

G2 ProTools

User's Guide

Version 2015



G2 ProTools User's Guide, Version 2015

December 2015

The information in this publication is subject to change without notice and does not represent a commitment by Gensym Corporation.

Although this software has been extensively tested, Gensym cannot guarantee error-free performance in all applications. Accordingly, use of the software is at the customer's sole risk.

Copyright (c) 1985-2015 Gensym Corporation

All rights reserved. No part of this document may be reproduced, stored in a retrieval system, translated, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Gensym Corporation.

Gensym®, G2®, Optegrity®, and ReThink® are registered trademarks of Gensym Corporation.

NeurOn-Line™, Dynamic Scheduling™, G2 Real-Time Expert System™, G2 ActiveXLink™, G2 BeanBuilder™, G2 CORBALink™, G2 Diagnostic Assistant™, G2 Gateway™, G2 GUIDE™, G2GL™, G2 JavaLink™, G2 ProTools™, GDA™, GFIT™, GSI™, ICP™, Integrity™, and SymCure™ are trademarks of Gensym Corporation.

Telewindows is a trademark or registered trademark of Microsoft Corporation in the United States and/or other countries. Telewindows is used by Gensym Corporation under license from owner.

This software is based in part on the work of the Independent JPEG Group.

Copyright (c) 1998-2002 Daniel Veillard. All Rights Reserved.

SCOR® is a registered trademark of PRTM.

License for Scintilla and SciTE, Copyright 1998-2003 by Neil Hodgson, All Rights Reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>).

All other products or services mentioned in this document are identified by the trademarks or service marks of their respective companies or organizations, and Gensym Corporation disclaims any responsibility for specifying which marks are owned by which companies or organizations.

Gensym Corporation
52 Second Avenue
Burlington, MA 01803 USA
Telephone: (781) 265-7223
Fax: (781) 265-7101

Part Number: DOC105-1200

Contents

Preface vii

About this Guide vii

Audience viii

A Note About the API viii

Organization viii

Conventions ix

Related Documentation xi

Customer Support Services xiii

Chapter 1 Overview of G2 ProTools 1

Introduction 1

Loading G2 ProTools 2

 Adding ProTools to Your Module Hierarchy 2

 Removing ProTools from Your Module Hierarchy 3

Generating the Online Reference Manual 4

 Using the Online Reference Manual 5

Chapter 2 G2 ProTools Palette 7

Introduction 8

G2 ProTools Palette Navigation Buttons 8

 Contents 9

 Control Panel 9

 Configuration 12

 README 14

G2 ProTools Palette Objects 14

 Internal Documentation Messages 14

 Automatic Documentation Objects and Templates 17

 The Accelerator Object 17

 The Cross Hair Object 18

 The Workspace Header Object 18

 The Under-Construction Object 19

User Menu Choices 19

| | |
|---|----|
| Enabling References to Inactive Items | 20 |
| Highlighting the Superior Item of a Workspace | 21 |
| Measuring the Memory of an Item | 21 |
| Making an Item Permanent | 21 |
| Listing the Attributes of an Item | 21 |

Chapter 3 Telewindows Collaboration Tools 23

| | |
|-------------------------------|----|
| Introduction | 23 |
| Window-Specific Message Board | 24 |
| Telewindows Broadcast | 25 |

Chapter 4 Application Analysis Tools 27

| | |
|--------------------------------------|----|
| Introduction | 27 |
| Creating Call Trees | 28 |
| Generating and Extending a Call Tree | 28 |
| Representing Method Calls | 30 |
| Other Special Cases | 31 |
| Using the Tree Shaker | 32 |
| Finding Unused Variables | 33 |
| Using the Profiling Facility | 33 |
| Procedure Pause | 34 |

Chapter 5 Automatic Documentation Tools 35

| | |
|---|----|
| Introduction | 35 |
| Automatic HTML | 36 |
| Documentation Tools on the G2 ProTools Palette | 37 |
| Producing a Reference Manual for Your Application | 37 |
| Customizing Item Selection | 38 |
| Adding Comments to Your Documentation | 40 |
| Adding Comments from Procedures and Methods | 40 |
| Adding Comments from Function Definitions | 42 |
| Adding Comments from Class Definitions | 42 |
| Adding Comments from Other Items | 43 |
| Understanding Documentation Templates | 44 |
| Creating a Structure | 44 |
| Converting a Structure | 45 |
| The protocols-documentation-template Class | 46 |
| The protocols-sequence-template Class | 48 |
| Understanding the Document Generation Process | 50 |

| | | |
|------------------|--|-----------|
| | The protocols-book Class | 50 |
| | The protocols-chapter Class | 50 |
| | The Document Generation Process | 51 |
| | Customizing Your Documentation | 52 |
| | Adding or Removing Individual Items | 52 |
| | Modifying Chapter Attributes | 52 |
| | Removing Unwanted Chapters | 53 |
| | Translating from English to Another Language | 53 |
| | Creating Custom HTML Templates | 54 |
| | Creating a New Chapter | 54 |
| | Changing the Output File Format | 55 |
| Chapter 6 | Formatting and Layout Tools | 57 |
| | Introduction | 57 |
| | Formatting Procedures and Methods | 58 |
| | Changing Objects to Uniform Size | 59 |
| | Arranging the Contents of a Workspace | 60 |
| | Undoing Arrangement | 61 |
| | Changing Workspaces to Uniform Style | 61 |
| Chapter 7 | Shortcuts and Convenience Tools | 63 |
| | Introduction | 63 |
| | Printing Workspaces | 64 |
| | Quick-Launching Procedures and Methods | 65 |
| | Accepting the Defaults | 65 |
| | Viewing Return Arguments | 66 |
| | Defining and Using Accelerators | 67 |
| | Viewing the Description of a Procedure | 67 |
| | Loading External Text Files into G2 | 68 |
| | Index | 69 |

Preface

Describes this guide and the conventions that it uses.

About this Guide **vii**

Audience **viii**

A Note About the API **viii**

Organization **viii**

Conventions **ix**

Related Documentation **xi**

Customer Support Services **xii**



About this Guide

This guide contains complete information about the G2 ProTools utility, and shows you how to use the module at any supported level. This guide:

- Provides an [overview of G2 Protools](#) and its features.
- Describes the [G2 ProTools palette](#).
- Describes how to use these features:
 - [Developer collaboration](#).
 - [Application analysis](#).
 - [Automatic documentation](#).
 - [Formatting and layout](#).
 - [Shortcuts and conveniences](#).

Audience

This guide assumes that you are generally familiar with G2 terminology and practices. If you encounter G2 terms or concepts that you do not understand, see the *G2 Reference Manual*.

A Note About the API

The ProTools API, as described in this guide, is not expected to change significantly in future releases, but exceptions may occur. A detailed description of any changes will accompany the G2 ProTools release that includes them.

Therefore, it is essential that you use ProTools exclusively through its API, as described in this guide. If you bypass the API, you cannot rely on your code to work in the future, since ProTools may change, or in the present, because the code may not correctly manage the internal operations of ProTools.

If ProTools does not seem to provide the capabilities that you need, contact Gensym Customer Support at 1-781-265-7301 (Americas) or +31-71-5682622 (EMEA) for further information.

Organization

This guide contains six chapters:

| | Title | Description |
|---|------------------------------------|--|
| 1 | Overview of G2 ProTools | Describes the features of G2 ProTools and how to load and incorporate G2 ProTools into your development environment. |
| 2 | G2 ProTools Palette | Describes the top-level workspace of ProTools, its cloneable objects, the ProTools control panel, and how to configure the ProTools environment. |
| 3 | Telewindows Collaboration Tools | Describes features supporting cooperative development. |
| 4 | Application Analysis Tools | Describes facilities for application analysis, including constructing call trees, finding dead code, and application profiling. |

| | Title | Description |
|---|-------------------------------|---|
| 5 | Automatic Documentation Tools | Describes the facilities for automatically generating documentation for your application. |
| 6 | Formatting and Layout Tools | Describes tools for formatting items and laying out workspaces in your application. |

Conventions

This guide uses the following typographic conventions and conventions for defining system procedures.

Typographic

| Convention Examples | Description |
|--|---|
| g2-window, g2-window-1, ws-top-level, sys-mod | User-defined and system-defined G2 class names, instance names, workspace names, and module names |
| history-keeping-spec, temperature | User-defined and system-defined G2 attribute names |
| true, 1.234, ok, "Burlington, MA" | G2 attribute values and values specified or viewed through dialogs |
| Main Menu > Start KB Workspace > New Object create subworkspace Start Procedure | G2 menu choices and button labels |
| conclude that the x of y ... | Text of G2 procedures, methods, functions, formulas, and expressions |
| <i>new-argument</i> | User-specified values in syntax descriptions |

| Convention Examples | Description |
|---|---|
| <i>text-string</i> | Return values of G2 procedures and methods in syntax descriptions |
| File Name, OK, Apply, Cancel, General, Edit Scroll Area | GUIDE and native dialog fields, button labels, tabs, and titles |
| File > Save Properties | GMS and native menu choices |
| workspace | Glossary terms |
| <i>c:\Program Files\Gensym\</i> | Windows pathnames |
| <i>/usr/gensym/g2/kbs</i> | UNIX pathnames |
| <i>spreadsh.kb</i> | File names |
| <i>g2 -kb top.kb</i> | Operating system commands |
| <i>public void main() gsi_start</i> | Java, C and all other external code |

Note Syntax conventions are fully described in the *G2 Reference Manual*.

Procedure Signatures

A procedure signature is a complete syntactic summary of a procedure or method. A procedure signature shows values supplied by the user in *italics*, and the value (if any) returned by the procedure underlined. Each value is followed by its type:

```
g2-clone-and-transfer-objects
(list: class item-list, to-workspace: class kb-workspace,
 delta-x: integer, delta-y: integer)
-> transferred-items: g2-list
```

Related Documentation

G2 Core Technology

- *G2 Bundle Release Notes*
- *Getting Started with G2 Tutorials*
- *G2 Reference Manual*
- *G2 Language Reference Card*
- *G2 Developer's Guide*
- *G2 System Procedures Reference Manual*
- *G2 System Procedures Reference Card*
- *G2 Class Reference Manual*
- *Telewindows User's Guide*
- *G2 Gateway Bridge Developer's Guide*

G2 Utilities

- *G2 ProTools User's Guide*
- *G2 Foundation Resources User's Guide*
- *G2 Menu System User's Guide*
- *G2 XL Spreadsheet User's Guide*
- *G2 Dynamic Displays User's Guide*
- *G2 Developer's Interface User's Guide*
- *G2 OnLine Documentation Developer's Guide*
- *G2 OnLine Documentation User's Guide*
- *G2 GUIDE User's Guide*
- *G2 GUIDE/UIIL Procedures Reference Manual*

G2 Developers' Utilities

- *Business Process Management System Users' Guide*
- *Business Rules Management System User's Guide*
- *G2 Reporting Engine User's Guide*
- *G2 Web User's Guide*
- *G2 Event and Data Processing User's Guide*

- *G2 Run-Time Library User's Guide*
- *G2 Event Manager User's Guide*
- *G2 Dialog Utility User's Guide*
- *G2 Data Source Manager User's Guide*
- *G2 Data Point Manager User's Guide*
- *G2 Engineering Unit Conversion User's Guide*
- *G2 Error Handling Foundation User's Guide*
- *G2 Relation Browser User's Guide*

Bridges and External Systems

- *G2 ActiveXLink User's Guide*
- *G2 CORBALink User's Guide*
- *G2 Database Bridge User's Guide*
- *G2-ODBC Bridge Release Notes*
- *G2-Oracle Bridge Release Notes*
- *G2-Sybase Bridge Release Notes*
- *G2 JMail Bridge User's Guide*
- *G2 Java Socket Manager User's Guide*
- *G2 JMSLink User's Guide*
- *G2 OPCLink User's Guide*
- *G2 PI Bridge User's Guide*
- *G2-SNMP Bridge User's Guide*
- *G2 CORBALink User's Guide*
- *G2 WebLink User's Guide*

G2 JavaLink

- *G2 JavaLink User's Guide*
- *G2 DownloadInterfaces User's Guide*
- *G2 Bean Builder User's Guide*

G2 Diagnostic Assistant

- *GDA User's Guide*
- *GDA Reference Manual*
- *GDA API Reference*

Customer Support Services

You can obtain help with this or any Gensym product from Gensym Customer Support. Help is available online, by telephone, by fax, and by email.

To obtain customer support online:

➔ Access G2 HelpLink at www.gensym-support.com.

You will be asked to log in to an existing account or create a new account if necessary. G2 HelpLink allows you to:

- Register your question with Customer Support by creating an Issue.
- Query, link to, and review existing issues.
- Share issues with other users in your group.
- Query for Bugs, Suggestions, and Resolutions.

To obtain customer support by telephone, fax, or email:

➔ Use the following numbers and addresses:

| | Americas | Europe, Middle-East, Africa (EMEA) |
|--------------|--|--|
| Phone | (781) 265-7301 | +31-71-5682622 |
| Fax | (781) 265-7255 | +31-71-5682621 |
| Email | service@gensym.com | service-ema@gensym.com |

Overview of G2 ProTools

Describes the features of G2 ProTools and how to load and incorporate G2 ProTools into your development environment.

Introduction 1

Loading G2 ProTools 2

Generating the Online Reference Manual 4



Introduction

G2 ProTools is a tool for G2 developers that helps you develop, test, debug, document and deploy your G2 applications faster than ever before. G2 ProTools provides a variety of useful tools, including tools for:

- [Developer collaboration.](#)
- [Application analysis.](#)
- [Automatic documentation.](#)
- [Formatting and layout.](#)
- [Shortcuts and conveniences.](#)

You access these features from the [G2 ProTools palette](#). You can easily add and remove G2 ProTools to and from your application, simply by merging or deleting the protools module, `protools.kb`. However, you will get the maximum benefit from ProTools when you make it a permanent component of your environment by adding it to your module hierarchy. This approach gives you instant access to ProTools, and allows you to store ProTools objects in your application's modules.

ProTools can be removed in a single easy step, using the **Delete Module** command.

Using ProTools is highly intuitive. You access most of ProTools' functionality through user menu choices, action buttons, and attribute tables. Or, to gain additional flexibility and power, you can use ProTools' extensive programmatic API. ProTools also provides two useful workspaces, a palette, which is the source of cloneable items, and a control panel, where the most frequently-used controls (such as those for switching operator mode) are located. From the palette, you can easily navigate to all workspaces of interest within ProTools.

Loading G2 ProTools

To load ProTools, start G2 in the normal manner and load your source files. Then, with G2 reset or paused, merge *protools.kb* into your environment, using the **Merge KB** command. You can use ProTools in a development or in a deployment environment.

When you merge ProTools, four modules are loaded:

- `protools`
- `ptroot`
- `sys-mod`
- `uilroot`

Adding ProTools to Your Module Hierarchy

In most cases, you will want to make ProTools a standard part of your environment by adding it to your module hierarchy, so that it is loaded each time you work on your application. The `protools` module should be added to your module hierarchy *under* your modules; that is, your modules should require ProTools. This allows you to put ProTools objects into your modules while maintaining consistent modularity.

Example

Assume you are developing three modules, named `top`, `middle`, and `bottom`, where:

- `top` requires `middle`.
- `middle` requires `bottom`.
- `bottom` requires `uil`, a utility module provided by Gensym.

Since `bottom` is the lowest-level module you are developing, you should make `bottom` require `protools`. You would change the directly required modules of

bottom to uil, protocols. To learn more about module hierarchies, see the *G2 Reference Manual*.

Removing ProTools from Your Module Hierarchy

Once ProTools is added to your module hierarchy, it normally will become a permanent part of your environment. As you use ProTools, you will put instances of objects defined by ProTools in your modules. ProTools objects help to organize and document your development activities.

You must exercise caution when changing your module hierarchy if you have ProTools objects, such as documentation details or developer notes, that you wish to preserve. In particular, if you remove both ProTools modules from your application, all objects defined by ProTools will be automatically deleted.

ProTools provides run-time functionality in a deployment environment. However, you may also want to remove ProTools when your application is ready for deployment. This operation is normally done using a *copy* of your source code, not the source code itself. In this way irreversible operations such as deleting testing procedures, text-stripping, making the application proprietary, and the like, do not affect the source code. Removing ProTools should be part of this *package preparation* procedure.

You may also need to *temporarily* remove ProTools, to enable your application to run on a machine where ProTools is not authorized, without losing your ProTools objects. In this case, you delete the module `protocols`, keeping the module `ptroot`. `ptroot` contains key object definitions for ProTools objects, and does not require authorization. As long as you do not delete `ptroot`, your objects will be retained. Later, `protocols` can be reintroduced.

Caution Deleting the module `ptroot` from your development environment will permanently delete all ProTools objects, such as documentation details and developer notes, from your application.

Temporarily Removing ProTools

To run your application on a machine where ProTools is not authorized, and yet retain ProTools objects, you delete the module `protocols`, leaving the module `ptroot` intact.

To delete the `protocols` module from a copy of your source code:

- 1 Select Main Menu > Miscellany > Delete Module.
- 2 Choose `protocols` from the menu.
- 3 Click the **All** button on the resulting dialog to delete all workspaces associated with ProTools.

- 4 Use Inspect to show the module hierarchy.
- 5 Remove `protools` from the list of required modules, wherever it is required, and replace it with `ptroot`.

In the previous example, you would change the directly required modules of `bottom` to `ptroot`.

Removing ProTools Permanently

To permanently remove ProTools from your source code, simply delete both ProTools modules from your source code. This action will delete all ProTools objects from your source code, including documentation details, developer notes, and any other ProTools items.

If you are preparing a deployment version of your application, be sure to perform this action on a copy of your source code, not the source code itself, because deleting both ProTools modules results in irreversible deletion of all objects defined by ProTools, even if they reside in your modules.

To delete ProTools and associated objects from your application:

- 1 Select Main Menu > Miscellany > Delete Module.
- 2 Choose `protools` from the menu.
- 3 Click the All button on the resulting dialog to delete all workspaces associated with ProTools.
- 4 Repeat the above procedure with the module `ptroot`.
- 5 Use Inspect to show the module hierarchy.
- 6 Remove `protools` and `ptroot` from the list of required modules, wherever they are required.

In the previous example, you would change the directly required modules of `bottom` back to `uil`.

Generating the Online Reference Manual

One of the most powerful features of G2 ProTools is its ability to automatically generate documentation for an application. You use this facility to generate the *G2 ProTools Reference Manual*.

To generate the G2 ProTools Reference Manual:

- 1 Load the ProTools KB and start G2.
- 2 Go to the `protools-top-level` workspace and click the README button.
- 3 Enter the name of the desired directory path into the edit box.

Note This must be an existing directory. If a suitable directory does not exist, first create it in the usual manner on your system.

4 Click the document protocols button.

On most systems, it will take less than a minute to complete the generation of the reference manual.

Using the Online Reference Manual

After you generate the online reference manual, the target directory contains many files of type *.htm* (hypertext markup) and a few of type *.dat*. Start your browser and load the *.htm* files for viewing in the usual manner for your system.

Key Files in the Online Reference Manual

The following table describes the contents of important files in the online reference manual:

| File Name | Contents |
|---------------------|---|
| <i>titlepag.htm</i> | The title page of the manual, providing links to table of contents, index, and first chapter. |
| <i>booktoc.htm</i> | The table of contents for the manual, providing links to all chapters. |
| <i>bookix.htm</i> | An alphabetical index for the manual, providing links to all documented classes, procedures, methods, user menu choices, and relations. |
| <i>book.htm</i> | The entire reference manual in a single HTML-formatted file. |

You may want to bookmark one or more of these files in your browser, to provide an easy entry point into the Reference Manual:

Obtaining a Printed Reference Manual

Using your browser, you can print any files in the Reference Manual, one at a time. However, because the reference manual is comprised of many files, it is very time consuming to individually load and print each file. To obtain the entire reference manual on paper, use the *book.htm* file. It contains the entire reference manual in a single file, and thus can be printed with a single print command.

G2 ProTools Palette

Describes the top-level workspace of ProTools, its cloneable objects, the ProTools control panel, and how to configure the ProTools environment.

Introduction **8**

G2 ProTools Palette Navigation Buttons **8**

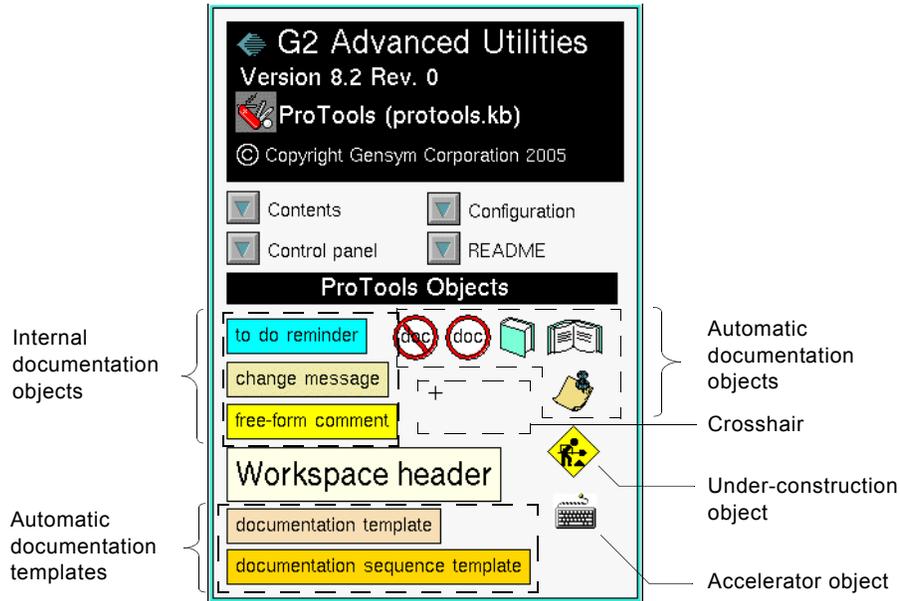
G2 ProTools Palette Objects **14**

User Menu Choices **19**



Introduction

The top-level workspace of G2 ProTools serves as the starting point for most G2 ProTools activities:



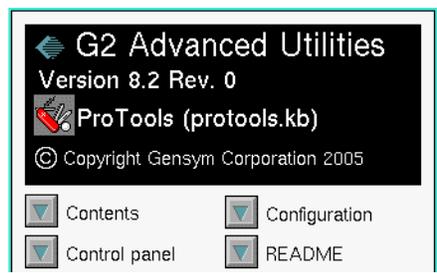
This workspace, named `protools-top-level`, is available from the G2 Main Menu:

Main Menu > Get Workspace > `protools-top-level`

The various items on this workspace are described in the following sections.

G2 ProTools Palette Navigation Buttons

This section explains the navigation buttons that appear in the top half of the G2 ProTools palette:



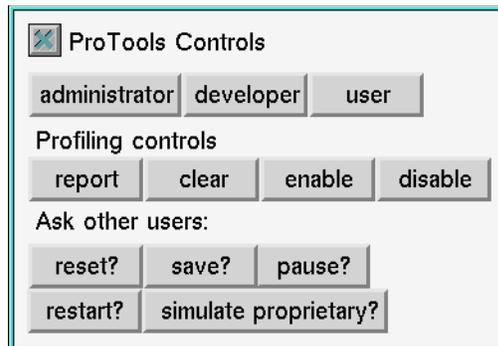
Contents

This navigation button allows you to navigate to the various functional parts of G2 ProTools. On the subworkspace of this button, you will see another set of buttons that correspond the chapters of this User's Guide, for example, Telewindow Collaboration Tools, Application Analysis Tools.

Each topic on the contents workspace contains one or more subworkspaces containing demos, explanations, and various G2 objects. To best familiarize yourself with G2 ProTools, you should explore the workspaces under the contents button in some detail. Often, you will find examples you can copy or emulate.

Control Panel

The subworkspace of this button is a control panel that contains several useful controls that you may use frequently during development.



You can keep this workspace visible in a corner of your G2 window for easy access. The buttons on the control panel allow you to:

- Change user mode.
- Run the profile.
- Communicate with other developers.

Changing the User Mode

Clicking the administrator, developer, or user button in the first row on the control panel changes the user mode to the mode specified on the button. The effect of these buttons is the same as entering Ctrl y and typing the user mode in the G2 Login dialog.

Using the Profiling Controls

G2 provides a useful profiling capability, which you can use to optimize the performance of your application. G2 ProTools offers a lightweight user interface to this facility.

Note G2 provides an alternative, full-feature user interface to the profiling facility called Profiler. Shipped with G2, the Profiler KB provides more data display options than the interface provided with G2 ProTools. The advantage of the G2 ProTools profiler interface is its lightweight character and independence from other utility module requirements, so it can be merged into any application without generating version conflicts.

To start profiling:

→ Click the enable button.

G2 records the number of calls, the average time, and the total time spent evaluating each executable item in your KB.

To stop profiling:

→ Click the disable button.

To show a summary report of the data collected during the profiling run:

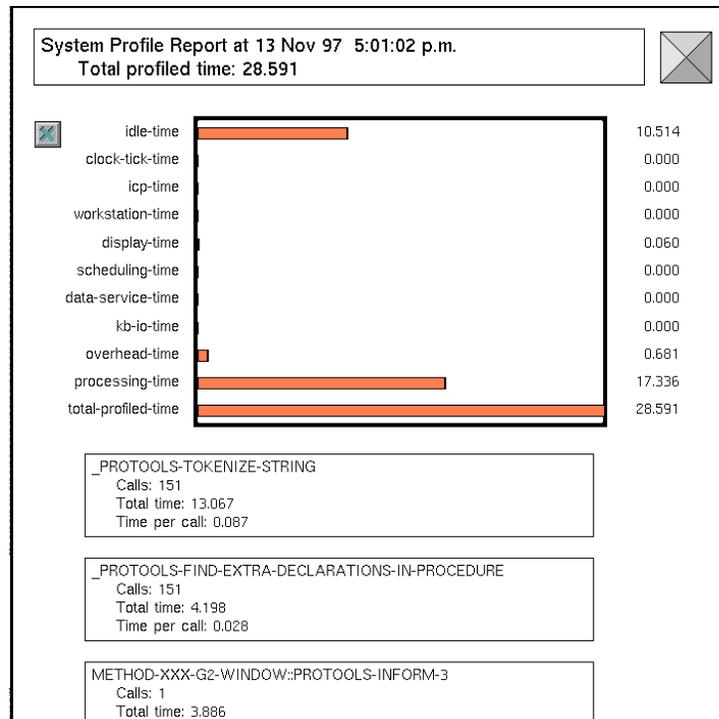
→ Click the report button.

To clear the data from the previous profiling run:

→ Click the clear button.

The Profile Report

Clicking the report button displays a workspace similar to the this:



This workspace contains a bar chart showing the breakdown of time spent in various types of activities, such as time for processing workstation events, time for G2 processing, idle time, and the like. Below, there is a list of report items showing the time spent in certain executable items in the KB. This list is ordered with the executable items accounting for the most total time listed first. By default, only items that have executed for a total time of more than 0.01 seconds are shown. You can specify a different cut-off time by editing the action of the report button.

More information on the activities within any profiled item can be obtained by selecting the report item. This additional information includes the number of times certain statements within the executable item have been evaluated.

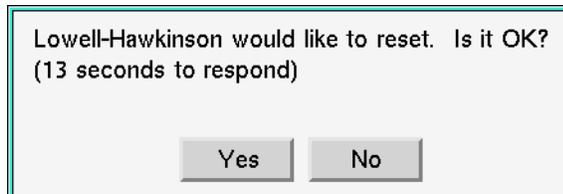
For more information on using the profiler, including the list of actions and statements that G2 profiles, see the *G2 Reference Manual*.

Communicating with Other Users

The last set of buttons on the control panel allow you to communicate with other users logged into the same G2 through Telewindows. When you develop as a team with multiple users logged into the same G2, you often need to communicate, particularly before taking actions that affect all users, such as

pausing or resetting G2, saving modules, entering simulate proprietary mode, and the like. G2 ProTools provides a simple dialog mechanism for communicating with your co-workers when you need to take an action that might affect their work.

For example, you need to reset G2. To ask the other users if this is OK, you click the **reset** button. G2 ProTools displays a dialog similar to the following on each of the Telewindows connected to the G2.



G2 automatically resets as soon as all users respond yes.

If there is no response within 15 seconds (default time-out) on a certain window, the **Yes** button is selected automatically on that window.

The time remaining before the default action occurs is displayed on each dialog. Any user clicking **No** before the time-out cancels the reset, and you receive a message to that effect. The message contains the user name of the user who responded negatively. You can change the time-out by editing the **reset** button.

Several of these buttons require that you take the action manually once all users have responded. Only the reset and pause actions are taken automatically.

For more information on this facility, see [Telewindows Broadcast](#).

Configuration

You can configure three aspects of the operation of G2 ProTools:

- User menu choices
- Accelerators
- The control panel

Configuring User Menu Choices

G2 ProTools offers more than a dozen user menu choices that appear on built-in classes such as workspaces and procedures. You may want to hide the menu choices not used, to reduce menu clutter.

To configure which menu choices you want to show:

- 1 Click the navigation button labelled User Menu Choices on the configuration workspace.

A workspace of user menu choices appears.

- 2 Do one of the following:

- Check the box next to the menu choice to hide it.
- Uncheck the box next to the menu choice to show it.

These user menu choice settings last only as long as your current G2 session. If you want these settings next time you load G2, you must save the protools module.

Installing Accelerators

You may find it convenient to use accelerators for certain actions that you use frequently in G2, such as hiding and showing workspaces, navigating to the subworkspaces of items, and the like. G2 ProTools makes it easy to define your own customized set of accelerators, and also provides a small set of built-in accelerators.

G2 ProTools defines these built-in accelerators:

| Action | Accelerator Shortcut | Applicable Class |
|---------------------------|-------------------------|------------------|
| go to subworkspace | Ctrl + any mouse button | item |
| highlight superior object | Alt + any mouse button | kb-workspace |
| hide workspace | Esc(ape) key | kb-workspace |

These built-in accelerators are inactive, by default.

To activate any of these built-in accelerators:

- 1 Click the Accelerators button on the Configuration workspace.
- 2 Click the button turn on default accelerators.

You can also turn the accelerators on or off individually, using the menu choices turn accelerator on and turn accelerator off.

Because accelerators are defined per G2 rather than on a per Telewindow basis, all Telewindows users must agree on a standard set of accelerators. Also, accelerators are not active in administrator mode, since they are implemented via item configurations.

For more information on defining your own accelerators, see [The Accelerator Object](#).

Configuring the Control Panel

The final configuration for G2 ProTools enables you to determine whether the control panel appears when G2 is started. By default, the control panel appears in the lower left corner of the screen each time G2 is started. If you do not want the control panel to appear, uncheck the check box on the subworkspace of navigation button labelled **Control Panel** on the configuration workspace. Also on this workspace, there is a button that enables you to start the G2 Profiler immediately when G2 is started, so you can profile start-up activities.

README

The subworkspace of the README navigation button contains last-minute information relating to the release, if any. In addition, the workspace contains instructions and an action button for generating files for the *G2 ProTools Reference Manual*. For details on how to generate the online reference manual, see [Generating the Online Reference Manual](#).

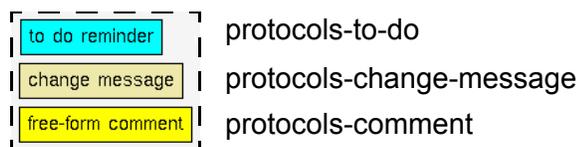
G2 ProTools Palette Objects

This section describes the objects that can be cloned from the G2 ProTools palette, that appear in the lower half of the palette workspace.

Internal Documentation Messages

G2 ProTools provides message objects that help you document your application and its development history. Three classes of messages are provided: `protocols-to-do`, `protocols-change-message`, and `protocols-comment`, along with the extensible parent class, `protocols-authored-message`.

You use these messages by placing them in your application, nearby a target object.



For example, if you have made modifications to a method, you might place a change message on the workspace where the method is located. The messages can be used freely anywhere in your application, since they can be removed from your application by deleting the ProTools module, a step usually taken during package preparation and described in [Removing ProTools Permanently](#).

At any time, you can use Inspect to locate and summarize the information contained in internal documentation objects. For example, using Inspect, you can make a table of all to-do reminders with high priority, or changes made by a particular individual.

These messages are extremely useful in the context of team development. They provide documentation and development history to anyone involved with code development or maintenance. To get maximum benefit, it is important that you establish standards for the use of internal documentation messages. Establishing use standards ensures that all team members use the messages consistently and that the information provided is thorough and accurate.

Caution If you use free texts (or borderless free texts) to annotate your application, it will be difficult to find and categorize your annotations. Furthermore, you will have difficulty isolating these annotations if you want to remove them during package preparation.

Creating Internal Documentation Messages

To add one of these messages to your application,

- ➔ Clone it from the G2 ProTools palette by clicking the mouse, which creates a new item, and then clicking again near the application item you wish to document.

When you place the message on a workspace for the first time, you will be prompted to create a relation with the nearest object. This relation serves to associate a message with its subject. If, subsequently, you move the subject object to a new position or workspace, the associated message will automatically follow.

If you decline to create the relation when the message is first created, you can use the menu choice `link-to-nearest-item` to create the relation at any time. Menu choices are also provided to unlink the message, and to navigate from the message to the subject item. Also, when an internal documentation message is created, its author and creation-date attributes are automatically filled out.

To reduce clutter in your application, internal documentation messages have a useful feature that allows you to hide the full text of a multi-line message, making it take up less workspace area. To shrink or enlarge a message, use the `hide-full-message` and `show-full-message` menu choices.

The following table summarizes the use of each class of internal documentation message:

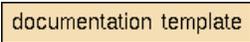
| Message | Description |
|-------------------|---|
| To-do reminders | You can use to-do reminders to mark incomplete places in the application, where more development work is required. You can also use these objects to mark places where bug fixes are required. Attributes for to-do reminders include the person responsible for carrying out the work, the priority of the work, the target software version for which the work is required, and the target completion date. |
| Change message | <p>Use the attributes of the change message to explain the nature of and reason for changes you make to your application. Change messages complement G2's change logging feature, which records the state of the subject item before and after the change but not the reason for the change. Also change logging alone cannot explain architectural changes to your application, that might affect more than one item.</p> <p>The attributes of a change message include the reason for the change, the description of the change, an optional cross reference (perhaps to a bug report number), and a description of the possible testing fan-out of the change. These fields accept any text.</p> |
| Free-form comment | You can use comment messages to add explanatory text anywhere you deem appropriate in your application. There are no attributes in a comment message besides author and creation date. |

For full details of the attributes, methods, relations and menu choices associated with these classes, see the *G2 ProTools Reference Manual*.

You may also create new classes of messages to meet your specific needs by subclassing `protools-authored-message`.

Automatic Documentation Objects and Templates

The G2 ProTools palette includes several objects related to the automatic documentation facility. Automatic documentation objects include the classes:

| Item | Class |
|--|---------------------------------|
|  | protools-book |
|  | protools-chapter |
|  | protools-documentation-details |
|  | protools-exclusion-marker |
|  | protools-inclusion-marker |
|  | protools-documentation-template |
|  | protools-sequence-template |

These objects are used in preparing your application for the automatic documentation generator. For information on how to use these objects, see [Automatic Documentation Tools](#).

The Accelerator Object



The `protools-accelerator` object represents the definition of an accelerator, a keyboard and/or mouse gesture that carries out an action. You use instances of this class to define your own accelerators. G2 ProTools also defines a small number of built-in accelerators, as described in [Installing Accelerators](#), whose use is optional.

To define your own accelerator:

- 1 Clone an accelerator object from the palette and place it on any application workspace.
- 2 Specify a native action or the label of a user menu choice that will be activated by the accelerator in the `protools-action-or-menu-choice` attribute.

Native actions are those actions you can enter into an item configuration following “implies”, such as `go-to-subworkspace`.

- 3 Specify an applicable class in the `protocols-applicable-class` attribute and the keyboard or mouse gesture that will activate the accelerator in the `protocols-keystroke` attribute.

To activate the accelerator:

→ Choose turn accelerator on from the accelerator menu.

Note Remember, even when the accelerator is on, it is only active when the user mode is not administrator.

To deactivate the accelerator:

→ Choose turn accelerator off from the accelerator menu.

Note that accelerators cannot be defined on a window-by-window basis, so all Telewindows users share the same accelerators.

The Cross Hair Object

+ X -155
Y 168

Use the cross hair object to determine the coordinates of any point on a workspace. Simply clone a cross hair object from the palette, and place it in anywhere on a workspace. The x and y coordinates are given in workspace units.

The Workspace Header Object

Workspace header

If you name too many workspaces in your application, the Get Workspace menu becomes cluttered and difficult to use. Therefore, it is good practice to minimize the number of named workspaces. The workspace header object provides another way to label a workspace, without requiring that the workspace have a name.

To add a workspace header to a workspace:

- 1 Clone the header from the palette and place it on any workspace in your application.
- 2 Change the text of the header to a suitable label for the workspace.

Like the internal documentation messages, workspace header messages can be easily located, using Inspect, and removed from the application during package preparation.

The Under-Construction Object



The under-construction object has a subworkspace which gives you a place to put unfinished work, tests, and other items that may not be ready to deliver with your application. It is also a handy way to find areas of your application that are incomplete.

Because the definition of this class is contained in the `protools` module, all under-construction objects and their subworkspaces are deleted when you delete the `protools` module. Thus, when you prepare your application for delivery by deleting the `protools` module, all items on the subworkspaces of under-construction objects are automatically deleted. This gives you a fast, reliable way to clean up your application prior to deployment.

User Menu Choices

G2 ProTools defines several menu choices that are available on high-level classes such as `kb-workspace` or `procedure`. The following table summarizes these menu choices:

| Menu Choice | Function | See |
|--|---|---|
| <code>format-ws-hierarchy</code> | Copies the style of the workspace to its subworkspaces | Changing Workspaces to Uniform Style |
| <code>enable-references-to-inactive-items</code> | Enables or disables the items from referring to disabled items | Enabling References to Inactive Items |
| <code>disable-reference-to-inactive-items</code> | | |
| <code>highlight-superior-item</code> | Highlights the item superior to a workspace | Highlighting the Superior Item of a Workspace |
| <code>measure-memory</code> | Reports on the memory required for any item | Measuring the Memory of an Item |
| <code>make-permanent</code> | Makes an item permanent | Making an Item Permanent |
| <code>parse-for-unused-variables</code> | Checks a procedure for local variables that have been declared but not used | Finding Unused Variables |
| <code>reformat-procedure</code> | Standardizes the formatting of procedures | Formatting Procedures and Methods |

| Menu Choice | Function | See |
|------------------------|--|--|
| get-signature | Reports on the input and return arguments of a procedure | Viewing the Description of a Procedure |
| list-attributes | Gives a complete list of the attributes of any item, with their current values | Listing the Attributes of an Item |
| launch-procedure | Generates a dialog that enables you to start a procedure or method | Quick-Launching Procedures and Methods |
| generate-call-tree | Creates a call tree rooted at the selected executable item | Creating Call Trees |
| arrange-workspace | Neatly arranges the items on a workspace | Arranging the Contents of a Workspace |
| undo-arrange-workspace | | |
| print-workspace | Launches a print dialog that gives you easy access to printing | Printing Workspaces |

The following sections discuss the menu choices that are not described elsewhere in this document.

Enabling References to Inactive Items

The `enable-references-to-inactive-items` and `disable-references-to-inactive-items` menu choices provide a quick way to toggle the evaluation attributes of an item to control whether it can reference inactive items. These menu choices are available on any item with evaluation attributes, such as rules, procedures, and user menu choices.

References to inactive items are normally disabled in G2. For example, if you loop programmatically over all tanks in your application, the loop will ignore tanks that are disabled or located on inactive workspaces. However, when you enable references to inactive items, the executable item can “see” items in your application that have been disabled or are located in an inactive workspace hierarchy. The tank loop, for example, will include all tanks, regardless of their activation status.

For more information about this feature of G2, see the *G2 Reference Manual*.

Highlighting the Superior Item of a Workspace

The `highlight-superior-item` menu choice enables you to find the item superior to a given workspace. When you select this menu choice, the workspace containing the superior item is shown, with a flashing “bull’s-eye” on the superior item.

This menu choice appears only if the workspace has a superior item.

Measuring the Memory of an Item

The `measure-memory` menu choice, shown on any item, is a convenient interface to the G2 system procedure, `g2-measure-memory`. When you select this menu choice, the message board displays results of `g2-measure-memory` for the selected item. For details on interpreting the results of this procedure, see the *G2 System Procedures Reference Manual*.

Making an Item Permanent

The `make-permanent` menu choice, shown on any item, allows you make an item permanent. It is a convenient shortcut to executing the expression, `make item permanent`.

Listing the Attributes of an Item

The `list-attributes` menu choice provides a comprehensive list of all the attributes of an item, and their current values. When you select this menu choice, the message board displays this list, including all the “virtual attributes” of the item, such as the workspace position, `item-width`, and `containing-module`. Therefore, the information provided is more comprehensive than showing the item’s table.

Telewindows Collaboration Tools

Describes features supporting cooperative development.

Introduction 23

Window-Specific Message Board 24

Telewindows Broadcast 25



Introduction

G2 Telewindows allows multiple users to work simultaneously on a single application hosted by one G2, in a process called *cooperative development*. Cooperative development is particularly useful for rapid prototyping and other activities where multiple developers can work simultaneously without excessive interference with each other.

G2 ProTools provides two useful tools to support cooperative development:

- The [window-specific message board](#) is similar to G2's built-in message board, but it is dedicated to a single user. Messages posted on the dedicated message board do not appear on other windows.
- The [Telewindows broadcast](#) facility helps multiple Telewindows users communicate. This makes it easier to work without the need for constant phone calls (or shouting) between team members.

Window-Specific Message Board

G2's built-in message board is common to all clients in a multiple-client environment. This works well for some purposes, particularly, whenever a message must be posted on all clients. However, in other applications messages should be designated for specific clients. G2's built-in message board does not provide this functionality.

Specifically, during debugging, you will often insert "inform the operator" statements in procedures, so you can trace the flow of control and print key variables. During cooperative development, you do not want these messages appearing on other windows, because they interfere with the activities of other developers.

G2 ProTools provides a method, `protools-inform`, which implements a client-specific "inform the operator". This method automatically creates a message board for the target window and displays the desired text on the message board. Optionally, you can provide a display duration and designate a source item for the message. Providing a source item enables you to navigate back to the source of the message, using the `go-to-originating-item` menu choice.

Note In administrator mode, both the G2 `go-to-message-origin` and ProTools `go-to-originating-item` are visible. The G2 menu choice navigates back to the `protools-inform` method. The ProTools `go-to-originating-item` navigates back to the originating item specified in the call to `protools-inform`, if one was specified.

In most respects, you can treat the window-specific message board exactly like the built-in G2 message board. You can set the maximum number of messages, the initial width and height of the message board, and the spacing between messages, using the Message Board Parameters system table.

An example of using `protools-inform` is given in the ProTools KB:

Protools-Top-Level > Contents > Telewindows Collaboration Tools >
Window-Specific Message Board

See the *G2 ProTools Reference Manual* for more details on `protools-inform`.

Telewindows Broadcast

The Telewindows broadcast facility allows you to notify users who are connected to the same G2 when you plan to take an action that might affect them, such as resetting G2. The simplest way to use Telewindows broadcast is by using the “Ask other users” buttons on the ProTools Control Panel. These buttons broadcast requests for:

- Resetting G2.
- Saving the knowledge base.
- Pausing G2.
- Restarting G2.
- Entering simulate proprietary mode.

Clicking one of these buttons displays a confirmation dialog on all windows, including your window, in case you want to change your mind. Other users can agree to, or cancel, the proposed action by clicking a button on the confirmation dialog. For more information about using these buttons, see [Communicating with Other Users](#).

You can also customize Telewindows broadcast by writing your own procedures. To do this, you use the `protools-confirm-action-with-other-users` procedure, and provide your own launching and callback procedures. You may provide a time-out that will allow the action to take place, if one or more users have not replied within the time-out period.

For details on using `protools-confirm-action-with-other-users`, see the *G2 ProTools Reference Manual*.

Application Analysis Tools

Describes facilities for application analysis, including constructing call trees, finding dead code, and application profiling.

| | |
|------------------------------|----|
| Introduction | 27 |
| Creating Call Trees | 28 |
| Using the Tree Shaker | 32 |
| Finding Unused Variables | 33 |
| Using the Profiling Facility | 33 |
| Procedure Pause | 34 |



Introduction

G2 ProTools contains a number of tools that help you understand the structure and performance characteristics of complex applications. These facilities include:

- A [call tree facility](#) that displays the relationships between the executable items in your application, allowing you to see control paths through procedures, methods, rules, action buttons and user menu choices.
- A [tree shaker](#) locates procedures that have no callers, and are therefore potentially dead code.
- A [parser](#) that finds variables in procedures that are declared but not used.

- A [profiling utility](#) that helps you determine where computation time is spent in your application.
- A [procedure pause facility](#) that suspends computational threads, allowing you to examine the structure of objects in the midst of procedural processing.

These facilities are described in the following sections.

Creating Call Trees

A call tree is a graph that depicts calling relationships between executable items. In G2, executable items include:

- Procedures
- Methods
- Rules
- Action buttons
- User menu choices

The ProTools call tree facility generates graphs whose nodes represent these executable items, and whose connections show call and start actions.

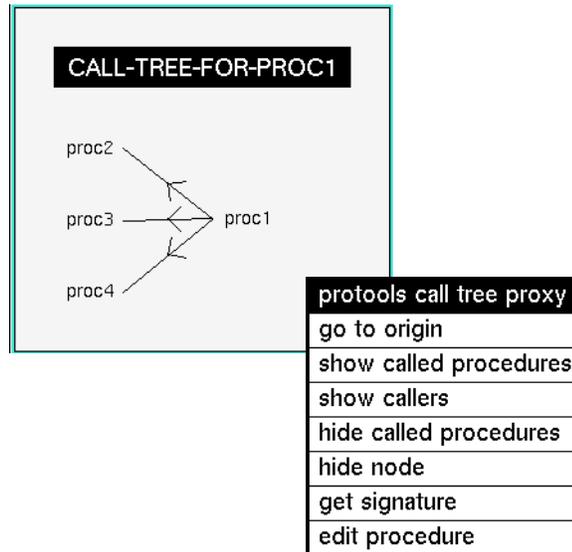
Of the five types of executable items, only procedures and methods can be called (or started) by other executable items. Therefore, only procedures and methods can be internal nodes in a call tree, with both input and output connections. Rules, action buttons, and user menu choices serve as entry points to the call tree, and only have output connections.

Generating and Extending a Call Tree

To generate a call tree:

- ➔ Go to the executable item you wish to trace and select the **generate-call-tree** menu choice.

A workspace similar to the following appears:



This example indicates that `proc1` calls (or starts) `proc2`, `proc3`, and `proc4`. Notice that the direction of the arrow indicates the calling direction.

Each executable item is represented by a proxy item that you can select to obtain a menu of further choices. These menu choices are:

| Menu Choice | Description |
|------------------------|--|
| go to origin | Navigates to the executable item represented by the selected node |
| show called procedures | Expands a call tree node to show all procedures called by the selected node |
| show callers | Expands a call tree node to show all executable items calling the selected node |
| hide called procedures | Hides all the nodes connected at an output of the selected node |
| hide node | Hides the selected node |
| get signature | Describes the calling arguments of the selected node (procedures and methods only) |
| edit procedure | Edits the procedure represented by the selected node (procedures and methods only) |

You can navigate from any node in the call tree to the original item, using the `go-to-origin` menu choice. You can extend the call tree from any visible node, using the `show-called-procedures` and `show-callers` menu choices. As you extend the call tree, you can graphically arrange the call tree by dragging the nodes to new positions. If the call tree becomes too cluttered, you can hide branches of the call tree or individual nodes, using the `hide-called-procedures` and `hide-node` menu choices.

As a convenience, you can view the calling signature of any procedure or method, using the `get-signature` menu choice. You can even edit any procedure or method directly from the call tree, using the `edit-procedure` menu choice.

Representing Method Calls

When you call a method, G2 determines which specific method to call by looking at the class of the first argument and the number of arguments, picking the most specific method that has the proper number of arguments. The process of choosing a specific method from all methods with a matching name is called **run-time dispatching**.

When developing a call tree, the class of the item involved in run-time dispatching is not precisely known. Consider, for example, the following procedure:

```
a-procedure(Obj: class my-object)
begin
  call my-method(Obj);
end
```

Until the procedure is executed, we do not know the exact class of `Obj`. It may be the class `my-object`, or it may be a subclass of `my-object`. Nonetheless, run-time dispatch of the call to `my-method` is constrained by the following considerations:

- Instances of `my-method` with two or more arguments will not be called.
- If there is an instance of `my-method` with one argument defined on class `my-object` or on a subclass of `my-object`, it may be called, depending on the class of `Obj`.
- If there is an instance of `my-method` with one argument defined on class `my-object`, and there is another instance of `my-method` defined on a parent class of `my-object`, the method defined on the parent class will not be invoked.
- If there is no method defined on the class `my-object`, but there are methods defined on both subclasses and superclasses of `my-object`, then any of the subclass methods and the most specific superclass method can be called.

ProTools uses these considerations to determine which instances of a method can be called. However, even with these constraints, it may be impossible to resolve which method will be called at run time. In this case, ProTools represents the possible method calls using *red* connection arrows. A red connection in the call

tree indicates that the method at the output end of the connection *might* be called, depending on the class of items at run time.

Call Next Method

The call next method statement is special type of method call that invokes another instance of the current method; specifically, a method of the same name with the same number of arguments, defined on the most specific superior class of the current method. Call next method statements do not depend on run-time dispatch, assuming that the class hierarchy does not change at run time. ProTools includes the methods invoked by call next method statements with black (normal) connections.

Qualified-Name Method Calls

Methods can also be invoked by including a specific class name, such as my-class::my-method. Since this is tantamount to a calling a procedure by name, there is no run-time dispatch. ProTools includes such qualified-name method calls, using black (normal) connection arrows.

Other Special Cases

Text-Stripped Procedures

The call tree facility must have access to the text of the executable items to determine what procedures are called by the item. Therefore, ProTools cannot develop nodes corresponding to text-stripped items nor find calls that originate from text-stripped executable items.

Recursive Calls

A **recursive call** is when a procedure or method calls itself. Recursive calls are depicted in the call tree as a connection that originates and terminates at the same node.

Indirectly Called Procedures

In G2, there are several ways to invoke a procedure without mentioning its name directly. Consider the following examples:

```

another-procedure()
P: class procedure = the procedure nearest to this procedure;
begin
  call P();
end

yet-another-procedure()
S: symbol;
T1: text = text-parameter-1;
T2: text = the callback-suffix of button-2;

```

```
begin
  S = symbol("[T1]-[T2]")
  call the procedure named by S();
end
```

In these cases, ProTools cannot determine which procedure is called, since making this determination would require running the application. ProTools uses a node labelled `determined-at-run-time` to represent indirectly invoked procedure calls.

Comments, Texts, and Inactive Items

ProTools ignores call statements that have been commented out, occur inside text string, or occur inside inactive items.

Using the Tree Shaker

Over the course of time, in developing a large KB, you may write many procedures that are not actually used in the finished application. Sometimes, you might forget to delete old, unused procedures that are no longer required in your application. To help you find this “dead code,” ProTools provides a facility to locate unused procedures and methods, called the tree shaker.

The simplest way to use the tree shaker is by invoking the procedure, `protools-report-uncalled-procedures`. This procedure analyzes one or more modules at a time, and posts on the message board a list of procedures that are not explicitly called by any other procedure, action button, user menu choice, rule, or method in the application. To get an item list of uncalled procedures, use `protools-find-uncalled-procedures`. See the *G2 ProTools Reference Manual* for details of these procedures.

When considering methods, if it is possible that a method *might* be called, depending on the class of run-time arguments, the method is not included in the list of uncalled procedures. In other words, if a method would be included in any call tree, via either a red or black connection, it will not be included in the list of uncalled procedures.

You should *not* automatically assume that procedures identified by the tree shaker are “dead code.” The procedures identified by the tree shaker include all procedures that are *not specifically called by name* in the application. However, some or all these procedures might be called indirectly in statements, such as, `call the procedure named by S`, where `S` is a symbol.

Additionally, since the tree shaker works by parsing the text of executable items, procedures called by text-stripped items might be included in the tree shaker’s list of uncalled procedures. Furthermore, procedures that make up the public interface (API) to a module might not be called in your application, but still might be required for completeness in other applications of your module. So, before you delete a procedure, make sure it is genuinely “dead code.”

Caution Be extremely careful before deleting any item located by the tree shaker. Procedures that are not explicitly called still might be invoked indirectly or called by a text-stripped procedure.

Finding Unused Variables

This facility helps you clean up the declarations section of procedures and methods. Specifically, it identifies local variables you have declared, but do not reference in the body of the procedure.

To find extra declarations in a specific procedure, use the `Protools` procedure, `protools-find-extra-declarations-in-procedure`. This procedure is conveniently launched through the user menu choice, `parse-for-unused-variables`, which is available on any procedure or method. The menu choice posts the list of declared but unused variables on the message board.

To locate extra declarations in all procedures in a module or application at once, use the `protools-check-module-for-extra-declarations` or `protools-check-all-modules-for-extra-declarations` procedures. The results are written to the file named by `protools-output-file-name`. The easiest way to use these procedures is by choosing the action buttons located under `Protools-Top-Level > Contents > Application Analysis Tools > Find Unused Variables`.

Using the Profiling Facility

The ProTools profiling facility provides a lightweight user interface to G2's Profiler. To implement profiling, use the action buttons on the ProTools Controls workspace, or use the following procedure calls:

| To... | Call this procedure... |
|---------------------------|---|
| Begin profiling | <code>g2-enable-profiling()</code> |
| Turn off profiling | <code>g2-disable-profiling()</code> |
| Display profiling results | <code>protools-generate-profile-report</code> (float, g2-window) |
| Clear profiling data | <code>g2-clear-profile()</code> |

When calling `protools-generate-profile-report`, you can specify a positive cutoff time as the first argument. If the total profiled time in any executable item is below this cutoff, it will be excluded from the list of profiled items.

For further discussion of this facility, see [Using the Profiling Controls](#).

Procedure Pause

ProTools provides a breakpoint function, `protools-pause`, that can help you debug your application. To use this facility, you insert a call to `protools-pause` at the desired location in your code. When the procedure is called, the thread is suspended, and ProTools displays a dialog with a button that allows you to continue the thread when you are ready to do so. During the pause, you can inspect the state of objects in your application.

This facility optionally displays an item on the pause dialog. You can use this feature to gain access to an item or display a value that is locally-defined within the calling procedure. For example, if you create a transient list in a procedure, you can use `protools-pause` to display the list and thus examine its contents during the execution of the procedure.

The arguments to `protools-pause` include:

- A text argument, which appears on the pause dialog.
- An item-or-false argument, which represents an optional item to be shown on the dialog.
- A client argument.

For more information about this procedure, see the *G2 ProTools Reference Manual*.

Automatic Documentation Tools

Describes the facilities for automatically generating documentation for your application.

| | |
|---|----|
| Introduction | 35 |
| Producing a Reference Manual for Your Application | 37 |
| Adding Comments to Your Documentation | 40 |
| Understanding Documentation Templates | 44 |
| Understanding the Document Generation Process | 50 |
| Customizing Your Documentation | 52 |



Introduction

One of the most powerful features of G2 ProTools is its ability to automatically generate documentation for an application. Using the automatic documentation facility, you can almost instantly generate a reference manual that is guaranteed to be accurate and up-to-date.

This system is based on G2's ability to describe its own knowledge base components. There are many expressions and procedures to determine, among other things, the attributes of an item, its methods, its inheritance, and its menu choices. In addition, by parsing the text of a procedure or method, you can determine its arguments, return types, and similar information. Imagine putting all this information into formatted text files, and you get the basic idea of G2 ProTools' automatic documentation facility.

You can generate automatic documentation at several levels. The simplest approach is to use the pre-defined book and chapter templates to generate a reference manual for your application. Very little customization or effort is needed at this level. At a more advanced level, you can modify the existing templates to change the format of the generated documentation. At this level you need to understand how G2 ProTools uses templates, but again, only limited programming is required. Advanced users can use the documentation facility and define entirely new formats and procedures to generate highly-customized documentation. This level requires G2 programming skills and, potentially, substantial investment of time and effort.

The automatic documentation generation tools include:

- [Producing a Reference Manual for Your Application](#) shows you how to generate a basic reference manual for your application, using the built-in HTML template.
- [Adding Comments to Your Documentation](#) shows you how to enrich the automatically generated documentation by adding comments to your application.
- [Understanding Documentation Templates](#) explains the use and customization of templates.
- [Understanding the Document Generation Process](#) shows you how to use classes and procedures to generate automatic documentation.
- [Customizing Your Documentation](#) discusses how to produce complete customized documents by using the full power of G2 ProTools.

Automatic HTML

An important aspect of the automatic documentation facility is **hypertext markup language** (HTML). HTML is a simple and universal text formatting language that can be interpreted by almost any Internet browser. By default, G2 ProTools produces HTML output. This format has the advantage of being readable on any platform with low-cost or free software. In addition, using this format, you can create context-sensitive links from your application to the documentation, using G2 On-Line Documentation (GOLD). For more information, see the *G2 OnLine Documentation Developer's Guide*.

Documentation Tools on the G2 ProTools Palette

When you generate automatic documentation, you work with several items from the G2 ProTools palette. The following table identifies the relevant items:

| Item | | Class |
|--|---------|---------------------------------|
|  | Book | protools-book |
|  | Chapter | protools-chapter |
|  | Details | protools-documentation-details |
|  | Exclude | protools-exclusion-marker |
|  | Include | protools-inclusion-marker |
|  | | protools-documentation-template |
|  | | protools-sequence-template |

Producing a Reference Manual for Your Application

Note G2 ProTools cannot document non-modular KBs.

To produce a reference manual for your application:

- 1 Clone a book object from the ProTools palette, and place it on a workspace in your application.
- 2 Specify the attributes of the book object.

For best results, you should fill out all unspecified attributes of the book, without changing default values. At minimum, you must provide the following information:

| Required Attribute Name | Description |
|---|--|
| protools-directory | The name of a directory to contain the documentation files. This directory must exist; if it does not, you must create it. |
| protools-title-information, full-product-name | The full name for your application, for example, "My G2 Application" |
| protools-title-information, product-acronym | A short string with no spaces, an abbreviation or acronym for your product, for example, "myapp" |
| protools-target-modules | A list of the modules comprising the application, as a sequence of symbols. To specify a single module, enter it as a sequence of one symbol, for example, <code>sequence(the symbol my-module)</code> . |

Caution Do not attempt to store two or more different reference manuals in the same directory. The file names will conflict.

- 3 Choose **create chapters** from the book menu to generate several `protools-chapter` objects, which are placed on the workspace near the book.
If you do not see this menu choice, make sure the `protools-chapters` attribute of the book has the empty sequence value, `sequence()`.
- 4 Choose **generate book** from the book menu to generate the documentation files and write them out to the specified directory.

You should now be able to open the documentation files using any browser. For a description of key files in your documentation, see [Using the Online Reference Manual](#).

To create another reference manual with different contents, simply clone another new book and repeat these steps, but be sure to designate a different target directory.

Customizing Item Selection

When you generate documentation from a given book, ProTools considers only the items contained in modules you specified in the `protools-target-modules` attribute. From this pool of items, when you use the default reference manual template, ProTools will document all public instances of the following classes:

- Class definition
- Procedure and method
- User menu choice
- Relation
- Function-definition

Each class appears in a separate chapter of the reference manual, and each instance appears as a separate HTML file.

The following sections describe how you can tailor the selection process to add or remove items from the default set of items documented by ProTools.

Adding Specific Items to the Documentation

If you want to add a specific item to the documentation, mark it with a Protools inclusion marker. Inclusion markers apply hierarchically to the item and all its subworkspaces, unless overridden at a lower level by an exclusion marker.

To add specific items to the documentation:

- 1 Clone an inclusion marker from the ProTools palette.
- 2 With the marker attached to the mouse, click the item you wish to include in the documentation.
- 3 When prompted to create an association between the marker and the item, click OK.

The items that you designate in this fashion appear in the last chapter of the reference manual, "Miscellaneous Items," unless they are instances of the classes documented in earlier chapters.

To reverse the designation, simply delete the marker.

Adding Private Items to the Documentation

By default, ProTools only documents public items. ProTools determines if an item is public by examining the item's name (or label, in the case of a user menu choice). If the name begins with an underscore character, the item is considered private, and the item is not documented. Otherwise, it is considered public. For more information on the public-private naming convention, see the *G2 User's Guide*.

ProTools uses the same criterion when documenting the attributes of objects. If the attribute name begins with an underscore character, by default, the attribute is not documented.

To include all private items in your documentation:

- Set the `protocols-document-private-items` parameter to `true`, before you generate the documentation.

To include only selected private items in your documentation:

- Attach inclusion markers to each item.

Suppressing Documentation of Selected Items

If you want to prevent a certain item from appearing in the documentation, mark the item with a ProTools exclusion marker. Exclusion markers apply hierarchically to the item and all its subworkspaces, unless overridden at a lower level by an inclusion marker.

To exclude the documentation of specific items:

- 1 Clone an exclusion marker from the ProTools palette.
- 2 With the marker attached to the mouse, click the item you wish to exclude in the documentation.
- 3 When prompted to create an association between the marker and the item, click OK.

Adding or Suppressing Attribute Documentation

By default, ProTools documents all user-defined public attributes of class definitions. If you want to add or remove specific attributes from the documentation of a class definition, you must use a documentation details object to specify which attributes you want to add or suppress. See [Adding Comments to Your Documentation](#) for details.

Adding Comments to Your Documentation

The generated documentation includes basic factual information about your application. This factual information includes the names of classes, the names of attributes, their type and initial values, the signature of procedures and methods, and the like. While necessary, the factual information is not always sufficient to communicate how the items are intended to be used, particularly if item and attribute names are less than descriptive. This section discusses how to add comments to your application, so they subsequently appear in your documentation.

Adding Comments from Procedures and Methods

In G2, you can add comments to procedures and methods by delimiting text in curly brackets `{}`. ProTools follows a convention to determine which of your

comments in procedures and methods should be included in the generated documentation. This convention is based on the location of the comments within the procedure or method:

- A comment appearing directly after any calling or return argument in the argument list is interpreted as the description of the preceding argument, and included in the documentation.
- A comment appearing directly after the procedure signature is interpreted as the overall description of the procedure, and included in the documentation.
- All other comments are ignored.

If you plan to generate automatic documentation, it makes sense to create comments following these conventions, each time you create a public procedure. This will greatly reduce the amount of effort needed to generate high-quality documentation, when the application is finished.

Example

Consider the text of the method, `protools-inform`, which includes five comments:

```

      ③           ②           ①
      |           |           |
      |           |           |
  ④ | protools-inform (Win: class g2-window {the window where the message is to be
      | presented}, T: value {a value presented as text to the user}) = (item-or-value
      | {the message posted to the message board, if the message is posted on a
      | window-specific message board, or false})
      |
      | {This procedure is used to post a message on a window-specific message
      | board. Use this procedure if there is no originating item, and you want the
      | message to remain on the message board indefinitely. Otherwise, use the
      | four-argument version of this method.}
      |
      | M: item-or-value;
      | begin
      |   {Simply call the full method with zero duration and no originating item}
      |   M = call protools-inform (Win, T, 0, false);
      |   return M;
      | end
      |
      | ⑤
  
```

Comments 1, 2, and 3 describe particular input and return arguments, and are located inside the calling signature. Comment 4, located after the signature but before its local declarations, describes the overall purpose of the method. Comment 5 is located in the body of the code.

Here is the resulting documentation page:

g2-window::protocols-inform

Synopsis

```
g2-window::protocols-inform
  (win: g2-window , t: value )
  ->   item-or-value
```

| Argument | Description |
|------------|---|
| <i>win</i> | the window where the message is to be presented |
| <i>t</i> | a value presented as text to the user |

Comment 1

Comment 2

| Return Value | Description |
|----------------------|--|
| <i>item-or-value</i> | the message posted to the message board, if the message is posted on a window-specific message board, or false |

Comment 3

Description

This procedure is used to post a message on a window-specific message board. Use this procedure if there is no originating item, and you want the message to remain on the message board indefinitely. Otherwise, use the four-argument version of this method.

Comment 4

[Prev](#) | [Next](#) | [Start of Chapter](#) | [End of Chapter](#) | [Contents](#) | [Index](#) | [Comments](#) | (1 out of 63)

Comments 1 - 3 from the procedure signature appear as descriptions of their respective arguments. Comment 4 appears as the description of the method. Comment 5 does not appear.

Adding Comments from Function Definitions

The same conventions are followed for function definitions, except that there are no return arguments in function definitions. Specifically, a comment appearing directly after an input argument is interpreted as documenting the preceding input argument. A comment appearing after the argument signature but before the “=” sign in the function definition is interpreted as the overall description of the function.

Adding Comments from Class Definitions



In G2, there is only a limited ability to add comments to class definitions. For example, you cannot add curly-bracket comments to the `class-specific-attributes` property of a class definition. To overcome this limitation, ProTools provides a class of “helper” objects to contain extra information about class definitions and other non-procedural items. The class is `protocols-documentation-details`, represented by the details icon on the Protocols palette. Although you can add a documentation details object to any instance of any class, it is particularly useful for adding information to class definitions.

To add documentation comments to any class definition in your application:

- 1 Clone a details object from the ProTools palette.
- 2 With the object still attached to the mouse, click on the class definition you want to document.
- 3 When prompted, associate the documentation details object with the class definition.
- 4 Enter a general description of the class in the `protocols-description` attribute of the details object.
- 5 In the `protocols-attribute-description` attribute there is a sequence of structures, one structure for each attribute. The structure has the following subattributes:
 - `attribute-name` (symbol)
 - `public` (T/F)
 - `allowable-values` (text)
 - `description` (text)

Enter a description of the attribute into the description field of each structure. You may also specify the allowable values of each attribute, or change the public/private designation of the attribute.

When ProTools documents a class that has an associated details object, it includes the additional information in the documentation.

If G2 is running and you modify a definition that already has a documentation details object, for example, by adding, removing, or renaming an attribute, the documentation details object automatically updates. If you change a definition while G2 is reset, the documentation details objects do not automatically update. In that case, you can manually cause a documentation details object to update, using the `update-documentation-details` menu choice, which appears on the class definition associated with the documentation details object.

Adding Comments from Other Items

You can also use a documentation details object to add documentation to other items, for example, parameters, user menu choices, and instances of your user-defined classes. Use the same procedure as described previous section to add information to the documentation details object.

Understanding Documentation Templates

This section explains how ProTools produces formatted text descriptions of items in your application. This information will enable you to customize the format of documentation pages. This section assumes you are comfortable with the basic procedures for generating documentation in the default format, including creating books, chapters, and adding comments and documentation details to your application.

For each documented item, there are two main steps to produce a documentation page:

- 1 [Create a G2 structure](#) that summarizes the key information about the target object.
- 2 [Convert the structure](#) into text using documentation templates.

Creating a Structure

ProTools provides several procedures to create the structure in Step 1. These include:

- `protools-get-procedure-signature`
- `protools-get-function-signature`
- `protools-describe-class`
- `protools-describe-umc`
- `protools-describe-relation`
- `protools-describe-instance`

Without going into detail, each of these procedures returns a structure that describes the important properties the target object, from a documentation point of view. This structure can sometimes be large, depending on the complexity of

the target item. For example, `protools-get-procedure-signature` returns this information about itself:

```
structure (
  procedure-name: the symbol protools-get-procedure-signature,
  description: "This procedure analyzes the header of procedures and
returns a structure containing the name, description, inputs, and
outputs.",
  inputs: sequence (
    structure (
      argument-name: the symbol proc,
      type: the symbol procedure,
      description: "the procedure to be analyzed"),

    structure (
      argument-name: the symbol client,
      type: the symbol object,
      description: "the client for this call")),
  outputs: sequence (
    structure (
      type: the symbol structure,
      description: "structure containing details of the calling arguments
of Proc"))))
```

This structure contains all the information that eventually appears on the documentation page for this procedure. For more details concerning the procedures that summarize item properties as structures, see the *G2 ProTools Reference Manual*.

Converting a Structure

The second step to produce the documentation page is to convert a structure, like the one above, into a formatted text. To do this, ProTools uses a class of message items of type `documentation-template`. A template defines the layout and appearance of a type of documentation page, or part of a page, omitting the specific information unique to instances of the page. For example, you might have a template (or group of templates) defining the appearance of a documentation page that describes a procedure. The template includes both fixed text, and places where specific data from the structure describing the procedure are to be inserted. We sometimes refer to the process of applying specific data to a template as **expanding** the template. The main procedure used to expand templates is `protools-expand-template`. See the *G2 ProTools Reference Manual* for more information about this procedure.

To produce formatted output, templates include formatting symbols that are specific to a given text-formatting language. Since ProTools' default file format is HTML, it's built-in templates contain HTML formatting commands. The commands, referred to as **tags** in HTML parlance, are delimited by angle brackets `<>`. For example, a level-1 heading in HTML is delimited by the pair of tags `<h1>` and `</h1>`. Knowledge of HTML formatting, or other formatting languages, is

necessary to create your own templates. A description of text formatting languages is beyond the scope of the current document.

The protocols-documentation-template Class

The primary class used to define documentation page formats is `protocols-documentation-template`. The template consists of bracketed terms, indicating places for text substitution, and non-bracketed text. When the template is expanded using the information in a structure, the non-bracketed text is not changed. Bracketed terms are substituted.

The use of square brackets to indicate substitution arguments is similar to native G2. G2 supports expressions such as:

inform the operator that “The current time is [the current real time]”.

When evaluated, [the current real time] is substituted with the value of this expression, resulting in a meaningful text.

In documentation templates, bracketed terms can have four different formats:

| Substitution Form | Action |
|---|--|
| <i>[attribute-name]</i> | Substitutes the value of the named attribute. |
| <i>[attribute-name, format]</i> where format is <i>title-case</i> , <i>upper-case</i> , or <i>lower-case</i> | Substitutes the value of the named attribute, applying the given format. |
| <i>[template-name]</i> | Substitutes the text resulting from expanding the named template, using the original data structure. |
| <i>[attribute-name, template-name]</i> | Substitutes the text resulting from expanding the named template, using the named attribute of the original data structure, which must be a sequence or structure. |

The easiest way to understand how these forms are used is through a series of examples.

Example 1

This example illustrates the first two substitution forms, *[attribute-name]* and *[attribute-name, format]*.

| Data | Template | Result |
|--|--|---|
| structure(company-name: "gensym corporation", founding-year: 1987) | "[company-name, title-case] was founded in [founding-year]." | Gensym Corporation was founded in 1987. |

Example 2

The third substitution form, *[template-name]*, allows you to nest templates. Suppose template-1 = "[company-name, title-case] was founded in [founding-year]." This template can be used in another template, by giving its name in brackets, as follows:

| Data | Template | Result |
|---|--------------------------------|--|
| structure(company-name: "gensym corporation", founding-year: 1987, CEO: "Lowell Hawkinson") | [template-1] Its CEO is [CEO]. | Gensym Corporation was founded in 1987. Its CEO is Lowell Hawkinson. |

Note that when template-1 is expanded, the data from the original data structure is used.

Example 3

The last substitution form, *[attribute-name, template-name]*, allows you to expand attributes of the original data structure that are themselves structures or sequences. Suppose our data is as follows:

```
gensym-information-1 =
  structure(
    company-data:
      structure(company-name: "gensym corporation",
        founding-year: 1987),
    product-data:
      structure(product-name: "G2", version-number: 7.0))
```

Because the two attributes, `company-data` and `product-data`, are structures, the `[attribute-name, template-name]` form must be used to access their data. To expand the product data, we define `template-2`, in addition to the already defined `template-1`.

```
template-2 =
    "Its main product is [product-name] Version [version-number]"
```

`Template-1` is then used to expand the company data, and `template-2` expands the `product-data`.

To combine the results, we define another template consisting of two `[attribute-name, template-name]` references, as follows:

| Data | Template | Result |
|-----------------------------------|--|--|
| <code>gensym-information-1</code> | <code>[company-data, template-1]</code> <code>[product-data, template-2]</code> | Gensym Corporation was founded in 1987. Its main product is G2 Version 7.0. |

The protocols-sequence-template Class

Sequence templates are a special kind of documentation template used to expand sequences. You use a sequence template when you want to generate text repeatedly, for each member of a sequence. The template is applied to each member of the sequence.

To achieve proper formatting of the sequence, the sequence template also defines a separator to print between elements of the sequence, and texts that can appear before and after the sequence. The template also defines an optional text that is printed if the sequence is empty.

Example

Suppose we define another attribute of the data structure, defining Gensym's field offices, as follows:

```
gensym-information-2 =
  structure(
    company-data:
      structure(company-name: "gensym corporation", founding-year:
        1987),
    product-data:
      structure(product-name: "G2", version-number: 7.0),
    field-offices:
      sequence(
        structure(location: "atlanta"),
        structure(location:"houston"),
        structure(location: "chicago"))))
```

To format the field-offices attribute, we must use a sequence template. The sequence template, `template-3`, is defined with the following attributes:

| Attribute | Value |
|--|-------------------------------------|
| <code>protocols-initial-text</code> | "Its field offices are located in:" |
| <code>protocols-separator</code> | " " |
| <code>protocols-final-text</code> | ." |
| <code>protocols-text-if-sequence-is-empty</code> | "" |
| <code>text</code> | [location, title-case] |

Now we can expand `gensym-information-2` as follows:

| Data | Template | Result |
|---------------------------------|---|--|
| <code>gensym-information</code> | [company-data, template-1] [product-data, template-2] [field-offices, template-3] | Gensym Corporation was founded in 1987. Its main product is G2 Version 7.0. Its field offices are located in: Atlanta, Houston, Chicago. |

Note that each element of the sequence must be a structure, so that the substitutions in `template-3` work with specific named attributes. The data structure cannot be given as `sequence("Atlanta", "Houston", "Chicago")`, because this would deprive `template-3` of a named handle to the location of the offices.

Also note that the phrase "Its field offices are located in:" is included as the initial text attribute of the sequence template. This allows the text to be conditional on the existence of elements in the sequence, and assures that it is printed only once, not repeated for each element of the sequence. Because the `protocols-text-if-sequence-is-empty` attribute is "" (the empty text), nothing would be printed by the bracketed term `[field-offices, template-3]` if the sequence of field offices contained no elements.

Understanding the Document Generation Process

This section discusses the classes and procedures used in the automatic documentation generation facility.

The protools-book Class

When you generate documentation for your application, you use an instance of the class `protools-book`. This is the central object in the documentation generation process. The book object gives the basic information needed to generate the book, including a listing of the chapters of the book, the title information for the book, a list of target modules, various templates to use for the book, the directory in which to write the resulting files, and other information. For detailed information about the `protools-book` class and its attributes, see the *G2 ProTools Reference Manual*.

You clone a book object from the palette as the first step in documenting your application. See [Producing a Reference Manual for Your Application](#), for information on how to create default-format documentation for your application.

If you want to customize your documentation, you may change certain default attributes of the book object, in particular, the default templates. For more information on customizing your documentation, see [Customizing Your Documentation](#).

The protools-chapter Class

You use an instance of the class `protools-chapters` to represent each chapter in your book. The chapter objects are associated with a book if they are named in the `protools-chapters` attribute of the book. The chapter object has attributes specifying the chapter title, giving a short description of the chapter, and a file prefix for the output files associated with the chapter.

Each chapter in the book uses one page layout. You cannot combine different page layouts into one chapter. Generally, this means you will document items of the same class in one chapter, so they can all share the same page layout. The chapter object specifies the templates used in this page layout.

In addition, each chapter names three procedures. The procedures and their functions are:

| Procedure | Function |
|-----------------------------|---|
| collection procedure | Returns a list of items to be documented in this chapter. |
| description procedure | Returns a structure describing the item to be documented. |
| cross-referencing procedure | Adds cross-reference tags to the page of text generated by expanding the description using the named item expansion template. |

For more details on the attributes of chapter objects and their attributes, see the *G2 ProTools Reference Manual*.

The Document Generation Process

The main procedure for generating documentation is `protools-generate-book`. This procedure accepts a book as an argument, and uses the attributes of the book to generate the documentation files.

To generate documentation:

- 1 Write out the title page of the book using the title page template specified by the book and the information given in the title information attribute of the book.
- 2 Write out the front matter file, which usually contains copyright information, disclaimers, trademark disclosures, and the like, using the front matter template specified by the book and the title information.
- 3 Generate an index for the book, as follows:
 - a Iterate through each chapter associated with the book, and calling the chapter's collection procedure,
 - b Associate a file name with each item,
 - c Sort the list of items alphabetically,
 - d Print the index using the index templates specified in the book object.

Steps a-c are carried out by `protools-generate-index`, and step d by `protools-print-index`.
- 4 Generate the *bookstb.dat* and *bookhid.dat* files, used only if you use G2 On-Line Documentation (GOLD) to create context-sensitive help links.

- 5 For each chapter, perform the following steps, using `protools-generate-chapter`:
 - a Write a title page for the chapter, including its table of contents.
 - b Call the collection procedure for the chapter, which returns a list of items for the chapter.
 - c For each item in the chapter, create a documentation page by calling its description procedure, expanding the resulting structure using the item expansion template designated by the chapter object, and then adding cross-references to the resulting text, using the cross-referencing procedure.
- 6 Create a book file containing all files appended together.

If you want to customize the documentation, you should review and understand the procedures involved in the documentation generation process. The text of the key procedures is provided in the ProTools module.

Customizing Your Documentation

There are several ways to customize the output of the automatic documentation generator. Before attempting to customize your documentation, it is important to be familiar with the automatic documentation system. For reference, most of the source code for the documentation system is available in the ProTools KB.

Adding or Removing Individual Items

You can easily add or remove individual items from the documentation. For details, see [Customizing Item Selection](#).

Modifying Chapter Attributes

Another simple customization is to change the title or short description of any chapter.

To modify the title or description of any chapter:

- ➔ Edit the `protools-chapter-title` or `protools-chapter-short-description` attributes of your chapter objects.

Removing Unwanted Chapters

The another simple customization involves removing unwanted chapters from the default reference manual.

To remove an unwanted chapter:

- ➔ Remove the name of the unwanted chapter from the `protools-chapters` sequence attribute of the book object.

Translating from English to Another Language

The templates that are provided produce English-language documentation. However, it is relatively simple to create a set of templates in another language. Of course, item and attribute names and embedded comments will appear in the language used inside the application, regardless of the language in the templates. Translating the templates directly affects the (relatively few) English-language words that appear in the templates.

To translate the default templates into a non-English language:

- 1 Clone the entire set of default templates, located under the ProTools palette:
 Contents > Automatic Documentation Tools > Reference Manual Template
- 2 Name each cloned template by extending its original name with the new language.
 For example, when you clone `protools-title-page`, you should name the new template `protools-title-page-german`.
- 3 Edit each template, and replace each embedded template names with the corresponding extended names. You can easily identify template names because they all begin with the prefix `protools-`.
- 4 Translate English-language words appearing in the documentation templates.
 If you use special character sets, enter them as you normally would in G2 messages or free texts.
- 5 In your chapter objects, edit the `protools-template-name`, inserting the modified template names.

- 6 (Special character sets only). Determine what character encoding you will use in your browser, and create a `text-conversion-style` whose `external-character-set` attribute is set to the same character encoding.

To create a `text-conversion-style`:

- a Select **New Definition > text-conversion-style** from the KB Workspace menu. Give a unique name to the object.
 - b In the table of the text conversion style object, edit the attribute, `external-character-set`, to correspond to the setting of your browser.
- 7 (Special character sets only). Enter the name of the text conversion style object, as a text, in the `protools-text-converter` attribute of your book object.

You can then produce your documentation in the normal manner, and the generated HTML files will appear in the desired language.

Creating Custom HTML Templates

With relatively minor effort, you can modify the appearance of your HTML output files. To do this, you create your own documentation templates, and then refer to your templates in the attributes of the chapter objects.

To create your own HTML templates:

- 1 Clone the desired template class (`protools-documentation-template` or `protools-sequence-template`) from the ProTools palette, and place it in your application.
- 2 Change the text of the template to reflect the desired format, as explained in [Understanding Documentation Templates](#).

To help you get started, you can copy any of the existing default templates. If you are directly substituting your own template for one of the default templates, be sure to refer to the same attribute names in your template.
- 3 In the corresponding chapter object, change the `protools-template-name` attribute to reflect the name of your new template.

When you are ready, you can generate the documentation using your new templates, using the `generate-book` menu choice.

Creating a New Chapter

You may find that the default reference manual format does not document certain items of interest in your application. The “quick fix” for this problem is to mark the missing items with inclusion markers, as discussed in [Adding Specific Items to the Documentation](#). The additional items will be documented in the Miscellaneous Items chapter. However, the generic format of the Miscellaneous

Items chapter may not be sufficient. In that case, you might want to create a separate chapter devoted to a certain class of your application items.

Defining a new chapter requires that you are comfortable programming in G2.

To define a new chapter:

- 1 Define an item collection procedure that will return an item-list containing the items to be documented in this chapter. You may want to emulate existing collection procedures, such as `protools-collect-class-definitions`.
- 2 Define a description procedure that summarizes the salient properties of the target object as a single structure. Usually, this involves iterating over the attributes of the object and placing the value of the attribute into the structure. For an example of a structure created to describe procedures, see [Creating a Structure](#). The contents of the returned structure are arbitrary, but there are several constraints on the form of the structure:
 - The structure cannot include items.
 - If there are sequences in the structure, the elements of the sequence must be structures.
- 3 Create a template, or set of nested templates, for formatting the items in your chapter. The template you create must be consistent with the structure generated by your description procedure. For more information on specifying templates, see [Understanding Documentation Templates](#).
- 4 Create a new chapter object, and fill in its attributes to reflect the names of your collection and description procedures, and your primary template.
- 5 Add the chapter object to the `chapters` attribute of your book.
- 6 Generate the documentation, using the `generate-book` menu choice.

Changing the Output File Format

By default, ProTools produces documentation in hypertext markup language (HTML). You may want output in another text-formatting language, such as RTF or Postscript. ProTools does not provide documentation templates for those file formats.

To change the output file format, you need to replace all HTML tags in the default templates with non-HTML equivalents. It is unlikely that the replacement will be a simple one-to-one replacement. A project to create an entirely new set of templates may require a substantial investment of time, from (at least) several days, to possibly a week or more.

The following steps summarize what you need to do to change the output file format:

- 1** For each default template, create a new template in the desired output file format. Use the same attribute names as in the original template.
- 2** Where templates refer to each other, be sure to change the names to refer to the new template set names.
- 3** In each of your chapter objects, change the `protools-template-name` attribute to refer to your new top-level template names.
- 4** Create procedures that generate the title page, index, and table of contents (both for each chapter and the overall book) in your output file format.

To do this, you should carefully study the code for `protools-generate-book`, and the procedures that it calls, and adapt them as necessary for your output format. It is likely that the overall flow of control of these procedures can be preserved, but some details may need to be changed. These details are highly dependent on the output file format.

Formatting and Layout Tools

Describes tools for formatting items and laying out workspaces in your application.

| | |
|---------------------------------------|----|
| Introduction | 57 |
| Formatting Procedures and Methods | 58 |
| Changing Objects to Uniform Size | 59 |
| Arranging the Contents of a Workspace | 60 |
| Changing Workspaces to Uniform Style | 61 |



Introduction

Whether you are building a large or small application, you are likely to spend a significant proportion of your time working on formatting and layout. Good layout and formatting not only makes your application more attractive, but also easier to understand and navigate. As a result, a well-formatted application may be easier to maintain and debug.

G2 ProTools provides several tools that help you format your application. These tools can help you:

- [Standardize the format of procedures and methods](#) for the best readability.
- [Standardize the size of items](#), enabling neat and compact workspace layouts.
- [Align and distribute objects](#) neatly on a workspace.
- [Standardize workspace styles](#).

Formatting Procedures and Methods

These guidelines help you format the text of procedures and methods for maximum readability and consistency. Although there is not one “right” way to format a procedure, Gensym does suggest a particular combination of naming, capitalization, and indentation as a “standard” for G2 procedures and methods. These instructions follow this standard. For more information on procedure formatting standards, see the *G2 Developer’s Guide*.

For the most part, reformatting only affects the “white space” (spaces, tabs, end-of-lines) in your procedure. G2 ProTools does not alter the text of your procedure. It does, however, change symbols to upper case.

Caution Reformatting causes the target procedure to recompile, which in rare cases, might cause items to become incomplete in applications where stability declarations have been used.

To format an existing procedure:

- ➔ Use the reformat-procedure menu choice on any procedure whose status is OK.

Here is an example of a (nonsense) procedure before and after reformatting:

Before reformatting

```

reformat-me()
I, J, K, N: integer;
V: class g2-variable;
Error: class error;
S, EN: symbol; ET: text;
begin
for I = 0 to N - 1 do J = I;
for K = N down to 0 do
begin
end on error (Error) inform the operator that
"something bad happened";
end;
end;
if I >= 10 then J = 88 else if
I <= 0 then begin J = 99; K = 100; end; end;
repeat exit if J /= N; J = J + 1; end;
case (S) of high:
begin
collect data (timing out after 10 seconds) V
= V; if timeout then inform the operator that
"timed out"; end; end;
medium: begin I = I + 1; end;
low: S = the symbol otherwise;
otherwise: begin end; end;
end on error (EN, ET) inform the operator that
"bombed!"; end

```

After reformatting

```

reformat-me ()
I, J, K, N: integer;
V: class g2-variable;
Error: class error;
S, EN: symbol;
ET: text;
begin
for I = 0 to N - 1 do
J = I;
for K = N down to 0 do
begin
end on error (Error)
inform the operator that "something
bad happened";
end;
end;
if I >= 10 then J = 88
else if I <= 0 then begin
J = 99;
K = 100;
end;
end;
repeat
exit if J /= N;
J = J + 1;
end;
case (S) of
high: begin
collect data (timing out after 10
seconds)
V = V;
if timeout then inform the operator
that "timed out";
end;
end;
medium: begin
I = I + 1;
end;
low: S = the symbol OTHERWISE;
otherwise: begin
end;
end;
end on error (EN, ET)
inform the operator that "bombed!";
end

```

To apply procedure reformatting to many procedure at once, you can iterate over the target procedures, and call the procedure, `protools-format-procedure`, for each target procedure.

Changing Objects to Uniform Size

You can change the size of items on a workspace to a uniform size. This is usually the first step in making a neat, compact arrangement of items on a workspace. For example, if you have many procedures on a workspace, you might want to reduce their size so they fit into a smaller space.

In addition to changing the sizes of items, you can also standardize the placement of name boxes and attribute displays.

To standardize the size, attribute displays, and name boxes of items on a workspace:

- 1 Create a “standard item” with the desired size and placement of name box and attribute displays, if any.
- 2 Transfer the standard item to the target workspace, if it is not already there.
- 3 Choose change others to match on the standard item.

All other items of the same class as the standard item will change to match the size and layout of the standard item. This action only affects items on the same workspace as standard item.

To change all items of a given class to a standard size and layout:

- 1 Create a “standard item” with the desired size and placement of name box and attribute displays, if any.
- 2 Programmatically, do the following:
 - Loop over the workspaces in your application, and transfer the standard item in turn to each workspace, and call the procedure, `protools-change-layout-of-items-to-match`.
 - When finished with each workspace, transfer the standard item off the workspace.

You must make the standard object transient during this operation.

Arranging the Contents of a Workspace

You can neatly arrange objects on a workspace. While preserving the general layout of your workspace, you can also neatly align and evenly distribute columns and rows of items.

To arrange objects on a workspace:

- 1 Put items near their desired locations.
- 2 Choose `arrange workspace` from the workspace menu.

Because this procedure always strives to preserve the existing general layout of your workspace, it will *not* work effectively on workspaces where the arrangement of items is totally random. Instead, G2 ProTools uses an “intelligent” approach that tries to determine your intent, by discerning groups of items that are already nearly aligned, horizontally or vertically, and aligning these items while equalizing the gaps between the items.

G2 ProTools’ approach, while having certain limitations, has significant advantages over other similar tools that do not preserve the general layout of items. For example, typical facilities to arrange folders on a desktop often

arbitrarily rearrange your folders to fit a regularly-spaced rectangular grid. This generates a neat arrangement, but also tends to destroy nearest-neighbor relationships. Folders adjacent to each other before arrangement become widely dispersed after arrangement.

A little experimentation with the `arrange-workspace` facilities may be required to optimize your results, but you will save significant time in the long run.

Undoing Arrangement

The `arrange-workspace` facilities allow you to undo your last `arrange` command, on a workspace-by-workspace basis.

To return the items on a workspace to their previous positions,

→ Choose `undo arrange workspace` from the workspace menu.

Changing Workspaces to Uniform Style

You can format a workspace hierarchy, so that all workspaces in the hierarchy have a uniform color scheme, frame style, and margin width. If you arrange the workspaces in each module into a single hierarchy per module, you can change all workspaces in a module to a uniform style with a single menu choice.

To change a workspace hierarchy to a uniform style:

- 1 Go to the workspace that is at the top of the hierarchy, or at the point in the hierarchy where you want all subworkspace to have the same format.
- 2 Change the foreground and background color, frame style, and margin to the desired settings.
- 3 Choose `format ws hierarchy` from the workspace menu to propagate the style of the workspace recursively to all its subworkspaces.

Shortcuts and Convenience Tools

Describes facilities for printing, launching procedures, keyboard accelerators, and more.

| | |
|--|----|
| Introduction | 63 |
| Printing Workspaces | 64 |
| Quick-Launching Procedures and Methods | 65 |
| Defining and Using Accelerators | 67 |
| Viewing the Description of a Procedure | 67 |
| Loading External Text Files into G2 | 68 |



Introduction

G2 ProTools offers a set of shortcuts and convenience tools designed to speed up the development process. These tools include the following:

- A [print routine](#) that saves you the effort of setting up System Tables each time you want to print a workspace.
- A [procedure launcher](#) that enables you to start a procedure quickly, without creating an action button.
- Definable [keyboard and mouse accelerators](#) that help you rapidly navigate your application, or perform other action commands.

- A [procedure viewer](#) that allows you to see the arguments of a procedure or method, without opening its table.
- A [utility for importing text files into G2](#).

Printing Workspaces

G2 ProTools makes it easier and faster to print workspaces. The normal procedure for printing a workspace involves several steps; specifically, going to the Printer Setup System Table, editing its attributes, returning to the workspace you want to print, and using the print command. G2 ProTools replaces this process with a lightweight print dialog that makes it easier and faster to select print options and carry out printing.

To print using G2 ProTools:

- 1 Choose print workspace from any workspace menu.

The following dialog appears:

- 2 Specify the name of the printer in the Printer edit box.
- 3 Specify the name of the server in the Server edit box.

Note The server depends on the configuration of your host system. At some sites, servers are aliased to have same name as the printer. In other cases, the server name is the name of the machine hosting the printer.

- 4 Specify any print options you wish to use.

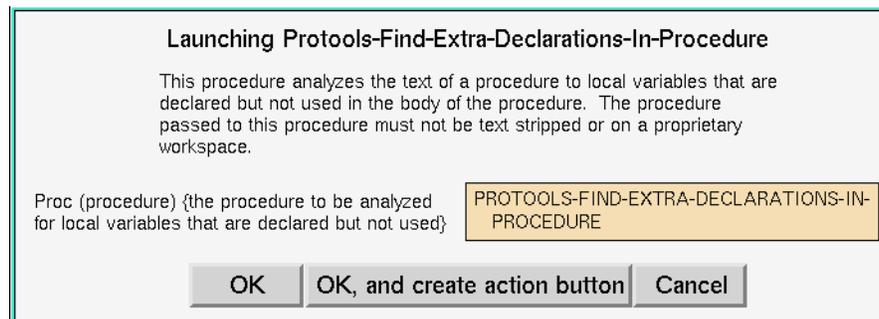
Clicking the Options button displays additional options, including paper size and margins. For information on the various printing options, see the *G2 Reference Manual*.

- 5 Click the OK button to print the workspace.

Quick-Launching Procedures and Methods

The launch-procedure menu choice, available on any procedure or method, enables you to start a procedure without having to create an action button. If no arguments are required, the procedure starts immediately.

If the procedure requires input arguments, this command will generate a lightweight dialog containing appropriate fields for you to specify the arguments to the procedure. For example:



The name of the procedure you are launching appears as the title of the dialog, in this case, `protools-find-extra-declarations-in-procedure`. This procedure requires one argument of type `procedure`, which is the procedure to be analyzed for extra declarations.

Note The dialog gives the local name and data type for the input argument, and includes the embedded comment from the procedure that describes this argument. The dialog also contains the embedded comment from the procedure that describes the overall purpose of the procedure. For comments to appear in the dialog, the comments in the procedure must conform to the standard discussed in [Adding Comments to Your Documentation](#).

Accepting the Defaults

For each field, the launcher proposes a default, when possible. If the procedure argument is an item, then the nearest item of the required class is selected as the

default. If the argument is type `g2-window`, the default is the current window, expressed as `this window`. Clicking the OK button accepts the defaults.

If you need to specify one or more input arguments, click the appropriate edit field and enter the item name or value.

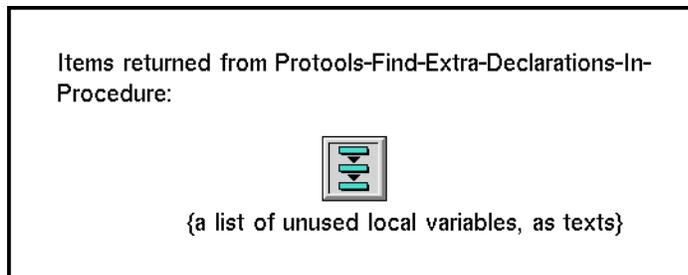
If you anticipate that you will need to launch the same procedure with the same arguments again, click the OK, and create action button button. This will start the procedure and also create an action button nearby the procedure that allows you to start the procedure with the same arguments again.

Note Since this method depends on parsing the target procedure to determine its arguments, it cannot be used if the target procedure is text-stripped.

Viewing Return Arguments

In addition to starting the procedure, the procedure launcher allows you to look at the items or values returned from the procedure, without inserting “inform the operator” statements into the source code.

If the procedure returns one or more items, and the workspace of the returned items does not exist, then the items are displayed on a temporary (transient) workspace; for example:



In this case, `protools-find-extra-declarations-in-procedure` returned a `text-list` containing the list of unused local variables. You can examine the list by describing its contents or viewing its table in the usual G2 manner. Note that the output workspace also contains an embedded comment describing the return argument, extracted from the procedure.

If the procedure returns a value or values, they will be shown on the Message Board.

Defining and Using Accelerators



Some navigation actions are performed so frequently that keyboard and mouse shortcuts, referred to as accelerators, can be very useful. G2 ProTools accelerators allow you to quickly define and install accelerators, to use in your G2 development environment.

To define an accelerator:

- 1 Clone an accelerator object from the G2 ProTools palette and place it on a workspace in your application.
- 2 Specify an action or menu choice, applicable class, and a keystroke in the table of the accelerator.
- 3 Activate the accelerator by using the turn-accelerator-on menu choice.

To turn off the accelerator, use the turn-accelerator-off menu choice. When an accelerator is off, it is indicated by a red circle with a diagonal bar over its icon.

For more information about how to specify the attributes of an accelerator, see the *G2 ProTools Reference Manual*. For a discussion of default accelerators, see [Installing Accelerators](#).

Viewing the Description of a Procedure

This G2 ProTools viewer allows you to see a description of any non-text-stripped procedure or method, without opening the table of the item. Typically, you use this viewer through its menu choice, *get-signature*. The output is displayed on the message board, for example:

```

MESSAGE-BOARD-FOR-DWR
#9 12:42:28 p.m.
This procedure analyzes the text of a procedure
to local variables that are declared but not
used in the body of the procedure. The
procedure passed to this procedure must not
be text stripped or on a proprietary workspace.

PROTOOLS-FIND-EXTRA-DECLARATIONS-IN-
PROCEDURE:
Inputs:
  Proc: procedure {the procedure to be
analyzed for local variables that are
declared but not used}
Outputs:
  text-list {a list of unused local variables,
as texts}

```

The description includes the input argument name, its declared type, and associated comment, if any. The type and associated comment for output arguments are also given. If there is a comment describing the overall purpose of

the procedure, it is also included. See [Adding Comments to Your Documentation](#) for information on the placement of comments, so they can be accessed by this facility.

This feature is particularly useful in conjunction with the call tree facility. Using `get-signature` menu choice on the call tree nodes, you can look at calling signatures without navigating to the items represented by the call tree nodes. See [Creating Call Trees](#) for more information about call trees.

Loading External Text Files into G2

The file uploader provides a simple way to load external text files into G2.

To load an external text file into G2:

- 1 Do one of the following to go to the workspace of the file uploader:
 - Follow the path:
 Protocols-Top-Level > Contents > Shortcuts and Convenience Tools > File Uploader
 - Using Inspect, enter:
 go to protocols-receiving-box
- 2 Edit the `protocols-input-file-name` edit box and specify the directory and file name of the file you want to load.
- 3 If your file uses a special character set, such as jis, ksc, or unicode, edit the `protocols-text-conversion-style` object to specify the desired encoding.

For more information about text conversion styles, see the *G2 Reference Manual*.
- 4 Click the read from file button.

The text of your file will be loaded into the receiving box below the button.

Once the text is in the receiving box, you can copy it using standard G2 text editing commands, or “swipe” the text into an open editor session by dragging the mouse over the text.

@ A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z

A

accelerators
 activating
 built-in
 activating
 deactivating
 user-defined
 creating
attributes
 listing

B

book.htm
bookix.htm
booktoc.htm

C

call next method statement
 and ProTools call tree
call tree
 comments
 generating
 inactive items
 indirectly called procedures
 proxy item menu choices
 edit procedure
 get signature
 go to origin
 hide called procedures
 hide node
 show called procedures
 show callers
 recursive calls
 texts
change message
chapter attributes
 modifying
class definitions
 adding comments
configuring user menu choices
Contents navigation button

control panel
 ask other users buttons
 using
 configuring
control panel navigation button
cooperative development
 and G2 ProTools
cross hair object
customer support services

D

dead code
 finding
debugging
 breakpoint function
defining accelerators
dialogs
 launching protocols-find-extra-declarations-
 in-procedure
documentation objects
 protocols-book
 protocols-chapter
 protocols-documentation-details
 protocols-documentation-template
 protocols-exclusion-marker
 protocols-inclusion-marker
 protocols-sequence-template
documentation templates
 customizing
 description
 page format
 defined by protocols-documentation-
 template
 sequence expansion
 defined by protocols-sequence-
 template

E

enabling references to inactive items

F

- files
 - text
 - loading into G2
- formatting a workspace hierarchy
- free-form comment

G

- G2
 - loading external text files with ProTools
- G2 ProTools
 - See* ProTools
- generating ProTools online reference manual

H

- hypertext markup language (HTML)

I

- inactive items
 - enabling references to
- internal documentation messages
 - change message
 - description
 - creating
 - enlarging
 - free-form comment
 - description
 - locating
 - shrinking
 - to-do-reminders
 - description
- item
 - listing attributes of
 - making permanent
 - measuring memory
 - superior
 - highlighting

L

- launching methods
- launching procedures
 - accepting defaults
 - viewing return arguments
- launching protools-find-extra-declarations-in-procedure dialog

- example of
- loading ProTools
- loading text files into G2

M

- menu choices
 - See* user menu choices
- merging ProTools
- message board
 - posting unused variables
 - viewing procedure description
 - window-specific
- message objects
 - protools-change-message
 - protools-comment
 - protools-to-do
- method calls
 - and ProTools call tree
- methods
 - formatting with ProTools
 - launching with menu choice
 - unused
 - finding
- module hierarchy
 - adding ProTools
 - removing ProTools

N

- newlink "Application Analysis Tools"
- newlink "The G2 ProTools Palette"

O

- online documentation
 - customizing
 - changing output file format
 - creating new chapter
 - modifying chapter attributes
 - removing items
 - removing unwanted chapters
 - translating into another language
 - producing with ProTools

P

- printing ProTools online reference manual
- printing workspaces

- Procedures
- procedures
 - displaying description on message board
 - formatting with ProTools
 - example
 - launching with menu choice
 - menu choice
 - get-signature
 - reformat-procedure
 - protools-check-all-modules-for-extra-declarations
 - protools-check-module-for-extra-declarations
 - protools-find-extra-declarations-in-procedure
 - protools-generate-book
 - protools-generate-profile-report
 - protools-pause
 - unused
 - finding
- profile report
 - example
- profiling
 - using control action buttons
- ProTools
 - accelerators
 - user-defined
 - adding to module hierarchy
 - arranging workspace objects
 - breakpoint function
 - call tree
 - proxy item menu choices
 - configuring
 - control panel
 - configuring
 - description
 - documentation generator
 - changing output file format
 - customizing
 - description
 - producing an reference manual
 - templates
 - file uploader
 - finding unused variables
 - formatting a workspace hierarchy
 - formatting procedures and methods
 - loading
 - making items uniform size
 - online reference manual
 - generating
 - key files
 - printing
 - using
 - printing workspaces
 - procedure launcher
 - accepting defaults
 - viewing return arguments
 - profiling procedures
 - removing from module hierarchy
 - required modules
 - Telewindows broadcast
 - customizing
 - top-level workspace
 - navigation buttons
 - palette objects
 - tree shaker
 - user menu choices
 - configuring
 - window-specific message board
 - used with Telewindows
- ProTools controls
 - changing user mode
 - communicating with other users
 - using profiling controls
- protools module
 - making it required
- ProTools objects
 - accelerator
 - cross hair
 - documentation
 - message
 - under-construction
 - workspace header
- ProTools.kb*
- protools-book class
- protools-chapter class
- protools-documentation-template class
- protools-inform method
- protools-sequence-template class
- protools-top-level workspace
 - Configuration
 - Contents
 - control panel
 - palette objects
 - accelerator
 - cross hair
 - documentation
 - internal documentation messages
 - under construction
 - workspace header
- README

Q

- qualified-name method calls
and ProTools call tree

R

- README navigation button
- reference manual
 - adding comments
 - customizing item selection
 - producing with ProTools

T

- Telewindows
 - communicating with other users
 - cooperative development
- Telewindows broadcast
 - customizing
- text-stripped procedures
 - and ProTools call tree
- titlepag.htm*
- to do reminder
- tree shaker
 - using

U

- under-construction object
- user menu choices
 - arrange-workspace
 - disable-references-to-inactive-items
 - enable-references-to-inactive-items
 - format-ws-hierarchy
 - generate-call-tree
 - go-to-originating-item
 - highlight-superior-item
 - launch-procedure
 - list-attributes
 - make-permanent
 - measure-memory
 - parse-for-unused-variables
 - print-workspace
 - reformat-procedure
 - turn-accelerator-off
 - turn-accelerator-on
 - undo arrange-workspace
- user mode
 - changing with Protools controls

- using accelerators
- using ProTools online reference manual

V

- variables
 - unused
 - finding

W

- workspace
 - menu choice
 - arrange-workspace
 - format-ws-hierarchy
 - undo arrange-workspace
 - neatly arrange contents of
 - undo
 - printing
 - superior item
 - finding
- workspace header object
 - adding to a workspace
- workspace hierarchy
 - uniform style format