# Contents

<table>
<thead>
<tr>
<th>Chapter 1</th>
<th>Overview</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Introducing ObjectStore Inspector</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>What Is Inspector?</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Opening ObjectStore Databases</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>How to Start Inspector</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Open the <strong>carsdemo</strong> Database.</td>
<td>20</td>
</tr>
<tr>
<td>Inspector Features</td>
<td>Main Database View Window</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>Data Views</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>Instance Window</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>Navigation Window</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>Physical Database Layout Window</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>Use Inspector to Create Multiple Views of the Same Database</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>User-Defined Methods</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>Edit Data Member Dialog Box</td>
<td>25</td>
</tr>
<tr>
<td>Sharing ObjectStore Database Information</td>
<td>Printing</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>Exporting</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>Using Inspector as an OLE Server</td>
<td>26</td>
</tr>
<tr>
<td>Inspector Options</td>
<td>Where You Set Options</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>How to Set Options</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>When Options Take Effect</td>
<td>28</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 2</th>
<th>Database Views</th>
<th>29</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Overview</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>What Is a Database View?</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Database Views Contrasted to Data Views</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>The Database View Window</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>Window Name</td>
<td>32</td>
</tr>
</tbody>
</table>
Contents

The Toolbar .......................................................... 32
The Database Roots Pane .......................................... 33
The Schema Pane ..................................................... 34
The Instance Pane ................................................... 35
Changing Pane Dimensions ........................................ 35
Creating a Custom Database View ................................. 36
Two Ways to Create .................................................. 36
Creating a Custom Database View Explicitly ..................... 36
Creating a Custom Database View Implicitly ..................... 36
Saving a Custom Database View ................................... 37
Deleting a Custom Database View ................................ 38
Modifying a Database View ........................................ 39
Schema Diagram Layout ............................................. 39
Abstraction Functions .............................................. 39
Instance Format ....................................................... 39

Chapter 3  
Data Views ......................................................... 41
What Is a Data View? ............................................... 42
A Data View Contrasted to the Instance Pane ................. 42
Working With Data Views ......................................... 42
Creating a Data View ............................................... 42
Opening a Data View ............................................... 44
Deleting a Data View ............................................... 45
Filtering a Collection ............................................... 45
What Is a Filter? ..................................................... 45
Two Ways to Define a Filter ........................................ 46
When Are Filters Applied? ........................................ 47
Defining a Filter Manually ......................................... 47
Defining a Filter Using Inspector ................................. 48
Defining Parameterized Constraints ............................. 50
Ordering a Collection ............................................... 53
When Does the Order Take Effect? ............................... 54
How to Define a Collection Order ................................. 54

Chapter 4  
Schema Diagrams ................................................... 55
Changing Diagram Notation ....................................... 55
Supported Notation .................................................. 55
How to Change Diagram Notation. ............................... 55
Changing the Default Notation ................................... 56
Changing the Appearance of the Schema Diagram ............ 56
Ways to Change the Diagram Appearance ....................... 56
Changing the Diagram Zoom Level ........................................... 56
Changing Class Layout ......................................................... 56
Changing the Shape of Relationship Routes ............................. 57
Hiding Relationships ............................................................. 57
Altering the Contents of a Schema Diagram ............................. 58

Chapter 5 Collection Grids and Lists ................................. 59
Overview ................................................................. 60
Collection Grid Compared to Collection List ....................... 60
Choosing a Display Format .................................................. 61
Populating Collection Grids and Lists ................................. 61
Refreshing Collections ......................................................... 62
Finding a string ................................................................. 63
Other Grid Features .......................................................... 65
Customizing a Collection Grid ............................................ 65
Customization Procedure ..................................................... 65
Changing Cell Dimensions ................................................... 66
Changing the Alignment of Cell Data ................................... 66
Customizing the Grid Border ............................................... 67
Changing the Font ............................................................ 68
Annotating the Grid .......................................................... 69
Saving Your Modifications ................................................... 70
Exporting a Collection Grid ................................................ 72
What Information Is Exported? .......................................... 72
How to Export a Collection Grid in XML ........................... 72
How to Export a Collection Grid in Text Format ................ 72

Chapter 6 Classes and Instances ................................. 73
Where Instances and Classes Are Displayed .................... 74
Instance Pane ................................................................. 74
Instance Window ............................................................. 74
Data Views ................................................................. 76
Instance: Physical Layout Property Dialog Box ................. 77
Class Details Dialog Box ................................................... 78
Physical Database Layout Window ..................................... 78
Customizing the Instance Display ..................................... 79
What Information Is Displayed by Default ....................... 79
Using the Instance Format Dialog Box ............................ 79
Displaying the Instance Format Dialog Box .................... 80
Selecting Data Members for Display ............................... 80
Associating an Icon with a Class ........................................ 82
Chapter 8  Roots  ...................................................... 115
Creating a Root ................................................. 116
  Two Parts to Creating a Root ................................. 116
  Choosing a Root Value ........................................ 116
  Creating a Root Only .......................................... 116
  Creating a Root and Defining the Root Value .............. 117
Working with Roots ............................................. 117
  Redefining a Root Value ..................................... 117
  Deleting a Root .................................................. 118

Chapter 9  Tools for Physical Analysis  ............... 121
Layout Window .................................................. 122
  Opening the Physical Database Layout Window .......... 122
  Three Panes .................................................... 122
  Filling the Panes .............................................. 123
  Refreshing the Panes ......................................... 123
  Window Options .............................................. 124
Segments ......................................................... 124
  Purpose of the Segments Pane .............................. 125
  What Segments Are Selected? ............................... 125
Segment Pages .................................................. 125
  The Segment Properties Dialog Box ...................... 126
  Segment Options ............................................. 127
Statistics ......................................................... 127
  What Information Is Displayed ............................. 127
Instances ........................................................ 128
  What Information Is Displayed ............................. 128
  Filling the Instances Pane .................................. 129
  Selecting Classes ........................................... 129
  Instances Options ........................................... 129
Working with Free Space .................................... 129
  Retrieving Space and Free Space Information .......... 130
Contents

Calculating Free Space ........................................... 130
Tools for Debugging ............................................. 131
How to Locate an Instance .................................... 131

Appendix A  
Printing. ......................................................... 133
Printing Schema Diagrams ..................................... 134
Before You Begin ............................................... 134
Controlling Printed Output .................................. 134
How to Print a Schema Diagram ............................. 134
Printing Collection Grids ...................................... 135
Before You Begin ............................................... 135
Page Setup Options .......................................... 136
Printing Navigation Trees .................................... 138
Before You Begin ............................................... 138
Controlling Printed Output .................................. 138

Appendix B  
Using Inspector as an OLE Server ......................... 141
Overview of Inspector as an OLE Server .................. 141
Modifying Inspector Objects ................................. 142
Inserting Inspector Objects ................................ 142
In-Place Editing ................................................. 144

Appendix C  
Working with Metaknowledge ............................... 145
Overview of Metaknowledge .................................. 146
Importing Metaknowledge ................................... 146
Updating Metaknowledge ................................... 148
Ignoring Metaknowledge ..................................... 148

Appendix D  
Configuring Inspector ......................................... 149
When to Use the Configuration Utility .................... 149
  Using ossetasp Instead ...................................... 149
About the Application Schema File ....................... 150
About ObjectStore Servers .................................. 150
  Rawfs Management ......................................... 150
  Multiple Servers on a Host ................................ 150
Configuration Process ....................................... 150
Starting the Configuration Utility ......................... 151
  Prerequisite ................................................. 151
  How to Start the Configuration Utility ................ 151
Evaluating the Installation Status ......................... 152
  Configuration Status is Summarized .................... 152
  Status Summary Supplements ............................ 153
Changing the Inspector Configuration .................... 153
Contents

Choosing a Server .................................................. 154
How to Change the Inspector Configuration ................. 154
Copying the Schema to a Local Drive ....................... 155
Copying the Schema to a Remote Server ..................... 156
Copying the Schema to a rawfs Database ................... 157
Error Conditions ..................................................... 157
  Error Copying to a Network Drive ............................ 158
  Error Copying to a rawfs Database .......................... 158
  Error Running ossetasp ......................................... 158

Index ............................................................... 161
# Preface

## Purpose

The *ObjectStore User Guide* introduces ObjectStore Inspector and describes how to use it to browse, edit, query, and report on data in ObjectStore databases. You will learn:

- About Inspector’s graphical interface and the tools that help with logical and physical aspects of ObjectStore database analysis
- How to create custom views of ObjectStore databases
- How to create, read, update, and delete ObjectStore objects from Inspector
- How Inspector helps you share ObjectStore database information with reporting, data exporting, and OLE server features

## Audience and Scope

This guide is for ObjectStore application developers. It assumes some level of familiarity with ObjectStore database concepts and procedures. If you are just getting started with ObjectStore, consider the following sources of information:

- *Building ObjectStore C++ Applications* provides information and procedures for generating schema, compiling, linking, and debugging.
- *ObjectStore C++ API User Guide* describes how to use the basic C++ programming interface to ObjectStore to create database applications.

## How This Book Is Organized

This book contains the following chapters and appendixes:

- Chapter 1, Overview, on page 17 describes the main features of Inspector and describes how to set options.
- Chapter 2, Database Views, on page 29 describes the Database Root, schema, and instance panes of the main database view window and tells you how to create custom database views.
- Chapter 3, Data Views, on page 41 describes how to create a data view, and provides procedures for filtering, ordering, and formatting collections.
- Chapter 4, Schema Diagrams, on page 55 describes how to work with the schema pane of the main database view window.
- Chapter 5, Collection Grids and Lists, on page 59 describes how to work with collection grids and lists in the instance pane of the main database view window and in data views.
- Chapter 6, Classes and Instances, on page 73 describes how Inspector handles class and instance information, including where and how instance information is.
displayed, how you can change the instance format, and how to edit data members.

- Chapter 7, User-Defined Methods, on page 103 tells you how to register and invoke user-defined methods in Inspector to perform create, read, update, and delete operations on ObjectStore objects.

- Chapter 8, Roots, on page 115 describes how to work with roots in Inspector.

- Chapter 9, Tools for Physical Analysis, on page 121 describes the Physical Database Layout window and other tools to help you examine an ObjectStore database at the segment and page level, and to work with free space.

- Appendix A, Printing, on page 133 describes how to print schema diagrams, collections grids in the instance pane, and navigation trees.

- Appendix B, Using Inspector as an OLE Server, on page 141 describes the Inspector objects you can use as OLE servers in OLE container application documents.

- Appendix C, Working with Metaknowledge, on page 145 describes what metaknowledge is and how to use it to leverage customization work across similar databases.

- Appendix D, Configuring Inspector, on page 149 describes how to use the ObjectStore Inspector Configuration Utility to make the application schema database associated with Inspector accessible to an ObjectStore server.

Prerequisites

ObjectStore Inspector requires ObjectStore Release 7.1 or later. An ObjectStore 7.1 client is bundled with Inspector. An ObjectStore 7.1 server must be reachable from the client machine on which Inspector has been installed.

For more information, see the online Help for the ObjectStore Inspector Configuration Utility.

Progress Software on the World Wide Web

The Progress Software ObjectStore Web site (web.progress.com/objectstore) provides a variety of useful information about products, news and events, special programs, support, and training opportunities.

Technical support is available to you as part of a valid maintenance contract. There are several ways to obtain information about purchasing maintenance:

- Through our Web site at web.progress.com/about/contact.html.
- By sending e-mail
  - In North America, customerservices-americas@progress.com
  - In Europe, the Midde East, and Asia, customerservicesemea@progress.com
  - In Asia Pacific, customerservices-asiapac@progress.com
- In North America, by calling 1-781-280-4833.

After you purchase a maintenance contract, you can:
Send questions to ostore-support@progress.com. Remember to include either the phrase "Technical Support Issue" or a valid Progress serial number as part of the subject line or your e-mail cannot be processed.

Call the technical support organization to get help resolving problems. If you are in North America, call 1-781-280-4999. If you are outside North America, refer to the Technical Support Web site at web.progress.com/objectstore/contact-objectstore-support.html.

File a report or question with technical support by going to http://web.progress.com/objectstore/objectstore-support.html.

Access the following on the technical support Web site:
- A template for submitting a support request. This helps you provide the necessary details, which speeds response time.
- Solution Knowledgebase that you can browse and query.
- Online documentation for all products.
- White papers and short articles about using Progress products.
- The latest versions of products, service packs, and publicly available, downloadable patches.
- Support matrix that lists platform configurations supported by this release.
- Support policies.
- Local phone numbers and hours when support personnel can be reached.

Technical Support can more quickly resolve your issue when you provide the appropriate information. You should always provide the following information when contacting technical support:
- Serial number for your software licence
- Case number (if you are calling about a previous problem)
- Your name
- Your company name and location (city, state or province, and country)
- Contact information
- Product name and version
- Operating system
- The urgency of the issue being reported
- Detailed problem description including specific symptoms, observations, and any troubleshooting steps you attempted

Education Services

To learn about standard course offerings and custom workshops, use the Progress education services site (web.progress.com/education). If you are in North America, you can call 1-800-477-6473 x4452 to register for classes.

Information on current course offerings and pricing is also available by sending an e-mail message to classes@progress.com.
Searchable Documents

In addition to the online documentation that is included with your software distribution, the full set of product documentation is available at the Progress Software Developers Network (PSDN) section of the Progress Communities site at http://communities.progress.com/pcom/docs/DOC-16071. The site provides documentation for the most recent release as well as archived documentation sets from previous releases. Be sure to check this site for new information or documentation clarifications posted between releases.

Your Comments

Progress product development welcomes your comments about its documentation. Send any product feedback to ostore-support@progress.com. To expedite your documentation feedback, begin the subject with Technical Support Issue; Doc: For example:

Subject: Technical Support Issue; Doc: Incorrect message on page 76 of reference manual

Third-Party Acknowledgments

This software makes use of the following third party products:

• Ant v1.6, Mortbay Jetty v6.1 and JXPath v 1.2. See the Apache License v2.0 in the installation directory in the docs/ThirdPartyLicenses folder for license information.

• Expat v9.5.1. Copyright © 1998, 1999, 2000 Thai Open Source Software Center Ltd and Clark Cooper. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

• The Java SE Runtime Environment (JRE) Version 6, developed by and copyright Sun Microsystems. See the Sun Microsystems, Inc. Binary Code License for the Java SE Runtime Environment (JRE) Version 6 and THIRDPARTYLICENSESREADME.txt in the installation directory in the docs/ThirdPartyLicenses folder for license information.
• Jchart2d v2.2.0. The contents of these files are subject to the GNU Lesser General Public License v.2.1 (the "license"). You may not use these files except in compliance with the license. You may obtain a copy of the license in the installation directory in the docs/ThirdPartyLicenses folder and a copy of the license and source code of these files can be obtained through www.psdn.com by following the instructions set forth therein.

• JSON. Copyright © 2002 JSON.org. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. The Software shall be used for Good, not Evil. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

• OpenDMK v 1.0-b2 and Restlet v 1.1m2. The contents of these files are subject to the Common Development and Distribution License (CDDL) Version 1.0 (the "License"). You may not use these files except in compliance with the License. You may obtain a copy of the License in the installation directory in the docs/ThirdPartyLicenses folder and a copy of the license and source code of these files can be obtained through www.psdn.com by following the instructions set forth therein.

• RSA Data Security, Inc. MD5 Copyright © 1991-2 RSA Data Security, Inc. Created 1991. All rights reserved. License to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing this software or this function. License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing the derived work. RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind. These notices must be retained in any copies of any part of this documentation and/or software.

• Sun RPC v3.9 - Sun RPC is a product of Sun Microsystems, Inc. and is provided for unrestricted use provided that this legend is included on all tape media and as a part of the software program in whole or part. Users may copy or modify Sun RPC without charge, but are not authorized to license or distribute it to anyone else except as part of a product or program developed by the user. SUN RPC IS PROVIDED AS IS WITH NO WARRANTIES OF ANY KIND INCLUDING THE
WARRANTIES OF DESIGN, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE. Sun RPC is provided with no support and without any obligation on the part of Sun Microsystems, Inc. to assist in its use, correction, modification or enhancement. SUN MICROSYSTEMS, INC. SHALL HAVE NO LIABILITY WITH RESPECT TO THE INFRINGEMENT OF COPYRIGHTS, TRADE SECRETS OR ANY PATENTS BY SUN RPC OR ANY PART THEREOF. In no event will Sun Microsystems, Inc. be liable for any lost revenue or profits or other special, indirect and consequential damages, even if Sun has been advised of the possibility of such damages. Sun Microsystems, Inc., 2550 Garcia Avenue, Mountain View, California 94043.

- Tanuki Software Java Service Wrapper. See the Tanuki Software, Inc. Development Software License Agreement, Version 1.0 in the installation directory in the docs/ThirdPartyLicenses folder for license information. This product includes software and documentation components developed in part by Silver Egg Technology, Inc. ("SET") prior to 2001. All SET components were released under the following license. Copyright © 2001 Silver Egg Technology. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sub-license, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

- Yahoo! User Interface Library - V 0.12.1. Copyright © 2006 Yahoo! Inc. All rights reserved. The name Yahoo! Inc. nor the names of its contributors may be used to endorse or promote products derived from this Software and products derived from this software without specific prior written permission of Yahoo! Inc. See the Yahoo! User Interface Library license V 0.12.1 in the installation directory in the docs/ThirdPartyLicenses folder for license information.
Chapter 1
Overview

This chapter describes ObjectStore Inspector and some of its main features, and tells you how to get started.

This chapter covers the following topics:

Introducing ObjectStore Inspector 18
Inspector Features 21
Use Inspector to Create Multiple Views of the Same Database 24
Sharing ObjectStore Database Information 26
Inspector Options 27
Introducing ObjectStore Inspector

The paragraphs that follow provide an overview of the Inspector software, and insight into the benefits of using Inspector to examine your ObjectStore databases.

What Is Inspector?

ObjectStore Inspector is a graphical tool that lets you browse, edit, query, and report on the data in an ObjectStore database.

Using Inspector, you can

- Analyze and document the logical information represented by ObjectStore database schemas. Inspector’s data views let you filter and order ObjectStore collections, and you can use database views to create custom schema diagrams. You can navigate instance relationships, and save the resulting navigation path for future use.
- Create, read, update, and delete ObjectStore application data in test and production environments — you can work with ObjectStore data directly from Inspector using manual editing tools and user-defined methods.
- Analyze the physical characteristics of ObjectStore databases at the segment, page, and instance level. The Physical Database Layout window provides detailed space and free space information and enables you to look up a particular instance based on its address or segment offset. You can also review layout, segment, and binary dump information for individual instances.
- Share ObjectStore database information by creating reports based on ObjectStore application data, printing schema diagrams and data views, and exporting collections to other applications. You can also use Inspector as an OLE server in documents from OLE container applications.
Opening ObjectStore Databases

You can open one ObjectStore database at a time using Inspector. If you want, you can start multiple Inspector sessions and use those sessions to open the same database.

Supported formats

Inspector supports ObjectStore databases stored in any of the following formats:
- File
- Remote
- Rawfs (raw file storage)

For more information on storing ObjectStore databases, see the ObjectStore C++ API User Guide.

Read/write access

Inspector opens ObjectStore databases in MVCC (Multiversion Concurrency Control) mode — this enables Inspector to browse databases that are concurrently accessed and updated by other ObjectStore processes.

Some Inspector operations, however, require that Inspector open a database in write mode. These operations include:
- Invoking a create or update user-defined method
- Saving metaknowledge when the metaknowledge is stored with the database
- Editing a data member from within Inspector, using the Edit Data Member dialog box

In these cases, Inspector opens the database in write mode only for as long as is required to complete the operation. If Inspector needs to access a remote database to perform one of these operations, you must ensure that the remote ObjectStore server grants Inspector write access.
How to Start Inspector

To start ObjectStore Inspector, select Programs -> ObjectStore -> ObjectStore Inspector from the Start menu.

The Inspector desktop appears.

The Inspector desktop is similar to those of other Windows applications — a title bar identifies the current database, a menu bar displays Inspector commands. An optional toolbar, a workspace, and a status bar that provides tool and menu prompts complete the desktop.

Open the carsdemo Database

Consider opening carsdemo.db, or one of the other sample databases installed with Inspector, before continuing with this guide. Having a database open will make it easier for you to explore Inspector as you learn about it.

Tip: The carsdemo.db database used in this manual, and other sample databases, are located in the <install>\examples\DemoDBs directory, where <install> is the directory in which you have installed ObjectStore Inspector.

To open the carsdemo database:

1. Click File -> Open Database on the menu bar.
   
   Alternative: Click the Open DB tool on the toolbar.

   The Open dialog box appears.

2. Select the carsdemo sample database in the <install>\examples\DemoDBs directory and click OK.

   The main database view window appears within the Inspector desktop.

See the next section, Inspector Features on page 21, to learn more about the database view window and other Inspector tools.
Inspector Features

This section introduces the main database view and other tools Inspector provides to help you view and modify ObjectStore information.

Main Database View Window

The main database view window appears in the Inspector desktop by default when you open a database.

Three panes

The main database view is composed of three panes, each of which provides you with a different view of information contained in the database. It is through these panes, and other windows and dialog boxes, that you interact with Inspector and the ObjectStore database.

- The Database Roots pane — Displays the roots that exist for the database. You can use Inspector to create roots. See Chapter 8, Roots, on page 115 for more information.
- The schema pane — Displays a schema diagram representing the ObjectStore database using one of several popular modeling notations. The schema pane is located below the Database Roots pane. See Chapter 4, Schema Diagrams, on page 55 for more information.
- The instance pane — Displays instance information for the selected root or class. The instance pane is empty until you fill it with a class or collection extent. By default, instances are displayed in a grid format, though a list format is available. The instance pane is located to the right of the Database Roots pane. See Chapter 6, Classes and Instances, on page 73 for more information.

You can create multiple database views

You can create custom database views based on the contents of the main database view. See Chapter 2, Database Views, on page 29 for more information.
Data Views

A data view is similar to the instance pane in a database view: both display ObjectStore collections in either a grid or list format.

Data views provide unique *filtering* and *ordering* features that help you customize the presentation of ObjectStore collections. Any time you want to manipulate the data displayed by an ObjectStore collection, you should use data views to take advantage of these features.

For more information on working with data views, see Chapter 3, Data Views, on page 41.

Instance Window

The Instance window displays the type, name, and value for every data member associated with a specific class instance.

*Tip:* You control the appearance of an instance — that is, which of its data members are displayed and whether or not it is associated with an icon — using the Instance Format dialog box.
See Chapter 6, Classes and Instances, on page 73 to learn more about working with instances.

**Navigation Window**

The Navigation window lets you see how instances are related to one another.

Inspector opens a Navigation window for you as you navigate from instance to instance — in the same database or across different databases. As you navigate, Inspector updates the Navigation window to record your path.

As an alternative, you can ask Inspector to navigate all relationship paths of an instance for you, using the Auto Navigation feature.

See Navigating Instances on page 85 to learn more about navigation.

**Physical Database Layout Window**

The Physical Database Layout window displays physical information about the ObjectStore database at the segment and page level.

For any group of segments and pages you select, Inspector displays the classes stored in that part of ObjectStore’s persistent memory. Once you select a class, Inspector displays the address and segment offset information for each instance in the class.

See Chapter 9, Tools for Physical Analysis, on page 121 for more information on the Physical Database Layout window and other features for working with physical aspects of ObjectStore databases.
Use Inspector to Create Multiple Views of the Same Database

You can use the tools you have just read about to create multiple views of an ObjectStore database. For example, for the same database you can create

- A database view that hides the inheritance relationships of a particular class
- A database view that displays the schema diagram using a different modeling notation
- A data view that contains an ObjectStore collection filtered using certain criteria
- Class-specific instance formats
- Data-view-specific instance formats

All this information, and other customization you perform in Inspector, can be saved with the database — the next time you open that database in Inspector, every custom database view, data view, instance format, and so on, is available to you.

How information is saved

Instance format information is saved implicitly when it is defined. You save custom database views and data views explicitly, either at the time you create them, or on closing the database (or exiting from Inspector). For example, if you create a data view but have not saved it when you close the database, Inspector displays a prompt giving you the chance to save it.

Metaknowledge

Metaknowledge is information Inspector stores about a particular database. Metaknowledge includes any custom database views and data views you have defined, as well as other types of information about the database. See Appendix C, Working with Metaknowledge, on page 145 for more information.

Editing ObjectStore Database Information

You can use Inspector to modify ObjectStore data.

User-Defined Methods

A user-defined method is a method that can be called by Inspector to

- Create persistent objects
- Read persistent objects
- Update persistent objects
- Delete persistent objects

In addition you can use a user-defined method to retrieve sets of persistent objects. You define user-defined methods in the ObjectStore DLL schema of your ObjectStore database. Once it is defined, you register and invoke the method in Inspector.

For more information, see Chapter 7, User-Defined Methods, on page 103.
Edit Data Member Dialog Box

You use the Edit Data Member dialog box to edit instances of classes you select using Inspector.

For more information, see Chapter 6, Classes and Instances, on page 73.
Sharing ObjectStore Database Information

Inspector has tools to help you present and share information about an ObjectStore database.

Printing

Using Inspector, you can

• Print database schema using several popular notations
• Print data views
• Print navigation trees

See Appendix A, Printing, on page 133 for more information.

Exporting

You can use Inspector to export ObjectStore collection data to files in several popular formats, including XML, Excel, and HTML.

See Exporting a Collection Grid on page 72 for more information.

Using Inspector as an OLE Server

You can also use the database schema or the instance pane as an OLE server, embedding live database information in any OLE container application, such as Microsoft Word or Excel.

See Appendix B, Using Inspector as an OLE Server, on page 141 for more information.
Inspector Options

Inspector has options that control both functional behavior and display characteristics. For example, there are options that let you

- Control the default notation used in schema diagrams
- Control whether or not external databases are opened to resolve cross-database navigation
- Tailor Inspector’s grid-loading performance
- Select the font you want to use for the display of instance information

Tip: Consider leaving Inspector options set with their default values until you have gained more experience using Inspector.

Where You Set Options

You set options using the Options dialog box.

The Options dialog box contains several tabbed pages; options are grouped by feature.

How to Set Options

To set options:

1. Click Tools -> Options on the menu bar.
   The Options dialog box appears.
2. Click the tab for the options you want to set.
3. Make your changes.
4. Click OK.
When Options Take Effect

Options can take effect

- Immediately
- The next time the database is opened
- When a database is opened without the metaknowledge saved with it previously, that is, when the metaknowledge for a database is recomputed
Chapter 2
Database Views

This chapter describes how to create and work with database views. A database view is a tool that helps you work with the logical information about an ObjectStore database.

For more information: To learn about using Inspector to manage physical aspects of an ObjectStore database, see Chapter 9, Tools for Physical Analysis, on page 121.

This chapter covers the following topics:

Overview 30
The Database View Window 32
Creating a Custom Database View 36
Overview

What Is a Database View?

A database view is a window that displays information about an ObjectStore database. It is called a view because it is just that—changes made in a database view do not affect the database itself.

Three panes

The database view window contains three panes, each of which displays different information about the database:

- Database Roots pane
- Schema pane
- Instance pane

For more information on the panes in a database view, see The Database View Window on page 32.

Database Views Contrasted to Data Views

A database view displays an ObjectStore database using a Database Root pane, a schema pane, and an instance pane.

A data view, on the other hand, simply displays an ObjectStore collection. The data view window is similar to the instance pane of the database view window in that both can display an ObjectStore collection using either a grid or a list. A data view also allows you to order and filter the ObjectStore collections it displays.

For more information about working with data views, see Chapter 3, Data Views, on page 41.
Custom Database Views

Inspector supports two types of database views:

• **Main database view** — The main database view appears by default each time you open a database in Inspector. It always displays all the information about an ObjectStore database. You cannot change the content of the main database view without changing the database itself.

• **Custom database views** — Custom database views are those you create by altering some element of the schema diagram in the main database view. For example, you can create a custom database view by collapsing a class hierarchy. You can create a custom view *explicitly*, using the menu bar, or *implicitly*, when you make certain types of changes to the main database view.

See Creating a Custom Database View on page 36 for more information.
The Database View Window

Window Name
The name of the window depends on whether the window contains the main database view or a custom database view.

- The title bar of the main database view displays the name of the database.
- The title bar of a custom database view displays DB View: Untitled until you save and name the database view. Thereafter, it displays DB View: <name>.

The Toolbar
The toolbar for a database view window consists of four groups of tools.

- Standard

The standard group contains tools to help you work with databases, database views, and schema diagrams.

- Schema

The schema group contains tools to help you work with schema diagrams.

- Instance

The instance group contains tools to help you format instances in Inspector, edit instances in ObjectStore, and display the instance pane using a collection grid or a collection list.
• Navigation

The navigation group contains tools to help you work with Navigation and Instance windows.

Hiding toolbar groups

By default, all four tool groups are displayed. You can hide any of the toolbar groups.

To hide a toolbar group:

1. Click Tools | Toolbar Settings on the menu bar.

   The Toolbar Settings dialog box appears.

2. Select the check boxes of the tool groups you want to hide.

3. Click OK.

   The toolbar is redisplayed, showing only the groups you have chosen to display.

Tip: Consider displaying all the toolbar groups until you become more familiar with Inspector.

The Database Roots Pane

The Database Roots pane displays all the roots that have been defined for the current database (or all the roots that exist within a particular database view).

The Database Roots pane displays the root name and address, and the root value type and address. Double-click any value in the Database Roots pane to populate the instance pane with instances for that root.

For more information on working with roots, see Chapter 8, Roots, on page 115.
The Schema Pane

The schema pane displays the database schema (or database schema view) using one of several popular modeling notations.

Available notations include

- Tree hierarchy
- UML (uniform modeling language)
- Booch
- OMT (object modeling technique)
- Coad-Yourdon
- Statecharts

The default notation is the tree hierarchy. You can change the default notation, or change the notation on the fly using the toolbar, menu, or schema shortcut menu. The shortcut menu also includes a Find Class choice, to help you quickly locate a specific class within the schema.

Changing schema diagram layout

You can change the layout of schema diagrams. You can

- Rearrange classes
- Reshape relationship lines

Changing schema diagram contents

You can alter the contents of schema diagrams. For classes you select, you can

- Remove the inheritance tree
- Collapse the inheritance tree
- Remove the super class
- Remove relationships

When you alter the default schema diagram in any of these ways, Inspector automatically creates a new, untitled database view that reflects these changes — you cannot alter the content of the default schema diagram.
For more information
See Chapter 4, Schema Diagrams, on page 55 for more information on working with diagrams.

The Instance Pane

The instance pane displays the extent of instances for a specific class or root.

The instance pane and the data view are the primary ways you interact with ObjectStore collections in Inspector.

When you first open a database, the instance pane is empty. You can populate the pane a number of ways — the easiest is to double-click a class in the schema pane or a root in the Database Roots pane.

By default, instances appear in a collection grid. You can also display instance information in a simple list if you prefer.

For more information on working with instances, see Chapter 6, Classes and Instances, on page 73. Also, see Chapter 5, Collection Grids and Lists, on page 59.

Changing Pane Dimensions

You can change the dimensions of the panes in a database view by dragging the borders that separate them.

To change pane dimensions:

1. Place the pointer on the border of the pane you want to resize.
   The pointer changes shape when it is placed on the pane border.

2. When the pointer changes shape, drag the border to change the size of the pane.
Creating a Custom Database View

This section describes how to create and work with custom database views. Once you have created a custom database view, see Modifying a Database View on page 39 to learn more about how to change a database view’s appearance.

Two Ways to Create

There are two ways to create a custom database view:

- Explicitly — You create a custom database view explicitly using menu commands.
- Implicitly — You create a custom database view implicitly by applying an abstraction function to the schema in the main database view.

The database views that result from both of these operations are considered custom database views. Remember that changes to database views do not affect the ObjectStore database itself.

Creating a Custom Database View Explicitly

When you create a database view explicitly, Inspector creates an exact copy of the currently active database view, whether it is the main database view or a custom database view. This database view is unnamed (actually, it is called Untitled n, where n is some number to make it unique); you give it a name when you save it.

To create a custom database view explicitly:

1. Select the database view that you want to use as the foundation for the custom database view you want to create.
   
   Note: The foundation database view can be either the main database view or another custom database view.

2. Click File | Database View | Create on the menu bar.

   Alternative: Click the Create Database View tool on the toolbar.

   Inspector creates a new database view window containing the custom database view.

Creating a Custom Database View Implicitly

Inspector creates a custom database view implicitly any time you apply an abstraction function to the schema diagram in the main database view. Applying an abstraction function to a custom database view simply further modifies that view; a new database view is not created.

An abstraction function is an operation that permanently alters the composition of a database schema diagram. Abstraction functions do not affect the database or the database schema, simply the way a database is represented in a schema diagram.
You might want to use an abstraction function to filter out classes belonging to a third-party library, for example, in order to better concentrate on the classes related to the ObjectStore database.

To learn more about abstraction functions, see Changing the Appearance of the Schema Diagram on page 56.

To create a custom database view implicitly:

1. Select the database view that you want to use as the foundation for the custom database view you want to create.
   
   Note: Applying abstraction functions to a custom database view does not create a new database view.

2. Select the class whose appearance in the diagram you want to alter.

3. Click Schema | Alter on the main menu.
   
   A drop-down menu appears.

4. Select the abstraction function you want to apply from the drop-down menu.
   
   Inspector creates a new database view window containing the custom database view.

Saving a Custom Database View

To save a custom database view:

1. Click File | Save on the main menu.
   
   Alternative: Click the Save tool on the toolbar.

   The Save Database View dialog box appears.

2. Type a name in the Database View Name field.

3. Click OK.

You do not explicitly save the main database view. If you make any changes to the main database view — changing the layout of the schema diagram or defining a new view instance format, for example — Inspector displays a prompt asking if you want to save those changes when you close the database in Inspector.
Opening a Custom Database View

You can display multiple custom database views for the current database. You might want to do this if you have defined views using different schema notation and abstraction functions, for example.

How to open a custom database view

To open a custom database view:

1. Click File | Database View | Open on the menu bar.
   
   *Alternative:* Click the Open Database View tool on the toolbar.

   The Open a Database View dialog box appears.

   *Note:* The Open a Database View dialog box is similar to the Save Database View dialog box.

2. Select the database view you want to open from the list box and click OK.
   
   *Alternative:* Double-click the database view name.

   The custom database view window appears.

Deleting a Custom Database View

You can delete a custom database view at any time. Consider deleting obsolete custom database views to eliminate unnecessary metaknowledge associated with the database.

How to delete a custom database view

To delete a custom database view:

1. Click File | Database View | Delete on the menu bar.

   The Delete a Database View dialog box appears.

   *Note:* The Delete a Database View dialog box is similar to the Save Database View dialog box.

2. Select the database view you want to delete from the list box and click Delete.
   
   *Alternative:* Double-click the database view name.

   Inspector displays a dialog box asking you to confirm the delete operation.

3. Click Yes to delete the database view; click No to cancel the delete operation.
Modifying a Database View

You can modify database views in several ways. This section describes those ways and tells you where to find more information.

Schema Diagram Layout

You can change the layout of the schema diagram in a database view in a number of ways. For example, you can change diagram notation, move classes, reshape relationship routes, and hide information. These types of changes affect only the appearance of the schema diagram itself; they do not cause a custom database view to be created.

See Chapter 4, Schema Diagrams, on page 55 for more information about working with schema diagrams.

Abstraction Functions

As described earlier in this chapter, abstraction functions enable you to permanently alter the composition of a schema diagram. Applying an abstraction function to the schema diagram in the main database view always results in the creation of a custom database view.

See Altering the Contents of a Schema Diagram on page 58 for more information on abstraction functions.

Instance Format

Inspector provides several features that let you customize the appearance of instances. Changes you make in the Instance Format dialog box affect the instance appearance everywhere it is displayed — custom database views, data views, and so on.

See Customizing the Instance Display on page 79 for more information.
Chapter 3
Data Views

This chapter describes how to work with ObjectStore collections using data views.

For more information: You can also work with ObjectStore collections using the instance pane of the main database view window. See Chapter 6, Classes and Instances, on page 73.

This chapter covers the following topics:

- Working With Data Views 42
- Filtering a Collection 45
- Ordering a Collection 53
What Is a Data View?

A data view is an Inspector window that displays the instances associated with an ObjectStore collection. A data view is unique to, and saved with, the ObjectStore database on which it is based.

A Data View Contrasted to the Instance Pane

Think of a data view as an instance pane with its own window.

A data view is similar to the instance pane in that you can choose to display information using a collection grid or a collection list. Only data views, however, allow you to manipulate the ObjectStore collection data itself. Consider creating a data view any time you want to

- Filter ObjectStore collection instances. Filters are described in Filtering a Collection on page 45.
- Control the order of ObjectStore collection instances. Ordering features are described in Ordering a Collection on page 53.

Tip: Remember that you control which data members are displayed for a given instance using the Instance Format dialog box. For more information, see Customizing the Instance Display on page 79.

Working With Data Views

This section describes how to create, save, open, and delete a data view.

Creating a Data View

When to create

You can create a data view any time you have an ObjectStore database open in Inspector. You must create a data view if you want to filter or order an ObjectStore collection.
How to create

To create a data view:

1. Display the main database view window.
2. Populate the instance pane with an ObjectStore collection.
   If you need help with this step, see Populating Collection Grids and Lists on page 61.
3. Click Data View | Create on the menu bar.
   Alternative: Select Create Data View from the instance pane shortcut menu.

A data view window appears. The title bar displays the
- Collection address
- Number of elements in the collection
- The class associated with the collection

Tip: When you save a data view, you can give it a name. The name you assign replaces the address; element and class information is retained in the title bar.

4. At this point, you can either save the data view, or begin customizing its layout or contents as described in the remaining sections of this chapter.

Creating a data view based on an array

If a data member in a class consists of an array, you can create a data view based on that array.

To create a data view of this type:

1. Display the Instance Window for the class that contains the data member.
2. Right click on the data member. A shortcut menu appears.
3. Select Create Data View from the shortcut menu.

A data view window appears, listing the data in the array.

Creating a data view based on a user-defined method

If a class includes a user-defined method to retrieve a set of objects (an extent method), you can create a data view based on the set of objects retrieved by the method.

To create a data view of this type:

1. Right click on the class in the Schema Pane.
2. Select User-defined Methods | Extent from the shortcut menu and then select the user-defined method.

A data view window appears, listing the data retrieved by the user-defined extent method.

In order to use a user-defined method in this way, it must be registered. For more information about defining and registering user-defined methods, see Chapter 7, User-Defined Methods, on page 103.
Saving a Data View

When to save
You can save a data view once it has been created. You might want to save a data view right after you create it — instead of waiting until you have finished format and data manipulation. Giving it a more meaningful name can help you stay focused on the data view’s purpose while you work with it.

How to save
To save a data view:

1. Click File | Save on the menu bar.

   *Alternative:* Click the Save tool on the data view window’s toolbar.

   The Save Data View dialog box appears.

2. Type a name in the Data View Name field and click OK.

   *Tip:* Give the data view a name that will help you remember its characteristics should you want to open it at a later time.

   The data view is saved with the name you give it. It is available any time you open the current database in Inspector.

Opening a Data View

You can open a data view only if it was saved with the ObjectStore database you are viewing in Inspector.

To open a data view:

1. Click Data View | Open on the menu bar.

   The Open Data View dialog box appears.

   *Tip:* The Open Data View dialog box is similar to the Save Data View dialog box.

2. Select the data view you want to open from the list box and click OK to open it.

   *Alternative:* Double-click the data view.

   The data view window appears.
Deleting a Data View

To delete a data view:

1. Click Data View | Delete on the menu bar.
The Delete Data View dialog box appears.

*Tip:* The Delete Data View dialog box is similar to the Save Data View dialog box.

2. Select the data view you want to delete from the list box and click the Delete button to delete it.

*Alternative:* Double-click the data view.
The data view window is deleted.

Filtering a Collection

This section describes what filters are and how they work for you.

What Is a Filter?

A *filter* is a constraint that you define for a data member value of an ObjectStore collection’s instances. It behaves the same way, and has the same purpose, as an ObjectStore query — to mine data from an ObjectStore collection.

Each data view can be associated with only one filter. Similarly, you cannot apply a filter to a data view other than the one for which it was created.

A filter can contain both static and parameterized constraints. A *static* constraint is one whose value does not change; a *parameterized* constraint is one that uses a value that must be supplied each time the filter is applied to the ObjectStore collection.

The parameter value is always provided by the user at the time the filter is applied. Depending on how you define the constraint, the user can either

- Enter his or her own value
- Select from a list of values — either collection values for that data member or values defined for the constraint

Regardless of which method you use, Inspector displays the Fill Query Parameters dialog box when the filter is applied to the ObjectStore collection.

See Defining Parameterized Constraints on page 50 for more information.
Two Ways to Define a Filter

Inspector provides two ways to define a filter for an ObjectStore collection.

- You can define the filter manually, using the Set ObjectStore Query Expression dialog box. This method requires command of the ObjectStore query syntax and should be considered only by advanced users of ObjectStore. See Defining a Filter Manually on page 47 for more information.

  *Note:* Once you define a filter manually, you cannot use the Data Member Constraints dialog box to build a filter for that data view.

- You can use the Data Member Constraints dialog box. This tool allows you to build the filter query expression using Inspector’s graphic interface.

You construct the query constraint by constraint, identifying the data members, operators, and values you want the query to include. See Defining a Filter Using Inspector on page 48.

Parameterized constraints are also defined using a graphic interface — one dialog box to help you construct the constraint, and another that appears at run time so you can provide the parameter value.

The filters you define using the Data Member Constraints dialog box are more versatile than those you can define manually, enabling you to do the following without any knowledge of ObjectStore query syntax. You can

- Use a regular expression parser
- Run parametric queries
- Query data members of subclasses
- Navigate Java references and collections
This section describes how to work with the Set ObjectStore Query Expression and Data Member Constraints dialog boxes. See the Advanced C++ API User Guide for information on ObjectStore queries.

When Are Filters Applied?

Inspector applies a filter to an ObjectStore collection at the time you create the filter, regardless of how you define it. This gives you the opportunity to verify that the filter is performing as required. If it is not, you can reopen it and then modify it.

How to turn filters on and off

Once a filter has been defined, you can turn it on and off whenever you want.

To turn a filter on and off, click Data View | Filter On on the menu bar, or click the Toggle Filter tool on the data view window toolbar.

Tip: A filter is on when a check mark appears alongside Filter On on the Data View menu, or when the Toggle Filter tool is selected.

How to reapply a filter

You can reapply a filter at any time. In particular, you might want to consider reapplying a filter to an ObjectStore collection

- If you know an application has changed instances in the ObjectStore database
- If the filter contains parameterized constraints and you want to run the filter with new constraint values

To reapply a filter, click Data View | Reapply Filter on the menu bar, or click the Reapply Filter tool on the data view window toolbar.

Defining a Filter Manually

You use the Set ObjectStore Query Expression dialog box to manually define collection filters.

If you are not familiar with ObjectStore query syntax, use Inspector to define the filter. See Defining a Filter Using Inspector on page 48.

How to use

To manually define a filter:

1. Click Data View | Set Filter Expression on the menu bar.
2. Enter the query expression — you must use ObjectStore query syntax.
3. Click OK to create the filter.
Defining a Filter Using Inspector

You use the Data Member Constraints dialog box when you want Inspector to help you define a filter.

![Data Member Constraints dialog box](image)

How to open
To open the Data Member Constraints dialog box, click Data View | Define on the menu bar, or click the Define Filter tool on the toolbar.

Usage tips
The Data Member Constraints dialog box has several features that simplify the process of defining a filter.

- The Data Member list box displays the data members of the collection’s instances. You can also include data members of navigable classes.
- The Operator drop-down list box contains a list of standard query operators: greater than (>) , not equal to (!=), and so on.
- The Data Member field lets you define a static constraint value. Other options appear below this field based on the type of the data member you select.

*Note:* The Data Member list box can display data members of classes that participate in recognized relationships only. A recognized relationship is one that is either

- Defined as part of the database,
- Created by Inspector

For more information on how Inspector works with ObjectStore relationships, see Identifying Relationships Between Classes on page 91.
Displaying the filter query

The filters you define using Inspector are translated into ObjectStore query language when the filter is created. You can display translated filter queries in the Show Filter Expression dialog box.

To display a translated filter query, click Data View | Show Filter Expression on the menu bar.

Tip: If you are defining filters manually, you can use the Copy button to copy query elements to the clipboard for pasting.

Definition process

You define a filter with the Data Member Constraints dialog box, using the following process:

1. Open the Data Member Constraints dialog box.
2. Select the first data member to which you want to apply a constraint.
3. Select the constraint operator and data member value. If you want to define a parameterized constraint, click the Define Parameterized Constraint button.
   
   See Defining Parameterized Constraints on page 50 for more information.
4. When you have defined the first constraint, click the Add button to place the constraint in the Data Member Constraints and Logical Operators list box.
5. Repeat steps 2 through 4 for the other constraints you want to include in the filter definition.
6. Use the And, Or, and Not buttons to apply these expressions to the selected condition. You can change the order of a constraint by dragging it to a new position in the list box.
7. Click OK to create the filter.
Defining Parameterized Constraints

You define a parameterized constraint using the Parameter Definition dialog box.

Note that the fields in this dialog box vary based on whether you are defining a parameterized constraint for string or integer types. The dialog box shown here is for string types.

To define a parameterized constraint for string types:

1. Follow the procedure for defining a filter as described in Definition process on page 49.
2. At step 3 of that process, select the data member with which you want to associate the parameterized constraint from the Data Member list box and click the Define Parameterized Constraint button.

   The Parameter Definition dialog box appears.

3. Enter text in the User Prompt field for the prompt you want to appear in the Description field of the Fill Query Parameters dialog box.
4. Select the comparison value you want to use for this data member.
5. In the Control group box, select the type of control for the Value field of the Fill Query Parameters dialog box:
   - List box
   - Drop-down list box
   - Edit box

   Tip: If you use the edit box control, the user must provide his or her own value at run time.
6. Optionally, name the parameter.
7 Are you using an edit box control for the Value field of the Fill Query Parameter dialog box?
   If yes, you are done defining the parameterized constraint. Go to step 13.
   If no, go to step 8.

8 Click the Multiple Selection and Sort check boxes if you want to enable this functionality in the Value field.

9 Do you want the user to select from a list of actual data members to provide the parameter value?
   If yes, click Get Values From Query and go to step 13.
   If no, go to step 10.

10 Click Enter Values By Hand and go to step 11.

11 Define the list of parameter values from which you want the user to be able to select:
   a Type a name in the New Value field.
   b Click Add to place the value in the Defined Values list box.
   Tip: Click Remove to remove a value from the Defined Values list box.

12 Select the Always Recompute Values check box if you want Inspector to refresh the collection each time the filter is applied.

13 Click OK to return to the Data Member Constraints dialog box.

14 Go to step 2 if you want to define a parameterized constraint on another data member in the filter. Otherwise, click OK to finish defining the filter.

   The filter is applied to the collection. Inspector displays the Fill Query Parameters dialog box. See Using the Fill Query Parameters dialog box on page 53.
How to define a parameterized constraint for integer types

To define a parameterized constraint for integer types:

1. Follow the procedure for defining a filter as described in Definition process on page 49.

2. At step 3 of that process, select the data member with which you want to associate the parameterized constraint from the Data Member list box and click the Define Parameterized Constraint button.

The Parameter Definition dialog box appears.

3. Enter text in the User Prompt field for the prompt you want to appear in the Description field of the Fill Query Parameters dialog box.

4. Select the operator you want to use for this data member.

5. Select the type of control for the Value field of the Fill Query Parameters dialog box from the Control drop-down list box:
   - Drop-down list box
   - Edit box
   
   Tip: If you use the edit box control, the user must provide his or her own value at run time.

6. Optionally, name the parameter.

7. Are you using an edit box control for the Value field of the Fill Query Parameters dialog box?
   - If yes, you are done defining the parameterized constraint. Go to step 10.
   - If no, go to step 8.

8. Define the list of parameter values from which you want the user to be able to select:
   - Type a name in the New Value field.
   - Click Add to place the value in the Defined Values list box.
   
   Tip: Click Remove to remove a value from the Defined Values list box.

9. Select the Always Recompute Values check box if you want Inspector to refresh the collection each time the filter is applied.
10 Click **OK** to return to the Data Member Constraints dialog box.

11 Go to step 2 if you want to define a parameterized constraint on another data member in the filter. Otherwise, click **OK** to finish defining the filter.

The filter is applied to the collection. Inspector displays the Fill Query Parameters dialog box. See Using the Fill Query Parameters dialog box on page 53.

If you defined a parameterized constraint for one of the data members included in the filter, Inspector displays the Fill Query Parameters dialog box when the filter is applied to the collection.

To complete the dialog box:

1 Double-click the Value field.

   The Value field becomes editable, displaying the type of control you defined for it.

2 If the control is a type of list, select the values from the list. Otherwise, type a value in the field.

3 If necessary, go to step 1 for any other parameterized constraints that are defined for this filter.

4 Click **OK** to close the dialog box and complete the filter.

   Inspector redisplayes the collection within the data view. The values displayed are based on the filter just applied to the collection.

5 To redisplay the entire collection, turn off the filter.

### Ordering a Collection

In addition to filtering an ObjectStore collection, you can order an ObjectStore collection based on the values of one of the data member’s instances. You use the Sort Order dialog box to define a collection order.
Selecting Data Members

Only data members with either an integer or string type can be used to order an ObjectStore collection. In addition, the Data Member list box can display data members of classes that participate in recognized relationships only. A recognized relationship is one that is either

- Defined as part of the database, or
- Created by Inspector

For more information on how Inspector works with ObjectStore relationships, see Identifying Relationships Between Classes on page 91.

When Does the Order Take Effect?

A collection order takes effect at the time you define the order. This gives you the opportunity to verify that the order sorts the collection as you want it to. If it does not, you can delete it, or open it and modify it.

How to turn orders on and off

Once a collection order has been defined, you can turn it on and off whenever you want.

To turn a collection order on and off, click Data View | Collection On on the menu bar, or click the Toggle Order tool on the data view window toolbar.

Tip: A collection order is on when a check mark appears alongside Order On on the Data View menu, or when the Toggle Order tool is selected.

How to Define a Collection Order

To define a collection order:

1. Click Data View | Define Order on the menu bar.
   The Sort Order dialog box appears
2. Select the data member on whose instances you want to order the collection.
3. Click the Ascending or Descending radio button to determine the collection order.
4. Click OK.
   The collection is redisplayed based on the order you just defined.

Note: The collection’s original order is restored when you turn the order off.
Chapter 4
Schema Diagrams

This chapter describes how to work with the schema diagram.

This chapter covers the following topics:
Changing Diagram Notation 55
Changing the Appearance of the Schema Diagram 56

Changing Diagram Notation

Inspector supports several popular diagram notations. This section identifies these notations, and tells you how to change them and specify the notation you prefer to use as your default.

Supported Notation

Inspector supports the following diagram notation styles:

- Tree hierarchy
- UML
- Booch
- OMT
- Coad-Yourdon
- Statecharts

The default notation style is the tree hierarchy. This style is more scalable than the others and can provide better performance when working with large databases.

How to Change Diagram Notation

To change the diagram notation:

1 Select Schema | Notation from the menu bar.

   Alternative: Select Notation from the schema diagram shortcut menu.

2 Select the style you want from the menu.

Saving the change

If you want to save the database view with the schema diagram notation you have selected, you must save the database.
Changing the Default Notation

The default schema diagram notation is specified on the Standard page of the Options dialog box. Changing the default notation affects the notation the first time you open a database. After that, the schema diagram is displayed with whatever notation you have saved it with.

To change the default schema diagram notation:

1. Select Tools | Options from the menu bar to open the Options dialog box.
2. Display the Standard page if it is not already displayed.
3. Select the notation style you prefer to use as your default.
4. Click OK.

Changing the Appearance of the Schema Diagram

This section describes the ways you can change the appearance of the schema diagram. You might want to consider changing the layout prior to printing a schema diagram.

Ways to Change the Diagram Appearance

There are several ways you can change the appearance of the schema diagram. You can

- Change the diagram zoom level
- Change the layout of classes
- Change the shape of relationship routes
- Hide relationships
- Alter the composition of the schema diagram using abstraction functions

Changing the Diagram Zoom Level

To change the diagram zoom level, select a value from the Zoom tool drop-down list, or type a value.

Alternative 1: Select Zoom In or Zoom Out from the schema diagram shortcut menu.

Alternative 2: Press Ctrl + to zoom in, press Ctrl - to zoom out.

Changing Class Layout

To change the class layout, drag the class to the location you want.

Tip: Decrease the zoom level to see more of the diagram.
Changing the Shape of Relationship Routes

Each relationship route has four points: two endpoints and two attraction points. All points become visible when you click the route where it meets a class, as shown here in the relationship between the classes `QuPart` and `ServiceItem`.

You change the shape of the relationship route by dragging one or both attraction points to a new location. You cannot change the placement of the route endpoints.

How to change a relationship shape

To change the shape of a relationship route:

1. Click the route you want to reshape where it meets a class.
   
   The route changes color to show it is selected; the four route points become visible.

2. Drag the attraction points until the curve has the shape you desire.

3. Click anywhere on the diagram background to stop the reroute mode.

Hiding Relationships

You can quickly simplify a schema diagram by hiding all relationships of a given type. Relationships hidden as described in this section are easily redisplayed.

Tip: You can remove individual relationships from a schema diagram using abstraction functions. To learn more about this feature, and about abstraction functions in general, see Altering the Contents of a Schema Diagram on page 58.

Types of relationships

You can hide the following types of relationships:

- All inheritance relationships
- Private inheritance relationships
- ObjectStore relationships
- One-way relationships

These features hide all relationships of the type you select.

Hide all relationships

To hide all relationships of a given type, select the type of relationship from the Schema | Hide menu.

Alternative: Use the schema diagram shortcut menu.

Redisplay hidden relationships

To redisplay a hidden relationship, repeat the procedure you used to hide it.
Another way to simplify the appearance of the schema diagram is to alter its contents using abstraction functions. Using abstraction functions is similar to hiding relationships, but abstraction functions differ in a few important ways:

- Abstraction functions permanently affect the composition of the schema diagram. Because of this, Inspector automatically creates a new schema diagram whenever you apply an abstraction function.
  
  *Note:* Abstraction functions do not affect the database or the database schema, simply the way the database is represented in a schema diagram.

- You cannot undo or reverse an abstraction function.

- Most abstraction functions are class-based and not relationship-based.

### Types of abstraction functions

The following abstraction functions are available on individual classes in a schema diagram. You can

- Remove a class’s inheritance tree — Remove the selected class and all its subclasses from the schema diagram.
- Collapse a class’s inheritance tree — Remove only the subclasses of the selected class from the schema diagram.
- Remove a class’s superclass — Remove only the superclass of the selected class from the schema diagram.

You can also use an abstraction function to hide individual relationships one at a time.

### How to apply an abstraction function

To apply an abstraction function:

1. Select the class or relationship to which you want to apply the abstraction function.
2. Select the abstraction function from the Schema | Alter menu, or from the class or relationship shortcut menu.

Inspector creates a new data view, with a schema diagram that reflects your changes.
Chapter 5
Collection Grids and Lists

Collection grids and collection lists are the primary way to work with ObjectStore collections in Inspector. This chapter describes the differences between collection grids and lists, and describes how to work with them.

This chapter covers the following topics:

Overview 60
Customizing a Collection Grid 65
Exporting a Collection Grid 72
Overview

Collection Grid Compared to Collection List

Collection grids and lists can be displayed in the instance panes of database views and in data views.

Example of a collection grid

Both grids and lists display the same information about an instance, that is, the data members, read user-defined methods, and icons you choose to associate with a particular instance. (See Customizing the Instance Display on page 79 for more information.)

Example of a collection list

Both grids and lists can be displayed at any time, and it is easy to switch a display from one to the other.
Choosing a Display Format

The following table summarizes some of the main differences between collection grids and collection lists. You might want to consider these factors when choosing a display format.

<table>
<thead>
<tr>
<th>Collection Grid</th>
<th>List</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can customize the grid display numerous ways. See Customizing a Collection Grid on page 65.</td>
<td>You can change only the instance display.</td>
</tr>
<tr>
<td>Information about navigated instances is easy to read in a grid layout.</td>
<td>Because lists display all instance information on a single line, understanding navigation relationships can be difficult.</td>
</tr>
<tr>
<td>You can export data in popular formats, including XML. See Exporting a Collection Grid on page 72.</td>
<td>You cannot export data from a collection list.</td>
</tr>
<tr>
<td>The grid is a snapshot of the underlying database and needs to be refreshed. See Refreshing Collections on page 62.</td>
<td>The list has a one-to-one correspondence with the database itself.</td>
</tr>
</tbody>
</table>

**Collection type dictates display format**

ObjectStore collections of heterogeneous objects — that is, objects that do not share a common ancestor object — are always displayed using a collection list; they cannot be displayed using a collection grid. Inspector displays an informational dialog box when you are loading a heterogeneous collection.

**Switching the collection display**

To turn the collection grid off and on (it is on by default), select Tools | Show Instances In Grid from the menu bar.

*Alternative:* Click the Use Grid Mode tool in the toolbar.

Populating Collection Grids and Lists

The first time you open a database, the instance pane displays an empty collection grid. Regardless of whether you choose to display ObjectStore collections using grids or lists, you populate them using the same procedure.

**How to populate a grid or list**

To populate a collection grid or list, select Instance | Show Class Extents from the menu bar, or select Show Class Extents on the class shortcut menu.

*Alternative:* Click the Show Class Extent tool in the toolbar; or double-click on a root in the Database Root pane, or on a class in the schema pane.
How Inspector Loads Collections

The way Inspector loads an ObjectStore collection varies based on whether you are displaying the collection using a collection grid or a collection list. Remember that Inspector always displays heterogeneous collections using a collection list.

Loading collection grids

Inspector loads the ObjectStore collection grid by fetching groups of instances from the ObjectStore database. The default size of each group is 150 instances. (Note that the actual number of instances might be more than the default, depending on the number of navigated instances associated with each instance fetched from the database.)

Inspector fetches the collection one group at a time on an as-needed basis. For example, if the collection you are displaying has 450 instances, Inspector loads only the first group, or 150 instances. The next group is fetched only if you request it (by scrolling the grid past the last instance in the first group of 150, for example).

Based on the size of your database and the number of instances associated with a typical collection, you might want to adjust this number up or down. You can change this number on the Collections Grid page of the Options dialog box. See Collection Grid Options on page 64 for more information.

Tip: You can force Inspector to load the entire collection into a grid, regardless of the default Options setting, by clicking Grid | Load Entire Collection on the menu bar, or by clicking the Load Entire Collection tool on the toolbar.

Loading collection lists

When you display ObjectStore collections in a list, Inspector loads only enough instances to fill the available window in which the list is displayed. Additional instances are fetched on-demand as you scroll the collection list.

Tip: When you drag the scroll bar, Inspector displays the current position in the list in a small pop-up caption that appears next to the scroll bar. This feature can help you quickly locate a particular instance in the collection.

Refreshing Collections

A collection list corresponds one-to-one with the ObjectStore database — it always reflects the current state of the database.

A collection grid, on the other hand, is a snapshot of the underlying database and needs to be refreshed periodically if you suspect the data might have changed since you first displayed the collection (it has been changed by an application, or by another Inspector instance, for example).

How to refresh a collection grid

To refresh a collection grid, click Grid | Refresh Collection on the menu bar.

Tip: Inspector refreshes a collection automatically any time you switch from the collection list to the collection grid display.
Finding a string

The collection grid enables you to search for a string within the collection, using the Find String dialog box:

To search for a string:

1. Click Grid | Find String on the menu bar.
2. Enter the string you want to find in the Search For field.
3. Click OK.
   
   If found, the cell containing the string you searched on appears in the upper left corner of the grid. The Edit Bar displays the cell location and value.
4. To continue the search, select Grid | Find Next on the menu bar.
Collection Grid Options

You use collection grid options to control performance and appearance characteristics of collection grids.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Instances to Load</td>
<td>Controls the number of instances in groups fetched from the database for display.</td>
</tr>
<tr>
<td>Use Most Recent Template</td>
<td>Automatically uses the last grid template associated with this collection.</td>
</tr>
<tr>
<td>Size Rows</td>
<td>Automatically resizes the cell width based on the cell’s content.</td>
</tr>
<tr>
<td>Size Columns</td>
<td>Automatically resizes the cell height based on the cell’s content.</td>
</tr>
<tr>
<td>Prompt to Save Modified Templates</td>
<td>Controls whether or not Inspector displays a prompt asking you if you want to save changes to the grid template.</td>
</tr>
</tbody>
</table>

To set collection grid options:

1. Click Tools | Options on the menu bar.
   The Options dialog box appears.
2. Click the Collection Grid tab.
3. Set the default values for the options you want.
4. Click OK.
Other Grid Features

In addition to changing the appearance of a grid, Inspector provides the following features to help you work with collection grids and share collection grid data.

<table>
<thead>
<tr>
<th>Feature</th>
<th>For More Information See</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exporting grid data to another application</td>
<td>Exporting a Collection Grid on page 72</td>
</tr>
<tr>
<td>Printing a collection grid (available on Windows platforms only)</td>
<td>Appendix A, Printing, on page 133</td>
</tr>
</tbody>
</table>

Customizing a Collection Grid

As described in Working With Data Views on page 42, Inspector provides several features to help you customize collection grids. This section describes these features.

Tip: You can save the changes you make to a grid using a grid template. See Saving Your Modifications on page 70 for more information.

Customization Procedure

The procedure for customizing a collection grid is the same regardless of which feature you are using.

1. Select the cell or cells you want to format.
   
   Tip: If you plan to save your formatting changes, consider selecting the entire column to ensure that the format is applied in a uniform way, regardless of the instance that occupies a particular cell.

2. Select the feature you want to use from the main database view menu bar or the data view toolbar.

3. Use the dialog box to define the feature you want to apply.

4. Repeat step 1 through step 3.

5. Optionally, save the customized grid. See Saving Your Modifications on page 70.

Note: The user interface for these features on UNIX platforms might vary from the interface described in this section.
Selecting Cells

Most cell operations require that you select the cell on which you want to perform the operation. There are several ways to select cells in collection grids. Most are probably familiar to you if you have worked with spreadsheet applications.

- To select an entire row or column, click the row or column heading, respectively.
  
  *Alternative:* Drag select the cells you want to select.

- To select an individual cell, click it.

- To select a group of contiguous cells, drag select them.
  
  *Alternative:* Click the first cell, press and hold the Shift key, then click the last cell you want to select.

- To select a group of noncontiguous cells, click on the first cell, press and hold the Ctrl key, then click the other cells you want to select.

*Tip:* You can use the Enter key and the arrow keys to move the cell focus around the grid.

Changing Cell Dimensions

You can change a cell’s height and width by dragging the appropriate border of the row or column header, respectively.

Changing the Alignment of Cell Data

You set the alignment of cell data using the Alignment dialog box.

![Alignment Dialog Box](image)

To change alignment:

1. Select the cells whose data you want to align.
2. Click **Grid | Cell Format** on the menu bar. Then select the **Align** tab.
   
   *Alternative:* Click the **Cell Format** tool on the toolbar. Then select the **Align** tab.
3. Select the horizontal and vertical alignment options you want.
4 Use the Word Wrap check box to indicate whether or not you want the text in a cell to wrap if the cell is not wide enough to display the cell name.

5 Click OK.

Customizing the Grid Border

You can select different patterns and colors for the grid cell borders using the Borders dialog box.

To change the grid border:

1 Select the cells whose border you want to customize.

2 Click Grid | Cell Format on the menu bar. Then select the Borders tab.
   
   Alternative: Click the Cell Format tool on the toolbar. Then select the Borders tab.

3 Select the pattern and color options you want.

4 Click OK.
Changing Cell Color and Pattern

You can fill cells with colors and patterns using the Pattern dialog box.

1. Select the cells whose color or pattern you want to change.
2. Click Grid | Cell Format on the menu bar. Then select the Color tab.
   *Alternative:* Click the Cell Format tool on the toolbar. Then select the Color tab.
3. Select the fill color (background), fill pattern (design), and pattern color you want to use.
   The Sample field displays how the cells will look with the pattern and color you have selected.
4. Click OK.

Changing the Font

You can select the font used to display data values in the collection grid using the Font dialog box.

1. Select the cells whose font you want to change.
2. Click Grid | Cell Format on the menu bar. Then select the Font tab.
Alternative: Click Cell Format tool on the toolbar. Then select the Font tab.

3 Select font elements you want to use (name, style, size, special effects, and so on). The Sample field displays how the data will appear using the font elements you have selected.

4 Click OK.

Annotating the Grid

A grid annotation is comparable to a screen caption (or callout).

You can create a grid annotation using

- Text — You can add text to empty cells. Just select an empty cell and start typing. You can format text using the same formatting tools available for cell data. See Changing the Font on page 68.

To annotate a grid with text:

1 Position the grid where you want to create the annotation.
2 Click the cell and type your comment.
3 Press Enter.
Creating Formulas

As with other spreadsheet applications, you can create formulas in collection grids. You can use them to perform operations on the values in one or more columns in a collection grid. (This feature is available on Windows platforms only.)

Copying formulas

Typically, a formula performs an operation on two or more cells representing ObjectStore instances to produce a value in a new cell. For example, a car dealership might want to obtain the cost of recharging an air conditioning unit by multiplying the amount of time required to perform the service by the mechanic’s hourly wage.

You define the formula in an empty cell, that is, one not already occupied by a collection instance. Once the formula is created, you can copy it down the entire column. Continuing the previous example, copying the formula down the column would provide the cost of all repairs listed in the collection — tune-ups, timing belt replacement, and so on.

How to create a formula

To create a formula:

1. Select a cell not that does not represent a value in the ObjectStore database.
2. In the cell, enter the formula you want to use.
3. Press Enter to perform the operation.

The value derived by the formula appears in the cell.

Saving Your Modifications

You can save the changes you make to a collection grid using a grid template. In addition, you can create multiple grid templates for a single collection; this allows you to display the same collection in different fashions for different audiences.

What is saved with the grid?

Inspector saves the following information for a grid:

- Grid format
- Cell format
- Data format
- Charts and graphs
- Annotation

How to save a grid template

To save a grid template:

1. Make the changes you want to the collection grid’s appearance.
2. Click Grid | Template | Save As on the menu bar.

Tip: If you want to save changes to an existing grid template, click Grid | Template | Save As on the menu bar
The Save Grid Template dialog box appears.

3 Type a name in the Grid Template Name field and click **OK**.

**How to apply a grid template**

The name of the current grid template is displayed in the upper left corner of the toolbar.

This field displays `<none>` if no grid template is applied to the collection.

*Note:* The collection is refreshed when you apply a grid template.

To apply a grid template:

1 Click **Grid | Template | Open** on the menu bar.
   
The Open Grid Template dialog box appears.

2 Select the template you want to apply from the Grid Templates for Class list box.

3 Click **OK**.

**Alternative**

To apply a template, select the template you want to apply from the grid template drop-down list in the toolbar.

**How to delete a grid template**

To delete a grid template:

1 Click **Grid | Template | Delete** on the menu bar.
   
The Delete Grid Template dialog box appears.

2 Select the grid you want to delete from the Grid Templates for Class list box.

3 Click the **Delete** button.
   
The system displays a message if you are attempting to delete the current template. If you choose to delete it, the collection grid is displayed without a grid template.
Exporting a Collection Grid

You can export a collection grid using XML or tabbed text.

You might want to export a collection grid in order to take advantage of a particular spreadsheet or presentation application using Inspector data.

Note: If you are using Inspector on UNIX, you can export a collection grid using XML and tabbed text formats only.

What Information Is Exported?

The collection grid information saved varies based on the format to which you are exporting the grid. For example, XML, Excel, and HTML are capable of preserving information about colors and fonts. For more information, consult the user documentation for the product in which you will be using the exported file.

How to Export a Collection Grid in XML

To export a collection grid in XML format:

1. Refresh the collection if you have not done so recently.
2. Click Grid | Generate XML on the menu bar.
   The Save As dialog box appears.
3. Enter a name in the File Name field.
4. Click Save.

How to Export a Collection Grid in Text Format

To export a collection grid:

1. Refresh the collection if you have not done so recently.
2. Click Grid | Export Data on the menu bar.
   The Save Table dialog box appears.
3. Enter a name in the File Name field.
4. Select the format type from the Save As Type drop-down list box.
5. Click Save.
Chapter 6
Classes and Instances

This chapter describes how you use Inspector to work with classes and instances in an ObjectStore database.

This chapter covers the following topics:

Where Instances and Classes Are Displayed 74
Customizing the Instance Display 79
Customizing a Collection Grid 84
Navigating Instances 85
Identifying Relationships Between Classes 91
Editing Data Members 93
Interpreting and Displaying Strings 98
Making Internal Classes Accessible 100
Where Instances and Classes Are Displayed

This section describes where and how Inspector displays instances and classes.

Instance Pane

The database view contains a Database Roots pane, a schema pane, and an instance pane. The instance pane occupies the right side of the database view window.

Default instance information

By default, Inspector shows the class’s address in persistent memory, along with an icon. You can change the information displayed for class instances to include its data members, and you can select different icons to represent the instance. See Customizing the Instance Display on page 79 for more information.

You can display instances using a list

Inspector displays instances using the collection grid by default. If you want, you can display instances in a collection list. See Chapter 5, Collection Grids and Lists, on page 59 for more information.

Instance Window

The Instance window shows all the data members for a particular instance, one instance at a time.
Information about each data member includes its name, type, and value.

**Purpose**

You can use the Instance window to

- Dump data member values to a file. See How to dump data member values to a file on page 75 for more information.
- Display character strings using hexadecimal notation. See How to display character strings using a hexadecimal viewer on page 76 for more information.
- Navigate the relationships between instances. See Navigating Instances on page 85 for more information.

**How to open the Instance window**

To open the Instance window:

1. Select the instance.
2. Select **Instance | Open Instance Window** from the menu bar, or select **Open Instance Window** from the instance shortcut menu.

   *Alternative:* Double-click on the instance (in the collections grid or instance list, for example).

**How to dump data member values to a file**

To dump data member values to a file:

1. Open the Instance window for the instance containing the data member whose values you want to dump to a file system.
2. Select **Dump to File System** from the shortcut menu of the appropriate data member.

   The Save As dialog box appears.

3. Type a name in the File Name field and click Save.
By default, data member values are displayed as character strings. You can display character strings in a hexadecimal viewer.

**Tip:** If you display a character string in a hexadecimal viewer, you can also
- Dump the hexadecimal value to a file.
- Show data member value’s offset inside the segment

To display a character string using a hexadecimal viewer:

1. Open the Instance window for the instance containing the data member whose values you want to display in a hexadecimal viewer.
2. Select **Show String in Hexadecimal Viewer** from the shortcut menu of the appropriate data member.

The field changes to display the character string in hexadecimal format.

3. Optionally, display the segment offset by selecting **Show Offset Inside Segment** from the hexadecimal viewer shortcut menu.

---

### Data Views

Data views are similar in appearance and functionality to the instance pane of the database view in that you can use them to display instances in either collection grids or collection lists.
Data views are distinct from the instance pane in that you can manipulate the instances in a data view using filters and by changing the order in which instances are displayed. Data views you define can be saved and reused.

For more information: To learn more about data views, see Chapter 3, Data Views, on page 41.

Instance: Physical Layout Property Dialog Box

The Instance: Physical Layout property dialog box displays information about the physical characteristics of a particular data member.

The dialog box has separate pages for the following types of information:

- **General** — The General page displays basic information about the data member: the object and instance it is associated with, and physical information such as address and size.

- **Segment** — The Segment page displays detailed information about the segment on which the data member resides, such as the segment’s number, size, number of pages, and amount of free space.

- **Binary Dump** — The Binary Dump page displays binary dump data for the data member.

You can open the Physical Database Layout window from this dialog box by clicking the Map Into Physical Layout button.

To open the Instance: Physical Layout dialog box:

1. Select the instance.

2. Select Instance | Show Physical Details from the menu bar.

   *Alternatively: Select Show Physical Details from the instance shortcut menu.*
Class Details Dialog Box

The Class Details dialog box displays the C++ declaration of the class.

In addition, it contains two pages, Set Instance Format and Class Icon, that let you customize the class and instance display within Inspector. See Customizing the Instance Display on page 79 for more information.

How to display the Class Details dialog box

To display the Class Details dialog box:

1. Select the class you want to work with from the schema diagram.
2. Click Schema | Show Class Details on the menu bar.

 Alternatively: Click Show Details on the class shortcut menu.

Physical Database Layout Window

The Instances pane of the Physical Database Layout window displays the instances that are stored within a given segment (the segment selected in the Segments pane).

Open Physical Database Layout window

To open the Physical Database Layout window, select Tools | Physical Database Layout from the menu bar. For more information: To learn more about the Physical Database Layout window, see Chapter 9, Tools for Physical Analysis, on page 121.
Customizing the Instance Display

This section describes how to use the Instance Format dialog box and other features to customize the display of instances in Inspector.

What Information Is Displayed by Default

By default, Inspector displays the following information for each class instance:

- Its address within persistent memory
- The ObjectStore icon

This information appears wherever the instance is displayed — collection grids, collection lists, the Navigation window, and so on.

Using the Instance Format Dialog Box

The Instance Format dialog box is a powerful tool for customizing instance formats.

Tip: These features are also available on the Class Properties dialog box, which you can display by clicking Schema | Show Class Details on menu bar.

You can use the Instance Format dialog box to control many aspects of how instances are displayed in Inspector. You can choose to

- Display data members belonging to subtypes of a supertype class
- Display read user-defined methods
- Display the member variable name
- Rename data members (for display purposes only)
- Associate an icon with the class

Note: You can display data members of classes that participate in recognized relationships only. A recognized relationship is one that is either
• Defined as part of the database,
• Created by Inspector

For more information on how Inspector works with ObjectStore relationships, see Identifying Relationships Between Classes on page 91.

Example
Consider the following simple database:

```
person
  -- name
  -- address

student
  -- courses
  -- tuition

professor
  -- courses
  -- salary
```

Using the Instance Format dialog box, you could create a display for instances of the person class that included, say, courses, tuition, and salary data members as appropriate. In addition, you could rename the courses data member associated with the professor class to courses taught, for example.

You could also associate different icons with each class to help distinguish the class to which a particular data member belongs. See Associating an Icon with a Class on page 82 for more information.

### Displaying the Instance Format Dialog Box

To display the Instance Format dialog box, select `Instance | Set Instance Format` from the menu bar or from the instance shortcut menu.

*Alternative:* Click the `Set Instance Format` tool on the toolbar.

### Selecting Data Members for Display

**How to identify data members for display**

To identify data members for display:

1. Open the Instance Format dialog box.
   The All Data Members list box displays all data members associated with the class.

   *Tip:* You can alter the contents of the list by clicking the Built-In Slots and Derived Classes check boxes.

2. To move a data member to the Data Members to Show list box, double-click the data member.

   *Alternative:* Click the data member and then click the right arrow button.

---

80 ObjectStore Inspector User Guide
3. If you want to display the member variable name as part of the data member, click the check box to the left of the data member.

4. Click the OK button.

If you change your mind

To remove a data member from the Data Members to Show list box:

1. Click the data member in the Data Members to Show list box.
2. Click the x button.

Alternative: Double-click the data member in the Data Members to Show list box.

How to reorder a list of data members

By default, data members are displayed for an instance in the order in which you select them from the All Data Members list box. If you want, you can change the display order.

To change the data member display order:

1. Click the data member you want to reorder in the Data Members to Show list box.
2. Click the up or down arrow button to change the order in which the data members will be displayed.

How to rename a data member

You can rename a data member. Note that this change affects the data member name only as it is displayed within Inspector. It does not alter the actual ObjectStore data member name. You might want to do this in order to simplify the presentation for reporting purposes, especially for a nontechnical audience.

To rename a data member:

1. Click the data member you want to rename.
2. Click the ab | button.

   An edit cursor appears at the beginning of the data member name.

3. Edit the name as needed. Be sure to delete any unneeded characters.
4. Press Enter.
Associating an Icon with a Class

By default, Inspector displays each class instance with the ObjectStore logo. You might want to associate more meaningful pictures with your classes. This can help you identify particular classes at a glance, which can be especially useful as your databases become more complex.

Choosing an icon

Inspector comes with a library of icons in different categories — computer, office, tools, for example. This library is visible from the Class Icon page of the Instance Format dialog box.

You can choose an icon from this library, or import bitmaps of your own.

Note: You cannot import bitmaps if you are using Inspector on UNIX.

Determining when to display an icon

By default, icons are associated with a class wherever the class appears. You can override this default in several places.

- Always Show Associated Icon check box on Class Icon page
- Tools | Show Associated Icon
- Instances page of Options dialog box (Always Show Associated Icon)

How to associate an icon with a class

To associate an icon with a class:

1. Open the Instance Format dialog box.
2. Click the Class Icon tab.
   The class name appears at the top of the Class Icon page.
3. Click the category of icons you want to browse.
4. Scroll the list of icons.
5. When you find the icon you want, click it. The icon appears in the preview field to the right of the icon list.
6. Click OK.
To import an icon:

1. Open the Instance Format dialog box.
2. Click the Class Icon tab.
3. Click the Import Icon button.

   The Open dialog box appears.

4. Open the icon (.ico) file you want to use.

   The icon appears in the preview field to the right of the icon list.

5. Click OK.

Instance Display Options

The Instances page of the Options dialog box contains a number of features that affect the display of instances in Inspector.

Choices on this page let you determine whether or not to

- Always associate icons with classes
- Hide static data members in instance windows and schema diagrams

Fonts

The Fonts page of the Options dialog box lets you select the font you want to use to display instance information in collection grids, collection lists, and the Instance window.

For more information: To learn more about Inspector options, see Inspector Options on page 27.
Customizing a Collection Grid

As described earlier, you can customize instances by selecting which data members you want to display and by associating icons with a class. In addition, you can customize the collection grid itself.

This section lists the different ways you can customize a collection grid, and collection grid features. See Chapter 5, Collection Grids and Lists, on page 59 for more information.

Ways You Can Customize a Collection Grid

Generally speaking, there are two ways to customize the appearance of a collection grid. You can

• Format cells
  - Specify a number format
  - Change the font
  - Change the alignment of data in a cell
  - Enhance the cell border
  - Add shading and color to the cell body
  - Annotate cells with shapes and text
  - Add formulas to perform computations based on cell data
• Attach charts that graphically summarize grid data (Windows only)

Other Collection Grid Features

In addition to customizing the appearance of a collection grid, Inspector provides other features to help you work with collection grids, whether they appear in the instance pane of the main database view or in data views.

• Grid templates — You can save customized collection grids as templates for use with other collections. For example, you might decide that you want to use a certain cell format for all finance-related collections. You can use grid templates to ensure that your collection grids have a uniform appearance.
• Collection grid options — The Collection Grid page of the Options dialog box provides a number of choices that affect both collection grid appearance (whether or not grid templates are used by default, for example) and performance (the number of instances to load each time the collection grid is displayed, for example).

For More Information

Collection grids can be displayed both in the instance pane of the main database view and in data views. Choices for customizing collection grid appearance are the same, regardless of where the collection grid is being used. See Chapter 5, Collection Grids and Lists, on page 59 for a complete description of the procedures you can use to customize collection grids and of other collection grid features.
Navigating Instances

_Navigation_ is the process of following relationships from one instance to another. When you navigate instances, Inspector records the resulting _navigation tree_ in the Navigation window.

This section describes the different ways you can navigate instances and the features of the Navigation window.

_For more information:_ You can also locate an instance based on its address or segment offset. See Tools for Debugging on page 131.

Two Ways to Navigate

There are two ways to navigate instances:

- **Manually** — When you navigate manually, you choose your own navigation path through the database by navigating from instance to instance, one instance at a time. You can navigate to any instance you choose. Manual navigation can originate only from the Instance window.
- **Automatically** — You can use Inspector’s Auto Navigate feature to quickly display all the instances related to the one you have selected. In addition, you can perform Auto Navigation from any place an instance is displayed.

Note, however, that Inspector is able to autonavigate recognized relationships only. See Navigating Automatically on page 86 for more information.
How to Navigate Manually

To navigate manually:

1. Open an Instance window for the instance whose relationships you want to navigate.

   Tip: See How to open the Instance window on page 75 if you need help.

2. Place the pointer on the related instance to which you want to navigate.

   Tip: The pointer changes to a magnifying glass when it is placed on a navigable instance.

3. Double-click to navigate to that instance.

   A new Instance window appears, displaying information for the instance to which you just navigated.

   In addition, Inspector opens a Navigation window that records the path to each instance you navigate. The window is minimized and appears in the lower left corner of the Inspector workspace.

4. Repeat step 2 and step 3 to navigate to other related instances.

Navigating Automatically

Recognizing relationships

When you use automatic navigation, Inspector navigates only recognized relationships. A recognized relationship is one that is either

- Defined as part of the database,
- Created by Inspector

For more information on how Inspector works with ObjectStore relationships, see Identifying Relationships Between Classes on page 91.

How to navigate automatically

To navigate all related instances automatically:

1. Select the instance you want to navigate.

2. Click Navigation | Auto Navigate from the menu bar, or click Auto Navigate on the instance shortcut menu.
Alternative: Press the Ctrl key and click the instance you want to automatically navigate.

Inspector opens a Navigation window that displays all navigable instances of the instance you selected in step 1.

About the Navigation Window

The Navigation window helps you keep track of the instance relationships you have navigated, whether manually, from the Instance window, or automatically, using the Auto Navigate feature.

![Navigation window symbols](image)

The Navigation window displays symbols that represent:

- **Instances** — The initial instance appears on the left side of the Navigation window. Subsequent instances follow, from left to right. An instance that is part of a one-to-many or many-to-many relationship is displayed in a collection list box containing the related instances.

  If an instance appears with a shaded background, it means that instance’s Instance window is no longer open.

  Tip: Remember that you control how instances are represented (icons, which data members are shown, and so on) using the Instance Format dialog box. See Customizing the Instance Display on page 79 for more information.

- **Relationships** — Relationships between instances are represented with arrows that point from the source instance to the one to which it is related, from left to right.

- **Data members** — The names of the data members used to establish the relationship are displayed alongside the instance with which the data member is associated.
You might want to consider changing the following characteristics of the navigation tree before printing a Navigation window.

- **Path layout** — The Path Layout dialog box lets you control the horizontal and vertical space between instances.

  ![Path Layout Dialog Box](image)

  The value controlling vertical spacing must be between 60 and 300, inclusive. The value controlling horizontal spacing must be between 10 and 200, inclusive.

  To display the Path Layout dialog box, click Navigation | Set Path Layout from the menu bar, or Set Path Layout from the Navigation window shortcut menu.

- **Navigation tree layout** — You use the Center Tree command to center the entire navigation tree within the Navigation window.

  To perform the Center Tree command, click Navigation | Center Tree from the menu bar, or Center Tree from the Navigation window shortcut menu.

  For more information: To learn more about printing Inspector information, see Appendix A, Printing, on page 133.

One of the benefits of the Navigation window is that it records the path you have taken through the database while inspecting classes and their instances. Because object databases can be complex, it can be valuable for you to be able to save the tree resulting from a particular navigation session, regardless of whether you created it manually or automatically.

To save a navigation tree:

1. Click File | Save on the menu bar.

   The Save Navigation Tree dialog box opens.
2 Name the navigation tree and click **Save**.
   The navigation tree is saved with the database.

Opening a navigation tree

To open a navigation tree you have saved previously:

1 Click **Navigation | Open Navigation Tree** on the menu bar.
   The Open Navigation Tree dialog box appears.
   
   *Tip:* The Open Navigation Tree dialog box is similar to the Save Navigation Tree dialog box.
   
   The list box displays all the navigation trees associated with the current database.

2 Select the navigation tree you want to open from the list box and click the **OK** button.
   
   *Alternative:* Double-click the navigation tree name.
   
   The Navigation window containing the navigation tree you selected appears.

Deleting a navigation tree

To delete a navigation tree:

1 Click **Navigation | Delete** on the menu bar.
   The Delete Navigation Tree dialog box appears.
   
   *Tip:* The Delete Navigation Tree dialog box is similar to the Save Navigation Tree dialog box.
   
   The list box displays all the navigation trees associated with the current database.

2 Select the navigation tree you want to delete from the list box and click the **Delete** button.

Other Navigation window features

In addition to the layout features already described, you can use the Navigation window shortcut menu to

- Open and close an Instance window
- Close all open Instance windows associated with instances in the navigation tree
- Perform another auto navigation task
Navigation Options


Choices on the Navigation page let you set default values for

- Navigation tree formatting characteristics
- Handling open Instance windows when quitting a navigation session
- Handling open Navigation windows related to the current Navigation window when quitting a navigation session

To learn more about Inspector options, see Inspector Options on page 27.
Identifying Relationships Between Classes

You typically create a relationship between two classes as part of defining an ObjectStore database. Among other things, the presence of these relationships enables Inspector to autonavigate from one class to another.

This section describes how Inspector uses information that is not part of the schema definition to identify relationships in an ObjectStore database.

Inspector Creates Some Relationships Automatically

Inspector creates a relationship between two classes when one class, say ServiceItem, contains a data member that is a templated ObjectStore collection. By definition, these collections consist of a homogeneous set of instances, making the relationship between the two classes meaningful.

Inspector Identifies Possible Relationships

If WorkOrder, on the other hand, is a nontemplated collection, Inspector first reviews the collection’s instances. If it determines that all of the instances are of the same type, it displays the following dialog box, suggesting that a possible relationship exists and giving you the opportunity to create it.

Click Yes to establish relationships between the two classes shown in the dialog box. If you click No, you can create the relationship later as follows.

1 In the Schema Pane right click on a class and select Examine Relationships from the shortcut menu. The Examine Oneway Relationships dialog box is displayed showing relationships that have already been created as well as possible relationships that have not yet been created.

2 Click on the checkbox next to the relationship you want to create.
Relationships Between Java Classes

In Java databases, Inspector automatically creates a one-way relationship between the abstract class `Object` and a subclass when the subclass has a field with a collection data type such as `com.odi.coll.set`. (ObjectStore maps the class `os_dma_singleton` to the Java class `Object`.)

In this example, a one-way relationship between `Customer` and `Object` is created based on the `myCars` field in the `Customer` class, which has the type `com.odi.coll.set`.

If you then open an instance of `Customer`, Inspector identifies a possible relationship between `Customer` and `Cars`. If you decide to create this relationship, Inspector removes the relationship it created between `Customer` and `Object`, leaving only the one between `Customer` and `Car`.

Other Areas Affected by Relationships

There are other areas of Inspector that are affected by the relationships Inspector is able to recognize. These include selecting data members for

- A custom instance display. See Selecting Data Members for Display on page 80.
- A data view filter. See Filtering a Collection on page 45.
- A data view order. See Ordering a Collection on page 53
Editing Data Members

This section describes how to edit data members using the Edit Data Member dialog box.

Use Care When Editing Data Members

When you edit a data member, you are editing a value in an ObjectStore database. Be sure that you are aware of all the effects changing a data member might have on your database and the applications that use it.

Example

Consider editing an instance of type char*. If you edit this instance using the Edit Data Member dialog box, you need to decide whether to

- Create a new block for the edited instance and retain the existing block
- Create a new block for the edited instance and delete the existing block
- Use the existing block

Making any of these choices might cause side effects that Inspector cannot predict. Alternatively, you could use manage this behavior using user-defined methods.

For more information: See Chapter 7, User-Defined Methods, on page 103 to learn more about user-defined methods.

Using the Edit Data Member Dialog Box

The Edit Data Member dialog box has a number of features that allow you to edit several types of instances.

The Editable Data Members list box appears on the left side of the dialog box. It displays all the editable data members of the instance you have selected from the database.

Tip: Once you edit a data member value, the data member type and name appear in italics.

The right side of the dialog box, which is initially empty, displays different tools based on the data member you select.
• The Member Variable Value field displays the data member’s current value. Different display options are available for different data member types. For example, you can display a `char*` type data member using either its literal value, or hexadecimal notation alongside the literal value. Inspector displays a list of valid values for data members with an enumerated type.

• The `>>>` button expands the dialog box to display other information, such as extent values for ObjectStore relationships. Once you have extended the dialog box, the `>>>` button changes to `<<`, which enables you to restore the dialog box to its original dimensions.

How to open the Edit Data Member dialog box

To open the Edit Data Member dialog box:

1. Select the instance you want to edit.
2. Click Instance | Edit on the menu bar, or select Edit from the instance shortcut menu.

How to Edit a Data Member

Prerequisite

In order to edit a data member, you must have write access to the ObjectStore database in which the instance to which it belongs is stored. If you have navigated to an instance in a database other than the one you opened in Inspector, your ability to write to that database is controlled by the Open Databases in MVCC to Resolve External References field on the Instances page of the Options dialog box.

This option is selected by default, which means that you will not have the write access required to edit navigated instances.

For more information: See Inspector Options on page 27 to learn how to set options.

Changing your mind

Inspector provides several ways for you to undo edits to data members before committing them to ObjectStore. You should be familiar with these methods before editing a data member value.

• Original Values button — The Original Values button appears beneath the edit field on the Edit Data Member dialog box once you have changed a data member’s value. Click this button to undo the current edit.

• Cancel button — The Cancel button on the Edit Data Member dialog box undoes the current edit and closes the dialog box.

• Confirmation dialog box — If you click OK on the Edit Data Member dialog box, Inspector displays a confirmation message.

Click No if you change your mind about the edit to the current data member but want to review other changes.

Click Cancel to return to the Edit Data Member dialog box. Note: Clicking Cancel does not restore edited data members to their original values.
The basic procedure for editing data members using the Edit Data Member dialog box is shown here. Information specific to editing different types is provided following the procedure.

To edit a data member:

1 Select the instance containing the data member you want to edit.
2 Open the Edit Data Member dialog box.
3 Select the data member whose value you want to edit from the Editable Data Members list box.
   The contents of the dialog box change based on the type of the data member you selected.
4 Make the edits you require.
   The Original Value button appears beneath the edit field.
5 Click OK to complete the change.
   Inspector displays a dialog box asking you to confirm the edit.
6 Click Yes to review each edit independently.
   Click Yes to All if you want Inspector to accept all edits without any additional confirmation.

Recommendations for Editing Specific Types

char and char*
You can edit character strings (type char) and pointers (type char*) using either the literal value or the hexadecimal value.

If you choose to edit the hexadecimal value, make sure that the string terminates with \00 (null). If you do not terminate the string with the null character, Inspector interprets the data member type as blob.

Enumerated types
When you edit an enumerated type, you choose from a list of valid values in the Member Variable Value field. For example, for a type of EnumSize, the Member Variable Value field might display a list containing small, medium, and large.
When you edit a data member variable that is either a block of memory or a pointer to memory, you can change the hexadecimal value.

You have three choices regarding the block of memory itself.

**Memory Block Option**

- **Make the changes in a new block and delete the existing one**
  - When the block of memory containing the instance you are editing is used only by that instance.

- **Make the changes in a new block and retain the old one**
  - When the block of memory containing the instance you are editing is used by other instances.

- **Make changes in the existing block**
  - When the new value is the same size as or smaller than the existing value. Note that you cannot add new bytes to the end of the block.

Note that new blocks of memory are always created in the default segment.

**Relationships**

You edit relationships by adding instances to or removing instances from the relationship as shown in the Member Variable Value list box.

*Tip:* The `<>` symbol identifies relationships in the Editable Data Members list box.

If you click the expand button (>>), the Edit Data Member dialog box displays the extent of the class related to the class whose instance you are editing in the Extent of Class list box. (If this list box is empty, it means either that the extent is empty or that Inspector does not know where to locate the extent.) You use the right arrow button...
to remove instances from the relationship; you use the left arrow button to add instances to the relationship.

Note that Inspector maintains referential integrity for ObjectStore relationships. For example, consider a Triumph vehicle instance associated with (owned by) a Smith customer instance. If you edit the Triumph instance and associate it with, say, Homer, that Triumph is no longer owned by Smith once you confirm the edit.

You edit pointers in the same way you edit relationships: by adding and removing instances. Remember, to display instances, click the expand (>>) button.

**Tip:** The => symbol identifies pointers in the Editable Data Members list box.

To make a pointer point to an instance:

1. Select the instance from the Extent of Class list box.
2. Click the left arrow to place the instance in the Member Variable Value entry field.
3. Click the OK button.

To set a pointer to NULL:

1. Select the instance in the Member Variable Value entry field.
2. Click the right arrow to remove the instance.
3. Click the OK button.
Interpreting and Displaying Strings

Overview

C++ enables you to encode strings in several formats, the most common of which is ASCII. The String Formats page of the Options dialog box contains options that

- Help Inspector interpret strings
- Let you control how strings are displayed
- Let you control how single characters are displayed

You can use these options to make the information you see in Inspector consistent with its native format, that is, the format in which you are familiar with it.

For more information: To learn more about options and how to set them, see Inspector Options on page 27.

When Are Interpretation and Display Rules Used?

By default, Inspector interprets (and displays) any string with more than 768 characters as an ASCII blob. Instances that are interpreted as blobs are displayed using a hexadecimal viewer.

Strings with fewer than 768 characters are interpreted and displayed based on the rules you select.

Interpreting Strings

By default, Inspector interprets strings using plain ASCII. You use the options in the Interpret Strings As group box to

- Specify a different standard
- Provide other information to help Inspector interpret strings
ASCII alternatives

You can use the Default Encoding Type drop-down list to select an alternative to plain ASCII:

- SJIS — the standard Japanese double-byte character string for Windows
  Note that there is a separate option to help Inspector interpret Japanese strings coded using formats other than SJIS.
- JIS — Japanese double-byte character strings for UNIX
- EUC — the standard UNIX double-byte character string
- UTF-8 — Unicode for Java
- Unicode little endian — Unicode in little endian format
- Unicode big endian — Unicode in big endian format
- Automatically Recognize SJIS, JIS, or EUC — inspector decides how to interpret strings

Interpreting unsigned arrays

By default, unsigned arrays are interpreted as Unicode encoded strings in little endian format; big endian format is also supported. If the Automatically Recognize SJIS, JIS, or EUC field is not checked, unsigned arrays are displayed as arrays of numbers.

Displaying Strings

The options in the Show Strings group box let you define

- How long a string must be before it is considered a blob
- How to display strings

When is a string a blob?

By default, a string is interpreted as a blob if it exceeds 768 characters.

Display options

If a string is not interpreted as a blob, you can display the string

- Up to the first NULL character (assumes the C++ standard of 0 for the NULL character)
- Skipping invalid characters; invalid characters are not included in the display
- Substituting a special character for invalid characters; the default special character is a question mark (?)

If the string is interpreted as a blob, you can choose to display the string in ASCII, up to the first NULL character; remaining characters are truncated.

Displaying Single Characters

By default, Inspector displays single characters along with their ASCII values. For example, the letter D is shown as ’D’ (68). If you want, you can display single characters by themselves.
Overriding Inspector Defaults

You can override default settings for individual data members using the Data Member String Format dialog box.

The settings on this dialog box are the same as those on the String Formats page of the Options dialog box.

How to display the Data Member String Format dialog box

To display the Data Member String Format dialog box:

1. Display a database view.
2. Select the schema pane.
   
   Note: You cannot use the tree hierarchy format for this procedure.
3. Click Schema | Data Member String Format on the menu bar.
   
   Alternative: Click Set Attribute Format on the diagram content menu.

   The Data Member String Format dialog box appears.

Making Internal Classes Accessible

By default, ObjectStore internal classes are not accessible in Inspector. These classes are part of every ObjectStore database, but Inspector filters them out to simplify the schema presentation.

This section describes how to override this default.

How Internal Classes Are Identified

Inspector uses these strings to identify which classes to omit from the schema:

- os_
- _

(ObjectStore internal classes typically begin with either of these strings.)
Use the Classes Option to Override

You can use the Classes page of the Options dialog box to override this default.

You use this page to define wildcards that represent the classes Inspector would otherwise hide. The wildcards you use should be as narrow as possible to avoid loading the schema with unnecessary detail.

Tip: If you do not know how to work with the Options dialog box, see Inspector Options on page 27.

When to use

Consider the Classes option if you have used either `os_` or `_` to prefix user-defined class names, or if you want Inspector to make certain ObjectStore classes accessible.

Important: Inspector makes internal classes available by redefining them as user-defined classes. Be careful when deciding which internal classes you want to make accessible. Redefining `os_collection`, for example, as a user-defined class can affect the collection and filtering functionality associated with these objects.

How to use the Classes page

To use the Classes page:

1. Display the Classes page of the Options dialog box.
2. Enter the wildcard in the Wildcard Name field.
3. Click the Add button to place the wildcard in the Defined Wildcards List list box.
   - If you change your mind about a wildcard, click the Delete button to remove the wildcard from the list box.
4. Repeat step 2 and step 3 to define other wildcards.
5. Click OK.
   - Inspector displays an informational message.
6. Click OK.
7. Reopen the database by clicking File | Open on the menu bar or by clicking the Open tool on the toolbar.
The Open dialog box appears.

8 Select the database from the list box.

9 Click the Ignore Stored Metaknowledge check box.

Inspector displays an informational dialog box notifying you that it is going to close all active documents.

10 Click Yes.

Inspector closes and then reopens the database. The Database Roots and schema panes display any classes defined using the wildcards you defined in step 2.

For more information: To learn more about metaknowledge, see Appendix C, Working with Metaknowledge, on page 145.
Chapter 7
User-Defined Methods

User-defined methods allow you to work directly with objects in an ObjectStore database from within Inspector. Specifically, user-defined methods allow you to perform the following operations by invoking the method from within Inspector:

• Create ObjectStore objects
• Read ObjectStore objects
• Update ObjectStore objects
• Delete ObjectStore objects

In addition you can use a user-defined method to retrieve sets of ObjectStore objects.

This chapter describes how to work with user-defined methods in Inspector.

This chapter covers the following topics:

User-defined Methods Overview 104
Defining User-Defined Methods 104
Registering User-Defined Methods 107
Unregistering User-Defined Methods 110
Invoking User-Defined Methods 111
User-defined Methods Overview

This section describes the process of using user-defined methods in Inspector.

Prerequisites

In order to use user-defined methods in Inspector, you must observe the following rules:

- You must have created the database you are working with in Inspector by using ObjectStore 6.0 or later.
- The database must have been created by an application linked to one or more ObjectStore DLL schemas.
- You cannot use user-defined methods in Inspector to open or close a database; you cannot use them to start, abort, or commit the top-level ObjectStore transaction.
- The DLL schemas must have been built specifying this directive:
  ```
  OS_SCHEMA_DLL_ID ("DLL:DLLName")
  ```
  where `DLLName` is the names of the DLL schema libraries, without any `.dll` or `.so` extension.
- The DLL schemas must be available in your system path (or, on UNIX, your library path).
- If you are using ObjectStore on UNIX, the application schemas referenced by the DLL schemas must have been generated specifying the `-store_member_functions (-smf)` directive to the `ossg ObjectStore` utility.

For more information about DLL schemas and the `ossg` utility, see *Managing ObjectStore and Building ObjectStore C++ Applications*.

Process

The process for using user-defined methods in Inspector involves the following:

1. Define the methods in the ObjectStore DLL schemas.
2. Register the methods using Inspector.
3. Invoke the methods.

The following sections describe each stage in the process in detail.

Defining User-Defined Methods

This section describes the purpose of the user-defined methods supported in Inspector and how to define them.
Purpose

The purpose of the user-defined methods you can invoke through Inspector is defined in the following table.

<table>
<thead>
<tr>
<th>Method</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>To create a persistent ObjectStore object</td>
</tr>
<tr>
<td>Read</td>
<td>To return computed values from a persistent ObjectStore object</td>
</tr>
<tr>
<td>Update</td>
<td>To modify a persistent ObjectStore object</td>
</tr>
<tr>
<td>Delete</td>
<td>To delete a persistent ObjectStore object</td>
</tr>
<tr>
<td>Extent</td>
<td>To retrieve a set of persistent ObjectStore objects</td>
</tr>
</tbody>
</table>

Required Signature

Each user-defined method supported by Inspector has a required signature that must be exposed in order for it to be registered. This section describes those signatures.

**Create**

static PersistentClass* PersistentClass::methodName(
    os_database* aDB,
    const os_Dictionary<char*, void*>& args
)
or

static PersistentClass* PersistentClass::methodName(
    os_segment* aSeg,
    const os_Dictionary<char*, void*>& args
)

You use the aDB and aSeg arguments to specify information about the persistent storage location to be used. The args argument can be used to specify arguments to the invoked method. The dictionary key is a mnemonic name for the argument; the dictionary value can be any of the following C++ types:

- int* (pointer to a 32-bit integer)
- double*
- char*
- os_collection*
- void*

If one of the registered arguments of a create method is of type os_collection*, Inspector builds a transient os_collection and lets you specify any number of persistent objects to populate it. If type void* is one of the method’s registered arguments, Inspector allows you to specify one persistent object to define the argument.

The create method cannot delete any key or value specified in the dictionary, and it should return the pointer to the newly created instance if no error conditions are detected (NULL otherwise). Inspector opens the database in write mode and starts an update transaction before calling the method.
Read

```
returnType PersistentClass::methodName(void)
```

where `returnType` can be any of the following C++ types:

- `signed/unsigned int`
- `signed/unsigned long`
- `signed/unsigned short`
- `signed/unsigned char`
- `float`
- `double`
- `long double`
- `char *`
- `PersistentClass*`
- `os_Collection<PersistentClass*>*`
- Any `os_collection` subclass

After you have registered a read method, Inspector is able to include it in the class instance format and to output the returned values as it would do with any class data member of the same type as the method’s return type. Inspector opens the database in MVCC mode and starts a read-only ObjectStore transaction before calling any read method. If the return value is a pointer to an object, the object itself is deleted by Inspector if the object is transiently allocated.

Update

```
int PersistentClass::methodName(
    const os_Dictionary<char*, void*>&args
)
```

The `args` argument can be used to specify arguments to the invoked method. The dictionary key is a mnemonic name for the argument; the dictionary value can be any of the following C++ types:

- `int*` (pointer to a 32-bit integer)
- `double*`
- `char*`
- `os_collection*`
- `void*`

If one of the registered arguments of an update method is of type `os_collection*`, Inspector builds a transient `os_collection` and lets you specify any number of persistent objects to populate it. If type `void*` is one of the method’s registered arguments, Inspector allows you to specify one persistent object to define the argument.

The update method cannot delete any key or value specified in the dictionary, and it should return a nonzero value if no error conditions are detected (0 otherwise).

Inspector opens the database in write mode and starts an update transaction before calling the method.
Delete

```c
static int PersistentClass::methodName(
    void* anObject
)
```

The `anObject` argument is used to specify the pointer to the persistent object that is to be deleted. Inspector assumes that `anObject` is actually a pointer to an instance of `PersistentClass`.

The `delete` method should return a nonzero value if no error conditions were detected (0 otherwise).

Inspector opens the database in write mode and starts an update ObjectStore transaction before calling the method.

Extent

```c
static int PersistentClass::methodName(
    os_database *aDB, os_collection& aCollection
)
```

The `aDB` argument is the currently Inspected database. The `aCollection` is a collection created by Inspector which the user needs to populate.

The `extent` method should return a nonzero value if no error conditions were detected (0 otherwise).

Inspector opens the database in MVCC mode and starts a read-only ObjectStore transaction before calling the method.

Registering User-Defined Methods

Once you have defined a user-defined method, you need to register it before it can be invoked by Inspector. Registering a method is the process of adding information about the method to Inspector’s metaknowledge about the current ObjectStore database.

This section describes how to register and unregister different user-defined method types.
The Register User-Defined Methods Window

You use the Register User-Defined Methods window to register all types of user-defined methods.

The window contains five pages, one for each method type. The main list box displays all the user-defined methods in the DLL schemas associated with the current database that satisfy the signature rules for the type of method you are registering.

Tip: A check mark to the left of the method indicates that the method is already registered.

How to open the window

To open the Register User-Defined Methods window:

1. Select an instance.
2. Select Instance | User-Defined Methods | Register from the menu bar.
   
   Alternative: Select User-Defined Methods | Register from the Instance shortcut menu.

   The Register User-Defined Methods window appears.

How to Register User-Defined Methods

The basic procedure for registering a user-defined method is the same, regardless of the type. Note that update and create user-defined methods also allow you to define arguments. The procedure for defining arguments is described separately.

To register a user-defined method:

1. Open the Register User-Defined Methods window.
2. Click the tab that corresponds to the type of method you want to register.
3. Select the methods you want to register.
   
   Alternative: You can double-click a single method to register it.

4. Are you registering either update or create user-defined methods?
If yes, see How to Add Arguments to Update and Create Methods on page 109.

If no, go to step 5.

5 Click the Register button.

6 To register another type of user-defined method, go to step 2. Otherwise, click OK to close the Register User-Defined Methods window.

How to Add Arguments to Update and Create Methods

When you register update and create user-defined methods, you also need to add the expected arguments and their types to Inspector's metaknowledge. For example, consider the following user-defined create method:

```cpp
CPerson* CPerson::createPerson(
    os_database * aDB,
    const os_Dictionary<char *, void *> & args)
{
    CPerson* newPerson=new (aDB, get_os_typespec()) CPerson;
    newPerson->set_m_name ((char*)args.pick("name"));
    newPerson->set_m_city ((char*)args.pick("city"));
    int* age=(int*)args.pick("age");
    if(age!=NULL)
        newPerson->m_age=*age;
    return newPerson;
}
```

In this example, the user-defined method expects to find three different arguments in the directory: one for name, city, and age. These arguments need to be explicitly added to the method in Inspector.

Note: Arguments are case-sensitive.

How to add an argument

To add an argument to a user-defined method:

1 Open the Register User-Defined Methods window.

2 Click the tab that corresponds to the type of method you want to register (Update or Create).

3 Select the method you want to register.

   The Arguments list box displays any arguments that have already been defined for that method.

4 Select the Name field and type the argument name.

5 Select a type for the argument from the Type drop-down list.

6 Click Add to associate the argument with the method.

7 Repeat step 4 through step 6 for each argument you want to add to the user-defined method.

8 Is the user-defined method already registered?

   If yes, click OK to close the Register User-Defined Methods window.

   If no, click Register to register the user-defined method and its arguments.
How to delete an argument

To delete an argument:

1. Open the Register User-Defined Methods window.
2. Click the Update or Create tab to display the method whose arguments you want to delete.
3. Select the argument you want to delete.
4. Click the **Delete** button.
5. Repeat step 3 and step 4 for any other arguments you want to delete.
6. Click **OK** to close the Register User-Defined Methods window.

How to rename an argument

To rename an argument:

1. Open the Register User-Defined Methods window.
2. Click the Update or Create tab to display the method whose arguments you want to rename.
3. Select the argument you want to rename.
4. Click the **Rename** button.
5. Repeat step 3 and step 4 for any other arguments you want to rename.
6. Click **OK** to close the Register User-Defined Methods window.

Unregistering User-Defined Methods

There are several reasons you might want to unregister a user-defined method from Inspector's metaknowledge. For example:

- The user-defined method might have been deleted from the schema DLL, so it no longer needs to be registered.
- You want to simplify the Inspector's metaknowledge by unregistering user-defined methods that are no longer needed.

This section describes how to unregister a user-defined method.

*Tip:* A check mark appears to the left of every registered user-defined method.

How to Unregister a User-Defined Method

To unregister a user-defined method:

1. Open the Register User-Defined Methods window.
2. Click the tab that corresponds to the type of method you want to unregister.
3. Select the methods you want to unregister.
   *Alternative:* You can double-click a single method to unregister it.
4. Click the **Unregister** button.
5. To unregister another type of user-defined method, go to step 2. Otherwise, click **OK** to close the Register User-Defined Methods window.
Invoking User-Defined Methods

Once you have registered a user-defined method with Inspector’s metaknowledge, you can invoke it during any Inspector session with the database with which you registered the method.

This section describes how to invoke different user-defined method types.

How to Invoke User-Defined Methods

The basic procedure for invoking a user-defined method is the same for all methods except extent methods. This section describes the basic procedure for invoking create, read, update, and delete methods. Type-specific procedures, including invoking extent methods, are detailed in the sections that follow.

To invoke a user-defined method:

1 Select an instance belonging to a class for which you have registered a method.
2 Select the method you want to invoke from the User-Defined Methods drop-down menu.

   Tip: The User-Defined Methods drop-down menu is available from the Instances choice on the menu bar, or from the instances shortcut menu.

3 Respond to the dialog box the system displays as a result of the method.

   Note: The dialog box displayed by Inspector varies based on the type of user-defined method you invoked.

4 Click Execute.
Invoking Read Methods

There are two ways to invoke user-defined read methods:

- Explicitly, following the procedure described above
- Indirectly, by adding the method to the class instance format

This section describes how to invoke a read method by adding it to the class instance format.

When is the method invoked?

User-defined read methods that are used in the definition of a class instance format are invoked each time Inspector computes the instance format for any object of the specified class.

How to add a method to the instance format

To add a user-defined method to an instance format:

1. Open the Instance Format window for an instance belonging to a class for which you have registered a method.

2. Expand the data members of the class to display the user-defined method.

3. Select the user-defined method and click the right arrow button to add the method to the Data Members You Want To Show list box.

The instance pane is updated to include the values associated with the user-defined method you selected.

Invoking Create and Update Methods

You invoke user-defined create and update methods using the procedure described in How to Invoke User-Defined Methods on page 111.

Once you have invoked the method, Inspector displays a dialog box that allows you to provide data members for the method’s argument.
The specifics of this dialog box vary based on the method and its arguments.

To enter values in the entry fields of the dialog box:

- Drag data members from an instance list or from a schema diagram to the entry field.
- Copy a data member to the clipboard and paste it in the entry field.

*Tip:* The trash bin that appears to the left of the entry fields on some dialog boxes can be used to remove data members from the entry field. If you change your mind about a particular data member, drag the data member from the list box to the trash bin.

Executing the method

Once you have provided the data members that satisfy the method’s arguments, click the *Execute* button to execute the method.

Inspector displays an instance window with the updated (or created) instance if the method executed successfully.

**Invoking Delete Methods**

You invoke user-defined delete methods using the procedure described in How to Invoke User-Defined Methods on page 111. Inspector displays a dialog box asking you to confirm the delete operation.

**Invoking Extent Methods**

User-defined extent methods are used to create data views. To invoke this type of user-defined method you create a data view as follows:

1. Right click on the class in the Schema Pane.
2. Select *User-defined Methods | Extent* from the shortcut menu and then select the user-defined method.

A data view window appears, listing the data retrieved by the user-defined extent method.

See Creating a Data View on page 42 for more information.
Chapter 8
Roots

A root (also referred to as a database root) is a persistent object that serves as an entry point into an ObjectStore database. This chapter describes how to create and work with roots in Inspector.

For more information: To learn more about roots, see the ObjectStore C++ API User Guide.

This chapter covers the following topics:

Creating a Root 116
Working with Roots 117
Creating a Root

Two Parts to Creating a Root

The process of creating a root in Inspector has two steps:

• Create the root.
• Define the root value.

You can perform these steps in either order; that is, you can create a root and then determine what values you want to set it to, or you can identify a set of values and define the root based on those values.

Note: Creating a root requires write access to the ObjectStore database.

Choosing a Root Value

You can define the root value using any of the following ObjectStore collection types:

• Data view
• Extent
• Instance

You can set these collections to new or existing roots. Setting a collection to a new root creates the root; setting a collection to an existing root redefines the root based on that collection.

Creating a Root Only

This procedure describes how to create a root. Once you create the root, you must set its value based on a collection. See Redefining a Root Value on page 117 for more information.

How to create a root

To create a root:

1. Click Roots | Create on the menu bar.
   The Define New Root dialog box appears.

2. Enter a name for the root in the Name field.

3. Click OK.
   The root is created and added to the Database Roots pane of the database view window.

4. Set the root value using the procedure described on Redefining a Root Value on page 117.
Creating a Root and Defining the Root Value

Use this procedure to define a root based on the value of a data view, extent, or instance. See Redefining a Root Value on page 117 to learn how to change values for an existing root.

To create a root based on a data view, extent, or instance:

1. Select the collection on which you want to base the root.
2. Click Roots | Set Value on the menu bar.
3. Select the appropriate collection type from the Set Value drop-down menu.

The Choose Root dialog box appears.

4. Enter a name for the root in the Root Name field and click OK.

Inspector displays a dialog box asking if you want to create a new root.

5. Click Yes.

Inspector displays another dialog box asking if you want to set the root value using the currently displayed collection.

6. Click Yes.

A new root is created and appears in the Database Roots pane of the database view window.

Working with Roots

You can redefine root values and delete roots. Before performing either of these operations, be sure that you understand how these operations can affect existing applications.

Redefining a Root Value

When you redefine a root value, applications that access the ObjectStore database using that root are unaffected. Note, however, that values identified by the previous root are no longer accessible — the root performs as before but uses new values.
How to change the value of an existing root

To change the value of an existing root:

1. Select the data view, extent, or instance on which you want to base the root.
2. Click **Roots | Set Value** on the menu bar.
3. Select the appropriate choice from the Set Value drop-down menu.
   
   The Choose Root dialog box appears.
4. Select the root you want to redefine from the list box and click **OK**.
   
   Inspector displays a dialog box asking if you want to set the root value using the currently displayed collection.

   ![ObjectStore Inspector 3.0 dialog box](image)

   Do you want to set the value of root 'vehicle_view' to a collection filled with the instances currently displayed in the active instances View?
   
   Options: **Yes**, **No**

5. Click **Yes**.
   
   Inspector displays a warning informing you that changing the value of an existing root might affect applications using that root.
6. Click **Yes** to proceed, **No** to cancel.

Deleting a Root

When you delete a root, you are effectively making persistent data unreachable — applications that attempt to access the ObjectStore database using that root will fail. Generally speaking, you should never delete a root unless you are certain that you have no use for the values identified by it.

Deleting root values

When you delete a root, you can also delete the root values themselves. This is a more extreme operation than merely deleting the root itself.

Warning

Generally speaking, deleting a root and deleting a root’s values are operations with a high degree of risk. You should perform these operations only when you are certain that

- Existing applications will not be negatively affected.
- There is no need for any of the persistent data represented by the root values.

How to delete a root

To delete a root:

1. Make sure that you no longer need to access the values identified by the root.
2. If you plan to delete the root values themselves, make sure that you no longer need that data.
3. Select the root from the Database Roots pane of the database view window.
4. Click **Roots | Delete** on the menu bar.
Inspector displays the Deleting a Root dialog box.

5 Indicate whether you want to delete
   - The root only
   - The root *and* the root values

6 Click OK to continue.
Chapter 9
Tools for Physical Analysis

Database views, data views, and Navigation windows are all ways to explore the logical relationships and data in an ObjectStore database. The Physical Database Layout window provides a way for you to see how your data is organized in ObjectStore’s persistent memory.

This chapter describes the Physical Database Layout window and how to use it, as well as other tools to help you work with free space and debug applications.

For more information: To learn about tools to help you work with the logical information associated with an ObjectStore database, see Chapter 2, Database Views, on page 29, and Chapter 3, Data Views, on page 41.

This chapter covers the following topics:
- Layout Window 122
- Segments 124
- Statistics 127
- Instances 128
- Working with Free Space 129
- Tools for Debugging 131
Layout Window

You use the Physical Database Layout window to examine how the data of an ObjectStore database is organized within the persistent memory managed by ObjectStore.

This section describes the Physical Database Layout window and the types of information it displays.

Opening the Physical Database Layout Window

There are three ways to open the Physical Database Layout window:

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
<th>What Is Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu bar</td>
<td>Click Tools</td>
<td>Physical Database Layout.</td>
</tr>
<tr>
<td>Toolbar</td>
<td>Click the Physical Database Layout tool.</td>
<td>Segment and statistic information for all user segments in the database</td>
</tr>
<tr>
<td>Instance: Physical Layout dialog box</td>
<td>Click the Map Into Physical Layout button.</td>
<td>Segment, statistic, and instance information for that instance</td>
</tr>
</tbody>
</table>

*Note: The menu bar changes when you open a Physical Database Layout window.*

Three Panes

The Physical Database Layout window contains three panes: the Segments pane, the Statistics pane, and the Instances pane. Each pane displays a different type of information about the ObjectStore database you are viewing with Inspector.

Changing pane dimensions

You can change the dimensions of the panes in the Physical Database Layout window view by dragging the borders that separate them.

To change pane dimensions:

1. Place the pointer on the border of the pane you want to resize.
   - The pointer changes shape when it is placed on the pane border.
2. When the pointer changes shape, drag the border to change the size of the pane.
Filling the Panes

The Segments pane and the Statistics pane are populated by default when you first open the Physical Database Layout window. (The Statistics pane displays information for all user segments in the database.)

To fill the Instances pane, click Refresh | Instances on the menu bar.

*Alternative:* Click the Refresh tool at the top of the Instances pane.

Refreshing the Panes

You need to refresh the Statistics and Instances panes of the Physical Database Layout window any time you change the focus of the information currently displayed in the window.

You do not have to refresh the Segments pane — it always shows all the segments for a given database.

**Examples:**

- Select a new segment in the Segments pane
- Change the page range of the current segment
- Change the selected class in the Statistics pane

**How to tell if a pane needs to be refreshed**

Inspector provides visual cues that can remind you that the Statistics pane or Instances pane needs to be refreshed.

- The Refresh tool in the pane title bar becomes active.
- The Pane title bar is grayed out.

**How to refresh**

The following table summarizes how to refresh the Statistics and Instances panes.

*Note:* The speed with which the information in the Physical Database Layout window is refreshed depends on a number of factors, including the database and segment size, the number of pages you have selected, and the information you are displaying. Inspector displays a dialog box that allows you to cancel a refresh operation once it has started.

<table>
<thead>
<tr>
<th>Pane</th>
<th>How to Refresh</th>
<th>Alternatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistics</td>
<td>Click Refresh</td>
<td>Statistics on the menu bar.</td>
</tr>
<tr>
<td>Instances</td>
<td>Click Refresh</td>
<td>Instances on the menu bar.</td>
</tr>
</tbody>
</table>
Window Options

The Physical Layout page of the Options dialog box has options for default display values.

For more information

- See Statistics Options on page 128 to learn about options that affect the Statistics pane.
- See Instance Display Options on page 83 to learn about options that affect the Instances pane.
- See Inspector Options on page 27 to learn how options work and how to set them.

Segments

The Segments pane displays a list of all the segments associated with the current database.

Segment 0, the first segment displayed, is reserved for ObjectStore. The other segments displayed in the pane are user segments.
Purpose of the Segments Pane

The Segments pane acts as the main filter for the other panes in the Physical Database Layout window. In other words, the Statistics pane and the Instances pane display information based on the currently selected segment and pages.

What Segments Are Selected?

By default, all user-defined segments in the database are selected. (Segments reserved for ObjectStore are not selected.)

How to select a segment

To select a segment, click it.

To select multiple segments,

- Use Shift + select to select contiguous segments.
- Use Ctrl + select to select individual segments.

To select all the segments in the database, click Segments | Select All on the menu bar, or click the Select Whole Database button on the Segments pane.

Tip: You need to refresh the Statistics and Instances panes any time you change the segment or page range. See Refreshing the Panes on page 123 for more information.

Segment Pages

The field to the right of the segment name shows

- The total number of pages in the segment
- The number of currently selected pages

By default, Inspector selects all the pages in a segment. You can use the slider bars to focus on a given block of contiguous pages. You can select different blocks of pages for different segments.

To select a block of contiguous pages:

1. Drag the first slider to identify the start of the block.
   
   Tip: When you drag a slider, Inspector displays a pop-up that identifies the current page.

2. Stop dragging when the pop-up displays the page number on which you want to define the start of the block.

3. Repeat step 1 and step 2 with the other slider to define the end of the block.

4. Refresh the Statistics and Instances panes.

   See Refreshing the Panes on page 123 if you need help with this step.
The Segment Properties Dialog Box

The Segment Properties dialog box displays general, free space, and current page information.

- General page — Displays segment, page, and cluster number.
  
  Note: If you have selected multiple segments, the Segment Number and Comment fields are not displayed. Other fields display combined information.

- Space page — Displays a pie chart that shows the relative amounts of allocated and free space on the segment. (This page is not available in UNIX.)
  
  Note: Information is combined if you have selected multiple segments.

- Segment Pages page — Shows the range of currently selected pages in the segment. You can change the selection either by using the slider bars or by changing the page numbers in the Start and End fields.

  Note: This page is not available if you have selected multiple segments.

To open the Segment Properties dialog box:

1. Select the segments you want to examine.
2. Click Segment | Properties from the menu bar.

Tip: To review the properties of individual segments one after another, click the pushpin button. When you select a new segment from the Segments pane, the Segment Properties dialog box remains open. Existing information is replaced with information for the segment you just selected.
Segment Options

The Physical Layout page of the Options dialog box contains a display option for the Segment Properties dialog box. The Always Compute and Show Free Space option determines whether or not Inspector updates the space information displayed on the General and Space pages of the Segment properties dialog box.

To learn more about options and how to set them, see Inspector Options on page 27.

Statistics

The Statistics pane displays class information for the currently selected segments and pages in the Segments pane.

What Information Is Displayed

The Statistics pane lists all the classes for the currently selected segments and pages in the Segments pane. By default, the classes are sorted by category and are displayed in this order:

- User-defined
- ObjectStore
- ObjectStore arrays
- C++ primitive types
- C++ primitive type arrays

The number of instances for each class is displayed by default. You can change sort and display options in the Statistics pane, and you can set new defaults in the Options dialog box. See Statistics Options on page 128 for more information.
Statistics Options

The Physical Layout page of the Options dialog box contains sorting and display options for the Statistics pane.

- **Display** — You can display the quantity (number of instances) or size (number of bytes) associated with a class. You can also show the percent of segment memory allocated to the selected class.
  
  The default is to show quantity only.

- **Sort** — You can sort the classes by category (user-defined, ObjectStore, and so on) or by quantity. You can also use an ascending sort order.

  The default is to sort the list by category in a descending order.

To learn more about options and how to set them, see Inspector Options on page 27.

Instances

The Instances pane displays instance information for the currently selected classes in the Statistics pane.

What Information Is Displayed

The Instances pane lists the instances for the currently selected classes in the Statistics pane. By default, Inspector displays the following for each instance:

- Segment number
- Page number
- Offset

If you want, you can also display the size in bytes occupied by the instance. You can change display options and set new ones in the Options dialog box. See Instance Display Options on page 83 for more information.

Instance format

The Instance Format dialog box controls how an instance appears in the Instances pane (and elsewhere in Inspector). To learn more about changing an instance’s appearance, see Customizing the Instance Display on page 79.
Filling the Instances Pane

To fill the Instances pane:

1. Select the class or classes you want in the Statistics pane. (See Selecting Classes on page 129 if you need help with this step.)

2. Click Refresh | Instances on the menu bar.
   
   Alternative: Click the Refresh tool at the top of the Instances pane.

Selecting Classes

You need to select one or more classes before you can fill the Instances pane. Individual classes are not selected by default.

How to select classes

To select a single class, click it.

To select multiple classes:

• Use Shift + select to select contiguous classes.
• Use Ctrl + select to select individual classes.

Tip: You need to refresh the Instances pane any time you select a class. See Refreshing the Panes on page 123 for more information.

Instances Options

The Physical Layout page of the Options dialog box contains sorting and display options for the Instances pane. You can show or hide the following for each instance.

• Segment Number — The number of the segment on which the instance is stored
  The default is to show the segment number.

• Page Number — The page number within the segment on which the instance is stored
  The default is to hide the page number.

• Object Size — The size in bytes of the instance
  The default is to hide the size.

To learn more about options and how to set them, see Inspector Options on page 27.

Working with Free Space

Inspector provides two ways to help you work with free space in an ObjectStore database. You can

• Retrieve space and free space information from the segment
• Calculate free space
Retrieving Space and Free Space Information

Inspector retrieves space and free space information from segment attributes and displays this information on the Space page of the Segment Properties dialog box. (This page is not available in UNIX.)

The amount of free space shown includes blocks of free space interspersed among other objects in the segment, as well as any free space at the end of the segment.

This information is always available to Inspector; because it does not need to be calculated, it can be obtained quickly.

For more information: To learn more about the Segment Properties dialog box, see The Segment Properties Dialog Box on page 126.

Calculating Free Space

Inspector calculates free space by default. The resulting information appears on the Statistics and Instances panes of the Database Physical Layout window.

Inspector calculates segment free space by scanning each segment. The amount of free space includes only those blocks of free space that are interspersed among other objects in the segment. Any trailing free space at the end of the block is not considered.

Because the free space is calculated by an active scan of a segment, the time it takes to calculate free space can vary widely from segment to segment, and from database to database.

For more information: To learn more about the Database Physical Layout window, see Layout Window on page 122.

The option on the Segment menu for calculating free space is on by default.

To change the option for calculating free space, click Segment | Compute Free Space on the menu bar.

Tip: If the option is off and you turn it on again, you need to refresh the Statistics and Instances panes in order to see the calculated free space information. See Refreshing the Panes on page 123 for more information.
Tools for Debugging

In order to debug applications and databases, it is useful to be able to locate the object within a database. Inspector provides a tool that allows you to locate an instance based on either of its

- Address
- Segment and offset

Tip: Consider using segment and offset to locate an instance. Using an instance’s hexadecimal address might not always provide correct results. For example, the same object in two ObjectStore applications might have the same hexadecimal address; but those objects will always have unique segment and address information.

How to Locate an Instance

To locate an instance in the Instances pane of the Physical Database Layout window:

1. Click Segment | Go To Address on the menu bar.

   The Go To dialog box appears.

2. Enter the instance’s address, in hexadecimal format, or the instance’s segment and offset.

3. Click OK.

   The instance you are trying to locate appears at the top of the Instances pane.
Appendix A
Printing

One of the ways Inspector helps you communicate information about an ObjectStore database is by giving you the ability to print:

- Schema diagrams
- Collection grids
- Navigation trees

This chapter describes how to use Inspector’s printing features and covers the following topics:

- Printing Schema Diagrams 134
- Printing Collection Grids 135
- Printing Navigation Trees 138
Printing Schema Diagrams

This section describes how to print schema diagrams.

Before You Begin

Before printing a schema diagram, you might want to

- Change the layout
- Change the notation
- Create a simplified data view

See Chapter 4, Schema Diagrams, on page 55 for more information.

Controlling Printed Output

There are several features that help you control the printed output of a schema diagram.

### Print Preview

Print Preview (File | Print Preview) shows you how the classes in your schema diagram will be printed relative to a single page. Note that the zoom level of the schema diagram itself does not affect print size.

### Print on Multiple Pages

Print on Multiple Pages (File | Print on Multiple Pages) lets you change the number of pages (horizontal and vertical) over which your schema diagram will be printed. Generally speaking, the more pages you use, the larger the classes on the printed page will be. The default is 1 x 1.

Tip: If you are printing a schema diagram for presentation purposes, consider increasing the number of pages you use to print the diagram.

How to Print a Schema Diagram

Use this procedure to print a schema diagram if you want to preview what you print:

1. Make any changes you require to the diagram itself. (See Chapter 4, Schema Diagrams, on page 55 for more information.)
2. Select File | Print Preview from the menu bar.
3. Will the printed diagram be large enough for your purposes?
   - If yes, click Print.
   - The Print dialog box appears. Go to step 5.
If no, click Close and select File | Print on Multiple Pages from the menu bar and increase the number of pages over which to print the diagram.

4 Select File | Print from the menu bar.
   The Print dialog box appears.

5 Select the print options you want.

6 Click OK to print the schema diagram.

### Shortcut procedure

Use this procedure if you do not want to preview what you print:

1 Select File | Print from the menu bar.

   *Alternative:* Click the Print tool on the toolbar.

   The Print dialog box appears.

2 Select the print options you want.

3 Click OK to print the schema diagram.

## Printing Collection Grids

This section describes how to print collection grids.

### Before You Begin

Before printing a collection grid, you might want to make some of the following modifications, which affect the appearance of the grid both in Inspector and on the printed page.

You can

- Modify the instance format
- Pick a different number format
- Change the font
- Change the alignment of data within a cell
- Change the placement, style, and color of cell borders
- Change the color and pattern of individual cells
- Use the Chart Wizard to add a chart to the grid
- Annotate the grid

*For more information:* See Chapter 6, Classes and Instances, on page 73 for information on modifying the instance format. See Chapter 5, Collection Grids and Lists, on page 59 to learn about customizing collection grids.
Page Setup Options

Page setup options differ from those listed above in that they affect only the printed page, and not the grid display itself.

You open the Page Setup dialog box by selecting Grid | Print Page Setup from the menu bar.

The following sections describe the page setup options.

Header/Footer

You use the Header and Footer fields to define the text you want to appear in the headers and footers of the printed page. By default, the header prints the class name and page count in the following format:

class name #/#

The footer prints the page number, as follows:

Page #

The following table describes the formatting codes you can use in headers and footers. These codes are case insensitive.

<table>
<thead>
<tr>
<th>Format Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;l</td>
<td>Left-aligns the characters that follow</td>
</tr>
<tr>
<td>&amp;c</td>
<td>Centers the characters that follow</td>
</tr>
<tr>
<td>&amp;r</td>
<td>Right-aligns the characters that follow</td>
</tr>
<tr>
<td>&amp;d</td>
<td>Current date</td>
</tr>
<tr>
<td>&amp;t</td>
<td>Current time</td>
</tr>
<tr>
<td>&amp;a</td>
<td>Current grid, that is, current instance</td>
</tr>
<tr>
<td>&amp;p</td>
<td>Page number</td>
</tr>
<tr>
<td>&amp;p+number</td>
<td>Page number plus the number you specify</td>
</tr>
<tr>
<td>&amp;p-number</td>
<td>Page number minus the number you specify</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>Ampersand</td>
</tr>
<tr>
<td>&amp;n</td>
<td>Total number of pages in the document</td>
</tr>
</tbody>
</table>
In addition, you can use the following codes to control character formatting. They must appear before the codes in the preceding table, or they are ignored.

<table>
<thead>
<tr>
<th>Format Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;b</td>
<td>Bold</td>
</tr>
<tr>
<td>&amp;i</td>
<td>Italic</td>
</tr>
<tr>
<td>&amp;u</td>
<td>Underlines the header</td>
</tr>
<tr>
<td>&amp;s</td>
<td>Strikes through the header</td>
</tr>
<tr>
<td>&amp;&quot;fontname&quot;</td>
<td>Font</td>
</tr>
<tr>
<td>&amp;nn</td>
<td>Font size (must be two digits)</td>
</tr>
</tbody>
</table>

Center

You use the check boxes in the Center group box to center the grid on the page. You can center the grid horizontally, vertically, or both.

Margins

You use the fields in the Margins group box to specify the area you want to leave for the top/bottom, left/right, and header/footer margins. The default unit is inches, but you can change it using the Units drop-down list box.

Print Options

You use the Print Options group box to indicate whether or not you want the following elements to appear in the printed grid:

- Grid Lines
- Black and White
  
  Note: This field overrides any color formatting you have done in the grid and prints all information in black and white.
- Row Heading
- Column Heading

Page Order

You use the choices in the Page Order group box to indicate the page order in which you want to print a grid that spans multiple pages. Top To Bottom prints all the instances first; Left To Right prints all data members first.

Scale

You use the fields in the Scale group box to control the scale of the grid relative to the number of pages on which you choose to print it.

- Fit To Page scales the grid so that it fits on the number of pages you indicate in the Pages Wide and Pages High field.
- Pages Wide identifies the number of pages across which you want to print the grid.
- Pages High identifies the number of pages down which you want to print the grid.
- Scale controls the size of the grid relative to the page. If you need the grid elements to print larger, increase the Scale value.
How to Print a Collection Grid

To print a collection grid:

1. Make any changes you require to the instance format or the grid page setup.
2. Select Grid | Print from the menu bar.
   The Print dialog box appears.
3. Select the print options you want.
4. Click OK to print the instance grid.

Printing Navigation Trees

This section describes how to print navigation trees, which are displayed in the Navigation window.

Before You Begin

Before printing a navigation tree, you might want to change the layout. See “Formatting navigation trees” on page 88 for more information.

Controlling Printed Output

There are several features that help you control the printed output of a navigation tree.

- **Print Preview**
  Print Preview (File | Print Preview) shows you how the navigation tree will be printed relative to a single page.

- **Print on Multiple Pages**
  Print on Multiple Pages (File | Print on Multiple Pages) lets you change the number of pages (horizontal and vertical) over which the navigation tree will be printed. Generally speaking, the more pages you use, the larger the symbols on the printed page will be. The default is 1 x 1.

  *Tip:* If you are printing a navigation tree for presentation purposes, consider increasing the number of pages.
How to Print a Navigation Tree

**Previewing what you print**

Use this procedure to print a navigation tree if you want to preview what you print:

1. Change the navigation tree layout if desired. (See “Formatting navigation trees” on page 88 for more information.)
2. Select File | Print Preview from the menu bar.
3. Will the printed navigation tree be large enough for your purposes?
   - If yes, click Print.
   
   The Print dialog box appears. Go to step 5.
   
   If no, click Close and select File | Print on Multiple Pages from the menu bar and increase the number of pages over which to print the navigation tree.
4. Select File | Print from the menu bar.
   
   The Print dialog box appears.
5. Select the print options you want.
6. Click OK to print the navigation tree.

**Shortcut procedure**

Use this procedure if you do not want to preview what you print:

1. Select File | Print from the menu bar.
   
   Alternative: Click the Print tool on the toolbar.
   
   The Print dialog box appears.
2. Select the print options you want.
3. Click OK to print the navigation tree.
Appendix B
Using Inspector as an OLE Server

This chapter describes how to use Inspector as an OLE (object linking and embedding) Server so that you can insert Inspector objects in other applications.

This chapter covers the following topics:
Overview of Inspector as an OLE Server on page 141
Modifying Inspector Objects on page 142
Inserting Inspector Objects on page 142
In-Place Editing on page 144

Overview of Inspector as an OLE Server

Inspector is enabled as an OLE server. This means that you can insert objects from Inspector into OLE container applications, including Microsoft Word, Excel, and PowerPoint.
Purpose

The ability to use Inspector as an OLE server means that you can provide end users with access to information from an ObjectStore database — in a document format the user is familiar with, and without the need to do any programming. Once the information is in the container document, the user can take advantage of the container application's features for reporting, calculation, and presentation purposes.

As an OLE server, Inspector can insert database schema diagrams in the container document. Data views, Instance windows, and the Physical Database Layout window are not considered Inspector objects for OLE purposes.

Schema diagrams are inserted as is. All layout, notation, and class/relationship information is preserved. Note that you cannot apply abstraction functions to schema diagrams that are used as Inspector objects.

Modifying Inspector Objects

You can modify schema diagrams and instance panes before you insert them or at the time you insert them.

If you choose to modify an Inspector object at the time you insert it into the container application, you will be prompted to save your changes when you exit from Inspector.

For more information

See Chapter 4, Schema Diagrams, on page 55 to learn more about working with diagrams.

Inserting Inspector Objects

Prerequisite

The database from which you want to insert objects cannot be open in Inspector.

How to insert an Inspector object

To insert an Inspector object:

1. Open the container application.
2. Click Insert | Object on the menu bar.
The Object dialog box appears.

3 Select ObjectStore Inspector Schema Diagram from the Object Type list box.

4 Click OK.

   The Inspector Open dialog box appears.

5 Select the database containing the information you want to insert in the container application and click OK.

   The Inspector desktop appears; the title bars of both the main window and the main database view window display the name of the container document.

6 Optionally, modify the diagram. For example, you can change the schema diagram notation.

7 Click File -> Exit & Return on the menu bar.

   If you made any changes, Inspector displays a prompt asking if you want to save those changes. Click Yes to save the changes in Inspector; click No otherwise.

   The Inspector object (that is, the schema diagram) appears in the container document.
In-Place Editing

*In-place editing* is a feature that allows users to take advantage of Inspector editing functionality directly in the container document.

*Tip:* See Chapter 4, Schema Diagrams, on page 55 to learn how to work with diagrams.

How to perform in-place editing

To perform in-place editing:

1. Open the container document.
2. Select the Inspector object.
3. Click *Edit | OS Inspector Object | Edit* on the menu bar.

*Alternatives:* Click *OS Inspector Object | Edit* on the object’s content menu, or double-click the Inspector object.

4. The menu and toolbar associated with the Inspector object appear in the container document.

5. Make the changes you want.
6. Save the document as you would normally.
Appendix C
Working with Metaknowledge

Metaknowledge is data that describes the way ObjectStore database information is presented in Inspector. It is distinct from information contained within an ObjectStore database.

This chapter covers the following topics:
Overview of Metaknowledge on page 146
Importing Metaknowledge on page 146
Updating Metaknowledge on page 148
Ignoring Metaknowledge on page 148
Overview of Metaknowledge

Metaknowledge is information about the way ObjectStore database information is presented in Inspector. The metaknowledge associated with a database is updated when you do any of the following:

- Make certain changes in the main database view:
  - Modify the schema layout
  - Change the default notation
  - Populate the instance pane
- Create a data view
- Change the default instance format
- Modify the string format interpretation/display rules
- Create a user-defined method

Metaknowledge is saved as a file in one of three places:

- In the file system
  The metaknowledge is saved in the `<install>\db_gph` directory as a file named `<file_name>.gph`.

- With the ObjectStore database
  Tip: This choice is recommended so that the metaknowledge is always stored with the database: if the database is moved, its metaknowledge is moved with it. However, it requires that you have write access to the database.

- With the Inspector database
  The metaknowledge is saved in a dedicated root named `__ivitos_meta_knowledge`.

The default for saving metaknowledge is set on the Standard page of the Options dialog box. See Inspector Options on page 27 to learn how to set Inspector options.

Importing Metaknowledge

You can import metaknowledge from one database to another. This feature enables you to leverage metaknowledge across databases with similar schemas.

Tips

Before importing another database’s metaknowledge,

- Make sure that the schemas of the two databases are as close to identical as possible. Importing metaknowledge to a database with a different schema will probably not provide the results you expect.
- Importing metaknowledge merges the existing metaknowledge with the metaknowledge you import.
• Importing metaknowledge forces Inspector to close all open windows and dialog boxes, except for the main database view window.

• Save the existing metaknowledge using File | Save As. You can always reimport the current metaknowledge if the new one does not provide the results you were expecting.

How to import metaknowledge

To import metaknowledge:

1 In Inspector, open the database into which you want to import metaknowledge.

2 Click File | Import Metaknowledge on the menu bar.

The Open dialog box appears.

3 Select the database whose metaknowledge you want to import and click OK.

Inspector displays a dialog box asking you to confirm whether or not you want to import another database’s metaknowledge.

4 Click Yes to continue; otherwise, click No.

5 Inspector compares the schemas of the source and target databases. If they are different, it displays a message enabling you to cancel the operation.

6 Click Yes to continue; otherwise, click No.

7 If you choose to continue, Inspector displays a dialog box that identifies the differences between the current metaknowledge and the one you are importing.

8 Click OK to continue.
Updating Metaknowledge

You might want to update the metaknowledge if you believe the database schema has changed since you first opened it in Inspector — for example, it has a new class or new data members.

How to update metaknowledge

To update metaknowledge:

1. In Inspector, open the database whose metaknowledge you want to update.
2. Click File | Update Metaknowledge on the menu bar.
   Inspector displays a message that asks you to confirm that you want to update the database’s metaknowledge.
3. Click Yes to continue; otherwise, click No.
4. If you are continuing, Inspector first compares Inspector’s version of the schema with that of the actual database.
   If they are the same, Inspector displays a message notifying you that no update is required.
   If they are not the same, Inspector displays a message enabling you to cancel the operation.
5. Click Yes to continue; otherwise click No.

Ignoring Metaknowledge

When you open a database in Inspector, you can open it without the metaknowledge that was saved with it. You might want to do this when

- You want to start working with a database from scratch.
- The schema stored in Inspector’s metaknowledge is not the same as that of the actual database and you do not want to update them.
- You have set an option that requires the metaknowledge to be recalculated and you do not want to update the metaknowledge at this time.

How to ignore metaknowledge

To ignore metaknowledge, click the Ignore Metaknowledge check box in the Open dialog box.

Tip: When you ignore metaknowledge, Inspector displays the Review Incompatibility Errors dialog box. See Importing Metaknowledge on page 146.
Appendix D
Configuring Inspector

The ObjectStore Inspector Configuration Utility is a tool you use to make the application schema database associated with Inspector accessible to an ObjectStore server. This involves

• Placing the Inspector application schema under the control of an ObjectStore server
• Configuring Inspector to find the application schema there
• Ensuring that the ObjectStore server is running

The Configuration Utility can help you with these steps.

For more information, see

• Configuration Process on page 150
• Starting the Configuration Utility on page 151
• Evaluating the Installation Status on page 152
• About ObjectStore Servers on page 150

When to Use the Configuration Utility

Inspector is an ObjectStore application. Like all ObjectStore applications, its application schema must be accessible to an ObjectStore server. Typically, Inspector is installed on a workstation on which an ObjectStore server is also running, ensuring that its application schema is accessible to that ObjectStore server.

You might need to run the Configuration Utility if you have

• Installed Inspector on a workstation on which an ObjectStore server is not running
• Moved the Inspector installation

Using ossetasp Instead

You can use an ObjectStore utility, ossetasp, to configure Inspector to look for its application schema in the directory you specify. However, ossetasp does not provide the additional functionality or user interface of the Configuration Utility. Building ObjectStore C++ Applications for more information.
About the Application Schema File

The Inspector application schema is stored in a file named `vomsch71.adb`. By default, it is located in the `<installation>\OSI7.1\lib` directory, where `<installation>` is the directory in which you have installed Inspector.

About ObjectStore Servers

The ObjectStore server is a process that controls access to ObjectStore databases on a host. This includes:

- Storage and retrieval of persistent data
- Arbitration of concurrent access by multiple client applications
- Recovery of databases to a transaction-consistent state if any of the processes aborts or any host crashes, or in the event of network failure

The server also manages pages of data on behalf of clients running applications.

Rawfs Management

For the rawfs, if there is one on the host, the server manages the hierarchy of directories and maintains permission modes, creation dates, owners, and groups for each entry.

Usually, a server must be running before any ObjectStore application can access databases on the host. (A locator file allows access to databases residing on a host that is not running a server.)

An application can use databases that are stored on different hosts and managed by different servers. A server can serve clients on any number of hosts.

Multiple Servers on a Host

A host can run one server of a given ObjectStore release. You can run two ObjectStore servers on the same host if they are different versions of ObjectStore. For example, you can run a Release 7 and Release 6 server on the same host. Start them on different ports and use the ports file to let clients know which one to contact.

A network can have a number of servers.

Configuration Process

The configuration process consists of the following steps:

2. Evaluate the installation status as shown in the Application Schema Configuration Utility main window.
The color of the light in the traffic signal indicates what to do next:

<table>
<thead>
<tr>
<th>If the Light Is</th>
<th>Then You</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>Can quit the Inspector Configuration Utility and run Inspector</td>
</tr>
<tr>
<td>Yellow</td>
<td>Need to change the Inspector configuration</td>
</tr>
<tr>
<td>Red</td>
<td>Need to change the Inspector configuration</td>
</tr>
</tbody>
</table>

For more information, see:
- Starting the Configuration Utility on page 151
- Evaluating the Installation Status on page 152
- Changing the Inspector Configuration on page 153

Starting the Configuration Utility

Prerequisite

You must exit ObjectStore Inspector before you start the Inspector Configuration Utility.

How to Start the Configuration Utility

To start the Inspector Configuration Utility, click ObjectStore Inspector 7.1 | Application Schema Configuration on the Start menu.

The Application Schema Configuration Utility main window appears.

See also Evaluating the Installation Status on page 152 and Configuration Process on page 150.
Evaluating the Installation Status

The Application Schema Configuration Utility main window provides

- Information about the current location of the Inspector application schema
- The status of the ObjectStore server
- The status of the application schema
- Functionality to help you address any problems with the Inspector configuration.

Configuration Status is Summarized

Inspector’s current configuration status is summarized by a traffic signal that appears in the center of the Application Schema Configuration Utility main window.

Green for Go
If Inspector is properly configured and ready to be run, the traffic signal in the center of the page shows a green light. This means that Inspector’s application schema is under the control of an ObjectStore server and the server is currently running. If the traffic signal shows green, you can quit the Configuration Utility and start Inspector.

Yellow for Caution
A yellow light indicates that while Inspector is configured to use an application schema under the control of an ObjectStore server that is running, the application schema file (vomsch71.adb) has not been copied to that location. In this case, you need to use the Configuration Utility to copy the Inspector application schema to the appropriate location.

Red for Stop
A red light indicates that Inspector is currently configured to use an application schema on a workstation where an ObjectStore server is not running and that the Inspector
application schema is not in the expected location. In this case, you need to use the Configuration Utility to change the Inspector configuration by

- Choosing a new ObjectStore server
- Copying the Inspector application schema to that location
- Updating the Inspector executable to look for the application schema in the new location

Status Summary Supplements

The status summary provided by the traffic light is supplemented by comments in the Application Schema group box at the bottom of the Application Schema Configuration Utility main window. These comments can help you pinpoint the specific problem with the current Inspector configuration.

There are separate fields for server status and application schema status.

Server Status
The server is either running or not running.

Application Schema Status
The application schema is either reachable or not reachable.

Together, these fields provide an explanation for the configuration status summary represented by the traffic signal.

<table>
<thead>
<tr>
<th>If the Server Status Is</th>
<th>And the Application Schema Is</th>
<th>Then the Light Is</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running</td>
<td>Reachable</td>
<td>Green</td>
</tr>
<tr>
<td>Running</td>
<td>Not reachable</td>
<td>Yellow</td>
</tr>
<tr>
<td>Not running</td>
<td>Not reachable</td>
<td>Red</td>
</tr>
</tbody>
</table>

For more information see

- Changing the Inspector Configuration on page 153
- About ObjectStore Servers on page 150

Changing the Inspector Configuration

Changing the Inspector configuration involves

- Placing the Inspector application schema under the control of an ObjectStore server
- Updating Inspector to make it aware of the application schema location

You accomplish both of these tasks using the Choose an ObjectStore Server dialog box.
Once you select the ObjectStore server and click the Choose button, the Configuration Utility

• Copies the Inspector application schema to that location
• Runs the ObjectStore ossetasp utility to update the Inspector executable

When the Configuration Utility is finished, the Choose an ObjectStore Server dialog box closes and you are returned to the main window. The installation status summary is updated based on the changes you made to the Inspector configuration.

Choosing a Server

You can copy the application schema to any of the following locations, provided an ObjectStore server is running there:

• Local network drive
• Remote server
• Rawfs database

How to Change the Inspector Configuration

To change the Inspector configuration:

1. Click the Change button on the Application Schema Configuration Utility main window.
   The Choose an ObjectStore Server dialog box appears.
   The list box is filled with the workstations available to you. The time it takes to populate the list box depends on the size and complexity of your network.

2. Tip: The workstation symbol appears with a question mark on it while the Configuration Utility checks your network. 📘
If a workstation is determined to be an ObjectStore server, the ObjectStore logo appears in place of the question mark.

3 Select the workstation and directory to which you want to copy the application schema. Remember that this must be an ObjectStore server and can be any one of the following:
- Local network drive
- Remote server
- Rawfs database

4 Click the Choose button.

Copying the Schema to a Local Drive

To copy the application schema to a file system on the network:

1 Double-click on a logical unit listed under one of the workstations displayed in the Choose an ObjectStore Server dialog box.

Tip: If you select a remote workstation which has no logical units connected to its file system, the program allows you to build a new connection.

2 Select the directory to which you want to copy the application schema and click the Choose button.

For More information see:
- Error Conditions on page 157
- About ObjectStore Servers on page 150
Copying the Schema to a Remote Server

Note: In order to perform this procedure, an ObjectStore server must be running on both the local computer and on the computer on which the selected directory is located.

To copy the application schema to an NFS directory:

1. Click the NFS Path button.

   The Choose Remote Server dialog box appears.

2. Enter any valid path using NFS format: `<host-name-or-address>:<path>` in the entry field. The following are examples of valid NFS paths:
   
   - `ciro:/usr/local`
   - `150.145.2.2:/opt`

3. Click the OK button.

   The Configuration Utility attempts to copy the application schema to the specified directory and configure Inspector to access it there.

For more information, see:

- Error Conditions on page 157
Copying the Schema to a rawfs Database

A rawfs (raw file storage) database is an ObjectStore database that resides in a special directory managed by the ObjectStore Directory Manager.

The Directory Manager process must be running to open and access a rawfs database.

Note: In order to perform this procedure, an ObjectStore server must be running on both the local and remote computers.

To copy the application schema to an ObjectStore rawfs database:

1. Double-click on a rawfs icon.
2. Select the directory to which you want to copy the application schema.
3. Click the Choose button.

The Inspector Configuration Utility attempts to copy the application schema to the specified rawfs directory and configure Inspector to access it there.

For more information, see:

• Error Conditions on page 157
• About ObjectStore Servers on page 150

Error Conditions

This topic covers possible error conditions you might encounter while using the Configuration Utility.
Error Copying to a Network Drive

The following message appears when the Configuration Utility is not able to copy the Inspector application schema to a network drive:

Possible Causes

- You do not have the appropriate rights to copy the file to the target directory.
- The network is down.

Error Copying to a rawfs Database

The following message appears when the Configuration Utility is not able to copy the application schema to a rawfs database:

Possible Causes

- You do not have the appropriate rights to copy the file to the target rawfs database.
- The ObjectStore server is not running on the local (or remote) machine.

Error Running ossetasp

The following message appears when the Configuration Utility executes the ObjectStore ossetasp utility on the osivitos.dll Inspector library and the operation does not succeed.
Possible Causes

• ObjectStore Inspector is running, so ossetasp cannot modify the osivitos.dll Inspector library. In this case, close Inspector and run the Configuration Utility again.

• The `\bin` directory where the Inspector DLLs are installed (usually `\ODI_x.x\Osi.x.x\bin`, where `x.x` is the version number) is not accessible.

For more information, see Changing the Inspector Configuration on page 153.
Index

A
abstraction functions
  how to apply 58
introduction 39
types of 58
Alignment dialog box 66
arguments, adding
  user-defined methods 109

B
Borders dialog box 67

C
C++ classes
  relationships between 91
C++ pointers
  editing 97
C++ references
  editing 97
calculating free space 130
carsdemo database, opening 20
character strings, editing 95
Choose Root dialog box 117
Class Details dialog box 78
class layout, changing 56
classes, selecting
  Physical Database Layout window Instances pane 129
collection grids
  annotating the grid 69
  changing alignment of cell data 66
  changing cell color and pattern 68
  changing cell dimensions 66
  changing font 68
collection lists
  compared 60
  main differences between 61
  creating formulas in 70
customizing 65, 84
  customizing the grid border 67
deleting grid template 71
eexample 60
exporting 72
exporting in XML 72
features 65, 84
grid template
  applying 71
  deleting 71
  saving 70
loading 62
options 64
populating 61
printing
  before beginning 135
  how to 138
refreshing 62
saving
  grid template 70
  saving modifications to 70
selected collection
selecting cells 66
collection order
  defining 54
collections
  exporting 26
  ordering 53
  refreshing 62
creating
  data views 43
  formulas in collection grids 70
multiple database views 21
a root 116
da root and defining its value 117
custom database view
creating explicitly 36
creating implicitly 37
deleting 38
opening 38
saving 37
customizing collection grids
procedure 65
ways to do 84

data
modifying 24
data members
editing
considerations 93
prerequisite 94
using the Edit Data Member dialog box 93
identifying for display 80
renaming 81
reordering a list 81
undoing edits 94
data views
compared to database views 30
creating 43
definition 42
deleting 45
introduction 22
opening 44
saving 44
database information
sharing 26
Database Roots pane
database view window 33
database view window
Database Roots pane 33
instance pane 35
schema pane 34
three panes 30
toolbar 32
window name 32
database views
compared to data views 30
compared to the instance pane 76
creating explicitly 36
creating implicitly 37
custom 31
definition 30
deleting custom 38
instance pane 74
main 31
modifying 39
multiple 21, 24
opening custom 38
saving custom 37
saving main 37
databases
opening 19
working with free space 129
debugging 131
Define New Root dialog box 116
defining
a collection order 54
filters
manually 47
using Inspector 48
parameterized constraints 50
deleting
arguments
user-defined methods 110
custom database views 38
data views 45
grid template 71
navigation trees 89
a root 118
Deleting a Root dialog box 119
desktop, Inspector 20
diagram notation 55
diagram zoom level
changing 56
diagram, schema
See schema diagrams
dialog box

Choose Root 117
Alignment 66
Borders 67
Class Details 78
Define New Root 116
Deleting a Root 119
Edit Data Member 25, 93, 96
Fill Query Parameters 53
Find String 63
Font 68
Go To 131
Inspector Open 143
Instance Format 79
Options 27
Page Setup 136
Parameter Definition 50
Path Layout 88
Pattern 68
Possible Relationship Discovered 91
Print Size 134
Save Database View 37
Segment Properties 126, 130
Toolbar Settings 33

E
Edit Data Member dialog box 25, 93, 96
editing
  C++ pointers 97
  C++ references 97
  character strings 95
data members using the Edit Data Member dialog box 93
  enumerated types 95
  in-place 144
  memory and memory pointers 96
  relationships 96
enumerated types
  editing 95
exporting
  collection grids 72
  collection grids in XML 72
  collections 26

F
Fill Query Parameters dialog box 53
filling the panes
  Physical Database Layout window 123
filters
  constraints 45
  defining
    manually 47
    using Inspector 48
definition 45
displaying the query 49
reapplying 47
turning on and off 47
when applied 47
Find String dialog box 63
Font dialog box 68
formulas
  creating in collection grids 70
free space
  calculating 130
  retrieving information 130

G
Go To dialog box 131
grid border
  customizing 67
grid template
  applying 71
  deleting 71
  saving 70

H
hiding relationships 57

I
icon
  associating with a class 82
  importing 83
importing an icon 83
in-place editing 144
Inspector
  desktop 20
  introduction 18
  objects 142
  starting 20
Inspector Open dialog box 143
Instance Format dialog box 79
  Class Icon sheet 82
instance grids and lists
  printing
    page setup options 136
instance pane
  compared to database views 76
database view window 35, 74
Instance window 22, 74
Instance: Physical Layout dialog property box 77
instances
  locating 131
  navigating
    automatically 86
    manually 86
Instances pane
   Physical Database Layout window  128
invoking user-defined methods
   delete  113
   extent  113
   general  111
   read  112
   update and create  112

J
Java classes
   relationships between  92

L
loading
   collection grids  62
locating an instance  131

M
main database view
   saving  37
main database view window  21
memory and memory pointers
   editing  96
metaknowledge
   definition  24
   working with  146
modifications
   to a collection grid, saving  70
modifying database views  39
multiple database views
   creating  21, 24

N
navigating instances
   automatically  86
   manually  86
navigation  85
navigation trees
   deleting  89
   formatting  88
   opening  89
   printing  138
   saving  88
Navigation window  23, 87

O
objects
   inserting Inspector  142
   modifying Inspector  142
ObjectStore data information
   modifying  24
OLE server
   introduction  26
   using Inspector as  141
opening
   custom database views  38
   data views  44
   databases  19
   navigation trees  89
   the carsdemo database  20
options
   setting  27
   when they take effect  28
Options dialog box  27
   Instances page  83
   Navigation page  90
Physical Layout page
   instances options  129
   segment options  127
   statistics options  128
   window options  124
ordering a collection  53

P
Page Setup dialog box  136
pane dimensions
   changing  35
Parameter Definition dialog box  50
parameterized constraints
   defining  50
   integer types  52
   string types  50
Path Layout dialog box  88
Pattern dialog box  68
Physical Database Layout window
   display  23, 78
   filling the panes  123
   Instances pane  128
   opening  122
   refreshing the panes  123
   Segments pane  124
   Statistics pane  127
pointers to memory
  editing 96
Possible Relationship Discovered dialog box 91
Print Size dialog box 134
printing
  collection grids
    before beginning 135
    how to 138
  instance grids and lists
    page setup options 136
  navigation trees 138
  schema diagrams 134
  what you can print 26
property dialog box
  Instance: Physical Layout 77

R

refreshing
  collection grids 62
  panes 123
Register User-Defined Methods window 108
registering user-defined methods 108
relationship routes
  changing the shape of 57
relationships
  between C++ classes 91
  between Java classes 92
  editing 96
  hiding 57
renaming arguments
  user-defined methods 110
required signature
  user-defined methods 105
retrieving free space information 130
root value
  choosing 116
roots
  creating 116
  creating and defining the root value 117
  deleting 118
  redefining 117

S

Save Database View dialog box 37
saving
  collection grids, modifications to 70
  custom database views 37
data views 44
  grid template 70
  main database view 37
  navigation trees 88
schema diagrams
  altering the contents of 58
  changing the appearance of 56
  changing class layout 56
  changing relationship routes 57
  changing the contents 34
  changing the layout 34
  changing zoom level 56
  hiding relationships 57
  layout 39
  printing 134
schema pane
  database view window 34
Segment Properties dialog box 126, 130
Segments pane
  Physical Database Layout window 124
setting options 27
starting Inspector 20
Statistics pane
  Physical Database Layout window 127

T

toolbar
  database view window 32
Toolbar Settings dialog box 33

U

unregistering user-defined methods 110
user-defined 113
user-defined methods
  adding arguments 109
  definition 24
  deleting arguments 110
  invoking
dele te 113
  general 111
  read 112
  update and create 112
  prerequisites 104
  process 104
  purpose 105
registering 108
renaming arguments 110
required signature
  create 105
delete 107
extent 107
read 106
update 106
unregistering 110

window
  database view 30
  Instance 22, 74
  main database view 21
  Navigation 23, 87
  Physical Database Layout 23, 78, 122
  Register User-Defined Methods 108
window name
  database view window 32