CLARION 5

Wizatron Handbook

COPYRIGHT 1998, 1999 by TopSpeed Corporation
All rights reserved.

This publication is protected by copyright and all rights are reserved by TopSpeed Corporation. It may not, in whole or part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from TopSpeed Corporation.

This publication supports Clarion 5 Wizatrons. It is possible that it may contain technical or typographical errors. TopSpeed Corporation provides this publication "as is," without warranty of any kind, either expressed or implied.

TopSpeed Corporation
150 East Sample Road
Pompano Beach, Florida 33064
(954) 785-4555

Trademark Acknowledgements:
Wizatrons™ is a trademark of TopSpeed Corporation.
TopSpeed® is a registered trademark of TopSpeed Corporation.
Clarion™ is a trademark of TopSpeed Corporation.
Microsoft® Windows®, Windows 95®, Windows 98®, and Windows NT® are registered trademarks of Microsoft Corporation.
All other products and company names are trademarks of their respective owners.

Printed in the United States of America (0699)

TABLE OF CONTENTS

1 - INTRODUCTION AND OVERVIEW 7

Introduction - The Age of Meta-Programming 7
Accelerating Your Productivity 7
Wizatrons Defined 9

Wizatron Overview 10
Registering the Wizatron Templates 10
The Clarion Wizatron Family 10

The Dictionary Wizatron 12
Third Normal Form 12
Anatomy of the Wizatrons 13

What You'll Find in this Book 15

2 - GETTING STARTED TUTORIALS 17

Files used in the Tutorials 17

Tutorial One: The Dictionary Wizatron 18
The BOOKS Dictionary 19
Starting the Dictionary Wizatron 20
Controlling Wizatron Detail 21
Auto-Increment and OVER settings 23
Field Interest Level 24
Controlling Third Normal Form 25
Edit-In-Place Options 26
Other Key File Calculations 27
Prompts and Headers 28
Dictionary Wizatron Summary 29
Examining the Dictionary Wizatron Results 30

Tutorial Two - The Application Wizatron 32
Creating an Application Using the Application Wizatron 32
Create a New Application Using the Wizard View 35
Detail Levels 37
Examining the Generated Wizatron-based Application 43

Tutorial Three - The Application Wizatron with an Alternate Style Folio 45
Creating a new application using the Application Wizatron 45
Using the Procedure Wizatrons 54

3 - THE WIZATRON DESIGN ENVIRONMENT 59

Wizatron Views 59
File Menu 63
Edit Menu 64
Options Menu 64
Folio Menu 65
Special Menu Items 76

4 - STATIC ELEMENTS 77

Naming Conventions 77
Categorization and Structure 78
Values (Properties) and Derivation 80
Space Allocation and Space Allocation Properties 84
Dynamic Property Values 88

Available Expansion Macros Listing 90

@BESTWINHEIGHT 90
@BESTWINWIDTH 90
@COUNT 90
@DEFAULT 91
@DESCRIPTION 91
@EVAL 91
@EXTRACT 92
@FIELDS 92
@FILES 92
@FULLNAME 92
@HEADING 93
@INDEX 93
@INHERIT 93
@KEYFIELD 94
@KEYS 94
@NAME 94
@PARENT 94
@PROMPT 95
@RELATED 95
@RELATIONEXPRESSION 95
@RELATIONFIELD 96
@RELATIONFOREIGNKEY 96
@RELATION 96

@REPORTCONTROLTYPE 97
@SCREENCONTROLTYPE 97
@SCREENPICTURE 97
@USER 98
Static Sub-properties 99
Dictionary Attributes 108
Summary of Wizatron Properties 111

5 - STYLE FILES 113

The Wizatron Style File 113

Documentation Strategy 113
Exporting Documentation 114
Viewing the HTML documentation 115
HTML Documentation Tags 116
Sample Documentation of STANDARD.STY 117

General Style File Design Tips 157

The Template Wizatron 158
Introduction 158
Overview 158
Default File Descriptions 159
The Embed Advisor 162
Summary 162

Using the Template Wizatron 163
Set Up - Source to Application Text (TXA) 163
Import the TXA file 163
Edit the Template Data 164
Check the Embed Advisor 166
Export to a Template Source File (TPW) 167
After the Template Wizatron 168

6 - ADVANCED TUTORIALS 169

Alternate Views of the Application Wizatron 169

Creating an Application 169
Types of views 170
Style View 172
Application View 177

Using The Wizatron Configuration Utility 185

Adding New Wizatron Components and Associated Acorn Properties 192

Overview 192

INDEX 203

1 - INTRODUCTION AND OVERVIEW

Introduction - The Age of Meta-Programming

Accelerating Your Productivity

One of our primary goals as programmers and application developers is to get the job done right in the shortest amount of time possible. This is a direct measure of our output quality and productivity.

TopSpeed's mission has always been to provide you with the tools that can help make this possible. This chapter introduces you to the *Wizatrons*, the latest cutting-edge development tool from TopSpeed.

The Evolution of Rapid Application Development

Clarion is a fourth-generation programming language which is optimized for writing business database applications in a Windows-based operating environment. It is supported by a richly integrated development system that eliminates a majority of the menial tasks faced by the everyday developer.

To further extend the productivity of the developer, Clarion 5 includes the object oriented Application Builder Class (ABC) Library, which provides a safe and efficient method of reusing and maintaining common pieces of program code.

Implementation of the library objects is accelerated by the use of the Clarion ABC templates.

The ABC Templates are highly configurable, interactive, and interpreted code generation scripts. When you provide these templates with information extracted from a series of prompts, they generate a custom set of source code based on your choices.

In Clarion 1.5, TopSpeed introduced *wizards*, which optimized and accelerated the use of templates. In Clarion 4, these wizards generated fully-functional applications and procedures built from the ABC template components.

The introduction of Wizards and Templates in Clarion has brought us to the dawn of an era we call “Meta-Programming.”

The art of Meta-Programming is to “transcend,” or “go beyond” the normal task of programming. In layman’s terms, the closer we can get to telling a computer what we would like it to do, in everyday language, the closer we approach meta-programming.

Wizatrons Defined

A new set of Rapid Application Development tools called *Wizatrons* are now available to help us reach a higher level of meta-programming and productivity. Let's explain exactly what they are.

We have many tools in our daily life that help us automate tedious tasks. The dishwasher to wash the dishes, the washing machine to wash the clothes, and the calculator to help us evaluate complex mathematical functions are examples of this automation.

Wizatrons can be viewed as software robots that perform our tedious programming tasks. We give the Wizatron general *rules* about the type of program we wish to generate, and the Wizatron will then calculate the specific program elements necessary. Unlike other RAD tools, these Wizatrons also *learn* from the commands that we give them by remembering our most often used instructions.

Wizatrons actually are special classes of compiled code. They feature an intuitive design interface which is integrated into the Clarion 5 development environment using TopSpeed's template wrapper technology.

One major difference between the new Wizatrons and the standard Clarion wizards is that the Wizatrons are *completely* configurable. You may have clients who prefer their programs to have special colors, button sizes, and certain data oriented features; other clients may have a completely different set of tastes and preferences. The Wizatrons allow you to create a family of styles for each client that can be reused again and again.

Let's look at some other features.

- Wizatrons are designed to increase your programming productivity by an "order of magnitude." OOP and templates were a good start, but Wizatrons raise the bar a little higher.
- Wizatrons are designed to take away the administrative burden of programming. This means a consistent "front-end" design to get your application near perfect the first time.
- Wizatrons are easily modified. Add on to existing Wizatrons at any time, and create an entirely new Wizatron family. Save your work in incremental steps.
- Wizatrons provide a multi-pass capability. If you don't like it the first time, modify it, and run it again and again. You can also prevent your Wizatrons from overriding work that is already completed.
- Develop different styles for different clients. The Wizatrons maintain accurate and consistent behavior in all of your applications.

We will demonstrate all of this and more in the tutorials of this handbook.

Wizatron Overview

This section describes the current suite of available Clarion Wizatrons. It also describes the steps required for their implementation.

Registering the Wizatron Templates

To begin using the Wizatron interface, they must be registered in the Template Registry. The default template wrapper supporting the Clarion Wizatrons is normally automatically preregistered during installation. However, if you need to reregister it, or if you wish to register a third party Wizatron, this is how to do it:

1. **Choose** Setup ➤ Template Registry.
2. In the Template Registry dialog, **press** the Register button.
3. In the Template File dialog, select the *WIZATRON.TPL* file in the ..\TEMPLATE subdirectory, then **press** the OK button.
4. In the Template Registry dialog, **press** the Close button.

The Clarion Wizatron Family

The Wizatrons in Clarion 5 can be divided into several categories:

Application Wizatron

Application Wizatron
The Application Wizatron creates a complete application from an existing data dictionary. By default, it creates a Main (Frame) procedure containing a menu with options calling all subordinate procedures it creates. It also creates Browse and Form (update) procedures for each specified file.

Procedure Wizatrons

Browse Wizatron
This Wizatron creates (by default) a multi-keyed Browse procedure from an existing dictionary file definition. The records displayed are sorted by each key you specify and are controlled by tab controls. If you specify that updates are allowed, it creates an appropriate Form (update) procedure or enables edit-in-place.

Form Wizatron
This Wizatron creates a specific Form procedure (update) from a selected dictionary file definition. Based on the file relationships defined in the data dictionary, additional child browses may also be included.

The Wizatron Configuration Utility (WIZARDx.EXE)

The Wizatron Configuration Utility (WCU) allows you to maintain the default style files that ship with Clarion 5, edit any third party style files, or create new styles.

A style file is the foundation of all Wizatrons. It contains all of a Wizatron's components and properties. You cannot access the Wizatron design environment (views) until a style folio that contains selected style files has been loaded.

Here are a few highlights of the functions and purpose of the Wizatron Configuration Utility:

- The primary function of the WCU is to allow the developer to create new properties to attach to existing default components.
- Existing components can be saved to a new style file and later combined with other components of other style files.
- Each property used in a style file can be explicitly documented in the WCU. You can also export this documentation to a file in HTML format.
- Hidden properties can be created and modified, removing a layer of complexity or clutter from the design stage.

Each feature mentioned above is documented, in detail, in the *Style File* chapter.

The Wizatron Configuration Utility is normally accessed through the Template Utility menu item found in the Application Generator.

However, for the style file developers who are specifically creating styles for multiple applications, you can run the Wizatron Configuration Utility outside of the Application Generator environment.

Each session of the WCU relies on the existence of a Wizatron Control File (*.WCF). This file is similar in structure to standard INI files:

```
!sample.wcf file - two lines required to load outside of environment:
[GENERAL]
 DEFEDIT
```

To load the WCU outside of the environment, use the following convention:

WIZARDX.EXE *filename*

where *filename* is the name of a specific Wizatron control file.

The Dictionary Wizatron

The Dictionary Wizatron is a special Wizatron that prepares your data dictionary for further optimization by the “applications-oriented” Wizatrons (Application, Procedure, etc.). Here are some of the actions that the Dictionary Wizatron performs to accomplish this:

- Controls what keys, and which element of those keys, are populated by the application/procedure Wizatrons.
- Controls population of special data fields that use the OVER attribute.
- Controls the interest level (described in detail in the *Static Elements* chapter) of all fields to control how and where they are included by the application/procedure Wizatrons.
- Enforces a third normal form database, using special field options that are generated by the Dictionary Wizatron. An explanation of this third normal form is described below:

  Background

  In relational database design, normalization is the process of organizing data to minimize duplication.

  In the Clarion 5 Dictionary Editor, normalization involves dividing a database into two or more files and defining relationships between them.

  TopSpeed Wizatrons are optimized for third normal form, which is the third highest level of database normalization.

  Third Normal Form

  In the third normal form, no duplicate information is permitted. For example, if two files in a Clarion dictionary require an order date field, the date field would be separated into a separate file. The order date field would then be accessed through a unique id field and key, stored in the file containing the order date. More importantly, any change to this order date would automatically be reflected in all files that link to the order date file.

  The Wizatrons support of third normal form is illustrated in the next chapter.

Anatomy of the Wizatrons

Here is a breakdown of the major components that embody the Wizatrons:

The **Dictionary Wizatron** helps to *prepare* the data dictionary by populating a variety of user options that are used by the application and Procedure Wizatrons to generate optimum data-centric applications.

The **Wizatron User Interface** is the interactive area of the Application/Procedure Wizatrons. The **styles** provide the components and properties available to the Wizatron's **views**. The **views** provide different design environments that are called by the developer and allow focus on specific Wizatron features (these views are examined in greater detail in the *Advanced Tutorials* chapter).

The **Wizatron Engine** embodies the internal assembly and generation of application elements that are represented by an Application Text File (*.TXA). This file is automatically imported into the Application Generator, where the resulting application may then be modified by the developer before calling the Project System that produces the specific compile and link.

What You'll Find in this Book

This handbook is divided into the following parts:

- **Introduction and Overview**

  Chapter 1 provides a general description of Wizatrons and their purpose

- **Getting Started Tutorials**

  Chapter 2 provides a series of tutorials designed to familiarize you with the Wizatron design environment. The tutorials are intended to get you up to speed with the Wizatrons in as little time as possible.

- **Design Environment**

  Chapter 3 covers the Wizatron design environment.

- **Static Elements**

  Chapter 4 provides information about the static properties and sub-properties contained in the Wizatron environment. The list of available expansion macros are also detailed here.

- **Style Files**

  Chapter 5 provides specific information concerning the standard style file. This includes; a detailed description of the static properties, information on custom style file design, and the Template Wizatron (a special design tool).

- **Advanced Tutorials - Wizatron Design Techniques**

  Chapter 6 provides a series of tutorials designed to explore the more advanced areas of the Wizatron design environment. A walk through of the Application and Style views is first presented, followed by a detailed look at the Wizatron Configuration Utility. This chapter concludes with an example showing how to customize a style file.

2 - GETTING STARTED TUTORIALS

This chapter presents a series of tutorials that will familiarize you with designing your applications using Wizatrons. The order of these tutorials is intended to present the logical steps to create a Wizatron-based application, starting with the Dictionary Wizatron, using the Application Wizatron with different default styles, and finally, polishing the application using the procedure Wizatrons.

Files used in the Tutorials

Throughout this handbook, there are a variety of tutorials that are designed to increase your productivity when using Wizatrons.

The files used are stored in the folders (directories) shown below. These folders are contained within your Clarion install folder.

WIZTUTOR
This is the main folder. All of your tutorial work is stored here.

START
Contains an assortment of files that you can copy into the WIZTUTOR folder, including data files and a data dictionary. This will enable you to redo the tutorials from time to time.

SOLUTION
When you complete a particular tutorial, you can compare your work with the completed applications that are stored here.

Let’s get started with the first tutorial!

Tutorial One: The Dictionary Wizatron

The following tutorial focuses on a unique type of Wizatron - the Dictionary Wizatron. This Wizatron is unique for the following reasons:

- The Dictionary Wizatron is only presented with one view - a wizard view.
- This Wizatron is not loaded as a template utility like other Wizatrons - it is loaded from a menu item in the Dictionary Editor.
- You cannot modify the Dictionary Wizatron - its values are static, and there is no style file associated with it.

The purpose of the Dictionary Wizatron is to prime an active data dictionary with User Options that are recognized and processed by the subsequent application and procedure Wizatrons. Using the Dictionary Wizatron with your active dictionary ensures that the Application Wizatron generates a solid application with built-in data integrity.

Although it is not a requirement to use the Dictionary Wizatron before using the Application Wizatron, it is stongly recommended. The Dictionary Wizatron optimizes your data dictionary for efficient performance by the Application Wizatrons.

You will probably need to run the Dictionary Wizatron the first time that you wish to use it with the Application Wizatrons, and after that only when changes are made to the dictionary.

The following tutorial explores the available options and their effects on an active data dictionary. Let’s begin by exploring our example data dictionary.

The BOOKS Dictionary

All of the tutorials in this handbook use a simple data dictionary named *BOOKS.DCT*, designed to manage a book collection. Using the Data Modeller included with the Clarion 5 Enterprise Edition, a graphic view of our dictionary's files, fields, keys and relations is shown here:

File/Field Descriptions

- The AUTHOR file contains a numeric ID and the fields which store the author's name.
- The BOOK file has title and synopsis information, including ID fields which link to related author and publisher information.
- The PUBLISHER file contains a numeric ID and the field that stores the publisher's name.

Defined Relationships and Additional Attributes

- There is a many-to-one relationship defined between BOOK and AUTHOR (an author writes many books) and a many-to-one relationship defined between BOOK and PUBLISHER (a publisher publishes many books).
- There are two **Validity Checks** defined in the *BOOK* file:
  AuthorID **Must Be In File** *AUTHOR*, and
  PublisherID **Must Be In File** *PUBLISHER*

Spend a couple of minutes reviewing this dictionary. When you are ready, proceed with the tutorial.

Starting the Dictionary Wizatron

Starting Point:
Open the Clarion 5 development environment (if you do not know how to do this, please refer to the Getting Started manual). The Pick dialog should be open. Make sure that the Wizatrons have been successfully included in the Template Registry. (See the *Introduction and Overview* of this manual*)*

1. **Press** the **Open...** button and **select** the BOOKS.DCT located in the *\CLARION5\WIZTUTOR* folder. Make sure that the **Files of Type** dialog is set to **Dictionary(*.dct)**.

The Dictionary Editor should now be loaded with the *BOOKS.DCT*. Let's examine some important areas before loading the Dictionary Wizatron. After running the Dictionary Wizatron, we will come back and review the effect on the following dictionary items.

2. In the *Files* listbox, **highlight** the *AUTHOR* file and **Press** the **Field/Keys...** button.
3. The *AuthorID* field should be highlighted. **Press** the **Properties** button.
4. **Select** the *Options* tab. **Verify** that the **Do Not Auto-Populate This Field** is not checked. Also, **verify** that the *User Options* dialog is empty(blank).
5. **Press** the **OK** button to close the *Edit Field Properties* window.
6. From the Field / Key Definition window, **select** the *Keys* tab.
7. The *AuthorIdKey* should be highlighted. **Press** the **Properties** button.
8. **Select** the *Options* tab. Verify that the **Do Not Auto-Populate This Field** is not checked. Also, **verify** that the *User Options* dialog is empty(blank).
9. **Press** the **OK** button to close the *Edit Key Properties* window.
10. **Press** the **Close** button on the *Field / Key Definition* window.
11. From the Dictionary's main menu, **choose** **File ➤ Dictionary Wizatron.**

Controlling Wizatron Detail

The first dialog of the Dictionary Wizatron is shown below:

The most important control to understand when navigating through the *Wizard* view is found near the bottom of the screen. It is a slider control called the **Prompt Detail.**

There are incremental settings on the Prompt Detail found in the Dictionary Wizatron. The current setting (slider to the far left) reflects the **Minimum Detail**. This setting identifies the minimum required information that the Dictionary Wizatrons deems important.

The various input controls located above the Prompt Detail are the actual *prompts* used by the Wizatron to gather the necessary information needed to populate the dictionary with appropriate user options. These prompts can be standard entry controls, drop controls with a fixed list of choices, or lookup (ellipsis) controls used for a variety of selection options.

The entire collection of prompts shown in the Wizard view make up the *Wizatron Components*. At times, the level of detail you set may force multiple properties that need to be set for a single Wizatron component.

Buttons near the bottom of the window are used to navigate through the Wizatron components.

The **Next** button navigates to the next logical Wizatron component.

The **Back** button navigates to the previous Wizatron component.

The **Finish** button (shown disabled) is enabled when the Dictionary Wizatron has enough information to update your dictionary. Pressing this button completes the Dictionary Wizatron.

1. Referring back to the Prompt Detail control, use the mouse to **move** the slider setting one notch to the right.

   At the middle setting, you should see three (3) prompts.

2. Move the your Prompt Detail slider to the far right (Maximum Detail).

   The window should now look as shown here:

Note: The pictures shown in this tutorial may have different default settings. If you have explored the Dictionary Wizatron prior to this tutorial, you may have modified these settings. Since the tutorial will guide you through each setting, please ignore any differences for now.

For the remainder of this tutorial, leave the Prompt Detail control set to maximum detail.

Auto-Increment and OVER settings

The first window of the Dictionary Wizatron controls how to treat all Auto-Incrementing fields and their keys, and provides population options for fields defined with an OVER attribute (the Place Over option in the dictionary - see the *User's Guide* and *Language Reference* for more information).

1. **Check** the check boxes as shown below:

This settings shown above specifies to populate only the non-incrementing fields that are components of any auto-numbering key.

The *AuthorID* field and *AuthorIDKey* key that we examined earlier from the AUTHOR file are affected by this Wizatron setting.

Since the dictionary does not have any fields defined with the OVER attribute, leave the appropriate choices on the first window unchecked.

2. **Press** the **Next >** button to continue to the next Dictionary Wizatron page.

Field Interest Level

Field Interest is an important area of the Dictionary Wizatron. This option allows you to override the default values set by the Dictionary Wizatron. All dictionary elements have a relative interest level that are detected by other Wizatrons. These levels are compared to interest thresholds set in the style files to control field population. Interest levels range from 1 to 10. Here are the default conditions:

- Data elements assigned low interest levels (1-4) normally are sparsely populated or not seen at all by your application.
- Data elements not modified by the developer are given a default value of five(5), considered “mid-range” by the Wizatrons and receive no special attention than they normally would. They are considered neutral.
- Interest levels from 6-10 normally imply that the Wizatrons give these data elements special treatment. They are populated and appear more frequently in your application.

It would be a tedious task to assign every data element in your dictionary a specific interest level. The Dictionary Wizatron makes this task easier. Let’s demonstrate this.

1. Referring to the window shown above, **Press** the **Insert** button. This initializes edit-in-place in the list box shown above.
2. Enter (**Insert**) two records as shown below:

Field ID Text Containing	Interest Level
Name	*9*
ID	*6*

These options tell the Wizatron to look for all field labels in your dictionary containing the word *name* or the word *ID* (e.g., First*Name*, Last*Name*, Author*ID*, Book*ID*) and apply the new interest levels you entered.

3. **Press** the **Next >** button to continue.

Controlling Third Normal Form

In the third normal form of database normalization, duplicate information is eliminated.

For example, if two files in a Clarion dictionary require an author name field, the author name is stored in a separate file. The author name field is accessed via a unique *AuthorID* field and associated key, stored in the file containing the author name. More importantly, any change to this author name is automatically reflected in all files that link (have relationships) with the *Author* file.

This part of the Dictionary Wizatron supports the third normal form in your applications. By checking the interest levels of the fields in the related files, the Wizatron calculates which related information is important from the lookup file, and alerts the Application Wizatron to substitute the appropriate fields in designated procedures. This is accomplished using the Wizatron *DISPLAY* option in the *User Options* of the related link field.

The **Substitute Field Interest** spin control causes the Wizatron to generate the *DISPLAY* option into those linking fields that have an interest level greater than or equal to the value entered here. Since our *ID* fields have an interest level of six(6) (set in the previous window) we should expect a *DISPLAY* option to appear in each of our linking fields' *User Options*.

The **Max number of fields** spin control specifies the number of fields that will be populated by the *DISPLAY* option.

The **Modify Interest Level** check box modifies the lookup fields' interest level (normally of low interest) to be the average of the substituted fields.

1. Accept the settings as shown on the window above. **Press** the **Next >** button to continue.

Edit-In-Place Options

The Browse Wizatron provides the ability to enable edit-in-place updates to a browsed file instead of a separate update (form) procedure. This option is recommended for files with one-way lookup relationships, such as a State Code file. Files with complex relationships are better managed with a separate update procedure.

1. **Enter** (or accept) the defaults as shown on the window above. This specifies that files with five (5) fields or less, with no applied validity checks, should use the edit-in-place option.
2. **Press** the **Next >** button to continue.

Other Key File Calculations

1. **Check** the **Calculate Interest Levels** box. This feature calculates the interest level of any “populating” key based on the interest level of the key fields.
2. **Check** the **Calculate “Text Primary” key** box. This forces the key with the highest interest level to be assigned the *TEXTPRIMARY* option. The key with the *TEXTPRIMARY* option becomes the primary sort key of Wizatron generated browses.
3. **Press** the **Next >** button to continue.

Prompts and Headers

The last option screen in the Dictionary Wizatron is shown above.

The Wizatrons allow you to take full advantage of the prompt and header information that is stored in the data dictionary. For example, the lookup fields that you designate via the DISPLAY User Options will often override the default linking fields' prompts and/or column headers with their own values.

Remember that you have total control of the prompts and header information that the Application Wizatron generates. This option allows you to fine tune the information that will be generated by the Wizatrons.

This screen provides a relational tree view of your data dictionary, with edit-in-place options for prompts and headers of the individual fields.

1. **Edit** the *Author.FirstName* and the *Publisher.Publisher* prompt fields as shown above, removing any text information stored.
2. **Press** the **Finish** button to complete the Dictionary Wizatron and display a complete summary of session activity.

Dictionary Wizatron Summary

At the completion of the Dictionary Wizatron, there is a summary window that shows a complete listing of field labels and the action that was performed on them.

If you need a copy of the results, **press** the **Copy** button. This will send your results to the Windows clipboard. You can later paste the results into the text editor of your choice for saving and/or printing.

1. **Press** the **Close** button to return to the Dictionary Editor.

Examining the Dictionary Wizatron Results

Let's examine the effect that our Dictionary Wizatron session had on *BOOKS.DCT:*

1. **Highlight** the *AUTHOR* file and **press** the **Fields/Keys...** button.
2. **Highlight** the *AuthorID* field and **press** the **Properties** button.
3. In the Edit Field Properties window, **select** the *Options* tab.

You can see in the *User Options* text field that the Dictionary Wizatron has given the *AuthorID* field an Interest of six(6). This is based on the interest level we assigned to all fields containing the text *ID*.

4. **Press** the **OK** button to exit the Edit Field Properties window.
5. **Press** the **Close** button to exit the Field / Key Definition window.

Let's examine some important fields in the *BOOK* file.

1. **Highlight** the *BOOK* file and **press** the **Fields/Keys...** button.
2. In the Field / Key Definition window, **select** the *AuthorID* field and **press** the **Properties** button.
3. In the **Edit Field Properties** window, **select** the *Options* tab.

In addition to setting the Interest level to six (6), the Wizatron added a DISPLAY attribute to automatically retrieve the remaining fields from the *AUTHOR* file in the event of a lookup. We instructed the Wizatron to retrieve all *AUTHOR* fields with an interest level greater than or equal to five (5), the default value. We set a maximum of three (3) fields to be applied to the DISPLAY option. These settings forced all *AUTHOR* fields to be applied as part of the DISPLAY option.

By now, you should feel comfortable navigating to the *User Options* text field of a dictionary's files, fields and keys.

1. **Examine** the remaining data elements' *User Options* as follows:
   - The *PublisherID* field in the *BOOK* file.
   - The remaining fields in the *BOOK* file (all should have a default interest of 5).
   - All keys of the *BOOK* file, with particular focus on the *ByAuthorID_Date* key. Notice the TEXTPRIMARY attribute set here. This attribute will give the *ByAuthorID_Date* key first display priority in its generated Browse procedure.
   - Each file in the *BOOKS.DCT* has some type of field validation, so the EDITINPLACE attribute should not be set by default in each files' *User Options*.
2. **Add** the EDITINPLACE attribute to the *User Options* of the *AUTHOR* and *PUBLISHER* files. These files are small, and can easily be updated from their respective Browse procedures.
3. When you are finished examining the *User Options*, **press** the **Close** button from the main dictionary window, and **save** your dictionary when prompted.

This completes the Dictionary Wizatron tutorial. We'll be using this dictionary in the next section, where we will generate a complete application using the Application Wizatron.

Tutorial Two - The Application Wizatron

Creating an Application Using the Application Wizatron

This tutorial is designed to familiarize you with the Application Wizatron, first using the Quick Start and later the Wizard View to generate an application.

Quick Start a new application using the Application Wizatron

1. **Press** the **New...** button and create a new empty *WT_ONE* application in the *\CLARION5\WIZTUTOR* folder, then **press** the **Save** button. (do not use the Quick Start option here).

2. In the Application Properties window, use the *BOOKS.DCT* as your *Dictionary File*. Make sure that the Application Wizard check box is not checked, and **press** the **OK** button to display the Application Tree.

3. **Choose** **Application ➤ Template Utility...**

4. **Highlight** the **ApplicationWizatron** branch under **Class Wizatrons**, then **press** the **Select** button. The following option window is displayed:

5. **Check** the **Create a QuickStart View?** option to launch a Quick Start View, then **press** the **Ok** button to continue.

6. **Select** the *STANDARD.STF* file in the *\Clarion5\Template* folder when asked to select a style folio, then **press** the **Ok** button to continue.

Welcome to the Quick Start View!

The Quick Start View is designed to create an application in a few easy steps. Each page has a detailed, user friendly description of the task or tasks that are about to be performed. For example, in the screen shown above, all Browses and Forms to be generated are controlled by answering a simple question.

7. **Press** the **Apply** button to start the Wizatron engine. A series of messages display to inform you of the Wizatron components created based on information in the data dictionary.

8. **Press** the **Close** button to close the QuickStart View.

9. **Press** the **Generate** button on the Wizatron toolbar.

With a few easy clicks, you have just generated your first Wizatron application. You should be automatically returned to the Application Generator, where your application tree displays:

Your application is ready to compile and run. For now, let's save this application and examine some additional features of the Wizatrons.

10. **Press** the **Ok** button to save and close the WT_ONE application.

Create a New Application Using the Wizard View

Another alternate view when working with Wizatrons, the Wizard view, will be demonstrated next. First, let's create a new application.

1. **Choose** **File ➤ New ➤ Application** and create a new empty *WT_TWO* application in the *\CLARION5\WIZTUTOR* folder (do not use the Quick Start here).

2. In the Application Properties window, use the *BOOKS.DCT* as your *Dictionary File*. Make sure that the Application Wizard check box is not checked, and **press** the **OK** button to display the Application Tree.
3. **Choose** **Application ➤ Template Utility...**
4. **Highlight** the **ApplicationWizatron** branch under **Class Wizatrons**, then **press** the **Select** button.

The *Prompts for ApplicationWizatron* window displays.

5. Make sure that all *QuickStart* options are turned **off** (Check boxes are unchecked), then **press** the **Ok** button to continue.
6. **Select** the *STANDARD.STF* file in the *\Clarion5\Template* folder when asked to select a style folio and **press** the **Open** button to continue.

Wizatron Views

After selecting the style folio, the Wizatrons assemble the available design components and available data from the selected dictionary. The initial view when not using Quick Start is shown below:

This is the *Wizard* View. There are three other views available (*QuickStart*, *Style*, and *Application*). We have previously explored the *QuickStart* View, but this tutorial focuses on the *Wizard* view. The two remaining views are presented later in the *Advanced Tutorials* chapter.

Detail Levels

The most important control to understand when navigating through the *Wizard* view is found near the bottom of the screen. It is a slider control called the **Prompt Detail Level.**

In the Application Wizatron, there are incremental settings on the Prompt Detail Level. The current setting (slider to the far left) reflects the **Minimum Detail**. This setting identifies the minimum required information necessary for the Wizatrons to generate an application.

Above the Prompt Detail Level are the *prompts* used by the Wizatron to gather the necessary information needed to generate your application (or procedure). These prompts can be standard entry controls, drop controls with a fixed list of choices, or lookup (ellipsis) controls used for a variety of selection options.

The entire collection of prompts shown in the Wizard view make up the *Wizatron components*. At times, the level of detail you set may force multiple pages for a single Wizatron component. When a Wizatron component has multiple pages, small buttons display to allow navigation between pages.

Back and Next Buttons

Buttons near the bottom of the window are used to switch to the next or previous logical Wizatron component.

The **Next** button is used to move forward to the next logical Wizatron component.

The **Back** button is used to review or modify the previous component.

1. Referring back to the Prompt Detail Level control, move the slider setting one notch to the right. At this level, you should see four or five prompts.
2. Move the setting one more notch to the right. At this level, you should now see seven prompts.

3. Move the slider bar again one setting to the right. At this level, you should still see seven.

The detail prompts in steps 2 and 3 did not vary. On some Wizatron pages, a Prompt Detail Level setting may not increase or decrease the number of prompts displayed. This is because the settings of this Wizatron determined that a particular detail level was not deemed important at this time.

4. Finally, move the Prompt Detail Level slider to the far right (Maximum Detail). The window should now look as shown here:

This detail level displays a two-page view.

5. To view the second page, **Press** the button. A partial view of your window should now look as shown below:

6. Move the Prompt Detail Level slider back to the far left (Minimum Detail) to display the least number of prompts.

In summary, the heart of all control in the Wizatron Wizard View is focused on the proper use of the Prompt Detail Level slider. You can be involved in all of the intricacies of your application or procedure, or, choose to limit your design to its basic prompt components.

Tip: **We strongly recommend keeping your Prompt Detail Level set to its minimum setting when designing your application in the Wizard View. Maximum level settings can actually confuse the overall design process. For *total* control of your application, it is better to use the Application View. The Application View is discussed in a later tutorial.**

Building an Application with the Wizard View

In this section, we will continue with the design of the *WT_TWO* application we started in the last section.

> **Starting Point:**
> **Make sure that you are still on the first component of the Application Wizatron, and that your Prompt Detail Level is set to Minimum (slider control to the far left).**

1. The drop control settings of the **Target Operating System** and **Destination Type** prompts should be set to **Windows - 32 bit** and **EXE** respectively. Verify this now, then **press** the **Next** button to continue.

The next prompt is shown below:

2. At the **Generate browses for these files** prompt, **press** the ellipsis (...) button to display a list of files.
3. **Press** the >> button to select all files contained in the *BOOKS.DCT* dictionary.

All files in the *BOOKS.DCT* dictionary should now display in the **Selected Items** list box.

4. **Press Ok** to continue in the Wizatron.

   Notice that the Wizatron compiles all files into related Wizatron components, actually increasing the available components and prompts available in the Wizard View. This can take a minute or so, depending on the number of files you select.

5. **Press** the **Next** button to continue.

The selected files from *BOOKS.DCT* generate multiple browses. Each of their possible sort orders is presented as separate component prompts.

6. **Press** the **Next** button repeatedly (about three to four times) to navigate through the remaining sort orders. When you reach the last Wizatron component, the **Next** button is disabled:

We have reached the end of the line. The Application Wizatron now has everything it needs to generate an application.

Controlling ToDo Procedures

The window in the lower left corner displays a list of ToDo Procedures. These are procedures that the Wizatrons allow you to control. For example, if you only want to create a Browse procedure without an Update (Form), you can ignore the ToDo Procedure *UpdateBook* and generate the application at this time.

1. **Highlight** the *UpdateBook* procedure in the *ToDo Procedures* list box and **press** the ▲ button to generate the procedure.

All possible procedures in this session are now created by the Wizatrons.

2. **Press** the **Generate** button on the toolbar to exit the Wizatrons and generate your application.

The finished application displays.

You can make and run this application at this time.

Examine the way the menu and window components are organized. This application and its look and feel are the result of the Application Wizatron, using the *STANDARD.STF* folio with the Dictionary Wizatron settings.

But what makes the Wizatrons more powerful than the standard wizards? One of the most powerful features is the ability to use different style folios. We will demonstrate this in the next tutorial.

Examining the Generated Wizatron-based Application

This part of the tutorial is essentially a brief examination of the Wizatron's generated output into an application format. Let's look at what our work has produced to this point.

Examining a Browse

Starting Point:
You should be back in the Application Generator's Tree View.

1. RIGHT-CLICK on the *BrowseBook* procedure, and **choose** **Window** to enter the Window Formatter.
2. RIGHT-CLICK on the list box, and **choose** **List Box Format...** from the pop up menu.

3. In the window shown above, **observe** the populated fields.

   You should see the *Publisher* field from the PUBLISHER file, and *First Name*, *Middle Initial* and *Last Name* from the AUTHOR file. The Wizatron has populated the BOOKS files' related fields based on our preliminary Dictionary Wizatron settings.

4. **Press** the **Ok** button to exit the List Box Formatter.
5. **Choose** **Exit!** from the Window Formatter's main menu to return to the Application Tree, and **press** the **No** button when you are asked to save Window changes.

Examining a Form

1. RIGHT-CLICK on the *UpdateBook* procedure, and **choose** **Window** to enter the Window Formatter.

In the window shown above, there are Lookup Buttons created for each many-to-one relationship defined in the *BOOK* file of the *BOOKS.DCT*. Appropriate procedures are called (*BrowseAuthor* for *Author*, *BrowsePublisher* for *Publisher*) to validate the correct linking fields and enforce third normal form.

2. **Choose** **Exit!** from the Window Formatter's main menu to return to the Application Tree.
3. From the Application Tree, **Choose** **OK** to save and close the WT_TWO application.

Summary

- Based on our *BOOKS.DCT* design, and the execution of the Dictionary Wizatron, we are maintaining database referential integrity, enforcing third normal form, controlling edit-in-place and field population options. All of this is accomplished through the Wizatron design interface.
- In addition, every part of the application is generated by default values and rules contained in the selected style (STANDARD.STF) folio.
- Total control of the application's *Visual Components* (e.g., control types, size, color, text, position, etc.) and *Template Components* (e.g., Browse, Form, Frame, etc.) are defined in the Application Wizatron.

In the next tutorial, let's examine how we can modify the Wizatrons to produce the custom Visual and Template components desired for our next application's specifications.

Tutorial Three - The Application Wizatron with an Alternate Style Folio

Creating a new application using the Application Wizatron

This tutorial is designed to familiarize you with using different styles in the Application Wizatron.

Create a new application and start the Application Wizatron

1. **Choose File ➤ New ➤ Application** and create a new empty *WT_THREE* application in the *\CLARION5\WIZTUTOR* folder using the BOOKS.DCT (do not use Quick Start or the Application Wizard options here).

2. **Choose Application ➤ Template Utility...**

3. **Highlight** the **ApplicationWizatron** branch under **Class Wizatrons**, then **press** the **Select** button.

At this time, you should see the *Prompts for ApplicationWizatron* window.

4. Make sure that all *QuickStart* options are turned **off** (Check boxes are unchecked), then **press** the **Ok** button to continue.

5. **Select** the *CUSTOM.STF* file in the *\Clarion5\Template* folder when asked to select a style folio.

CUSTOM.STF is an alternative style folio designed to demonstrate the Wizatron's ability to affect the visual components of your application.

Make sure that you are still on the first component of the Application Wizatron, and that your Prompt Detail Level is set to Minimum (slider control to the far left).

6. The drop control settings of the **Target Operating System** and **Destination Type** prompts should be set to **Windows - 32 bit** and **EXE** respectively. Verify this now, then **press** the **Next** button to continue.

   The next set of component prompts is shown below:

7. At the **Generate browses for these files** prompt, **press** the ellipsis (...) button to display a list of files.

8. **Press** the >> button to select all files contained in the *BOOKS.DCT* dictionary.

9. All files in the *BOOKS.DCT* dictionary now display in the **Selected Items** list box. **Press Ok** to continue in the Wizatron.

Notice that the Wizatron compiles all files into related Wizatron components, actually increasing the available components and prompts available in the Wizard View.

10. **Highlight** the *UpdateBook* procedure in the *ToDo Procedures* list box and **press** the button to generate the procedure.

All possible procedures in this session are now created by the Wizatrons.

11. **Press** the **Generate** button on the toolbar to generate your application and exit the Wizatrons.

The finished application displays:

This looks similar to the previous tutorial.

12. **Make** and **Run** this application at this time.

Let's examine this application at runtime in more detail:

The window shown above contains the *BrowseBook* procedure and the *About window* of the WT_THREE application. Several features are different from the WT_TWO application:

- The tab controls of BrowseBook (and other browse procedures) are positioned below the list box.
- There are now icons associated with each button control.
- Each window (with the exception of the main application frame) now has wallpaper associated with it).
- There are significant color changes in the opening splash screen.

Examining the CUSTOM.STY file

1. When you have finished examining the WT_THREE executable, **exit** the program to return to the Application Generator's Application Tree.

Let's examine the CUSTOM.STY file in more detail. We will look at the specific differences that exist between the CUSTOM.STY and the STANDARD.STY file.

2. **Choose** **Application ➤ Template Utility...**

3. **Highlight** the **WizardConfiguration** branch under **Class Wizatrons**, then **press** the **Select** button.

The Wizatron Configuration Utility looks similar to the Application Wizatron interface, but allows complete modification to a style file.

4. **Select** the *CUSTOM.STF* file in the *\Clarion5\Template* folder when asked to select a style folio. This folio contains the *CUSTOM.STY* file that we will examine.

Folios are collections of style files that allow you to combine multiple styles into a single wizatron design session. They use a default file extension of *.STF*.

Style Files use the default extension of *.STY*.

The tree on the left represents the building blocks used to construct applications. Each entry represents a single component. The list box on the right contains the properties associated with the currently highlighted component. Anything we change and save here is automatically applied to all future applications that use this style sheet (*CUSTOM.STY*).

All of the TopSpeed components are prefixed TOPSPEED and grouped together. Third-party components would appear in their own sections.

The branch structure of the tree mimics the way the components' properties are inherited. Let's examine some specific properties of *CUSTOM.STY* in more detail.

5. RIGHT-CLICK anywhere on the list box tree on the left side of the window.

   A pop up menu displays.

6. **Choose** **Find** on the pop up menu.

   The Find dialog window displays.

7. **Type** *TopSpeed:Button#BrowseQBE* in the **Find What** entry field and **press** the **Next** button.

   This selects the *TopSpeed:Button#BrowseQBE* component.

8. **Scroll** the *property/value* list box on the right and **highlight** the *ICON* property. A partial view of your display is shown:

Here is the location of icon files that are associated with a particular Wizatron button.

9. **Scroll** the list box tree on the left side of the window and notice the assigned icon files to the following button components:

Component Name	Icon
TOPSPEED:Button#BrowseSelect	wizOk.ico
TOPSPEED:Button#Cancel	wizCncl.ico
TOPSPEED:Button#Change	wizEdit.ico
TOPSPEED:Button#Close	wizCncl.ico
TOPSPEED:Button#Delete	wizDel.ico
TOPSPEED:Button#Insert	wizIns.ico
TOPSPEED:Button#Save	wizOk.ico

Wizatron Components and Properties

Let's conclude this section of the tutorial by reviewing other Wizatron components whose selected properties were modified or added in *CUSTOM.STY.* Use the **Find** option as shown in *step 6* of this section, and **examine** the contents of the appropriate properties. This shows the differences between the *STANDARD* and *CUSTOM* styles.

Component Name	Property
TOPSPEED:Entry	COLOR.BASIC
(Overrides its default color with COLOR:WINDOW)	
TOPSPEED:Entry#Locator	ALLOCATION.ZONE
(Moves the position of the Entry Locator)	
TOPSPEED:Checkbox	COLOR.BASIC
TOPSPEED:Checkbox	TRANSPARENT
(Modifies color and transparent properties of all checkboxes.)	
TOPSPEED:String	TRANSPARENT
(Makes all string fields transparent.)	

Component Name	**Property**
TOPSPEED:String#Splash#Date	FONT.COLOR

(Custom color for the Splash Screen Date font text)

TOPSPEED:String#Splash#Developer	FONT.COLOR

(Custom color for the Splash Screen Developer font text)

TOPSPEED:String#Splash#HeadingSmall	FONT.COLOR

(Custom color for the Splash Screen Small font text)

TOPSPEED:String#Splash#HeadingLarge	FONT.COLOR

(Custom color for the Splash Screen Large font text)

TOPSPEED:String#Splash#HeadingLarge#Bottom	FONT.COLOR

(Custom color for the Splash Screen Bottom font text)

TOPSPEED:Prompt	TRANSPARENT

(Add the TRANSPARENT attribute to all prompt fields)

TOPSPEED:Prompt#Locator	ALLOCATION.ZONE

(Change the default position of all locator prompts)

TOPSPEED:Box#Splash#BoxTop	COLOR.FILL

(Change the fill color of the splash screen box control)

TOPSPEED:Box#Splash#BoxTopInternal	COLOR.FILL

(Change the fill color of a second splash screen box control)

TOPSPEED:Sheet	COLOR.BASIC
TOPSPEED:Sheet	TABPOSITION

(Change the default color/ tab control position on all sheet controls.)

Component Name	Property
TOPSPEED:Window	GRAY
(Remove the GRAY attribute on all windows.)	
TOPSPEED:Window	WALLPAPER
(Assign a default wallpaper background.)	
TOPSPEED:Window#Frame	WALLPAPER
(Override the default wallpaper with no wallpaper on the frame.)	
TOPSPEED:Window#Splash	GRAY
TOPSPEED:Window#Splash	WALLPAPER
TOPSPEED:Window#Splash	WALLPAPERPLACEMENT
(Override the wallpaper with another type on the splash screen.)	

This completes our look at the Application Wizatron. To finish this section of our tutorial:

Choose File ➤ Quit to exit the Wizatron Configuration Utility. When asked to save your style file changes, **press No.**

Summary

- There are about 22 style properties modified in approximately 17 different Wizatron components in the CUSTOM.STY file. These primarily affect the Visual Components in your application.
- This tutorial demonstrated the flexibility and power of Wizatrons. Using the same dictionary, we rapidly created two applications with similar functionality; each with a different look and feel.
- More complex style component modifications are found later in this handbook in the *Advanced Tutorials* chapter.

Using the Procedure Wizatrons

This tutorial explores the Procedure Wizatrons. These Wizatrons are especially useful when you wish to expand or enhance selected parts of existing applications.

Starting Point:
Begin this tutorial from the Application Generator's Application Tree. You should have the *WT_THREE* application loaded and active.

1. **Highlight** the *BrowseAuthor* procedure and **press** the DELETE key on your keyboard.

2. **Press** the **Yes** button when asked to delete this procedure.

   The *BrowseAuthor* procedure should now appear as a **ToDo** procedure.

3. **Highlight** the *BrowseAuthor* procedure and **press** the ENTER key.

4. In the Select Procedure Type window, **highlight** the *Browse* procedure type and **press** the **select** button. Make sure that the *Use Procedure Wizard* check box is checked.

The following window displays.

5. **Highlight** the **BrowseWizatron** branch under **Class Wizatrons**, then **press** the **Select** button.

   The following window displays.

6. When prompted to **select** which file you would like to browse, **press** the ellipses button and **choose** the *AUTHOR* file from the file list provided.
7. Make sure that all *QuickStart* options are unchecked, and **press** **Ok** to continue.
8. **Select** the *STANDARD.STF* file in the *\Clarion5\Template* folder when asked to select a style folio.

> **Tip:** **You may notice at this point that the Wizatrons give you the ability to apply different styles to procedures within the same application. This is only one of the many ways that you can have the Wizatrons work for you!**

Unless you choose to modify the default values, the Browse Wizatron has all of the information that it needs to generate the appropriate procedure. Before we generate, let's examine the Wizatron's application view:

9. **Choose** **Window** ➤ **Application** from the Wizatron design menu.

The big difference here from the previous tutorials is that the Application View is only showing you a single procedure component and other related Wizatron components (list box, locator, etc.) that it needs to generate *BrowseAuthor*.

10. **Press** the **Generate** button on the toolbar to generate your browse procedure and exit the Wizatrons.

11. **Make** and **Run** the application at this time.

When the *WT_THREE* application is loaded, review the *BrowseAuthor* procedure and confirm that its look and feel is different from the other browses in the program.

This completes the Procedure Wizatron tutorial!

Chapter Summary - What Did I Just Do?

Obviously, you could spend more time with the Wizatrons to polish the look and feel of any application. To summarize what we have just learned in this chapter:

- You examined the Dictionary Wizatron and the importance of providing necessary information to the application and procedure based Wizatrons. This includes the concepts of interest levels, population, edit-in-place, and the third normal form.
- You created an application using the Application Wizatron and the *STANDARD* style folio using the Quick Start View technique.
- You created an application using the Application Wizatron Wizard View and the *STANDARD* folio file provided.
- You created a second application, using a different style folio file (*CUSTOM*) and examined the differences with the initial *STANDARD* folio. This affected the visual presentation of your application.
- Finally, you used the Browse Procedure Wizatron to add a procedure to an existing application. Your browse had a different look and feel from the other browses in your application as a direct result of applying a different style folio.

3 - THE WIZATRON DESIGN ENVIRONMENT

The following chapter covers the Wizatron design environment. While this is *not* specific to any style folio, you may find it helpful to have the Application Wizatron loaded while reviewing this material.

Wizatron Views

The Wizatron Design Environment is comprised of a simple Frame structure, a set of graphic toolbar buttons, and a combination of design windows, or *views.*

Wizard View

A *wizard view* provides a series of Wizatron components that are organized into pages and displayed in a linear fashion. You can control components of the next page based on the answers you provided on previous pages.

The **Next** and **Back** buttons allow navigation between the primary Wizatron components. Based on the amount of detail you have chosen to view, there are also navigation buttons which control movement within pages of certain Wizatron components.

Application View

The *Application View* provides a list of the specific Wizatron application components that are generated into the active application (the application loaded when you launched the Wizatron).

These components usually exist based on the primary property values entered into designated Application Components. For example, a number of Browse component procedures(nodes) are generated based on the number of files you specify in the *Frame.BrowseFiles* property. Its contents are derived from the active style folio.

Style View

The *Style View* displays the style components used as the building blocks and foundations of the *application view* (which in turn is used to generate your application). Style components are not application specific. They are designed to be applied to a group of applications that share a common set of attributes and features.

Quick Start View

Finally, the *Quick Start View* is a special view that, at first, closely resembles the Wizard View. However, the wizard view is a tabular projection of the *application* view, while the Quick Start view is a tabular projection of the *style* view. In addition, the Quick Start View provides a much simpler set of prompts, while including more text based information for each one. Its primary function is to populate multiple Wizatron components with a “meta” set of choices, and to quickly generate a Wizatron based application.

ToDo Procedures Window

In the lower left of the Wizatron design environment, a special window is displayed which allows further control of the Wizatron's generated output.

Procedures listed in this window are simply identified by the Wizatrons, but not yet included in the generated output. To activate these procedures, highlight the procedure that you wish to include and press the ▲ button.

If you do not want a procedure generated, leave it in the ToDo Procedures list.

Other options include:

Multi-select	Press CTRL + CLICK
Select All	Press the + button.
Unselect All	Press the − button.
Help	Press the ? button.

File Menu

Application ➤ Load Application

Loads a set of specific application components to apply to the current Wizatron design session. The default extension for this file type is ***.APT**.

Application ➤ Save Application

Saves the current set of changes that were made to the application tree that was first loaded by the current wizatron design session. The default extension for this file type is ***.APT**. This option is disabled *until* there are changes to be saved. This option is available on the Wizatron environment toolbar.

Style ➤ Load Style

Loads an existing set of default style components to apply to the current Wizatron design session. The default extension for this file type is ***.STY**.

Style ➤ Save Style

Normally, it is better to modify and save large scale changes to your style components in the Wizatron Configuration Utility. This option is provided here to give the developer additional flexibility when smaller changes to the style component need to be made. The default extension for this file type is ***.STY**. This option is also available on the toolbar, and is enabled only if there are any changes to be saved.

When saving a style file to an existing style file name, the set of style components changed in the active design session are merged into that file. If a file name does not exist, the file is created and only the style components new changes are written to that file.

Generate

Starts the Wizatron engine to generate an output which is automatically imported into the Application Generator. If the button is disabled, there are certain required properties (indicated in red) that need to be enetered before generation can occur. This option is also available on the toolbar.

Quit

Allows you to exit without saving any changes or generating your Wizatron settings into your current application.

Edit Menu

Undo/Cut/Copy/Paste

As in all standard Windows-based applications, the ability to cut, copy and/or paste selected property values is available in the Wizatron design environment.

Options Menu

Show Wizatron Image Toolbar

This is a toggle menu item which displays or removes the graphical image and property key.

Show Controls Toolbar

This is a toggle menu item which displays or removes the toolbar and associated buttons.

Message Display Threshold

This option allows you to control the amount of messages displayed in the Wizatron's progress window. As you increase the amount of message generation, the generation process time also increases in proportion.

Reset Factory Importances

The default style folio you load when starting a Wizatron session contains default importances for all primary properties. In the Wizatron Configuration Utility, you have the option to manually set these values. Other times, the Wizatron will automatically modify importances based on the frequency (or infrequency) that you access a particular property.

Selecting this option will restore all of the default importances to your current design session.

Folio Menu

The Folio Menu allows you to add, combine, remove, load, and save your Wizatron style files into an active style folio file. Style folio files default to an extension of *.STF.

A style folio is simply a collection of style files. You may have up to 255 style files in a single folio. The idea here is to start with a base, or foundation style and *incrementally* add styles that give added features to your generated applications.

A simple example of this would be to add the *WEB.STY* file to the *Standard* folio, enabling your applications to incorporate the Clarion Internet Connect Extension Template. In fact, we have already done that for you in the *Web* style folio. (*WEB.STF)*

Current Folio Management

Edit

Each style file you have loaded is represented by a style folio entry. You can edit each folio entry as follows:

Name	The literal description of a style file you have loaded. This name displays in the folio list. For example, the *STANDARD.STY* file can be represented as 'TopSpeed Standard Style File'
Filename	This is the name of the style file found on disk.
Requires	A style folio has the ability to force one particular style file to require the componets and properties of another. If that style file is removed from the folio, the dependent style file is not loaded.

Load

This option allows a direct load of a style file to be applied to the active folio. Entries of a loaded style file are prepended with a FILE: identifier. You can later modify this using the **Edit** option.

Save

To update any changes to your folio session, use this option to update the contents of the current active style folio. Default file extension is *.STF.

Advanced Folio Management

The Advanced Folio Management section allows you to merge, change order, and add or remove any style file that is part of the current style folio.

Move Up/Move Down

Use the **Move Up/Move Down** buttons to better organize your style folio. If your style folio begins to grow larger, the more important style files can be placed at the top of the list.

The order of style files also determine what is loaded by the Wizatrons first. Any subsequent styles in the list with similar components will override the previous component property values.

Combine

This powerful feature allows you to combine existing folio entries into a single new entry. In effect, you can then save this entry as a new style file (an amalgamated file). This technique can be used to combine a group of smaller styles into a single entity, which is then easier to load into new style folios**.**

Delete

Press this button to delete the current highlighted folio entry. This will remove the related style file from the current folio. This delete is not permanent until the style folio is saved.

Add

Allows you to add additional style files into the folio. Designate the physical style file to be loaded, and whether or not it requires the existence (presence) of another style folio entry.

Edit

This option is a duplicate of the **Edit** option found in the *Current Folio Maintenance* tab section.

Save To

Allows you to save the incremental style component changes to the highlighted style file in the folio list.

Save

To save your current folio set of entries to an existing or new style folio, press this button.

Window and Help Menus

The Window and Help Menus in the Wizatron interface provide standard options for tiling, cascading, arranging icons of the different views, a version information screen, and standard help features.

Create New View

Allows you to create an additional view, giving you the ability to examine and edit different parts of the same style or application view. For example, it may be useful to copy a property value from one component to another where there isn't a suitable line of inherence in the tree; like a text property or alignment handle. This option is also available directly from the Wizatron toolbar menu.

Web Sites... ➤ Change URL List...

For updates and additional information not available at press time, this option provides links to Topspeed and 3rd-party Wizatron web sites. You can also add or update the default URL information:

Application View Popup Menus

In the Wizatron Design Environment's Application View, there are two list box controls, each with entry type, find anywhere locators:

The left list box contains a relational tree of application components. A RIGHT-CLICK on any component displays the following popup menu.

Application Components Popup

Find

Use this option to quickly locate an application component. The value entered will select the component that most closely matches your selection criteria.

Delete Branch

One power of Wizatron technology is its ability to remove pieces of selected application design components. Select this option if you wish to delete *only* the component branch that is currently selected in the application view. Only applications generated in the future will be affected.

Expand Branch/Contract Branch

The application view components list box is formatted to display as a collapsible hierarchical list. The plus (+) sign indicates a collapsed level that expands when you CLICK on the plus (+) sign. The Expand Branch menu option is equivalent to this action.

Conversely, the minus (-) sign indicates an expanded level that collapses when you CLICK on the minus (-) sign. The Contract Branch menu option is equivalent to this action.

Property Popup

In the application view, the list box on the right side of the window displays the primary properties of the selected component on the left. A RIGHT-CLICK on any property displays the following popup menu:

Find property

This option allows you to search and locate a property in the list. The search criteria you enter finds the first and closest matching property value. There are additional options to control the direction and case sensitivity of your search.

Revert to inherited/default value

This menu option allows a property that has been overridden to restore its original inherited or default value.

Style View Popup Menus

Similar to the Wizatron Design Environment's Application View, the Wizatron Style View contains two list box controls, each with entry type,find anywhere locators:

The left list box contains a relational tree of style components. Each component's associated properties are shown in the right list box.

Style Components Popup

A RIGHT-CLICK on any style component displays the following popup menu.

Find

Use this option to quickly locate a style component. The value entered will select the style component that most closely matches your selection criteria.

Expand Branch/Contract Branch

The style view components list box is formatted to display as a collapsible hierarchical list. The plus (+) sign indicates a collapsed level that expands when you CLICK on the plus (+) sign. The Expand Branch menu option is equivalent to this action.

Conversely, the minus (-) sign indicates an expanded level that collapses when you CLICK on the minus (-) sign. The Contract Branch menu option is equivalent to this action.

In addition to the menu items just described, the following items are only found in the Wizatron Configuration Utility when you RIGHT-CLICK over a selected style component:

Delete branch

To remove the selected style component from the active style, choose this option.

New Property

This option shifts focus to the selected components property sheet, where you can enter the name of a new property that you may need to reference. When you enter a property name, it is sorted in alphabetical order in the property window. You may then assign it a value, or RIGHT-CLICK to modify its sub-properties.

New Acorn

An acorn is a special type of property that allows new Wizatrons components to be generated based on their defined values. Because of their importance, a special dialog is displayed to help quick start an acorns' creation.

New Property Name
: Sets the primary property name and *Acorn.Name* sub-property.

Wizatron Type
: Populates the *Acorn.Wizard* sub-property with the name of the Wizatron that will be created when this property is enabled.

Set Properties Populates the *Acorn.Context* sub-property. This is normally a comma delimited set of values designed to populate other properties which are created with this acorn.

Always Deploy Sets the *VALUE* sub-property with a context string. The choices are:

YES	(Always ON)
NO	(Always OFF)
File Dependent	(Based on a file's existence)
Context Dependent	(Based on an evaluated expression)
Whim	(Controlled at design time)

Create New Defaults Based On...

Use this option to derive a new set of property defaults from the selected style component and create a new style component based on those defaults. A compound name is generated for the new default style (e.g., TOPSPEED:Button#Close).

Style Properties Popup

In the property list box of the Wizatron Configuration Utility, the following popup menu items are available:

Find property

This option allows you to search and locate a property in the list. The value entered finds the first and closest matching value. There are additional options to control the direction and case sensitivity of your search.

Revert to inherited/default value

This menu option is enabled for properties that have been overridden and restores them to the original inherited or default value.

Edit Selected Property

By choosing this option, the selected property's sub-properties display in a list for viewing and editing.

Delete Property

Choose this option if you want to remove the selected property from the list for a particular style component.

Special Menu Items

There are two special menu items only found in the Wizatron Configuration Utility:

Property Documentation

File ➤ Save Documentation

This option allows you to generate an HTML document which contains detailed information of the components and properties found in an active style file. Each primary property contains a sub-property named *Documentation*, and values entered there are exported to the HTML document (See the *Style Guide* chapter).

Hidden Properties

Options ➤ Show Hidden Properties

This toggle item reveals properties whose hidden sub-property is set to ‘Yes’.

The reason for hiding these properties is two-fold. First, to limit the clutter of properties that are infrequently modified by the developer. Second, there are some properties hidden that can be sensitive to incorrect modifications. However, it is recommended that these hidden properties be displayed when attempting to modify a style folio’s style components. This will give you a full picture of its features and capabilities.

4 - STATIC ELEMENTS

The following chapter deals with the structure and usage of the Application Wizatron components and their associated properties. While this chapter is *not* specific to a single style folio, you may find it helpful to have the standard style folio loaded into the Wizatron Configuration Utility while reviewing this material.

Naming Conventions

The Wizatron name is used to cover a number of diverse development tools. The Application and Dictionary Wizatrons are for use in the Application Generator and Dictionary Editor respectively. The Wizatrons discussed in this chapter are Application Wizatrons.

Wizatrons are actually classes of compiled code. Although there is a specific process for a user to add or incorporate new Wizatrons into the Clarion IDE, this chapter focuses on the Wizatrons bundled with Clarion. These Wizatrons and their associated components are referred to as *static elements*.

In fact, it is possible to add new Wizatrons to your system using our drop in DLL technology. This technology spawns one important side effect that we *do* need to consider.

Each Wizatron is identifiable by two distinct names. Its physical (or specific) name and its logical name. The physical name of a Wizatron is *set into the code*, and cannot be modified. The logical name is specified in *c5wiz.ini*, and can be changed:

```
[Wizatron Version] !the C5WIZ.INI file
Version=10
HelpFile=C:\CLARION5\BIN\WIZATRON.HLP

[WizardDLLs]
1=WIZTPS.DLL

[WizardMappings]
1=Application,TOPSPEED:Application !logical name, physical name
2=Frame,TOPSPEED:Frame
3=Menu,TOPSPEED:Menu
4=Browse,TOPSPEED:Browse
5=Report,TOPSPEED:Report
6=List,TOPSPEED:ListBox
7=Button,TOPSPEED:Button
```

The physical name specifies the given code-implemented Wizatron, while the logical name is the name of the Wizatron with responsibility for a given task. As an example, the physical name may be “Jim DeFabia” but the logical name is “author”.

In TopSpeed's Wizatrons, the physical name usually has **TOPSPEED:** prepended to the logical name. Thus the default assignment for *Browse* (logical name) is *TopSpeed:Browse* (physical name).

When naming a Wizatron, you should always consider whether you mean a specific Wizatron (the browse shipped by TopSpeed) or the logical one (the browse which may be a third party Wizatron).

Correct use of these names allows you to integrate third party extensions throughout your application with one change to the Wizatron INI file.

Categorization and Structure

If you examine the standard style file (*STANDARD.STY*) in the **style view** tree, you will find that the *ApplicationComponent* node is at the same level as the *Application* node.

Expanding the ApplicationComponent(AC) node you will find that the AC Wizatrons fall into three broad categories:

- visual components,
- template components, and
- others

This distinction is vital since it corresponds to a division that templates have traditionally concealed. For example, when you selected the Application Wizard in previous versions of Clarion, the detail behind this wizard was hidden from view. Modifications of the wizard itself required hand coding. Wizatrons promote meta-programming by eliminating the direct need to modify a template. Instead, they allow configuration in the Wizatron design environment prior to code generation.

- Visual components relate to the controls or windows you populate in the Window Formatter. In the Wizatron tree views, they branch into two categories: windows and controls. The control branch then spreads further, with Wizatrons corresponding to each visible control type.
- Template components correspond to the control templates (or procedure extensions) that are attached to visual controls. In the Wizatron tree views, they branch into procedures and the various control and extension templates.
- The others are typically “logic-bundles” that give you single places to control the interaction of the visual and template components. In the Wizatron tree views, components include the TopSpeed:MenuManager, the TOPSPEED:LocatorClass, and several others.

Values (Properties) and Derivation

The Wizatrons, from a visual perspective, are collections of values called *properties*. The versatility and power of the Wizatrons comes from how these values, or properties, are computed.

Think of Wizatrons as a series of default values for template prompts and control properties. The style view contains rules for placement of windows and controls, and these rules are evaluated during code generation. Therefore, instead of hard coding these placement values, the Wizatrons use heuristic calculations to determine window and control positions.

If you select a Wizatron component (e.g., TopSpeed:Control) in the Style View, you see on the right all of its pertinent properties and their values:

The icons beside each property signify the following:

Standard
This property does not appear in the parent Wizatron. For example, TOPSPEED:Window has a specific CENTER property, but its parent Wizatron TOPSPEED:VisualComponent does not (all visual components do not share a CENTER attribute). The CENTER property only applies to the window.

Inherited
The parent has a specific value for this property and the current Wizatron is using that value. (e.g., *At.Height*).

Overridden
The parent has a value for this property and so does the current Wizatron. It does not necessarily mean the local value is different, simply that it has one. This distinction is important if you later change the parent value. When you do, the current Wizatron component would also reflect the change, unless it is overridden.

These property types are examined in more detail in the *Advanced Tutorial* chapter.

Remember, these property settings are the defaults used when you create an *instance* of this Wizatron for the specific application. Specifically, there could be thousands of TOPSPEED:BUTTONs in the Application View, but only one in the Style View.

In the Application View tree there is another specific type of property which is exclusive to that area:

Default
The value used in the Application view tree is the value found in the Style view tree. Any other symbol in the Application view means that either the Wizatron's *internal code* or the user's action has made this a specially modified value for some reason (usually related to a dictionary setting).

There are also two special icons which help to identify important characteristics of a Wizatron property. They are:

Acorn
An acorn is a property, that when set to a particular value, creates additional Wizatrons dependent on the content of the property values. For example, *BrowseFiles* is an acorn. Based on the number of files you specify, it creates that number of Browse Procedures. Each Browse Procedure can possess a variety of acorns that can create Update Procedures, Lookup Buttons, and so on. Although many acorns are boolean, they are not restricted to that particular type (e.g., MenuItem#Splash - Item.Procedure).

Dynamic
This property identifies DYNA properties. Simply, properties with the active DYNA attribute inherits their values from a "parent" Wizatron, which is loosely any Wizatron located above itself on the tree. For example, *BrowseAuthor* is a parent Wizatron created by the *BrowseFiles* acorn, and has a DYNA *ParentFile* property set to AUTHOR. Any child Wizatron belonging to the *BrowseAuthor* Wizatron, and possessing the *ParentFile* property (like *UpdateAuthor*), will also have AUTHOR set as its parent file default value.

Derived Defaults

Another feature contained in the Wizatron interface gives you added design control. Consider the TOPSPEED:Button Wizatron:

There are many properties that are consistent for all of the button controls in a style. However, at times there will be special modifications you will need to make that fall into specific sections. For example, your default style could have different types of **Close**, **OK** and **Update** buttons.

Deriving a new set of Wizatron defaults eliminate the need to create (or hand code) a new Wizatron.

RIGHT-CLICK on any Wizatron style component, and select the **Create New Defaults Based On** option from the pop up menu:

You specify a custom name for your new default set. A compound name is then generated for the default style (e.g., TOPSPEED:Button#Close). You are deriving properties from the TOPSPEED:Button Wizatron, but using a new Close set of defaults. The same original property values are used, but new values can be entered to override these original values.

It is also possible to derive a new set of Wizatrons from one that has been already derived. In the example above,TOPSPEED:Button derived TOPSPEED:Button#Close. From TOPSPEED:Button#Close, we can derive TOPSPEED:Button#Close#CloseAll and TOPSPEED:Button#Close#Shutdown. A whole set of derived families can be created using this technique.

Tip: **The technique of deriving a new Control Template and adding a custom acorn property that references that Control Template, is one of the easiest and most powerful ways of extending the functionality of an existing style folio.**

You can review and practice this technique later in this handbook. (See the *Advanced Tutorial* chapter)

Space Allocation and Space Allocation Properties

Central to the Wizatron technology is an advanced space (control layout) allocation algorithm. This algorithm takes the various visual components that comprise a window, and populates them in an *optimum* and consistent pattern. Of course, what *optimum* means is highly subjective. Therefore, a large number of properties are designed to allow the style-sheet user to configure how screens should look for their specific environment.

AT.(property)

The AT. group of properties define the extent (size, area, proportions) of a control or window.

AT.WIDTH
AT.HEIGHT

Define the preferred size of a control. If left blank these values are computed automatically by the Wizatron depending upon the data defined within the control instance itself (such as a string's *text* property).

One challenge is that a collection of controls, at their ideal size, will not necessarily fit on an ideally sized window.

In order for controls to align themselves properly, it is possible that some of these ideal sizes just the task extremely complex. To simplify this process, each visual component has additional properties:

AT.NORTHMARGIN
AT.SOUTHMARGIN
AT.EASTMARGIN
AT.WESTMARGIN

Used in conjunction with *AT.WIDTH* & *AT.HEIGHT* to specify an outside boundary, or zone, around a particular control. You can use this to control spacing on a control by control basis in Windows, Regions, Groups, and Options. For example, you can specify that your update buttons are grouped closely together, while placing a little more distance between the OK and Select button controls. These controls are grouped by the Wizatron using a group control.

AT.NORTHOFFSET
AT.SOUTHOFFSET
AT.EASTOFFSET
AT.WESTOFFSET

These are used in conjunction with each control to specify the amount of offset in dialog units from a control's ideal placement.

AT.MINWIDTH
AT.MAXWIDTH
AT.MINHEIGHT
AT.MAXHEIGHT

These properties define the boundaries of how much the Wizatron can stretch or shrink a control from its ideal size to a size that fits in better with its siblings.

AT.COSTFACTOR

Defines the flexibility a control has to move towards its minimum or maximum boundaries. The default is 1, and any positive numeric value, including fractions, is valid.

For example, tabs on a sheet can be different sizes due to the space that the controls on them take up. If it is more important that the size of the controls on those tabs are not reduced (because of the tab cost factor), then you might decrease the CostFactor for tabs to 0.5 or perhaps zero to allow the tabs to adjust their size to the controls populated within them.

ALLOCATION.(property)

Unguided, the Wizatron could (and would) simply scatter controls over the screen in a haphazard manner. The allocation properties are a way of specifying to the Wizatron rules to follow when populating controls.

ALLOCATION.TAB

Used to specify the tab label of the tab on which this control should be populated. No value (blank) means populate this control on the window.
Toolbar means populate this control on the toolbar.
SHEET means populate this control on the sheet.

A tab element is automatically created for each unique *Allocation.Tab* value specified in the application tree.

ALLOCATION.ZONE

Each space allocator (window / tab / toolbar, etc.) is divided into nine zones corresponding to the eight points of the compass plus the center. The zone is the area in the parent where the control will (roughly) be placed.

ALLOCATION.PRIORITY

Within a given zone, controls are populated in a numeric sequence accordingly to the diagram shown above. The Priority property is used to specify which controls are placed or populated first within the zone.

ALLOCATION.XPOS
ALLOCATION.YPOS
ALLOCATION.WIDTH
ALLOCATION.HEIGHT

It is common to populate controls in a certain position relative to each other. These properties provide this mechanism.

If two (or more) control instances have the same label (with a length greater than one letter) in these properties, the space allocator logic ensures that the resultant values placed in those properties are identical.

For example, if you want all entry fields to have the same x position, you make the *Allocation.Xpos* for all entries to be *'FEntryFieldXPos'*.

Note that *EntryFieldXPos* is a unique label and has no other meaning. However, the first character of the label does have meaning.

F Specifies to use the first possible value in a zone.
L Specifies to use the last possible value in a zone.
N Specifies to use the minimum possible value in a zone.
X Specifies to use the maximum possible value in a zone.

Therefore, to make all entries the same width you could use:

FEntryWidth (all the same width as the first entry populated)
LEntryWidth (all the same as the last)
XEntryWidth (most probable, all same as the biggest)
NEntryWidth (all same as the smallest)

Dynamic Property Values

Since Wizatrons are collections of visual properties, then setting their proper values is of vital importance. The simplest method of doing this is to type in a value. However, the real power comes when you use the available set of *expansion macros*. Expansion macros and built-in functions are statements recognized by the Wizatrons. They allow property values to become *dynamic* in nature.

Expansion macros

Let's examine the *Use.Label* property found in the MenuItem Wizatron:

?%Item.Procedure%

the expression between the %% is a **macro substitution**. It means look for a property called *Item.Procedure* and place its value in its place.

Item.Procedure contains the name of a procedure to call when the menu item is selected. When the *Item.Procedure* gets set to 'BrowseNames' the *Use.Label* automatically will have the value '?BrowseNames'.

Built-in functions

In addition to providing values by macro substitution, there are built-in functions to allow computations. This further extends the flexibility of dynamic values.

For example, the **text** property of a *TOPSPEED:Menu* Wizatron looks like:

&@EVAL(SUB('%USE.LABEL%',2,LEN('%USE.LABEL%')))

@EVAL means take the contents as a string and evaluate it as a Clarion expression.

Functions are evaluated using the same rules as in the Clarion language.

The expression above means take all but the first character of *use.label* (which will be a ?) and append it to an ampersand (to give the menu text a hot key). If the contents of the *Use.Label* property is equal to **?Browse_Customers**, the result of the expession after the Wizatron evaluation should be:

&Browse_Customers

Populator Symbol (^)

The other symbol that appears in property fields is the **^,** which means *populator*.

A visual component is typically, but not exclusively, populated from a template component. The parent of that visual component is usually the window it resides on. However, you may wish to get at some of the property values from the template component that originally populated it. This is done with **%^PropName%.**

For example, in the TOPSPEED:VisualComponent style,

INTERNAL.SEQ(%^Instance%)

designates that the control which is populated inherit its population sequence from the dictionary (where the instance value is initially set).

Built-in Macros

In the sub-properties of a multi-valued property the macro %_VALUE_% evaluates to the current value of the multi-valued property.

In addition to valid Clarion functions, there are special macro symbols and extensions which are unique to the Wizatrons.

Symbols	Return Value
%YEAR%	YEAR(TODAY())
%DATESHORT%	FORMAT(TODAY(),@d17)
%DATELONG%	FORMAT(TODAY(),@d18)
%TIMESHORT%	FORMAT(CLOCK(),@t7)
%TIMELONG%	FORMAT(CLOCK(),@t8)

Available Expansion Macros Listing

@BESTWINHEIGHT

@BESTWINHEIGHT(*factor*)

@BESTWINHEIGHT
Returns the number of dialog units required to fill the factor percent of the screen height at the current resolution. Normally implemented with Wizatron **AT** properties.

factor A numeric constant from 1-100.

```
EXAMPLE:
800 X 600 Resolution: @BESTWINHEIGHT(50) returns 300
1024 X 768 Resolution:@BESTWINHEIGHT(50) returns 384
```

@BESTWINWIDTH

@BESTWINWIDTH(*factor*)

@BESTWINWIDTH
Returns the number of dialog units required to fill the factor percent of the screen width at the current resolution. Normally implemented with Wizatron **AT** properties.

factor A numeric constant from 1-100.

```
EXAMPLE:
800 X 600 Resolution: @BESTWINWIDTH(50) returns 400
1024 X 768 Resolution:@BESTWINWIDTH(50) returns 512
```

@COUNT

@COUNT(*property*)

@COUNT Returns the number of instances of a property value.

property A valid property name (normally multivalued).

```
EXAMPLE:
RECORDORDER('@EVAL(CHOOSE(@COUNT(SortOrders)=0))')
```

@DEFAULT

@DEFAULT(*property*)

@DEFAULT	Returns the default value of a given property, even if it is locally overridden. Normally used when you need to extract a value originally set higher up the Wizatron tree.
property	A valid property name.

```
EXAMPLE:
Property('@DEFAULT(Property)')
```

@DESCRIPTION

@DESCRIPTION(*string*)

@DESCRIPTION	Returns the description contained in the associated dictionary of a particular field or file.
string	A fully qualified field name, file name, or expansion macro.

```
EXAMPLE:
@DESCRIPTION(%PrimaryFile%) returns 'My dictionary file description'
@DESCRIPTION(%^Field%) returns 'This is my field description'
@DESCRIPTION(Author.AuthorName) returns 'Enter the full name'
```

@EVAL

@EVAL(string)

@EVAL	Evaluate string as a Clarion expression.
string	A valid Clarion expression.

```
EXAMPLE:
@EVAL(SUB('%USE.LABEL%',2,LEN('%USE.LABEL%')))
returns 'MYLABEL' where %USE.LABEL% = '?MYLABEL'
```

@EXTRACT

@EXTRACT(AcornName,Property)

@EXTRACT	Extracts a value of a Wizatron acorn property.
AcornName	The name of an acorn property (any property with the *Acorn* property set to 'TRUE').
Property	A property name in the Wizatron component that the acorn points to.

```
EXAMPLE:
@EXTRACT(MyAcorn,MyProperty)
extracts the value of property MyProperty from the Wizatron pointed to by
MyAcorn.

@EXTRACT(LookupStyle.Child.Entry^EditControl,Use.Label)
extracts the use label of the acorn Lookupstyle.child.entry. This primes
the Lookup Field Button to return focus to the proper entry control.
```

@FIELDS

@FIELDS(Filename)

@FIELDS	Returns a comma delimited list of all the fields in a file.
Filename	The label of a file in the active dictionary.

```
EXAMPLE:
@FIELDS(BOOKS) returns 'bookid,bookname,datepublished'
```

@FILES

@FILES()

@FILES	Returns a comma delimited list of all files in the active dictionary

```
EXAMPLE:
@FILES() returns 'Author,Publisher,Book' from BOOKS.DCT (the active
dictionary)
```

@FULLNAME

@FULLNAME(string)

@FULLNAME	Translates the *String* into full period syntax.
string	The field label in *prefix:label* format.

```
EXAMPLE:
@FULLNAME(CUS:ID) returns 'CUSTOMER.ID'
```

@HEADING

@HEADING(string)

@HEADING	Returns the column heading from the active dictionary for the selected field. Normally found in the *TOPSPEED:ListBoxColumn* Wizatron component in the *Column.Header* property.
string	A fully qualified field name or valid macro.

```
EXAMPLE:
@HEADING(%Column.Id%) returns 'Book Title' (value stored in dictionary)
```

@INDEX

@INDEX(property,value)

@INDEX	Returns the numeric order of a property value in a multivalued list.
property	A valid property name (normally multivalued).
value	A string value to search for in a multivalued list.

```
EXAMPLE:
@INDEX(StandardMenuItems,Quit) returns '1' (first item in list)
```

@INHERIT

@INHERIT(property)

@INHERIT	Returns the value of the named property underneath the locally defined property. It returns whatever would have been retrieved using the inheritance mechanism if there had been no local value defined.
property	A valid property name.

```
EXAMPLE:
ALLOCATION.WIDTH('@EVAL(CHOOSE(%Memo%=1,''A'',''
@INHERIT(Allocation.Width)''))')
```

@KEYFIELD

@KEYFIELD(keyname,[order])

@KEYFIELD	Returns the fully qualified field name of a key's populating element.
keyname	The dot syntax label of a key in the active dictionary.
order	The relative populated order of the key field. If order is not referenced, it is assumed to be one(1).

```
EXAMPLE:
@KEYFIELD(BOOK.BYID) returns 'Book.BookId'
@KEYFIELD(CUSTOMER.FullNameKey,2) returns 'Customer.Firstname'
```

@KEYS

@KEYS(filename)

@KEYS	Returns a comma delimited list of all keys in the selected file.
filename	The label of a file in the active dictionary.

```
EXAMPLE:
@KEYS(BOOK) returns 'ById,ByTitle,PublisherKey'
```

@NAME

@NAME(string)

@NAME	Translates the *String* into prefix:label format.
string	The field label in full period syntax format.

```
EXAMPLE:
@NAME(CUSTOMER.ID) returns 'CUS:ID'
```

@PARENT

@PARENT(property)

@PARENT	Returns the property value of the dynamic parent, even if it is overridden locally. Normally used in ALLOCATION type properties.
property	A valid property name.

```
EXAMPLE:
AT.EASTMARGIN('@EVAL(@PARENT(ALLOCATION.XSPACING)*2)')
```

@PROMPT

@PROMPT(string)

@PROMPT	Returns a prompt value from the active dictionary from a specific field.
string	A fully qualified field name or valid macro.

```
EXAMPLE:
@PROMPT(Author.AuthorName) returns 'Author:' from the BOOKS.DCT
```

@RELATED

@RELATED**(**filelabel**)**

@RELATED	returns a comma delimited list of files related to a specific file in the active dictionary.
filelabel	The label of a file in the active dictionary.

```
EXAMPLE:
@RELATION(AUTHOR) returns 'Book,Publisher'
```

@RELATIONEXPRESSION

@RELATIONEXPRESSION**(**primarylabel, foreignlabel**)**

@RELATIONEXPRESSION	Returns an equivalence expression that can be used to express the relationship between two files. Used in the case where you cannot use a simple range-limit on a child browse (See the *Filter* property of *BrowseSortOrder*).
primarylabel	The label of a primary file in the file relationship.
foreignlabel	The label of the secondary file in the file relationship.

```
EXAMPLE:
@RELATIONEXPRESSION(AUTHOR,BOOK) returns 'Author.BookId = Book.BookId'
```

@RELATIONFIELD

@RELATIONFIELD(primarylabel, foreignlabel,[order])

@RELATIONFIELD
Returns the fully qualified name of the field used in the foreign key of a file relationship in the active dictionary.

primarylabel The label of a primary file in the file relationship.

foreignlabel The label of the secondary file in the file relationship.

order The relative populated order of the key field. If order is not referenced, it is assumed to be one(1).

```
EXAMPLE:
@RELATIONFIELD(AUTHOR,BOOK,1) returns 'Book.BookId'
```

@RELATIONFOREIGNKEY

@RELATIONFOREIGNKEY(primarylabel,foreignlabel)

@RELATIONFOREIGNKEY
Returns the name of the foreign key in the two files' relationship defined.

primarylabel The label of a primary(one) file in the file relationship.

foreignlabel The label of the secondary(many) file in the file relationship.

```
EXAMPLE:
@RELATIONFOREIGNKEY(AUTHOR,BOOK) returns 'Book.ByAuthorKey'
```

@RELATION

@RELATION([filelabel, filelabel])

@RELATION Returns the relation between two specific files defined in the active dictionary.

filelabel The label of a file in the active dictionary.

```
EXAMPLE:
@RELATION(BOOK,AUTHOR) returns 'Author<Book'
```

@REPORTCONTROLTYPE

@REPORTCONTROLTYPE(fieldlabel)

@REPORTCONTROLTYPE
Returns the report control defined by the active dictionary of this specific field.

fieldlabel A fully qualified field name.

```
EXAMPLE:
@REPORTCONTROLTYPE( Author.AuthorName) returns 'STRING'
```

@SCREENCONTROLTYPE

@SCREENCONTROLTYPE(fieldlabel)

@SCREENCONTROLTYPE
Returns the window control defined by the active dictionary of this specific field.

fieldlabel A fully qualified field name.

```
EXAMPLE:
@SCREENCONTROLTYPE( Author.AuthorName) returns 'ENTRY'
```

@SCREENPICTURE

@SCREENPICTURE(string)

@SCREENPICTURE
Returns the default screen picture from the active dictionary of this specific field.

string A fully qualified field name or valid macro.

```
EXAMPLE:
@SCREENPICTURE(Author.AuthorName) returns '@S20'
@SCREENPICTURE(%^Field%) returns '@S20'
```

@USER

@USER(string, attribute, [position])

@USER	Extracts the value of the *attribute* from the User Options section of the active dictionary.
string	A fully qualified field name or valid macro.
attribute	A valid name found in the User Options section of the active dictionary. If an *attribute* exists, but there is no value assigned, then the value is implicitly taken to be 1.
position	A numeric value. If the attribute is multi-valued, *position* identifies the specific attribute from the comma delimited set.

```
EXAMPLE:
In the BOOKS.DCT, the following attributes are defined in the User Options
section of Book.AuthorId (the AuthorId field from the Book file):
'DISPLAY(Author,Author),INTEREST(5),MYCUSTOMOPTION(1,2,3)'
@USER(Book.AuthorID,INTEREST) returns '5'
@USER(Book.AuthorID,MYCUSTOMOPTION,2) returns '2'
```

Static Sub-properties

You may think of a sub-property as a ‘meta’ property, properties of a property, or a hint as to how the specific component’s *primary* property is used and behaves (In the window shown below, *BUTTONCONTROL* is the primary property). The sub-properties unleash some of the most powerful aspects of the Wizatron technology.

Note: Sub-properties can only be viewed and/or modified through the Wizatron Configuration Utility.

In the Wizatron configuraton utility, if you RIGHT CLICK a property and select **Edit Selected Property** you are presented with a list of **sub-properties**. Here is a sample of the sub-properties window:

The following list of available sub-properties are not totally static (there can be other sub-properties provided by third party Wizatrons). The following information documents the most common sub-properties available in the TopSpeed Wizatrons.

Acorns

When set to **YES**, specifies that the primary property is an **acorn**. Acorns are extremely powerful. If the *value* of a primary property that is an acorn is also set to true, a new Wizatron instance is produced (whose exact nature is controlled by the *other* acorn. properties).

If the property is also multi-valued then a whole *family* of new Wizatron instances may be produced.

A great number of primary properties use the ACORN attribute. A partial listing is shown here:

```
PostLookupProcedure,DisplayColumns,Controls,Tabs,Toolbar,Sheet,Resizer,
ToolButtons,Image,Developer,Heading,StandardMenuItems,Menus,
WindowManager,BrowseBox,CloseButtonControl,SaveButtonControl,
CancelButtonControl,Fields,ChildBrowses,MenuManager,BrowseFiles,
ReportFiles,SplashWindow,CloseButton,BrowseListControl,SortOrders,
HasUpdate,HasSelect,HasQuery,HasToolbox,InsertButtonControl,
DeleteButtonControl,ChangeButtonControl,UpdateProcedure,ButtonControl,
UpdateProcedure,LookupStyle.Child.Combo,LookupStyle.Child.Button
```

Acorn.Context

A Wizatron instance generated as an acorn automatically inherits all the values of the properties of the parent Wizatron. However, the *Acorn.Context* allows specific properties in the child Wizatron to be set upon its creation.

The context value must be a comma delimited attribute list. For example, if you had an acorn that wanted to generate a button instance but you always want that particular button to be black (irrespective of button default settings) you would have an *acorn.context* value of:
Populator(%NAME%),COLOR.Basic(0)

> **Note:** **Without the use of Acorn.Context, modifying other sub-properties will modify all related style components. For example, if we change the color of a Close Button control, it changes the color of *every* generated Close Button of every application Wizatron using that style file.**

Acorn.InitPriority

When a Wizatron instance contains a number of acorn properties, the creation of the Wizatron instance causes a number of child Wizatrons to be created. The *Acorn.InitPriority* sub-property allows the style-sheet writer to specify the **order** in which these child Wizatrons are created. The valid range is from 1- 255. The higher numbers have higher priority and are created first. This is important if one child Wizatron relies on properties being set by other children(*See EXTRACT).* It is also important to note that an acorn with a priority greater than 128 is created *before* the context properties are set. Context is set *after* the acorn is created if the priority is less than or equal to 128.

Acorn.Name

This is the name given to the specific Wizatron instance. Wizatron instance names have to be unique among common siblings of a tree. (With multi-valued acorns, this tends to mean *Acorn.Name* is dependant upon %_VALUE_%)

```
EXAMPLE:
In ButtonControls, Acorn.Name tends to be '%ButtonControlType%Button'(take
the value of the ButtonControlType, 'Close', and append 'Button').
Therefore, this acorn branch name in the Application View is
'CloseButton'.
```

Acorn.Value

This is the value of the acorn that spawned this acorn instance. If this value is blank, than the property referenced is not acorn generated.

Acorn.Wizard

This is the name of the Wizatron type to be created if the acorn is set to **YES**. You may use logical or physical names, although logical names allow maximum flexibility.

Multi-valued acorns combined with the # capability of Wizatron naming (i.e., the naming of specific instances) can be used to provide heterogeneous lists of Wizatrons (a single style from different sources). For example, the *STANDARDMENUITEMS* property of the MenuManager has an *Acorn.Wizard* of 'MenuItem#%_VALUE_%':

This will allow a family of MenuItem Wizatron instances to be produced, each with a different set of default values and identified in a comma-delimited list:

A leading ***** in the *Acorn.Wizard* field value has special meaning. It designates that the Wizatron instance being created should have a globally unique name and should be *shared* by other acorns naming the same item. Typically this is used where the wizard names a procedure (such as BrowseCustomers) and you want these procedures shared in your application (like using BrowseCustomers to select a record, validate an entry field, etc.).

Blocker

A blocker property set to **YES** prevents the Generate button (Wizatron generation capability) from being enabled unless it has a local value assigned. This is roughly equivalent to required fields in a standard Clarion application. When a property is blocked, the prompt information is displayed in red. Blocked properties also have the highest importance applied for up front (first) viewing in the Wizard view.

```
EXAMPLE: In the TOPSPEED:ProcedureComponent, the following properties are
"blocked": Output.ProcName, Output.From, Output.Category, IsProcedure
```

CheckType

CheckType is used to express the wording used for boolean display types. Valid values here are **YesNo** and **OnOff.** If left blank, the default boolean display type is Yes or No.

Description

This sub-property contains the prompt text as it will appear on the Wizard view for the given primary property.

> **Note:** **Make sure that all visible properties shown in the Wizard View are given a logical description. If you don't feel comfortable with the prompts assigned by TopSpeed, you can modify them to any descriptive text that you want.**

DevImp / DevImp.List

These sub-properties hold the default factory settings for how important (or unimportant) a given property is judged to be. The values entered controls the position of the detail slider on which the property first appears. *DevImp* is the value chosen and *DevImp.List* are the available list of possibilities. The list of available default settings (in order of importance) is shown here:

```
ASK         (most important)
COMMON
USEFUL
DETAILED
BAROQUE     (least important)

INTERNAL    (only available in Wizatron Configuration Utility)
TRANSIENT   (only used by TopSpeed development staff)
```

Documentation

This sub-property is described in more detail in the *Style Guide* chapter. It is a text based value that describes the primary property in more detail. The information entered here displays in a tool tip in the Wizard view.

Dyna

When *Dyna* is set to **NO**, the primary property defaults to the value in the Wizatron defaults. When the *Dyna* property is set to **YES**, the primary property defaults to the value of the parent component in the Application view.

This is commonly used for visual component properties where you want a particular control to default to the font, color, etc. of its parent Wizatron control. Dyna sub-properties are identified with a circular arrow icon.

Dtype (display type property)

This sub-property controls the display type of its primary property as seen in the available views . Currently, the available values are :

CHECK
Displays a checkbox on a tab. In list boxes with Edit-In-Place, it uses a drop list control (See *CHECKTYPE*) in place of a check box.

COLOR
Uses a popup color dialog. When displayed, common color values are replaced with textual equivalents (COLOR:Red, or R:255 G:255 B:0, etc.)

CURSOR
Uses a cursor file or one of the runtime library's built-in cursors.

ENTRY
Displays an entry control.

FIELD
Uses a file schematic dialog to select a field from a file in the dictionary.

FILE
A popup file dialog to allow a file to be chosen (a file not required by the current Wizatron component).

FONT
Uses the standard popup font dialog window.

MULTIDROP
Provides a multi-select type of drop list. The “drop” list actually pops “up”.

SINGLEDROP
A single selection drop list.

Note: A multidrop or singledrop sub-property automatically generates another sub-property of *Multidrop.List* or *Singledrop.List*. This sub-property is a multi-valued symbol defining the possible values for the singledrop or multidrop sub-properties, presented in comma delimited format.

SPIN
Displays standard spin control.

RADIO
On a window an option / radio structure displays. In Edit-In-Place mode, this is translated to a drop list.

TEXT
Displays a standard text control. Used in the *Documentation* sub-property.

File.Mask

The default file mask to use in association with the **FILE *dtype***.

File.pattern

The default file pattern to use in association with the **FILE *dtype***.

Implesseq (Importance is less than the default)

A major feature of the Wizatrons is that they *learn to adapt the user interface to the mode of operation of the user*.

All properties have a default factory importance setting. If a Wizatron session completes without a given property being touched by the user, then the function specified in this sub-property is applied to create a new importance level for the associated primary property.

Impmoreeq (Importance is more than the default)

If a property is used (touched) in a Wizatron session then sub-property value is evaluated to produce a new importance level for the associated primary property.

Inherit

If this is set to **YES**, child Wizatrons inherit this primary property (see the **Stop** sub-property)

List

Set to **YES**, if this is a *.LIST* property (mainly used for internal processing efficiency)

ListFill.Auto

This sub-property is applied on an element-by-element basis to the *.LIST* and provides default candidates for the underlying property. A value of 1will cause all *.LIST* elements to be selected. A value of 0 will cause no elements to be initially selected.

A macro based upon %_VALUE_% allows a more specific selection mechanism based on a current instance value.

ListFill.Convert

This sub-property is applied normally based upon %_VALUE_% whenever a multivalued symbol member is retrieved.

```
Example - ListFill.Convert(hello:%_VALUE_%), with items 1,2,3,4 will
return hello:1,hello:2,hello:3,hello:4
```

ListFill.Func

This sub-property is used to generate a comma-delimited list which contain the default values of a *.LIST* property

Locked

If set to **YES**, the primary property is read-only.

MacroOwnerContext

This sub-property is used to evaluate a macro in the context in which the macro is defined (as opposed to locally). Therefore, if a property definition is found up the parent tree, then evaluate and apply the value(s) contained in the parent tree.

Multi

If set to **YES,** the property can have multiple values.

Output.Depends

This sub-property defines a symbol that is dependant on the multi-valued symbol specified. (e.g., The *ControlHPosition* property is dependant on the *Controls* property of the *ControlTemplate#Resize* Wizatron. If there are seven controls, there will be seven horizontal positions. The *Output.Depends* sub-property value of *ControlHPosition* is ‘Controls’.)

Output.Name & Output.Type

These sub-properties correspond to the prompt template symbol and template type as found in the Application Text file (*.TXA). *Output.Name* should correspond to a valid template prompt symbol without the leading ‘%’ (i.e., *ControlToLookup* is entered into the *Output.Name* sub-property to correspond to the *%ControlToLookup* template symbol.) *Output.Type* corresponds to the data type assigned to the template symbol (BYTE, SHORT, LONG, etc.).

It is possible using this mechanism to create Wizatrons for most simple procedure extensions and control templates and preset any values needed by the template through the Wizatron interface.

Output.Value

Allows an expression to be used to set the value of a template symbol that appears in the control template #PROMPT statement.

QuickStart

A boolean value. When set to TRUE, this property is visible in the Quick Start Wizatron view. See *WhenApply*.

Stop

The stop attribute is used to block a value that would otherwise be inherited from the parent. This is used in cases where a parent has multiple children of which only some should inherit a property. Inherit is set “on” in the parent, and stop is set “on” in the children that should *not* inherit the property value.

Type

This is the storage type of the primary property:

```
UNKNOWN (Any data type)
NUMERIC
STRING
DATE
BLOB
MEMO
BOOLEAN
NODE
```

Validation

This must be a boolean expression returning true if the property value is deemed valid. Used to validate what is entered in the property value.

```
EXAMPLE:
@EVAL(%_VALUE_% > 0)
```

Value

This is the default value of the property, the same as is visible on the main right-hand window, *except* that the display type is always implicitly ENTRY. This sub-property allows macro expressions to be specified for non-string, non-entry fields.

WhenApply

A special context string. When the property with the WhenApply sub-property is changed, the Wizatron engine processes the WhenApply value accordingly.

For example, you may have a Boolean primary property called *GenerateAll*. Its WhenApply sub-property contains the following expession:

WhenApply(GenerateBrowses(%_Value_%))

In other words, if the user wants to generate everything, all browses are generated. Setting the *GenerateAll* property to TRUE results in the *GenerateBrowses* value also getting set to TRUE.

There is a WhenApply attribute set on all *QuickStart* properties.

Dictionary Attributes

An important way of setting property values for specific Wizatron instances is from attributes within the dictionary. Many of the macro functions deal with extracting particular pieces of data from the dictionary. In addition to the built in properties, the @USER function allows arbitrary (or custom) attributes to be extracted from the *User Options* fields. While this list is open ended (new values can be added), what follows are some values that the TopSpeed Wizatrons use and the dictionary Wizatron produces:

Interest

A value from one to ten, where one is low and ten is high. A field/file/key without an interest level is deemed to have a default level of five.

This value is usually compared against a property called *InterestThreshold* to ascertain whether this field, file or key (etc) is deemed worthy of inclusion in a particular list.

For example, an address file with ‘LastName’ and ‘FirstName’ fields may have interest level set to ten(10 - high interest). A ‘State’ field may have interest level of seven(7). ‘Address’ fields use a default of five (5). Alternative delivery addresses (low priority) can be set down to 2.

In the Wizatron, a certain browse designed to select a record would have an interest threshold of 8 (to only show names). A browse with full edit capability would have an interest threshold set to 1 (to show all available fields).

Files with an interest level greater than five(5) will cause the application Wizatron to automatically populate those files into a Browse procedure.

```
EXAMPLE:
INTEREST(9)
```

Display(File,Field[,Field]*]

This option focuses on the heart of the third normal form capability of the Wizatrons (See the *Introduction and Overview* chapter).

From a field which is a foreign key component (the many side) of a file relationship, you may name the File & Field(s) which form the **textual** lookup (not the linking field) in the child file. Currently the effect of this field is:

- Browses populate textual fields from the lookup file, not the linking fields.
- Forms populate a field class which may in turn populate a drop combo, field lookup button, or validation condition containing the textual field(s).

For example, there is a many-to-one file relationship defined between a BOOK file, and an AUTHOR file. The link field name in both files is an AuthorId field. In the BOOK file *User Options* for the Book.AuthorId field, there is:

DISPLAY(Author,Author)

meaning ‘whenever the third normal form is detected by the Wizatron(s) for this relationship, substitute (populate) the Author field instead of the link field (AuthorId)’.

TextPrimary

This is a user option set on a key that signifies it should be the preferred sort order when presenting information to the end user (For example, a browse select). The key fields with the highest interest level receive the TEXTPRIMARY attribute.

Small

This attribute can be applied to any file or relation.

If SMALL is set in the *User Options* of a file, it implies that the number of records in the file is small. By default, files with SMALL in their user options are populated by the application Wizatron as a drop combo lookup (as opposed to the field lookup button).

If SMALL is set in the *User Options* of a relationship, it implies that the number of children of a given parent is small.

SMALL is interpreted by the Wizatrons to mean that reading all the records is acceptable and won’t affect performance. This can force browses using this file to load *all* records instead of the default page-loading.

NoPopulate(target)

This option takes a target parameter which defines where you *don't* want a particular field populated.

Valid values are *BROWSE* & *FORM*.

```
EXAMPLE:
NOPOPULATE(BROWSE)
NOPOPULATE(FORM)
```

Summary of Wizatron Properties

While it is evident that all of the properties documented in this chapter are entirely dynamic (i.e., defined in the style files), most of them are powerful, and open up major aspects of the Wizatron technology.

It is very probable they will appear in many types of style files.

The next chapter explores the application of these properties in the TopSpeed style files.

5 - STYLE FILES

The Wizatron Style File

The core and genesis of all Wizatron design is contained in an ASCII file of Wizatron properties and components called a *style file*.

The typical developer creates a Wizatron-based application by applying a core style folio to the Wizatron design environment and optionally including additional styles.

This chapter focuses on important documentation that is contained in each TopSpeed style file. It concludes with a special section of general guidelines, tips, and available tools you can use when creating new styles or modifying existing ones.

Documentation Strategy

The dynamic nature of Wizatron style files presents a challenge to provide meaningful on-line help and pertinent information to the developer. The solution is to store documentation for properties within the property itself.

Property documentation is available in two ways; tooltip help and exporting to HTML.

Documentation is a sub-property embedded within each Wizatron property. Moving your cursor over a property in a selected view reveals the embedded documentation in a tooltip.

The exporting feature is only available in the Wizatron Configuration Utility. Each Wizatron property's internal documentation can be exported to a HyperText Markup Language (HTML) format.

Editing Documentation

When you RIGHT-CLICK on a selected property, you can locate and edit the associated *Documentation* sub-property.

Exporting Documentation

To export all property documentation of an active style file into an external HTML document:

1. **Choose** **Application ➤ Template Utility...** from the Application Generator main menu.
2. **Select** the *WizatronConfiguration* utility from the *Class Wizatrons*, and **Press** the **Select** button.
3. When the file dialog window appears, **select** the style folio you wish to document from the list, and **press** the **Open** button.
4. In the *TopSpeed Wizatrons* main menu, **choose** **File ➤ Save Documentation...** . The following dialog window should appear:

5. You have the option of writing your documentation to a file name with an .HTML or .HTM extension. **Navigate** to a desired directory, and **type** your output file name in the appropriate entry field.
6. **Press** the **OK** button to begin the export process.

Viewing the HTML documentation

The HTML file generated by the Wizatron Configuration utility can be viewed with any common HTML browser. A sample of the output HTML is shown below.

The beginning of the HTML documentation file contains an alphabetical listing of all default style components:

Alpha sorted index of Standard sets.

TOPSPEED:Application **(Hyperlinks to detailed information)**
TOPSPEED:ApplicationComponent
TOPSPEED:ApplicationComponent#Field
TOPSPEED:ApplicationComponent#Locator

The remaining contents of this file specifically documents each individual style component and the types of associated properties.

TOPSPEED:VisualComponent **(Logical Name of Style Component)**

Inherited Properties **(A list of properties inherited by this component)**

ACORN.VALUE **(Hyperlinks to inherited properties)**
WIZATRON

ALRT **(A default property attached to this component)**

Alert attributes **(Contents of the DESCRIPTION sub-property)**
MultiValued,Inheritable **(A property's primary characteristics)**
*Data Type:*String **(Data type processed by the Wizatron)**
*Display Type:*Entry **(Data type user/developer interface control)**
*Visibility:*Detailed **(Level of importance - where this property appears in the Wizard view)**

A multi valued list of keycodes that will be used in the `ALRT` attribute of the window or control

(Contents of the DOCUMENTATION sub-property)

HTML Documentation Tags

You will notice throughout the *STANDARD.STY* file and primarily in the default *DOCUMENTATION* sub-property the use of embedded HTML tags. Although most HTML tags are supported, common tags found in the *STANDARD.STY* include:

<CODE></CODE>

Text within the <CODE> tag appears in a monospace font when viewed through a browser. This is used to identify Clarion language statements and attributes.

<SAMP></SAMP>

Direct the browser to display the specified text in a smaller font. Used to identify other properties referenced.

<EM></EM>

Emphasis. Directs the browser to italicize the rendering of text. Used to denote property categories.

<B></B>

Display text in bold.

<I></I>

Display text in italics.

<STRONG></STRONG>

Another method of using bold text.

The Wizatron Configuration Utility handles the remaining HTML formatting during the export process, to include font sizes, justification, hyperlinks and paragraph formatting.

Sample Documentation of STANDARD.STY

The following properties listed have been condensed from the exported HTML documentation. It is meant to be used as a quick reference to commonly used properties of the *STANDARD.STY* file. For complete documentation of these items and additional information, please consult the latest *STANDARD.HTM* file installed with the Wizatrons, or read the tooltip for any specific property.

This section is provided as a quick reference and is not intended to document all possible properties.

Allocation.Functional

Description: Purely functional control

Documentation:
Sometimes it is necessary to populate hidden controls onto a window simply to act as a place holder for an underlying control template. As well as marking that control as hidden, you can set this property so that the space allocator groups function together and overlays them on to the window.

Allocation.Height

Description: Space allocator's height handle

Documentation:
A string identifier used to denote attributes of different controls that must be set to identical values.

Allocation.Ignore

Description: Avoid screen allocation

Documentation:
This prevents the space allocator from affecting this control or any other control.

Allocation.Output

Description: Avoid screen output

Documentation:
This prevents the space allocator from outputting this control. It will still affect the allocation of other controls unless ALLOCATION.IGNORE is on. This is a boolean value set. Set to 1, output is not generated. Used to prevent clashes with resize strategy.

Allocation.Priority

Description: Population priority

Documentation:
A number to indicate the order in which controls should be populated on a window. A high number indicates the control is populated first.

Allocation.ShowSingleTab

Description: One-tab sheet logic

Documentation:

If this property is non-zero then a sheet and tab is populated on the window even if there is only one tab required. If set to zero then one-tab sheets are suppressed.

Allocation.Tab

Description: Tab name to allocate on

Documentation:
The tab upon which this control should be populated. Leaving this blank means the control is not place upon a tab. A value of **TOOLBAR** implies the control goes on to the toolbar. *All* means the control is placed upon the sheet (and thus all tabs).

Allocation.Width

Description: Space allocator's width handle

Documentation:
A string identifier used to denote attributes of different controls that must be set to identical values.

Allocation.Xpos

Description: Space allocator's X handle

Documentation:
A string identifier used to denote attributes of different controls that must be set to identical values.

Allocation.XSpacing

Description: Control Horizontal Spacing

Documentation:
This property is used to specify the default horizontal spacing between controls in dialog units.

Allocation.Ypos

Description: Space allocator's Y handle

Documentation:
A string identifier used to denote attributes of different controls that must be set to identical values.

Allocation.YSpacing

Description: Control vertical spacing

Documentation:
This property is used to specify the default vertical spacing between controls in dialog units.

Allocation.Zone

Description: Zone to populate control

Documentation:
All group controls (windows, tabs, etc.) are split into 9 zones corresponding to points of the compass and center. This property denotes which of those nine zones the control should be placed in.

Allocation.Zone.List

Description: Available zones

Documentation:
The list of zones available within a group control.

Alrt

Description: Alert attributes

Documentation:
A multi valued list of keycodes that will be used in the ALRT attribute of the window or control.

At.CostFactor

Description: Resize cost

Documentation:
Ideally all controls will be populated with size *At.Height* by *At.Width*. If necessary to make all the controls fit, the space allocator will shrink or expand a control towards at.minwidth/height or at.maxwidth/height. This property specifies a floating point multiplier for the cost of this control being a misfit. 1 is the default. So a value of 0.1 means this control is 10 times more likely to change size than normal.

At.EastMargin

Description: Control east margin

Documentation:
This property allows a margin between the groups constituent controls and its border on the east side.

At.EastOffset

Description: Control east offset

Documentation:
This property allows a offset to the east of a control to be specified.

At.Height

Description: Control height

Documentation:
The preferred height of the window or control. If left blank the Wizatron will compute this value based upon the other properties of the visual component using the defaults specified in the run time library.

At.MaxHeight

Description: Control maximum height

Documentation:
The maximum height that the Wizatron may make this window or control in an attempt to match constraints specified by other visual components. For controls, the cost of stretching to this size is specified by *AT.CostFactor.*

At.MaxWidth

Description: Control maximum width

Documentation:
The maximum width that the Wizatron may make this window or control in an attempt to match constraints specified by other visual components. For controls the cost of stretching to this size is specified by *AT.CostFactor*.

At.MinHeight

Description: Control minimum height

Documentation:
The minimum height that the Wizatron may make this window or control in an attempt to match constraints specified by other visual components. For controls, the cost of stretching to this size is specified by *AT.CostFactor*.

At.MinWidth

Description: Control minimum width

Documentation:
The minimum width that the Wizatron may make this window or control in an attempt to match constraints specified by other visual components. For controls the cost of stretching to this size is specified by *AT.CostFactor.*

At.NorthMargin

Description: Control north margin

Documentation:
This property allows a margin between the groups constituent controls and its border on the north side.

At.NorthOffset

Description: Control north offset

Documentation:
This property allows a offset to the north of a control to be specified.

At.SouthMargin

Description: Control south margin

Documentation:
This property allows a margin between the groups constituent controls and its border on the south side.

At.SouthOffset

Description: Control south offset

Documentation:
This property allows a offset to the south of a control to be specified.

At.WestMargin

Description: Control west margin

Documentation:
This property allows a margin between the groups constituent controls and its border on the west side.

At.WestOffset

Description: Control west offset

Documentation:
This property allows a offset to the west of a control to be specified.

At.Width

Description: Control width

Documentation:
The preferred width of the window or control. If left blank the Wizatron will compute this value based upon the other properties of the visual component using the defaults specified in the run time library.

At.XPos

Description: Control X position

Documentation:
The x position of the visual component. Usually this is left blank to be computed by the Wizatron. Hints about the required X position are specified in *Allocation.xpos.*

At.YPos

Description: Control Y position

Documentation:
The y position of the visual component. Usually this is left blank to be computed by the Wizatron. Hints about the required Y position are specified in *Allocation.YPos.*

BrowseFiles

Description: Generate browses for these files

Documentation:
This is a list of all of the files for which browses are generated with calls from the menu. Note that there may be more browses generated if they are required by other procedures that are generated.

BrowseFiles.List

Description: Available files

Documentation:
This list of all files for which it may be reasonable to produce browses. By default that is all the files in the dictionary.

BrowseListControl

Description: Identifies list control name

Documentation:
This property is used to point to the list control owned by this browse control template. It is there primarily for the convenience of the Wizatron code.

ButtonControl

Description: No Description

Documentation:
A node pointing to the Wizatron that will populate the actual button for this control template.

ButtonControlType

Description: No Description

Documentation:
The ButtonControl Wizatron is rarely populated directly, but acts as a place holder for most of the Wizatron code dealing with various button control templates. This property is used by child Wizatrons to identify themselves in many of the parent macros and source code. For example, the *CancelButton* Wizatron places **Cancel** in this field causing the *ControlTemplate* property to expand to *CancelButton*.

Case

Description: Case of entries

Documentation:
This property specifies the case sensitivity of the control. It can produce an attribute of CAP or UPR.

Case.List

Description: Available cases

Documentation:
List of possible case values. CAP for capitalize as a proper name, UPR means capitalize all.

ChangeButtonControl

Description: No Description

Documentation:
A node pointing to the Wizatron that populates the change button for this control template.

Check

Description: Is Menu item a toggle type?

Documentation:
If this property is set to non-zero then the CHECK attribute is placed on the menu item specifying that it may be on or off.

ChildBrowses

Description: Child Browses

Documentation:
This multi-valued acorn is a list of child browses to appear on the form.

ChildBrowses.List

Description: Available child browses

Documentation:
These are the possible files that can be used to make a child browse for this form. By default this is a list of the MANY sides of all of the One-to-Many relationships that the primary file participates in.

CIF

Description: SQL Connect in frame

Documentation:
This property is used to set the frame property in the template system that causes a file to be opened in the frame procedure. Generally this should be set on if you are using an SQL database as it causes the connect to happen as soon as possible, rather than being delayed until the first browse.

Color.Basic

Description: Basic color

Documentation:
This property corresponds to the first parameter of a COLOR attribute. Usually it refers to the default background color of a control. To change the foreground color use *Font.Color.*

Color.SelectedBack

Description: Selected background color

Documentation:
This property corresponds to the second parameter of a COLOR attribute. It refers to the background color of a control when the control is selected.

Color.SelectedFore

Description: Selected foreground color

Documentation:
This property corresponds to the third parameter of a COLOR attribute. It refers to the foreground color of a control when the control is selected.

Column.Colors

Description: Column colors enabled?

Documentation:
If this property is non-zero then this column in a list box has coloring enabled.

Column.Header

Description: Column header

Documentation:
This specifies the header to be used for this column in the list box.

Column.HeaderIndent

Description: Column header indentation

Documentation:
Used in conjunction with *Column.HeaderJustification* to finely specify the justification of header within the list box column.

Column.HeaderJustification

Description: Column header justification

Documentation:
This property specifies the justification of the header within the list box column.

Column.HScroll

Description: Column horizontal scroll

Documentation:
If non-zero then this list box column is marked as having a horizontal scrollbar.

Column.Icon

Description: Column icon enabled?

Documentation:
If this property is non-zero then this list box column has icons enabled.

Column.ID

Description: Unique identifier for list box column

Documentation:
This property uniquely identifies the list box column. Currently the value used in here is the period qualified field name of the underlying variable.

Column.Indent

Description: Column indentation

Documentation:
Used in conjunction with *Column.Justification* to finely specify the justification of data within the list box column.

Column.Justification

Description: Column justification

Documentation:
This property specifies the justification of the data within the list box column.

Column.MaxWidth

Description: Column maximum width

Documentation:
The maximum tolerated width of the column in the list box.

Column.Picture

Description: Column display picture

Documentation:
The picture specified for the list box column.

Column.Resizable

Description: Column resizable

Documentation:
If non-zero then the list box column is marked as resizeable.

Column.RightBorder

Description: Column right border

Documentation:
If non-zero then the list box column is marked as having a right border.

Column.Tree

Description: Column tree enabled?

Documentation:
If this property is non-zero then the list box column is used as a tree column.

Column.Width

Description: Column width

Documentation:
The preferred width of the column in the list box. If left at zero then the desired width is computed as a function of the picture specified for the column.

ControlLabels

Description: Controls to Resize

Documentation:
A list of control labels to be managed by the resizer.

Controls

Description: Controls in group

Documentation:
A multi-valued acorn of radio Wizatrons. This is a list of the radio items that will appear within the option structure.

ControlResize

Description: Controls to Resize

Documentation:
All controls to be managed by the resizer.

ControlTemplate

Description: Control Template

Documentation:
The template name of the control template being managed by this Wizatron.

ControlToLookup

Description: Control to perform lookup upon

Documentation:
This property specifies the control to be looked up when the lookup button is pressed. By default the *LookupButtonControl* assumes it has been populated from a FieldClass and uses the *Use.Label* of the associated Entry control.

ControlType

Description: Control Type

Documentation:
This property gives the textual or Clarion name for a control.

CostTarget

Description: Cost target to attempt to reach

Documentation:
This specifies a threshold at which the screen evaluation Wizatron defines a screen as being good. The lower the number the better the screens and the slower the Wizatron. This property is used in conjunction with *MaxCostCycles*

Cursor

Description: Cursor when over control

Documentation:
A single drop that allows one of the standard built in cursor types to be chosen for this visual component. The range of cursors available can be extended by modify the *Cursor.List* property.

Cursor.List

Description: Available Cursors

Documentation:
A multi-valued symbol that defines all of the possible cursors that can be selected by the CURSOR property

Default

Description: Button is default

Documentation:
If this property is non-zero then the DEFAULT attribute is added to this button making in the target for the entry key on this window.

DeleteButtonControl

Description: Delete Button

Documentation:
A node pointing to the Wizatron that will populate the delete button for this control template.

Disable

Description: Set Disable attribute

Documentation:
If the value of this property is non-zero then the DISABLE property is set on the corresponding control.

DisplayColumns

Description: Fields to display

Documentation:
A multi-valued acorn giving a list of all of the columns to be populated in the list box.

DisplayColumns.List

Description: Available fields to display

Documentation:
This is the list of all fields that could potentially be populated in this listbox. By default any field in this list with an *Interest* level above the current *InterestThreshold* and without the NOPOPULATE(BROWSE) attribute set, is copied into *DisplayColumns*. By careful selection of interest levels within the dictionary, which the Dictionary Wizatron will help with, it is possible to control which fields are populated in each Browse box control.

DragID

Description: List boxes drag ID

Documentation:
This property corresponds to the DRAGID property of the listbox.

Drop.Count

Description: List boxes drop count value

Documentation:
This property corresponds to the first parameter of the DROP attribute.

Drop.Width

Description: List boxes drop width value

Documentation:
This property corresponds to the second parameter of the DROP attribute.

DropID

Description: Windows drop ID

Documentation:
A multi-valued list of the DROPIDs to be specified for this visual component

EditControl

Description: Edit Control

Documentation:
This node is used to hold the control to be populated to edit this field.

EIP

Description: Edit using edit-in-place

Documentation:
If this property is set to true then the corresponding value is set in the UpdateButton control template to enable edit-in-place upon the parent browse. This property is automatically set from the EDITINPLACE user option in the dictionary.

EnforceMinimum

Description: Enforce minimum window size

Documentation:
If set to one this property will cause the minimum size prompt to be checked on the resize template prompt screen.

Field

Description: Current field being managed

Documentation:
This property is the period qualified field name of the data to be edited.

FieldPriority

Description: Field allocation priority

Documentation:
Controls the Allocation.Priority for the controls populated from this control.

Fields

Description: List of Fields to be edited

Documentation:
This multi-valued acorn is a list of all of the fields populated on this form. The Wizatrons pointed to are **FieldClass** Wizatrons. These in turn populate the prompts, controls, and control templates as appropriate. By default, this is all the fields in the primary file that meet the *InterestThreshold*.

Fields.List

Description: Available fields

Documentation:
This is the list of all the fields that can be populated on this file. By default this is simply all the fields in the primary file.

Filter

Description: Filter Expression

Documentation:
The filter to be used for this sort order (not including anything generated by the range restrictions).

Flat

Description: Is the button displayed flat?

Documentation:
If this property is non-zero then the FLAT attribute is added to this button.

FocusKey

Description: Focus hot-key

Documentation:
This property corresponds to the KEY attribute of a control and is used to provide a particular hot key for an active control or window.

Font.CharSet

Description: Font typeface

Documentation:
This property corresponds to the fifth parameter of a FONT attribute. It is the character set used for the font of the control or window.

Font.Color

Description: Font color

Documentation:
This property corresponds to the third parameter of a FONT attribute. It refers to the text or foreground color of a control or window. To change the background color use *Color.Basic*.

Font.Size

Description: Font size

Documentation:
This property corresponds to the second parameter of a FONT attribute. It refers to the size (in points) of the text of a control or window.

Font.Style

Description: Font style

Documentation:
This property corresponds to the fourth parameter of a FONT attribute. It refers to the strike weight and style of the font of the control or window.

Font.Typeface

Description: Font typeface

Documentation:
This property corresponds to the first parameter of a FONT attribute. It is the textual name of the font of the control or window.

ForceRefresh

Description: Force Refresh indicator

Documentation:
If this property is non-zero then the actions are set so that the window refreshes when this control is accepted.

Format

Description: List box format string

Documentation:
The FORMAT string to be given to the list box. By default, this is computed from the information stored in the column Wizatrons.

FreeElement

Description: Variant field of sort order

Documentation:
This field is computed by the Wizatron code. It identifies the free element of a browse sort order. The free element of a browse is the most major component of the sort key that will vary throughout the data contained within the browse. Typically the free element is used for the locator field and / or the selection field within a select browse.

From

Description: List boxes FROM attribute

Documentation:
This attribute is used to set the name of the queue that the listbox will use as a FROM attribute.

FromDictionary

Description: Use properties of control from the active dictionary

Documentation:
When populating a field the Wizatron has the option of using the properties specified in the dictionary or not.
The options are :

Always	Use properties specified in the dictionary.
Never	Use Wizatron defaults.
IfDifferent	If the Wizatron default matches the dictionary value then leave the value as default. This does make a difference! It means that if the default later changes, the values are automatically updated by the Wizatron.

FromDictionary.List

Description: Use dictionary properties in list box controls

Documentation:
The three options for whether or not control properties come from the dictionary.
The options are :

Always	Use properties specified in the dictionary.
Never	Use Wizatron defaults.
IfDifferent	If the Wizatron default matches the dictionary value then leave the value as default. It means that if the default later changes, the values are automatically updated by the Wizatron.

Full

Description: Full attribute

Documentation:
If the value of this property is non-zero then the FULL attribute is set on the underlying control.

Generate

Description: Suppress generation

Documentation:
This property is used by the procedure Wizatrons to suppress the generation of the Wizatron frame.

GenerateBrowses

Description: Generate Browses by default

Documentation:
Setting this property to zero suppresses the generation of browses unless they are required for third normal form reasons by other procedures in the generation tree. Used as the root of QuickStart activity.

HasQuery

Description: Has QBE button

Documentation:
If this property is non-zero, a **BrowseQBE** Wizatron is created to manage the browse query button.

HasSelect

Description: Has Select Button

Documentation:
If this property is non-zero then a **SelectButtonControl** Wizatron is created to manage the browse select button.

HasToolbox

Description: Has toolbox button

Documentation:
If this property is non-zero then a **BrowseToolbox** Wizatron is created to manage the browse toolbox button.

HasUpdate

Description: Allow updates to browse

Documentation:
If this property is non-zero then a **BrowseUpdate** Wizatron is created to manage the browse update buttons.

HelpId

Description: Help Identifier

Documentation:
This corresponds to the HLP attribute of a control or window.

Hidden

Description: Control initially hidden

Documentation:
If the value of this property is non-zero then the HIDE attribute is placed on the underlying control. Screen space is still allocated to the control. To suppress space allocation use *Allocation.Ignore.*

Icon

Description: Menu icon

Documentation:
This property contains a file name to use as the icon for this menu item.

IconPosition

Description: Icon placement

Documentation:
This property changes the placement of the icon relative to any text on the button. A value of *Left* specifies the LEFT attribute, *Right* specifies the RIGHT attribute.

IconPosition.List

Description: Available icon placements

Documentation:
The list of possible icon positions used by the *IconPosition* property.

Image

Description: Image File

Documentation:
This property contains a file name of a graphic to use for this image.

ImageMode

Description: Image mode attribute

Documentation:
This property is used to set the CENTERED or TILED attributes on an image control.

Immediate

Description: IMM attribute

Documentation:
This property corresponds to the IMM attribute of a control or window. If set to a value other than zero the IMM attribute is placed on the underlying control or window

InsertButtonControl

Description: No Description

Documentation:
A node pointing to the Wizatron that populates the insert button for this control template.

Instance

Description: No Description

Documentation:
This property contains the template instance number as used by the Application Generator. It is automatically computed within Wizatron code, upon population of the template component, by incrementing the value of *Next.Control.Instance* in the parent procedure.

InterestThreshold

Description: Detail of field population

Documentation:
This property is a numeric threshold for the population of fields. An entity's interest level is compared to this threshold. By default, fields in a dictionary have an interest level of 5, the INTEREST() user option can then be used to alter that. Raising the interest threshold reduces the number of fields populated on the procedure.

Internal.FIELDS

Description: Used internally to control the number of fields

Documentation:
An internal property used to set the #FIELD attribute used by the Application Generator to tie a field list to a control.

Internal.LINK

Description: Used internally to control the #LINK attribute

Documentation:
An internal property used to set the **#LINK** attribute used by the Application Generator to define controls that are grouped together within a control template.

Internal.ORIG

Description: Used internally to control the #ORIG attribute

Documentation:
Internal property used by the Application Generator to specify the USE label that the control template uses to reference this particular control.

Internal.SEQ

Description: Used internally to control the #SEQ attribute

Documentation:
Internal property used by the Application Generator to link this control to a particular instance of a control template.

IsChild

Description: Is this browse a child browse?

Documentation:
This internal property is set to one if this browse is functioning as a child browse (i.e., range-limited to some **ParentFile**)

IsProcedure

Description: Name of procedure

Documentation:
This property is just a replica of the internal name of this Wizatron. It allows the parent procedure to be readily identified from within child Wizatrons for the purposes of locating other Wizatrons that are a known location within the procedure. It acts like the leading \ in a DOS file name.

Item.Procedure

Description: Procedure to invoke from the menu

Documentation:
This optional property specifies a procedure to call if this menu item is selected.

Item.Stack

Description: Stack size to allocate new procedure

Documentation:
If the procedure specified in *Item.Procedure* is started on a new thread, and controlled by *Item.Thread*, this property gives the size of stack allocated to it.

Item.Thread

Description: Invoke procedure on new thread

Documentation:
If set to non-zero, the procedure specified in *Item.Procedure* is not simply called but is started on a new Clarion thread.

Justification

Description: Controls justification

Documentation:
This property is used to set a justification attribute for the underlying control. This produces an attribute of LEFT, RIGHT, CENTER, or DECIMAL on the underlying control. An offset may also be specified, designated by *Justification.Offset.*

Justification.List

Description: Available justifications

Documentation:
List of possible justification attributes.

Justification.Offset

Description: Controls offset attribute

Documentation:
This property is used in conjunction with *Justification* and provides the offset for the attribute generated.

Key

Description: Sort Key

Documentation:
The key used for this sort order. This property is not required.

Label

Description: A WINDOW label

Documentation:
This property is used to specify the label that the window receives in generated code.

LoadMethod

Description: Load Method flag

Documentation:
This attribute determines whether the browse is page or file loaded.

Locator

Description: Locator indicator

Documentation:
An internal property used to point to the node acting as the locator Wizatron for this sort order.

Locator.Entry

Description: Entry Field

Documentation:
This property specifies the entry field to be used for the locator.

Locator.Prompt

Description: Prompt Field

Documentation:
This property identifies the prompt field to be used for the locator.

LocatorType

Description: Locator Type

Documentation:
Setting this property to a value other than **None** creates a Locator Wizatron. These, in turn, cause each sort order to have a locator populated of the corresponding type.

LookupStyle

Description: How to handle DISPLAY fields

Documentation:
This property stores a value to denote the style to be used when a third normal form field is encountered (typically by a DISPLAY() in the user options).

LookupStyle.Child.Button

Description: Lookup Button Control acorn

Documentation:
An acorn used to generate the Wizatron to manage the field lookup button control template if the **Lookup Button** option is chosen from *LookupStyle*.

LookupStyle.Child.Combo

Description: Child hook

Documentation:
An acorn used to generate the Wizatron to manage the file drop combo template if the **Combo** option is chosen from *LookupStyle*.

LookupStyle.Child.Entry

Description: Default Entry Field

Documentation:
This acorn is used to point to the EntryField class used to create the control used by all lookup styles other than *Combo*.

LookupStyle.Child.Prompt

Description: Default Prompt Field

Documentation:
This property identifies the Prompt field used by all lookup styles other than *Combo*.

LookupStyle.List

Description: Available lookup styles

Documentation:
Possible values for *LookupStyle* are:

Ignore	just use an entry
Combo	populate a file drop combo control template
Lookup on Error	populate the field options so that the field value is looked up in a select browse if it is entered wrong
Lookup Button	same as Lookup on Error except a field lookup button is also populated

Mask

Description: Controls mask attribute

Documentation:
If this property is non-zero then the MASK attribute is set on the underlying control.

MaxCostCycles

Description: Maximum number of times to interfere

Documentation:
The space allocator populates all controls using common-sense heuristics. Then it will go back over the screen looking for inconsistencies according to the criteria set in the *Allocation* properties. If it finds any it will move the controls to try to reduce these problems, then it will reapply its evaluation criteria. This property specifies the maximum number of times the criteria are reapplied.

Menu

Description: Menu to place item on

Documentation:
This property provides the TEXT property of the menu to which this menu item should be attached. Although case insensitive, the match must otherwise be complete.

MenuGrouping

Description: Menu item grouping

Documentation:
This property specifies the method used to order menu items within a menu.
Name sort alphabetically
Priority sort by priority and then name

MenuGrouping.List

Description: Available menu item groupings

Documentation:
The possibilities for order menu items within a menu.

MenuManager

Description: The MenuManager Wizatron

Documentation:
This is an acorn for the menu manager for the frame.

Menus

Description: Menus

Documentation:
This is a multi-valued acorn that lists all of the menus that are required by the populated menu items. By default, a value of *xyz* corresponds to a Wizatron of type Menu#*xyz*. The list of possible values is stored in *Menus.List.*

Menus.List

Description: Available menu items

Documentation:
The list of all menu headers that can be named by a populated menu item. *Default* is used for any menu item that does not specify a parent menu.

Message

Description: Windows message attribute

Documentation:
This corresponds to the MSG attribute of a window or control and specifies the text to display in the first location of the status bar.

Next.Control.Instance

Description: Internal instance allocated to control template

Documentation:
The instance number (as used by the Application Generator) of the next control template to be populated in this procedure. It is used in the production of *Instance* properties in control templates.

Node.ChildProcedures

Description: An internal procedure list

Documentation:
This multi-valued property lists all of the procedures that have been created at the request of this one.

Node.Controls

Description: Window Control list

Documentation:
An internal list maintained by the Wizatron code of all of the controls appearing within a window.

Node.ControlWizards

Description: Internal procedure control Wizatrons

Documentation:
Internal multi-valued symbol containing a list of all of the Template Component Wizatrons populated in this procedure.

Node.Frame

Description: Frame property class

Documentation:
A property used to point to the frame used in this application.

Node.Items

Description: Items on menu

Documentation:
A multi-valued property containing a list of all of the menu items populated. This property is maintained internally by the Wizatron code and should not be edited.

Node.SheetAllocator

Description: Sheet space allocator

Documentation:
An internal property used to hold the space allocator for the sheet if there is one.

Node.SpaceAllocator

Description: Window space allocator

Documentation:
An internal property used to hold the space allocator for the window.

Node.ToolAllocator

Description: Toolbar space allocator

Documentation:
An internal node used to hold the space allocator for the toolbar if there is one.

NoMerge

Description: Menu merging flag

Documentation:
If this property is non-zero then the NOMERGE attribute is set on the underlying control.

Output.Category

Description: Procedure type

Documentation:
This property provides the category to be used in generated code for this procedure. Although the default here is blank, most of the derived procedure types provide meaningful defaults for this property.

Output.Description

Description: Procedure description

Documentation:
This property provides the description to be used in generated code for this procedure.

Output.From

Description: Template Name

Documentation:
This property provides the Template Name to be used in generated code for this procedure. The default is **ABC Window**.

Output.ProcName

Description: Procedure name

Documentation:
This property provides the Procedure Label to be used in generated code for this procedure. Although the default here is blank, most of the derived procedure types provide meaningful defaults for this property.

ParentFile

Description: Range limit file name for a child browse

Documentation:
For a child browse (where *IsChild* is set) this property is used to denote the file used to range-limit the browse.

Password

Description: Controls password attribute

Documentation:
If this property is non-zero then the PASSWORD attribute is set on the underlying control.

Picture

Description: Controls picture attribute

Documentation:
This property is the picture for the control, this is specified in the Clarion language as the first parameter of the control statement.

Populator

Description: Control template parent

Documentation:
This is a property specifying the Template Component that populated this control. This is the field that is followed when a macro is expressed as *%%^MacroName%%*.

PostLookupField

Description: Field for validation lookup

Documentation:
This is a component of the *PostLookupKey* used to verify the value in the entry control.

PostLookupKey

Description: Key for validation lookup

Documentation:
This property specifies a key in a related file that can be used to lookup whether or not this control contains a valid value. This property is automatically filled in by the Wizatron if third normal form attributes are used in the dictionary.

PostLookupProcedure

Description: Procedure for validation lookup

Documentation:
This is the name of a procedure to be used to select a correct value for this field if a valid one is not entered by the user. This is an acorn property which will cause a browse with selection enabled to be generated if one does not already exist. If the *Acorn.Name* subproperty names something other than *Browse %_VALUE_%* then a unique select browse is generated.

PrimaryFile

Description: Primary file this Wizatron is based on

Documentation:
The primary file (as it would appear in the file schematic) that this control template is based upon.

PrimaryKey

Description:
Primary key this Wizatron is based on

Documentation:
The (optional) primary key used by this control template.

ProcsPerModule

Description: Procedures per module

Documentation:
Determines procedures to populate in each module. We recommend between 5 to 20.

RangeField

Description: Range Field

Documentation:
The field of the present sort order to be range limited.

RangeFile

Description: Range Limiting File

Documentation:
For a *RangeLimitType* of **File Relationship** this provides the file used to range limit the current one.

RangeHigh

Description: Range High

Documentation:
For a *RangeLimitType* of **Range of Values** this provides the field acting as the high bound of the range.

RangeLimit

Description: Range Limit

Documentation:
For a *RangeLimitType* of **Single Value** this provides the field to act as the limiting value.

RangeLimitType

Description: Range Limit Type

Documentation:
The type of range limit to be used.

RangeLow

Description: Range Low

Documentation:
For a *RangeLimitType* of **Range of Values** this provides the field acting as the low bound of the range.

ReadOnly

Description: Controls read-only attribute

Documentation:
If this property is non-zero, the READONLY attribute is set on the underlying control.

Required

Description: Buttons REQ attribute

Documentation:
If this property is non-zero, the REQ attribute is added to this button meaning that when this button is pressed all required fields are checked to contain valid values.

Resizer

Description: Resizer Wizard

Documentation:
If this property is non-zero, it becomes an acorn for the Wizatron that generates the Resizer procedural extension. Setting this value to zero disables resizing in the window. By default this property is set to true if the parent window has a frame type of resize.

Right

Description: Menu RIGHT attribute

Documentation:
If the value of this property is non zero, the RIGHT attribute is set on the menu causing it to go at the right hand end of the action bar.

Scroll

Description: Controls scroll attribute

Documentation:
If the value of this property is non-zero, the SCROLL attribute is placed on the underlying control.

Scrollbars

Description: List boxes scroll bars

Documentation:
This property is used to set the HSCROLL, VSCROLL, or HVSCROLL attributes on a listbox.

ScrollBars.List

Description: Available scroll bar attributes

Documentation:
The list of attributes pertaining to the scrollbar behavior of a ListBox.

ScrollBehavior

Description: Scroll Behavior

Documentation:
The vertical thumb strategy to use.

ScrollKeyDistribution

Description: Scroll Key Distribution

Documentation:
For a *ScrollBehavior* of **Movable Thumb** this provides the distribution to use.

SecondaryFiles

Description: Secondary files list

Documentation:
A multi-valued list of secondary files (as they would appear in the file schematic) used by this control template. This property is automatically set by the Wizatron code based upon DISPLAY attributes within the *User Options* fields of the dictionary.

Separator

Description: Menu item is separator

Documentation:
If this property is set to non-zero, other properties in this Wizatron are ignored and a SEPARATOR is placed on the item.

Sheet

Description: Sheet Indicator

Documentation:
An acorn used to specify the sheet for the current window if there is one.

Show

Description: Show on procedure screen

Documentation:
If this property is set to non-zero, this control template is populated on the main procedure window.

Skip

Description: Controls skip attribute

Documentation:
If the value of this property is non-zero, the SKIP attribute is placed on the underlying control.

SortOrders

Description: Browse Sort Orders

Documentation:
A multi-valued acorn with an entry for each sort order to be populated in the control template. The entry is the key name, but ****RecordOrder**** is a special value and means no sort is applied.

SortOrders.List

Description: Available browse sort orders

Documentation:
This list of all possible (keyed) sort orders for this browse. For a child browse this will be the order that range-limits the browse. For a regular browse this is every key sequence in the file.

SplashWindow

Description: Show Splash Window

Documentation:
This acorn causes a splash window to be generated if set to non-zero.

Standard

Description: Standard action attribute

Documentation:
This property specifies the value of the STD attribute used to set standard behavior for this control.

Standard.List

Description: Standard actions

Documentation:
The list of all standard actions currently available in the runtime library.

StandardMenuItems

Description: Standard Menu Items

Documentation:
This is a multi-valued acorn property specifying the list of default menu items to be populated. By default, it is configured so that selected *xyz* will cause Wizatron MenuItem#xyz to be populated into the procedure. The list of possible menu items is specified by *standardmenuitems.list*.

StandardMenuItems.List

Description: Available standard menu items

Documentation:
The list of all of the menu item value (#) derivations currently available in the system.

TabName

Description: Primary Tab name

Documentation:
The name of the tab upon which you wish the whole browse to appear. This defaults to ALL which means each sort order appears on its own tab. The Wizatron code automatically changes this in the case of a child browse to force the browse (including update buttons) to appear on a child tab.

Tabs

Description: Tabs to be on the procedure

Documentation:
A multi-valued acorn of all tabs appearing on the current window. The default *Acorn.Wizard* property allows for a heterogeneous list whereby an assigned value of **MyTab#Fred** produces a Wizatron of type **TopSpeed:Tab#Fred** with text attribute of *MyTab*. This list is added to incrementally by the Wizatron code.

TabText

Description: Text for Tab

Documentation:
The text to appear on the tab for this sort order. Note that the text can be suffixed with #TabType to encourage the Wizatrons to select a particular variant of the Tab Wizatron default values. Specifically, the Wizatrons generate MyTabText#ChildBrowse if the parent browse is populated as a child upon a form.

TargetResolution

Description: Target Desktop Resolution

Documentation:
The Wizatrons are capable of designing screens to fit a multitude of resolutions. The option chosen here cause the Wizatrons to try to create a perfect fit window size. This property is used to calculate the maximum size available.

TemplateChain

Description: Active Template Chain

Documentation:
The template chain of the control template managed by this Wizatron.

Text

Description: Control Text property (PROP:TEXT)

Documentation:
The text to be displayed on the menu. This may include & characters to set a hot key for the menu item.

Toolbar

Description: Toolbar control

Documentation:
An acorn used to specify the toolbar used for the current window if there is one.

Tooltip

Description: Controls tooltip

Documentation:
This property is used to provide the TOOLTIP attribute for the control.

Transparent

Description: Controls transparent attribute

Documentation:
If this property is non-zero, the TRN attribute is set on the underlying control.

TypingMode

Description: Controls typing mode

Documentation:
This property specifies the typing mode of the control. It can produce an attribute of INS or OVR.

TypingMode.List

Description: Available typing modes

Documentation:
List of possible typing modes. INS, OVR, or none.

UpdateProcedure

Description: Create update procedure

Documentation:
If this acorn is set to non-zero, it causes a Form procedure to be generated for the current primary file. By default the current primary file is simply inherited from the Wizatron that owns the *BrowseUpdate* Wizatron. Note that this Wizatron asks to create a shared procedure. In other words, if a procedure already exists called **UpdateMyPrimaryFile** then that procedure is used.

Use.AltEquate

Description: Alternative field equate label

Documentation:
This corresponds to the third parameter of a USE attribute. It is used to specify a unique field equate label where the default compiler mangling of the field name would not.

Use.Label

Description: Field equate label of use variable.

Documentation:
This corresponds to the first parameter of a USE attribute. It is used to specify a field for the underlying control or just the field equate.

Use.Number

Description: Field equate value

Documentation:
This corresponds to the second parameter of a USE attribute. It is used to provide a value for the field equate label generated when the default compiler numbering scheme is not adequate.

WindowManager

Description: Window Wizatron

Documentation:
Internal property used to point to the Window Wizatron active in this procedure.

General Style File Design Tips

The open architecture of Wizatrons encourages development of additional style files and Wizatron DLLs which will create a variety of application features limited only by the imagination of Wizatron authors. In addition, you may also need to create your own custom style file.

Here are some general guidelines to consider when creating new style files:

- It is not a good idea to start from scratch. Instead, copy a default style file that best matches your desired features to another file name, and begin modifying from that.
- Modifications you make depend on what you specifically want to do. If it is a visual style, pick something that you do not like the look of and experiment with different property values.
- If you are making structural changes (new procedures, menu items, etc.) complete those first before working on the visuals.
- Try to make as much of your design dependent on a user option to eliminate the need of frequent style property changes. In other words, include flexibility into your style files that allow settings at design time.
- Learn the power and benefits of acorns.
- Finally (and most important), always try to think of ways of modifying the parent Wizatron so that the children will *inherit* the new values that you need (or use the macro language to allow them to compute the results).

In addition, the *Advanced Tutorial* chapter provides guidance in style design and extending functionality in Wizatron generated applications. Our next section introduces a tool that will help you to accelerate your style file development.

The Template Wizatron

Introduction

The Template Wizatron is a tool to help you add functionality to existing style files. Its purpose is to convert existing embedded source code extracted from an application and process them into template syntax.

The Template Wizatron creates a template from a completed procedure. It creates either an extension template or a control template which you can then use in any application.

Additional functionality in the Wizatrons is most commonly added by deriving new control template Wizatrons. These control template Wizatrons are then added to the concrete schema (application view) through the use of special acorn properties. The Template Wizatron is designed to help you "write" these control templates.

Overview

The following section describes the steps recommended to convert existing application embedded source into a control template format which is compatible with the Wizatron environment.

1. Find a procedure which has a desired feature that you wish to incorporate into the Wizatrons. From the Application Generator menu , **choose** the selective export option to create a TXA representation of that procedure and its associated embeds.
2. Run the Template Wizatron (TW.EXE). Choose **File ➤ Import TXA File.** Locate the TXA file you wish to process and **press** the **OK** button.
3. **Browse** and **edit** the *EMBEDS*, *DATA*, and *CONTROLS* files to eliminate any redundant or unwanted components.

   The Template Wizatron extracts all embedded source and window controls in the procedure. This step allows you to choose only the portions needed for this template.
4. **Add** any additional *INFO* and/or *PROMPTS* that you anticipate will be necessary for the template you will be creating.

   This allows you to add flexibility by providing template choices.
5. **Run** the *Embed Advisor* to detect numeric instances contained in your embedded code which may need to be resolved with a replacement template symbol.

6. When you are ready to generate the template script, **choose File ➤ Export (Write Template)**. You will be asked to provide an output file name to generate to. This would normally be a file with a .TPW or .TPL extension.

The Template Wizatron process is now complete.

To better understand the information that is formatted for you by the Template Wizatron, let's examine the file formats in a little more detail.

Default File Descriptions

The following section describes the file, field, and key information contained in the default Template Wizatron data dictionary:

Data

The *DATA* file describes the local data elements (variables) extracted from the imported procedure(s).

The *NAME* field contains the valid label of the local data (up to 50 characters).

The *TYPE* field describes the data type (BYTE, SHORT, LONG, STRING, etc.) and any optional comments describing the data (up to 40 characters).

Controls

The *CONTROLS* file contains a single field (TEXT) which stores all window controls of the imported procedure(s). There is a single key defined to prevent duplicate names.

Embeds

The *EMBEDS* file contains extracted source code and its associated embed points.

The *CONTENTS* field is a MEMO field with a 40000 character maximum length per embed point.

The *PLACE* field stores up to 200 characters which describe the name of the embed point (its template symbol).

The last field, *PRIORITY,* is a numeric storage element which holds the embed priority.

Info

The *INFO* file allows you to add template parameters to the generated output based on your embed requirements.

The *ExtensionName* field holds the name of the template that you are creating. You can enter up to 199 characters, but no space characters are allowed in the name.

The REQ attribute of control templates specifies that the template you are generating requires a previously placed Code, Control, or Extension template before it may be used. The *REQUIRES* field in the *INFO* file holds the name of the template that would be required.

The *DESCRIPTION* field is a string constant (up to 199 characters) which describes the control template you are creating.

Prompts

The *PROMPTS* file allows you to add the necessary template prompts which can prime your embed points with necessary information.

The *FIELDNAME* field is the name of the template symbol that will be primed by the generated template #PROMPT statement. Enter the name of the template symbol here **without** the leading ‘%’ syntax. (Example - If your Prompt is priming a template symbol ‘%author’ simply enter ‘author’ in the *FIELDNAME* field.

REPLACES is the data field name that the *FIELDNAME* template symbol value will replace. (Example - Your embedded code references a field named AUT:Author. You wish to prime this field with the value stored in the template symbol ‘%author’. Enter ‘AUT:Author’ in the *REPLACES* field and ‘author’ in the *FIELDNAME* field.

The *TYPE* field in the *INFO* file identifies the prompt’s picture token or prompt type. (*See the Programmer’s Guide for a complete listing of all valid keywords*).

The *PROMPT* field is the text that displays for each prompt. For example, if you needed to enter a file name into a control template you were creating, you could enter *Enter a valid file name:* into the *PROMPT* field.

Others

There are two other files used in the Template Wizatron which are utility files. The *INPUT* file is an ASCII file which allows the direct import and processing of a selected .TXA file. The OUTPUT file is also an ASCII file, allowing a smooth export to a standard template file format (.TPW).

The Embed Advisor

A special area of the Template Wizatron is called the *Embed Advisor*. Choose **Advisor ➤ Embed** from the main menu to call this procedure.

At this point, your embeds are analyzed by the *Advisor.* If it detects an instance number which was generated by the TXA, it will “advise” you to replace the instance number with a dynamic template symbol, *%ActiveTemplateParentInstance*. Press the **OK** button to accept the replacement.

Summary

Wizatrons become a more powerful tool when you can save your favorite embed techniques into control or extension templates. In turn, the templates can be used by the Wizatrons automatically and often. The Template Wizatron helps you to quickly achieve this goal. In addition, the application and dictionary is provided to you so you can customize the tool to directly fit your particular needs.

The next section contains a brief tutorial to help get you started using the Template Wizatron.

Using the Template Wizatron

This section provides a quick look at using the Template Wizatron.

Set Up - Source to Application Text (TXA)

The first step to using the Template Wizatron is to identify a process or feature of an existing application's procedure that is using embedded source.

This example uses an area of embedded source that is used to limit a Browse procedure to a single thread instance.

Starting Point:
Examine the *TLSOURCE.APP* application found in the *\EXAMPLES\WIZTOOLS* folder located in the Clarion5 folder. The *BrowseAuthor* procedure contains the embedded source code that we have selectively exported to an Application Text File (TL.TXA) for use in this tutorial.

Import the TXA file

1. **Start** the Template Wizatron (TW.EXE) from the Tools menu located in the Clarion Program Group. If you do not have this menu option, you can start the TW.EXE program located in the \BIN folder of your Clarion install.
2. **Choose** File ➤ Import TXA File from the menu.
   The following window displays.

3. **Press** the ellipsis button to open the **Select a TXA file...** dialog window.
4. **Navigate** to the ..*\EXAMPLES\WIZTOOLS\TW* folder, **highlight** the *TL.TXA* file, and **press** the Open button.
5. **Check** the **Clear Files on Import** option, and **press** the Open button to complete the import process.

   The **Clear Files on Import** option removes any Template data from a previous session. Let's examine and edit the information imported.

Edit the Template Data

The next few pages involve understanding and editing the imported Template data files.

Local Data

1. **Choose** **Browse ➤ Browse Data** from the menu.

   Three data records are present. The first data type in the list box, *FilesOpened*, is a template generated data type and must be removed. The other data elements (easily identified with the *LOC:* prefix naming convention) are part of the thread limiting data needed.

2. **Highlight** the *FilesOpened* data record, and **press** the **Delete** button. The confirmation window is shown below.

3. **Press** the **Yes** button to delete the record, then **press** the **Close** button to close the *Browse Local Data Definitions* window.

Controls

If your imported Template data requires a control to be associated with it, you can examine its definition here. You can also delete other controls that are not needed.

1. **Choose Browse ➤ Browse Controls** from the menu.

   In our thread-limiting example, there are no controls associated with this process. We will need to delete all controls at this time.

2. **Press** the **Clear All Controls** button.
   The following confirmation window displays.

3. **Press** the **Yes** button to delete all Template controls, then **press** the **Close** button to close the *Browse Controls File* window.

Embeds, Info, and Prompts

This example is nearly complete. However, there may be other times when the TXA file you have imported has other embed points that are not directly related to the process you are trying to abstract. You can choose to edit your output TPW file after exporting, or you can directly view and edit your embed points via the *Embed* Template data.

1. **Choose Browse ➤ Browse Embeds** from the menu.

   You should see three embed records. The text box in the window displays the actual source code associated with each record.

2. **Examine** the remaining Template records. When you have finished, **press** the **Close** button to close the *Browse Embeds File* window.

   Info and Prompts are not required in this exercise. We need to make one more stop in the Template Wizatron before exporting.

Check the Embed Advisor

1. **Choose Advisor ➤ Embed** from the menu.

   If there are any numeric template instances detected by the Embed Advisor, you would have the opportunity to resolve those embed points at this time. However, out thread-limiting Template has yielded a different result. The message window is displayed.

2. **Press** the **Ok** button to close the Embed Advisor. We are now ready to export our results.

Export to a Template Source File (TPW)

1. **Choose File ➤ Export (Write Template)** from the menu.

   The following window displays.

2. **Press** the ellipsis button to open the **Create or Select a .TPW file** dialog window.

3. **Navigate** to the ...*TEMPLATE* folder, **type** *TL.TPW*, and **press** the `Open` button.

   Your path and filename should now be visible.

4. **Press** the `Ok` button to export your Template data.

   The results window displays.

Several features are evident here:

- The output is in a text file and you have the opportunity to print the results from this window.
- If there are no controls necessary in our template, a #EXTENSION template statement is written. A #CONTROL statement is written if controls are included.

After the Template Wizatron

How can we now incorporate our results into the template registry, and later the Wizatrons? Consider the following file:

```
#!  Contents of USER.TPL
#!  Wizatron Tools
#!
#!
#TEMPLATE(UserTools,'User Defined Templates'),FAMILY('ABC')
#!Thread Limiting
#INCLUDE('tl.tpw')
#!Play Sound
#INCLUDE('playsnd.TPW')
```

The text above represents how easy it is to create a family of templates and include templates generated by the Template Wizatron. After creating this file, you can register it in the Template Registry, and later add their functionality to the Wizatrons.

In the next chapter, we will explore more of this process.

6 - ADVANCED TUTORIALS

Alternate Views of the Application Wizatron

This tutorial is designed to familiarize you with the additional interface features and tools available when using the Application Wizatron and the Wizatron Configuration Utility. You will examine several elements that were documented in earlier chapters.

Creating an Application

Starting Point:
Make a copy of *STANDARD.STF* and *STANDARD.STY* which are located in the *|CLARION5|TEMPLATE* folder, and rename your copies to *TUTORIAL.STF* and *TUTORIAL.STY.* In this tutorial, you will change the contents of the style file and folio.

Create a new application and start the Application Wizatron

1. Start the Clarion 5 development environment. The Pick dialog should be open. **Press** the **New...** button and create a new empty *WIZTUTOR* application in the *\CLARION5\WIZTUTOR* folder using the BOOKS.DCT (do not use Quick Start or the Application Wizard).

2. **Choose Application ➤ Template Utility...**
3. **Highlight** the *ApplicationWizatron* branch under *Class Wizatrons*, then **press** the **Select** button.

At this time, you should see the **Prompts for ApplicationWizatron** window.

4. Make sure that all *QuickStart* options are turned **off** (Check boxes are unchecked). **Press** the **Ok** button to continue.
5. **Select** the *TUTORIAL.STF* file in the *\Clarion5\Template* folder when asked to select a style folio, and **press** the **Open** button.

   The Application Wizatron opens and the Wizard View displays.

Types of views

The heart of Wizatron understanding lies in the abstract and concrete schemas. The concrete schema is the representation of the application you're creating. The abstract schema is the representation of the *type* of application you're creating (the style sheet).

The Wizatrons have 2 view types: the **Wizard View** (using a standard Wizard-type interface) which only allows access to the concrete schema; and the **Tree Views** (a property inspector-type interface) which allow access to both the abstract (**Style**) and concrete (**Application**) schemas.

This is the Wizard View.

1. **Move** the **Prompt Detail Level** slider to the right end (It may be there already from a previous session).

The Prompt Detail Level slider allows you to vary the amount of detail presented in the standard Wizard-type interface.

The Tree Views appear when you choose the **Window ➤ Style** or **Window ➤ Application** menu items.

We will only use the Tree Views for this tutorial.

Style View

1. **Choose** Window ➤ Style

The tree on the left represents the “building blocks” or components used to construct applications. Each entry represents a single component. The list box is formatted to display as a collapsible hierarchical list. The plus (+) sign indicates a collapsed level that expands when the user CLICKS on the plus (+) sign. Conversely, the minus (-) sign indicates an expanded level that collapses when the user CLICKS on the minus (-) sign.

The list on the right are all the properties associated with the highlighted component. Anything we change and save here is automatically applied to all future applications that use this style folio (*TUTORIAL.STF*).

All of the TopSpeed components are prefixed TOPSPEED and grouped together. Third-party components will appear in their own sections.

The branch structure of the tree mimics the way the components’ properties are inherited. Let’s examine these properties in more detail.

2. **Highlight** *TOPSPEED:VISUALCOMPONENT*.

   Observe how all of the properties of this Visual Component are also shown in TOPSPEED:Control.

3. **Highlight** *TOPSPEED:CONTROL*.

TOPSPEED:Control also adds some properties of its own, besides those that it inherits from TOPSPEED:VisualComponent. Observe that TOPSPEED:Menu also inherits properties from TOPSPEED:Control.

4. **Highlight** *TOPSPEED:MENU*.

   On this level, most of the parent's (*TOPSPEED:VISUALCOMPONENT*) properties are hidden.

   You may notice that some entries have a # in their name. This is because they are all essentially using the same Wizatron, but each derives its own unique set of default property values.

5. **Highlight** the first three MENUITEM# entries in order, one at a time, and notice the differences in the property values. You may need to expand the *TOPSPEED:MENU* branch.

   There's more to this, but we'll come back to this later.

Property Inheritance

Lets look a one particular component in more detail.

1. **Highlight** *TOPSPEED:LISTBOX*. If you have trouble finding this component, RIGHT-CLICK on the *Style Components* list box, and select the **Find** option. (Hint: Expand *TOPSPEED:ENTRY*)

This Wizatron is responsible for populating list boxes onto windows. Many of the properties you can see map directly to standard Clarion list box attributes. The symbols tell you where the current property values were obtained. The types of symbols include:

Standard
This property does not appear in the parent Wizatron. For example, TOPSPEED:Window has a specific CENTER property, but its parent Wizatron TOPSPEED:VisualComponent does not (all visual components do not share a CENTER attribute). The CENTER property only applies to the window.

Inherited
The parent has a specific value for this property and the current Wizatron is using that value. (e.g., *At.Height*).

Overridden
The parent has a value for this property and so does the current Wizatron. It does not necessarily mean the local value is different, simply that it has one. This distinction is important if you later change the parent value. When you do, the current Wizatron component would also reflect the change, unless it is overridden.

2. **Highlight** the *CASE* property.

CASE has a circle symbol indicating its value is inherited. Notice the status bar at the lower left corner of the window. The first part ("Case of entries") displays the prompt text that you would see in the Wizard View (the Wizard-type interface), and the second part ("TOPSPEED:Entry") tells you from where the property's value is inherited.

Property Inheritance in Action

Let's see how the inheritance works.

1. **Highlight** the *COLOR.BASIC* property.

The three COLOR: properties that you can see here map directly to the three parameters of Clarion's COLOR attribute. Let's choose a color:

2. **Press** the ellipsis button (**...**) at the right end of the entry control.

3. **Highlight** COLOR:Lime (3rd across, 3rd down from the top left corner of the color dialog). **Press** the **Ok** button to continue.

Note the use of Edit in Place (EIP). All of the EIP classes are part of the ABC library. There is EIP support for font dialog, file dialog, multi-select list, and drop lists.

Note how the COLOR.BASIC change has cascaded through the components under TOPSPEED:ListBox (TOPSPEED:ComboBox, TOPSPEED:Listbox#Browse etc.).

4. **Highlight** *COLOR:BASIC* for the *TOPSPEED:COMBOBOX*, then *TOPSPEED:LISTBOX#BROWSE* and notice the values it has and from where they were inherited.

Property Macros

Properties can have values as we have seen. However, they can also contain macros and functional macros.

1. **Highlight** *TOPSPEED:STRING#SPLASH#DEVELOPER.*

   The Splash#String#Developer Wizatron is responsible for populating the author string on the splash window.

2. **Highlight** the *TEXT* property.

Notice the Text property contains a macro *%Author%*. This means that the area represented by the percent signs (%) will be replaced with the value from the *Author* property (located in the TOPSPEED:Application component).

3. **Type** *on %DATELONG%* at the end of the property's current value as shown below.

We've now added a built-in macro to the property.

4. **Press** the DOWN ARROW cursor key.

Notice that the %DATELONG% macro can be resolved now, but the %Author% macro cannot. %Author% can only be resolved in the Application View because it is a value that will be dependent on the specific application you're creating (the concrete schema) and is not available to the style (the abstract schema).

Lets look at some more property macros:

5. **Highlight** *TOPSPEED:Button#BrowseQBE.*

6. **Highlight** the *Use.Label* property.

The caret character (^) in the macro (located just before the 'I' in INSTANCE) means that the property is looked up from the populating Wizatron rather than from the parent(s) in the tree. The populator is simply the Wizatron that populated this Wizatron (You may have noticed a little bit of pre-processing as you started this tutorial. That was the populator in action.) In this example, the *Use.Label* attribute of the button control will be *?Query:* followed by the value of the Instance property of the populating Wizatron when it is resolved.

Property Functions

Properties can also contain function calls. This adds flexibility to style sheet design.

1. **Highlight** *TOPSPEED:ENTRY.*

2. **Highlight** the *PICTURE* property.

Here the *@SCREENPICTURE* function is called to retrieve the screen display picture for a given field label. The function takes a field label and returns the picture attribute stored in the dictionary. Note that the value of the field property comes from the populating Wizatron (the ^ character does this). This is very powerful, but let's "raise the bar another notch".

3. **Highlight** *TOPSPEED:MENU*.
4. **Highlight** the *TEXT* property. You should see the following text:

&@EVAL(SUB('%USE.LABEL%',2,LEN('%USE.LABEL%')-1))

The *@EVAL* function is very useful, since it gives you access to the Clarion EVALUATE statement. Here we call the Clarion SUB function to obtain the Text property from the *USE.LABEL* property. A *USE.LABEL* value of '?AUTHOR' returns a value of 'AUTHOR' in the *TEXT* property.

This completes our examination of the Style View.

Application View

Let's look at another Wizatron interface view that is provided for you.

1. **Choose** **Window ➤ Application**

   The Application View provides an interface which is almost identical to the Style View.

   Let's look at a typical window procedure

2. **Highlight** the About Window's *Window* branch. Make sure that the **Prompt detail level** slider is set to **Maximum Value.**

The window tree shows what is populated onto the window (visual components). The CloseButtonControl item is derived from the Close Button control template (template components) defined in its property values.

3. **Highlight** the Splash Window's *Window* branch.

The Splash window normally consists of a close button, an image, a title field, an author field, and a date field.

Inheritance from the Default Components

Previously, we changed the TEXT property of the Developer field in the Style View. Let's see if our change has affected other components.

1. **Highlight** the *Developer* branch located in the *SplashWindow* node.
2. **Highlight** the *TEXT* property.

   See how the property contains our revised macro. Notice how the %Author% macro can now be resolved, since a value for %Author% can be found in the APPLICATION WIZATRON component.

3. In the *Application Components* list box, **highlight** *APPLICATION WIZATRON*.
4. **Highlight** the *AUTHOR* property

Let's change it and see what happens.

5. **Type** in *TopSpeed Tutorial*.

   We should now see the change reflected in the Splash Window's author string.

6. **Highlight** the SplashWindow > Window > *Developer* branch in the Application View and look at the value in the *TEXT* property. Confirm that the text has been modified.

Window Component Positioning

While we're here lets take a look at the splash window definition.

1. **Highlight** the Splash Window's *Window* branch.

   Again, most of the properties here correlate directly to Clarion window attributes. There are a couple of special ones, AT.MAXHEIGHT and AT.MAXWIDTH for example. These properties define the maximum size allowed for this window.

   We also have a component called a Space Allocator that is responsible for laying out windows. These properties limit how large it allows the window to become.

2. **Highlight** the *AT.MAXHEIGHT* property.

   The *@BestWinHeight* function returns a maximum height for the window based upon the current target resolution. Likewise, there is an *@BestWinWidth* function. Obviously, if the TARGETRESOLUTION property in the APPLICATION WIZATRON changes, the macro is re-evaluated.

In the case of the standard procedure wizards, there isn't an Application Wizatron to provide the target resolution. In this case the BestWidth and BestHeight properties are normally set by the calling templates.

Understanding Space Allocation

Space Allocation refers to the method that the Wizatrons use to calculate and position the controls that we have defined. In this section, we will explore some of the basics in controlling this method.

In general, it is not possible to put a control at a specific XY position on a window. This is because an individual Wizatron does not know about the existence of other Wizatrons, or their "real estate" requirements. Therefore, we have a "double-blind" situation. The Wizatrons use the space allocator to calculate these positions.

The space allocator divides the window into nine segments, labeled like a compass; north, northeast, east, etc. The center zone is called center.

A Wizatron populates its controls into a zone, and the allocator is responsible for making sure that the zone looks acceptable. A "handle" mechanism is used for maintaining control alignment. The space allocator also sizes the

window to fit nicely around the populated controls once everything is populated.

1. **Highlight** the *AboutWindow* > *Window* > *Close#AboutButton* branch.

   Note that the close button is allocated to Zone:SouthEast. Let's move it to the bottom left (the SouthWest zone).

2. **Highlight** the *ALLOCATION.ZONE* property.

3. **Select** *Zone:SouthWest* from the droplist.

   Notice that a diamond symbol now appears, to indicate that this property has an overridden value.

4. **Highlight** the *AboutWindow* > *Window* > *Copyright* branch.

5. **Highlight** the *ALLOCATION.ZONE* property.

6. **Select** *Zone:SouthEast* from the droplist.

   Also, notice a property above called "ALLOCATION.TAB". This is used to group controls onto an individual tab. You can use an existing tab label, or a new one, in which case a new tab is automatically created. The default value of this property is the default tab label.

Building an Application in the Application View

Let's add some more procedures to our WIZTUTOR application. We will now add browse and update procedures using the files in our dictionary (BOOKS.DCT).

We will continue to work in the Application View:

1. **Highlight** the *Frame* branch.

2. **Highlight** the *BROWSEFILES* property.

3. **Press** the ellipsis button (**...**) at the right end of the entry control.

   The available items list box list all the files in this dictionary. We can select some, all, or none from this list. Let's select them all.

4. **Press** the >> button.

   This adds all available items to the Selected Items list.

5. **Press** the **OK** button.

   The Wizatron processes your selected files. Look at the comma-delimited display of the *BROWSEFILES* property. You should now see the files you have selected.

6. **Scroll** the *Application Components* list box and look at the new browse procedures attached to the *Frame* branch. The window shown below is a contracted view of your application to this point.

More inheritance from the Style Sheet

Previously, we changed the default color list boxes. Let's examine the new procedures to verify that this color change has been maintained:

1. **Expand** the *BrowseBook* branch.
2. **Highlight** the *Window* branch located under *BrowseBook*.
3. **Highlight** the *ListBox* branch located under *Window*.

   See that the *COLOR.BASIC* property is indeed "COLOR:Lime":

4. **Contract** the *Window* branch.

Exploring Acorn Properties

The browse box item effectively represents the browse box control template.

1. **Expand** the *BrowseBox* branch.

   Notice how the BrowseBox Wizatron has automatically populated the update, select, QBE, and toolbox Wizatrons. Again, think of these as the control templates. The *Order:* items represent the browse orders.

2. **Highlight** the *Order: Book.ByTitle* branch.

   One of the many things we can control here is the locator type. Lets change this one to Entry.

3. **Highlight** the *LOCATORTYPE* property.

4. **Select** *Entry* from the droplist.

This property is called an **Acorn** property. This means that by changing its value you will cause other Wizatrons to come into existence or be destroyed. There are many Acorns throughout the system. The BROWSEFILES property is also an Acorn which spawns the creation of many other Wizatrons as you have already seen. Notice that no special Wizatron code exists to create these Acorns, it's purely a function of the Wizatron engine.

5. **Scroll** the *Application Components* tree until the *BrowseBook* branch is the top item displayed in the list. Expand the *BrowseBook* > *Window* branch. Located near the bottom are *Entry: Book.Title* and *Prompt: Book.Title.*

   Notice that since an entry locator needs an ENTRY control, an ENTRY and PROMPT control have been added under the *BrowseWindow* branch—automatically.

Save and Generate the Application Component

Let's see the result of our work to this point, and allow the Wizatron to generate our application. Let's also save the work to this point, and use it in a later tutorial.

1. **Press** the **Save Application...** button located on the Wizatron toolbar.

   The **Save Application** dialog displays.

2. **Type** *TUTORIAL.APT* in the File name dialog, then **press** the **Save** button.

   Our application is now saved to use again.

3. **Press** the **Generate** button located on the Wizatron toolbar.

   The Wizatrons are automatically creating your application for you. When completed, your application should now look something like this:

4. On the toolbar, **press** the **Run** button to compile and run your application.

Examining the Running Application

1. **Choose** **Help ➤ About** from the running application's main menu.

   Notice the button on the About window is in the Southwest region and the author text is in the Southeast as we requested it:

2. **Press** the About Window **Close** button.

Let's examine the browse procedures.

3. **Choose** **Browse ➤ Browse Book.**

   You should see a lime colored list box and an ENTRY control for the locator.

4. When you have finished examining the Application, **choose** **File ➤ Exit** to close the application.

Summary - What Did I Just Do?

Obviously, you could have spent more time with the Wizatrons to polish the look and feel of the application. In the next tutorial, we will examine the Wizatron Configuration Utility, which will allow us to create a more permanent repository of our various property modifications. To summarize what we have just learned in this tutorial:

- You examined the Style View, its layout of the Wizatron's components, and its associated properties.
- You demonstrated the technique of property inheritance by modifying the default color of all list boxes created with a default style.
- You used property macros to modify the text on a Splash Window.
- You examined property functions that are used to dynamically retrieve information stored in your Data Dictionary.
- You examined the Application View, its layout of the application's components, and associated properties as generated by the Wizatron.
- You examined the specific properties which you can modify for a particular application using the Application View.
- You examined the concept of Space Allocation as it applies to certain Wizatron control components.
- You built an application from the Application View, and saved your work for later use.
- You learned how to add or modify specific features of your application using the Application View.

Using The Wizatron Configuration Utility

In our last tutorial, we examined the Style View which displays default Wizatron components and their properties that are contained in a particular style folio of an active application.

The Wizatron Configuration Utility is a further extension of this view, adding additional control and modification capabilities to a selected style file. This utility is designed as a aid to Wizatron style and folio development.

Starting Point:
Starting where we left off in the last tutorial, you should now be back in the Application Generator interface. If you took a break after the last section, make sure to reload the WIZTUTOR.APP before continuing.

1. **Choose** **Application ➤ Template Utility...**
2. **Highlight** the *WizardConfiguration* branch under *Class Wizatrons*, then **press** the **Select** button.

3. **Select** the *TUTORIAL.STF* file in the *\Clarion5\Template* directory when asked to select a style folio, then **press** the **Open** button.

The Wizard Configuration utility allows you to maintain style folios, but it allows much more than that.

Folio Management

Since we are planning modifications in this next section, let's create a new style file to store these changes.

1. **Choose** **Folio**. The *Folio* window is displayed.
2. **Select** the **Advanced Folio Management** tab.

The Folio window displays.

3. **Press** the **Combine** button.
4. **Highlight** both style files listed in the *Combine Styles* window, and **press** the **Ok** button.

   This combines the highlighted styles into a new style file.
5. In the **Save Style** dialog, type *TUTORIAL.STY* and **press** the **Save** button.
6. **Press** the **Yes** button when you are asked to replace the existing *TUTORIAL.STY* file.

   We now have an amalgamated file listed below. This confirms that our combine function has been succesfully completed:

7. **Press** the **Save** button to save the changes to the *TUTORIAL.STF* folio. **Press** the **Yes** button when you are asked to replace the existing *TUTORIAL.STF* file.

   Our preliminary setup work is now complete. We are now ready to make changes to the active folio.
8. **Press** the **Close** button to close the **Folio** window.

Examining Acorn Properties

Lets look at some of the things you can do with acorn properties:

1. RIGHT-CLICK on the **Style Components** list box.
2. **Choose** **Find** from the popup menu.
3. **Type** *TOPSPEED:FRAME* in the **Find What** field, then **press** the **Next** button.
4. **Highlight** the **SplashWindow** property located in the **Property** list box.

   This property is an acorn, as discussed in the last tutorial. Since it is a boolean property(Yes/No) it causes another Wizatron to exist if it is true and destroys an existing Wizatron it if it is false.

   You can control the Wizatron that it manages by the use of its sub-property attributes:

1. RIGHT-CLICK on the **SplashWindow** property located in the **Property** list box.
2. **Choose** **Edit selected Property** from the popup menu.

Let's focus on the first three property attributes shown in the window.

The *ACORN.CONTEXT* property contains an optional context string. This is a comma delimited list of properties and values, and is applied to the Wizatron automatically when created. It allows specific properties in the child Wizatron to be set upon creation.

The *ACORN.NAME* property identifies the name that will appear in the **Application View** for the created Wizatron (the name given to the Wizatron instance).

The *ACORN.WIZARD* property names the specific Wizatron to create.

3. **Press** the **Close** button to leave the current sub-property window.

Here is another example of the use of acorn properties:

1. **Highlight** the **TOPSPEED:MenuManager** component located in the **Style Components** list box.
2. **Highlight** the **StandardMenuItems** property located in the **Property** list box. RIGHT-CLICK and **choose** **Edit selected Property** from the popup menu.

Note here that our acorn properties contain macros. This demonstrates the acorn's hidden power.

- The *ACORN.NAME* property contains **ITEM: %_VALUE_%** which identifies each comma delimited value to be assigned its own distinct branch in the **Application View** (i.e. ITEM:Quit, ITEM:Cut, etc.)
- The ACORN.WIZARD property contains **MenuItem#%_VALUE_%** which identifies each comma delimited value to be assigned its own distinct branch in the **Style View** (i.e. - TOPSPEED:MENUITEM#:Quit, TOPSPEED:MENUITEM#:Cut, etc.)
- *ACORN.CONTEXT* contains a context string, **POPULATOR(%NAME%)** which in this case simply forces the populator property of the newly created Wizatron to %Name% (i.e., the value of this Wizatron's name property).

To summarize, acorns are very powerful when used with multi-valued properties to produce a whole family of new Wizatron instances.

3. **Press** the **Close** button to close the sub-property window.

Creating New Wizatron Style Components

Let's apply the concepts just discussed to practical uses.

In this section, we want to create a new type of CloseButton. Lets call it a "shutdown" button.

In this example, the shutdown button is similar to the close button, but has additional functionality such as disconnecting from an external data source (common in Client/Server applications).

We will base this new button upon the existing close button, but in real practice it may come from a custom button, in which case you have hand coded a shutdown process and already converted it to an associated template.

Our new button is going to be directly derived from *TOPSPEED:BUTTON*.

1. **Highlight** the **TOPSPEED:Button** property located in the **Style Components** list box.

2. RIGHT-CLICK and **Choose** **Create New Defaults Based On...** from the popup menu.

3. **Type** *Shutdown* in the field shown above, and **Press** ENTER**.**

You should now have a new TopSpeed:Button#ShutDown component. Let's set up its default properties:

1. **Highlight** the *TOPSPEED:BUTTON#SHUTDOWN* property located in the **Style Components** list box.
2. **Highlight** the **Text** property located in the **Property** list box.
3. CLICK on the **Text** property to enter the edit-in-place mode.
4. **Type** *&ShutDown* in the Value field. **Press** the DOWN ARROW key to advance to the next property.
5. In the **Tooltip** property, **Type** *Shutdown and disconnect from data source.*
6. **CLICK** on the **Use.Label** property (which saves your last entered value) and **Type** *?Shutdown.*
7. **CLICK** back on the **Text** property to save and review your work.

   Here is a part of what you should see at this point:

   Now we are going to change the close button control component to populate this item rather than the standard close button control.

1. **Highlight** *TopSpeed:ButtonControl#Close* in the *Style Components* list box.

   You may wish to use the **Find Wizatron Defaults** option to help expedite your search.

2. CLICK on the **Button Control Type** property located in the **Property** list box, and **type** *Shutdown* in the Value field provided.
3. CLICK on the **Control Template** property located in the **Property** list box, and **type** *CloseButton* (replacing the previous text) in the Value field provided.

   At this time our ShutDown button will act just like a normal Close Button. In practice, you would write your ShutDown Template and provide its name instead.

Saving Component Changes

For now, let's save our work to this point.

1. **Choose ➤ Folio** from the menu options.

   The Folio window displays.

2. **Select** the Advanced Folio Management tab.

   The amalgamated file, *TUTORIAL.STY*, is listed.

3. **Highlight** the *TUTORIAL.STY* entry, and **press** the **Save To** button.

4. **Press** the **Yes** button, when you are asked to save your changes.

> **Note:** **It is important to remember that a normal save of a Style or Application text file is always based on *incremental* changes. Always write your changes to a new file name, and later you can combine your changes into the original base style file if you wish. If you attempt to use a standard save to the original folio style file, it will overwrite all other base components not changed. Use the Save To option instead to preserve your changes *and* the original components.**

5. From the menu, **choose File ➤ Quit.** Since we have already saved the style folio changes, **press** the **No** button when asked to save the style again.

   You should now return to the Application Generator.

Applying the Changes

Let's remake the WIZTUTOR application that was generated in the last tutorial, this time using the modified style file.

1. **Choose Application ➤ Template Utility...**

2. **Highlight** the *ApplicationWizatron* branch under *Class Wizatrons*, then **press** the **Select** button.

At this time, you should see the *Prompts for ApplicationWizatron* window.

3. Make sure that all *QuickStart* options are turned **off** (Check boxes are unchecked). **Press** the **Ok** button to continue.
4. **Select** the *TUTORIAL.STF* file in the *\Clarion5\Template* folder when asked to select a style folio, then **press** the **Open** button.

   We will continue to work in the Application View.
5. **Choose Window ➤ Application.**
6. **Choose File ➤ Application ➤ Load**.
7. **Select** the *TUTORIAL.APT* file in the *\Clarion5\WizTutor* folder when the **Load Application** window is displayed, then **press** the **Open** button.

   Our work from the previous tutorial should now be loaded, now merging the Shutdown button addition from this tutorial.
8. **Press** the **Generate** button.
9. **Make** and **run** the WIZTUTOR application.
10. When the WIZTUTOR application is running, Open the *BrowseBook* procedure.

    We can now see that our new Shutdown button and control template have been populated where a standard close would have previously been used.

This concludes this section of the tutorial.

Adding New Wizatron Components and Associated Acorn Properties

Overview

This section illustrates one of the most powerful techniques you can use to add new functionality to existing Wizatrons.

The Wizatrons contain a finite set of features which you can use in your applications. But what do you do when you need additional features added to your Wizatrons, those that are not part of the Clarion set?

To add additional features to a normal application outside of the Wizatrons, you can add additional templates to the Application Generator.

Similarly, all Application and Procedure Wizatrons rely in part on registered templates to generate the source code for your generated applications.

To add new functionality to an existing Wizatron style file, you can derive new control and extension templates as new Wizatron components.

These new Wizatron components are then implemented by new acorn properties which conditionally generate these new templates based on the property values you enter.

Let’s illustrate this technique.

Step One: If not in template format, convert your new functionality to a Control or Extension template.

If you already have your functionality included in a template or template wrapper, you would proceed to *Step Two*. Stay with us for a moment as we walk you through a typical example.

If your functionality is in the form of embedded source, see the *Template Wizatron* section in the previous chapter. You can use the Template Wizatron as a template writing tool, to produce a template that can be included into the Clarion Template Registry.

The example we will show you in this tutorial has a template that is ready to register. Let's locate and register it now.

1. **Choose** **Setup ➤ Template Registry.**
2. In the Template Registry dialog, **press** the **Register** button.
3. In the Template File dialog, select the *WIZTOOLS.TPL* file in the ..\EXAMPLES\WIZTOOLS subfolder, then **press** the **OK** button.

   The registered template components are displayed.

4. In the Template Registry dialog, **press** the **Close** button.

Template Description

Our WIZTOOLS template family contains a template extension designed to play a sound file when a window opens. This action is desirable when you wish to grab the user's attention. We will add this functionality to the Wizatron's About Window.

Step Two: Add your new templates to the Wizatron style components.

There are two extension templates that we will need to add to the Wizatrons in this tutorial.

- A global extension that prototypes the “Play Sound” API .
- A procedure extension that calls the API after a window is opened.

Starting Point:
Make sure to reload the WIZTUTOR.APP that you saved in the last tutorial. We will start this tutorial from the Application Tree.

Add the APIMap Global Template Extension

1. **Choose** **Application ➤ Template Utility...**
2. **Highlight** the *WizardConfiguration* branch under *Class Wizatrons*, then **press** the **Select** button.

3. **Select** the *CUSTOM.STF* file in the *\Clarion5\Template* directory when asked to select a style folio, then **press** the **Open** button.

   It really doesn’t matter which folio we select at this time, because we will save all of our work to a new style and new folio before completing this tutorial.

4. RIGHT-CLICK on the **Style Components** list box, and **choose** **Find** from the popup menu.
5. **Type** *TOPSPEED:ControlTemplate#GlobalTemplate* in the **Find What** field, and **press** the **Next** button.

   The following item is selected:

6. RIGHT-CLICK and **Choose** **Create New Defaults Based On...** from the popup menu.

7. **Type** *APIMap* in the field shown above, and **press** the **Ok** button.

   You should now have a new component called TopSpeed:ControlTemplate#GlobalTemplate#APIMap. Let's set up its default properties.

8. **Highlight** the **TOPSPEED:ControlTemplate#GlobalTemplate#APIMap** component located in the **Style Components** list box.

9. CLICK on the **ControlTemplate** property, and **type** *ApiMap* in the Value field.

10. CLICK on the **TemplateChain** property, and **type** *WizTools* in the Value field.

    We are telling the Wizatrons to use the APIMap template extension located in the WizTools template family. Here is our new Global Template Wizatron:

Add the PlaySound Procedure Template Extension

1. **Highlight** the **TOPSPEED:ControlTemplate** style component.

   The following item is selected:

2. RIGHT-CLICK and **choose** **Create New Defaults Based On...** from the popup menu.

3. **Type** *PlaySound* in the field shown above, and **press** the **Ok** button.

   You should now have a new TopSpeed:ControlTemplate#PlaySound component. Let's set up its default properties:

4. **Highlight** the **TOPSPEED:ControlTemplate#PlaySound** component located in the **Style Components** list box.

5. CLICK on the **ControlTemplate** property found in the **Property** list, and **type** *PlaySound* in the Value field.

6. CLICK on the **TemplateChain** property, and **type** *WizTools* in the Value field, and **press** the **Ok** button.

   We now need to add an additonal property in this Wizatron component that stores the name of the WAV file to play.

7. **Highlight** the **TOPSPEED:ControlTemplate#PlaySound** component again, **RIGHT-CLICK,** and **choose** **New Property** from the popup menu.

   The following window displays.

8. **Highlight** the *(Property Name)* property, and **type** *SoundFile* in the Value field.

   This is the name of the property that will hold the WAV file name.

9. **Highlight** the *Description* property, and **type** *Select a WAV File:* in the Value field.

   The description to use in the Wizard View.

10. **Highlight** the *DType* property, and **select** *FILE* from the drop list.

    The *SoundFile* property will now show an ellipsis button to allow you to select a file name via a file dialog.

11. **Highlight** the *Output.Name* property, and **type** *SoundFile* in the Value field.

    This is the name of the template symbol that will receive our WAV file name value.

12. **Press** the **Close** button to return to the Property List.

13. Finally, **type** *TEST.WAV* in the *SoundFile* Value field that you just created.

    This will be the default WAV file to play if you do not override it.

We have completed the process of adding new Wizatron components to our current design session. Continue now to *Step Three*.

Step Three: Add new acorn properties to target components that will grow your new Control Template components.

We now need to attach our new control template Wizatrons to other appropriate style components. We'll attach the *ApiMap* Wizatron to the *Application* component, and attach the *PlaySound* Wizatron to the *Window* component of *AboutWindow*.

EnableSoundAPI Global Acorn Property

1. **Locate** and **highlight** the *TOPSPEED:Application* style component.

   This is the component located at the bottom of the Style Components tree.

2. RIGHT-CLICK and **choose** **New Acorn** from the popup menu.

   The Create New Acorn window displays.

3. Enter the values *exactly* as shown below:

   Prompt **Value**

   Property Name: EnableSoundAPI

   Wizatron Type: ControlTemplate#GlobalTemplate#APIMap

   Leave the *Set Properties:* prompt blank, and leave the *Always Deploy* value set to **YES**.

4. **Press** the **Ok** button to add the Acorn Property.

   The Property Attributes window displays.

5. **Press** the **Close** button to return to the Property List.

   This acorn property automatically links in the proper API prototyping information.

   Let's add one more acorn to allow us to call the Play Sound API from the Wizatron generated About Window.

PlaySoundOnOpen Window Acorn Property

1. **Locate** and **highlight** the **TOPSPEED:Window#About** style component.

   This is a child component of the TOPSPEED:Window Style Component, located about halfway into the tree.

2. RIGHT-CLICK and **choose** **New Acorn** from the popup menu.

   The following window displays.

3. Enter the values *exactly* as shown below:

Prompt	Value
Property Name:	PlaySoundOnOpen
Wizatron Type:	ControlTemplate#PlaySound

   Leave the *Set Properties:* prompt blank, and leave the *Always Deploy* value set to **YES**.

4. **Press** the **Ok** button to add the Acorn Property.

   The Property Attributes window displays.

5. **Press** the **Close** button to return to the Property List.

   This acorn property automatically links the proper API call to play a WAV file when the About Window opens. We have a default WAV file ready and have instructed the Wizatron to automatically play it.

   Before testing, let's save our work and incorporate it into a new folio.

Step Four: Test and save your work.

The most important step in this process is to save your work at this time.

1. **Choose ➤ Save Style** from the menu options.

   We will write all changes to a single style file, and merge it into a new folio. You should automatically be in the ..*TEMPLATE* folder when the *Save Style* dialog is opened.

2. **Type** *SOUND.STY* in the file name prompt, and **press** the **Save** button to write the incremental changes.

3. **Choose ➤ Folio** from the menu options.

   The Folio window should now be active.

4. **Select** the Advanced Folio Management tab.

5. **Press** the **Add** button to add our style changes to the active folio.

   The following window displays.

6. **Enter** all values as shown in the window above, and **press** the **Ok** button to update and close the window.

   If you loaded the CUSTOM.STF folio at the start of this tutorial, your folio entries should now look like this.

7. **Press** the **Save** button.

   We will now write these folio entries to a new folio file, SOUND.STF, which we will now create.

8. **Type** *SOUND.STF* in the File Name entry field, and **press** the **Save** button to save the new folio name.

   This completes our Wizatron modifications. Let's exit the Wizatron Configuration Utility and test the results.

9. From the menu, **choose File ➤ Quit.** Since we have already saved the style folio changes, **press** the **No** button when asked to save the style again.

   You should now return to the Application Generator.

Applying the Changes

Let's remake the WIZTUTOR application that was generated in the last tutorial.

1. **Choose Application ➤ Template Utility...**
2. **Highlight** the *ApplicationWizatron* branch under *Class Wizatrons*, then **press** the **Select** button.

   At this time, you should see the **Prompts for ApplicationWizatron** window.
3. Make sure that all *QuickStart* options are turned **off** (Check boxes are unchecked). **Press** the **Ok** button to continue.
4. **Select** the *SOUND.STF* file in the *\Clarion5\Template* folder when asked to select a style folio.

   This application does not need to generate any additional procedures at this time. We are simply testing the new functionality we have added. The *About Window* is already prepared for us by the Wizatrons.

   Since we have preset all options to play the WAV file automatically, all we need to do at this time is generate.
5. **Press** the **Generate** button.
6. **Make** and **run** the WIZTUTOR application.
7. When the WIZTUTOR application is running, **choose** **Help ➤ About** from the menu.

If you hear a familiar sound of glee, you have completed the tutorial correctly.

INDEX

Symbols

%DATELONG% ... 89
%DATESHORT% ... 89
%TIMELONG% ... 89
%TIMESHORT% ... 89
%YEAR% ... 89

A

Acorn Properties ... 182
Acorn Properties, using ... 187
Acorn Property ... 81
Acorn.Context ... 100
Acorn.InitPriority ... 100
Acorn.Name ... 101
Acorn.Value ... 101
Acorn.Wizard ... 101
Acorns ... 100
Advanced Tutorials ... 169
Allocation. ... 85
Allocation.Functional ... 117
Allocation.Height ... 86, 117
Allocation.Ignore ... 117
Allocation.Output ... 117
Allocation.Priority ... 86, 118
Allocation.ShowSingleTab ... 118
Allocation.Tab ... 85, 118
Allocation.Width ... 86, 118
Allocation.XPos ... 86
Allocation.Xpos ... 118
Allocation.XSpacing ... 119
Allocation.YPos ... 86
Allocation.Ypos ... 119
Allocation.YSpacing ... 119
Allocation.Zone ... 86, 119
Allocation.Zone.List ... 119
Alrt ... 119
Alternative Style Folio ... 45
Always Deploy ... 74
Anatomy of the Wizatrons ... 13
Application, Building an ... 40
Application Components Popup ... 70
Application View ... 60, 177
Application View, Popup Menus ... 70
Application Wizatron ... 10
At. ... 84
At.CostFactor ... 85, 120
At.EastMargin ... 84, 120
At.EastOffset ... 84, 120
At.Height ... 84, 120
At.MaxHeight ... 85, 120
At.MaxWidth ... 85, 121
At.MinHeight ... 85, 121
At.MinWidth ... 85, 121
At.NorthMargin ... 84, 121
At.NorthOffset ... 84, 121
At.SouthMargin ... 84, 122
At.SouthOffset ... 84, 122
At.WestMargin ... 84, 122
At.WestOffset ... 84, 122
At.Width ... 84, 122
At.XPos ... 122
At.YPos ... 123
Auto-Increment settings ... 23

B

BestWinHeight ... 90
BestWinWidth ... 90
Blocker ... 102
Browse Wizatron ... 10
BrowseFiles ... 123
BrowseFiles.List ... 123
BrowseListControl ... 123
Built-in Functions ... 88
Built-in Macros ... 89
ButtonControl ... 123
ButtonControlType ... 124

C

Calculate Interest Levels ... 27
Calculate Text Primary key ... 27
Case ... 124
Case.List ... 124
Categorization and Structure ... 78
Change URL List... ... 69
ChangeButtonControl ... 124
CHECK ... 103, 124
CheckType ... 102
ChildBrowses ... 125
ChildBrowses.List ... 125
CIF ... 125
COLOR ... 103
Color.Basic ... 125
Color.SelectedBack ... 125
Color.SelectedFore ... 126
Column.Colors ... 126

Column.Header ... 126
Column.HeaderIndent ... 126
Column.HeaderJustification ... 126
Column.HScroll ... 126
Column.Icon ... 127
Column.ID ... 127
Column.Indent ... 127
Column.Justification ... 127
Column.MaxWidth ... 127
Column.Picture ... 127
Column.Resizable ... 128
Column.RightBorder ... 128
Column.Tree ... 128
Column.Width ... 128
Contract Branch ... 71
ControlLabels ... 128
ControlResize ... 129
Controls ... 128
ControlTemplate ... 129
ControlToLookup ... 129
ControlType ... 129
CostTarget ... 129
COUNT ... 90
Create New Defaults Based On... ... 74
Create New View ... 69
Creating New Wizatron Components ... 188
CURSOR ... 103
Cursor ... 129
Cursor.List ... 130
CUSTOM.STF ... 46

D

DEFAULT ... 91
Default ... 130
Default Property ... 81
Delete Branch ... 70, 73
DeleteButtonControl ... 130
Derived Defaults ... 82
DESCRIPTION ... 91
Description ... 102
Design Environment ... 59
Design Tips, Style File ... 157
DevImp ... 103
DevImp.List ... 103
Dictionary Attributes ... 108
Dictionary Wizatron ... 12, 18
Dictionary Wizatron, loading ... 20
Dictionary Wizatron Results ... 30
Dictionary Wizatron Summary ... 29
Disable ... 130
Display ... 109
DISPLAY attribute ... 31
DisplayColumns ... 130
DisplayColumns.List ... 131
Documentation ... 103
Documentation, exporting ... 113
Documentation, HTML Tags ... 116
Documentation, Property ... 76
Documentation, Samples ... 117
Documentation Strategy ... 113
DragID ... 131
Drop.Count ... 131
Drop.Width ... 131
DropID ... 131
Dtype ... 103
Dyna ... 103
Dynamic Property, defined ... 81
Dynamic Property Values ... 88

E

Edit-In-Place Options ... 26
EditControl ... 131
EDITINPLACE attribute ... 31
EIP ... 132
EnforceMinimum ... 132
ENTRY ... 103
EVAL ... 91
Expand Branch ... 71
Expansion Macros ... 88
Exporting Documentation ... 114
EXTRACT ... 92

F

Factory Importances ... 64
FIELD ... 104
Field ... 132
Field Interest Level ... 24
Field Interest, Substitute ... 25
FieldPriority ... 132
FIELDS ... 92
Fields ... 132
Fields.List ... 132
FILE ... 104
File.Mask ... 104
File.Pattern ... 104
FILES ... 92
Filter ... 133
Find ... 70
Find (Style) ... 72
Find property ... 71, 75
Flat ... 133
FocusKey ... 133
Folio Management ... 185

Folio Menu 65
FONT 104
Font.CharSet 133
Font.Color 133
Font.Size 133
Font.Style 134
Font.Typeface 134
ForceRefresh 134
Form Wizatron 10
Format 134
FreeElement 134
From 134
FromDictionary 135
FromDictionary.List 135
Full 135
FULLNAME 92
Functions, Property 176

G

Generate 63, 135
Generate Application 42
Generate Browses 46
GenerateBrowses 136
Getting Started Tutorials 17

H

HasQuery 136
HasSelect 136
HasToolbox 136
HasUpdate 136
HEADING 93
HelpId 136
Hidden 137
HTML Documentation Tags 116

I

Icon 137
IconPosition 137
IconPosition.List 137
Image 137
ImageMode 137
Immediate 138
Implesseq 104
Impmoreeq 105
INDEX 93
INHERIT 93
Inherit 105
Inherited Property 80, 173
InsertButtonControl 138
Instance 138
Interest 108
Interest Level, Modify 25
Interest Levels, Calculate 27
InterestThreshold 138
Internal.FIELDS 138
Internal.LINK 139
Internal.ORIG 139
Internal.SEQ 139
Introduction 7
IsChild 139
IsProcedure 139
Item.Procedure 140
Item.Stack 140
Item.Thread 140

J

Justification 140
Justification.List 140
Justification.Offset 140

K

Key 141
Key File Calculations 27
KEYFIELD 94
KEYS 94

L

Label 141
List 105
ListFill.Auto 105
ListFill.Convert 105
ListFill.Func 105
Load Application 63
Load Style 63
LoadMethod 141
Locator 141
Locator.Entry 141
Locator.Prompt 141
LocatorType 142
Locked 105
Logical Name 77
LookupStyle 142
LookupStyle.Child.Button 142
LookupStyle.Child.Combo 142
LookupStyle.Child.Entry 142
LookupStyle.Child.Prompt 142
LookupStyle.List 143

M

MacroOwnerContext 105
Macros, Built-in 89
Macros, Property 175
Mask 143
MaxCostCycles 143
Menu 143
Menu Commands 63
MenuGrouping 144
MenuGrouping.List 144
MenuManager 144
Menus 144
Menus.List 144
Message 144
Message Display Threshold 64
Meta-Programming 8
Modify Interest Level 25
Multi 106
MULTIDROP 104

N

NAME 94
Naming Conventions 77
New Acorn 73
New Property 73
New Property Name 73
New Wizatron Components 188
Next.Control.Instance 145
Node.ChildProcedures 145
Node.Controls 145
Node.ControlWizards 145
Node.Frame 145
Node.Items 145
Node.SheetAllocator 146
Node.SpaceAllocator 146
Node.ToolAllocator 146
NoMerge 146
NoPopulate 110

O

Output.Category 146
Output.Depends 106
Output.Description 146
Output.From 147
Output.Name 106
Output.ProcName 147
Output.Type 106
Output.Value 106
OVER settings 23
Overridden Property 81, 174

P

PARENT 94
ParentFile 147
Password 147
Physical name 77
Picture 147
Populator 89, 147
Popup Menus, Application View 70
Popup Menus, Style View 72
PostLookupField 148
PostLookupKey 148
PostLookupProcedure 148
PrimaryFile 148
PrimaryKey 148
Procedure Wizatrons 10
Procedure Wizatrons, using the 54
ProcsPerModule 149
PROMPT 95
Prompt Detail 21
Prompt Detail Level 37
Prompts 21
Prompts and Headers
  Dictionary Wizatron 28
Properties (Values) 80
Property
  Acorn 81
  Default 81
  Dynamic 81
  Functions 176
  Inherited 80, 173
  Macros 175
  Overridden 81, 174
  Standard 80, 173
Property Documentation 76
Property inheritance 173

Q

Quick Start 106
QuickStart View 61

R

RADIO 104
RangeField 149
RangeFile 149
RangeHigh 149
RangeLimit 149
RangeLimitType 149
RangeLow 150
ReadOnly 150

Registering the Wizatron Templates 10
RELATED 95
RELATIONFIELD 96
RELATIONKEY 95, 96
RELATIONS 96
REPORTCONTROLTYPE 97
Required 150
Reset Factory Importances 64
Resizer 150
Revert to inherited/default value 71, 75
Right 150

S

Save Application 63
Save Style 63, 76
SCREENCONTROLTYPE 97
SCREENPICTURE 97
Scroll 150
Scrollbars 151
ScrollBars.List 151
ScrollBehavior 151
ScrollKeyDistribution 151
SecondaryFiles 151
Separator 151
Set Properties 74
Sheet 152
Show 152
Show Controls Toolbar 64
Show Hidden Properties 76
Show Wizatron Image Toolbar 64
SINGLEDROP 104
Skip 152
Small 109
SortOrders 152
SortOrders.List 152
Space Allocation 84, 179
SPIN 104
SplashWindow 152
Standard 153
Standard Property 80, 173
Standard.List 153
STANDARD.STF 33, 35
StandardMenuItems 153
StandardMenuItems.List 153
Static Elements 77
Stop 106
Structure, Wizatron 78
Style File Design Tips 157
Style Properties Popup 75
Style View 60, 78, 172
Style View Popup Menus 72
Sub-Properties 99
Substitute Field Interest 25

T

TabName 153
Tabs 154
TabText 154
TargetResolution 154
Template Components 79
Template Wizatron 158
TemplateChain 154
TEXT 104
Text 154
Text Primary Key, Calculate 27
TextPrimary 109
Third Normal Form 12, 25
ToDo Procedures Window 62
  Multi-select 62
  Select All 62
  Unselect All 62
Toolbar 155
Tooltip 155
Transparent 155
Tutorial Three - Alternate Style Folio 45
Tutorial Two - The Application Wizatron 32
Type 107
TypingMode 155
TypingMode.List 155

U

UpdateProcedure 155
Use.AltEquate 156
Use.Label 156
Use.Number 156
USER 98

V

Validation 107
Value 107
Values (Property) 80
Viewing HTML documentation 115
Visual Components 79

W

Web Sites 69
When Apply 107
WindowManager 156
wizard 59
Wizatron Components 21, 37
Wizatron Configuration Utility 11, 185
Wizatron Engine 14
Wizatron Family 10
Wizatron features 9
Wizatron Type 73
Wizatron User Interface 14
Wizatron Views 170
Wizatrons, defined 9