the **Stepping** checkbox will be unchecked and the **Stack** drop-down list will be grayed out. To cause the debugger to stop the next time a script line is executed, click the **Stepping** checkbox. You may want to do this when a script is displaying a dialog box after you have clicked **Go** and you wish to regain control.


The **Stepping** checkbox becomes unchecked only when you execute a command that leaves stepping mode (Go, Step Out or Step Over).

Breakpoints

A breakpoint is a place in the script where execution is halted when that breakpoint is reached. There are three kinds of breakpoints: line breakpoints, subroutine breakpoints and script breakpoints.

Line Breakpoints

Execution stops when a line breakpoint is reached. Breakpoint lines are marked with a red circle (●). A line must be executable to set a breakpoint on it. When setting breakpoints on multiline commands (dialogs, etc.), set the breakpoint on the first line of the command.

To set a breakpoint on a line, double-click the line in the script window. You can also click the  button to toggle the breakpoint on the line highlighted in the script source window.

The line numbers of line breakpoints are remembered across invocations of the debugger, but if lines are added or removed from a script, the breakpoints may not stay with the exact same lines.

Subroutine Breakpoints

When you set a breakpoint on a subroutine name, execution stops when that subroutine is called.

To set a subroutine breakpoint, click the **Breakpoints** (p. 256) button. A subroutine breakpoint is also added when you set a breakpoint on a subroutine.

Subroutine breakpoints are remembered across invocations of the debugger. Since they are not line-based, they remain valid if the script is changed.

Script Breakpoints

When you set a breakpoint on a script, execution stops the first time control is transferred to that script, and every time a subroutine is called in that script.

To set a script breakpoint, click the **Breakpoints** (p. 256) button.

Script breakpoints are remembered across invocations of the debugger. Since they are not line-based, they remain valid if the script is changed.

Removing All Breakpoints

Click the **Remove All** button to remove all line, subroutine and script breakpoints.

Viewing Variables

There are two ways to view variable values: direct view and the context view.

Direct Variable View

Click a variable name in the script source window to see its value. A tip window with the variable name and value will be displayed. If you click a variable and it is not defined, no tip window will be displayed.

Click anywhere else in the source window to make the tip window disappear.

The variable will be searched for in the local context first, then the script context, then the original character sheet context and finally the destination character sheet context (if there is one).

Context View

The context view, on the upper right side of the debugger window, displays the variables defined in the script, the subroutine (Local) and the character sheet. When a conversion is being run there are Src and Dst contexts for those character sheets. Click the tab for the desired context.

The context view displays the actual variable values at all times.

Find

Find a text string in the script source window.

Next

Find the next occurrence of the last text string searched for in the script source window.

Goto

Go to a line number in the script source window.

Open

Open a file for viewing in the script source window.

Breakpoints

Set breakpoints (p. 256) on subroutines and scripts.

Rmv All

Remove all line, subroutine and script breakpoints.

Stack

The stack drop-down list displays the name of the currently active subroutine and script. Selecting an entry in the list displays the line at which the last subroutine call was made.

This control is inactive when execution is not halted in the debugger.

Script

A list of the scripts that have been loaded since the debugger window was opened. To view a script, select an entry in the **Script** drop-down list.

To clear the list of scripts, close the debugger window and then open it again.

Source Window Context Menu

Right-click the debugger source window to bring up the debugger command menu.

Toggle Breakpoint (F9)

Toggles the breakpoint at the highlighted line.

Breakpoints

Brings up the Breakpoints (p. 256) dialog, where breakpoints can be set on subroutine and script names.

Remove All Breakpoints

Removes all line, subroutine and script breakpoints.

Go (F5)

Resumes execution of the script.

Step Into (F7) Trace execution into a subroutine on the current line, if there is a subroutine.

Step Over (F8)

Execute the current line, without stepping into any subroutine on the line.

Step Out (SHIFT+F7)

Resume execution in the caller of the current subroutine.

Find (CTRL+F)

Find text in the current script.

Find Next (F3)

Find the text pattern set by the Find command

Goto Line Number (CTRL-G)

Go to line by number.

Open File

Open a file in the script source window.

Limitations

- You cannot step into character sheet or script functions (as opposed to subroutines).
- If multiple commands are grouped on the same line, the highlighted line will stay on the same line as each command is stepped through.
- If \$\$include directives are used in filters, the line numbers in the script source window will not conform to the line numbers in the original filters.
- The debugger window must be opened before a character sheet is opened so that the load script is defined for the debugger.

Breakpoints

The Breakpoint dialog allows you to set breakpoints on subroutines and scripts, and view and remove line breakpoints.

Whenever a subroutine that has a subroutine breakpoint is called, execution of the script is stopped in the debugger.

Whenever a call is made to a script that has a script breakpoint (either the script itself or a subroutine in the script), execution of the script is stopped in the debugger.

Subroutine

Enter the name of the subroutine to set the breakpoint on in the **Subroutine** field. Then click the **Set** button beside the field. The subroutine will be added to the list of subroutine breakpoints.

Set

Click **Set** to add the subroutine named in the **Subroutine** field to the list of subroutine breakpoints.

Remove

Click **Remove** to remove the subroutine highlighted in the subroutine breakpoint list.

Script

Enter the name of the script to set the breakpoint on in the **Script** field. Then click the **Set** button beside the field. The script will be added to the list of script breakpoints.

Set

Click **Set** to add the script named in the **Script** field to the list of script breakpoints.

Remove

Click **Remove** to remove the script highlighted in the script breakpoint list.

Line Breakpoints

All line breakpoints are listed here. To remove a line breakpoint, click the line and then click the **Remove** button below the line listbox.

Goto Line

Enter the number of the line you wish to view in the **Line Number** field. The script source window will attempt to display the specified line.

Find Text

Enter the text you wish to find in the **Text** field. Click **Find** to begin the search for that text in the script source window. The search begins at the line following the line highlighted in the source window.

If the text is found, the line containing it is scrolled into view and highlighted. If the text is not found on any other line in the script, a message will be displayed.

Convert Character Sheet

The **File | Convert...** command converts character sheets from one level to the next. Conversion is required when substantial enhancements are made to the character sheet template. Character sheets prepared with previous versions of the template must be updated to the current version to take advantage of these new features.

To convert a character sheet prepared with an older version of a character sheet template to the newer version:

- Select the current template. **GURPS Character Builder** will automatically select the best choice for this. If you're not sure what you should select, use the automatic choices.
- Select (None) for the Conversion Script, or select a script if needed to convert between levels of the character sheet template. As above, **GURPS Character Builder** will automatically select the best match.
- Click **OK**. **GURPS Character Builder** will recreate the character using the selected template. If you have different character sheet templates for the same game system you can convert back and forth the same way.
- To convert a character sheet from one game system to another:
 - Select the character sheet template of the new character from the list of New Character Templates.
 - Select the Conversion Script. Again, **GURPS Character Builder** will select the best match automatically, so if you're not sure what to do, use the automatic selection.
 - Click **OK** and the character sheet will be converted to the new character sheet type. The selected conversion script must be appropriate for the original and new character sheet templates.

New Character Template

This list displays the character sheet templates for the game system specified in the Game System drop-down list. The text box beneath the list gives a description (if available) of the highlighted template.

Conversion Script

This list displays the conversion scripts for the game system of the character sheet you are converting. The text box beneath the list gives a description (if available) of the highlighted script.

If no conversion script is selected, all items in the character sheet are converted to the equivalent items in the currently loaded data sheet. Items are sought by their internal (original) names first, then by the displayed name.

Game System

This drop-down list controls the display of the character sheet templates in the New Character Template list. By default it is "(All)", but you can change it to a specific game system and only those templates will be listed.

Other Options

Converting Multiple Files at Once (p. 17)

Writing Conversion Scripts (p. 262)

Converting Character Sheets from Other Applications (p. 294)

Conversion Details (p. 259)

Conversion Details

When a character sheet is converted to another character sheet from the same game system, a blank character sheet created from the specified character sheet template is opened.

When a script is specified, only those changes specified in the script are applied to target character sheet.

When no script is indicated, the values in all dialogs are copied from the source to the destination character sheet. Then items are added to the destination character sheet by going through the item lists in the original character sheet, getting the original names of those items, and adding the items in the current data sheet that have those same original names. The item levels and option values are set to the values of the corresponding item in the original character sheet.

Copied Information

Some information from the original character sheet is copied to the new character sheet:

- The notes for the character sheet.
- The description of the character sheet.

In addition, when the source and destination character sheets are of the same game system, the following information is copied from the original character sheet:

- The date created.
- The print template.
- The copy filter.
- The character generation template.
- The character generation script.
- The sheet type is copied if the new template's sheet type is not set. If the new template's sheet type is set, the converted character has the new sheet type.
- The list of data sheets that is loaded for the character sheet.

Command Scripts

Command scripts are very similar to conversion scripts (p. 262), except that they operate on only a single character sheet. They can be used to automate character creation tasks, randomly select attributes and skills, etc.

Including Command Scripts in the Data Menu

When **GURPS Character Builder** starts up, it looks in the directory where it is installed for all files that end in `.scp`. If these are properly formatted conversion scripts with identical "to" and "from" game systems, they are included in the Data menu for all character sheets in that game system.

For example,

```
# Script for randomly generating characters.

"Conversion script v. 1"
gamesystem "SRPS" "SRPS"
title "&Generate Random Character..."
```

would cause the command **Generate Random Character...** to appear on the **Data** menu for all SRPS character sheets. The `&` symbol is a flag to Windows to underline the (p. 262)following character. If you wish to include the `&` character in the command, double it (for example, "&Generate Powers && Skills").

The `submenu` (p. 291) and `menuitem` (p. 291) commands can be used to change the name of the command on the menu, and to create cascading menus.

Generate Character

You can use the information in one character sheet to generate another character sheet. To do this, you need to write a conversion script (p. 262). You also need to set the Genchar Template and the Genchar Script in the Character Sheet Info (p. 45) dialog. This command will remain inactive until those fields are set.

This command will create a new character sheet, using the Genchar Template as a basis. The new character sheet will be modified as per the instructions in the Genchar Script. You can use this to create characters that require multiple steps, many different randomly selected items, and so forth.

Conversion Script Format

A conversion script is a list of instructions for taking the contents of one character sheet and, based on that, deciding what values to set and which items to select for the new character sheet.

Writing a conversion script is very similar to programming, so it is not for the faint-hearted. Brief instructions follow.

Conversion scripts must be standard text files (they must not be saved as documents from a word processor). The extension is .CNV. The first real line (excluding comments) of the script must be the string "Conversion script v. 1", which identifies the file as a conversion script.

The second line must be `gamesystem` command, which has as its first argument the game system of the original (source) character sheet. Its second argument must be the game system of the new (destination) game system.

A `#` character indicates a comment that continues to the end of the current line.

For example, the following is a typical header:

```
# Conversion script for Game system A to Game system B.
"Conversion script v. 1"
gamesystem "Game System A" "Game System B" # src dst
convlevels 3 5;
title "Convert A to B"
copyright "Copyright © 2001 ABC Corp."
...
```

If used in the file, the `title`, `menuitem` and `submenu` commands must appear immediately after the `gamesystem` (p. 279) command. The `convlevels` (p. 269) command is a helper command that indicates what the intended level for the source character sheet and the level for the destination character sheet template.

Topics

- Variables (p. 293)
- Expressions (p. 263)
- Comparing Character Sheet Totals (p. 263)

Commands

Assignment (p. 265)	<code>exec</code> (p. 276)	<code>optqualifier</code> (p. 286)
<code>add</code> (p. 266)	<code>exit</code> (p. 276)	<code>pauseProgress</code> (p. 286)
<code>addautoitems</code> (p. 267)	<code>fail</code> (p. 277)	<code>purgedatasheet</code> (p. 286)
<code>addoption</code> (p. 267)	<code>for</code> (p. 277)	<code>purgedatasheetitems</code> (p. 286)
<code>appendProgress</code> (p. 267)	<code>foreach</code> (p. 277)	<code>return</code> (p. 286)
<code>array</code> (p. 268)	<code>func</code> (p. 278)	<code>selectitem</code> (p. 287)
<code>checkallreq</code> (p. 268)	<code>gamesystem</code> (p. 279)	<code>scriptmode</code> (p. 286)
<code>checkreq</code> (p. 268)	<code>getfilename</code> (p. 279)	<code>set</code> (p. 287)
<code>closeProgress</code> (p. 268)	<code>getitem</code> (p. 279)	<code>showallreq</code> (p. 289)
<code>closesublist</code> (p. 284)	<code>if</code> (p. 280)	<code>showProgress</code> (p. 288)
<code>convlevels</code> (p. 269)	<code>include</code> (p. 281)	<code>showreq</code> (p. 289)
<code>copyall</code> (p. 269)	<code>insertitem</code> (p. 281)	<code>skip</code> (p. 289)
<code>copydefines</code> (p. 269)	<code>item</code> (p. 281)	<code>sortlist</code> (p. 289)
<code>copydialog</code> (p. 270)	<code>keepnext</code> (p. 275)	<code>sub</code> (p. 290)
<code>copyitem</code> (p. 270)	<code>list</code> (p. 282)	<code>submenu</code> (p. 291)
<code>copyleft</code> (p. 270)	<code>matchbox</code> (p. 274)	<code>switch</code> (p. 292)
<code>copyoption</code> (p. 270)	<code>matchlist</code> (p. 274)	<code>title</code> (p. 293)
<code>datasheet</code> (p. 270)	<code>menucommand</code> (p. 284)	<code>unconverted</code> (p. 283)
<code>deleteitem</code> (p. 271)	<code>menuitem</code> (p. 291)	<code>var</code> (p. 293)
	<code>notes</code> (p. 284)	<code>warn</code> (p. 293)

dialog (p. 271)	opensublist (p. 284)	while (p. 293)
dialogsize (p. 275)	openwindow (p. 285)	
dieroller (p. 275)	option (p. 285)	
edititem (p. 276)	options (p. 285)	
exclude (p. 276)		

Comparing Character Sheet Totals

The Update Character Sheets command can compare character sheet totals after the conversion. This is implemented in the script by setting the variable "identical" in the conversion script context. If the two character sheets are identical, you must assign a non-zero value to the variable "identical". If they are different, you must assign a value of zero. If you don't set the value, the character sheets are assumed to be identical.

For example, the following assignment compares the totals of a Fuzion character:

```
identical = $@t_powers = $*t_powers and
    $@t_complications = $*t_complications and
    $@t_chars = $*t_chars and $@t_options = $*t_options;
```

Note that if one item in a list cost a point more in the converted list, and one item costs a point less, the totals will be the same and the character sheets will appear identical even though they really aren't.

Conversion Expressions

Conversion expressions are the same as normal Creator expressions (p. 169), except that you can reference variables (p. 293) in the original character sheet, the converted character and in the context of the conversion.

Within the arguments following assignments and add commands, you can reference variables in the conversion context by preceding them with "\$\$".

There are some special values that can be used in conversions:

Item Already Present in List

The \$+inlist function returns 1 if an item is in the specified list in the target data sheet, and 0 if it is not:

```
$+inlist(listName, itemName)
```

Item Present in Source List

The \$+inSourceList function returns 1 if an item is in the specified list in the source character sheet, and 0 if it is not:

```
$+inSourceList(listName, itemName)
```

Random Item Selection

To select an item at random from the category in the loaded data sheets associated with a named list use the following:

```
$+randselect(list.sublist)
```

One of the items in the data sheet list is selected at random. If a sublist is specified, the search is narrowed to that sublist. If a sublist is selected at random, then the search is continued in that sublist.

In this example, a random number of skills from the Combat sublist is selected. Each time through the loop another skill is selected at random. If it's already in the list, it is not added. To prevent this from looping forever, you must ensure that you have enough skills in the list you're selecting from.

```
nskills = rand(6)+rand(6);
for (i = 0; i < nskills; )
    skill = "$+randselect(Skills.Combat)";
    if (!$+inlist(Skills, $$skill))
        add Skills.$$skill;
```

```

        i = i + 1;
    endif
endfor

```

A more reliable method is to obtain all the items in the Combat category using the `foreach(*Skills : Combat)` command (the Combat category must be specified with each such item), add their names to an array with `addElement` and then draw items from that array at random, removing them with `removeElement` as they are added, terminating the loop if the array is depleted.

```

array skills[0];
foreach(*Skills : Combat)
    if (!$+inList(Skills, $n))
        n = addElement(skills, @name);
    endif
endforeach
nskills = rand(6) + rand(6);
i = 0;
while (i < nskills and sizeof(skills) > 0)
    n = rand(sizeof(skills));
    skill = skills[n];
    add Skills.$$skill;
    n = removeElement(skills, n);
    i = i + 1;
endwhile

```

Item Available for Selection

The `$+available` function returns true if the named item exists in one of the data sheets for the converted character sheet.

```
$+available(listName, itemName)
```

You might use it in the following fashion:

```

list "Skills"
destination "Skills";
default ~
    if ($+available(Skills, $o))
        add $o[$n] = convert($c);
    else
        copyitem "Skills";
    endif
end
endlist

```

This script might be used to convert from one version of a character sheet to another, when the definition of an item changes. If an item is available in the data sheet, the item is added, doing some conversion on the cost, otherwise the item is copied to the destination verbatim.

Index of Selected Item

The `$+selectedIndex` function returns the index (starting from one) of the currently selected item in the listbox of the specified list window.

```
$+selectedIndex(listName)
```

If the window is not open, or there is no selected item, `$+selectedIndex` will return -1.

This is useful for working with the currently selected item in the list with the `item` (p. 281) command.

Count of Items

The `$+listCount` function returns the number of items at the highest level of the specified list.


```
$+listCount(listName)
```

A sublist is counted as one item. If there are no items in the list, zero is returned.

Current Window Name

The `$+currentWindow()` function returns the name of the currently active window. There are no arguments. If you are assigning this to a variable, you should place it inside a string:

```
window = "$+currentWindow()";
```

Current Field Name

The `$+currentField()` function returns the name of the currently active field, if the current window is a dialog. If you are assigning this to a variable, you should place the reference inside a string:

```
fieldName = "$+currentField()";
```

Current Date

The `$+date()` function returns the name of the current date, formatted according to the short date format defined in Windows. If you are assigning this to a variable, you should place the reference inside a string:

```
today = "$+date()";
```

Check Requirements Flag

The `$$$checkRequirements` variable is non-zero if the Check Requirements checkbox is checked in the Available Items dialog. It is non-zero if requirements checking is turned off.

User Canceled Script

The `$+cancel` flag is non-zero if the user has clicked the **Cancel** button in the progress dialog (p. 288). This flag is reset to false (zero) whenever `closeProgress` (p. 268) or `dialog` is executed.

Tick Count

The `$+tickCount()` function returns the "tick count" for the system clock. To perform timing on scripts, save the tick count in a variable where you want to begin timing in the script, then subtract that variable from the tick count where you want to end timing. The difference is the number of milliseconds that elapsed.

Assignment

Set the value of a field in a dialog, or of a variable in the conversion context. It can also be used similarly to the `add` command, to set the value of an existing item in a list to a desired value.

```
Main.st = $*str;  
Attributes.Strength = skills[$1];  
st = rand(6)+rand(6)+rand(6);
```

The dialog or list name is separated from the field or item name by a period. The assignment must be terminated by a semicolon (;).

To set fields in a dialog box to the values of fields in a dialog box from the original character sheet, precede the original dialog box name with `$$`. For example,

```
Picture.Picture = $$Description.Picture;
```

If you assign a string to a Picture field, ***GURPS Character Builder*** will attempt to find a file by that name and read it into the picture.

The text values of dialog fields may be assigned to variables in the same fashion:

```
history = $$Information.History;
```

To define (or assign) variables in the destination character sheet, prefix the variable name with `$@`. For example,

```
$@i_strength = $*i_strength;
```

would create a new variable in the define list for the destination character sheet (if it didn't already exist). The variable will also be marked as "script added." If the variable already exists in the define list, it will be marked as script-added. The variable name may not already be defined for a dialog item or list total variable.

When character sheets are converted without a script, the values of script-added variables are retained in the new character sheet. When a conversion script is used, it is up to the script to deal with any script-added variables.

Remember that variables can have bonuses associated with them. If you set the value of a variable on a character sheet that has items that use adjustments to put bonuses and penalties on a variable, the assignment will set the base value, but the bonus will be unchanged. The only way to eliminate the bonus is to remove the items that adjust it.

You may also specify options to be added to an item with an `option` (p. 285) command.

add

The `add` command is used to add items to lists in the converted character sheet. The list and name of the item to add immediately follow the `add` command. The list name is separated from the item name by a period (.). The list name may be abbreviated. The item name must be specified in full, and must be a reference to an item in the selection list in the data sheets for the specified list.

If the item to be added can have a level, an assignment and an expression may follow the name. The expression can refer to variables (p. 293) in the original character sheet, the destination character sheet or the conversion context. For example,

```
add Skills.Survival = skills[$1];
```

If any spaces appear in the item name, it and the list name must be enclosed in quotes. For example:

```
add Skills.Area Knowledge = 9+round($int/5);
```

adds the skill Area Knowledge with a value of 9 plus the value of the variable "int" in the destination character sheet divided by 2.

If the item name is followed by [] enclosing another name, the item name is looked up in the destination character sheet's data sheet by that first name, and then the actual item that is added to the new character sheet is a copy of that item, renamed with the name in the [] brackets. For example,

```
add Skills.Generic Skill[$n] = sk($1);
```

looks up the skill "Generic Skill" but names it with the same name as the item in the original character sheet.

If the item name contains reserved words (END, for example), you must surround the name with double-quotes to prevent the script processor from becoming confused.

If an item can't be found with the specified name, the "original name" field of the items in the data sheet will be searched against the specified name. In some cases the displayed name and the original name are not the same, usually when the game system defines items with identical names but different definitions. This allows the identity of the item to be maintained when the character sheet is converted.

You can reference variables defined in the conversion context by preceding them with "\$\$". For example, the following lines define four benefits in an array and then randomly select one to add to the Benefits list:

```
array bennies[4] = 'Beautiful', 'Common Sense',  
    'Danger Sense', 'Toughness';  
benefit = bennies[rand(sizeof(bennies))];  
add Benefits.$$benefit;
```

You may also specify options to be added to the item at the same time, by following the `add` command with an `option` (p. 285) command.

The options of the most recently added item can be referenced with the `$@@option` keyword, where *option* is the name of the option. Such references can be made only in the context of `list` and other similar commands.

addautoitems

The `addautoitems` command adds automatic items to the last item added. If there is no last item an error is reported.

The syntax is:

```
addautoitems "autoItems", retVal,
    [ autochargecost | deleteauto | dontmarkauto | noautodup ] ...;
```

The *autoItems* value is an automatic items (p. 79) string, with each entry separated by a semicolon. *retVal* receives the return value -- non-zero if the command is successful and zero if the user canceled.

The optional arguments at the end of the command set flags used for adding the items:

Flag	Meaning
<code>autochargecost</code>	Charge the cost of automatic items.
<code>deleteauto</code>	Delete the automatic items when the parent item is deleted.
<code>dontmarkauto</code>	Add the automatic items as normal items, without indicating that they are automatic items.
<code>noautodup</code>	If the item is already present in the list, don't add it again.

For example, the following command allows the user to choose three skills from the Weapon category.

```
addAutoItems "?list:Skills;?cat:Skills;?choose:Choose Weapon Skills,3,*Weapon",
    retVal, noautodup, dontmarkauto, autochargecost;
```

addoption

The `addoption` command adds the named option to the end of the option list for the current item.

```
addoption DR = 7;
option Hobby;
option Text[Note] = Special Bonus;
```

It is otherwise identical to the `option` (p. 285) command.

See also: `optqualifier` (p. 286).

appendProgress

The `appendProgress` command adds text to the text field of the progress dialog (p. 288).

```
appendProgress Some text;
appendProgress "Current Age: $$age";
```

The text should be quoted if it contains any special characters (such as semicolons) or references to variables. Each time text is appended it will start on a new line.

If the progress dialog was previously opened by a `showProgress` (p. 288) command the title bar of the dialog will remain what `showProgress` set it to. If there was no previous `showProgress` the title bar will be "Progress Dialog."

See also: `showProgress` (p. 288), `pauseProgress` (p. 286), `closeProgress` (p. 268).

array

The `array` command is used to define an array to the conversion context (see variables (p. 293)). After `array` comes the name of the array, followed by `[]` brackets surrounding the number of elements in the array. The number of elements may be omitted if the array is initialized to a list of values. A '=' precedes the list of array elements, separated by commas, and the list must be terminated by a semicolon.

```
array skills[5] = 10,12,14,16,18;
array optArray[0];
array rogueSkills[] = "Shadow", "Stealth",
    "Lockpicking", "Pickpocket";
```

Arrays are indexed from 1. If the index is less than 1, the first value in the array is returned. If the index is greater than the number of items in the array, the last value is returned.

An array may be defined as zero-length. The **addElement** (p. 171) predefined function can be used to add entries to the array. This is useful for building lists of items that the user chooses from in the `dialog` (p. 271) command.

checkallreq

The `checkallreq` command checks the requirements for all items in the character sheet.

```
checkallreq ok;
```

The variable `ok` is set to non-zero if the requirements for all items have been satisfied, and 0 if not.

See also `showallreq` (p. 289).

checkreq

The `checkreq` command has two forms. The single-argument form checks the requirements of the most recently added item, or the current item of the `item`, `foreach`, or `list` commands. The three-argument form checks the requirements for an item in the available items list. The four-argument form checks the requirements specified in the string for the item in the specified list.

```
checkreq ok;
checkreq list, item, variable;
checkreq list, item, reqstring, variable;
checkreq Skills, Shield, ok;
checkreq Equipment, Magic Sword, "#1,Classes:Spellcaster>=12", ok;
```

The variable `ok` is set to non-zero if the requirements of the item have been satisfied, and 0 if they have not been satisfied.

The four-argument version is useful for checking requirements for working with items in scripts. For example, magic items can have one set of requirements when they're added to a character's equipment list, and another set when the magic item is created by a wizard. In that case, the requirements might be contained in an option that is read from the option when the script is run that creates the item. See data sheet requirements (p. 220) for details on formatting the requirements string. For example, the four-argument example above checks to see that at least one entry in category Spellcaster exists in the Classes list with a level greater than or equal to 12 -- that is, that the character is a level 12 spellcaster.

See also `showreq` (p. 289).

closeprogress

The `closeProgress` command closes the progress dialog if it is open, also resetting the `+$canceled` (p. 265) flag to false.

```
closeProgress;
```

See also: `showProgress` (p. 288), `pauseProgress` (p. 286), `appendProgress` (p. 267).

context

The `context` command allows direct access to the variable context of the destination or source character sheet.

```
context(destination)
# ...
endcontext

or

context(source)
# ...
endcontext
```

If `destination` is specified, the destination character sheet's variables may be referenced directly within the context, without having to use the "\$*" decoration to access them. If `source` is specified, any variable references will be to the source character sheet's variables.

Any assignments will be made directly to the character sheet's variables. All references to functions such as `listiteminfo` will function correctly.

Care must be taken with `context` to ensure that you don't change any character sheet variables unintentionally.

convlevels

The `convlevels` command indicates the level of the source character sheet and the destination character sheet for conversion scripts. This is not required command, but its inclusion at the beginning of the file (after `gamesystem`) allows **GURPS Character Builder** to highlight the appropriate conversion script for a specific character sheet.

```
convlevels 4 5;
```

The first number is the level of the source character sheet, while the second number is the level of the destination character sheet template.

copyall

The `copyall` command copies all information in one character sheet to another.

```
copyall;
```

This command is most useful for converting character sheets that need only minimal adjustments during conversion. It is the same as executing `copydialog` and `copyleft` for all the dialogs and lists in the character sheet, plus it also copies other character sheet information that is done during unscripted conversions.

copydefines

The `copydefines` command copies all script-added variables defined in the source character sheet to the destination character sheet.

```
copydefines;
```

You should use this command when writing conversion scripts that convert a character sheet from one level to another, if you assign any character sheet variables in your command scripts (p. 265). That is, you make assignments of the form:

```
$@variable = value;
```

The `copydefines` command can be used only when converting from one character sheet to another.

copydialog

The `copydialog` command copies a dialog from the old character sheet to a dialog in a new character sheet. The first argument is the dialog in the source (old) character sheet, and the second argument is the dialog in the destination (new) character sheet.

```
copydialog "Main" "Main";
```

The new dialog should have every field that the old dialog has (it may have more). If this isn't the case, you'll have to copy the dialog field by field, using the following syntax:

```
Main.name = $$Main.name;  
Main.int = $$Main.int;  
# etc.
```

This is useful for converting from one version of a character sheet to the next, when dialog names, list names or field names have changed.

copyitem

```
copyitem "List";
```

The `copyitem` command copies the current item verbatim to the specified list in the destination character sheet. This command is intended for use in converting a version of a character sheet to another version for the same game system. It can only be used within the command list for a matching item in a `list` command.

copylist

The `copylist` command copies all items from one list to another, selecting the current version of the item from the data sheet. The first argument is the list in the source (old) character sheet, and the second argument is the list in the destination (new) character sheet.

```
copylist "Skills" "Skills";
```

This is useful for converting from one version of a character sheet to the next, when dialog names, list names or field names have changed.

copynotes

```
copynotes;
```

The `copynotes` command copies the notes from the current item to the last item added to the destination character sheet.

copyoption

```
copyoption;
```

The `copyoption` command copies the current option verbatim to the last item added to the destination character sheet. This command is intended for use in converting a version of a character sheet to another version for the same game system. It can only be used within the command list of an `options` command.

datasheet

Used to define the data sheets that will be included with the character sheet (in addition to any that the character sheet template defines).

```
datasheet datasht1.cds, datasht2.cds;  
datasheet @Magic.dlist;
```

List any data sheet file names after the `datasheet` keyword, separated by commas. The list must be terminated by a semicolon. Those data sheets will be added to the ones that are automatically opened when the converted character sheet is opened.

If the file name is preceded by "@" the file is assumed to be a text file containing the names of data sheets that will be loaded.

deleteitem

Deletes the last-added or current item (the last item added is tried first). This is the item just added, or the item in an `item ... enditem` construct, or the current item in a `foreach`.

```
deleteitem;
```

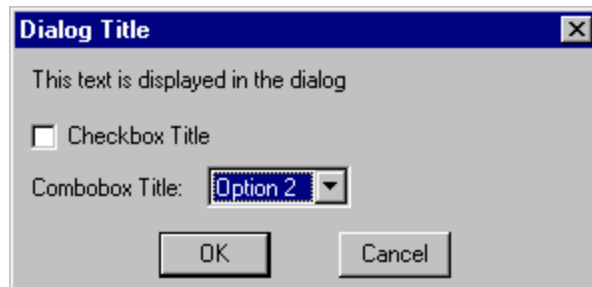
If you delete an item that has automatically added other items, it is possible to "lose" the current item with `deleteitem`, because the next item will also be deleted along with the parent item. In this case the `foreach` may terminate early.

dialog

The dialog command solicits input from the user. It has the general format:

```
dialog ok, Dialog Title;
  text "This text is displayed in the dialog";
  checkbox check1, "Checkbox Title";
  combobox comb1, "Combobox Title:", Option 1, 1, Option 2, 2,
    Option 3, 3;
end
```

When the above example is executed, the following dialog is displayed.



If the progress dialog (p. 288) is open, execution of a dialog command will close it.

dialog

The variable named after the dialog keyword is set to true (non-zero) if the user clicks the **OK** button, and false (zero) if the user clicks the **Cancel** button. If `button` commands (p. 274) are specified the variable is set to the value of the clicked button. The text after the variable (terminated by a semicolon) is displayed as the title of the dialog box. Generally, you should exit (p. 276) the script if the user cancels.

The following controls are available within the dialog:

text

Displays the text following the `text` keyword. This must be terminated by a semicolon.

checkbox

Displays a checkbox, whose initial value is checked if the variable named in the first argument is 1, and unchecked if the variable is 0. This variable will reflect the final state of the checkbox after the user clicks **OK**. The label for the checkbox follows the variable name and is terminated with a semicolon.

combobox

Displays a combo-box drop-down list. Following the `combobox` keyword is the variable that will be set with the value for the combobox. Following the variable is the title for the combo box. Following the title are

(text, number) pairs. The text values are displayed in the drop-down list. The combobox variable will be set to the number corresponding to the text value that is selected when the user clicks the **OK** button. The last (text, number) pair must be terminated by a semicolon.

number

```
number variable, Prompt;;
```

Displays a text-edit window, whose initial value is the value of the variable, and a prompt. The variable is set to the value of the text-edit window when the **OK** button is clicked, and can be used for computations in the conversion script.

dropdown

```
dropdown variable, Prompt:, string1, $$var, $$arr, ...;
```

Displays a drop-down list. The user can select a string from the list. It is different from `combobox` because it returns the selected string rather than a corresponding number.

<code>variable</code>	The variable name into which the string value selected for the <code>dropdown</code> will be placed. The initially selected value will be set to the value of the variable. When the OK button is clicked <code>variable</code> will be set to the string value of the currently highlighted entry in the list.
<code>Prompt:</code>	The prompt displayed for the user.
<code>string1</code>	The list of strings that are displayed in the list.
<code>\$\$var</code>	Reference to a variable. The value of the variable is added to the drop-down list.
<code>\$\$arr</code>	Reference to an array. All the elements in the array are added to the drop-down list.

listbox

```
listbox variable, Prompt:, string1, $$var, $$arr, ...;
```

Displays a listbox. The user can select a string from the listbox.

<code>variable</code>	The variable name into which the string value selected for the <code>listbox</code> will be placed. The initially selected value will be set to the value of <code>variable</code> . When the OK button is clicked <code>variable</code> will be set to the string value of the currently highlighted entry in the list. If an entry in the list is double-clicked the effect is the same as single-clicking the entry and then clicking the OK button. If <code>variable</code> is an array, the control allows multiple selection. All highlighted entries in the listbox will be added to the array when the dialog box is closed.
<code>Prompt:</code>	The prompt displayed for the user.
<code>string1</code>	The list of strings that are displayed in the listbox.
<code>\$\$var</code>	Reference to a variable. The value of the variable is added to the drop-down list.
<code>\$\$arr</code>	Reference to an array. All the elements in the array are added to the listbox.

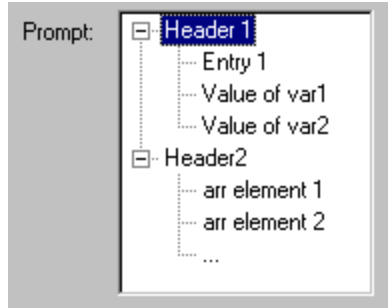
Tabs in the text of an item added to the listbox will be expanded to the standard Windows tab width.

treelist

```
treelist variable, Prompt:, Header 1, {, Entry 1, $$var1, $$var2, },  
Header2, {, $$arr, ..., };
```

Displays a tree view list box. The user can select a string from the tree list. Nodes in the tree list can be normal entries or sublists (that is, contain other nodes beneath them). To indicate a sublist, enclose the sublist entries with `{` and `}`. The braces *must* be separated from the other entries by commas.

The above command will produce the following control (with the sublist nodes fully expanded):



The sample argument values have the following interpretation:

<code>variable</code>	The variable name into which the string value selected for the <code>treelist</code> will be placed. The initially selected value will be set to the value of <code>variable</code> . When the OK button is clicked <code>variable</code> will be set to the string value of the currently highlighted entry in the list. If a sublist node is double-clicked, it is either opened or closed. If a normal entry is double-clicked the effect is the same as single-clicking the entry and then clicking the OK button.
<code>Prompt:</code>	The prompt displayed for the user.
<code>Header 1</code>	A top level node in the tree list that contains a sublist. It will have a + button to indicate that it can be opened. You can double-click the node name, or single-click the + button to open the sublist list and display the nodes within it. A header node is followed by a { to indicate that the following entries are nested within a sublist. A } indicates the end of the sublist. Sublists may be nested.
<code>Entry 1</code>	The list of strings that are displayed in the tree list.
<code>\$\$var1, \$\$var2</code>	References to variables. The value of the variables is added to the drop-down list.
<code>Header 2</code>	A top-level node in the tree list, similar to Header 1.
<code>\$\$arr</code>	Reference to an array. All the elements in the array are added to the tree list, indented one level under Header 2 because the <code>\$\$arr</code> reference is surrounded by { }.

editcombo

```
editcombo variable, Prompt:, string1, $$var, $$arr, ...;
```

Displays a drop-down list with an editable text box. It is similar to `dropdown`, with the addition of being able to edit the text.

<code>variable</code>	The variable name into which the string value selected for the <code>editcombo</code> will be placed. The initially selected value will be set to the value of the variable. When the OK button is clicked <code>variable</code> will be set to the string value of the edit box portion of the combination edit/list.
<code>Prompt:</code>	The prompt displayed for the user.
<code>string1, ...</code>	The list of strings that are displayed in the list.
<code>\$\$var</code>	Reference to a variable. The value of the variable is added to the drop-down list.
<code>\$\$arr</code>	Reference to an array. All the elements in the array are added to the drop-down list.

edittext

```
editcombo variable, Prompt;;
```

Displays a text-edit window, whose initial value is the value of the variable, and a prompt. The variable is set to the value of the text-edit window when the **OK** button is clicked.

button, defbutton

```
button expression, Label;
button "Help", Help;
defbutton 1, Next;
button 0, Cancel;
```

Displays a button with the specified label. The dialog variable is set to the evaluation of *expression*. The *defbutton* command indicates the default button, which has a dark border around it. The default button should return the value 1, and should be treated as the **OK** button, though it may have any name. The button that returns 0 will be treated as the **Cancel** button, though it too may have any name. The Help button should return the value "Help", though it may again have any name.

Consecutive *button* commands are placed on the same row and, if there is enough room, they are resized to the size of the widest button.

matchbox

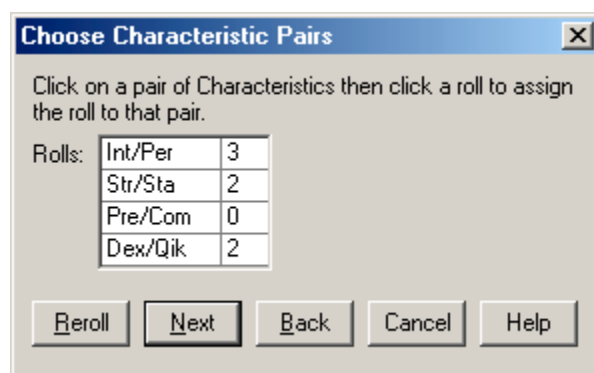
```
matchbox array1, array2, Choices::;
```

Displays a matchbox control. This allows the user to match up two sets of values, which are displayed side by side. The items in *array1* are displayed on the left, while the items in *array2* are displayed on the right. The number of entries in the arrays must be identical.

The user indicates a match by clicking an item on the left, then clicking the item that should match it on the right. The item clicked on the right then switches positions with the current value associated with the item on the left. The choice on the right can be clicked first, then the selection made by clicking an item on the right.

When the dialog exits, the elements of *array2* will be reordered according to the choices made.

For example:



matchlist

```
matchlist Skills:, $$skillList;
matchbox skills, levels, Selected::;
```

The *matchlist* and *matchbox* controls work together. The user selects items from the *matchlist* control, which places them in the *matchbox* control. The user then associates the selected items with values in the *matchbox*.

To select an item to be moved to the *matchbox*, click the item in the *matchlist*. The item will be moved to the first empty location in the *matchbox* and removed from the *matchlist*. If the *matchbox* has no empty locations, first click a *matchbox* location to highlight it, then click an entry in the *matchlist*. The highlighted *matchbox* entry will be moved back to the *matchlist* and the selected *matchlist* entry will be moved to the *matchbox*.

Matchlist entries are alphabetized. The syntax for setting entries in matchlist is the same as listbox, except that there is no variable.

keyword

```
keyword "Generate Character";
```

Specifies the Help keyword for the dialog. If not specified, the dialog title is used as the help keyword in the game system help file when the user clicks the dialog's **Help** button.

dialogsize

```
dialogsize 400, 200;
```

Specifies the minimum size of the dialog. The first number is the width, in pixels, and the second number is the height. If 0 is specified for one of the dimensions, the size required to contain the controls is used. For example, to indicate that the dialog should be 400 pixels wide and as tall as necessary, specify

```
dialog 400, 0;
```

ctrlsize

```
ctrlsize 300, 40;
```

Specifies the size of the subsequent control in the dialog. This allows specifying multiline edit controls and controlling the size of listboxes. Controls cannot be reduced to less than a single line height.

keepnext

```
keepnext;
```

Keep the next two controls together on the same line. For example,

```
keepnext;
number labText, Lab Text Bonus;;
checkbox ownLabText, Own Lab Text;
```

displays the edit field and the text box on the same line. To put three controls on the same line, precede each of the first two controls with `keepnext` commands.

rdonly

```
rdonly;
```

Makes the subsequent `edittext` control read only:

```
rdonly;
ctrlsize 300, 60;
edittext info, Information;;
```

This is useful for displaying large amounts of textual information to the user in a scrollable control whose size can be limited. If a `text` control is used, it will make the dialog grow to an arbitrarily large size.

Die Roller

Rolls dice and displays the results in the Die Roller (p. 24) window.

```
dieroller "diespec" [ , result ];
```

The dice indicated in the die specification (p. 26) are rolled and the result is placed in *result* if present.

The Die Roller window will be opened and set to always be on top.

The **rolldice** (p. 178) function can be used to generate random numbers without opening the Die Roller window.

Examples

Display a value from 3-18 in the Die Roller window.

```
dieroller 3d6;
```

Draw five playing cards and place the result in the variable named result.

```
dieroller 5p52, result;
```

edititem

Allows the user to edit the last-added item.

```
edititem ok;
```

This brings up the appropriate item editing dialog for the item. If the user clicks the **OK** button, the value of `ok` will be true. If the user clicks the **Cancel** button, the value of `ok` will be false. The return variable may be omitted.

exclude

The `exclude` command excludes the cost of the most recently added item from the total for the list. For example,

```
add Sublist;
if (@excluded)
    exclude;
endif
```

exec

```
exec file.scp;
```

Transfer control to the named conversion script. This "jumps out" to the other script and does not return to the script containing the `exec` command.

If this is executed from a normal script, all subroutines, functions and variables defined in the initial script are retained. Performing an `exec` inside the body of a loop (`foreach`, `while`, etc.) terminates the loop.

If this is executed from a control command script, an item script or the character sheet load script, a new variable context and a new subroutine namespace are created to avoid changing the context of the character sheet load script. This means that variables, functions and subroutines from the load script are *not* available in the script loaded by `exec`, and cannot be modified by the `exec`'ed script. Character sheet variables are still accessible through `$@` references.

exit

When the `exit` command is encountered in the script, the script terminates immediately. This is most useful when a user cancels a dialog (p. 271). For example,

```
genStats = 1;
dialog ok, Generate Character;
checkbox genStats, Generate Stats;
combobox charType, Character Type:,
    Random, 0,
    Warrior, 1,
    Wizard, 2,
    Rogue, 3,
    Priest, 4;
end
if (!ok)
    exit;
endif
```

fail

To display an error message and stop the conversion:

```
fail Error message;
```

This command displays the error message indicated by the string and terminates the conversion. The message must be terminated with a semicolon.

for

The `for` construct is used to create loops. If you want to loop over a list of items, you need to use the `list` (p. 282) command. The `for` command has the following syntax:

```
for (i = 1; i <= limit; i = i + 1)
# ...
endfor
```

The first part of the `for` must be an assignment of a variable to some value, followed by a semicolon, then a test expression. As long as the expression is true, the `for` construct will loop. This part must also be followed by a semicolon. The final part of the `for` construct is the increment. This must be the loop variable, followed by an assignment (=), followed by the increment expression. This will usually be something like "i = i + 1" or "i = i - 2". You may omit the increment part, and increment the control variable within the body of the `for`.

Be aware that if you don't include the loop variable in your test expression, or change your loop variable in the increment part, your conversion script will loop forever. To get out of such "infinite loops" press the ESC key, or if things are really hung, CTRL+BREAK.

foreach

The `foreach` construct is used to loop through items in a source list, or through the available items in the data sheet corresponding to a list.

```
foreach(Skills)
# ...
endforeach

or

foreach(*Skills)
# ...
endforeach

or

foreach(*Skills : Category Name)
# ...
endforeach
```

This is very much like the `@foreach` (p. 194) filter command.

If item selection rules (p. 57) are active, items that do not satisfy the active rules will not be processed.

The body of the `foreach` (the commands between `foreach` and `endforeach`) is executed for each item in the named list. As with `@foreach`, you can reference information about the item with item references (p. 194) such as `@name`.

If a "*" is present before the list name, the *available items in the data sheet* for that list are iterated through instead of the items in the list itself. This is useful for getting at information about items (such as their names, costs, etc.). To examine only items from a specific category, place a colon (:) and a category name after the list name: "`foreach(*Skills : Combat)`". The items from the specified category will be iterated through in alphabetical order.

When repeatedly adding items to the same list that is the subject of a `foreach` you must be careful to avoid adding them after the current item of the `foreach`. Doing so can result in an "infinite loop".

The `foreach` command can be stopped by pressing the ESC or CTRL+BREAK key.

Example

The following script opens the Skills window and adds all available skills to the Skills list.

```
openwindow Skills;
foreach(*Skills)
  if (!@sublist)
    skill = @name;
    add Skills.$$skill;
  endif
endforeach
```

The following script adds all items from the list and category specified to an array and then displays the contents of that array in a dialog.

```
sub AddItemsToArray(listName, catName, itemArray) {
  n = removeAllElements(itemArray);
  foreach(*$$listName : $$catName)
    # Only add items that would normally be selected.
    if (!@see and @rand >= 0 and !@sublist)
      n = addElement(itemArray, @name);
    endif
  endforeach
}

# Other code...

# Get the Combat Skills and put them in an array.

array skillList[0];
AddItemsToArray("Skills", "Combat", skillList);

dialog ok, Combat Skills;
  text "Choose a combat skill.";
  listbox selection, Skills:, $$skillList;
end
```

func

To define a function *for the conversion only*, use the `func` command. For example,

```
func cs(lev,att) = att+integer((lev-50)/10);
```

defines a single-expression function called `cs` with two arguments, `lev` and `att`, that returns a value that is 1 less or greater than `lev` for each 10 `att` is less than or greater than 50.

To define a procedure, use `func` and surround the procedure body with `{/}`. For example:

```
func min(x, y) {
  if (x < y)
    return x;
  endif
  return y;
}
```

Conversion script commands may not be used within a function: they must be defined as procedures (p. 182). Commands that accept user input such as `warning` and `dialog` may therefore not be used. Use a

subroutine (p. 290) to use conversion commands. Functions are "compiled" into compact and efficient form, and execute more quickly than subroutines, though they are more limited.

Within the body of the function you cannot use any conversion script-specific variables, such as \$1, \$c, etc.

gamesystem

The `gamesystem` command indicates the game systems for the source character sheet file and destination character sheet template:

```
gamesystem Fuzion "Call of Cthulhu"
```

This command must appear second in a conversion or command script. The first argument is the game system of the original (source) character sheet. The second argument must be the game system of the new (destination) game system.

Command scripts that work on a single character sheet should use the same game system for both source and destination.

getfilename

The `getfilename` command displays a file open dialog box for the user to select a file.

```
getfilename fileName, title, pattern;
```

fileName

The name of the file is placed in this variable. If the variable is valued before `getfilename` is called, the file name in the File Open dialog is set to it. The search starts in the source directory (p. 17).

If the resulting file is in the source directory, the returned file name is made relative to the source directory.

If the user cancels the dialog, this variable is set to zero.

title

The title of the File Open dialog box. It is not required to enclose the title with quotes, but if it contains commas, semicolons or reserved keywords, it should be quoted.

pattern

The file patterns in the File Type drop-down dialog. List the name of the file type, followed by a vertical bar character (|) followed by the wildcard pattern that matches that file type (usually something like `*.bmp`). If there are multiple types, separate each (name, wildcard pattern) pair with a vertical bar. For example:

```
File Type|*.jpg;*.bmp|All|*.*
```

This field must be quoted.

Example

The following code gets the current value of a text edit field from a dialog and gets a file name, placing the name back in the original text edit field.

```
logo = $$Configuration.Logo;
getfilename logo, "Logo Graphic", "Pictures|*.jpg;*.bmp;*.png";
if (logo)
    Configuration.logo = logo;
endif
```

getitem

```
getitem [ title , ] list [ : category ], variable;
getitem Skills:Social, skillName;
getitem "Choose Spell to Mimic", "Spells", spellName;
```

The `getitem` command displays a dialog that presents a list of the available items from the item list called *list* and places the name of the selected item in the variable called *variable*. If the user clicks the **Cancel** button, *variable* will be set to 0. If the *title* is present, the Available Item dialog's title bar will contain the string. Quotes are optional around the arguments, but are required if the text contains characters such as commas, for example.

A category can be specified after the list, separated by a colon (:), to limit the items displayed to those in a single category. You might do this to allow the user to choose a Combat skill, for example.

To indicate the list or category, you can use a \$\$variable reference. For example, the following script solicits the name of a category from the user, and if one is entered asks the user to select a skill from that category. Otherwise, all skills are displayed and the item, if one is selected, is added to the Skills list.

```
dialog ok, Choose Category;
    text "Enter the name of the category";
    edittext category, Category;;
end

if (category != '')
    getitem "Choose Skill from $$category", "Skills:$$category", itemName;
else
    getitem Skills, itemName;
endif

if (itemName != 0)
    openwindow Skills;
    add Skills.$$itemName;
endif
```

See also: `insertitem` (p. 281).

if

The `if` command is used to conditionally perform actions during the conversion process.

The general syntax is:

```
if (expression)
    # commands...
elseif (expression)
    # commands...
else
    # commands...
endif
```

The parentheses surrounding the expressions (p. 263) must be present. If the expression after the `if` is true (non-zero), then the commands between the `if` and the following `elseif`, `else` or `endif` are executed. If false, then the expressions for subsequent `elseif`s are evaluated. The commands for the first one that evaluates to true (non-zero) are executed. If none of the expressions is true, the commands for the `else` are executed. Of course, if there are no `elseif`s or `elses`, then nothing is executed.

The following is an example:

```
if ($*ver < 1)
    fail The version of the original character sheet template is wrong;
endif
```

If the value of `ver` in the original character sheet is less than 1, the `fail` command will be invoked.

include

The `include` command allows common subroutine and array definitions to be included. Any other conversion script commands may be included as well.

```
include FileName.inc;
variable = "FileName.inc";
include $$variable;
```

The named file is read into memory and the commands are executed in order.

The `include` command is not a "preprocessor" command. It may occur only where another regular command may appear. That is, it cannot be placed within the body of a `list` command in place of a case, or contain a partial `if` structure.

insertitem

```
insertitem List;
insertitem Skills:Social;
```

Open the named item list and bring up the available item dialog to allow the user to select items for the list. If the optional category name is specified after a colon, only items belonging to that category will be displayed.

The `insertitem` command should be the last command in a script, because execution will continue after `insertitem` is executed. This is because the Available Items dialog is modeless. Use the `getitem` command to get an item name for use later in the script.

See also: `getitem` (p. 279).

item

The `item` construct operates on a single item in a list in the destination character sheet. It is similar to the `foreach` (p. 277) construct, except that it operates only on the specified item.

```
item (Skills, name)
# ...
enditem
```

or

```
item (Skills, index)
# ...
enditem
```

The body of the `item` (the commands between `item` and `enditem`) delimits the commands that can make item references (p. 194) for the specified item. You can also use the `options` (p. 285) command to loop through the options on the item, and the `option` (p. 285) and `set` (p. 287) commands to modify the item.

In the first form, `name` is an expression that must evaluate to a string that is an item name. If no such item exists in the specified list, an error will be reported. If there are two items in the same list with the same name, the first item will be operated on. If you "hard code" an item name, be sure to put it in quotes (because an expression is required).

In the second form, `index` is an expression that must evaluate to an integer that is an index into the specified list. In this case the list must be open (see `openwindow` (p. 285)). The index is the actual ordinal value of the item in the listbox of the list window, starting from one. That is, the first item in the list is 1, the second is 2, etc. If an item is inside a closed sublist, it cannot be accessed because it is not displayed in the listbox and has no index. This form is most useful when used in conjunction with the `+$selectedIndex` (p. 264) function to modify the selected item in a list.

If you add items (p. 266) within the body of the `item` item references will be to the added item. If your references to the item values don't seem to be working, make sure that you haven't added an item, either directly within the `item` or in subroutine calls.

Example

The following script increases the selected item in the Skills list by d10 if that item is a skill, and a d100 roll is greater than the current level of the item.

```
selected = $+selectedIndex(Skills);
if (selected < 1)
    warning "No item selected in Skills list.";
    exit;
endif

item (Skills, selected)
    if (@class != "Skill")
        warning "$*@name is not a skill.";
        exit;
    endif

    if (rand(100) > @level)
        set level = min(99, @level + rand(10));
    endif
enditem
```

list

The `list` command provides instructions for converting the items of the specified list into the new character sheet. The general syntax is:

```
list "List Name"
destination "Destination List Name";
options ... endoptions
Item 1 ~ [commands] ... end
Item 2, Item 3 ~ [commands] ... end
...
sublist ~ [commands] ... end
Item 4, sublist ~ [commands] ... end
category Category 1 ~ [commands] ... end
category Category 2, Category 3 ~ [commands] ... end
samename ~ [commands] ... end
default ~ [commands] ... end
unconverted "Unconverted List Name"
endlist
```

Following the `list` command is a list of items that can normally appear in the list named "List Name". If there are blanks in the list name, it must be surrounded by quotes.

When the `list` command is processed, the commands following the "~" for "Item 1" will be executed for all items matching the name "Item 1". An item name matches if the current name is identical, or if the converted item's name is identical to the original name when it was added. Multiple items can be specified for the same conversion by separating them by commas.

options

The `options` command (p. 285) may appear within the list, and within the command list for a matching item. When it appears within the list, it is performed for all items that match. When it appears within a matching item command list, it is performed only for that item.

sublist

Matches any sublist and performs the specified commands. This is case sensitive, so that you can name items "Sublist" and catch them in a previous clause. The `sublist` can appear in a list of item names.

category

Matches any item in a category that matches.

destination

Names the destination list for `add` commands that don't already specify one.

samename

The `samename` command matches all items that exist in the target datasheet. This is a shortcut for writing scripts that convert older versions of character sheets to newer versions, and for game systems that have a lot of overlap in item names.

default

The name `default` (or the empty string, `""`) matches any non-sublist items in the list that have not already been matched. You may omit the default item conversion if you do not desire one.

unconverted

The `unconverted` command indicates which list in the destination character sheet should receive any unconverted items. These will simply be added to the destination list, marked as unconverted. This allows you to tell Creator where to place items that were not found in the list. If you omit `unconverted`, any unconverted items will be automatically added to one of the lists in the new character sheet.

General Tips

The commands that are most likely to be used are `add` (p. 266) and assignments (p. 287). The `if` command can also be helpful for determining what to do with items that may go in one list or another, depending on the level. For example, in some game systems an item may be on a range from -5 to +5, while the converted item would appear in the "Failing" list if negative, and the "Benefit" list if positive.

Special Symbols

There are special symbols available while the commands following the "~" are being processed:

<code>\$c</code>	The cost of the item.
<code>\$l</code>	The level of the item.
<code>\$n</code>	The name of the item. This will be inserted directly into the text wherever it appears. It is not a variable reference.
<code>\$o</code>	The original name of the matching item in a <code>list</code> , or the name of the option within an <code>options</code> command. This will be inserted directly into the text wherever it appears. It is not a variable reference.
<code>\$v</code>	The value of the current option within the body of an <code>options</code> command.
<code>@variable</code>	The name of a variable associated with the item being converted (in the original character sheet). The variable names are the same as for the <code>foreach</code> (p. 194) command for filters. For example, you can access the Category name of the current item with <code>@category</code> . This can be especially useful for the <code>default</code> case in the list. In some instances you'll need to use the <code>\$*@variable</code> syntax (below) to access this information.
<code>@`option`</code>	References the value of option.
<code>\$*@variable</code>	References variable accessible within a <code>foreach</code> (p. 194) as above, except that you can access the values anywhere -- for example, within a string value or a warning command. <code>warning Not converted: \$*@item;</code>
<code>\$@@option</code>	References the value of an option in the item just added with the <code>add</code> (p. 266) command.

Example

The following is an example of a list conversion:

```
list Skills
Animal Handling      ~ add Skills.Animal Care = sk($1); end
Area Knowledge      ~ add Skills.Area Knowledge = sk($1); end
# Any other explicit conversions.
# ...
default             ~ add Skills.Medium Skill[$n] = sk($1); end
endlist
```

If the "Animal Handling" skill is in the list of skills for the original character sheet, the skill "Animal Care" added to the new character sheet. "Area Knowledge" is added as "Area Knowledge". Any other skills are added to the new character sheet as copies of "Medium Skill", renamed to be the same as the original skill. These examples also show setting the level of the items in the destination character sheet. The "sk(\$1)" indicates that the user-defined function is invoked with the level of the original item to compute the level of the new item in the destination character sheet.

menucommand

Executes an application menu command.

```
menucommand FileLoadDataSheet;
```

The argument after `menucommand` is the keyword for the the menu command (p. 245).

Care must be exercised with this command. Commands that create new character sheets or execute scripts should be avoided.

notes

Sets the notes in the destination character to the specified value.

```
notes This is a note.;
variable = "This is a note.";
notes $$variable;
notes "$*@notes";
```

To set the notes to the value of the source character sheet, reference "\$*@note" in the notes command argument. Other variables and values may be referenced normally.

To specify new lines in the body of the notes, enclose the value in double quotes use the sequence "\n" to indicate a new line. To continue to the next line, end the line with "\". For example, the following command

```
history = $$Information.History;
notes "History\n\
$$history\n\n\
Notes\n\
$*@notes";
```

produces notes similar to this:

```
History
Went to college at MIT.
```

```
Notes
Practices Psychometry regularly by concentrating on any money he receives.
```

opensublist, closesublist

Open and close the specified sublist.

```
opensublist Skills.Sublist;
closesublist Skills.Sublist;
```

Useful for adding items to a sublist after adding a sublist. After opening, the sublist itself is selecting, which means that any item added will be inserted as the sublist's first item.

openwindow

Open the specified window of the character sheet being converted or operated on.

```
openwindow Skills;
```

This is a convenience feature for scripts that preselect items for a user during character generation.

option

The `option` command sets the named option to the named value on the current item.

```
option DR = 7;
option Hobby;
option Text[Note] = Special Bonus;
```

The named option is looked up in the option list in the data sheet for the new character sheet. If the option already exists in the item the value of the current option is set to the value following the "=". (Use the `addoption` command to always add a new option.) If the option does not exist, it is added. A semicolon must terminate the option.

If the "=" is omitted, the name and value of the option as it exists in the data sheet is used. If the new option name is followed by some text in [] brackets, the option is added with the name specified in the brackets.

In a command script, the `option` command refers to the current matching item, or the item referenced in an `item` command. This allows you to cycle through all items in a list and add or change options.

See also: `addoption` (p. 267), `foreach` (p. 277), `item` (p. 281).

options

```
options
Option 1 ~ [commands] ... end
Option 2, Option 3 ~ [commands] ... end
default ~ [commands] ... end
endoptions
end
```

An `options` command cycles through all options on the current item in a `list`. When an option matches, the commands after the "~" are executed. The `default` command matches any option.

The `options` command may appear within the command list for a matched item, or in the list itself. In the former instance, the `options` command is executed when all other processing on the item is finished, allowing a "default" option conversion to be specified for a list.

For example, the following section from a `list` command shows how `options` can be used:

```
Weather Control ~
  add Change Environment[$n] = $1;
  options
  Increased Radius ~
    option Radius Multiplier = @optvalue/5+1;
  end
  Effect ~
    if (@optvalue)
      option Vary Environment = "Yes";
```

```
endif
end
endoptions
end
```

This adds a "Change Environment" item to the destination list when a "Weather Control" item is found in the source. It then sets the Radius Multiplier option in the destination item to the value of the Increased Radius divided by 5 plus 1. It then set the Vary Environment option to "Yes" if the Effect option in the original item had a non-zero cost.

optqualifier

The `optqualifier` command sets the qualifier (p. 96) for the option added by the previous `option` or `addoption` command.

```
option Spell = "Fireball";
optqualifier 10;
```

The `optqualifier` command must immediately follow an `option` or `addoption` command.

See also: `option` (p. 285), `addoption` (p. 267).

pauseProgress

The `pauseProgress` command pauses execution of the script and displays the progress dialog (p. 288).

```
pauseProgress;
```

See also: `showProgress` (p. 288), `appendProgress` (p. 267), `closeProgress` (p. 268).

purgedatasheet

These two commands remove items for the specified data sheets:

```
purgedatasheet MyData.cds;
purgedatasheetitemsOldData.cds, OlderData.cds;
```

Data sheets can be purged to avoid selecting items from them during conversion. You might wish to do this when converting a character sheet that has a different name for its data sheet than the new character sheet template. In this case, both data sheets can be loaded. To avoid getting items from the old data sheet, you can purge it.

The first form, `purgedatasheet`, completely removes the data sheet from the list of currently loaded data sheets and all the items that were loaded from it.

The second form, `purgedatasheetitems`, removes all the items loaded from the data sheet from the list of available items. The data sheet itself is still listed in the loaded data sheets list. This allows you to remove all the items from the data sheet, preventing them from being used, but if subsequent character sheets reference the data sheet it will not be loaded again. This prevents repetitive loading and unloading of out-of-date data sheets during mass conversions using the multiple character sheet update feature.

return

```
return;
return expression;
```

Exits the current subroutine (p. 290), returning the value in `expression`. If executed outside a subroutine, exits the conversion script. If no expression is specified, the return value of the subroutine will be zero.

scriptmode

```
scriptmode
```

Sets "script mode," rather than conversion mode. This should be used when writing a conversion script that is used for character generation only, without any reference to the character sheet that was open when the **File | Generate Character** command was issued.

Normally a script invoked from **File | Generate Character** command can reference items from the currently open character sheet (the source). This means that it cannot iterate through and modify item lists with the `list` command. If your character generation script does not reference the source character sheet, you should set script mode so that you have access to all commands in your character generation script.

selectitem

```
selectitem (listName, listIndex);
selectitem (listName, itemName);
```

Selects the specified item in the specified list. If the second argument is a number, the item whose index is indicated by `listIndex` is highlighted (if the list is open). If the second argument is a string, the first item whose name is `itemName` is highlighted.

If the list is not open, this command has no effect. `SelectItem` is useful for ensuring that the last item in a list is selected just before adding another item. For example,

```
selectItem (History, $+listCount(History));
add History.Log Entry[$$note];
```

ensures that the Log entry is added after the last item in the History list.

set

```
set level = expression;
set cost = expression;
set constcost = 1;
set default = 0;
set $*level = expression;
set $$$checkRequirements = 1;
set $@level = expression;
set $@name = expression;
set $@origname = expression;
set $@autoParent = parent;
set $@autoID = 0;
```

Sets a value for the current item (no prefix, or the `$*` prefix) or the item just added (indicated by the `$@` prefix). The current item exists within an active `foreach` (p. 277), `list` (p. 282) and `item` (p. 281) structures. The item just added can be set within those same confines, after an `add` (p. 266) has been executed.

The keyword values that can currently be set for an item include:

<code>autoID</code>	The item's auto ID (p. 76). Setting this to 0 will disassociate this item from all its child items.
<code>autoParent</code>	The item's parent ID (p. 76). Setting to the auto ID of another item will cause this item to be deleted when the parent with the specified auto ID is deleted. Setting this to 0 will disassociate this item from its parent. The item's Delete Auto-added Items (p. 77) flag must also be set. This is particularly useful for adding an item to a character sheet that you wish to have deleted when the parent item is deleted.
<code>default</code>	The item's default checkbox. The expression should evaluate to 1 or 0.

<code>cost</code>	The item's cost. The expression may be any numerical value. This should be used only for items that have no cost formula. Items with a cost formula should set the level instead.
<code>level</code>	The item's level. The expression is any expression that the item's level can assume.
<code>name</code>	The item's name. The expression should evaluate to a string.
<code>notes</code>	The item notes. The expression should evaluate to a string.
<code>origname</code>	The item's original name. The expression should evaluate to a string. This is useful for forcing an item that has just been added to have a different original name, which controls the actual item name that is used when the item is converted.
<code>req</code>	The item's requirements. This string must be formatted in the same way as the requirements for text data sheets (p. 220).
<code>varname</code>	The variable name. The expression should evaluate to a string.

The following sets are character sheet wide:

<code>constcost</code>	If set to 1, sets the constant cost checkbox (p. 46) in the Modify Character Sheet Info dialog. If 0, unchecks the constant cost checkbox.
------------------------	--

The following sets are application-wide:

<code>\$\$\$checkRequirements</code>	If set to 1, turns on the Check Requirements checkbox in the Available Items dialog. Any items with requirements will be checked. If set to 0, requirements checking is turned off.
--------------------------------------	---

For an example of `set level` see the `item` (p. 281) command.

Example

The following shows how the `set @$level` command can be used. In the conversion the level of the source skill is multiplied by 10 and added to the value of the Base option of the destination skill. Since the destination skill hasn't been added yet, its options can't be accessed in the `add` command, so this must be done in two steps.

```
list Skills;
# ...
Firearms ~
  add Handgun;
  set @$level = $1 * 10 + $$$Base;
end
# ...
endlist
```

showProgress

The `showProgress` opens the Progress dialog (p. 295) during a conversion script for display of informational text to the user. The one-argument form of the command opens the dialog (if it's not already open) and sets the title bar to the argument. The two-argument form of the command additionally sets the text field of the dialog to the second argument.

```
showProgress "Current Year: $$year";
showProgress "Current Year: $$year", "$$firstResult";
```

The text should be quoted if it contains any special characters (such as semicolons) or references to variables.

The progress dialog will remain open until the next `closeProgress` or `dialog` (p. 271) command is executed.

The `appendProgress` (p. 267) adds text to the text field of the dialog. The `pauseProgress` (p. 286) command stops execution of the command script until the user clicks the **Continue** button in the progress dialog.

This command is typically used when a command script begins a long process. The `showProgress` command is often used with a single argument to open the dialog and give it a title. During each step of the process an `appendProgress` command is executed to let the user see what's going on. The `showProgress` command may be executed again to change the title bar. At the end of the process `pauseProgress` may be executed to pause the script and allow the user to see the contents of the dialog -- otherwise, it will just disappear the next time a `dialog` command is executed.

The script can check the `+$canceled` (p. 265) flag to see if the user has clicked the **Cancel** button in the progress dialog. If the script detects this, it should call `closeProgress` to close the dialog and reset the `+$cancel` flag to false (to avoid future erroneous detection of the same cancelation).

See also: `appendProgress` (p. 267), `pauseProgress` (p. 286), `closeProgress` (p. 268).

showreq

The one-argument version of the `showreq` command shows the requirements dialog for the most recently added item, or the current item of the `item`, `foreach`, or `list` commands. The three-argument version shows the requirements for the item in the specified list. The four-argument version allows the requirements string to be specified independently of the item.

```
showreq ok;
showreq Spells, Fireball, ok;
showreq $$listName, $$spellName, returnVariable;
showreq list, item, variable;
showreq list, item, reqString, variable;
```

The return variable is set to non-zero if the user clicks the **OK** button in the requirements dialog, and to zero if the user pressed **Cancel**.

See also `checkreq` (p. 268).

showallreq

The `showallreq` command shows the requirements for all unsatisfied items.

```
showallreq;
```

The user can use the tools in the Unsatisfied Requirements dialog to make the necessary changes to satisfy requirements.

skip

```
skip;
```

When `skip` appears in the command list for a matching item, no conversion is done on the item, and it is not marked as converted.

sortlist

Sort the specified character sheet list in the destination character sheet..

```
sortlist listName [, sortKey [, direction, [ sublistPos [, recursion ]]]];
```

The second through fifth arguments are optional, though if any optional argument is present, the previous arguments are required.

The `listName` is the name of the list to sort. The optional `sortKey` is one of the following values:

<code>name</code>	Sorts by item name.
<code>category</code>	Sorts by item category.
<code>class</code>	Sorts by item class.
<code>cost</code>	Sorts by item cost.
<code>level</code>	Sorts by item level.

If the `sortKey` is omitted, `name` is assumed.

The possible values for `direction` are:

<code>forward</code>	Sort lowest values first.
<code>reverse</code>	Sort highest values first.

The default is `forward`.

The possible values for `sublistPos` are:

<code>intersperse</code>	Sublists are interspersed among other entries.
<code>beginning</code>	Sublists are placed together at the beginning.
<code>end</code>	Sublists are placed together at the end.

The default is `intersperse`.

The possible values for `recursive` are:

<code>recursive</code>	Sublists are also sorted.
<code>top</code>	Only the top level is sorted.

sub

```
sub addValues(param1, param2) {
    return param1 + param2;
}
```

```
value = addValues(2, 2);
```

Define a conversion script subroutine. Any number of parameters may be named (including none). When the subroutine is executed, the parameter values will be evaluated and assigned to "local" variables that exist only within the scope of the subroutine.

A subroutine is different from a function (p. 278) because it can use any conversion script commands, including `dialog` and `warning`. Parameters may be passed to the subroutine.

A subroutine is called in two ways:

```
value = addValues(2, 2);
complain("Error", "This is the text of the prompt");
```

If you call a subroutine like a function, assigning its value to a variable, the subroutine call must be the only component of the expression.

To call a subroutine directly (without assigning the return value to a variable), name the subroutine and enclose any arguments in parentheses. The return value, if specified by a `return` command, is simply ignored.

Subroutines may call other subroutines, but are limited to a total depth of 10 calls.

The following example displays a dialog, using the parameter values as the title and text of the dialog.

```

sub complain(prompt, msg) {
    dialog ok, $$prompt;
        text "$$msg";
        text "Click OK to continue, or Cancel to exit.";
    end
    if (!ok)
        exit;
    endif
}

complain("Error", "You have made a mistake");

```

submenu and menuitem

This command describes a cascading menu that will be added to the Data menu for the command script containing it.

```

submenu Submenu;
    menuitem Menu Item 1, Ctrl+1 { command1(); }
    menuitem Menu Item 2 { command2(); }
    submenu Submenu;
        menuitem Menu Item 2, Ctrl+3 { command3(); }
        menuitem Menu Item 4, Ctrl+Shift+4 { command4(); }
        menuitem Menu Item 5 { command('5'); }
    endsubmenu
endsubmenu

```

The name in the menu appears after the `submenu` keyword. An ampersand (&) may be specified to indicate the key to depress when the menu pops up. To display & you must double it.

Following the `submenu` command is a list of `menuitem` and `submenu` commands. Each `submenu` must be closed by an `endsubmenu`.

Only one `submenu` may appear in a command script file. It must be at the beginning of the file, after the `gamesystem` command. A single `menuitem` command may appear in place of the `submenu`. This can be used to indicate the accelerator key for that menu item without adding a cascading menu.

menuitem

A menu item describes the command name that will appear on the **Data** menu, an optional key accelerator that the user can press to select the menu item, and the script commands that are executed when the menu item is selected.

```

menuitem Menu Item 4, Ctrl+Shift+4 { command4(); }
menuitem Menu Item 5 { command('5'); }

```

The `menuitem` command may appear only inside a `submenu` command or immediately after the `gamesystem` command.

The command name may include a "&" character to indicate that the next character will be underlined when the command is displayed on the menu. Pressing that character on the keyboard when the menu is displayed will select that item. Doubling the "&" displays a single "&" in the menu item.

The optional key accelerator consists of a sequence of modifier keys, separated by "+", followed by a letter, number or function key. The accelerator must include CONTROL or ALT. The following are examples of valid accelerator key sequences: Alt+Ctrl+R, or Ctrl+Shift+A, Ctrl+1, Alt+Ctrl+Shift+Q. If the accelerator key is present, it must be separated from the command name by a comma. If you specify a sequence that is defined by the application, the application's definition will override yours. Thus, Ctrl+C (copy), Ctrl+X (cut), Ctrl+V (paste), etc., are reserved for system use. You can see which accelerator keys are reserved by looking at the menus. Only accelerators that are specified when a character sheet is being used are reserved; accelerators used only when editing a print template may be freely used.

The commands to be executed must be surrounded by { and } brackets. Any commands may appear, but typically a single subroutine call is used to keep the definition of the `submenu` simple.

Script File Execution

When one of the items in a cascading menu is selected, the entire command script for the item is read and executed. Then the script commands for the selected menu item (the command list between the braces) are executed.

The execution of the entire script allows the script to define any common variables and subroutines (p. 290) for use by the menu item commands. It also means that you should have only initialization code in the main body of the script. Any command-specific code should be encapsulated in a subroutine definition.

In the case where a single `menuitem` is specified in order to define an accelerator key for the command script, you may choose to put the actual logic in the main script, and have a "no-op" command list for the menu item:

```
menuitem Optmi&ze Character, Ctrl+Shift+Z {;}
```

switch

The `switch` command selects a group of statements to execute based on the value of an expression and a matching case.

```
switch (expression)
expression1 ~
    commands;
end
expression2, expression3 ~
    commands;
end
default ~
    commands;
end
endswitch
```

The expression between the parentheses after the `switch` keyword is evaluated, and then compared to all the case expressions that follow. If a case expression matches, the commands between the tilde (~) and the `end` are executed. Control then passes to the statement following the `endswitch`. Once an expression matches, no other cases will be examined.

Multiple expressions may appear in each case, separated by commas.

If no case matches the expression and a `default` clause is present, the commands for the default are executed. Only one default may appear, and it must be the last clause in the switch.

Note that any type of expression may be used in the case expressions, not just constants, so your cases can be more flexible:

```
switch (step)
1 ~
    warn Step 1;
end
2 ~
    warn Step 2;
end
step = 3 and type = "Skill" ? step : 0 ~
    # This trick lets you match a complex condition.
    warn Step 3a;
end
3 ~
```

```
warn Step 3;
end
default ~
  warn Step = $$step;
end
endswitch
```

The `switch` statement is essentially an `if ... elseif ... else ... endif` statement that allows you to omit the first half of an equality expression in each case.

title

This indicates the title of a command script. It must appear at the beginning of the script, immediately after `gamesystem`. This title is displayed on the Data menu for character sheets of the indicated game system, unless a `submenu` or `menuitem` command is specified.

var

You can define variables for the duration of the conversion script. The following would create a variable called "varname" with the value 10:

```
var varname = 10;
```

You simply reference it by name in conversion expressions.

warn

If you want to tell the user about some condition, use the `warn` command. Its format is:

```
warn Warning message;
```

The warning message must be terminated by a semicolon.

while

You can loop with the `while` command:

```
while (i < 10)
  # ...
endwhile
```

As long as the expression within the parentheses after the `while` is true, the commands between the `while` and the `endwhile` will be executed. Be careful to avoid infinite loops! Pressing ESC or CTRL+BREAK will break out of a `while`.

Conversion script variables

Where conversion scripts call for expressions, you can use variables from three different "contexts." The first context is that of the conversion script. Variables in this context are defined by the `array` (p. 268), and `var` (p. 293) commands. Any functions (p. 278) you define in the conversion script are defined in its context.

The second context is that of the original character sheet. References to variables in that context are prefixed with "\$*". For example, in a character sheet with a strength variable named `str` you could reference it with `$*str`.

The third context is that of the new character sheet. References to variables in that context are prefixed with "\$@".

References to variables defined in the conversion script context are made without any special indicators. The following example sets the `dx` field of the dialog named `main`, setting it to value obtained by indexing into the array defined in the conversion context by the `dexterity` variable in the original character sheet.

```
Main.dx = rating[$*dexterity];
```

There are special variable references that are in effect during the processing of the members of a list (p. 282) structure: `$l` is the level of the current item being processing in the list. `$c` is the cost of the current member. `$n` is the name of the current member.

You can also reference variables in the conversion context by preceding them with "\$\$".

Note that `$@`, `$*` and `$$` substitutions are simple textual replacement, and as such cannot be used to access elements of arrays. If you wish to treat the value as a string, you must enclose the reference within quotes. For example, if the value of `$@name` is a string and you wish to assign it to a variable in the script, you would write:

```
var = "$@name";
```

Converting Character Sheets from Other Applications

With the appropriate setup, **GURPS Character Builder** can automatically convert character sheets from other applications to **GURPS Character Builder** format. This requires some setup, a conversion script and a specially written conversion program.

External Converter Setup File

Each external converter requires a file with the .CSX extension to be present in the source directory for **GURPS Character Builder**. The source directory is where all the data sheets and character templates are stored. This setup file has the following format:

```
[Conversion]
Extension=*.cha
Template=champ.cst
Description=HeroMaker Files
Application=cvhromkr.exe
Script=cvhromkr.cnx
```

Application	The name of the converter application to run. This application must take a particular set of arguments on the command line (see below).
Extension	The extension that identifies the type of external character sheet, in the form *.xxx. The * <i>must</i> be present.
Description	The textual description of the files represented by the extension.
Template	The name of the GURPS Character Builder character template that the named script uses as a basis. This may be omitted, in which case the template specified in the output file from the conversion program will be used as the template.
Script	The name of the script to run after the external converter. These end in .CNX by convention. The script is optional, but if not specified, the character sheet produced by the converter application must be identical in format to the character template that is specified.

The Process

When **GURPS Character Builder** starts up, it searches the source directory for all files with .CSX extensions. Whenever the Open Character Sheet dialog comes up, **GURPS Character Builder** adds the extension and description for that file type to the list of file types in the dialog.

When you choose a file with one of those extensions, **GURPS Character Builder** will automatically run the associated converter application. This application must read the name of the input and output files from the command line, and write a **GURPS Character Builder**-compatible character sheet to the specified output file name.

GURPS Character Builder will then take read that file, and use the specified script to convert it to a character sheet based on the specified character template.

The Converter Application

The Converter Application has the following command line:

```
cvrt input.ext output.chr error.txt
```

Argument	Meaning
cvrt	The name of the converter application.
input.ext	The name of the input file, with the specified extension.
output.chr	The name of the <i>GURPS Character Builder</i> file that the converter must generate. The application will create as specified below.
error.txt	The name of the file for error messages. If an error occurs, the converter should write the error message to this file and then return a non-zero value as the exit code for the application.

If the conversion is a success, the converter should return 0 as the exit code for the program.

The *GURPS Character Builder* Output File

The converter application must create a file in the ***GURPS Character Builder*** character sheet format (p. 234). Normally ***GURPS Character Builder*** saves files as "binary" to save space, but there are equivalent binary and text formats. It's easiest to write the text format.

To see examples of how ***GURPS Character Builder*** formats character sheets, turn off the Binary Save checkbox in the Preferences dialog. Thereafter, character sheets will be saved as text only. You can examine them in a text editor to see examples of how other ***GURPS Character Builder*** files are formatted.

Progress Dialog

This progress dialog appears during command scripts. It displays information allowing you to monitor the progress of the command script during long processes that don't require user interaction.

You may pause the script while it is running by click the **Pause** button. The **Pause** button will become inactive and the **Continue** button will become active.

To restart the script, click **Continue**. The **Continue** button will become inactive and the **Pause** button will become active.

To cancel the execution of the script, click **Cancel**. The exact behavior when you cancel will depend on the script. You will usually be asked whether you wish to resume the execution of the script, or cancel it, though this will vary from script to script.

The script may automatically pause so that you can view the contents of the progress dialog when the script's extended process is complete. When this occurs, the **Continue** button will become active and the **Pause** button will become inactive, exactly as if you had clicked the **Pause** button yourself. When this occurs, click the **Continue** button to have the script proceed.

You may also click **Cancel** at this point to cancel the script. The exact behavior will again depend on the script involved.

Text Editor

The GURPS Character Builder text editor can be used to edit text files such as include files, conversion scripts, etc. The maximum size of a text file that GURPS Character Builder can edit is 1,000,000 characters.

Click here for help on general menu commands (p. 2). The following commands are specific to text editor windows.

File | Page Setup...

Invoke the Text File Page Setup dialog (p. 7), which sets the margins and whether the footer will be printed.

Edit | Copy (CTRL+C)

Copy the highlighted text to the clipboard. If no text is highlighted, the current line is copied to the clipboard.

Edit | Paste (CTRL+V)

Paste text from the clipboard into the file. If the text on the clipboard is a line (see above), the line is pasted after after the line where the caret is.

Edit | Delete Line (CTRL+D)

Delete the current line.

Edit | Search (CTRL+F)

Search for text.

Edit | Search Again (F3)

Search again for the last text pattern.

Edit | Search and Replace (CTRL+R).

Search for text and replace it with other text.

Edit | Go to Line

Enter a line number and click **OK**. The caret will be placed at the beginning of that line.

Edit | Font...

Select the font to use for displayed text files. After you select the font, all text edit windows created subsequently will use that font. The default font is 10-point Courier New.

Scroll (CTRL+UP and CTRL+DOWN arrow)

Scroll the text up and down without moving the caret.

Save Text File

Select the folder where the file should be saved. Enter the name that the file should be saved under in the **File name** edit box.

Search

Find Text

Enter the text to find here.

Match Case

If this is checked, the text will match only if the case is identical. For example, "Name" will not match "name."

Whole Word

If this is checked, the text will match only if it occurs as a separate word. For example, "name" will not be matched in the string "varname" because it is not a separate word.

Search

Find the next instance of the text in the **Find Text** field.

Search and Replace

Find Text

Enter the text to find here.

Replace with

Enter the replacement text here.

Match Case

If this is checked, the text will match only if the case is identical. For example, "Name" will not match "name."

Whole Word

If this is checked, the text will match only if it occurs as a separate word. For example, "name" will not be matched in the string "varname" because it is not a separate word.

Search

Find the next instance of the text in the **Find Text** field.

Replace

Replace the first instance of the text in the **Find Text** field found after the caret with the text in the **Replace with** field.

Replace All

Replace all instances in the file of the text in the **Find Text** field with the text in the **Replace with** field.

Terms

Character Sheet

A character sheet is a repository for all the information about a character. A character sheet must be defined for a particular game system (p. 298).

Depending on the game system, a character sheet may contain dialogs for specifying your character's name, physical description, numerical attributes, etc., and lists of skills, advantages, possessions, and so forth.

Game System

A game system defines what sorts of abilities your character has and (typically) how those abilities are expressed in numerical terms. It also defines the types of skills, advantages, disadvantages, powers, etc. that your character may have.

Examples of game systems include Fuzion™, the Hero System®, GURPS®, Call of Cthulhu®, etc.

Character Sheet Template

A character sheet template is the "master copy" of all character sheets (p. 298) using the same game system (p. 298). When you create a new character sheet (p. 2), you base it on an existing character sheet template.

You may have more than one character sheet template for a particular game system. For example, you might have different templates using the same game system for medieval campaigns, one for modern-day campaigns, one for cyberpunk campaigns and one for super hero campaigns.

Data Sheet

A data sheet (p. 144) is the "master list" of items that you can select for your character. It may contain skills, advantages, possessions, super powers, magic spells; anything that can be added to the lists that describe your character's abilities and characteristics.

A data sheet must be dedicated to a particular game system (p. 298). You may have more than one data sheet for a game system. You might separate out the basic components of your game system into one data sheet, and have separate data sheets for super powers, magic spells, futuristic weapons, etc.

Binary Data Sheet

A binary data sheet is similar to a data sheet (p. 298), except that it uses a different method for editing the data. Binary data sheets are more work to maintain than data sheets because no include file is used and there are no macro definitions. That means changes to the definitions for the items are not automatically reflected when the macros are changed, which means that each item has to be modified individually.

For this reason, binary data sheets are no longer recommended.

Print Template

A print template is a file that is used as a pattern for printing a character sheet on paper. A print template must be dedicated to a single game system (p. 298). You can create (p. 2) as many different print templates for a game system as you like. You may wish to have different templates for medieval, futuristic and super-hero campaigns, or have different templates for printing out different aspects of your character (for example, a main print template for all characters and a secondary template for wizards that lists spells).

Groups

Groups are used to list character sheets that you work with together as a group. A group may be a party of player characters, or a list of NPCs the player characters will encounter, or a group of templates. You can print the character sheets together, copy the text of character sheets to the clipboard, or save the text to files.

Filters

Filters are used to produce text based on the contents of your character sheet. They can be used to copy (p. 9) a textual representation of your character sheet to the clipboard, to save (p. 3) your character sheet to a text file, or to produce text in print templates (p. 109).

Index

!, 169
 \$\$argdesc, 230
 \$\$argtype, 231
 \$\$define, 227
 \$\$elseif, 228
 \$\$exp, 228
 \$\$hidemacro, 232
 \$\$if, 227, 228
 \$\$ifdef, 227, 228
 \$\$ifnull, 227, 228
 \$\$include, 233
 \$\$macrodesc, 230
 \$\$repeat, 229
 \$\$scan, 228
 \$@@@checkRequirements, 288
 \$+available, 264
 \$+canceled, 265
 \$+currentWindow, 265
 \$+inlist, 263
 \$+inSourceList, 263
 \$+listCount, 264
 \$+randselect, 263
 \$+selectedIndex, 264
 \$+tickCount, 265
 .bak files, 15
 ?addcat, 82
 ?adj, 80
 ?allowdup, 82
 ?cat, 80
 ?charge, 81
 ?checkreq, 82
 ?choose, 82
 ?clear, 81
 ?clearopt, 81
 ?close, 83
 ?datasheet, 83
 ?default, 83
 ?defitem, 82
 ?dupopt, 79
 ?if, 81
 ?itemref, 81
 ?list, 80
 ?ncopt, 79
 ?nocharge, 81
 ?noreq, 84
 ?nrand, 83
 ?open, 83
 ?opt, 79
 ?qualifier, 84
 ?rand, 82
 ?rmopt, 80
 ?selexp, 83
 ?seloptval, 80
 ?setval, 83
 ?zref, 81
 @array, 188
 @assign, filter command, 188
 @bgcolor, 189
 @blankcharsheet, 185
 @bold, 189
 @bolditalic, 189
 @clipformat, 189
 @dest, 190
 @exit filter command, 191
 @extension, 191
 @fail filter command, 192
 @firstindent, 192
 @font, 192
 @fontinfo, 193
 @footer, 190
 @for, 194
 @foreach, filter command, 194
 @gamesystem, 186
 @header, 190
 @hline, 198
 @if, filter command, 199
 @italic, 199
 @keepnext, 199
 @landscape, 200
 @leftindent, 200
 @message filter command, 200
 @need, 200
 @optexp, 201
 @option, 201
 @options, 201
 @optname, 201
 @optnumbervalue, 201
 @optqualifier, 201
 @opttextvalue, 201
 @opttype, 201
 @optvalue, 201
 @output, 202
 @parinfo, 202
 @plain, 203
 @ruleset, 186
 @samepage, 203
 @sortexp, 204
 @sortorder, 204
 @sub filter command, 204
 @tabs, 205

- @textcolor, 205
- @trans, 206
- @translate, 206
- @var filter command, 206
- @vline, 207
- @while, filter command, 207
- @wrtab, 208
- @writepicture, 208
- abbreviations, 75, 97
- abs, 171
- absolute value, 171
- accelerator keys, 291
- add as list, 77
- addautoitems, 267
- addcat, automatic item directive, 82
- addelement, 171
- adding campaign-specific items, 137
- adding controls, 155
- adding dialogs and lists, 41
- Adding graphic objects, 109
- addition option, 95
- addoption conversion command, 267
- adj, automatic item directive, 80
- adjustment option, 95
- adjustments, 87
 - reverse, 88
- adjustments, not performing, 77
- adjustments, syntax, 241
- alias, 217
- aliases, 75, 97
- aliases, using, 248
- align menu, 114
- aligning controls, 156
- allowdup, automatic item directive, 82
- alternate format, 37
- alternate formats, 44
- alternate item formats, 103
- alternate list format, 34
- AND operator, 169
- appendprogress conversion command, 267
- apply selection rules, 57
- array, 268
- array, adding element, 171
- array, lookup, 104, 175
- array, removing element, 177
- array, removing elements, 177
- array, sorting, 179
- arrays, 188
- Arrays
 - defining, 167
- arrays, indexing, 174
- assign filter command, 188
- assignment, 265
- assignment in procedures, 184
- associated files, 5
- associated files, adding, 6
- associated items, 12
- auto ID, 76
- auto parent, 76
- autoadded, 195
- autodelete, 76
- automatic item ID, 16
- automatic items, 55, 75, 79, 81
 - editing, 84
- automatic items, adding in script, 267
- automatic items, avoiding duplication, 77
- automatic items, charging cost of, 78
- automatic items, deleting, 76, 77
- automatic items, not adding, 77
- automatic items, random selection, 82, 83
- automatic items, undeleting, 9
- automatic items,options, 81
- automation, editing properties, 107
- auxiliary character sheet files, 46
- auxiliary cost option, 93
- available, 264
- Available Items dialog, 56
- available items, context menu, 57
- available items, displaying all, 56
- background color, 158, 160, 189
- background color, dialog window, 42
- background color, dialogs, 16
- background page, 120
- backups, 15
- bad value dialog, 15
- bak files, 15
- baseable on default, 77
- based on default, 77
- basic character sheet info, 45
- basic properties, 75
- binary data sheet, 298
- binary format, 16
- bitmap control, editing, 162
- bitmap files, inserting into print template, 110
- bitmap files, reading into character sheet, 50
- bitmap files, writing, 208
- bitmap, button, 161
- bitmap, dialog control, 156
- bitmaps, 50
- blank character sheets, 187
- blank character sheets, printing, 187
- blankcharsheet, 185
- BMP, 54
- BNF, 236
- bold italic text, 189
- bold text, 189
- bonuses, 170
- bonuses, getting only base value, 170
- bonuses, getting only bonus value, 170
- border, omitting in control, 158, 159
- breakpoints, setting, 256
- bring to front, 10

- button bar, 15, 63
- button control, editing, 161
- button, dialog control, 156, 274
- calculations, character sheet info, 46
- canceling script execution, 265
- cards, drawing at random, 28
- cascading menus, 291
- cat, automatic item directive, 80
- categories, 78
- categories on options, 173
- categories, adding in macro data sheets, 147
- categories, data sheets, 126
- category name, new, 62
- category, in conversion, 283
- category, selecting, 72
- categoryBonus, 171
- ceil, 171
- ceiling, 171
- cell lines, 113
- cells, 113
- centering objects, 114
- change expression, 44
- changing items, 10
- changing option list name, 128
- changing print templates, how to, 131
- character generation, 138
- character list files, 22
- character sheet, 298
- character sheet format, 234
- character sheet info, 45
- character sheet references, 185
- character sheet template, 45, 298
- character sheet type, 45, 117
- character sheet, choosing, 74
- character sheet, loading as data, 137
- character sheets
 - group files, 140
- character sheets, converting, 130
- character template, creating, 132
- chargen script, 47
- chargen template, 47
- charging for automatic items, 81
- check box control, editing, 160
- check expression, 41, 174
- check expression violation, disallowing, 77
- check expression, computed option value, 101
- check expression, list control in dialog, 165
- check expression, list window, 43
- check expression, option, 101
- check expression, option, 223
- check expressions, 105, 106
- check message, 41, 44, 158
- check message, computed option value, 101
- check requirements, 265
- check requirements for choosing automatic items, 82
- checkallreq, conversion script, 268
- checkbox, 271
- checkbox option format, 225
- checkbox, dialog control, 156
- checkexp, 217, 223
- checking requirements, 15
- checkreq, conversion script, 268
- checkRequirements, 171, 288
- childiteminfo, 171
- childiteminfo, example, 183
- children inherit options, 72
- choose directory, 19
- choose macro argument value, 152
- choose value, 73
- choose, automatic item directive, 82
- chrval, 171
- circle, 109
- clearing automatic item modifiers, 81
- clearing options in automatic items, 81
- clipboard format command, 189
- close, automatic item directive, 83
- closeprogress conversion command, 268
- closesublist, 284
- closing files, 4
- coin flipping, 26
- color, background, 16, 158, 160
- combat record sheet, 22
- combat record sheets, 19
- combo box script, editing, 164
- combo box, dialog control, 156
- combo box, editing, 157
- combo boxes, editing, 157
- combobox, 271
- command script, item, 219
- command scripts, 260
- commands, filter, 187
- comparing character sheet totals, 263
- computed option value, 101, 224
- computed value option, 71
- concat, 171
- concatenating strings, 171
- conditional automatic items, 81
- conditional macro commands, 227
- ConfigParam, 171, 247
- configuration parameters
 - setting, 178
- confirm multiple deletion, 16
- conflicts, merge, 128
- constant cost, 46
- constant costs, 45
- constraining object shapes, 111
- context conversion command, 269
- continuation page, 120
- continuation, example, 122
- Control menu, 155
- controls, adding, 132

- controls, aligning, 156
- controls, ordering, 164
- conversion expressions, 263
- Conversion script format, 262
- conversion, details, 259
- converting character sheets, 130, 258
- converting character sheets from other applications, 3, 294
- converting item lists, 282
- converting old character sheets, 131
- convlevels, conversion script command, 269
- copy, 9
- copy filter, 5
- copy formats, 10
- copyall, conversion script command, 269
- copydefines, conversion script command, 269
- copydialog, 270
- copying associated items, 12
- copying dialogs, 270
- copyitem, conversion script, 270
- copyleft, 270
- copynotes, 270
- copyoption, 270
- copyright information, 64
- copyright, character sheet, 45
- cost expression, computed option value, 101
- cost expressions, 105
- cost formula, total, 107
- cost function, 174
- cost prompt, 43
- cost prompt for lists in dialogs, 43, 165
- costs, computing with options, 92
- costs, constant, 45
- costs, floating, 45
- count of items in category, satisfying, 34
- countItems, 171
- create unique variable, 77
- creating a new character, 130
- ctrlsize, conversion command, 275
- current window, 265
- currentFile, 181
- currentFile, getglobal argument, 173
- custom screen layouts, 139
- custom.sct, 242
- cut, 9
- Data Menu, 32
- data menu, modifying, 41
- data sheet
 - binary, 298
 - loading delay, 16
 - macro, 298
- data sheet example, text format, 215
- data sheet ID, 76
- data sheet info, 126
- data sheet, editing, 126
- data sheet, loading, 4
- data sheet, macro, 144
- data sheet, specifying in, 270
- data sheet, text format, 211
- data sheet, referencing all items in category list, 197
- data sheets, creating, 134
- data sheets, defines, 213
- data sheets, merging, 127
- data sheets, old-style, 134
- data sheets, reloading all, 5
- data sheets, setting for character sheet, 45
- data sheets, specifying for character sheet, 46
- data, loading character sheet as, 137
- datasheet conversion command, 270
- datasheet, automatic item directive, 83
- datasheets, 186
- date, 186
- datecreated, 186
- debugger, script, 253
- default, 283
- default formulas, 105
- default item list conversion, 283
- default print template formats, 12
- default value, 106
- default values, 105
 - rules, 252
- default, automatic item directive, 83
- default, based on, 77
- DefaultValues, 252
- define, in data sheets, 213
- defines, 166
- defines, assigning to, 265
- defines, clipboard functions, 167
- defines, macro data sheet, 147
- Defining arrays, 167
- defining functions, 166
- defining variables, 166
- deleteitem conversion command, 271
- deleting dialogs and lists, 41
- deletion, multiple, 16
- description of character sheet template, 45
- destinations, filter, 190
- details, 186
- details window, 13
- details, page setup, 7
- detailshtml, 186
- detailsrtf, 186
- detailstext, 186
- Dialog, 50
- dialog editing, 155
- dialog fiels, referencing in filters, 186
- dialog type, 43, 104, 165
- dialog, background color, 42
- dialog, bad value, avoiding, 15
- dialog, conversion script command, 271
- dialog, font, 42

- dialog, skin, 42
- dialogFieldValue, 172
- dialogs lists, 164
- dialogs, adding and changing, 41
- dialogs, new, 41
- dialogs, tab order, 10
- dialogsize, conversion command, 275
- Die Roller, 24
- die rolling function, 178
- die specifications, 26
- dieroller, conversion command, 275
- directory, 19
- directory listing, 174
- disallow check expression violation, 77
- disallow satisfying requirements, 78
- display all available items, 56
- display expression, options, 98
- distributed cost adjustment, 95
- distributed level adjustment, 96
- drag and drop, 55
- draw menu, 113
- dropdown, 272
- dround, 172
- duplicate, 9
- duplicate item priority, 76
- duplicate string function, 172
- duplicates expected, 77
- dupopt, automatic item directive, 79
- dupstr, 172
- edit field format, 158
- edit field formula, 159
- Edit function definition, 166
- edit item name, 73
- Edit Menu, 9
- edit on insertion, 97
- edit text, dialog control, 155
- edit text, editing, 159
- edit variable, dialog control, 155
- editcombo, 273
- editing categories, 78
- editing combo boxes, 157
- editing defines, 166
- editing dialogs, 155
- editing graphic objects, 110
- editing item properties, 75
- Editing Items, 68
- editing keys, special, 73
- editing options, 96, 97
- editing print templates, 109
- editing requirements, 88, 89
- Editing Variable Definitions, 166
- editing variables in dialogs, 157
- edititem conversion command, 276
- editor, 16
- edittext, dialog control, 273
- ellipse, 109
- English measurement, 8
- eval, 172
- evaluateDefault, 172
- evaluation function, 172
- exclude conversion command, 276
- excluding cost from total, 75
- excluding items from total, 55, 68
- exec, conversion command, 276
- exit, conversion command, 276
- exit, filter command, 191
- exiting the application, 8
- expression option, 93
- expression, check message, 158
- expressions, 105, 169
- expressions, check, 106
- extension, 191
- face, font, 114
- fail, 277
- fail, filter command, 192
- File menu, 2
- file name of character sheet, referencing in filter, 186
- file, new, 2
- fileExists, 172
- files, printing auxiliary, 30
- filter
 - @for, 194
 - filter @fontinfo, 193
 - filter commands, 185, 187
 - @output, 202
 - gamesystem, 198
 - gettext, 198
 - filter commands, @assign, 188
 - filter commands, @clipformat, 189
 - filter commands, @fontinfo, 193
 - filter commands, @for, 194
 - filter commands, @foreach, 194
 - filter commands, @if, 199
 - filter commands, @options, 201
 - filter commands, @sub, 204
 - filter commands, @trans, 206
 - filter commands, @var, 206
 - filter commands, @while, 207
 - filter description, 190
 - filter destinations, 190
 - filter files, creating, 185
 - filtering multiple character sheets, 181
 - filtering multiple files, 19
 - filters, 185, 298
 - getting text from users, 198
 - turning off output, 202
 - filters, asking for, 16
 - filters, setting, 5
 - find items, 57
 - finding names, 23
 - findItemValue, 172

- ul style="list-style-type: none; padding-left: 0;">
- first indent, 125
- firstindent, 192
- flags, item, 76
- flip coins, 26
- float, 173
- floating costs, 45
- floating point values, 182
- floating point, converting to, 173
- floor
 - mathematical, 174
- flowing text, 111
- font, 192
- font expression, 163
- font face, 114
- font index, 163
- font list, 163
- font, dialog window, 42
- fontinfo filter command, 193
- footers, printing for text files and details, 8
- for, conversion script command, 277
- for, filter command, 194
- foreach, conversion script command, 277
- foreach, filter command, 194
- form setup, 6
- form size, 117
- format, 172
 - adjustments, 87
 - automatic items, 79
 - format, 102
 - requirements, 220
- format string, default for list members, 43, 165
- format, character sheet, 234
- format, data sheet text, 211
- format, edit field, 158
- format, list display, 102
- format, paragraph, 14
- format, picture description files, 251
- formats, alternate item, 103
- formats, copying, 10
- formats, default print template, 12
- formula control, editing, 161
- formula, dialog control, 156
- formula, edit field, 159
- formula, total cost, 107
- fraction option, 93
- fractions, 16
- func, 278
- func, conversion script command, 278
- function definition, 166
- functions, 182
- functions, predefined, 171
- game system, 45, 198, 298
- game system level, 117
- game system preferences, 15
- game system, configuration parameters, 247
- game system, creating new, 130
- game system, setting, 117
- gamesystem, 186
- gamesystem, conversion script command, 279
- genchar script, 138
- genchar template, 138
- generate character, 261
- generic item flag, 76
- get size, 118
- getAssoc, 173
- getCategoryList, 173
- getfilename, conversion command, 279
- getFileValue, 173
- getglobal, 173
- getitem, conversion script command, 279
- getMap, 173
- getsourcedir, 173
- global variables, 181, 183
- go to line, 151
- goto line, 256
- graphic objects, editing print templates, 110
- group box, dialog control, 156
- group box, editing, 160
- group file
 - save as, 143
- group files, 140
 - adding character sheets, 141
 - copying text of character sheets to the clipboard, 143
 - creating, 140
 - defined, 298
 - info dialog, 141
 - previewing text printing, 142
 - printing, 142
 - printing through print templates, 142
 - renaming character sheets, 142
 - working with the window, 140
- header, list window, 43
- help keyword, 42, 43, 275
- Help Menu, 49
- horizontal line, 198
- hourglass, 16
- HTML, 9
- IDH_DIEROLLEROPTIONS, 29
- if filter command, 199
- if, procedures, 183
- ifdef, 227
- ifnull, 227
- image library, 52
- images, 50
- incategory, 173
- include files, macro data sheet, 146
- include files, print templates, 117
- include, conversion command, 281
- include, text data sheet, 233
- incrementing options, 71
- indents, 125

- index, 174
- index, quick, 177
- inherited options, 98
- inlist, 263
- insertitem, conversion script command, 281
- inSourceList, 263
- integer function, 174
- integer values, 182
- italic text, 199
- item command script, 219
- item conversion command, 281
- Item Lists, 55
- item name, editing, 73
- item notes, 73
- item options, 70
- item priority, 76
- item reference, getting, 73
- item reference, option, 224
- item reference, option, 223
- item references, 102
- item selection in options, 99
- item selection rules, 57
- itemCheckExpression, 174
- itemCostFunction, 174
- iteminfo, 174
- iteminfo, example, 183
- itemref, automatic item directive, 81
- items, adding to macro data sheets, 148
- items, choosing, 73
- Items, editing, 68
- items, how to add, 134
- items, variable name, 75
- JPEG, 54
- keep cost, 77
- keep value on conversion, 97
- keepnext, 199
- keepnext, conversion command, 275
- keys, accelerator, 291
- keyword, 275
- keyword, help, 42, 43
- landscape, 117
- landscape mode, filters, 200
- left indent, 125, 200
- level prompt for lists in dialogs, 43, 164
- level, character sheet, 211
- line color, 113
- line style, 113
- line up controls, 156
- line, goto, 256
- lines, cell, 113
- lines, filling a rectangle with, 113
- lines, plotting horizontal in text, 198
- lines, plotting in text, 207
- link, text box, 122
- linking text boxes, 111
- list, 282
- list change expression, 44
- list display format, 102
- list format command, 34
- list summary, 36
- list value option, 71
- list, automatic item directive, 80
- list, conversion script command, 282
- list, counting items in, 171
- list, dialog control, 156
- list, number of items, 186
- list, totaling items in, 179
- listbox, dialog control, 272
- listCount, 264
- listDirectory, 174
- listentries, 174
- listiteminfo, 174
- listiteminfo, example, 181
- listMap, 171, 174
- listOptions, 175
- lists in dialogs, 164
- lists, adding and changing, 42
- lists, new, 41
- load script, 47
- loading data sheets, 4
- local variables in expression procedures, 183
- log, 175
- log10, 175
- lookup, 175
- lookup array, 104
- macro argument list values, 152
 - free-form, 153
 - list only, 153
- macro argument value, 152
- macro data sheet, 144, 298
- macro data sheet, comments, 149
- macro data sheet, converting .cds files, 150
- macro data sheet, creating, 144
- macro data sheet, deleting items, 149
- macro data sheet, errors, 151
- macro data sheet, headers, 149
- macro data sheet, information, 146
- macro data sheet, options, 149
- macro data sheet, properties, 152
- macro data sheet, text, 149
- macro data sheet, trying out, 150
- macro data sheet, window, 145
- macro data sheet, finding text, 151
- macro data sheet, go to line, 151
- macro data sheets
 - basic info, 146
 - defines, 147
 - include files, 146
 - loaded data sheets, 147
 - merging, 150
- macros
 - argument type, 231

- local, 147
- repeat, 229
- macros, text data sheets, 226
- main.sct, 242
- maps, 177
- margins, prompting for, 117
- margins, setting for details and text files, 8
- matchbox, conversion command, 274
- matchlist, conversion command, 274
- max, 176
- max. attempts, 46
- maximum value, 176
- measure, 16
- measurement, 8
- menu command keywords, 245
- menu, Edit, 9
- menu, file, 2
- menu, menu, 32
- menu, Tools, 33
- menu, Utilities, 13
- menucommand, conversion script command, 284
- menuitem, 291
- menus, 2
- menus, cascading, 291
- menus, Help, 49
- menus, Modify, 41
- menus, Tools (character sheet), 33
- menus, Tools (data sheet), 39
- menus, Window, 48
- merge conflicts, 128
- merging data sheets, 127
- merging macro data sheets, 150
- message, filter command, 200
- metric, 8, 248
- min, 176
- minimum value, 176
- mod operator, 169
- Modify Menu, 41
- modulo, 169
- move, 33
- moving items, 55
- multiple deletion confirmation, 16
- multiple file filter, 19
- multiplier option, 94
- name file, example, 249
- name files, creating, 249
- name files, opening, 24
- Name Finder, 23
- naming pages, 120
- ncopt, automatic item directive, 79
- need space, 200
- new file, 2
- new game system, 130
- no clear option, 79
- noreq, automatic item directive, 84
- normal page (print template), 120
- not, 169
- Notes Window, 13
- notes, conversion script command, 284
- notes, copying during conversion for items, 270
- notes, item, 73
- notes, referencing in filters, 186
- number, dialog control, 272
- numerical values, 182
- object, editing, 111
- open recent files, 8
- open, automatic item directive, 83
- opening files, 3
- opening sublists, 55
- opensublist, conversion command, 284
- openwindow, 285
- operands, 182
- operators, 169
- opt, automatic item directive, 79
- optAvailable, 176
- optCount, 176
- option, 285
 - requiring specific values, 90
- option check expression, 101, 223
- option display expression, 98
- option formats, checkbox, 225
- option formats, text data sheet, 222
- option inheritance, 72
- option item reference, 223
- option value list, 71
- option value list, text format, 224
- option value type, 99
- option value, expression, 100
- option value, item reference, 100
- option value, list, 100
- option value, plain text, 99
- option, conversion script command, 285
- option, item reference, 224
- option, item references, 102
- optional page (print templates), 120
- options, 92
 - inherited, 98
 - renaming options added to automatic items, 80
 - requirements, 36, 90
- options, addition, 95
- options, adjustment, 95
- options, auxiliary cost, 93
- options, categories, 173
- options, computing costs, 92
- options, conversion script, 285
- options, creating new, 92
- options, data sheet, 127
- options, editing, 96, 97
- options, expression, 93
- options, filtering, 201

- options, fractions, 93
- options, in conversion, 282
- options, incrementing, 71
- options, item editing, 70
- options, item selection, 99
- options, multiplier, 94
- options, percentage, 94
- options, picture search, 53
- options, selecting, 72
- options, text, 96
- optNumberValue, 176
- optpresent, 176
- optqualifier, conversion command, 286
- optTextValue, 176
- optvalue, 176
- OR operator, 169
- ordering controls, 164
- orientation, print template, 117
- original name, 75
- output
 - turning off in filters, 202
- outputFileName, 181
- pack windows, 48
- page name, 120
- page setup, 7
- page setup, text files, 7
- pageNumber, 186
- pages, print template, 119
- pages, print templates, 124
- paragraph format, 14
- paragraph formatting in filters, 200
- paragraph information, 202
- parentiteminfo, 176
- paste, 9
- pauseprogress conversion command, 286
- percentage option, 94
- picture definition, 157
- picture description file format, 251
- picture library, 52
- picture reference, 118
- picture search, 51
- picture search directory, 53
- picture search options, 53
- picture, dialog control, 156
- picture, reading into picture field during
 - conversion, 265
- picture, saving, 54
- picture, writing, 208
- pictures, 50
- pictures, adding to library, 53
- plain text, 203
- playing cards, 28
- PNG, 54
- polygon, 110
- portrait, 117
- portrait mode, 200
- predefined functions, 171
- Preferences Dialog, 15
- print filters, 5
- print preview, 6, 7
- print setup, 7
- print template, 5, 298
- print template information, 117
- print template pages, 119
- print template, bitmap files, 110
- print template, rulers, 125
- print templates, changing, 131
- print templates, creating, 135
- print templates, editing, 109
- print templates, include files, 117
- print templates, modifying pages, 120
- print templates, page types, 120
- printing, 6
- printing blank character sheets, 187
- printing forms, 6
- printing on forms, 117
- printing problems, troubleshooting, 65
- printing, auxiliary files, 30
- printing, multiple files at once, 21
- PrintInheritedOptions, 248
- priority, item, 76
- procedures, 182
- progress dialog, 295
- prompt for margins, 117
- properties, basic, 75
- purgedatasheet, conversion command, 286
- putAssoc, 176
- putMap, 177
- qindex, 177
- quack, 298
- qualifier, automatic item directive, 84
- qualifier, option, 96, 97
- quitting the application, 8
- rand, 177
- random item selection, 57, 76, 263
- random item selection, automatic items, 82, 83
- random number function, 177
- random option value generation, 224
- random pips, 76
- randselect, 263
- rdonly, conversion command, 275
- recalculate, 30
- recent files, 8
- rectangle, 109
- rectangle, rounded, 109, 111
- reference item, 77
- references, character sheet, 185
- referencing dialog fields in filters, 186
- reference items, 219
- regular expressions, 11
- release level, 45
- reload all data sheets, 5

- removeallelements, 177
- removeelement, 177
- repeat macro command, 229
- replaceString, 177
- replaceToken, 178
- requirement rule, 61
- requirements, 35, 88
 - option, 90
 - options, 36
- requirements, automatic satisfaction, 34, 36
- requirements, checking, 15, 34, 265, 288
- requirements, checking only when inserting items, 77
- requirements, disallow satisfying, 78
- requirements, editing, 89
- requirements, not checking, 78
- requirements, showing, 289
- requirements, text format for data sheets, 220
- requirements, value in dialog, 90
- return in procedures, 184
- return, conversion command, 286
- reverse adjustments, 88
- rich edit DLL, 66
- rich text details, 13
- riched20.dll, 66
- rmopt, automatic item directive, 80
- rolldice, 178
- rolling dice, 24
 - specifications, 26
- round, 178
- rounded rectangle, editing, 111
- rounding, 46
- rounding down, 172
- RTF, 9
- rule file, selecting, 62
- rule lines, showing and hiding, 16
- rule set
 - adding, 62
- rule sets, 59
- rule values, 60
- ruled, 113
- rulers, 125
- rules
 - default values, 252
 - file format, 252
- rules, creating new, 60
- rules, editing, 60
- rules, item selection, 57, 59
- ruleset, 186
- run on new, 47
- samename
 - list conversion command, 283
- samepage, 203
- satisfy requirements, automatic, 34, 36
- satisfy, count of items in category, 34
- save all files, 4
- save picture, 54
- saving data sheets, 127
- saving files, 3
- screen layouts, custom, 139
- script debugger, 253
- script, character-sheet loaded, 47
- script, executing when text control changes, 157, 159, 160, 161
- script, reload character sheet, 5
- script-added, 166
- script-added variables, 265
- scriptmode, 286
- scripts
 - breakpoints, 256
- search and replace, text file, 297
- searching items, 10
- see, 219
- selectedIndex, 264
- Selecting Items, 56
- selecting options, 72
- selection rule sets, 59
- selection rules, 59
- selectitem, conversion command, 287
- selexp, automatic item directive, 83
- seloptval, automatic item directive, 80
- send to back, 10
- set conversion command, 287
- Set Copy/Print Filters, 5
- setConfigParam, 178
- setElement, 178
- setGlobal, 178
- setval, automatic item directive, 83
- setVariable, 178
- sheet type, 45, 117, 186, 203
- shortcut, formats, 243
- shortcut BNF, 244
- shortcut file, 47
- shortcuts, 67
- shortcuts,format, 242
- show alphabetically, 56
- show ruler, 124
- showallreq, 289
- showProgress conversion command, 288
- showreq, conversion script, 289
- sizeof, 179
- skin, dialog window, 42
- skip, conversion script, 289
- sort expression, in filters, 204
- sortelements, 179
- sorting
 - filtered file sort order, 204
 - filtered files, 204
- sorting lists, 37
- sortlist, conversion command, 289
- source directory, 17
- split, 179

- sqrt, 179
- square root, 179
- status bar, 15
- strindex, 179
- string formatting function, 172
- string index, 179
- string length, 179
- string replacement function, 177
- string values, 182
- strlen, 179
- sub, conversion script command, 290
- sublist properties, 78
- sublist, in conversion, 283
- sublist, opening and closing, 284
- sublists, adding to macro data sheets, 147
- sublists, data sheets, 127
- sublists, editing, 72
- sublists, opening, 55
- submenu, 291
- substr, 179
- substring, 179
- suppress sublist, 61
- switch, conversion script, 292
- tab order, setting, 164
- tab order, setting in dialog, 10
- tab-delimited macro invocation, 226
- tabs command, 205
- tabs, setting, 125
- tabs, wrapping, 115
- tag, text link, 122
- template file, 45
- template pages, 119
- template, character sheet, 45, 298
- template, print, 298
- templates, editing print, 109
- text
 - getting text from user in filters, 198
- text boxes, linking, 111
- text color, 205
- text control, editing, 160
- text data sheet structure, 211
- text data sheets, macros, 226
- text editor, 16, 296
- text file, saving, 296
- text file, search and replace, 297
- text file, searching, 296
- text files, page setup, 7
- text filters, creating, 135
- text format, 16
- text format, data sheets, 211
- text format, example, 215
- text link, 122
- text menu, 114
- text option, 96
- text output from multiple files, 19
- text properties, 115
- text, dialog control, 155, 271
- text, finding in debugger, 257
- tickcount, 265
- tiling windows, 48
- tips window, 17
- title, conversion script, 293
- toggle option, 100
- token pasting, 98
- tolower, 179
- tools menu, 33, 40
- Tools Menu (character sheet), 33
- Tools Menu (data sheet), 39
- total cost formula, 107
- totalItemValues, 179
- totalItems, 179
- toupper, 180
- translate, 206
- translate characters, filter command, 206
- transparent controls, 160, 161
- transparent edit variable control, 158, 160
- treelist, 272
- unconverted, 283
- unconverted items, 283
- undo, 9
- unique variable, 77
- units, 115
- units of measurement, 16
- update character sheets, 17
- updating character sheet tips, 131
- updating character sheets, 75
- UseAliases, 248
- UseMetric, 248
- Utilities Menu, 13
- value type, options, 99
- var, 293
- variable name, item, 75
- variables, 182
- variables in dialogs, 157
- variables, conversion script, 293
- variables, defining, 166
- variables, local, 183
- variables, script-added, 166, 265
- variables, setting in character sheets, 265
- variteminfo, 180
- vertical lines in text, 207
- view menu, 124
- warning, 293
- warning, conversion script command, 293
- while, 293
- while filter command, 207
- while, procedures, 182
- Window Menu, 48
- windows, overlapping, 48
- windows, packing, 48
- wrapping text to next line, 115
- wraptab filter command, 208

writepicture, 208
zero column width, 115

zref automatic item directive, 81