ORACLE

Oracle®Transall

Oracle Transall User's Guide

Version 12m R1(12.4.0)

Part number: E57338-01

January 2015

Copyright © 2015, Oracle and/or its affiliates. All rights reservedThis software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Where an Oracle offering includes third party content or software, we may be required to include related notices. For information on third party notices and the software and related documentation in connection with which they need to be included, please contact the attorney from the Development and Strategic Initiatives Legal Group that supports the development team for the Oracle offering. Contact information can be found on the Attorney Contact Chart.

The information contained in this document is for informational sharing purposes only and should be considered in your capacity as a customer advisory board member or pursuant to your beta trial agreement only. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle Software License and Service Agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

THIRD PARTY SOFTWARE NOTICES

This product includes software developed by Apache Software Foundation (http://www.apache.org/).

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright © 2000-2009 The Apache Software Foundation. All rights reserved.

This product includes software distributed via the Berkeley Software Distribution (BSD) and licensed for binary distribution under the Generic BSD license.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright © 2009, Berkeley Software Distribution (BSD)

This product includes software developed by the JDOM Project (http://www.jdom.org/).

THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE JDOM AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (C) 2000-2004 Jason Hunter & Brett McLaughlin. All rights reserved.

This product includes software developed by the Massachusetts Institute of Technology (MIT).

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.

Copyright © 2009 MIT

This product includes software developed by Jean-loup Gailly and Mark Adler. This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Copyright (c) 1995-2005 Jean-loup Gailly and Mark Adler

This software is based in part on the work of the Independent JPEG Group (http://www.ijg.org/).

This product includes software developed by the Dojo Foundation (http://dojotoolkit.org).

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (c) 2005-2009, The Dojo Foundation. All rights reserved.

This product includes software developed by W3C.

Copyright © 2009 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved. (http://www.w3.org/Consortium/Legal/)

This product includes software developed by Mathew R. Miller (http://www.bluecreststudios.com).

Copyright (c) 1999-2002 ComputerSmarts. All rights reserved.

This product includes software developed by Shaun Wilde and distributed via Code Project Open License (http://www.codeproject.com).

THIS WORK IS PROVIDED "AS IS", "WHERE IS" AND "AS AVAILABLE", WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES OR CONDITIONS OR GUARANTEES. YOU, THE USER, ASSUME ALL RISK IN ITS USE, INCLUDING COPYRIGHT INFRINGEMENT, PATENT INFRINGEMENT, SUITABILITY, ETC. AUTHOR EXPRESSLY DISCLAIMS ALL EXPRESS, IMPLIED OR STATUTORY WARRANTIES OR CONDITIONS, INCLUDING WITHOUT LIMITATION, WARRANTIES OR CONDITIONS OF MERCHANTABILITY, MERCHANTABLE QUALITY OR FITNESS FOR A PARTICULAR PURPOSE, OR ANY WARRANTY OF TITLE OR NON-INFRINGEMENT, OR THAT THE WORK (OR ANY PORTION THEREOF) IS CORRECT, USEFUL, BUG-FREE OR FREE OF VIRUSES. YOU MUST PASS THIS DISCLAIMER ON WHENEVER YOU DISTRIBUTE THE WORK OR DERIVATIVE WORKS.

This product includes software developed by Chris Maunder and distributed via Code Project Open License (http://www.codeproject.com).

THIS WORK IS PROVIDED "AS IS", "WHERE IS" AND "AS AVAILABLE", WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES OR CONDITIONS OR GUARANTEES. YOU, THE USER, ASSUME ALL RISK IN ITS USE, INCLUDING COPYRIGHT INFRINGEMENT, PATENT INFRINGEMENT, SUITABILITY, ETC. AUTHOR EXPRESSLY DISCLAIMS ALL EXPRESS, IMPLIED OR STATUTORY WARRANTIES OR CONDITIONS, INCLUDING WITHOUT LIMITATION, WARRANTIES OR CONDITIONS OF MERCHANTABILITY, MERCHANTABLE QUALITY OR FITNESS FOR A PARTICULAR PURPOSE, OR ANY WARRANTY OF TITLE OR NON-INFRINGEMENT, OR THAT THE WORK (OR ANY PORTION THEREOF) IS CORRECT, USEFUL, BUG-FREE OR FREE OF VIRUSES. YOU MUST PASS THIS DISCLAIMER ON WHENEVER YOU DISTRIBUTE THE WORK OR DERIVATIVE WORKS.

This product includes software developed by PJ Arends and distributed via Code Project Open License (http://www.codeproject.com).

THIS WORK IS PROVIDED "AS IS", "WHERE IS" AND "AS AVAILABLE", WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES OR CONDITIONS OR GUARANTEES. YOU, THE USER, ASSUME ALL RISK IN ITS USE, INCLUDING COPYRIGHT INFRINGEMENT, PATENT INFRINGEMENT, SUITABILITY, ETC. AUTHOR EXPRESSLY DISCLAIMS ALL EXPRESS, IMPLIED OR STATUTORY WARRANTIES OR CONDITIONS, INCLUDING WITHOUT LIMITATION, WARRANTIES OR CONDITIONS OF MERCHANTABILITY, MERCHANTABLE QUALITY OR FITNESS FOR A PARTICULAR PURPOSE, OR ANY WARRANTY OF TITLE OR NON-INFRINGEMENT, OR THAT THE WORK (OR ANY PORTION THEREOF) IS CORRECT, USEFUL, BUG-FREE OR FREE OF VIRUSES. YOU MUST PASS THIS DISCLAIMER ON WHENEVER YOU DISTRIBUTE THE WORK OR DERIVATIVE WORKS.

This product includes software developed by Erwin Tratar. This source code and all accompanying material is copyright (c) 1998-1999 Erwin Tratar. All rights reserved.

THIS SOFTWARE IS PROVIDED "AS IS" WITHOUT EXPRESS OR IMPLIED WARRANTY. USE IT AT YOUR OWN RISK! THE AUTHOR ACCEPTS NO LIABILITY FOR ANY DAMAGE/LOSS OF BUSINESS THAT THIS PRODUCT MAY CAUSE.

This product includes software developed by Sam Leffler of Silicon Graphics.

THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

IN NO EVENT SHALL SAM LEFFLER OR SILICON GRAPHICS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE

Copyright (c) 1988-1997 Sam Leffler
Copyright (c) 1991-1997 Silicon Graphics, Inc.

This product includes software developed by Guy Eric Schalnat, Andreas Dilger, Glenn Randers-Pehrson (current maintainer), and others. (http://www.libpng.org)

The PNG Reference Library is supplied "AS IS". The Contributing Authors and Group 42, Inc. disclaim all warranties, expressed or implied, including, without limitation, the warranties of merchantability and of fitness for any purpose. The Contributing Authors and Group 42, Inc. assume no liability for direct, indirect, incidental, special, exemplary, or consequential damages, which may result from the use of the PNG Reference Library, even if advised of the possibility of such damage.

This product includes software components distributed by the Cryptix Foundation.

THIS SOFTWARE IS PROVIDED BY THE CRYPTIX FOUNDATION LIMITED AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE CRYPTIX FOUNDATION LIMITED OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

Copyright © 1995-2005 The Cryptix Foundation Limited. All rights reserved.

This product includes software components distributed by Sun Microsystems.

This software is provided "AS IS," without a warranty of any kind. ALLEXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANYIMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN AND ITS LICENSORS SHALL NOT BELIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Copyright (c) 1998 Sun Microsystems, Inc. All Rights Reserved.

This product includes software components distributed by Dennis M. Sosnoski.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright © 2003-2007 Dennis M. Sosnoski. All Rights Reserved

It also includes materials licensed under Apache 1.1 and the following XPP3 license

THIS SOFTWARE IS PROVIDED "AS IS"' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright © 2002 Extreme! Lab, Indiana University. All Rights Reserved

---

This product includes software components distributed by CodeProject. This software contains material that is © 1994-2005 The Ultimate Toolbox, all rights reserved.

This product includes software components distributed by Geir Landro.

Copyright © 2001-2003 Geir Landro (drop@destroydrop.com) JavaScript Tree - www.destroydrop.com/hjavascripts/tree/version 0.96

This product includes software components distributed by the Hypersonic SQL Group.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

Copyright © 1995-2000 by the Hypersonic SQL Group. All Rights Reserved

---

This product includes software components distributed by the International Business Machines Corporation and others.

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.

Copyright (c) 1995-2009 International Business Machines Corporation and others. All rights reserved.

This product includes software components distributed by the University of Coimbra.

University of Coimbra distributes this software in the hope that it will be useful but DISCLAIMS ALL WARRANTIES WITH REGARD TO IT, including all implied warranties of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. In no event shall University of Coimbra be liable for any special, indirect or consequential damages (or any damages whatsoever) resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of this software.

Copyright (c) 2000 University of Coimbra, Portugal. All Rights Reserved.

This product includes software components distributed by Steve Souza.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS "AS IS"' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright © 2002, Steve Souza (admin@jamonapi.com). All Rights Reserved.

This product includes software developed by the OpenSymphony Group (http://www.opensymphony.com/.)"

Copyright © 2001-2004 The OpenSymphony Group. All Rights Reserved.

PANTONE (R) Colors displayed in the software application or in the user documentation may not match PANTONE-identified standards. Consult current PANTONE Color Publications for accurate color. PANTONE(R) and other Pantone LLC trademarks are the property of Pantone LLC. (C) Pantone LLC, 2011.

Pantone LLC is the copyright owner of color data and/or software which are licensed to Oracle to distribute for use only in combination with Oracle Documaker. PANTONE Color Data and/or Software shall not be copied onto another disk or into memory unless part of the execution of Oracle Documaker.

This product includes software developed by Dave Gamble and distributed via SourceForge.net (http://sourceforge.net/projects/cjson/)

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.

Copyright (c) 2009 Dave Gamble

Contents

Windows (32-bit) 17

AIX (32-bit) 17

IBM System z Linux (64-bit) 18

Linux (32-bit) 19

Sun Solaris (32-bit) 19

Installation 21

Installing Transall (PC) 21
Installing Transall (UNIX) 26

Setting Up the Environment 26

Running the Installation Script 27

Transall Applications 31

What can Transall do for Me Technically 33
Characteristics of a Transall Application 34
Handling Technologies 34

Integrating Applications via ActiveX Automation 34
Integrating Access to Data via ODBC and JDBC 35
Supporting Win32 Platforms 35

Components of a Transall Application 35

Sources, Destinations, and Maps 36
Tables and Sets in the Transall Database 38
Logic Trees 39
Transall Scripts and Script Modules 39
The Transall Product Package 40

About the Transall Editor 41

Transall Projects 43

Creating a Project 44
Naming a Project 44

Default Project Components 45

Menu Bar and Toolbars 45
Control Bars and Workspace Area 45
Component Explorer Control Bar 46

Opening a Component from the Component Explorer Control Bar 46

Component Inspector Control Bar 48
Editing Component Properties 48
Opening Source Code for a Component's Built-In Methods 49
Revealing and Hiding Control Bars 51
Rearranging Control Bars and Toolbars 51
Workspace Area 53
Other Bars and Child Windows 53
Global Renaming of Components and Subcomponents 54
Transall Project Sharing 54
Transall Project Command-Line Compilation 55
Additional Menus 57
File Menu 58
Edit Menu 64
View Menu 72
Component Explorer 72
Component Inspector 73
Output Bar 75
Logic Bar 76
Resource Bar 77
SQL Bar 78
Debug 78
Breakpoints 80
Toolbars 80
Project Menu 83
Add Source 83
Add Destination 84
Add Map 85
Add Logic Tree 86
Add Script Module 88
Wizards 89
XML to Docuflex 89
Flat File Wizard 91
Import Table Layout 92
VRF to XML Plus 95
Synchronize 97

Data Sources 97
Links 97
View Links 98
View Queries 99
DMG Report 100
Compile 101
Next Error 101
Previous Error 101
Project Settings 101
Debug Menu 103
Go 103
Stop Debugging 103
Break 104
Step Into 104
Step Over 104
Run to Cursor 105
Toggle Breakpoint 105
Clear All Breakpoints 105
External Runtime Error 106
Tools Menu 107
Options 107
Using the Options Dialog and Views 107
Using the Editor Tab 108
Using the Formats Tab 109
Using the Separators Tab 110
Using the Documanage Tab 111
Window Menu 112
New Window 112
Close All 113
Cascade 113
Tile 113
Arrange Icons 113
Help Menu 114
Help 114
About Transall Editor 114

About Sources and Destinations 115
New Export Capabilities 115
Enhanced Read Flexibility 116
File-Based Sources and Destinations 116
ODBC-Compliant Sources and Destinations 117
Details on Files 117
About Record Subcomponents 117
Defining a Record's Identifier Field 118
Data Types 119
Using Format Options 119
Using Separator Options 120
Creating Sources and Destinations 120
Describing a Fixed File Source 124
Adding a Record Subcomponent 124
Changing the Name of a Record Subcomponent 125
Copying Field Descriptions Into an Empty Record Subcomponent 125
Deleting a Record Subcomponent 125
Describing a Delimited File Source 126
Describing a COBOL File Source 126
Describing a PPS File Source 129
Reference for Component Properties - Traditional Files 130
Source Properties 130
Fixed File 130
Delimited File 134
COBOL File 138
PPS File 142
Destination Properties 143
Fixed File 143
Delimited File 147
COBOL File 151
Documaker File - Variable Replacement File (VRF) 154
Reference for Component Properties - SQL 156
Reference for Component Properties - ODBC 156
Source Properties 156
ODBC Data Source 157
Destination Properties 158

ODBC Data Source 158
Describing an ODBC-Based Source 159
Create the Source Component 160
Define a Query 160
Query Properties 161
Identify One or More Tables for the Query 163
Selecting Table Columns 164
Defining Join Criteria for the Query 166
Defining Filter Criteria 168
Define Sort Criteria 169
Describing an ODBC-Based Destination 173
Query Properties 174
Details on SQL 175
SQL Bind Variables 175
Processing SQL statements on non-WIN32 platforms 175
Details for Accessing Databases from UNIX via SQL 176
Process Overview 176
Creating DSN Files for UNIX 178
Linux ODBC Support 182
Creating DSN Files for the PC 183
Building the Transall Executable 183
Transferring the Transall Executable to UNIX 183
Details for Accessing Databases via JDBC 184
Mapping of ODBC Data Source Column Data types to Transall Record Field Data Types 187
Overview 189
Setting up a Docuflex File Destination 189
How Data Moves Through a Docuflex File Destination 191
Defining a DataSet Identifier 192
Using a Docuflex Destination as a Data Source 192
Overview 197
FP Plus Built In Features 197
How FP Plus Works 198
FpPlus Business Logic 199
FpPlus Data Destination Details 201
Documaker FP Plus Data Destination Properties 201
Documaker FP Plus Data Destination Record Properties 204

Examples 205
Overview 221
Scripted Assistant 221
Scripted Sources and Destinations Operations (Events) 224
Scripted Source and Destination Properties 224
Overview 225
XML Plus Features 225
Adding Records and Fields 227
Element IdentifierValue Properties 228
Field Usage 229
Transaction Boundaries 230
XML Plus and the LogicTree 230
XML Plus Source vs. XML Source 231
XML Plus Source Component Property Details 232
XML Plus Source 232
XML Plus Source Record 234
XML Plus Source Record Field 234
Overview of Event-based XML Support 237
How Event-based XML Parsing works in Transall 237
Setting up an Event-based XML Source 237
Adding a Record for an XML Data Source 242
Setting up an XML Destination 249
Adding a Records to an XML Data Destination 252
Transall and Unicode 257
Data Sources and Destinations 259
ASCII, EBCDIC and Unicode 259
Editing Configuration Files with Unicode 260
Overview 261
Java support script syntax overview 261
Example Script Calling a Java Application 262
Java data types vs. Transall data types 262
Java Object Data Types 264
Running Java Class Applications 265
Running Java Applications Located in JAR Files 266

Get and set Java Object Field Values 266
Transall Java Support Syntax Details 267
Overview 269
Maps and Records 269
Multiple Maps for the Same Destination 271
Creating a Map 272
Interacting with the Map Assistant 273
Using the Resource Control Bar 275
Interacting with the Expression Builder Dialog 276
Enhanced Expression Builder 277
Performing a Map 278
Overview 279
Tables 279
Operations on Tables 280
Resource Considerations for Tables 280
Adding a Table 280
Adding Columns to a Standard Table 280
Sets 281
Adding a Set 282
How a Set Establishes Relations between Rows 282
Rows Become Related as They Are Inserted 284
Walking the Related Parent and Child Rows in a Set 285
Reference for Component Properties 285
Table Properties 285
Set Properties 285
Overview 287
How a Logic Tree Works 288
Adding a Logic Tree 289
Adding an Instruction 290
Deleting an Instruction 291
Cloning an Instruction 291
Reordering Instructions 291
Enabling and Disabling Instructions 292
Logic Tree Instructions 292
Standard Instructions 292

Condition Instruction 293
ControlBreak Instruction 294
DoWhile Instruction 295
Execute Instruction 296
Input Instruction 296
Map Instruction 296
Output Instruction 297
Walk Instruction 297
Testing in the LogicTree 298
Documaker fp Instructions 299
Starting Application Execution in a Logic Tree 299
Setting Up Control-Break Processing 300
Example of Identifier Field Control-Break Processing 301
Example of Break Fields Control-Break Processing 302
Using Variables in Logic Tree Instructions 304
Overview 305
Compiling Versus Building 305
Files Produced when Compiling a Transall Application 306
Using Source Control with Transall Files 306
Editing Build Settings 306
Using the Project Settings Dialog and Tabs 307
Using the General Tab 308
Using the Compile Tab 309
Using the Register Tab 312
Using the Debug Tab 313
Specifying Offline Debugging 317
Using the Locale Tab 319
Running a Transall Application in Debug Mode 321
Compile Settings for Producing a Debug Version 321
Breakpoints in Transall 322
Operating Transall under Debugging Control 323
Viewing Debug Variables in the Variables and Watch Bars 324
Changing Display Scope for Variables 326
Building a Transall Application for Release 326

Deploying Transall Applications 326
Windows Command Line 327
Batch (*.BAT) files 327
ActiveX Automation 328
DLL Application Programming Interface (API) 330
IDS Interface 332
Windows Service 335
Writing to the System Events Logs from a Transall Application Running as a Service 337
Overview 339
Creating a Script Module 340
Coding the Declarations Section 341
Adding a Script 342
Editing a Script's Source Code 344
Viewing the List of Scripts in a Script Module 345
Built-In Component Methods 346
Overview 351
Project Sharing Details 351
Transall How To's: 355
Creating a Transall Project 355
Open an Existing Transall Project 358
Adjusting Control Bars and Windows 360
Setting Up an SQL Data Source or Destination (ODBC connection) 362
Adding an SQL Query (Statement) to an SQL Data Source or Destination 367
Have One SQL Statement Reference the Results of Another 373
Setting Up a Delimited File Data Source 374
Why There Are Multiple Records for Some File Data Sources or Destinations 380
Adding a Record Type to a Delimited File Data Source or Destination 380
Introduction 387
Components 387
Examples 388
Connecting Transall and Docuflex to the TDM 392
Advanced Example of Data Gathering and Document Composition 392
Sample Batch Command Script 394
Named Jobs in the TDM 395

Command-Line Reference 395
TRANDMAN 395
TDMWAIT 396
TDMJOBS 396
TDMTERM 397
Transall 397
Docuflex 397
Starting the TDM Server in Authentication Mode 398
TRANDMAN 398
TDMTERM 399
Transall 399
Docuflex 400
Setting up Transall Projects to use the TDM 401
ACCESSING FILES THROUGH A TRANSALL GATEWAY WITH TRANEXE 408
RUNNING THE GATEWAY SERVER 409
Web Access Functions 459
Simple Functions 459
Advanced Functions 460
Conditional Syntax 461
Like Conditional Operator Syntax 462
Syntax 462
Remarks 462
Other rules for pattern matching 463
Expression Syntax 464
Formatting Syntax 466
Format Functions 470
Sending SMTP Email Messages 472
SMTP Email Functions 474
Recap Log File 480
Limitations 481
EC Regulation 1103/97 481

System Requirements

WINDOWS (32-BIT)

The minimum hardware requirements to install and operate Transall are:

- Processor: Intel-compatible; 512 MHz or Intel® Core™ 2 CPU.2.13GHz processor required, or equivalent
- Memory: 1.99 GB available RAM or more recommended
- Harddisk: 12 MB of available hard disk space required for installation (hard disk usage will vary based on configuration)
- Monitor: recommend 19" monitor or better at 1280 x 1024 resolution or better for viewing and working with documents in workstation software

The minimum software requirements to install and operate Transall are:

- Windows 7, Windows 2003 Server or Windows 2008 Server (the latest service packs for each operating system are highly recommended)
- Database support requires Microsoft ODBC Level 2 drivers (Level 3 drivers recommended)
- Oracle Common Objects 11.2 (required for some sources and destinations)

AIX (32-BIT)

The minimum hardware requirements to install and operate Transall are:

- Processor: POWER 4 or later processors; 1.2 GHz or faster processor required
- Memory: 1.99 GB available RAM or more recommended
- Harddisk: 40 MB of available hard disk space required for installation (hard disk usage will vary based on configuration)

The minimum software requirements to install and operate Transall are:

- AIX 5.2 and 5.3 (32-bit mode only); the latest service packs for each operating system are highly recommended
- Databases supported:

- Oracle 8 and later supported via 32-bit Oracle 8 driver
- Sybase 12.5

IBM SYSTEM Z LINUX (64-BIT)

The minimum hardware requirements to install and operate Transall are:

- Processor: IBM s390x
- Memory: 1.99 GB available RAM or more recommended
- Hard disk: 90 MB of available hard disk space required for installation.

The minimum software requirement to install and operate Transall is:

- Kernel 2.6.16 or later
- Databases supported:
  - Oracle databases with supported ODBC or JDBC driver
  - MySQL version 5.1.51 with supported ODBC or JDBC driver

LINUX (32-BIT)

The minimum hardware requirements to install and operate Transall are:

- Processor: Intel-compatible only; 1 GHz or Intel® Core™ 2 CPU.2.13GHz processor required
- Memory: 1.99 GB available RAM or more recommended
- Harddisk: 40 MB of available hard disk space required for installation (hard disk usage will vary based on configuration)

The minimum software requirements to install and operate Transall are:

- Kernel 2.4.9 or later; formal Linux distribution is desirable (e.g., Red Hat Enterprise Linux or Novell SuSE Linux Enterprise Server); all current fixes are required for each kernel release.
- Databases supported:
    - Oracle 9 and later supported via Oracle 9 driver
- Linux 2.4.9 or later with the following libraries:
    - libdl.so.2
    - libstdc++-libc6.2-2.so.3
    - libm.so.6
    - libc.so.6
    - /lib/ld-linux.so.2
    - libpthread.so.0
    - libns1.so.1

SUN SOLARIS (32-BIT)

The minimum hardware requirements to install and operate Transall are:

- Processor: UltraSPARC IIIi processors or later; 1.06 GHz or faster processor required
- Memory: 1.99 GB available RAM or more recommended
- Harddisk: 40 MB of available hard disk space required for installation (hard disk usage will vary based on configuration)

The minimum software requirement to install and operate Transall is:

- Sun Solaris 8 and 9 (5.8 and 5.9)
- Databases supported:
    - Oracle 9 and later supported via Oracle 9 driver
    - MySQL version 4.0 and later using MyODBC

Installing Transall

INSTALLATION

Transall operates on WIN32 and UNIX platforms. Because of the differences in platforms, this guide provides separate installation routines.

To Install Transall on	See this:
WIN32	*Installing Transall (PC)* on page 21
UNIX	*Installing Transall (UNIX)* on page 26

INSTALLING TRANSALL (PC)

The following installation procedure assumes that you have no other active applications running on your computer. Transall is installed via a Graphical User Interface (GUI) routine.

To Install Transall (PC)

1. Insert the Transall Installation CD in the appropriate drive.
2. Select **Start>Run** and Windows displays the Run dialog box.

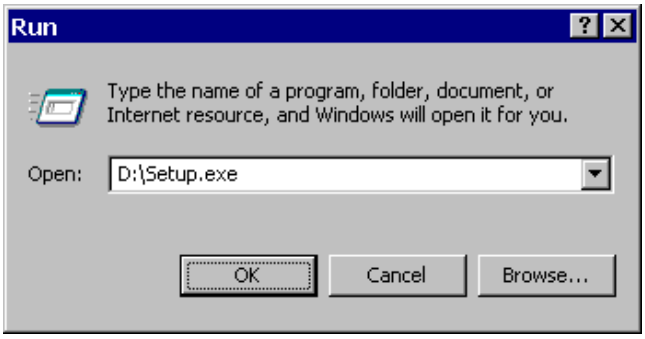


Figure 1: Run Dialog Box

3. Type **D:\SETUP.EXE** in the Open text box and click **OK** or **Browse** to locate the program. If the installation CD is in a drive other than D:, enter the appropriate letter specification.

The installation routine displays a dialog box indicating the InstallShield Wizard's progress, followed by the Transall Welcome screen.

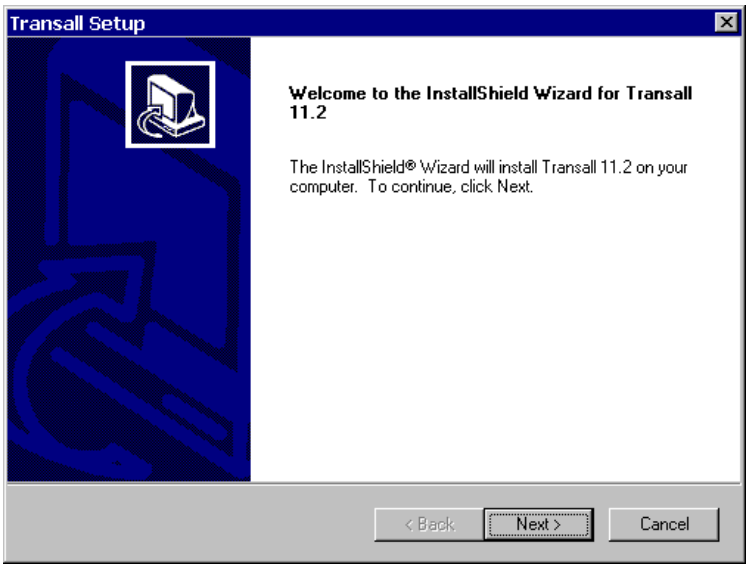


Figure 2: Welcome Screen

4. Click **Next>** to continue with the installation or **Cancel** to quit the program.

   The routine then displays the Customer Information dialog box, prompting you for your name and the name of the company for whom you work.

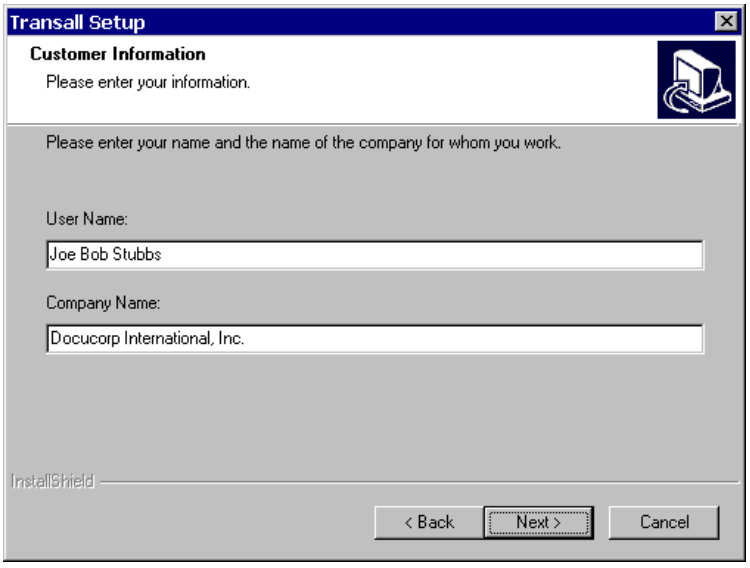


Figure 3: Customer Information Screen

5. Click **Next>** to continue with the installation if you haven't already done so. You can also choose **<Back** to return to the previous screen or **Cancel** to abort the routine.

The routine then displays the Choose Destination Location dialog box, prompting you for the folder name/directory path into which you want to install the program

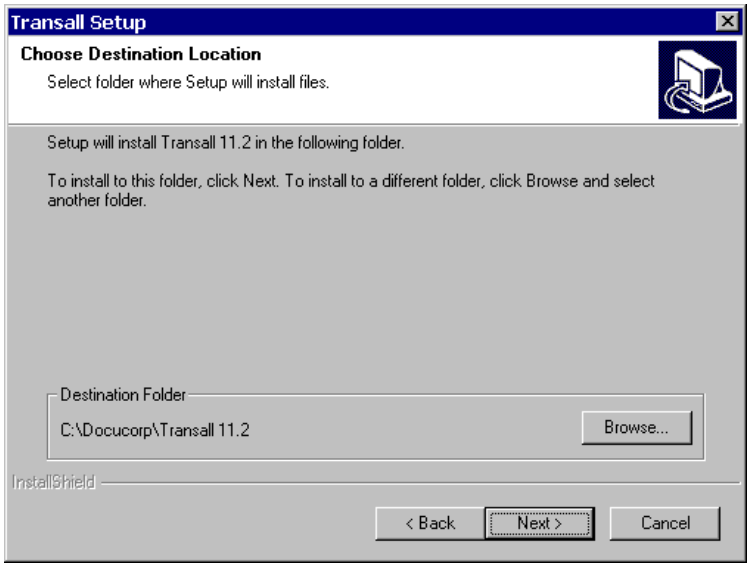


Figure 4: Choose Destination Location Screen

6. Perform one of the following procedures:

To	Perform this action:
Accept the default path the installation routine proposes	Click **Next>**.
Enter another path for the installation	Click **Browse**, select a new **path**, and then choose **Next>.** **Note**: The Destination Folder lists the last location to which the program was installed. If you're re-installing the program to a different location, you should **Cancel** the routine, un-install the program from its previous location, and then install the program to the desired location.

7. Click **Next>** to continue with the installation if you haven't already done so. You can also choose **<Back** to return to the previous screen or **Cancel** to abort the routine.

The Setup Type dialog box displays asking which setup type to install.

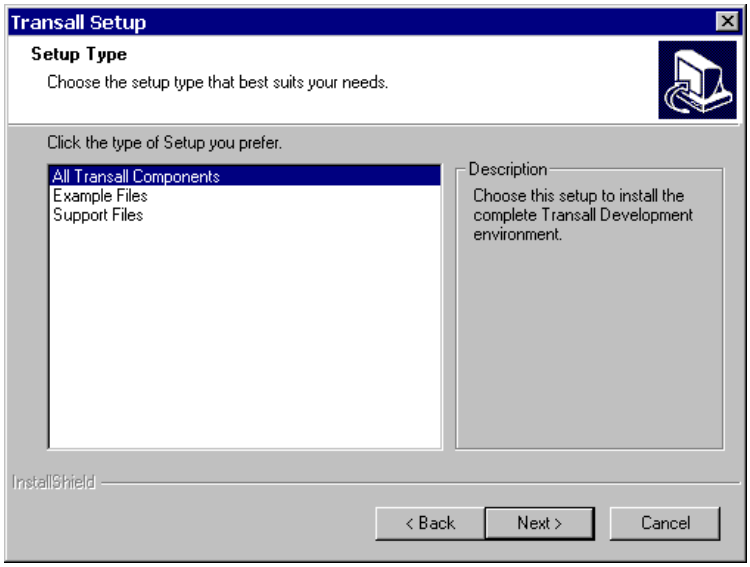


Figure 5: Select Setup Type

8. Choose the Setup you prefer, then choose **Next>** to continue with the installation. You can also choose **<Back** to return to the previous screen or **Cancel** to abort the routine.

   The Select Program Folder dialog box displays the folder where you'll store the program files.

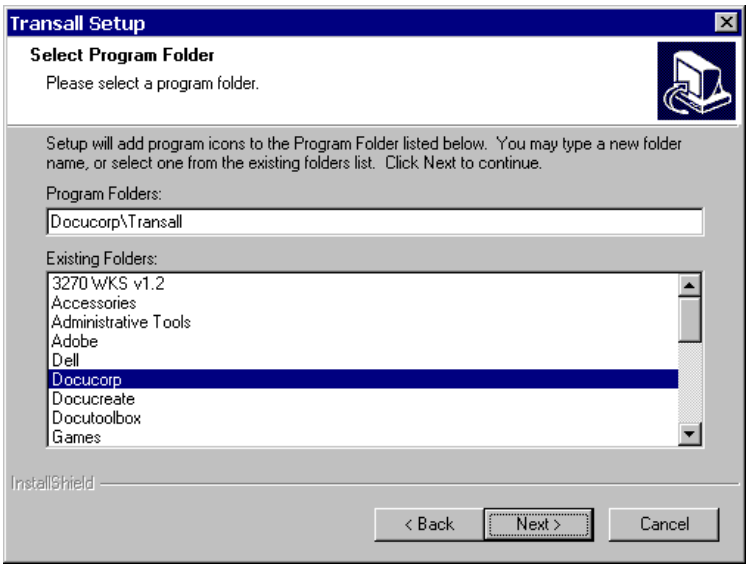


Figure 6: Select a Program Folder

9. Perform one of the following procedures:

To	Perform this action
Accept the default program folder the installation routine proposes	Click **Next>**.

To	Perform this action
Select an existing folder for the installation	Use the scroll bar to click on an existing folder; then click **Next>**.

10. Click **Next**> to continue with the installation if you haven't already done so. You can also click **<Back** to return to the previous screen or **Cancel** to abort the routine.

    A dialog box indicating the setup program's progress displays until all the files have been copied to your PC.

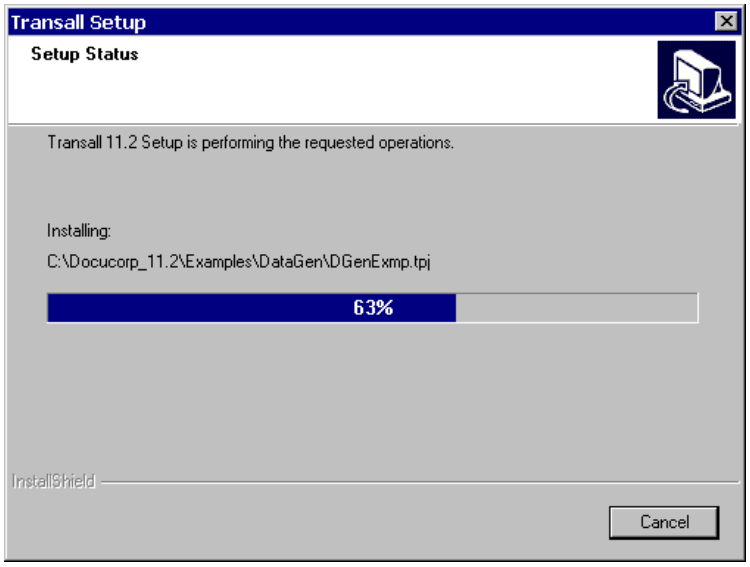


Figure 7: Follow the Installation's Progress

When the installation is 100% complete, the program displays the Transall Setup Complete dialog box.

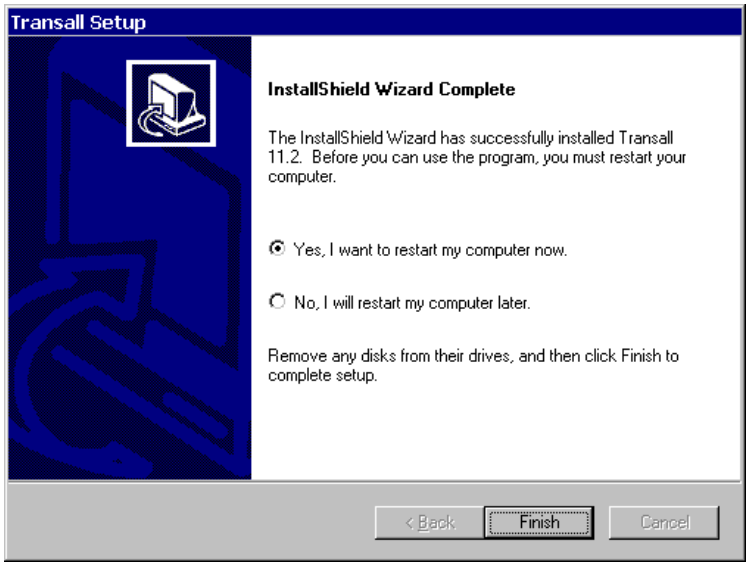


Figure 8: Restart you Machine

11. Click **Finish** to complete the installation program and reboot your workstation

INSTALLING TRANSALL (UNIX)

- Copy the appropriate files from the installation disc to a directory to which all users of the product have access:

If you're on this platform:	Copy these files:
IBM AIX	setuptm12.01.aix
Linux	setuptm12.01.lnx
Sun	setuptm12.01.snx
IBM System z Linux	setuptm12.01.zlnx

Note If you want to install the product into the **/home/oracle/transall** directory, first create the directory and then copy the files into it. You should perform this operation with root authority so that file permissions can be set for users and groups, but using root authority isn't absolutely necessary.

SETTING UP THE ENVIRONMENT

Setting up Transall on UNIX is a multiple-step process involving

- setting up access permissions for directories and users (see *To Set Up Permissions* on page 26)
- running the installation script (see *To Run the Installation Script* on page 27)
- verifying the environment variables (see *To Verify the Environment* on page 29)

To Set Up Permissions

1. Ensure that you're logged on with the proper authority for the directory and installation files. You might need to assign user and group permissions.
   a. Verify that the directory in which you're about to run the setup script has write permission. If the permission isn't set correctly, do so now.

Example
If user **jxsmith** created the **/home/oracle/transall** directory, use that ID to run the setup script. Verify that **/home/oracle/transall** has write permission using the "ls" command:

```
ls -ald /home/oracle/transall
```

You should see the following results:

```
drwxr-xr-x  2 jxsmith  staff  512 Apr 19 07:52 /home/oracle/transall
```

The most important part of this message is the third letter of the string "dr**w**xr-xr-x". The "w" indicates that the owner of this directory has write permission. If the third letter isn't a "w", assign write permission with the following command:

```
chmod 755 /home/oracle/transall
```

b. Make sure you're in the directory containing the installation files:

```
cd /home/oracle/transall
```

c. Change the file permission for the installation file.

- the files should have read permission (read/write for the user, if desired):

```
chmod 644 setuptm12.01.aix (or .lnx, or .snx or .zlnx)
```

2. Adjust the file permission of the installed files for the various users and groups needing access.

RUNNING THE INSTALLATION SCRIPT

The Transall 12.1 installations on the Unix platforms (AIX, Linux, Sun, zLinux) have been enhanced to use the Unix *uuencoded text* encoding method instead of the *tar* files, which will no longer be provided. The new *uuencoded* files are text-based files instead of binary *tar* files. This should make the transmission and distribution of these installation files easier and more portable to different machines in your network.

You can perform either a new installation or update an existing one. The installation process detects whether you are updating an existing installation when you enter the directory name of the target or destination of the installation. If the directory does not exist, then a new install is performed.

If the directory already exists and "tranexe" is found in the existing directory, then an update to an existing installation is performed. Both these conditions must be true. The installation update process will ask if you first want to "back up" the existing installation. If you choose "back up", all the existing files in the target directory will be copied to a subdirectory called "backup". Since this is an installation update, only the new binary files will be copied/installed to your target/destination directory, preserving any existing data files you may already have in that directory (e.g., .ini files and .dde files).

To Run the Installation Script

- Run the installation script you want to install.

  The naming convention for these scripts is as follows:

  setuptrnvv.rr.sss

  where:

vv	the version number	
rr	the revision number	
sss	the Unix platform system	
	aix	IBM's AIX
	lnx	Linux
	snx	Sun
	zlnx	IBM's System z Linux

Therefore, if you were installing Transall 12.1 on the Linux operating system, the script name will be “setuptm12.01.lnx”.

Following is an example of running the Transall installation on the Linux operating system and its display and response. In this example, we will install Transall all into the same directory named “transall12.n”.

Note The script will automatically invoke the Transall installation scripts. You can choose to install or bypass the installation of any one of these components.

```
$ setuptrn12.01.lnx

* * * Transall 12.01 Installation/Update * * *
Loading, Please Wait ...

Do you want to Install Transall 12.01?
Enter 'Y'es or 'N'o, or 'X' to cancel this process.
?>y

Please select an Installation/Update Directory:          <= Enter the directory name where you want to install.
Press <Enter> to accept default: </home/jxsmith/trn12.01> This is a qualified or non-qualified directory name.
?>transall12.n                                           In this
                                                         example, a sub-directory named “transall12.n” will be
Install Transall 12.01 to <transall12.n>?                created in the current directory.
{y or n}?>y

Installation/Update in progress. Please wait...
Updated Base Components ...

...
Transall Installation Complete.
```

In the following example, we will update an existing Transall installation on the Linux operating system. We already have Transall installed into the same directory named “transall12.n”.

```
$ setuptrn12.01.lnx                                        <= Enter the directory name where you want to install.
* * * Transall 12.01 Installation/Update * * *
Loading, Please Wait ...                                   In this example, the directory name already exists
                                                           and thiswill cause the installation process to update
Do you want to Install Transall 12.01?                     the existing installation.
Enter 'Y'es or 'N'o, or 'X' to cancel this process.
?>y

Please select an Installation/Update Directory:            <= Choose 1 to first back up your existing
Press <Enter> to accept default: </home/jxsmith/trn12.01>  installation files to a sub-directory named “backup”,
?>transall12.n                                             then the new modules will be installed.

Please select a update type
1 = Backup and Update Installation
2 = Update Installation
3 = Quit ?>1

Update Transall 12.01 to <transall12.n>?
{y or n}?>y

Installation/Update in progress. Please wait...
Updated Base Components ...

Transall Installation Complete.
```

To Verify the Environment

Note In the following two steps, LD_LIBRARY_PATH is the name of the environment variable on Linux and Sun, while the name is LIBPATH on AIX.

1. All Transall users must add the library directory to their LD_LIBRARY_PATH environment variable. Edit the login profile and modify the environment variable so that the operating system can find the shared object (.so) files that Transall uses.

```
LP=/home/oracle/transall
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$LP
export LD_LIBRARY_PATH
```

2. Be sure to refresh the LD_LIBRARY_PATH environment variable by executing the login profile before trying to use Transall (e.g., first logout and then log back in). To verify that the environment variable has been updated correctly, type:

   ```
   echo $LD_LIBRARY_PATH
   ```

   you should see something similar to the following:

```
/usr/lib:/usr/bin/lib:/home/oracle/transall
```

3. All Transall users must also add the executables directory to their PATH environment variable.

Example
If *tranexe* is in **/home/oracle/transall**, update the PATH with the following script fragment in the user's profile:

```
XA=/home/oracle/transall:
if [[ "$XA" != "$PATH" ]]
then
   PATH=$XA$PATH
   export PATH
fi
```

The syntax of the script fragment may vary depending on the shell you're using.

Transall Overview

This chapter introduces the purpose of the Transall™ product, describes Transall's enabling technologies, introduces the Transall Application and its major components, and identifies the tools that are included in the Transall product package.

TRANSALL APPLICATIONS

Transall is an Extract Translate Load (ETL) tool for creating software objects, called **Transall Applications**, that allow the information found in dissimilar systems to work together in a new application. A Transall Application does this by establishing new paths for data between existing systems and by performing flexible, record-oriented operations across those data paths.

A Transall Application can create a single stream of data records from a database to a desktop application. Or it can create several streams—from each of several files, databases, or data producing applications—that have the same desktop application as the destination.

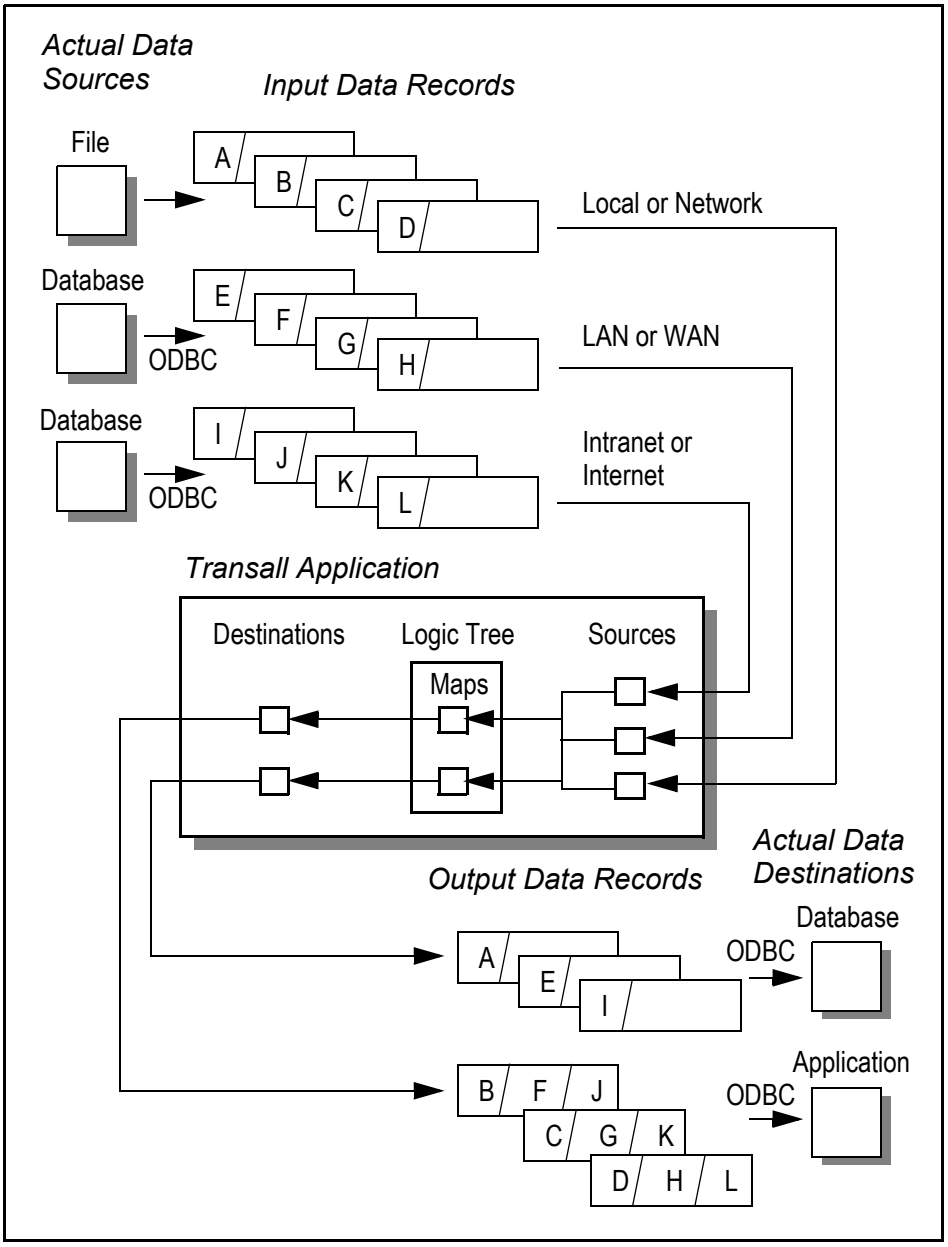


Figure 9: *Transall Applications Establishes Data Paths Between Applications*

323 summarizes the kinds of data paths that a Transall Application can establish. As shown, one Transall Application can obtain input data records from more than one data source and provide output data records to more than one data destination. Sources and destinations can be files including XML, databases accessed via Microsoft's Open Database Connectivity (ODBC), and applications accessed via ODBC.

The top of the diagram shows three streams of input data records that a sample Transall Application might obtain. The three streams originate from a database, a file, and a data-producing ODBC-enabled application, respectively. The center of the diagram depicts the components in a Transall Application that construct output data records from input data records. The bottom of the diagram shows two streams of output data records being directed to a database and a data-consuming ODBC-enabled application.

For example, a Transall Application can provide record-oriented data to an Excel spreadsheet running on your desktop by obtaining data records from local or network files or from an ODBC-enabled database that your workstation can access across your organization's local-area network (LAN), wide-area network (WAN), enterprise-wide Intranet, or the Internet.

The figure also shows that:

- The Transall Application can create one stream of output data records from multiple streams of input data records.
- The Transall Application can create one output data record using data from multiple input data records that originate in different data sources.

WHAT CAN TRANSALL DO FOR ME TECHNICALLY

Transall can copy data from one point to another while translating and reorganizing the data. The tool organizes the process of translating and reorganizing data in to five reusable components. These are:

1. Sources
2. Destinations
3. Maps
4. Logic trees
5. Scripts

By organizing all Transall applications into these five consistent components, Transall reduces errors and improves the ease of on-going maintenance. Transall data traditionally consists of files or Microsoft's Open Database Connectivity (ODBC) data stores but non-traditional data can also be processed. Files can contain records that are either delimited or undelimited, with one or many record definitions per file. Files can also be in Extended Binary-Coded Decimal Interchange Code *(*EBCDIC) or American Standard Code for Information Interchange (ASCII) with binary values that are organized as Big Endian or Little Endian (byte order conventions).

Transall can import Common Business Oriented Language (COBOL) copybooks that define a file's records and COBOL data types such as COMP, COMP-1, COMP-2, COMP-3 and redefines are natively supported. Transall can also read and write to ODBC data stores that are ODBC level 2 or higher. This includes SQL Server, ORACLE, Database 2 (DB2), DB2/400, Sybase, SQL Base, Access, Excel, and many more. Transall can process both files and ODBC data stores at the same time moving information freely between files and ODBC.

Transall applications can execute on Windows 32-bit (WIN32), Advanced Interactive Executive (AIX) 4.x, and Linux platforms. On the WIN32 platform Transall applications can also execute as COM servers that are available to any ActiveX enabled application such as Microsoft Visual Basic, Microsoft Office Products, Active Server Pages (ASP) Web pages, and many other software applications. Transall can also process SQL statements on non-WIN32 platforms. This enhances Transall's portability to the supported non-WIN32 platforms: AIX, Solaris, and Linux (Intel).

CHARACTERISTICS OF A TRANSALL APPLICATION

A Transall Application is executable software that runs in an environment provided by the Transall product's Transall Host run time library. A Transall Application can run anywhere the Transall Host run time library is installed. Transall can run on your personal computer or workstation or on a shared resource, such as a departmental server.

For example, you can incorporate a Transall Application into an existing three-tiered application environment though Transall's support for ActiveX and COM. That is, a Transall Application and the calling application can each run on different client workstations, or the Transall Application might run on a server node that can be accessed by one or more client workstations, or both the Transall Application and itscalling application can reside on a server node. Your choices for deploying a given Transall Application depend upon whether the calling application is itself only an ActiveX client or also a server application to its own client applications.

A Transall Application user typically does not interact directly with the Transall Application while it runs. Rather, as illustrated in *Figure 9 on page 32*, the Transall Application behaves as a "glue" application. Without being visible to the user, the Transall Application allows data to pass between other existing systems and manipulates that data along the way if necessary.

HANDLING TECHNOLOGIES

The Transall Application can serve its purpose as "glue" between other systems by utilizing technologies such as: ActiveX™ Automation, Open Database Connectivity (ODBC), and the Win32™ application-programming interface (API).

Integrating Applications via ActiveX Automation

Each Transall Application utilizes Microsoft's ActiveX Automation technology as an ActiveX or COM server. Thus, a Transall Application can be controlled by any ActiveX-enabled application, including the entire Microsoft Office suite of applications, Microsoft Internet Explorer, Microsoft Internet Information Server, Microsoft Visual Basic applications, and other "off-the-shelf" ActiveX-enabled applications such as Visio.

Note Each Transall Application is an out-of-process ActiveX executable.

This means that you can build a Transall Application that consists of *components* (also known in the object-oriented programming world as *methods*), any of which can be called by an ActiveX-enabled application for a specific purpose via ActiveX Automation.

Integrating Access to Data via ODBC and JDBC

Transall utilizes both Microsoft's Open Database Connectivity (ODBC) and IBM's Java Database Connectivity (JDBC) standards for application interoperability. This means that you can create a Transall Application that reads data from, or writes data to, an ODBC or JDBC data source. This can be an important capability for an application that must utilize a Transall Application to work with disparate databases that can be located anywhere on your organization's LAN, WAN, Intranet, or over the Internet.

For ODBC and JDBC operability, Transall expects that a complete and valid ODBC or JDBC installation is available on your personal computer or workstation

Note Transall supports ODBC Level 2 and later.

Supporting Win32 Platforms

Because Transall Applications use Microsoft's Win32 technology, you can deploy them under either Windows 7, Windows 2003 Server or Windows 2008 Server and later, throughout your organization.

Note Transall does not support Windows ME.

Components of a Transall Application

You build a Transall Application from a set of related *components*. Some of these components are required and others are optional, depending on the tasks that the Transall Application must perform. The tool organizes the process of translating and reorganizing data in to five reusable components. These are:

1. Sources
2. Destinations
3. Maps
4. Logic trees
5. Scripts

You use the Transall Editor, a part of the Transall package, as the development environment for creating and organizing these components. *Chapter 4 - Interacting with Transall Editor* on page 43 introduces this tool's features.

Figure 10 on page 37 summarizes the features of the components that can be part of a Transall Application.

SOURCES, DESTINATIONS, AND MAPS

A Transall Application organizes data into Records. Thus, a Transall Application might have knowledge about one or more input records, one or more output records, and how to use information in the input records to store information in the output records. This knowledge is based upon components called **Sources**, **Destinations**, and **Maps**.

When the Transall Application runs, it obtains input records by interacting with Source components and produces output records by interacting with Destination components. A Transall Application can interact with a variety of Sources and Destinations at the same time:

- Files whose record formats use fixed-length or variable-length fields
- Files whose record formats are based on COBOL copybooks
- Files whose format is “well formed” XML
- ODBC data sources
- Oracle product-specific file organizations

Each Source or Destination component describes at least one structure, called a Record. Each Record contains a series of “placeholders” for information, called *fields*. Each field is defined to contain a particular piece of information with its own characteristics. For instance, a field's *datatype* determines whether the Transall Application considers that piece of data to be a number, a date/time, or a string of characters.

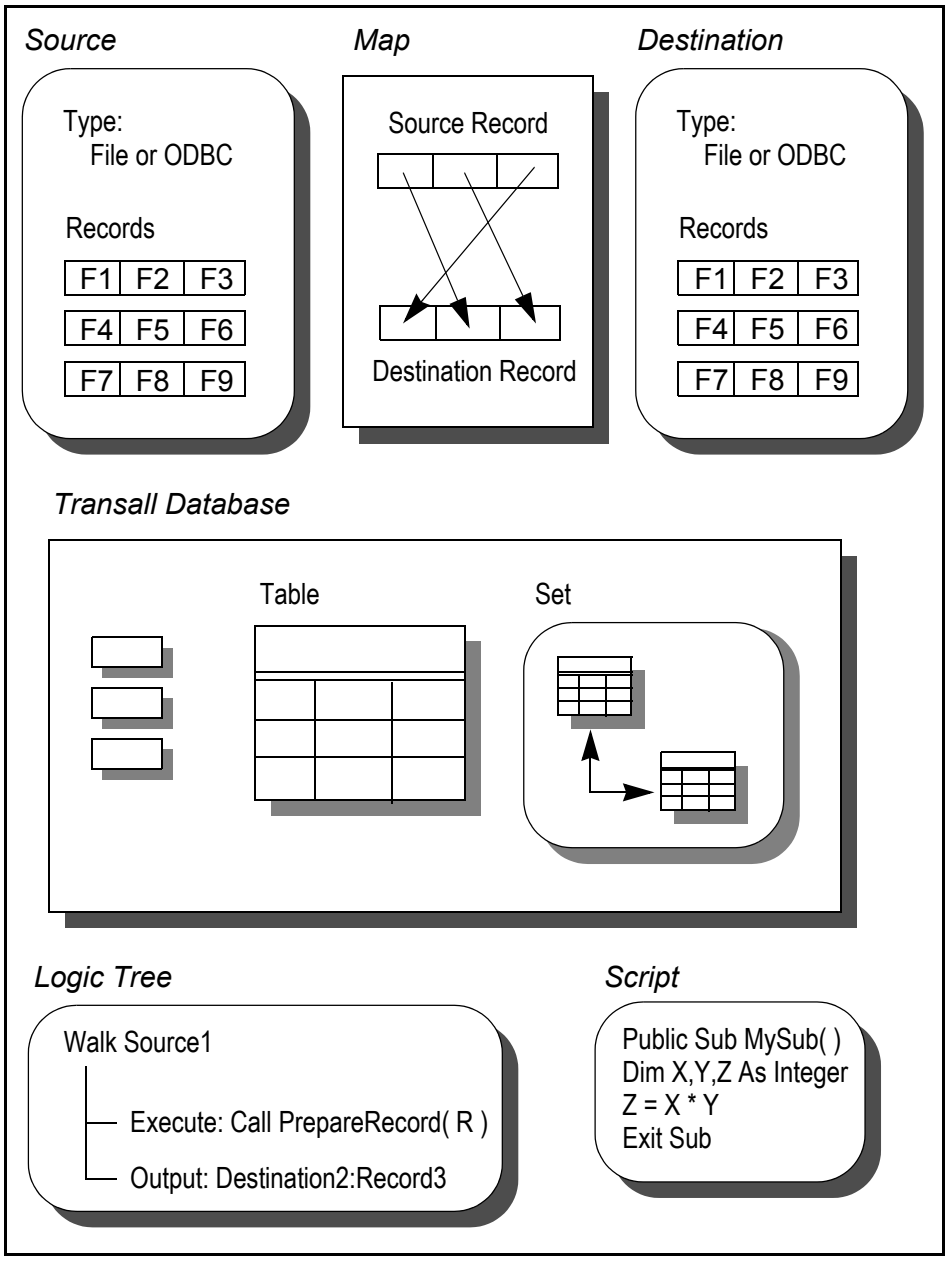


Figure 10: Components in a Transall Application

A Source or Destination can refer to more than one related record structure, which means that it contains more than one Record description. For example, your Transall Application might be required to work with a data set whose record structure varies—that is, where the set of fields present can vary from data record to data record, either according to a fixed pattern or to a "record type" field that is itself part of the record. To process varying data records, you can create a Source or Destination that contains more than one Record description.

Typically, the origin of data for a given field in a Destination's Record is some field in a Source Record, or the origin is a expression that involves one or more fields in one or more Source Records. Maps describe where data is pulled from to populate a Destination Record.

For a description on how to create Sources and Destinations, see *Working with Sources and Destinations* on page 115. *Working with Maps* on page 269 describes how to create Maps.

To direct the Transall Application to take information obtained from one Source, or from more than one Source, and transform it into information that is written to one or more Destinations, you create a component called a Map. A Map describes the *origin* for each piece of data that becomes stored in each field of the specified Destination's Record.

TABLES AND SETS IN THE TRANSALL DATABASE

Optionally, you can define temporary storage in computer memory for a Transall Application to use. This storage is distinct from the storage that the Transall Application creates implicitly and on-the-fly as it works to buffer Source and Destination data records.

To make available this distinct storage, you create components Tables and Sets in the Transall database.

- A Table describes a set of data organized into "columns." Each "row" in a Table has the same number of values, and those values occur in the same pattern in every row.
- When the rows in a pair of Tables have a particular "parent-child" relationship, you can create a Set to describe that relationship. A parent-child relationship between a pair of Tables means that each row in the "parent" Table is related to, or "owns," certain rows in the "child" Table.

The Transall Application can operate on rows in Tables similarly to instances of Records that are defined in a Source or Destination. That is, you can insert a new row into a Table, read or write a Table's next row, and delete a Table row. These operations are listed in *Appendix A:Statement Syntax* on page 411

Parent-child relationships can be defined via sets to describe the relationship between tables, in the Transall database. The Transall Database is just a temporary "Scratch Pad" in computer memory while the application is running. For example, assume that your Transall Application's Transall Database contains a pair of Tables that hold the data for a set of insurance *policies* written by the same agent and for the *endorsements* on those policies. Each policy has one or more endorsements. The policy data is stored in one Table, and the endorsements data is stored in a second Table. In this case, you could create a Set that relates the "parent" Table of policy data to the "child" Table of endorsements data. Each row in the policy Table relates to one or more rows in the endorsements Table.

For more detail on how to create these components, see *Working with the Transall Database* on page 279.

LOGIC TREES

A Logic Tree component describes a series of operations. These operations describe the flow of data through the Transall Application. Typically, an application starts a Transall Application by invoking one of its Logic Trees.

With the goal of making it simple to express the operations that your Transall Application performs, a basic Logic Tree contains the following kinds of instructions:

- Perform a series of iterative, or “looping,” instructions.
- Perform alternate instructions when there is a change in the type of input data record or in the value of a field from input data record to input data record.
- Test any data value known to the Transall Application, and if some condition is true, perform an alternate series of instructions.
- Input the next data record from a Source.
- For the current data record in some Destination, perform the data movements and calculations described in a Map.
- Execute any Transall Script statement.
- Output the next data record to a Destination.

A Logic Tree might contain only a simple series of instructions, which the Transall Application performs in sequence, from the first instruction to the last.

However, certain Logic Tree instructions allow “branching” behavior. These instructions represent an alternate path, or detour, from the Logic Tree's main line series of instructions down one of the logic tree branches. Branch instructions appear nested in the Logic Tree Graphical User Interface (GUI).

For descriptions in more detail on how to create Logic Trees and specify its instructions, see *Working with Logic Trees* on page 287.

TRANSALL SCRIPTS AND SCRIPT MODULES

The Transall editor generates Transall script for the data definitions and operations you setup in the editor’s data sources destinations, maps, and logic trees. In this regard the Transall editor is a code generator. This Transall script is then *compiled* (converted into a form of machine language).

Transall’s Editor also enables you to create Transall Script source code for custom routines, written by you, that you can use in your Transall Application. Editor automatically associates a certain set of built-in methods, called events, with each component that you create, such as OnOpenAfter and OnCloseBefore for file-based Sources and Destinations that you can perform custom operations in via scripts you create.

You can also create your own custom stand alone Transall Script routines. These custom routines can perform tasks not expressed by the instructions and calculations contained in your Transall Application’s Logic Trees and Maps. The custom routines can be called from:

- An Execute instruction in a Logic Tree
- A target expression in a Map
- Any built-in Transall Application method

Using the Editor, you can code one or more related, custom, Transall Script routines in a component called a Script Module. You define each routine as either a *function*, which returns a value to its caller, or as a *subroutine*, which does not return a value to its caller.

For descriptions in more detail on how to create Transall Script routines and integrate them into a Transall Application, see *Working with Transall Scripts and Script Modules* on page 339.

THE TRANSALL PRODUCT PACKAGE

Your Transall product package includes these tools for building and deploying Transall Applications:

- **Transall Editor** (the executable file `tranedit.exe`) provides an *integrated development environment* (IDE) for building a Transall Applications from a collection of related components called a project. Transall Editor offers a look-and-feel similar to other popular IDEs, such as Visual Basic. You also use Transall Editor to debug a running Transall Application.
- **Transall Compiler** (the executable file `trancc.exe`) produces a Transall Application's executable file (`.TEX`). Transall Editor invisibly calls Transall Compiler as needed, but Transall Compiler can also be started from the Windows command line, or a Windows batch file.
- **Transall Host** (the executable file `tranhost.exe`) starts, or *instantiates*, a Transall Application as an ActiveX Automation server. In fact, Transall Host provides the run-time environment for a Transall Application when executing as an ActiveX server.
- **Transall Exe** (the executable file `tranexe.exe`) permits starting or calling a Transall Application from the Windows command line or from a Windows batch command file. This allows you to run and debug a Transall Application in a context that does not require an independent, calling ActiveX-enabled client application.

For more descriptions on how to operate the Transall Editor development environment, see *Transall Projects* on page 43.

For more detailed descriptions on how the Transall Compiler produces a Transall Application, how to use Transall Editor to debug a running Transall Application, and how to use the Transall Host and Transall Exe facilities, see *Debugging and Deploying Transall Applications* on page 305.

ABOUT THE TRANSALL EDITOR

Use the About Transall Editor command to view the copyright information and the current version and release number.

To View the About Box

1. Select **Help>About Transall Editor**.

   The About Transall Editor dialog displays.

2. Click **Info** for System Information that lists the loaded DLLs and their locations, sizes, and versions. This information may be helpful when working with Customer Support if mismatched software is suspected.

3. Click **OK** to return to the Editor.

Interacting with Transall Editor

This chapter describes how to use the Transall Editor development environment to create a Transall project and how to recognize and operate the Transall Editor's most prominent user-interface features.

The Transall product includes the Transall Editor whose look-and-feel is similar to other popular integrated development environments (IDEs) on the Windows platform.

The Transall Editor provides an integrated environment for:

- Creating the components from which you build a Transall Application
- Customizing the source code for built-in Transall Script component methods
- Producing a Transall Application executable file
- Examining the Transall Script source code generated from producing a Transall Application
- Starting an existing Transall Application independently of a separate, ActiveX-enabled, client application
- Debugging a running Transall Application

For more information about using the Transall Editor to create and organize project components, see the Table of Contents of this book for more information.

For more information about using the Transall Editor to run and debug Transall Applications, see *Debugging and Deploying Transall Applications* on page 305.

Transall Projects

As described in the Transall Overview, you build a Transall Application from a set of **components** that are defined in the Transall Editor. The Transall Editor helps you organize these components as a **project**.

The Transall Editor opens one project at a time for you to work with. You build a Transall Application using the components in the open project. The Transall Editor supports a project component sharing feature that let's you reuse components developed in other Transall Applications.

Transall project sharing enables top level components of Transall projects such as sources, destinations, maps, logic trees, scripts, Transall database tables and sets to be shared between projects.

CREATING A PROJECT

After you start the Transall Editor for the first time, your first step should be to use the **File>New** command to create a new project. When you create a new project, you can specify its *name* and the *target directory* (or folder) where the Transall Editor stores the project's contents in the Component Inspector.

NAMING A PROJECT

Naming a project is an important step. The project's name determines the names of the files in which the Transall Editor stores the project's contents. Most importantly, the name of the Transall Application that you produce is based on the name of the project that is currently open in the Transall Editor. That is, a Transall Application's executable file (`project.TEX`) takes the name of the *open project*.

You can store a project's files in any directory that your workstation can access. If other Transall developers depend upon the components in your project, you might prefer to specify a target directory that resides on a File Server or some other shared storage resource.

After you create a project, the Transall Editor displays as shown in *Figure 11*.

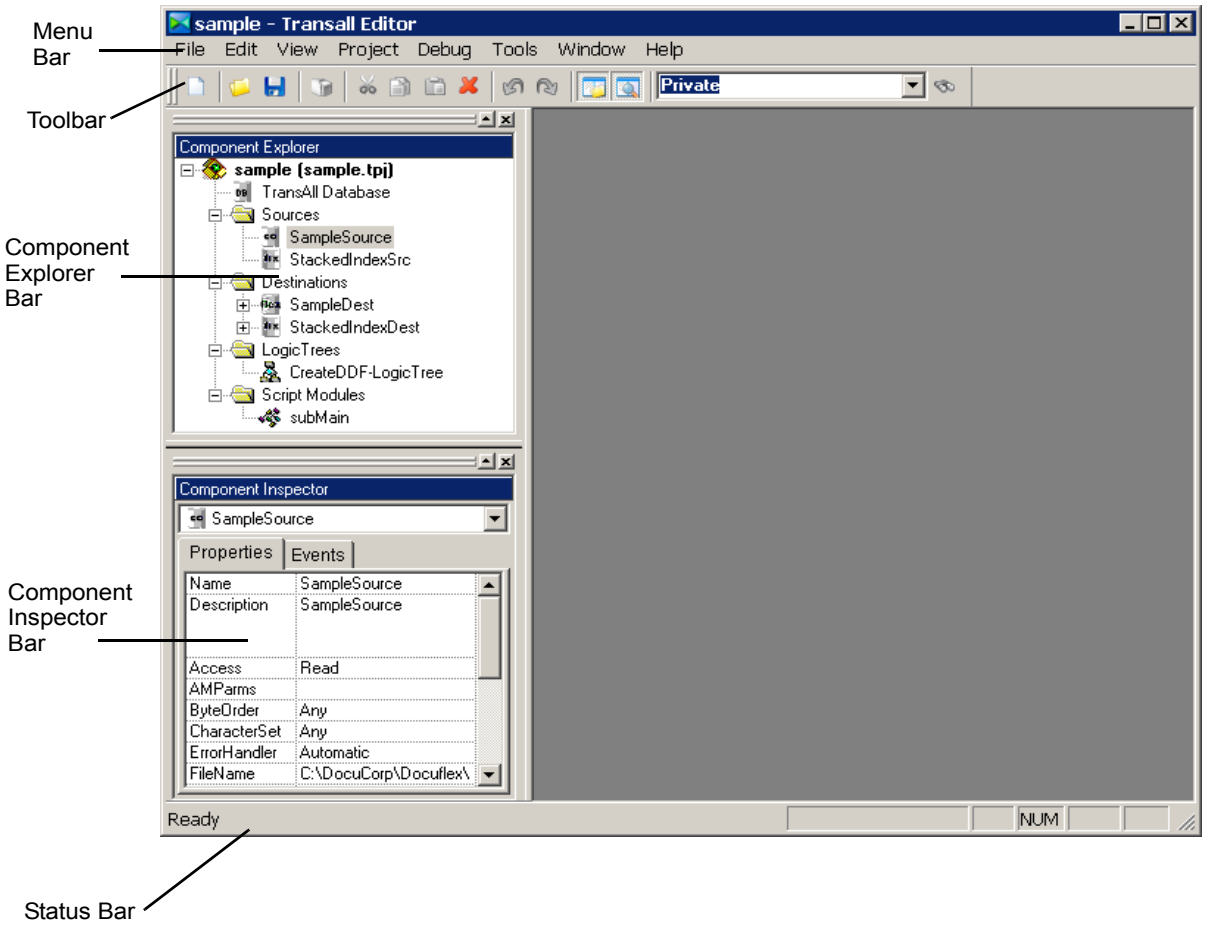


Figure 11: Default Organization of Transall Editor

Default Project Components

The Component Explorer displays a project's contents or its set of components, in a "tree". By default, a new project always contains an empty container component named Transall Database. The Transall Database always appears as the first "node" in the tree of components displayed in the Component Explorer. For more information about the components found in a project's Transall Database, see *Working with the Transall Database* on page 279.

Menu Bar and Toolbars

The Transall Editor's **menu bar** presents drop-down menus. On those menus appear selectable menu commands.

Figure 12: Menu Bar

The **toolbar**, located by default just beneath the menu bar, presents command buttons; each button is a shortcut to a command also found in one of the Transall Editor's drop-down menus.

Figure 13: Toolbar

The **status bar** is located by default at the bottom of the Editor. The status bar's left end displays a one-line message about the results of the most recently completed Transall Editor operation. The status bar's right end displays the state of the CAPS LOCK, SCROLL LOCK, and NUM LOCK keys on your computer's keyboard.

Figure 14: Status Bar

Control Bars and Workspace Area

The Transall Editor presents information about the open project's components in a **Component Explorer** control bar, **Component Inspector** control bar, and an empty **workspace area**.

Click on the name of a component in the Component Explorer bar to *select* it and double-click on the name to *open* it in the workspace area.

COMPONENT EXPLORER CONTROL BAR

The Transall Editor's Component Explorer bar displays a project's contents, or its set of components, as a tree. As shown in *Figure 15*, each tree node may have a clickable + or - control. Clicking on these controls will reveal or hide a portion(s) of that node.

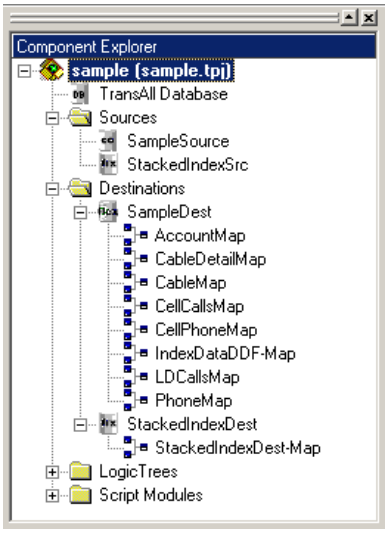


Figure 15: Tree Controls in the Component Explorer Control Bar

As you create new components in the open project, the Transall Editor adds them as tree nodes under the project name node. The Transall Editor displays a container node for the open project's Transall Database, followed by container nodes for the project's components, such as Sources, Destinations, Maps, Logic Trees, and Script Modules.

Opening a Component from the Component Explorer Control Bar

Double-click on the name of a component in the Component Explorer bar to open it. Opening a component causes the Transall Editor to display its contents in an **Assistant** that always appears within the Transall Editor's workspace area. More than one component can be open at the same time; thus, more than one Assistant can appear in the Transall Editor's workspace area.

Figure 16 indicates the location of two Assistants and also illustrates how the selected Assistant corresponds to the selected component in the Component Explorer bar.

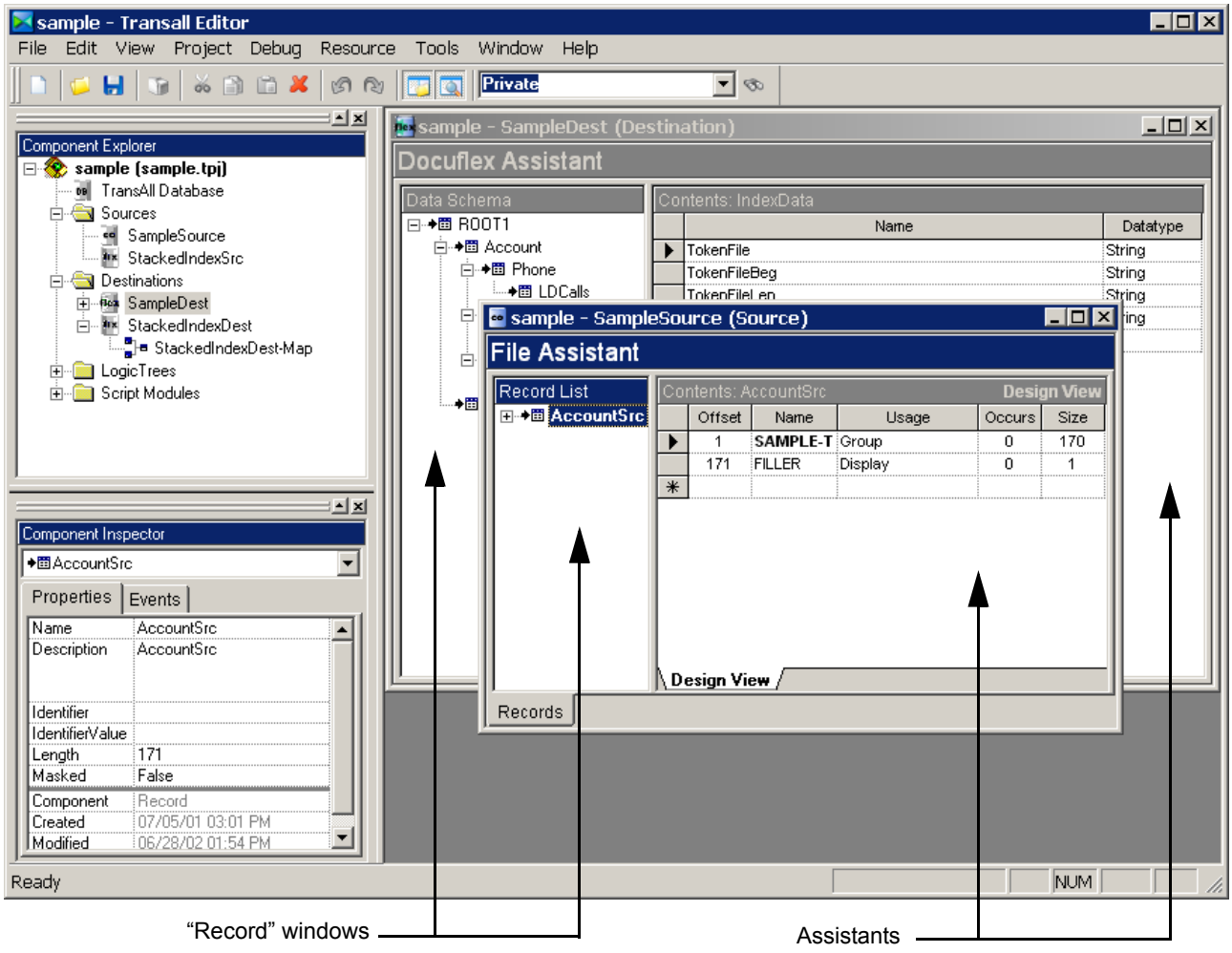


Figure 16: Component Assistants in Workspace Area

You can edit the component's contents in its Assistant. Each kind of component has a characteristic interface for editing its contents.

COMPONENT INSPECTOR CONTROL BAR

After you select a component in the Component Explorer bar, you can examine and edit its details properties in the Component Inspector bar.

The Component Inspector bar presents two sets of information for the selected component:

- Under the Properties tab, component properties and property values
- Under the Events tab:
  - Names of events that pertain to the component; each event name corresponds to a built-in Transall Script subroutines that you can customize for that component.
  - Names of built-in methods that Transall generates for the component. Some of these generated script subroutines are read only and can not be modified or customized by the user, some of the subroutines are available to be modified and customized by the user. The subroutines that are available for user modification are usually the “On” events subroutines such as “OnOpenAfter” or “OnWriteAfter”.

Editing Component Properties

When the Component Inspector's Properties tab is selected, the selected component’s property names and values are displayed, as shown below in *Figure 17*.

The Properties view allows you to view and edit the detailed values of a component’s properties.

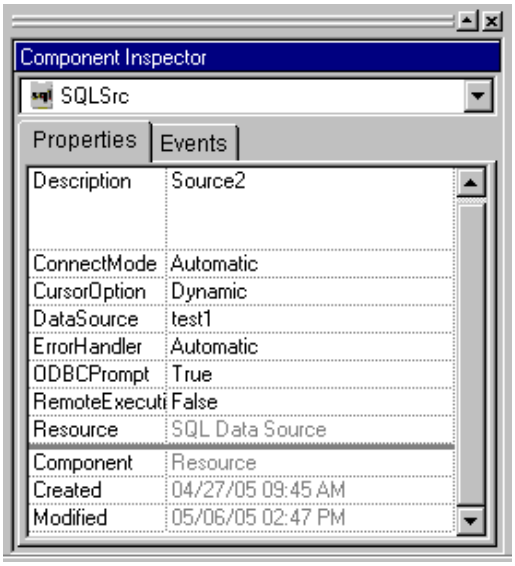


Figure 17: Properties View in the Component Inspector

To Edit Component Properties

- Determine whether a property's value is directly editable in this view by clicking the mouse in the property's value cell. If the value is directly editable, in the cell appears one of the following:
  - A text editing cursor, for typing a new property value
  - A drop-down button with a triangle, for selecting a new property value
  - A browse button, for using a Browse dialog to select a directory or file

  Figure 18, shows the appearance of the Component Inspector as you edit a property value by selecting from a drop-down list.

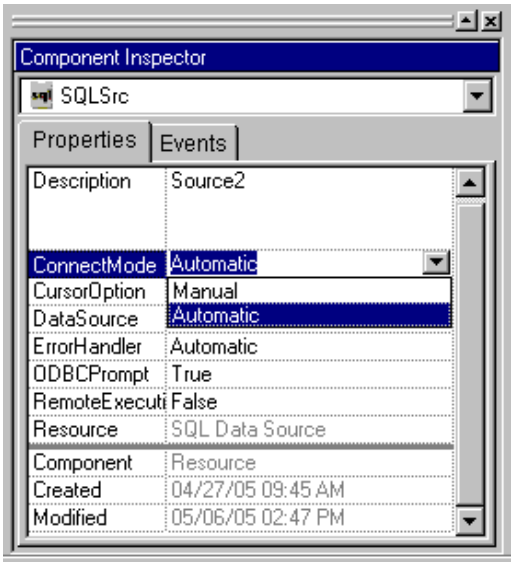


Figure 18: Selecting a Property Value in the Component Inspector

Opening Source Code for a Component's Built-In Methods

When the Component Inspector's Events tab is selected, the names of all *events* that a Transall Application can detect for this kind of component are displayed. Each event represents a point in the Transall Application's activity where a set of operations can be performed. Some of these events are optional. The optional events are noted with a lighting bolt. Optional events are defined by the Transall developer and they are called by Transall based on the events name. For instance the "OnOpenAfter" event is called after the Open subroutine for source or destination has been executed. These optional events provide the Transall developer a place holder for custom script logic. These place holders are sometimes called "User-Exits" in other systems similar to Transall.

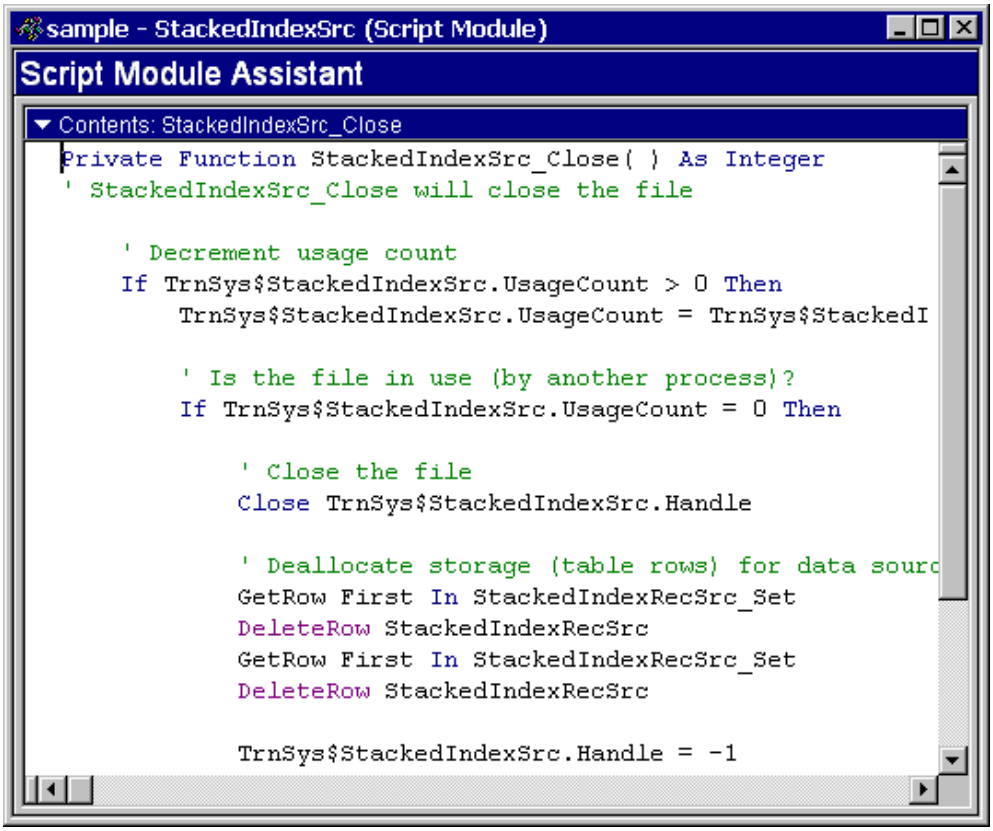


Figure 19: Display of Event Method Source Code

Note The Transall Editor forms the name of a component's corresponding optional events by appending an underline character (_) to the selected component's name, then appending the name of the event.

After you create a optional event for a component, the method's source code is stored as part of the component, not in a separate component.

Each optional event that you create becomes part of the executable code of the Transall Application that is produced from the open project.

For a Transall Application produced from the project shown in *Figure 19*, as the Transall Application runs it automatically calls the optional event `Source1_OnOpenAfter` whenever it detects an "open-source" event that pertains to Source1

Note In the Component Inspector, the name of an event appears in boldface if source code has been created for its corresponding optional event.

You can specify a customized set of operations in a **optional event**, which is a routine that is coded by you in the Transall Script language. For more information about how to code routines in the Transall Script language, see *Working with Transall Scripts and Script Modules* on page 339.

If you decide to code an optional event for a particular component, double-click on the event's name. This causes the Transall Editor to open a Script Module Assistant in which you can add new Transall Script source code for that event. Transall prebuilds the events source code with a shell for the events.

For example, *Figure 19*, shows the shell source code generated for the optional event `Source1_OnOpenAfter`. This event will be executed after the mandatory Open event is executed for the "Source1" component.

Revealing and Hiding Control Bars

By default, if not hidden, the Component Explorer control bar appears in the upper left portion of the Transall Editor; the Component Inspector control bar appears in the lower left portion.

To Reveal and Hide Control Bars

- Select **View>Component Explorer** or **View>Component Inspector** to reveal or hide these bars. You can also perform the following tasks:
  - To maximize a bar, click on the control button in the bar's upper right border.
  - If the bar is already maximized, click on the control button in the bar's upper right border to display the bar in its default size.
  - To hide the bar, click on the **X** button in the bar's border.
  - To reveal and hide the bars, you can also click on the Transall Editor toolbar's Component Explorer or Component Inspector icons.

  For other control bars, se the View menu for a list of the available control bars and toolbars.

Rearranging Control Bars and Toolbars

You can change the appearance of the Transall Editor by rearranging its toolbar and control bars (e.g., Component Explorer and Component Inspector bars). Each toolbar and control bar has a "handle" that you can grab with the mouse and drag to a different location within the Editor.

To Rearrange Control Bars and Toolbars

- Drag-and-drop the bar so that it appears as a distinct window within the Transall Editor (e.g., *floating*).

  -or-

  Drag-and-drop the bar along one of the Editor's four edges (e.g., *docking*).

Figure 20 on page 52, shows the handles for the Transall Editor bars.

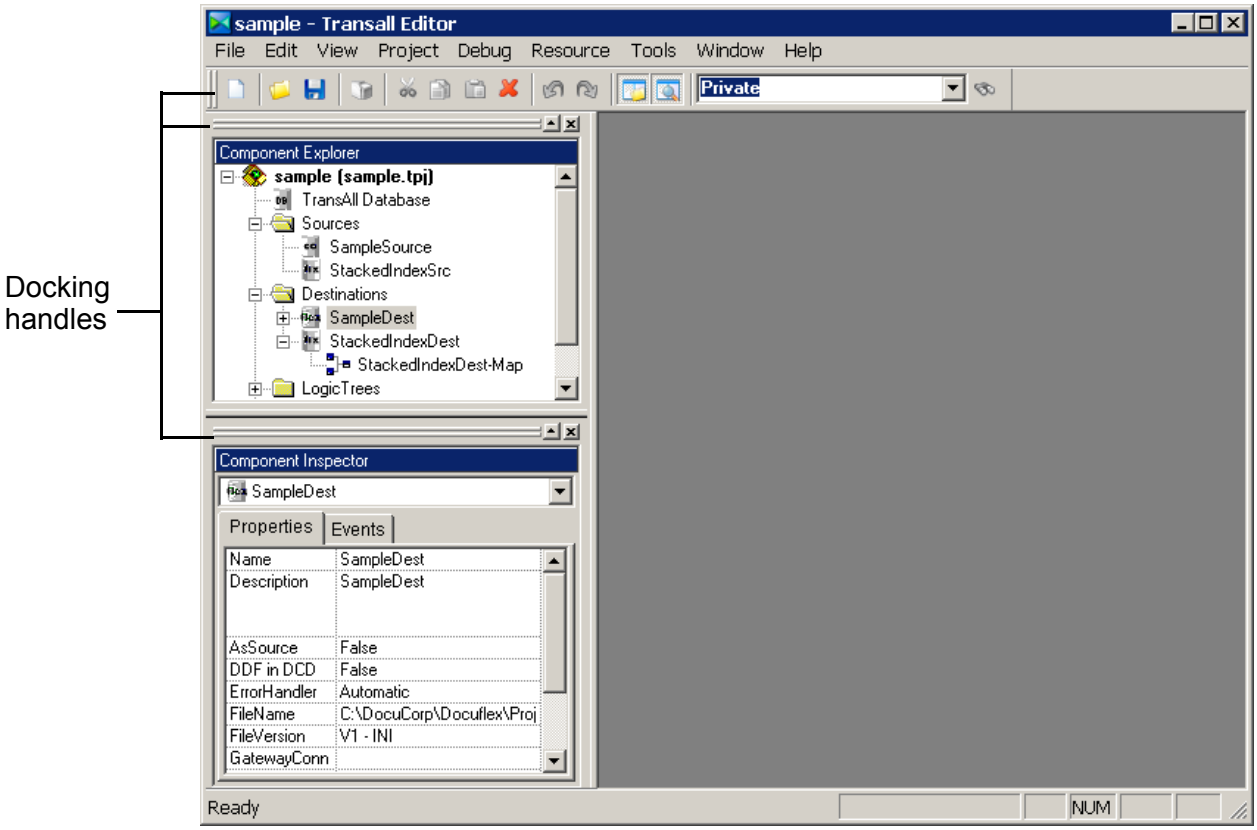


Figure 20: Default Location of Docking Handles for Toolbar and Control Bars

Why rearrange the Transall Editor toolbar and control bars? As you begin to add components to a new Transall project, you might find that you interact with the Component Explorer and toolbar most often. So, you might prefer to hide the Component Inspector and to rearrange the toolbar and the Component Explorer to allow more space for the workspace area and for each new component's Assistant.

As you grab a docking handle and begin to move it, you will see a dim rectangular outline of the item. As you move the item toward an edge, the rectangle will "dock," or jump towards that edge, then change its shape to conform to the height or width of that edge. To cause a bar not to dock, hold down the **CTRL** key while moving it.

Figure 21 on page 53, shows that the user has grabbed the toolbar's docking handle, dragged it toward the right edge, and released it. The toolbar becomes docked along the right edge.

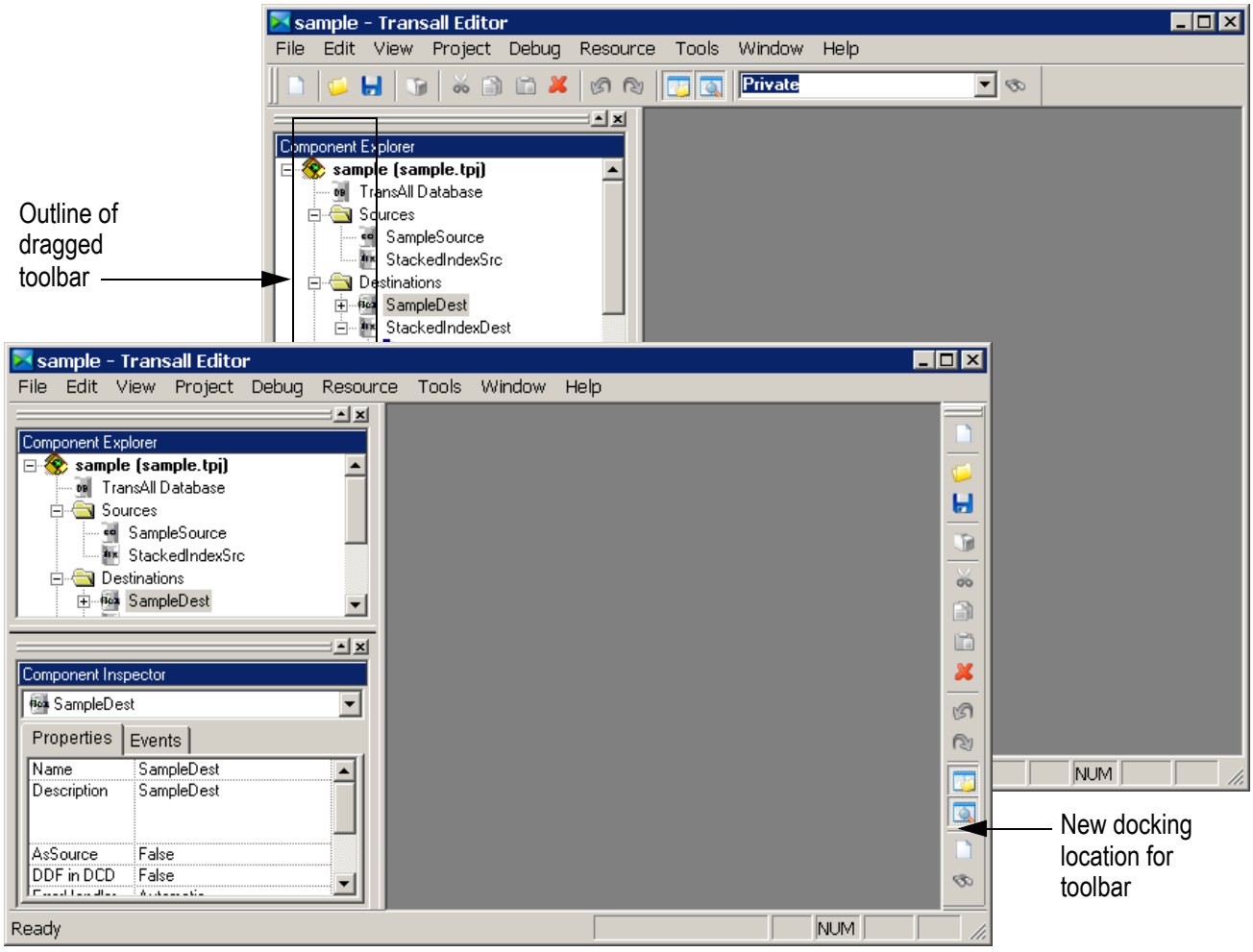


Figure 21: Dragging the Toolbar by Its Docking Handle

Workspace Area

This region of the Transall Editor contains the Assistants of all open components in the open project. Each Assistant is a "child" of the Transall Editor—that is, an Assistant can be maximized only to fill the workspace area.

Other Bars and Child Windows

The Transall Editor can also display an Output control bar to present the results of compiling a Transall Application, the results of performing the **Project>Compile** command.

When you use the Transall Editor to debug a running Transall Application, the Transall Editor can open and display additional Watch, Call Stack, and Variables child windows to assist in debugging.

For more information, see *Debugging and Deploying Transall Applications* on page 305.

Global Renaming of Components and Subcomponents

If you rename a component, the Transall Editor saves you time by automatically replicating that name change wherever that component or subcomponent is referenced by name in the open project.

The rename feature is optional and can be turned off by selecting **Tools>Options** and under the Preferences Tab, click on the check box "Remove and/or invalidate references on delete" to remove the check mark and the optional will be turned off.

Transall Project Sharing

Transall project sharing enables top level components of Transall projects such as sources, destinations, maps, logic trees, scripts, Transall database tables and sets to be shared between projects.

There are two ways to shared project components. The first is called linking. Let's say you have three inputs (sources) and four outputs (destinations), let's make each source and destination its own Transall project, for a total of seven individual Transall projects. You can then create a master Transall project, that references all the other project definitions, and links all the information from all the individual projects into the master project. Changes are made in the individual shared projects and gets updated in the master project though a synchronizing feature, that allows you to synchronize a single project, two projects, three projects, or all projects at any time. This allows several different people to work on different projects at the same time, and then use the synchronize feature to update the master project.

The other way of project sharing, is called copying. The copy share actually does a copy, but does not maintain a connection (link) between the other projects. What the copy does is physically pick up the data from a source project and drop it into a target project, thus becoming part of that project. The copied resources can be edited in the new project, independent of the source project copied from. The copy share facility has no synchronizes feature like the link share facility. Copy sharing is mostly used when a company may want a starting point for a new project and has other projects that have similar data already defined and rather than redefining all of the similar data, you could just copy from the other projects and put it into your new project.

Transall tries to minimize sharing problems by disallowing duplicate names and forcing destinations to be included with shared maps. It is the responsibility of the user to assure that the shared items have the resources they need. For example, Transall logic trees may reference many items. If a logic tree is shared as a link, all items referenced by the logic tree must also be shared as a linked item in the receiving project.

TRANSALL PROJECT COMMAND-LINE COMPILATION

The Transall Editor accepts two optional command-line parameters:

- /Build
- /SyncLinks

The syntax of the command line is as follows:

```
Tranedit <project name> [/Build] [/SyncLinks]
```

where:

Build	Generates project code and compiles it. The results of the compile are displayed on the console and the compiler return value is passed back as the return from the Transall Editor.
SyncLinks	Refreshes project links and saves the project file. Results of the synchronize are routed to the console.

Syntax sample:

```
Tranedit "C:\Transall\Test Projects\Project1.tpj" /Build /SyncLinks
```

If a valid file name is passed with either or both parameters, the Editor is never displayed and runs in batch mode; otherwise, the Editor starts as if a parameter wasn't passed.

You can use **Build** and **SyncLinks** separately or together. If you use them together, SyncLinks is processed first.

Both parameters are case-insensitive and can be in either order.

Sample batch file text:

```
@ECHO OFF

Tranedit "C:\TransAll\Test Projects\Test1\CopyTable.tpj" /
Build /SyncLinks

IF errorlevel 1 GOTO FAILURE

:SUCCESS
ECHO "*** Successful compile ***"
GOTO THE_END

:FAILURE
ECHO "*** Compile Failed  ***"
GOTO THE_END

:THE_END
pause
```

Transall Editor Menu Options

This chapter describes the items in the eight menus that comprise the Transall Editor's menu bar. The Transall Editor has the following menus:

- File Menu
- Edit Menu
- View Menu
- Project Menu
- Debug Menu
- Tools Menu
- Window Menu
- Help Menu

Additional Menus

Along with these permanent menus, the Transall Editor also displays an additional menu in the menu bar whenever you open one of the five Assistant editor windows for a particular component. This extra menu appears between the **Debug** and **Tools** menus.

This menu...	Appears when opening the...
Resource	File Assistant XML Plus Assistant
Database	Database Assistant
ScriptModule	Script Module Assistant
Logic Tree	Logic Tree Assistant
Map	Map Assistant

FILE MENU

The File menu comprises the following menu options:

If you want to	See
Create a project	*New* on page 58
Open an existing project	*Open* on page 59
Close the current project without exiting	*Close* on page 60
Save the project	*Save* on page 60
Save the project under a different name	*Save As* on page 60
Share Transall components between projects	*Share* on page 62
Compile the open project to produce a release version of the Transall Application.	*Make project.tex* on page 63
Exit Transall	*Exit* on page 63

Note that if you enable the Documanage interface using the **Tools>Options>Documange** command, you'll see an additional menu item for Documanage under the following menu items:

- File>Open
- File>Save
- File>Save As

This additional menu item allows to open/save a project file or a Documanage file:

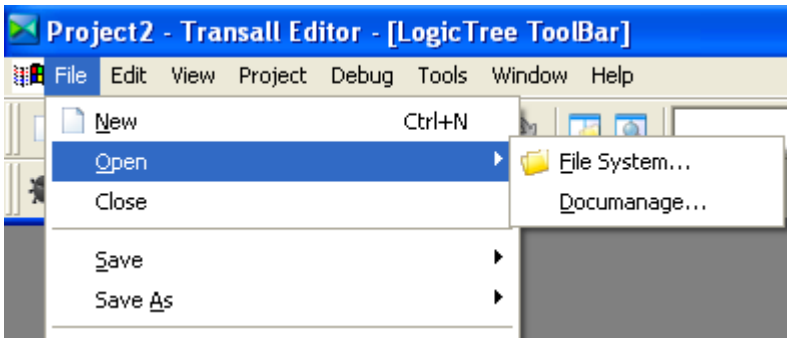


Figure 22: Documange Menu Item

Refer to *Using the Documanage Tab* on page 111 for further information.

NEW

Use the **New** command to create a Transall project. When you create a new project, you can specify its name and the target directory (or folder) where the Transall Editor stores the project's contents in the Component Inspector.

To Create a Project

- Select **File>New**.

-or-

Click in the Standard Toolbar.

The project is given the default name of "Project1.tpj". This name appears in the Title bar at the top of the page.

You can rename the file by selecting **File>Save**.

-or-

Click in the Standard Toolbar.

-or-

Select **File>Save As**.

OPEN

Use the **Open** command to open an existing Transall project.

To Open a Project

1. Select **File>Open**.

   -or-

   Click in the Standard Toolbar.

   The Open dialog displays.

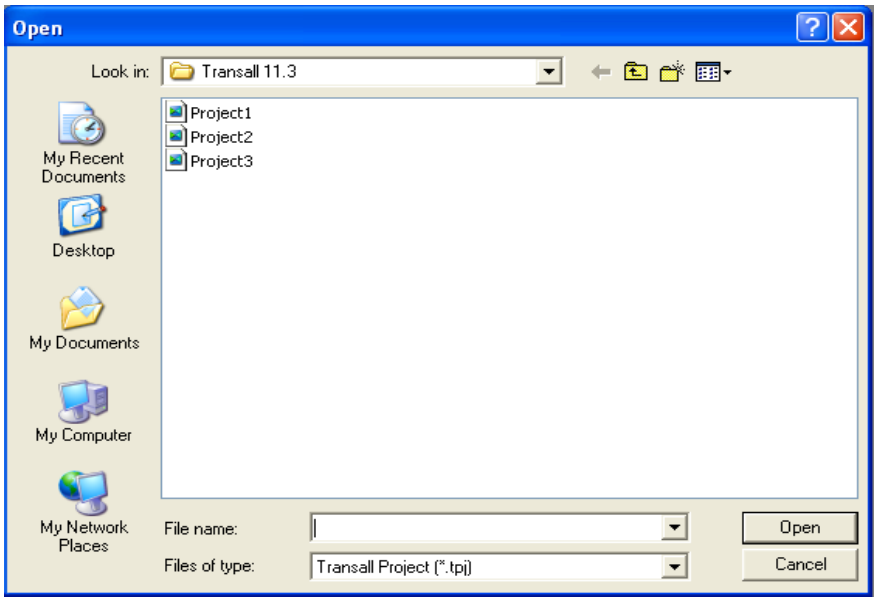


Figure 23: Open a Project

2. In the Look in: drop-down list box, select the directory containing the projects.
3. Select the project that you want to open from the list of projects.

4. Click **Open**.
5. The name of the selected project appears in the title bar of the Transall Editor.

CLOSE

Use the Close command to close the current Transall project. This command expedites switching between projects.

To Close a Project

- Select **File>Close**.

SAVE

Use the **Save** command to save the current Transall project.

To Save a Project

- Select **File>Save**.

  -or-

  Click in the Standard Toolbar.

SAVE AS

Use the Save As command to save the current Transall project under a new name.

To Save a Project with a New Name

1. Select **File>Save As**.

   The Save As dialog displays.

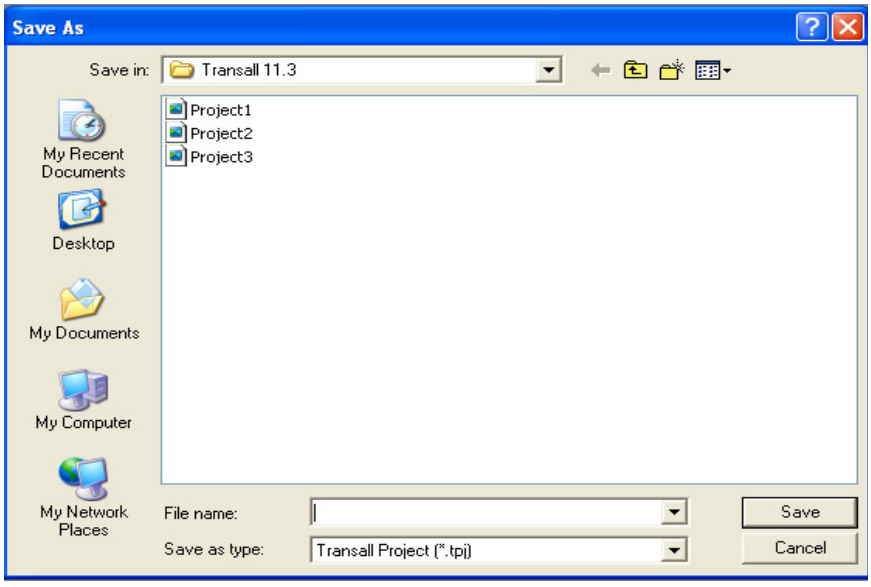


Figure 24: Save a Project

2. In the Save in: drop-down list box, select the directory where you want to save the document project.
3. In the File name: list box, type the name you want to assign to the project.
4. Click **Save**.

SHARE

Use the **Share** command to share items in the current project with another project.

See *Project Sharing Details* on page 351 for a complete explanation of Project Sharing.

To Share a Project

1. Select **File>Share**.

   The Share dialog box opens.

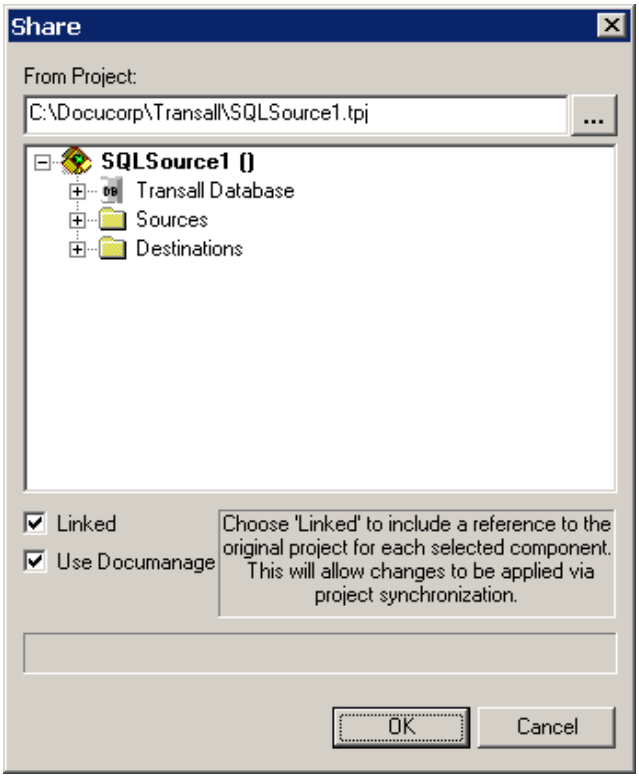


Figure 25: Share Components between Projects

2. Choose the source project by typing in the project name or by clicking the ellipsis, button and using the file open dialog. If you manually enter the project name, you must leave the edit field for the project to open.

   A tree of top-level items from the source project is displayed below the project name.

3. Select items by clicking the check boxes to the left of each item, or right click for a popup menu that allows all children of the highlighted node to be selected.

   The *Linked* checkbox determines whether the item(s) are shared as a link or copy. A project can share any number of links and/or copied items from one or more source projects.

   The *Use Documanage* checkbox means that you want to select the item(s) from a Documanage Cabinet/Folder (i.e., check Use Documanage and then click the browse button).

Source project items for sharing that are grayed out indicate that the item name, or a key sub item name, already exists in the destination project, and sharing is disallowed. To include grayed out items into the target project you must rename the target project items causing the name collision.

4. Click **OK** once you have made your selections.

MAKE *PROJECT.TEX*

Use the **File>Make *project.tex*** to compile the open project and produce a *release version* of the Transall Application.

Prior to using this command you need to specify the project setting that control the release version. Refer to *Editing Build Settings* on page 306 for instructions on how to build a release version of the Transall Application.

EXIT

Use the **Exit** command to close Transall and return to the Windows Desktop.

To Quit Transall

1. Select **File>Exit**.

   If you've chaned the current project, a Transall message box displays.

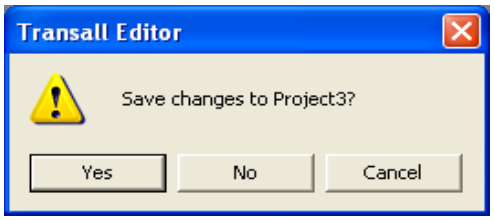


Figure 26: Save Changes to Project

2. To save the changes, click **Yes**. Otherwise, click **No**.

EDIT MENU

The Edit menu allows you to edit text in the active window and comprises the following menu options:

If you want to	See
Reverse the last action	*Undo* on page 64
Repeat the last action	*Redo* on page 64
Remove the selected text and copy it to the Windows Clipboard	*Cut* on page 64
Copy the selected text to the Windows Clipboard	*Copy* on page 65
Paste the contents of the Windows Clipboard into the project	*Paste* on page 65
Delete the selected document component	*Delete* on page 66
Select the entire contents of the Source window	*Select All* on page 66
Find a word or phrase in the Source window	*Find* on page 66
Replace a word/phrase in the Source window with another word/phrase	*Replace* on page 67
Find a specific component in the project	*Find in Project* on page 68
Replace a word/phrase in a project with another word/phrase	*Replace in Project* on page 71

UNDO

Use the Undo command to restore the last action you performed.

To Undo Your Most Recent Editing Action

- Select **Edit>Undo**.

  -or-

  Press **CTRL+Z**.

REDO

Use the Redo command to repeat the action for which you used Undo.

To Redo Your Most Recent Editing Action

- Select **Edit>Redo**.

CUT

Use the **Cut** command to move text you've selected onto the Windows Clipboard. Unlike the action of the **Copy** command, **Cut** removes the selected text from its original location.

To Move Text to the Clipboard

1. Select the text.

   1 Select **Edit>Cut**.

   -or-

   Click in the Standard Toolbar.

Transall places the text on the Windows Clipboard. You can now insert the text in other locations in the active document. You can also insert the text in other Windows applications. For details, see *Paste* on page 65.

COPY

Use the **Copy** command to place a copy of text you've selected onto the Windows Clipboard. Unlike the **Cut** command, **Copy** leaves the selected text in its original location.

To Copy Text to the Clipboard

1. Select the text.
2. Select **Edit>Copy**.

   -or-

   Click in the Standard Toolbar.

Transall copies the text to the Windows Clipboard. You can now insert the text in other locations in the active document. You can also insert the text in other Windows applications. For details, see *Paste* on page 65.

PASTE

Use the **Paste** command to copy text from the Clipboard to a location where you have clicked an insertion point.

To Paste Text from the Clipboard

1. Click an insertion point at the location where you want the text to be inserted.
2. Select **Edit>Paste**.

   -or-

   Click in the Standard Toolbar.

Transall copies the text from the Windows Clipboard. The text stays on the Clipboard until you replace it with text from the next **Cut** or **Copy** operation.

DELETE

Use the **Delete** command to remove a component from Component Explorer.

Tip If you want to delete text in a property field, highlight the text and press **DELETE**.

To Delete a Component

1. In the Component Explorer, select the component you want to delete.
2. Select **Edit>Delete**.

   -or-

   Click in the Standard Toolbar.

   A Transall message box displays.

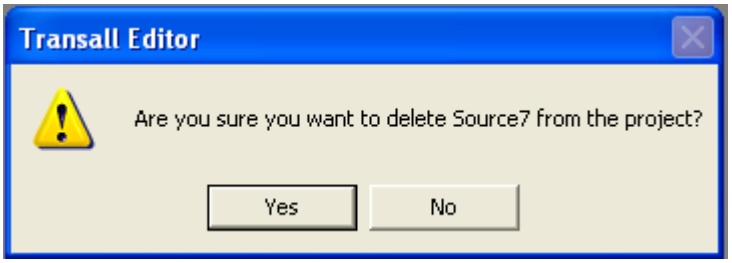


Figure 27: Delete Component

3. If you're certain you want to delete this component, click **Yes**; otherwise, click **No**.

SELECT ALL

Use the **Select All** command to select the entire contents of the Source window.

To Select All of the Text

1. Place an insertion point in the Source window.
2. Select **Edit>Select All**.

Transall highlights the entire contents of the Source window. You can use Cut (see *To Move Text to the Clipboard* on page 65) or Copy (see *To Copy Text to the Clipboard* on page 65) to manipulate the data.

FIND

Use the **Find** command to locate a word or phrase in the Source window.

To Find a Word or Phrase

1. Click an insertion point in the upper half of the Source window.
2. Select **Edit>Find**.

   -or-

Press **CTRL+F**.

The Find dialog box displays.

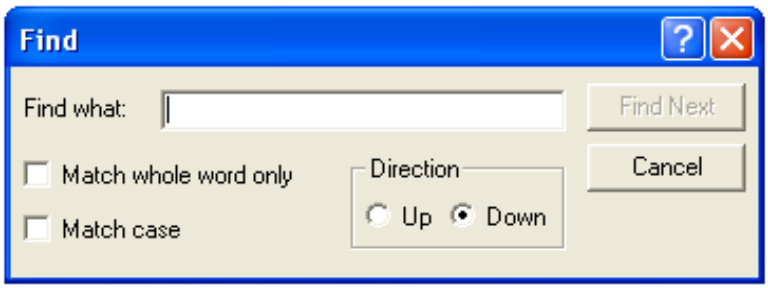


Figure 28: Find Dialog Box

3. In the Find what: text box, type the word or phrase you're trying to locate.
4. Enable the Match whole word only check box if you suspect this word might be contained in larger words (e.g., you search for *drive* and the procedure finds *driven*).
5. Enable the Match case check box if you want to locate the word or phrase exactly as you've typed it.
6. Click **Find Next** to initiate your search; otherwise, click **Cancel**.

Transall highlights the first occurrence of your search phrase.

REPLACE

Use the **Replace** command to search for an replace one phrase with another while in the Source window.

To Replace a Search Phrase

1. Select **Edit>Replace**.

   -or-

   Press **CTRL+H**.

   The Replace dialog box displays.

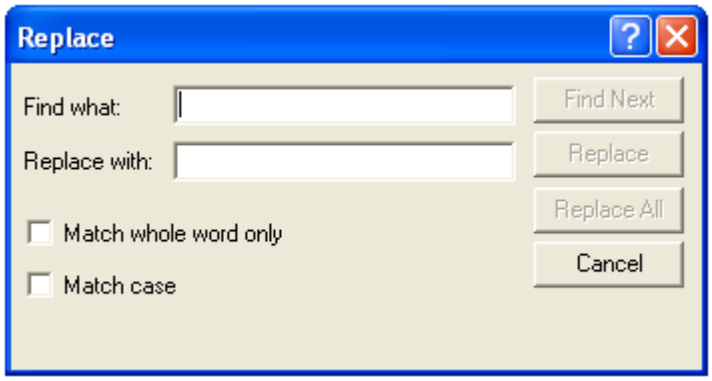


Figure 29: Replace Dialog Box

2. In the **Find what:** text box, type the word or phrase you're trying to locate.
3. In the **Replace with:** text box, type the word or phrase you want to replace the search phrase.
4. Enable the **Match whole word only** check box if you want to search exclusively on the word or phrase you've typed.
5. Enable the **Match case** check box if you want to locate the word or phrase exactly as you've typed it.
6. Click **Find Next** to initiate your search: then, do any of the following:

If you want to	Do this:
Replace only this instance of the search phrase	Click **Replace**.
Replace all instances of the search phrase	Click **Replace All**.
Cancel your search-and-replace task	Click **Cancel**.

FIND IN PROJECT

Use the **Find in Project** command to locate a word or phrase found in a script within the current project.

To Find a Word or Phrase in a Project

1. Select **Edit>Find in Project**.

   -or-

   Press **CTRL+SHIFT+F**.

The Project Search dialog box displays.

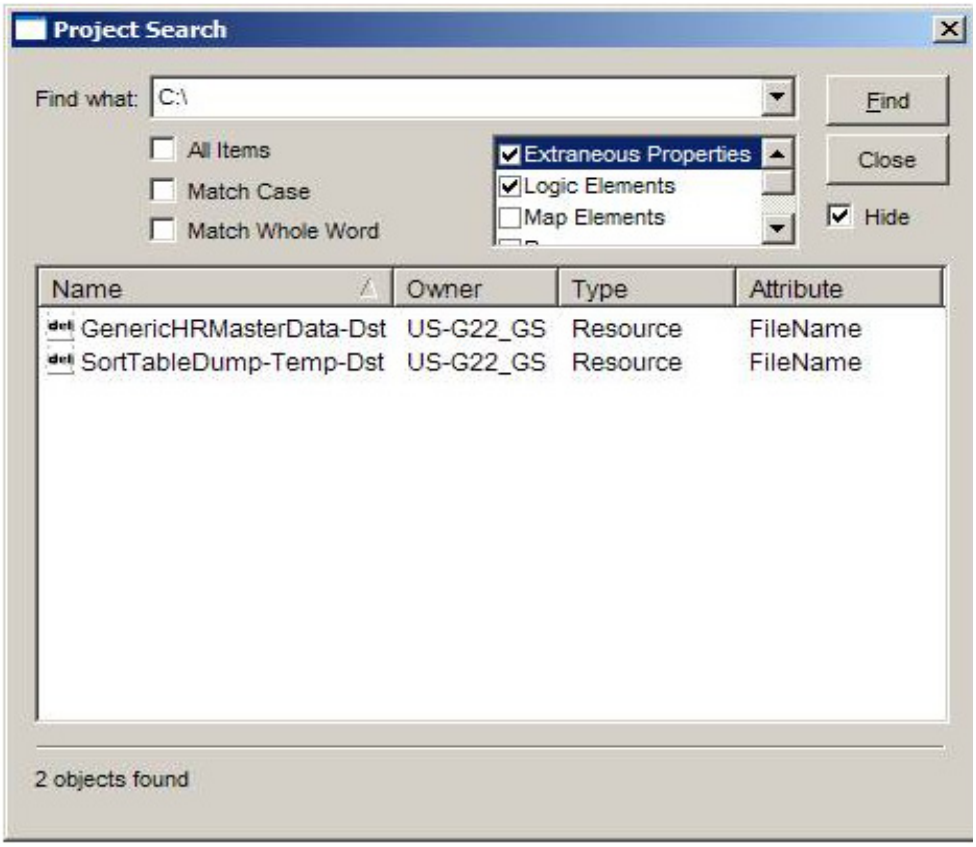


Figure 30: Find a Component in the Current Project

2. In the **Find what:** text box, type a word or phrase found in the component that you're trying to locate.
3. Leave the **All Items** box checked to search across all items

   -or-

   Remove the check mark and check the individual item(s) in the list on the right that you wish to search on. The first item in the list, **Extraneous Properties**, can be used to search for file-related properties such as file names as well as other miscellaneous properties.
4. Enable the **Match whole word** only check box if you want to search exclusively on the word or phrase you've typed.
5. Enable the **Match case** check box if you want to locate the word or phrase exactly as you've typed it.
6. Click **Find** to initiate your search.

A search on all scripts containing the term, “assign file handle”, returns the following lists of scripts.

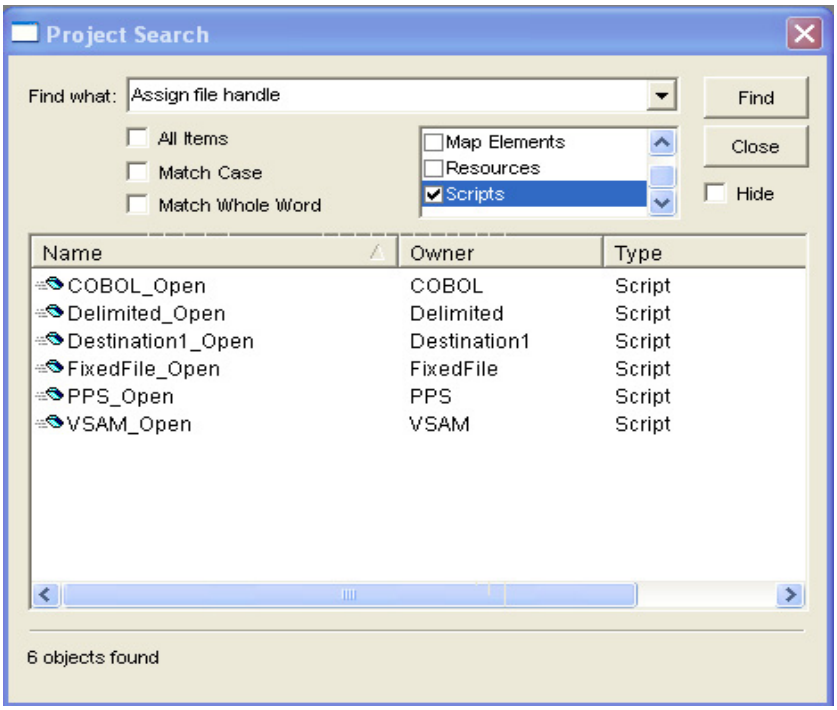


Figure 31: Results of a Project Search

7. Double-click on a script name in the Name column.

   The script opens in the Script Module Assistant with the first occurrence of the phrase in the script highlighted.

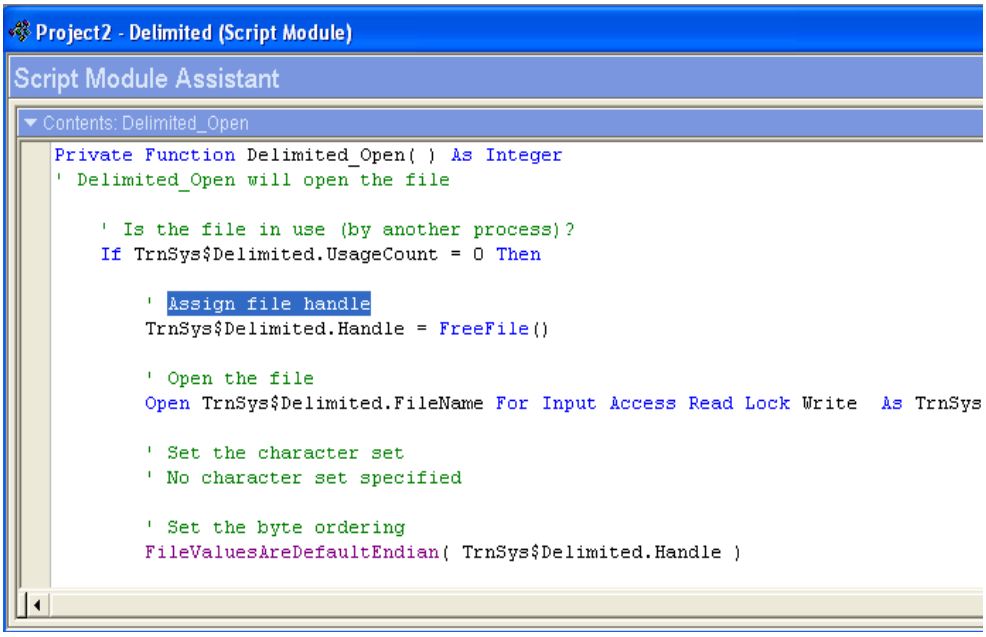


Figure 32: Open a Script from the Project Search Dialog Box

8. Click on a column header in the Project Search dialog box to sort the results in ascending/descending order by the selected column name.

   The **Find what:** text box contains a drop-down list which retains your last four searches.

REPLACE IN PROJECT

Use the **Replace in Project** command to find non-generated text in a project and replace each instance with different text. It DOES NOT include names or objects but instead mainly includes items such as user-entered scripts and file locations.

To Replace Text in a Project

1. Select **Edit>Replace in Project**.

   The Project Replace dialog box opens:

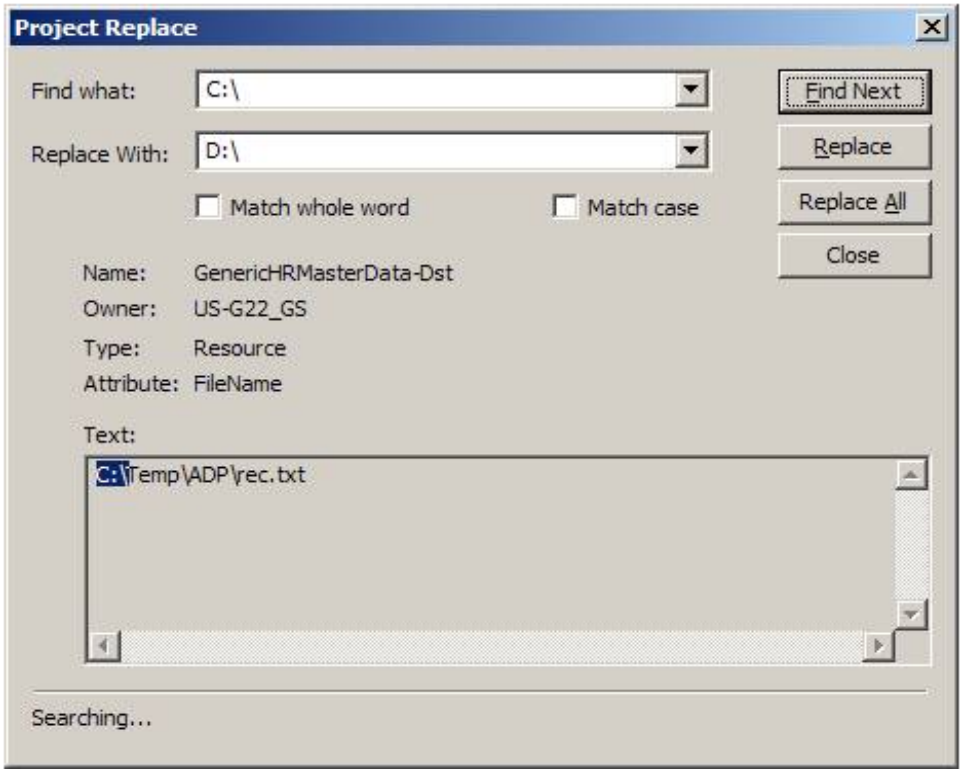


Figure 33: Replace non-generated Text in a Project

2. Type in an item in the **Find what:** text box.
3. Type in an item in the **Replace With:** text box.
4. Enable the **Match whole word** only check box if you want to search exclusively on the word or phrase you've typed.
5. Enable the **Match case** check box if you want to locate the word or phrase exactly as you've typed it.

6. Select **Replace** to replace each instance individually or **Replace All** to replace each instance all at once.

VIEW MENU

The View menu comprises the following menu options:

If you want to	See
Display the Component Explorer	*Component Explorer* on page 72
Display the Component Inspector	*Component Inspector* on page 73
Display the Output Bar	*Output Bar* on page 75
Display the Logic Bar	*Logic Bar* on page 76
Display the Resource Bar	*Resource Bar* on page 77
Display the SQL Bar	*SQL Bar* on page 78
Display the Transall Editor's three debugging control bars	*Debug* on page 78
View the breakpoints you have set for the current project	*Breakpoints* on page 80
Display/hide or change the appearance of the File and Debug toolbars	*Toolbars* on page 80

COMPONENT EXPLORER

Use the **Component Explorer** command to open the view of the Component Explorer. The Component Explorer is the central location for accessing, viewing, adding, and deleting the components in your project.

To Display the Component Explorer

- Select **View>Component Explorer**.

  -or-

  Click in the Standard Toolbar.

Transall displays the Component Explorer.

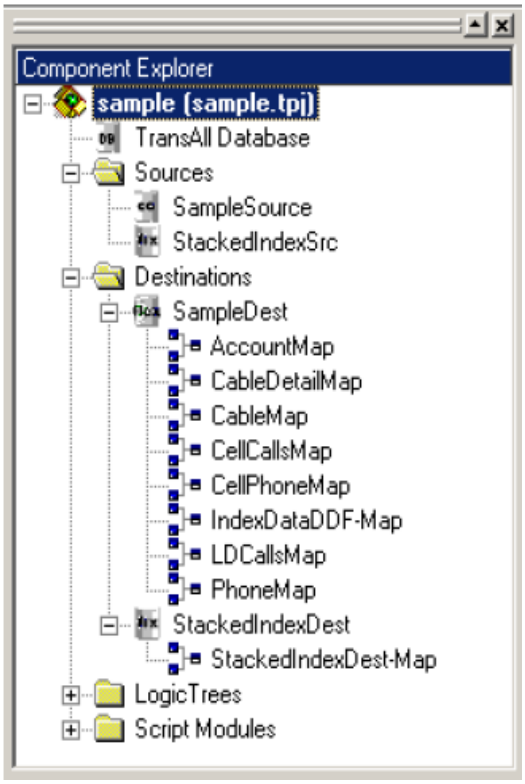
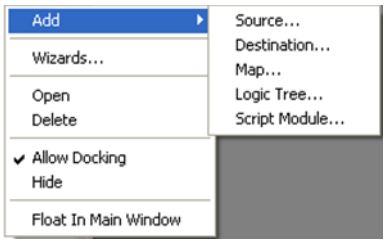


Figure 34: Open Component Explorer

If you move the cursor over the Component Explorer and right-click, the following menu opens:



Refer to *Component Explorer Control Bar* on page 46 for a description of how to use the Component Explorer.

COMPONENT INSPECTOR

Use the **Component Inspector** command to open the Component Inspector bar. The Component Inspector allows you to examine and edit the detail properties of a component selected from in the Component Explorer.

To Display the Component Inspector

1. Open the Component Explorer.
2. Select a component by double-clicking a name in the Component Explorer.

3. Select **View>Component Inspector**.

   -or-

   Click in the Standard Toolbar.

The Component Inspector opens and displays the detail properties of the selected component:

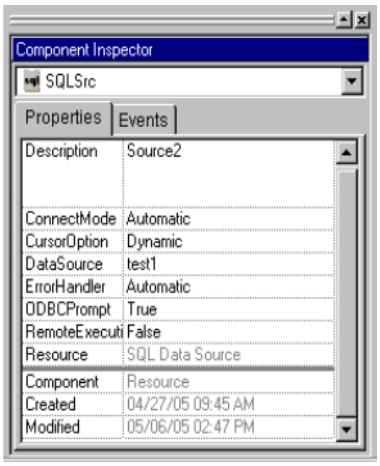


Figure 35: Open Component Inspector

The Component Inspector has two tabs:

- **Properties** - this is the default view. The Properties tab displays the component's properties and allows you to edit the detailed values of the properties.
- **Events** - the Events tab displays the following:
  - Names of events that pertain to the component; each event name corresponds to a built-in Transall Script subroutines that you can customize for that component.
  - Names of built-in methods that Transall generates for the component. Some of these generated script subroutines are read only and can not be modified or customized by the user, some of the subroutines are available to be modified and customized by the user. The subroutines that are available for user modification are usually the "On" events subroutines such as "OnOpenAfter" or "OnWriteAfter"

For a complete description on how to use the Component Inspector, refer to *Component Inspector Control Bar* on page 48.

OUTPUT BAR

The **View>Output Bar** command lets you manually open the Output Bar.

The Output Bar automatically opens whenever you compile a project using the **Project>Compile** command or synchronize the data source or links in a project using the **Project>Synchronize** command. The Output bar displays messages that show the progress and results of the compilation or synchronization process as well as any errors that may occur.

To Display the Output Bar

- Select **View>Output Bar**.

Transall displays the Output Bar.

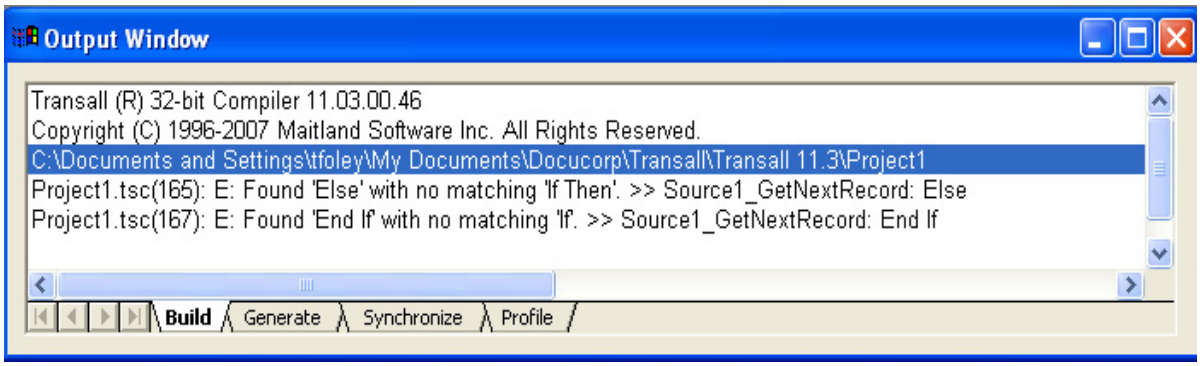


Figure 36: Open Output Bar

The Output Bar has four tabs:

- **Build** - Displays the results of the compilation process. Any errors that may have occurred appear in this tab. It also indicates whether the project successfully compiled.
- **Generate** - Displays status messages showing the progress of the compilation process.
- **Synchronize** - Displays the results of the data source or link synchronization process. Any error messages appear in this tab. It also indicates whether the synchronization was successful.
- **Profile** - Displays the Profile Report. The profile report is in two sections. The first section lists up to 100 of the most time consuming lines of script from the project, sorted from most time-consuming to least time-consuming. The second section lists up to 100 of the most-executed lines of script from the project, sorted from most-executed to least-executed.

  To display the profile report, check the "Runtime Profiler" checkbox in the Debug tab of the **Project>*project* Settings** dialog box.

  For further information, refer to Step 4 under *To Specify Command Line Debugging* on page 315.

For further information on compiling a project, refer to *Debugging and Deploying Transall Applications* on page 305.

For further information on the synchronization process, refer to *Project Sharing Details* on page 351.

LOGIC BAR

Use the **Logic Bar** command to display a set of icons that are used in building a logic tree.

To Display the Logic Bar

- Select **View>Logic Bar**.

  -or-

  Press **Ctrl+Alt+L**.

Transall displays the Logic Bar.

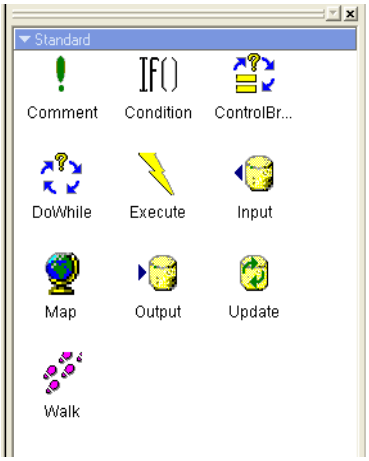


Figure 37: Open Logic Bar

There are three sets of Logic Tree icons. Use the control in the Logic bar's top border to select from the following sets of LogicTree instructions:

- **Standard** - The **Standard** set of instructions includes the Comment, Condition, ControlBreak, DoWhile, Execute, Input, Map, Output, Update, and Walk instructions. Every new Logic Tree will use at least some of these instructions.
- **Documaker FP -** The **Documaker FP** set of instructions includes the AddForm, AddFormsLibrary, AddTag, EndMergeSet, MergeSetBreak, SetEffectiveDate, SetRulebase, StartMergeSet, and SubmitVRF instructions. Select the Documaker FP set of instructions if you are coding this Logic Tree to produce output data records for a Documaker FP File (VRF) Destination.
- **Documaker FP Plus -** The **Documaker FP Plus** set of instructions includes the FpAddTag, FpComment, FpDataGroup, FpDataHeader, FpFooter, FpForm, FpHeader, FpKeepOnSamePage, FpLayout, and FpPageBreak instructions. Select the Documaker FP Plus set of instructions if you are coding this Logic Tree to produce output data records for a Documaker FP Plus (VRF) Destination.

RESOURCE BAR

Use the **Resource Bar** command to display a list of resources contained within the current project. The Resource Control Bar is automatically invoked and closed, by default, when a Map Assistant is opened and closed. You can change the Default behavior via **Tools>Settings** by and unchecking "Automatically show Resource control bar".

To Display the Resource Bar

- Select **View>Resource Bar**.

  -or-

  Press **Ctrl+Alt+R**.

Transall displays the Resource Bar.

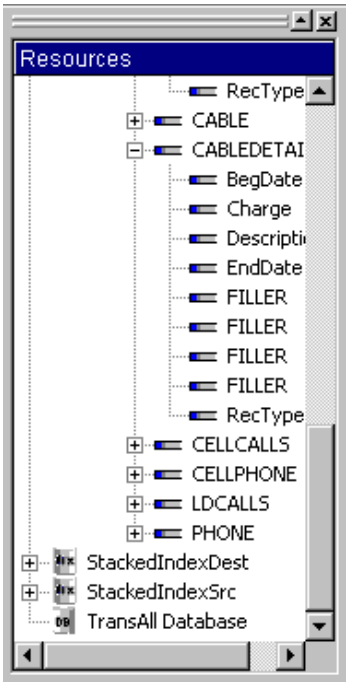


Figure 38: Open Resource Bar

The Resource control bar contains the following context menu items:

- **Sorted**—provides control concerning displayed resources. You can Sort the display of the tree listing in alphabetical order, or deselect Sort and display the tree in physical order.
- **Flattened**—provides control concerning displayed resources. You can Flatten the hierarchy of the tree listing, or deselect Flattened and display the expanded tree listing.

Note Each time you start the Transall Editor, it reverts to the default settings of Sorted=On and Flattened=Off. Toggling either of these menu items causes a repopulation of the entire control bar.

- **Automap**—If a Map Assistant has focus and some item in the Resource Control Bar is selected, then Automap is enabled. If you're mapping a Source record to a target record that uses a one-to-one relationship, Automap attempts to assign map expressions by matching the field names.
- **Map Expression**— achieves the same results as dragging-and-dropping from the Resource control bar to the Target Expression field of a record. First, select a Target Expression field in the Map Assistant; then, right-click a Resource and select **Map Expression**. The Resource is displayed in the Target Expression column.

SQL Bar

Use the **SQL Bar** command to display a control bar that provides access to, and allows modification of, SQL Filter, and Join data. By default, this control bar opens and closes along with the DataSource Assistant, but you can uncouple it by using **Tools>Settings** and unchecking the **Automatically show SQL control bar** checkbox.

If the control bar contains Filter or Join expressions, you'll see a check mark on that tab.

To Display the SQL Bar

- Select **View>SQL Bar**.

  Transall displays the SQL Bar.

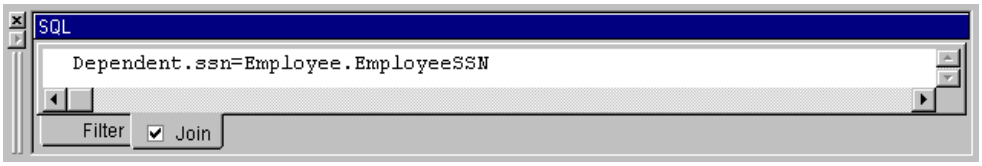


Figure 39: Open SQL Bar

If the control bar contains Filter or Join expressions, you'll see a check mark on that tab.

For further information on defining Join expressions, refer to *Defining Join Criteria for the Query* on page 166.

For further information on defining filters, refer to *Defining Filter Criteria* on page 168.

Debug

Use the **Debug** command to individually display the Transall Editor's three debugging control bars:

- Watch Window
- Variables Window
- Call Stack Window

To Display the Watch Window

- Select **View>Debug>Watch Window**.

Transall displays the Watch Window.

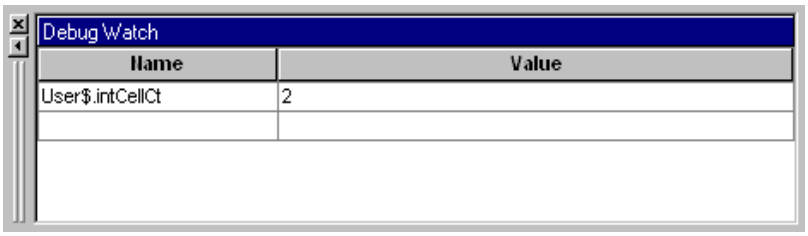


Figure 40: Open the Debug Watch Window

The Debug Watch bar displays the variables whose values you want to monitor, regardless of their location.

To Display the Variables Window

- Select **View>Debug>Variables Window**.

Transall displays the Variables Window.

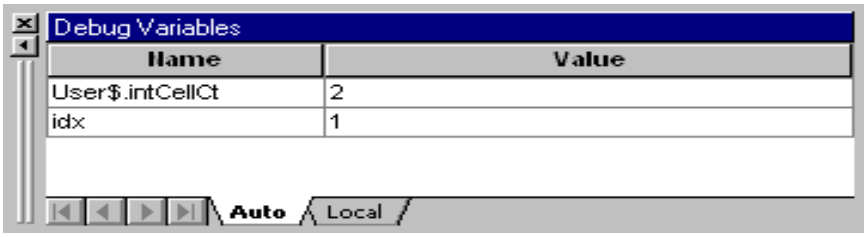


Figure 41: Open the Debug Variables Window

The Debug Variables bar displays both Auto and Local variables.

The Auto tab displays the variables, for the current and previous lines, that you're debugging at the current time. The Local tab displays all variables defined inside the function.

To Display the Call Stack Window

- Select **View>Debug>Call Stack Window**.

Transall displays the Call Stack window.

Figure 42: Open the Debug Call Stack Window

The Debug Call Stack bar displays the current business rule being processed. If you have nested rules, this bar displays the tree view and indicates the current rule.

For further information, refer to *Operating Transall under Debugging Control* on page 323 and *Viewing Debug Variables in the Variables and Watch Bars* on page 324.

BREAKPOINTS

Use the **Breakpoints** command to see the breakpoints you have set for the project. You can also use this command to remove individual breakpoints or all of the breakpoints you have set.

To View Breakpoints

- Select **View>Breakpoints**.

The View Breakpoints dialog appears:

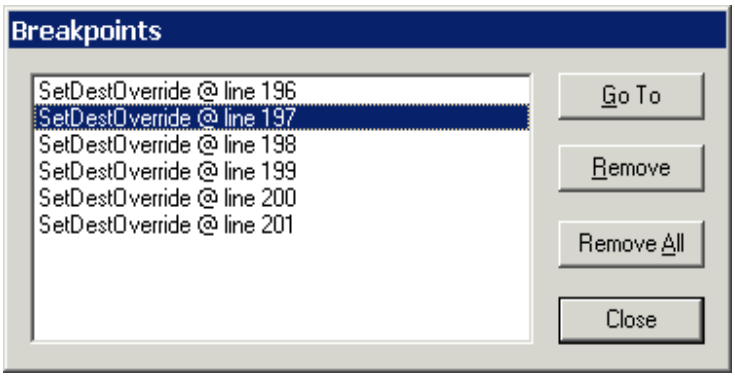


Figure 43: View Breakpoints

For further information on setting breakpoints in Transall, refer to *Breakpoints in Transall* on page 322.

TOOLBARS

Use the **Toolbars** command to toggle the view of the various toolbars (e.g., display or hide) and use the Customize option to adjust the buttons available on the editor's toolbars.

To Display/Hide the Standard Toolbar

- Select **View>Toolbar>Standard**.

  -or-

  Press **Ctrl+Alt+5**.

  If the Standard toolbar is already displayed then it will be hidden. If it is hidden, then it will be displayed.

To Display/Hide the Debug Toolbar

- Select **View>Toolbar>Debug**

  -or-

  Press **Ctrl+Alt+D**.

  If the Debug toolbar is already displayed then it will be hidden. If it is hidden, then it will be displayed.

To Display the Customize Dialog Box

- Select **View>Toolbar>Customize**

The Customize dialog box displays.

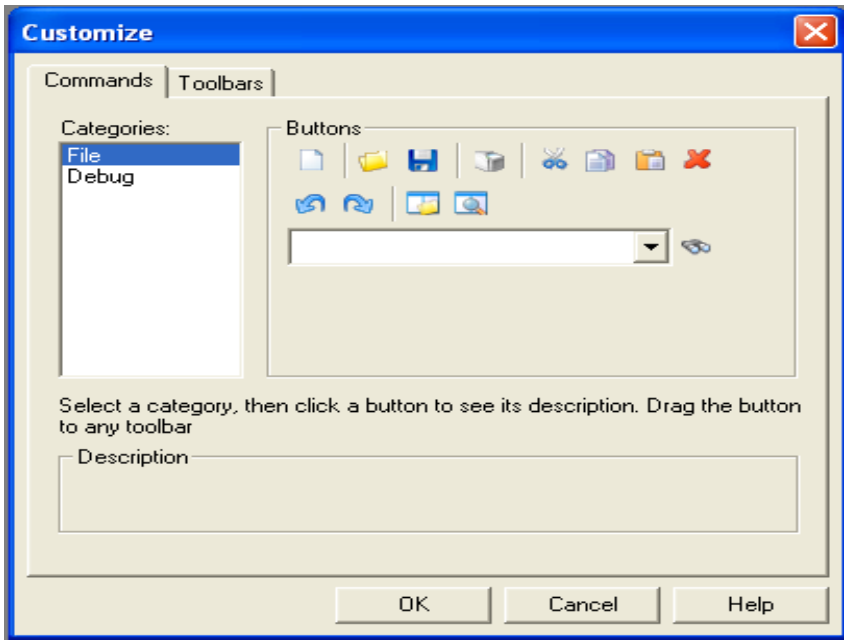


Figure 44: Open the Customize Dialog Box

The Customize dialog box has two tabs:

- **Commands** - The Commands tab shows the buttons under the **File** and **Debug** toolbars. You can see a description of a button by clicking on it.

- **Toolbars** - The Toolbars tab allows you to change the look of the buttons in the File and Debug toolbars and add a new toolbar to the interface. The options on this tab lets you turn off/on tooltips, give the buttons a 3D appearance, and increase the size of the buttons.

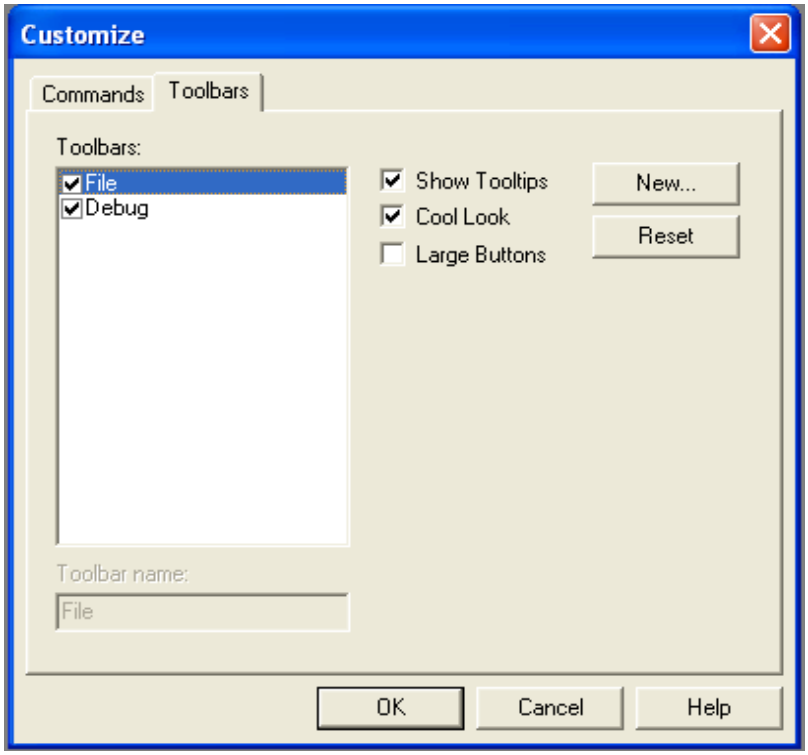


Figure 45: Open Toolbars Tab

To Add a New Toolbar

1. Select the **Toolbars** tab.
2. Click **New**.

   The New Toolbar dialog box appears.

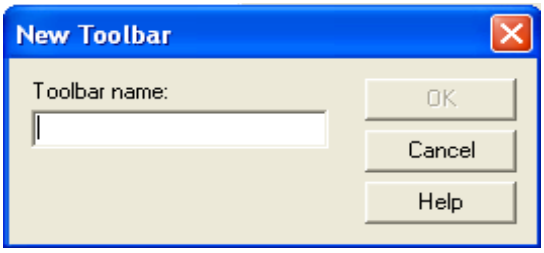


Figure 46: New Toolbar Dialog Box

3. Enter a name for the new toolbar and click **OK**.
4. The new toolbar will appear in the corner of the screen.
5. Go to the **Commands** tab.
6. Drag any button under the **Buttons** section on the right side of the **Commands** tab to the new toolbar as desired.

PROJECT MENU

The Project menu comprises the following menu options:

If you want to	See
Create a new data source	*Add Source* on page 83
Create a new destination	*Add Destination* on page 84
Create a new map	*Add Map* on page 85
Create a new logic tree	*Add Logic Tree* on page 86
Add a new script module to your project	*Add Script Module* on page 88
Invoke a series of wizards with which to create destinations and Transall tables	*Wizards* on page 89
Synchronize all data sources and links in a project	*Synchronize* on page 97
Compile a version of the Transall Application for debugging	*Compile* on page 101
Use this command to go to the next error in a script	*Next Error* on page 101
Use this command to go to a previous error in a script	*Previous Error* on page 101
Customize the Transall default settings	*Project Settings* on page 101

ADD SOURCE

Use the **Add Source** command to add a data source to a project.

To Add a Data Source to a Project

- Select **Project>Add Source**

  -or-

  Select **Add>Source** from the Component Explorer's context menu.

The Add Source dialog displays.

Figure 47: Add Source Dialog

Refer to *Creating Sources and Destinations* on page 120 for an explanation of the properties available in this dialog box and the complete steps for creating a data source.

ADD DESTINATION

Use the **Add Destination** command to add a destination to a project.

To Add a Destination to a Project

- Select **Project>Add Destination**

  -or-

  Select **Add>Destination** from the Component Explorer's context menu.

The Add Destination dialog displays. The properties in this dialog box are identical to the Add Source dialog.

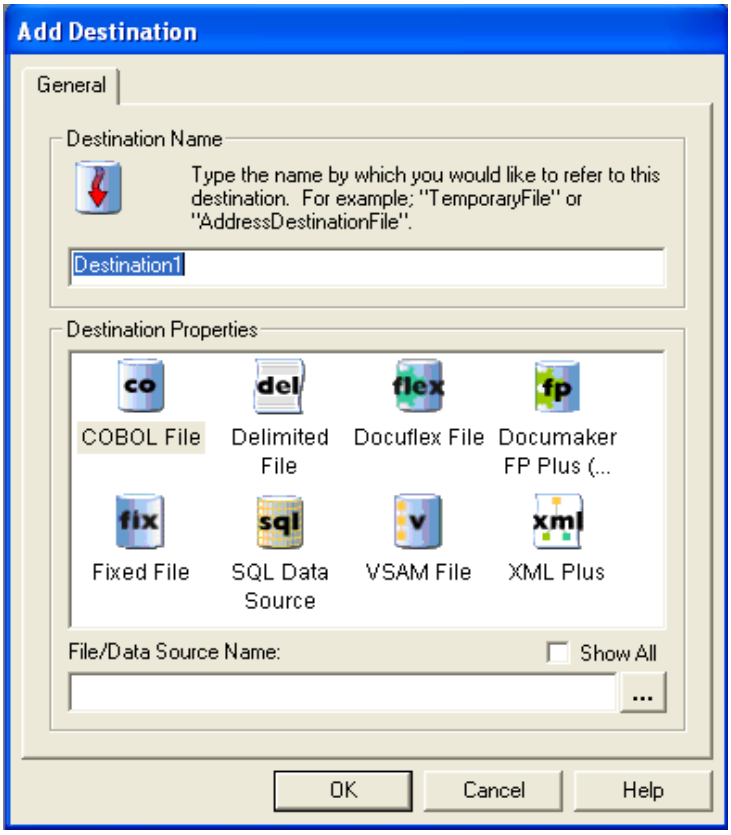


Figure 48: Add Destination Dialog

The steps for creating a destination are identical to the steps for creating a data source. Refer to *Creating Sources and Destinations* on page 120 for the complete steps for creating a destination.

ADD MAP

Use the **Add Map** command to create a new map.

To Add a Map to a Project

- Select **Project>Add Map**

  -or-

  Select **Add>Map** from the Component Explorer's context menu.

The Add Map dialog displays.

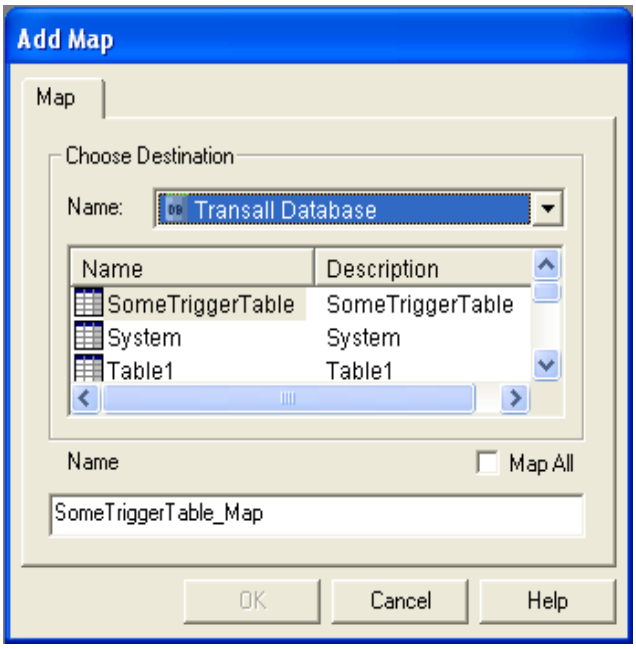


Figure 49: Add Map Dialog

Refer to *Working with Maps* on page 269 for the complete instructions on how to create a map.

ADD LOGIC TREE

Use the **Add LogicTree** command to create a new Logic Tree.

To Add a Logic Tree to a Project

1. Select **Project>Add Logic Tree**

   -or-

   Select **Add>Logic Tree** from the Component Explorer's context menu.

The Add Logic Tree dialog displays.

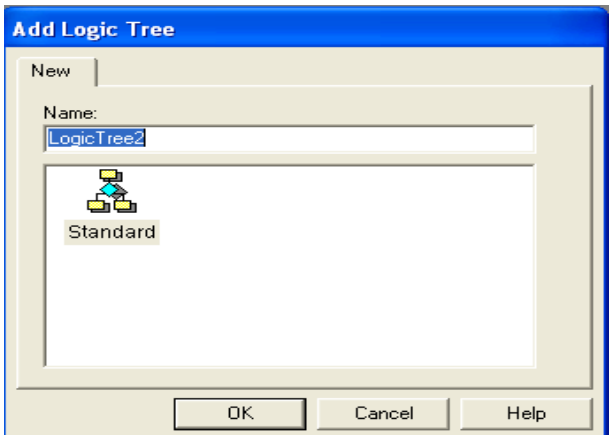


Figure 50: Add Logic Tree Dialog

2. Enter a name for the new Logic Tree.
3. Select the **Standard** type and press **OK**.

   The Transall Editor displays the LogicTree Assistant for the new Logic Tree and opens the Logic control bar that contains a set of icons.

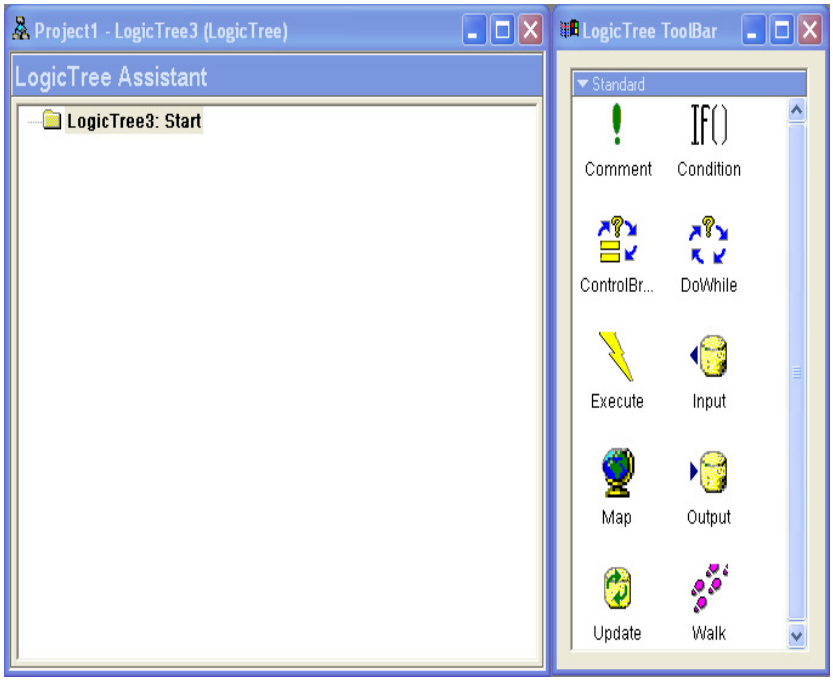


Figure 51: Logic Tree Assistant

Refer to *Adding a Logic Tree* on page 289 for a complete description of this screen and instructions on how to build a logic tree.

ADD SCRIPT MODULE

Use the **Add Script Module** command to a project.

To Add a Script Module to a Project

1. Select **Project>Add Script Module**

   -or-

   Select **Add>Script Module** from the Component Explorer's context menu.

   The Add Script Module dialog displays.

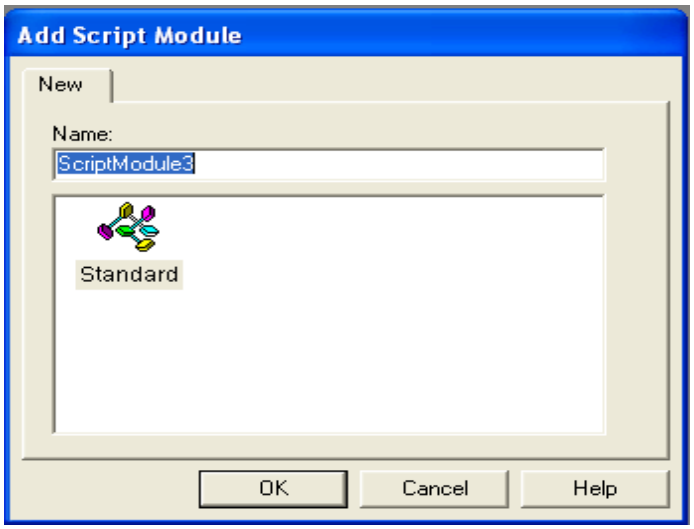


Figure 52: Add Script Module

2. Enter a name for the new script module.
3. Select the **Standard** type and press **OK**.

The Transall Editor opens the Script Module Assistant for the new Script Module in the workspace area. By default a new Script Module's Script Module Assistant presents the Module's Declarations section.

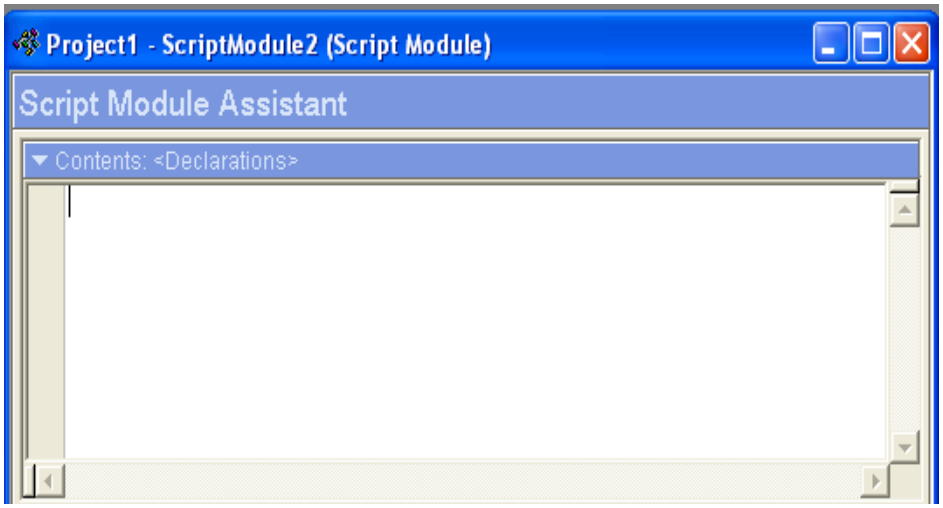


Figure 53: Script Module Assistant

Refer to *Creating a Script Module* on page 340 for complete instructions on how to code the Declarations section, add/edit a script, and delete a script.

WIZARDS

The **Wizards** command opens the Transall Wizards dialog box which provides four separate wizards with which to create destinations and Transall tables.

To Open the Transall Wizards Dialog Box

- Select **Project>Wizards**

  The Transall Wizards dialog box opens.

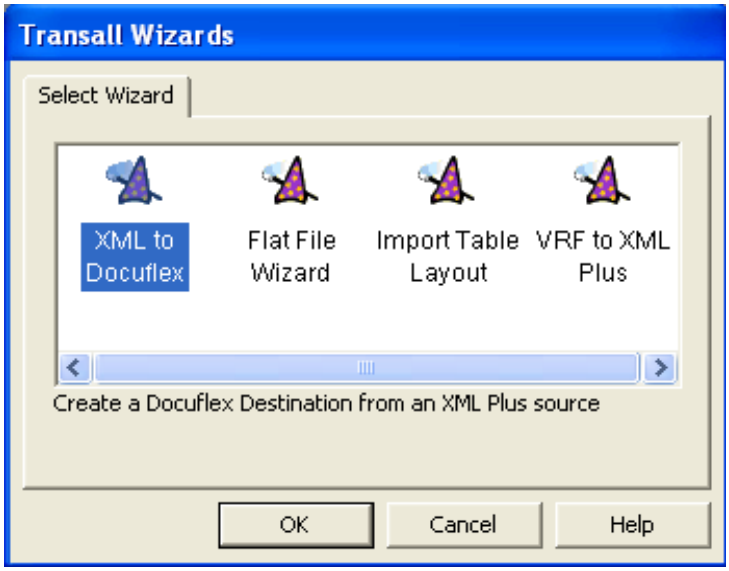


Figure 54: Wizards Dialog

Use this Wizard	To
XML to Docuflex	Create a Docuflex Destination from an XML Plus source
Flat File Wizard	Create a flat file destination from a flat file source
Import Table Layout	Create an internal Transall table based on an external table layout
VRF to XML Plus	Create an XML Plus destination from a Documaker FP Plus VRF destination

XML to Docuflex

Use the **XML to Docuflex** wizard to create a docuflex destination from an XML Plus data source.

To Create a Docuflex Destination from an XML Plus Source

Before you use this wizard, you must create an XML Plus data source. See *Add Source* on page 83 and *Creating Sources and Destinations* on page 120 for the complete steps to create a data source.

1. In the Transall Projects dialog box, select "XML to Docuflex" and click **OK**.

   The Destination Creation wizard opens:

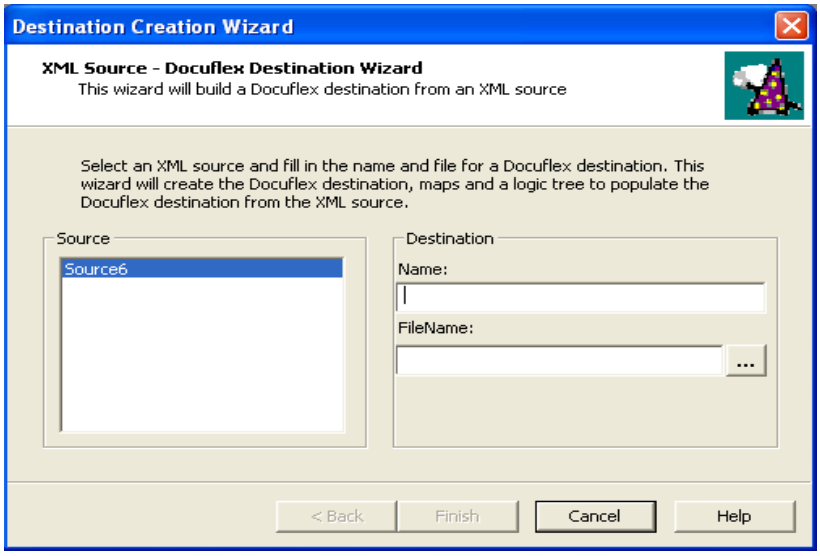


Figure 55: XML PLUS Source - Docuflex Destination Wizard

2. Highlight an XML source in the Source list.
3. Type in a name for the Docuflex destination.
4. Click ... to select a file name for the Docuflex destination.
5. Click **Finish**.

The wizard creates the Docuflex destination, maps, and logic tree to populate destination from the XML source.

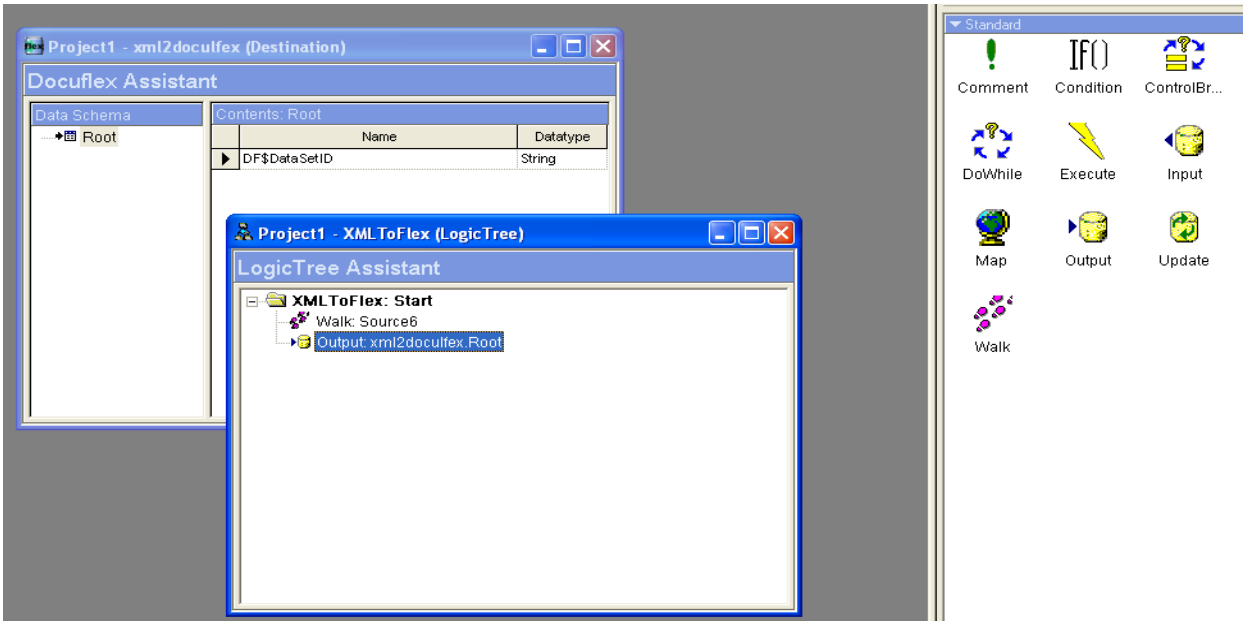


Figure 56: Docuflex Destination and Maps

Flat File Wizard

Use the **Flat File Wizard** to create a flat file destination from a flat file source.

To Create a Flat File Destination from a Flat File Source

Before you use this wizard, make sure that you have already created a Flat File data source.

See *Add Source* on page 83 and *Creating Sources and Destinations* on page 120 for the complete steps for creating a data source.

1. In the Transall Projects dialog box, select “Flat File Wizard” and click **OK**.

The Destination Creation wizard opens:

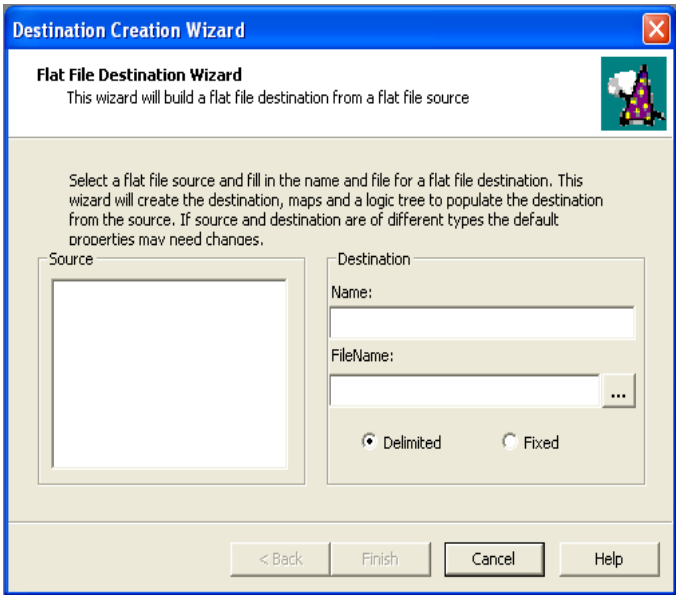


Figure 57: Flat File Destination Wizard

2. Highlight a flat file source in the Source list.
3. Type in a name for the flat file destination.
4. Click ... to select a file name for the flat file destination.
5. Select the **Delimited** or **Fixed** radio button.
6. Click **Finish**.

   The wizard creates the destination, maps, and logic tree to populate destination from the XML source.

Import Table Layout

Use the **Import Table Layout** wizard to create an internal Transall table based on an external table layout.

To Create an Internal Transall Table Based on an External Table Layout

1. In the Transall Projects dialog box, select “Import Table Layout” and click **OK**.

The Table Import wizard opens:

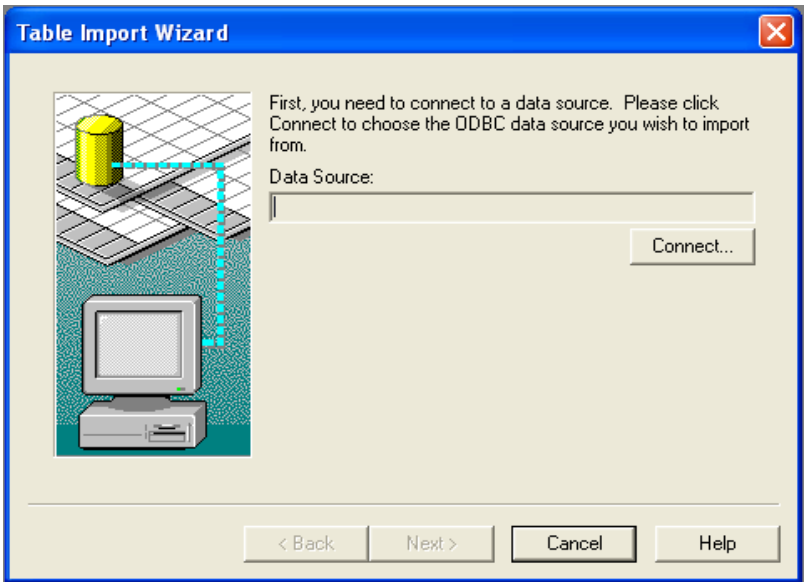


Figure 58: Locate and Connect to an ODBC Data Source

2. Click the **Connect** button to locate and select an ODBC data source.
3. Click the **Next** button.

   The following screen appears:

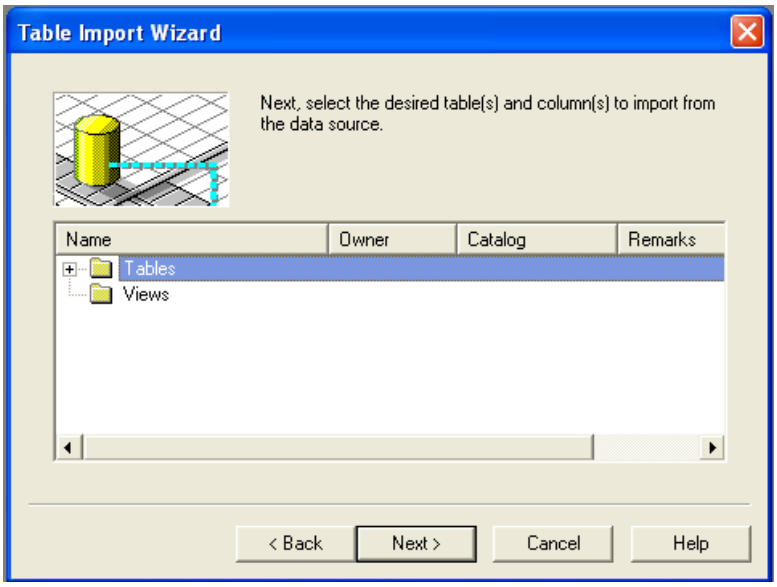


Figure 59: Select the Tables and Columns to Import from the Data Source

4. Expand the Tables folder and select the tables and columns that you want to import from the data source.
5. Click the **Next** button.

The following screen appears:

Figure 60: Display Warning Messages During the Import

6. Leave the **Yes** radio button checked and click the **Next** button.

   A summary screen showing your choices appears:

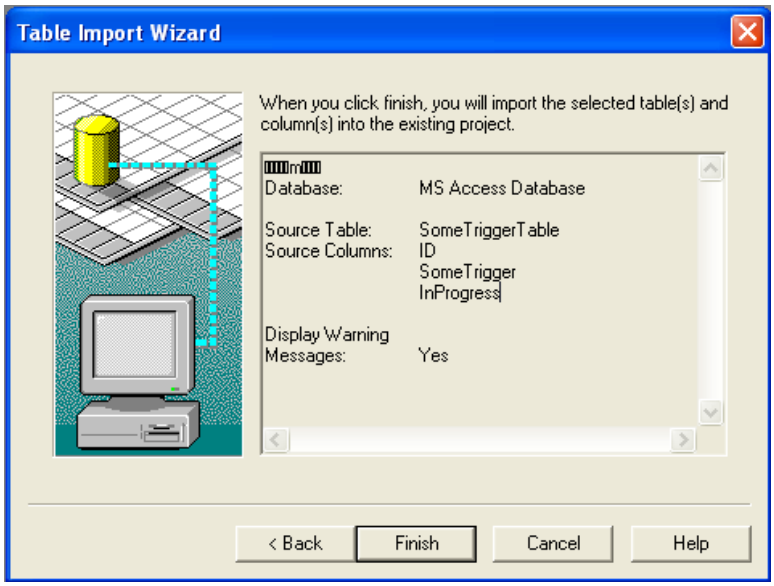


Figure 61: Summary Screen

7. Click the **Finish** button to start the import.

The Database Assistant appears containing the newly imported tables and columns:

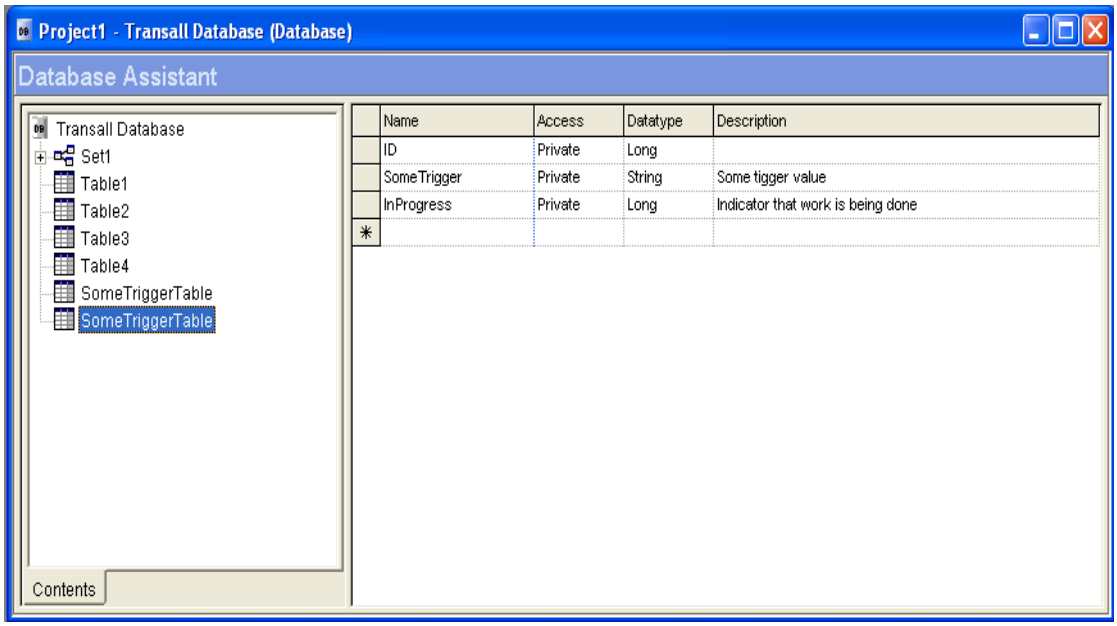


Figure 62: Database Assistant with Imported Tables and Columns

Refer to *Working with the Transall Database* on page 279 for a complete description of how to work with the components in a Transall database.

VRF to XML Plus

Use the **VRF to XML Plus** wizard to create an XML Plus destination from a Documaker FP Plus VRF destination.

To Create an XML Plus Destination from a Documaker FP Plus (VRF) Destination

Before you use this wizard, you must create a Documaker FP Plus (VRF) destination. See *Add Destination* on page 84 and *Creating Sources and Destinations* on page 120 to learn the complete steps for creating a data destination.

1. In the Transall Projects dialog box, select “VRF to XML Plus” and click **OK**.

The Destination Creation wizard opens:

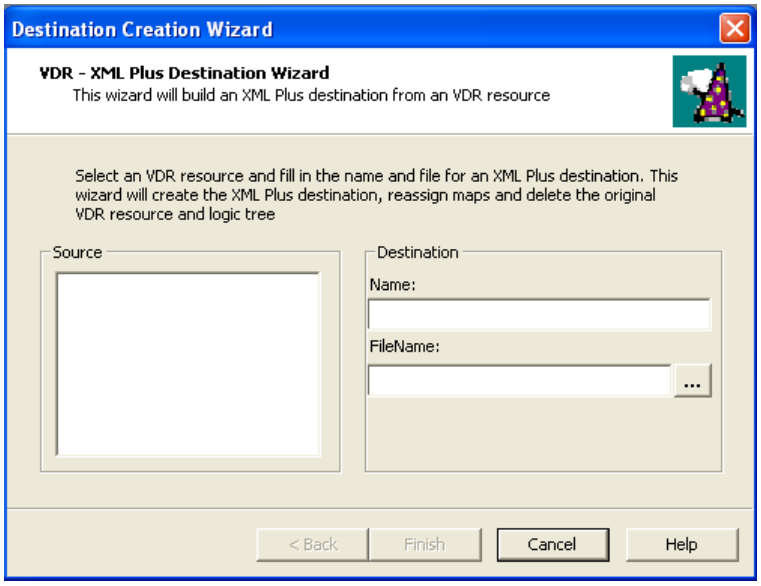


Figure 63: VDR - CML Plus Destination Wizard

2. Highlight a VDR source in the Source list.
3. Type in a name for the XML Plus destination.
4. Click ... to select a file name for the XML Plus destination.
5. Click **Finish**.

   The wizard creates the XML Plus destination and displays it in the XML Plus Assistant.

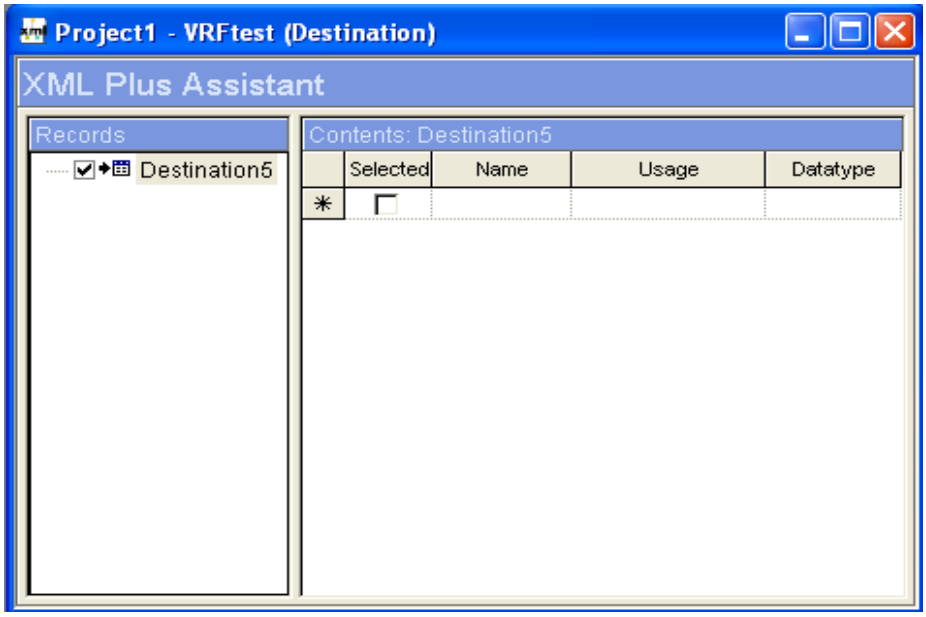


Figure 64: XML Plus Assistant

Refer to *XML Plus Data Source* on page 225 for a complete explanation of how to work with XML Plus.

SYNCHRONIZE

Use the **Synchronize** menu items to synchronize/view all linked data sources and other linked components in a project.

Links allow data sources and other components to be shared between projects. Before you can use the commands under this menu you must first create the links using the **File>Share** command.

For further information on using the **File>Share** command, refer to *Share* on page 62.

For a comprehensive overview of project sharing, refer to *Project Sharing Details* on page 351.

Use	If you want to
Data Sources	Synchronize all linked data sources
Links	Synchronize all links
View Links	View, update, and synchronize a link
View Queries	View all queries in the linked project(s) and synchronize the data sources or destinations containing the queries
DMG Report	Determine which Documanage documents will supply link requirements.

Data Sources

Use the **Data Sources** command to synchronize all linked data sources.

To Synchronize Data Sources

- Select **Project>Synchronize>Data Sources**

  All data sources in the shared project are synchronized and the results are displayed in the Output Bar's Synchronize tab.

Links

Use the **Links** command to synchronize all links in the shared projects.

To Synchronize Links

- Select **Project>Synchronize>Links**

  All existing links in the shared project are synchronized and the results are displayed in the Output Bar's Synchronize tab.

View Links

Use the **View Links** command to view, open a source project and edit any linked components it contains, and synchronize all linked data sources.

To View Links

- Select **Project>Synchronize>View Links**

  The Links dialog box opens. The Links dialog box not only displays all linked components in the current project but it also lets you synchronize and edit individual links.

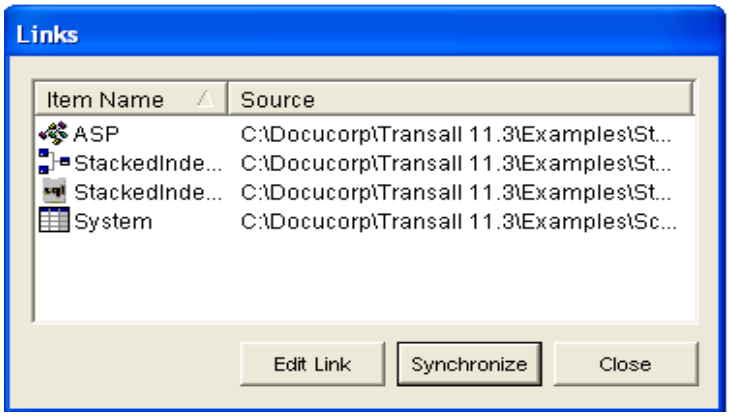


Figure 65: Display Links in Current Project

To Edit a Link

Linked components can still be edited in the source project. The **Edit Link** button allows you to absorb these changes to the source project into the target project.

1. Open the source project, make your changes, and save the project.
2. Open the target project containing the links to the source project.
3. Open the Links dialog box using **Project>Synchronize>View Links**.
4. Click on a link and select **Edit Link**.

To Synchronize a Link

1. Click on an item in the list.
2. Select **Synchronize**.

   The selected link will be synchronized and the results displayed in the Output Bar's Synchronize tab.

The Select Link Directory box opens. This box shows the directory where the source project is located.

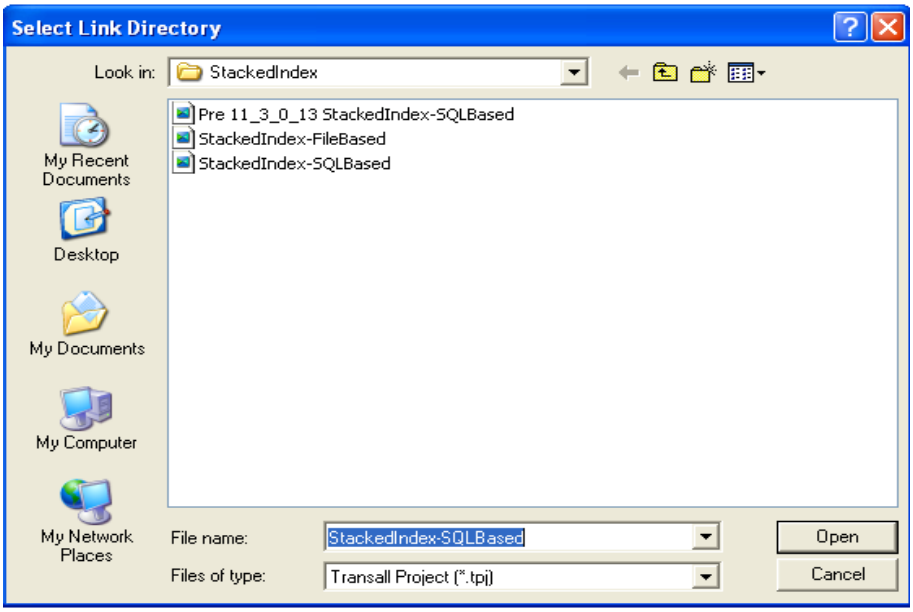


Figure 66: Display Source Project

3. Select the source project in the **File name:** box and select **Open**. A message box appears and ask you to confirm replacing the current linked project with the updated project.
4. Select **Yes** to make the change.

View Queries

Use the **View Queries** command View all queries in the linked project(s) and synchronize the data sources or destinations containing the queries

To View Queries

Use the **Project>Synchronize>View Queries** command to view any queries in a linked project and synchronize the data source or destination containing the queries. If the shared projects do not contain any queries then this menu item will be grayed out.

- Select **Project>Synchronize>View Queries.**

The Query Synchronize dialog box appears. This box lists all existing queries within the shared projects.

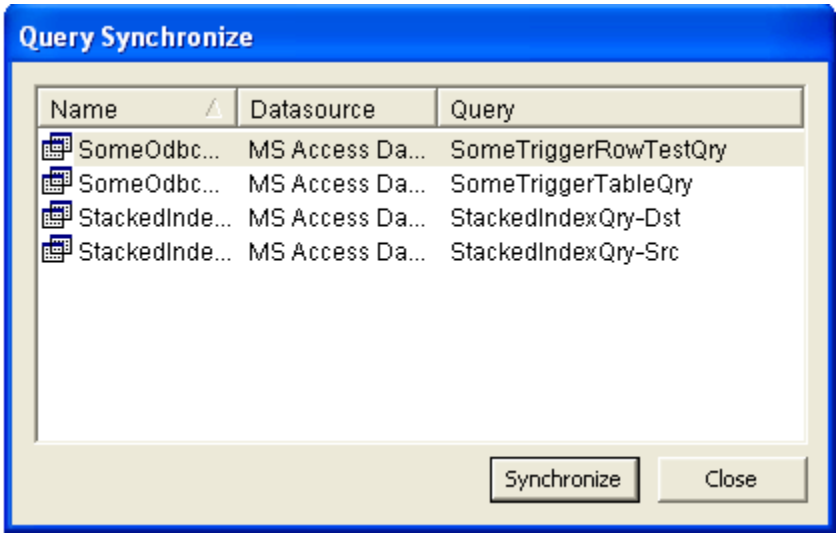


Figure 67: Queries within Shared Projects

To Synchronize a Link

1. Highlight a link in the list.
2. Select **Synchronize**.

   The selected link will be synchronized and the results displayed in the Output Bar's Synchronize tab.

DMG Report

Use the **DMG Report** command to determine which Documanage documents will supply link requirements.

To Check for Documanage Links

The DMG Report indicates which Documanage documents will supply link requirements.

This menu item only appears if you have checked the "Enable Documanage Interface" in the **Documanage** tab under **Tools>Options**.

- Select **Project>Synchronize>DMG Report**

  The Transall Editor checks for Documanage links and displays the results in the Output Bar's Synchronize tab.

COMPILE

Use the **Compile** command to produce a version of the Transall Application for debugging.

To Compile a Project

- Select **Project>Compile**

  -or-

  Press **F7**.

The results will be displayed in the Output Bar's Build tab.

Refer to *Running a Transall Application in Debug Mode* on page 321 for further details on creating a debugging version of the open project's Transall Application.

NEXT ERROR

Use this command to go to the next error in a script.

PREVIOUS ERROR

Use this command to go back to a previous error in a script.

PROJECT SETTINGS

Use the ***Project*** **Settings** command to open the Project Settings dialog box. This dialog box allows you to customize the default settings of Transall.

To Specify Project Settings

- Select **Project>*project* Settings**

The Project Settings dialog displays.

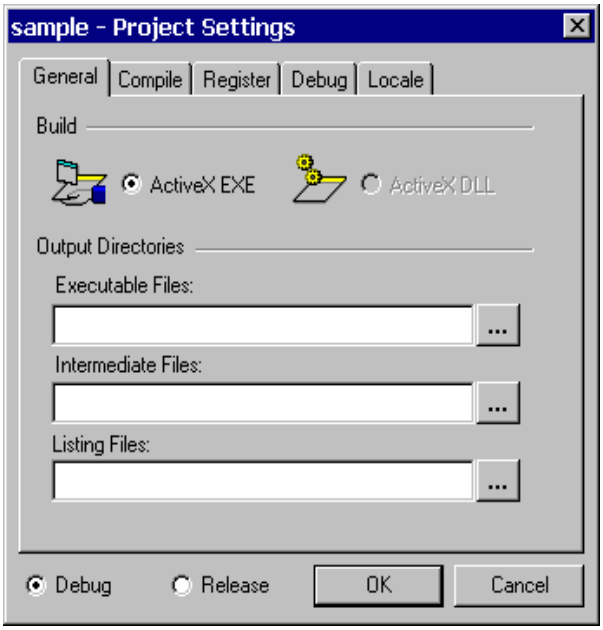


Figure 68: Project Settings Dialog Box

The Project Settings dialog box contains five tabs:

- **General** - Specifies the type of build and directories for the ancillary files.
- **Compile** - Specifies how the Transall Editor produces the Transall Application's executable file and other auxiliary files.
- **Register** - Specifies how the Transall Editor registers the Transall Application's executable file in the Windows Registry.
- **Debug** - Specifies the debug settings for a debug version of the Transall Application.
- **Locale** - Specifies the major language of your geographical region and whether to use the default Windows formatting symbols for currency, dates, and times.

Refer to *Using the Project Settings Dialog and Tabs* on page 307 for a complete description of the contents of each of these tabs.

DEBUG MENU

The Debug menu comprises the following menu options:

If you want to	See
Begin debugging your document	*Go* on page 103
Cease debugging your document	*Stop Debugging* on page 103
Pause debugging at the current point	*Break* on page 104
Process the current member and stop at the next member, regardless of level	*Step Into* on page 104
Process all the children of the current member and stop at the next member of the same level (i.e., sibling)	*Step Over* on page 104
Run the debugger until it reaches the point in the source code where the cursor is set.	*Run to Cursor* on page 105
Set/Remove a breakpoint	*Toggle Breakpoint* on page 105
Clear any and all breakpoints	*Clear All Breakpoints* on page 105
Locate a specific line number in a script.	*External Runtime Error* on page 106

For further information on the Transall Debug process, refer to *Running a Transall Application in Debug Mode* on page 321.

GO

Use the **Go** command to start the debugging utility.

To Start Debugging

- Select **Debug>Go**.

  -or-

  Click in the Debug Toolbar.

  -or-

  Press **F5**.

  For a detailed description of the Debugging process, refer to *Operating Transall under Debugging Control* on page 323.

STOP DEBUGGING

Use the **Stop** command to cease debugging your project.

To Stop Debugging

- Select **Debug>Stop Debugging**.

  -or-

Click in the Debug Toolbar.

-or-

Press **CTRL + F5**.

BREAK

Use the **Break** command to pause the debugging routine at the current point.

To Demand a Debugging Break

- Select **Debug>Break**.

  -or-

  Click in the Debug Toolbar.

  In the open source code window, a yellow arrow will point to the line where the debugging routine left off.

STEP INTO

Use the **Step Into** command to continue viewing the debugging process after the debugger has reached the initial breakpoint. If the next statement calls another routine within the Transall Application, that routine's source code is opened and execution continues to the first statement in that routine.

To Step Into

1. Select **Debug>Step Into**.

-or-

Click in the Debug Toolbar.

-or-

Press **F8**.

3. Continue clicking the icon to process each additional statement.

STEP OVER

Use the **Step Over** command to continue viewing the debugging process after the debugger has reached the initial breakpoint. If the next statement calls another routine within the Transall Application, execution pauses at the next statement in the calling routine *after* the calling statement.

To Step Over

1. Select **Debug>Step Over**.

-or-

Click in the Debug Toolbar.

-or-

Press **CTRL + F8**.

4. Continue clicking the icon to process additional statements.

Run to Cursor

Use the **Run to Cursor** command to run the debugger until it reaches the point in the source code where the cursor is set.

Toggle Breakpoint

Use the **Toggle Breakpoint** command to place breakpoints manually on a selected source code or LogicTree statement. This is an effective troubleshooting mechanism because you can place a breakpoint on or near a statement that's causing an error message. The debugger then steps through the suspect statement so you can see the exact cause of the error.

For further information, refer to *Breakpoints in Transall* on page 322.

To Toggle a Breakpoint

1. Open the Component Explorer.
2. Open a Script Module or LogicTree component.
3. When the pertinent Assistant editor opens, set the cursor at the desired statement.
4. Select **Debug>Toggle Breakpoint**.

   -or-

   Click in the Debug Toolbar.

   -or-

   Press **CTRL + F9**.

   A red dot will be placed at the start of the statement.

   Repeat the above steps to remove an existing breakpoint.

Clear All Breakpoints

Use the **Clear All Breakpoints** command to remove the breakpoints from all of the components in your Component Explorer.

To Clear All Breakpoints

- Select **Debug>Clear All Breakpoints**.

Transall removes all breakpoints from all statements.

EXTERNAL RUNTIME ERROR

Use the **External Runtime Error** command to locate a specific script line number in the open window.

To Invoke External Runtime Error

1. Select **Debug>External Runtime Error**.

   The Find Error Line dialog appears:

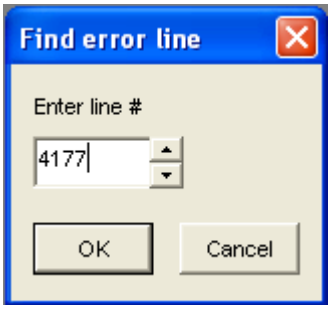


Figure 69: Find Error Line Dialog Box

2. Enter a line number and select **OK**.

   This will bring you to the selected line in the open window.

TOOLS MENU

The Tools menu contains a single menu item, **Options**.

OPTIONS

Select the **Tools>Options** command to open the Options dialog box.

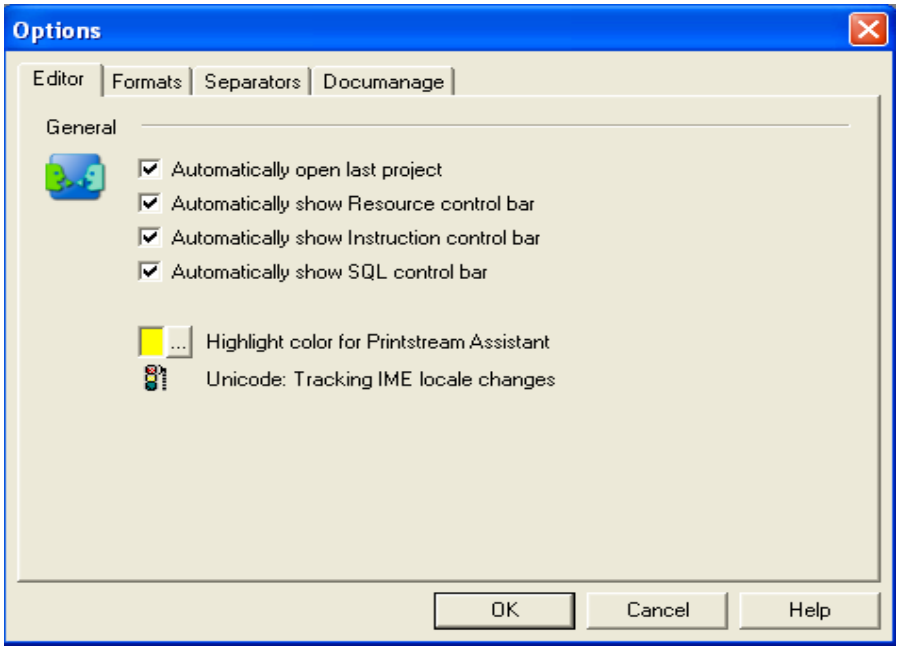


Figure 70: Tools>Options Dialog Box

Using the Options Dialog and Views

The Options dialog box contains four tabs. You complete an option by specifying general operating parameters under these tabs.

To Use the Options Tab

- Do one of the following:

To	See
Specify the features that automatically open upon first opening the Transall Editor and to select the highlight color for the Printstream Assistant.	Editor
Create and maintain formats to apply to destination fields	Formats
Create and maintain separators for delimiting records and fields.	Separators
Set and maintain Documanage interface settings.	Documanage

To Save or Close the Options Dialog

- Do one of the following:

To	Do this
Apply the specifications you've provided and return to the Transall Editor	Click **OK**
Return to the Transall Editor without applying the specifications you've provided	Click **Cancel**

Using the Editor Tab

Use the Editor tab to choose which features should automatically open/display when the Transall Editor is first opened. It also allows you to change the color to use to highlight items in the Printstream Assistant.

To Display the Editor Tab

- If the Editor tab isn't already showing in the Options dialog box, click on the Editor tab.

The Editor tab displays.

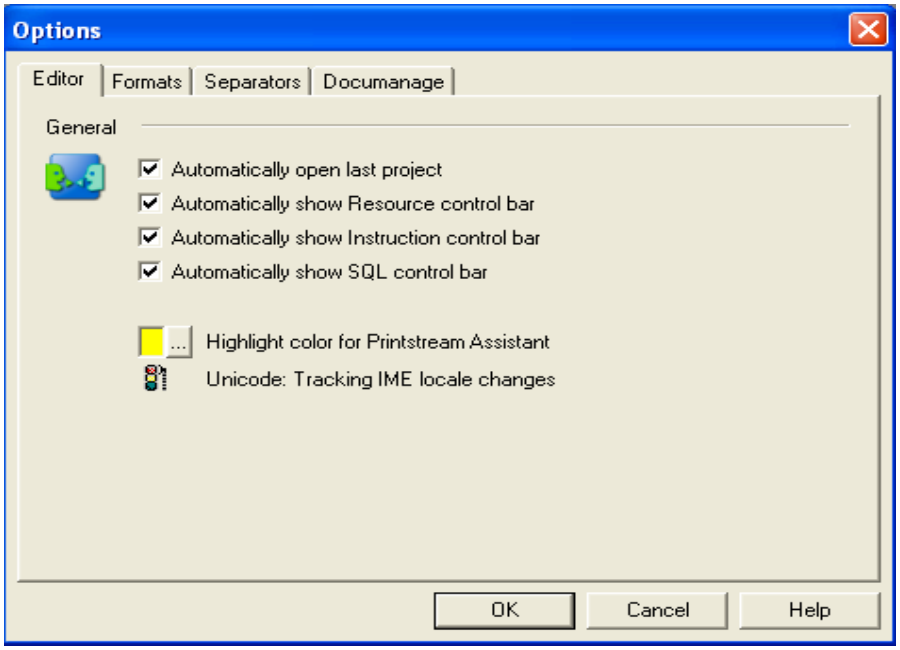


Figure 71: Display the Editor Tab

To Specify which Features will Automatically Appear when the Transall Editor Starts

- Remove the checkmarks next to the items that you DO NOT want to automatically appear when the Transall Editor opens. By default, the four items in this list are checked.

To Change the Color of Highlighted Text in the PrintStream Assistant

1. Click ... to open the Color dialog box. The default color is yellow

2. Select a color from the color grid.
3. Click **OK** to apply the change.

Using the Formats Tab

The Formats tab provides a prepopulated set of field formats for use in the fields of Destination Records and Queries. Use the Formats Tab to add, edit, delete field formats that your Transall Application requires.

To Display the Formats Tab

- Click **Formats** on the Options dialog box.

  The Formats tab displays.

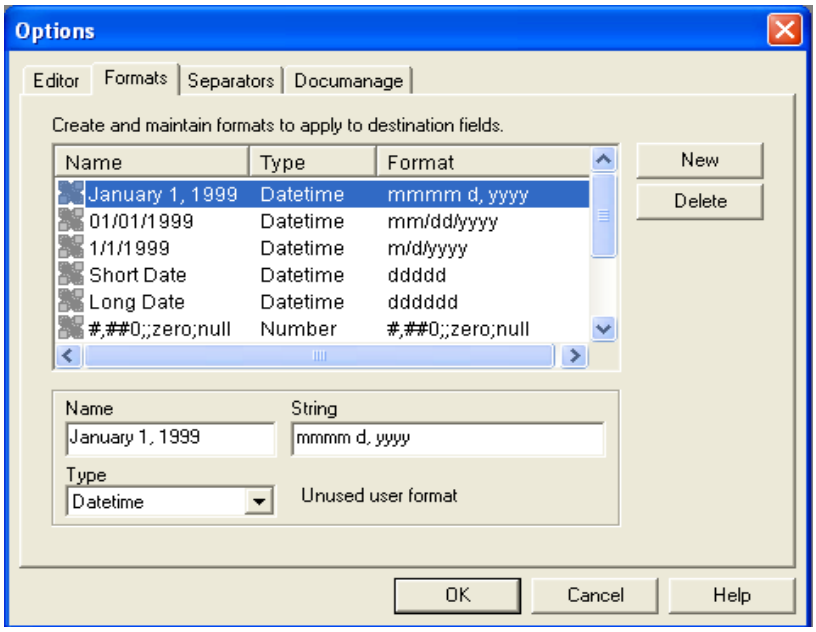


Figure 72: Display the Formats Tab

To Create a New Field Format

1. Click **New**.

   The text “UserFormat” will appear in the Name and String text boxes beneath the list of field formats. The type in the “Type” pull down list will default to “Number”.

2. Enter the new name, string, and type in their respective fields.
3. Click **OK**. The **Options** dialog box closes.
4. Select **Tools>Option>Formats**. The new field format appears at the bottom of the list.

   New or edited field formats instantly become available in the choice lists in the pertinent Destination Record and Query properties shown in the Component Inspector.

To Delete a Field Format

1. Click on the row containing the field format.
2. Click **Delete**. The field format is deleted from the list.
3. Click **OK**. The **Options** dialog box closes.

Using the Separators Tab

The Separators tab provides a prepopulated set of record separators for use in the fields of Destination Records and Queries. Use the Separators Tab to add, edit, delete record separators that your Transall Application requires.

To Display the Separators Tab

- Click the **Separators** tab on the Options dialog box.

  The Separators tab displays.

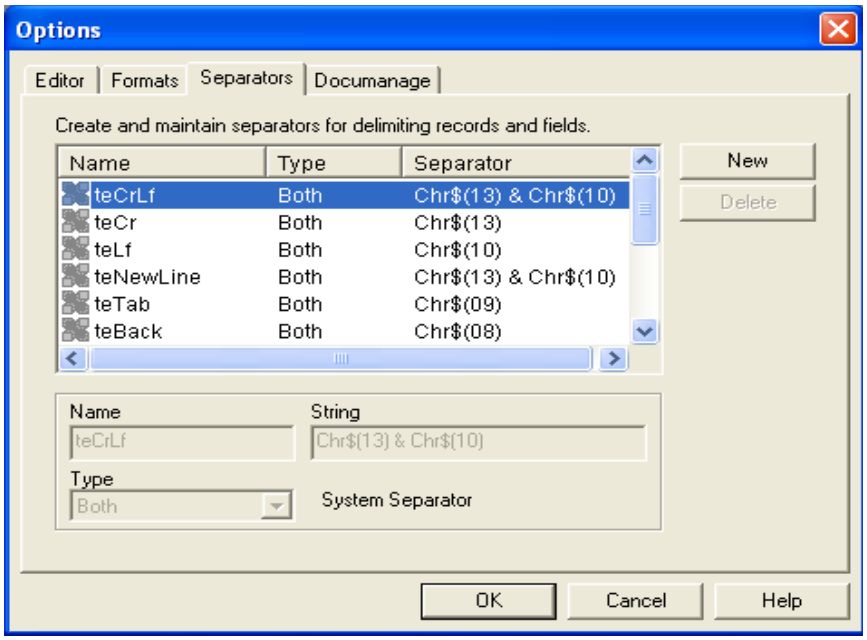


Figure 73: Display the Separators Tab

To Create a New Record Separators

1. Click **New**.

   The text “Separator” will appear in the Name and String text boxes beneath the list of record separators. The type under the “Type” pull down list will default to “Field”.

2. Enter the new name, string, and type in their respective fields.
3. Click **OK**. The **Options** dialog box closes.
4. Select **Tools>Option>Separators**. The new record separator appears at the bottom of the list.

New or edited separators instantly become available in the choice lists in the pertinent field properties shown in the Component Inspector for Record and Query subcomponents of Sources and Destinations.

To Delete a Record Separators

1. Click on the row containing the record separator.
2. Click **Delete**. The row is deleted from the list.
3. Click **OK**. The **Options** dialog box closes.

Using the Documanage Tab

Use the Documanage Tab to set and maintain Documanage interface settings.

Note that if you enable the Documanage interface using the **Tools>Options> Documange** command, you'll see an additional menu item for Documanage under the following menu items:

- File>Open
- File>Save
- File>Save As

This additional menu item allows to open/save a project file or a Documanage file:

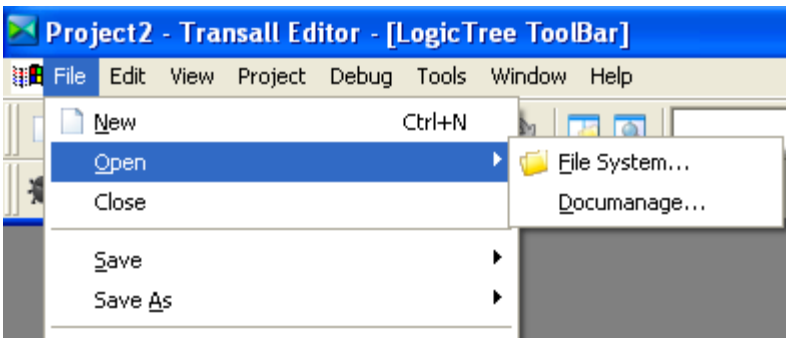


Figure 74: Documange Menu Item

To Display the Documanage Tab

- Click the **Documanage** tab on the Options dialog box.

The Documanage tab displays.

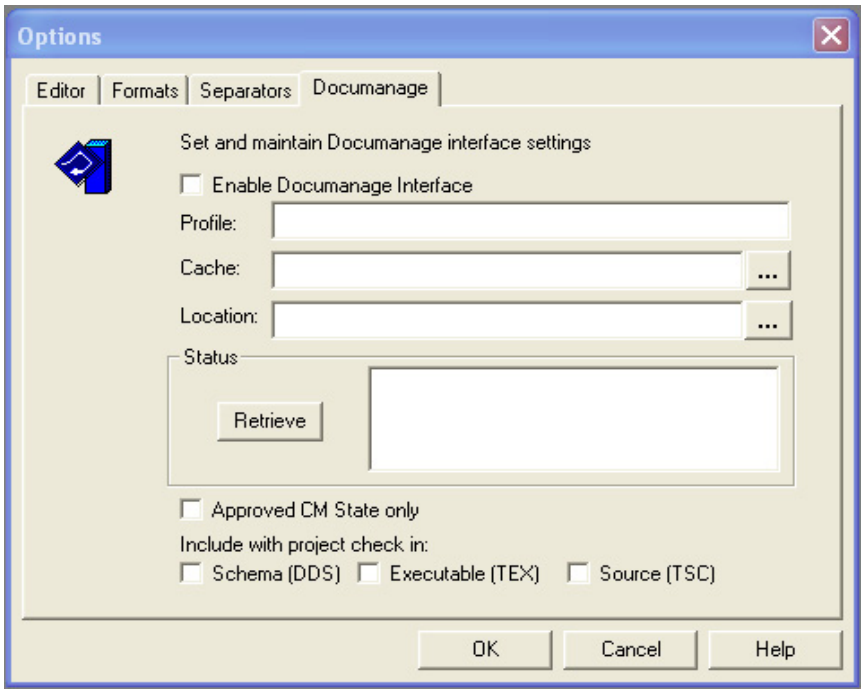


Figure 75: Display the Documanage Tab

WINDOW MENU

The Window menu comprises the following menu options:

If you want to	See
Open a duplicate of the current window	*New Window* on page 112
Close the active window	*Close* on page 113
Close all windows	*Close All* on page 113
Place windows in a overlapping sequence	*Cascade* on page 113
Place windows next to each other from left to right	*Tile* on page 113
Line up the minimized windows along the bottom of the screen	*Arrange Icons* on page 113

NEW WINDOW

Use the **New Window** command to open a duplicate of the current window.

To Open a New Window

- Select **Window>New Window**.

A duplicate of the current window will open in front of the current window.

CLOSE

Use the **Close** command to close the window that's currently active (e.g., Layout, Preview, Report, Rule, or Source).

To Close a Window

- Select **Window>Close**.

CLOSE ALL

Use the **Close All** command to close all open windows (e.g., Layout, Preview, Report, Rule, and Source).

To Close All Windows

- Select **Window>Close All**.

CASCADE

Use the **Cascade** command to stagger the display of the windows in an overlapping manner. Each window is slightly offset so you can see the window beneath it.

To Cascade Windows

- Select **Window>Cascade**.

Transall resizes the windows and stacks them on top of each other.

TILE

Use the **Tile** command to place the windows side by side from top to bottom. Transall takes up to three windows, stretches them so they're wider than they are tall, and uses all available space not taken up by the various windows.

To Tile Horizontally All Open Windows

- Select **WIndow>Tile**.

Transall stretches the windows horizontally and stacks them from top to bottom.

ARRANGE ICONS

Use the **Arrange Icons** command to line up all minimized windows along the bottom of the screen.

To Arrange Icons

1. Minimize all open windows in the Transall Editor.
2. Select **Window>Arrange Icons**.

   The windows will be lined up in a row across the bottom of the screen.

HELP MENU

The Help menu comprises the following menu options:

HELP

Use the **Help** command to view the compiled HTML help system for the Transall Editor.

To View the Help System

1 Select **Help**

-or-

Press **F1**.

The Transall Editor help system displays.

3. Use the navigation buttons to explore the help system.

ABOUT TRANSALL EDITOR

Use the **About Transall Editor** command to view the copyright information and the current version and release number.

To View the About Box

1. Select **Help>About Transall Editor**

   The **About Transall Editor** dialog displays.

2. Click **More** for additional copyright information or **OK** to return to the Transall Editor.

Working with Sources and Destinations

This chapter introduces the roles and features of Source and Destination components, including their Record subcomponents.

In your Transall project you create components called Sources and Destinations to describe files, databases, and other applications that provide data to, or receive data from, the Transall Application that you want to produce.

Each Source and Destination component must contain at least one Record subcomponent. Each Record subcomponent contains a list of field descriptions. Each field description provides the characteristics of a piece of data, such as whether the data represents a number, datetime, or series of characters, the field's size in bytes, and so on.

After creating the appropriate Sources and Destinations, you create Map components that describe how the Transall Application moves data from sources to destinations. For more information about Map components, see *Working with Maps* on page 269.

Although the data values that the Transall Application places into a Destination Record's fields typically originate from fields in Source records, they can also originate from components in the Transall Application's Transall Database, or as return values from functions coded in Transall Script Modules.

For more information about the components in the Transall Database, see *Working with the Transall Database* on page 279. For more information about coding a Script Module, see *Working with Transall Scripts and Script Modules* on page 339.

About Sources and Destinations

A **Source** and a **Destination** are components that identify either a file or an ODBC data connection.

A Source component describes an external resource from which the Transall Application obtains data and describes how that data is organized.

A Destination component describes an external resource where the Transall Application adds, updates, or deletes data and describes how that data is organized.

New Export Capabilities

Transall exports text files from many of its Assistants. The **Export** context menu item is available wherever exporting is supported and it presents the File Export dialog.

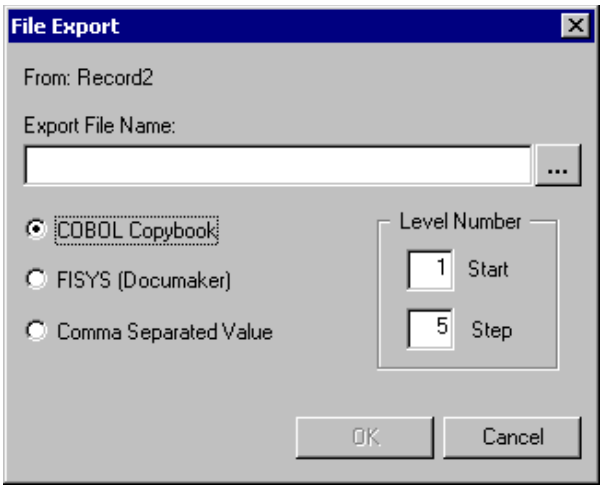


Figure 76: File Export Dialog Box

The layouts available are enabled based on the context. For comma separated value output (CSV), the format is FieldName, DataType (unless a map is being exported), then its MapField= Expression. For COBOL copybook exports, you can indicate the start level number and the increment amount.

Enhanced Read Flexibility

For Sources with multiple records, you can specify the order in which they're read for matching on reading a next record. The *Change Read Order* menu item displays the Set Read Order dialog.

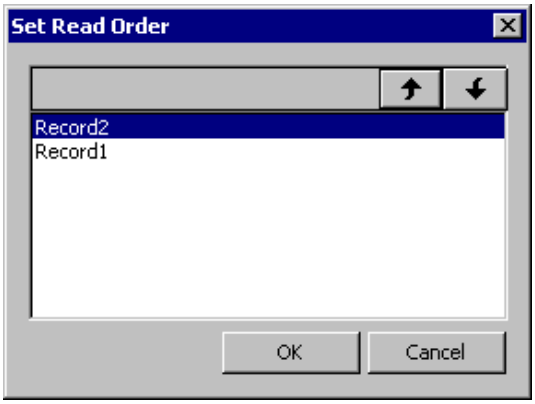


Figure 77: Set Read Order Dialog Box

Use the arrow keys to move the highlighted record. The Assistant won't visually reflect the changed read order.

File-Based Sources and Destinations

A Source or Destination that refers to a file contains the information that the Transall Application requires to work with that file. For instance, when you create a new file-based Source or Destination, you must provide its location or path.

Transall allows you to work with simple to very complex file organizations. A Source or Destination can refer to a file that:

- Contains data with fixed-length fields
- Contains data with variable-length fields delimited by a specified character
- COBOL defined data including support for `OCCURS` and `REDEFINES`
- Contains “Well Formed” XML

While the Transall Application runs, the file referenced by the Source or Destination must have these properties:

- The file must exist, or if used for writing, the file can be created.
- The file must be available to the computer where the Transall Application runs.

ODBC-Compliant Sources and Destinations

A Source or Destination can refer to an ODBC data source, which must conform to the Microsoft Open Database Connectivity standard for application interoperabilit

Note Transall requires ODBC Level 2 and later.

DETAILS ON FILES

About Record Subcomponents

Each Transall Application is designed to read and write *record-oriented* data. That is, it expects to read and write “chunks” of data structured as a series of fields that have a predictable order.

A *data record* contains a series of data values. Each data record contains information about an entity that is of interest to your application, such as employees, business transactions, and so on.

Each data value occupies a portion of the data record called a *field*. Each field in a record can contain a value that has a set of properties, such as datatype, size in bytes, and so on. A description of a field defines the set of properties that allow the Transall Application to work with a value stored in that field.

To represent the set of fields in a data record that the Transall Application works with, a file-based Source or Destination must contain at least one **Record subcomponent**. Each Record subcomponent contains a list of field descriptions

A file-based Source or Destination can have more than one Record subcomponent defined. For instance, you should create more than one Record subcomponent in a Source when the actual data source provides record-oriented data than can conform to more than one possible set of fields. (i.e., Record Types)

Defining a Record's Identifier Field

When a Source contains more than one Record subcomponent, you can use each Record's Identifier property to indicate which field contains the value that identifies a given data record's structure. Your Transall Application can utilize this feature to perform *record-identifier-based* control-break processing. This should be used when the data records obtained from a Source might vary in structure from record to record, and the Transall Application cannot predict that variance based on the data obtained from previous data records.

For example, a Source might provide a stream of fixed-length records with two fields, RECORD-TYPE and EXPIRATION TIME, but the EXPIRATION-TIME field can contain either a datetime value or a string value depending upon the RECORD-TYPE field's value.

Your Transall Application can use this information when executing a Logic Tree that contains a Walk ControlBreak instruction sequence. In this case, as the Logic Tree's processing obtains the next record from the Source, the correct ControlBreak instructions are triggered based on the value in the RECORD-TYPE field and the field in question is processed appropriately.

Nested after the ControlBreak instruction can be one or more Condition instructions for handling the data record based on the value of its RECORD-TYPE field. For instance, if the condition RECORD-TYPE field contains "T", the next Logic Tree instruction should perform a Map instruction that pertains to the Source Record whose Identifier property contains RECORD-TYPE and whose IdentifierValue property contains "T".

Note If a Source has more than one Record subcomponent, but they describe a stream of data records that follow a "master-and-detail" pattern, then a Logic Tree can process those data records using *field-based* control-break processing. This is defined in the Field(s) property of a ControlBreak instruction in a Logic Tree.

See the description of defining a ControlBreak instruction in *Working with Logic Trees* on page 287.

Data Types

In the internal workings of Transall there are eight native data types: Integer, Long, String, DateTime, Double, Float, PNum, and UNum.

Integer	16 Bit signed integer.
Long	32 Bit signed integer.
String	up to 2GB in length.
DateTime	DateTime structure in the form of CCYY/MM/DD.HH.MI.SS.MMMMMMMMM
Double	64-bit (8-byte) floating-point number (i.e., double precision)
Float	32-bit (4-byte) floating-point number (i.e., single precision) or less precise version of double
PNum	Packed Decimal - contain a number (acts like strings only unprintable and packed)
UNum	UnPacked Decimal - contain a number (acts like string or regular printable recorders and unpacked)

Using Format Options

Under the Formats tab of the **Tools>Options** dialog, the Transall Editor provides a pre-populated set of field formats for use in the fields of Destination Records and Queries. If the pre-populated list does not contain the field formats that your Transall Application requires, you can add, edit, or delete field formats.

Figure 78 on page 119 shows the display of the Formats tab.

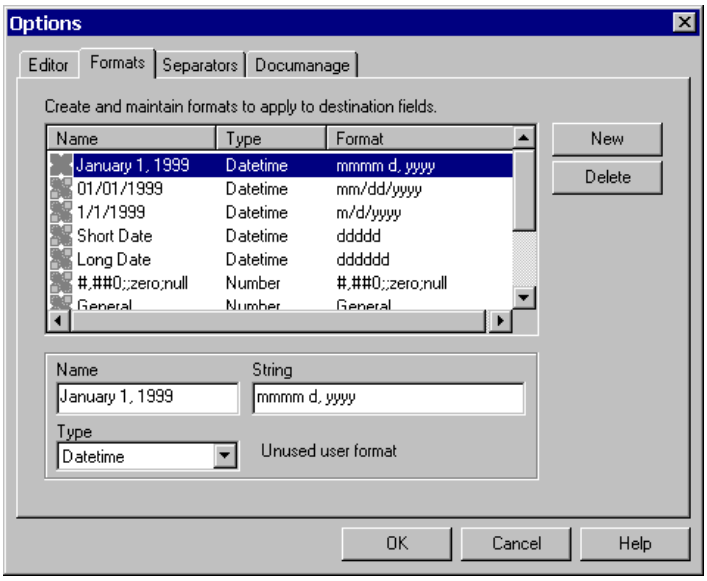


Figure 78: Formats for Destination Fields

New or edited field formats instantly become available in the choice lists in the pertinent Destination Record and Query properties shown in the Component Inspector.

Using Separator Options

Under the Separators tab of the **Tools>Options** dialog, the Transall Editor provides a pre-populated set of record separators. If the pre-populated list does not contain the separators that your Transall Application requires, you can add, edit, or delete record separators.

Figure 79 on page 120 shows the display of the Separators tab.

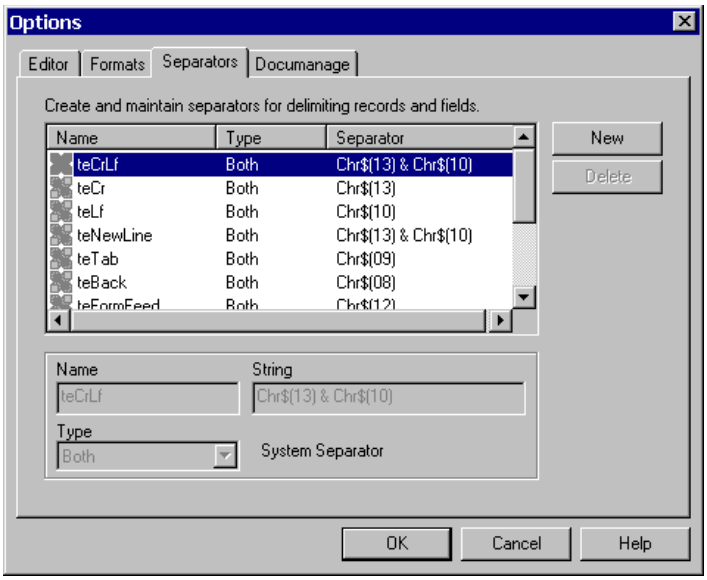


Figure 79: Separators for Field Descriptions

New or edited separators instantly become available in the choice lists in the pertinent field properties shown in the Component Inspector for Record and Query subcomponents of Sources and Destinations.

Creating Sources and Destinations

The Transall Editor presents the same user interface for creating all Source and Destination components. *In the descriptions that follow, we present examples of creating a Source; you can use the same actions to create a Destination.*

To Create Sources and Destinations

1. Select **Project>Add Source**.

The Transall Editor displays the Add Source dialog, as shown in *Figure 80 on page 121*.

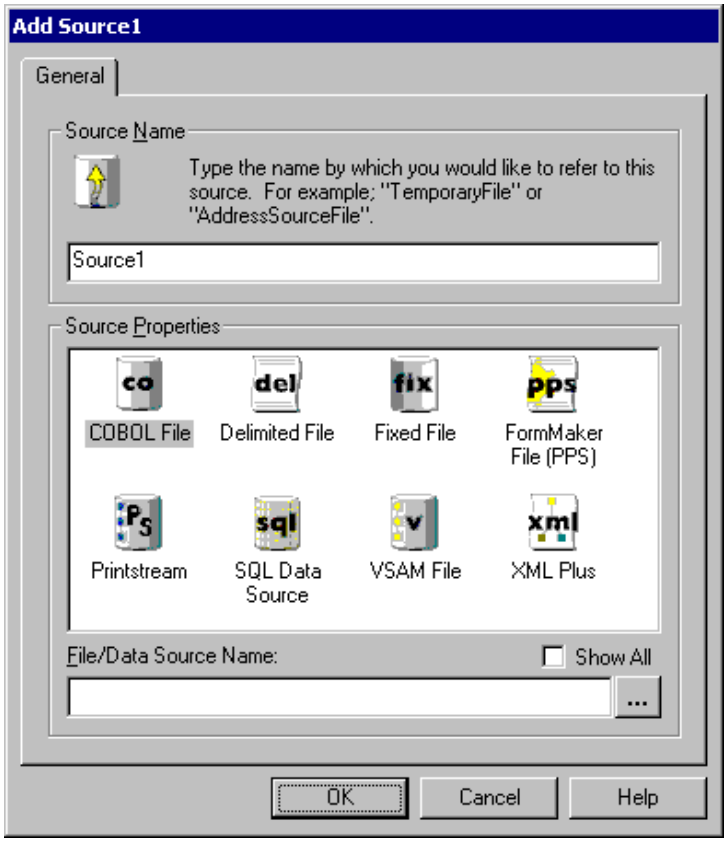


Figure 80: Add Source Dialog

2. Type a name for the new Source.

Hint To complete the next step, you must already know whether the new Source refers to a file or to an ODBC data source. In other words, before creating a Source, you must already know whether your Transall Application expects to obtain this Source's data records from a file or ODBC data source.

3. In the Source Properties group box, click the icon representing the type of Source:

   You can choose from the following Source types:

   - **COBOL File**: The data values in each record correspond to fields defined in a separate COBOL copybook file; otherwise, this will behave like a fixed file.
   - **Delimited File**: The length of each record is unpredictable and, therefore, the Transall Application expects to find a separator character between the data values in each file record.
   - **Fixed File**: Each record is of a fixed length; multiple records definitions are supported, with a predictable delimiter, or *separator*, character between those records. No separator is also supported.

- **PPS File (PPS)**: The file's data records are organized and formatted as a PPS file. (See your Documaker Workstation/PPS documentation for more information about the characteristics of this file.)
- ***Generic File (Scripted)**: The data is in a format that isn't supported by existing File, XML, or SQL options. Scripted data sources and destinations provide a framework of empty script subroutines, where you'll insert script to perform various operations.
- **Printstream**: The data is in a print file format, such as IBM AFP, HP PCL, Xerox Metacode, or text.
- **SQL Data Source**: The data is accessed using an ODBC driver.
- **VSAM File**: The data is in a file layout conforming to IBM's **V**irtual **S**torage **A**ccess **M**ethod organization.
- ***XML**: The Transall Application allows low-level access to well-formed XML files.
- **XML Plus**: The Transall Application interacts with XML sources very much like the way it interacts with multi-record file data sources.

You can choose from the following Destination types:

- **COBOL File**: Same as for a Source.
- **Delimited File**: Same as for a Source.
- **Docuflex File**: The data is organized and formatted for use in creating a Docuflex project for document publication.
- ***Documaker FP File (VRF)**: The file's data records are organized and formatted as a Documaker VRF file. (See your Documaker documentation for more information about the characteristics of this file.)
- **Documaker FP Plus (VRF)**: The objective of the FP Plus data destination is to let the Transall developer concentrate on the business logic for selecting forms and tags that are required for various business scenarios versis thinking about making calls to the Documaker VDR API.
- **Fixed File**: Same as for a Source.
- ***Generic File (Scripted)**: Same as for a Source.
- **SQL Data Source**: Same as for a Source.
- **VSAM File**: Same as for a Source.
- **XML**: The Transall Application interacts via a hierarchy of records that are setup in the XML destination.

Tip Sources and Destinations preceded by an asterisk are available by enabling the **Show All** check box.

4. In the File/Data Source Name text box, type (or browse for) a file path. When the Transall Application opens this Source to obtain data records, it uses this file by default.

You can set this property by two alternate methods:

- After you create the Source, type (or browse for) a file path by editing the FileName property in the Component Inspector control bar.
- If the file path for this Source is subject to change, or cannot be specified until the Transall Application's run-time, use the built-in `SetFileName` Event so that the Transall Application determines the file path as it runs.

5. Click **OK** to create the Source.

Because you can create a wide variety of Sources and Destinations, this guide provides separate topics for each type:

If you chose	See:
Sources:	
COBOL File	*Describing a COBOL File Source* on page 126
Delimited File	*Describing a Delimited File Source* on page 126
Fixed File	*Describing a Fixed File Source* on page 124
Oracle File (PPS)	*Describing a PPS File Source* on page 129
Generic File (Scripted)	*Scripted Assistant* on page 221
SQL Data Source	*Describing an ODBC-Based Source* on page 159
VSAM File	*Transall Gateway* on page 405
XML	*Setting up an Event-based XML Source* on page 237
XML Plus	*XML Plus Source* on page 232
Destinations:	
COBOL File	*Describing a COBOL File Source* on page 126
Delimited File	*Describing a Delimited File Source* on page 126
Docuflex File	*Setting up a Docuflex File Destination* on page 189
Documaker FP File (VRF)	*FP Plus Destination* on page 197
Documaker FP Plus (VRF)	*FP Plus Destination* on page 197
Fixed File	*Describing a Fixed File Source* on page 124
Generic File (Scripted)	*Scripted Data Sources and Destinations* on page 221
SQL Data Source	*Describing an ODBC-Based Source* on page 159
VSAM File	*Transall Gateway* on page 405
XML	*XML Data Destinations* on page 249

DESCRIBING A FIXED FILE SOURCE

After you complete the Add Source dialog, the Transall Editor opens the Source's File Assistant as shown in *Figure 81 on page 124*.

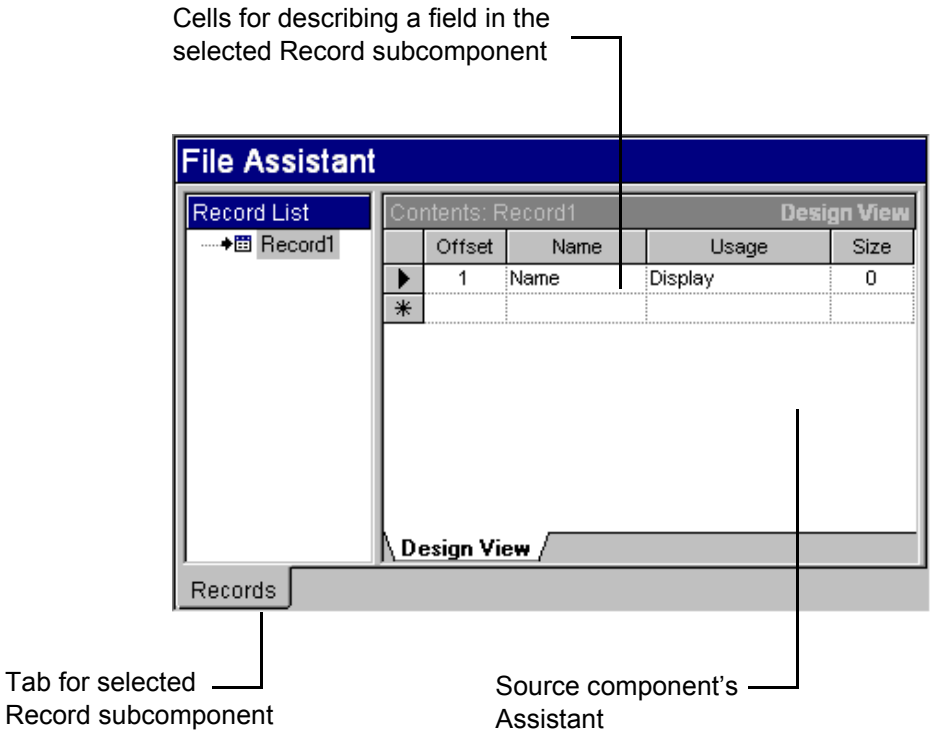


Figure 81: File Assistant for Fixed File Source

Next, you must enter descriptions of the fields for values available in the file's data records.

As shown in Figure 81 on page 124, you do so by adding or editing field descriptions in the Source's selected Record subcomponent using a spreadsheet-style interface.

You must enter a field Name, Usage data type or format, and Size as a length in bytes for each field that you add to a Fixed File Source Record. The Transall Editor automatically updates the Offset column for each field description that you add.

Type or select a Usage data type for the new field. Presently, the Usage cell can only contain a COBOL data type.

After you finish adding field descriptions for the selected Source Record, and after you finish editing each new field's component properties in the Component Inspector, you can close the File Assistant.

Adding a Record Subcomponent

To add another Record subcomponent in a Source, select the **Resource>Record Add** menu command, or right-click the mouse anywhere within the File Assistant's Record List view and select the **Add Record** command from the pop-up menu.

Adding another Record subcomponent to the Source causes a new row to show in the record list for the source. You can now add new field descriptions for the new Record in the Contents view.

Changing the Name of a Record Subcomponent

To change the name of a Record subcomponent in a Source or a Destination, click on the Record's name in the File Assistant's Record List view, then edit the Record's Name property in the Component Inspector.

Copying Field Descriptions Into an Empty Record Subcomponent

You can conveniently add field descriptions in an *empty* Record subcomponent by copying them from another Record in any Source or Destination in the open project.

To do so, select the **Resource>Copy From** menu command, or right-click the mouse in the File Assistant's Record List view and select the **Copy From** command from the pop-up menu. In the dialog that appears, use the + and - tree node controls to display the names of the Record subcomponents for each Source or Destination in the project. Select the name of the Record whose field descriptions you want to copy, and press OK.

Note If you copy field descriptions from a Source of a different kind, like copying from an ODBC Source to a file Destination, the Transall Editor leaves default values in the property cells for the new fields in the target Record. Edit the new field descriptions' properties as needed.

The **Resource>Add From** command is also available from the drop-down and pop-up menus. It behaves like Copy From, but creates a new child record to act as the destination.

Deleting a Record Subcomponent

To delete the selected Record subcomponent in a Source, select the **Resource>Record Delete** menu command, or right-click the mouse on the record name in the File Assistant's Record List view and select the **Record Delete** command from the pop-up menu.

DESCRIBING A DELIMITED FILE SOURCE

After you complete the Add Source dialog, the Transall Editor opens the File Assistant as shown in *Figure 82 on page 126*.

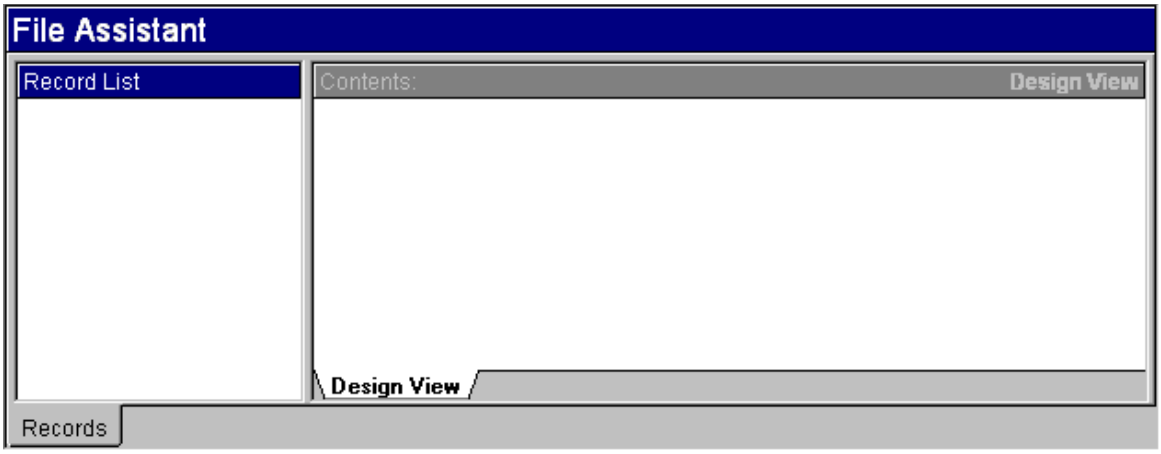


Figure 82: File Assistant for Delimited File Source Component

For a Delimited File Source, in the selected Record subcomponent you can enter only a Name property for each field. The Transall Editor automatically updates the Field # column for each new field that you describe.

You can also edit the properties of the new Source in the Component Inspector. The allowable values for this component's properties are found in *Delimited File* on page 147 of this chapter.

After you finish adding field descriptions for the selected Source Record, and after you finish editing each new field's component properties in the Component Inspector, you can close the File Assistant.

DESCRIBING A COBOL FILE SOURCE

After you complete the Add Source dialog, the Transall Editor opens the File Assistant.

To add a set of field descriptions that correspond to an existing COBOL copybook source file, select the **Resource>Import Copybook** menu command, or right-click the mouse in the File Assistant's Record List view and select the **Import Copybook** command from the pop-up menu. In the Import Copybook dialog, you can browse for a file to select.

After you specify a COBOL copybook source file, the Transall Editor automatically scans the file's COBOL source and adds field descriptions in the new Source's empty Record subcomponent based on the copybook file's field definitions. The Transall Editor adds one field in the Record subcomponent for each COBOL field found in the copybook file.

For the entire COBOL record or structure, the Transall Editor creates one "root" field in the Source's Record whose Usage property is Group. For other group fields found in the copybook file, the Transall Editor also sets the Usage property value to Group in the corresponding Source Record's group fields.

The Transall Editor sets a new field description's Occurs property to a non-zero value when its corresponding COBOL field is defined with an `OCCURS` clause.

For each field description that is imported into the Source Record, the Transall Editor sets the value of the field's Usage property to a value based on the copybook's field COBOL definitions and sets its Size property based on its COBOL field definition.

For example, assume that you are creating a Source for a file containing data records whose fields correspond to the COBOL copybook shown in *Figure 83 on page 127*.

Note Transall requires a starting 01 group level to import copy books. If several 01 groups are found, each is imported into a Transall record. Also note that Transall ignores "copy" directives in the copybook being imported. To handle "copy" directives the Transall user should assemble the copybook fragments in something like notepad before importing a complete copybook.

```
01  ACCT-REC-TYPE.
        10  SORT-INFO                   PIC  X(46).
        10  JSG1-ACCT-REC-TYPE          PIC  X(03).
        10  JSG1-BAR-CODE.
            15  JSG1-INSERT-CODES       PIC  X(04).
            15  FILLER REDEFINES JSG1-INSERT-CODES.
                20  JSG1-BYTE-2         PIC  X.
                20  JSG1-BYTE-3         PIC  X.
                20  JSG1-BYTE-4         PIC  X.
                20  JSG1-BYTE-5         PIC  X.
            15  JSG1-ENVELOPE-NO        PIC  X(6).
            15  JSG1-FEEDER-TYPE        PIC  X.
        10  FILLER                      PIC  X.
        :
        :
01  ACTV-REC-TYPE.
        10  SORT-INFO                   PIC  X(46).
        10  JSG5-ACTIVITY-RECORD-TYPE   PIC  X(03).
        10  JSG5-ACTIVITY-DESC1         PIC  X(200).
        :
        :
01  MPOL-REC-TYPE.
        10  SORT-INFO                   PIC  X(46).
        10  JSG78-MPOL-RECORD-TYPE      PIC  X(03).
        10  JSG78-EXPIRE-POLICY-DESC    PIC  X(10).
        :
        :
01  MSG-REC-TYPE.
        10  SORT-INFO                   PIC  X(46).
        10  JSG-MESSAGE-RECORD-TYPE     PIC  X(03).
        :
        :
01  POL-REC-TYPE.
        10  SORT-INFO                   PIC  X(46).
        10  JSG4-POL-RECORD-TYPE        PIC  X(03).
        10  JSG4-POLICY-DESCRIPTION     PIC  X(147).
        :
        :
```

Figure 83: Sample COBOL Copybook

Figure 84 on page 128, shows the Assistant for the COBOL file Source component after importing the COBOL copybook shown in *Figure 83 on page 127*. In the figure, notice that the Transall Editor creates only one Record subcomponent that contains all the imported field descriptions.

In COBOL copybook file:

```
01  ACCT-REC-TYPE.
...
01  ACTV-REC-TYPE.
...
01  MPOL-REC-TYPE.
...
01  MSG-REC-TYPE.
...
01  POL-REC-TYPE.
...
```

In COBOL file Source component's Record:

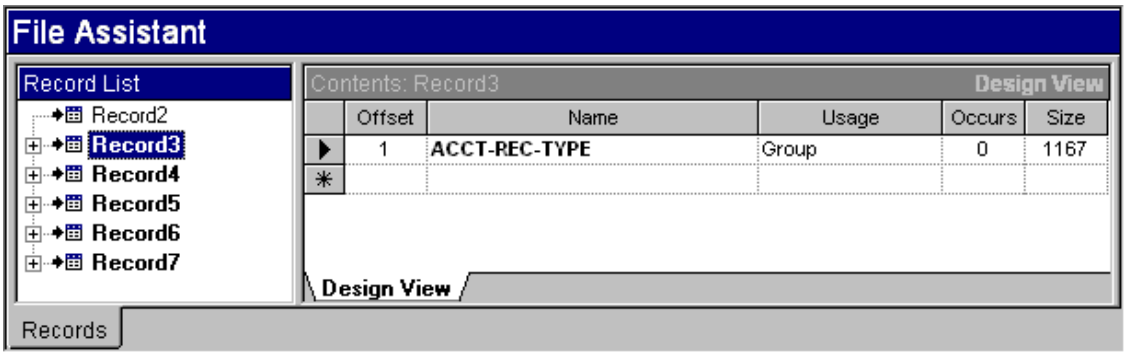


Figure 84: Group Fields Resulting From Importing Sample COBOL Copybook File

In certain cases, the Transall Editor creates a "container" field description in the Source Record with a Usage property of Group, as follows:

- A root field represents the entire COBOL record; its name is the same as the COBOL record name.
- Each COBOL field that defines groups of sub-fields.

As shown in Figure 84 on page 128, after the Transall Editor has populated the COBOL file source's Record in the File Assistant's Record List view, you can view the structure of any group or redefined fields that were added to the Record, as follows:

- click on the + and - controls to show and hide any Group fields.
- click on a Group field name to view its member fields.

Figure 85 on page 129, shows the same COBOL file source's File Assistant after navigating to the fields contained in the root field named ACCT-REC-TYPE.

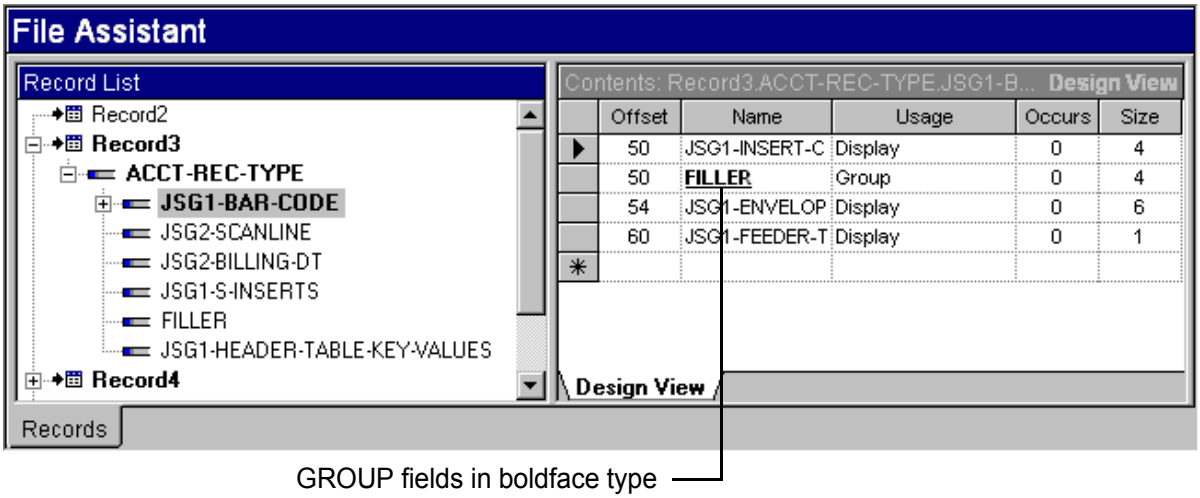


Figure 85: Fields Contained in Root Group Field of Imported Sample COBOL Copybook

In the figure above, notice that:

- The name of each Group field appears in boldface.
- Redefined fields are underlined.

Note The name of a field that appears in boldface and is underlined corresponds to group of fields that *redefine* another field.

After you finish viewing the populated field descriptions, you can close the File Assistant.

Describing a PPS File Source

After you complete the Add Source dialog, the Transall Editor opens the File Assistant. For the record descriptions you can enter an Identifier Value. The Identifier Value refers to the name of an image in the PPS export file. For the field descriptions in a PPS File Source's Record, you can enter only two properties: Name and Tag Name. The Tag Name is a text field that is the name of a "tag" (or field name) found in a PPS Form; the tag name identifies the name of a field used elsewhere in the data record.

The Transall Editor automatically updates the Field# cell for each new field description.

Note There are two special Identifier Values "PPS Session Header" and "PPS Form Header" that represent the Data String identifying a data entry session and a PPS Form Selected for data entry respectively. These special Identifier Value records must each have a field name to receive PPS data named "PPS Session Data" and "PPS Form Header Data" respectively.

REFERENCE FOR COMPONENT PROPERTIES - TRADITIONAL FILES

This section presents the allowable values for the unique properties for each kind of traditional file Source and Destination component. These values can be viewed by going to the Component Inspector, under the Properties tab, then find the properties listed below. By clicking in the box to the right of the property, will select the word and display a control, that when clicked on will display a drop down box, that will show the subcomponent's properties.

SOURCE PROPERTIES

There follows a list of the allowable values for the unique properties for each kind of Source component.

Fixed File

Property	Meaning
Access	Controls the file access method. The valid choices are: • **Read** - (default) If this is a new file (destination only), Transall will create it with read access. • **Read Write** - If this is a new file, Transall will create it with read / write access.
FileName	Pathname for the file containing data records.
LockMode	Controls the file access permission. The valid choices are: • **Write** - (default) Transall will set the file's access permission to lock out other processes from writing to the file while Transall has the file open for reading. • **Read** - Transall will set the file's access permission to lock out other processes from reading the file while Transall has the file open for reading. • **Read Write** - Transall will set the file's access permission to lock out other processes from having any access to the file while Transall has the file open for reading. • **Shared** - Transall will NOT set the file's access permission to lock out other processes from having access to the file while Transall has the file open for reading.
Mode	Controls the file input access. The valid choices are: • **Input** - (default) Transall will open the file for input access in "cooked" or translated mode. In this compatibility mode, carriage-return / line-feed combinations are converted to line-feed symbols. This yields a common compatibility with UNIX file systems. • **Binary Input** - Transall will open the file for input access in "raw" or untranslated mode. In this mode, carriage-return / line-feed combinations are NOT converted to line-feed symbols.

Property	Meaning
OpenMode	Controls the generation of logic to open a file. The valid choices are: • **Automatic** - (default) Transall will automatically generate logic to open this file for you. • **Manual** -Transall will generate logic to open this file but will not automatically call the logic. You must manually place a call to the generated logic in your Transall Scripts or LogicTrees to have Transall run the instructions that open this file. You might want to select **Manual** if you need to have extra control over when a file is opened by Transall. This may be the case if you have a need to open and close a file several times during a Transall run. By default, the **Automatic** open mode opens the file and holds the file open for the life of the Transall run, closing the file when Transall is ready to terminate.
Separator	This is a reference to a Transall resource (set up under **Tools>Options>Separators**) that is either a single character or a string of characters that Transall will look for in the file to locate each data record.
ByteOrder	Controls the processing of binary data in the file. The valid choices are: • **Any** - (default) Transall will assume any binary data being accessed in this file is in the native binary format of the platform Transall is running on. • **Little-Endian** -Transall will assume any binary data being accessed in this file is in little endian (least significant bits in) format and Transall will perform any binary manipulation required to process the data on the platform that Transall is operating on. **Note:** Data on the WIN32 platform is Little Endian. • **Big-Endian** -Transall will assume any binary data being accessed in this file is in big endian (most significant bits in) format and Transall will perform any binary manipulation required to process the data on the platform that Transall is operating on.

Property	Meaning
CharacterSet	Controls the processing of string data in the file. The valid choices are: • **Any** - (default) Transall will assume string data being accessed in this file is in the native character set of the platform Transall is running on. • **ASCII** - Transall will assume string data being accessed in this file is in the ASCII character set and Transall will perform any string manipulation required to process the data on the platform that Transall is operating on. • **EBCDIC** - Transall will assume string data being accessed in this file is in the EBCDIC character set and Transall will perform any string manipulation required to process the data on the platform that Transall is operating on. Or • **Unicode encodings** - UTF-8, UTF-16, UTF-16LE, UTF-16BE, UCS-2, UCS-2LE, UCS-2BE, UCS-4, UTF-32, UCS-4LE, UTF-32LE, UCS-4BE, UTF-32BE, UTS-6 • **Windows code pages** - W_CENTRAL_EUROPE, W_CYRILLIC, W_LATIN1, W_GREEK, W_LATIN5, W_HEBREW, W_ARABIC, W_BALTIC, W_VIETNAMESE, W_THAI, W_JAPANESE, W_KOREAN, W_S_CHINESE, W_T_CHINESE • **DOS code pages** - D_USLATIN, D_ARABIC1, D_GREEK, D_BALTIC, D_LATIN1, D_LATIN2, D_CYRILLIC, D_TURKISH, D_LATIN1EURO, D_PORTUGUESE, D_ICELANDIC, D_HEBREW, D_CANADIANFRENCH, D_ARABIC, D_NORDIC, D_CYRILLICRUSSIAN, D_GREEK2, D_THAI, D_ARABICASMO • **ISO code pages** - ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, ISO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, ISO_8859_11, ISO_8859_13, ISO_8859_14, ISO_8859_15, ISO_8859_16 • **Other code pages** - O_KOI8R, O_KOI8U, O_KOI8RU, O_KOI8UNI, O_BIG5, O_GB12345, O_GB2312, O_JIS0201, O_JIS0208, O_JIS0212, O_JOHAB, O_KSC5601, O_KSX1001, O_WANSUNG, O_GB18030

Property	Meaning
	• **EBCDIC code pages** - E_DFXDEFAULT, E_USCANADA, E_LATIN5TURKISH, E_INTERNATIONAL, E_GREEK, E_HEBREW, E_ROECELATIN2, E_JAPANESEKATAKANA_EX, E_ARABIC, E_KOREAN_EX, E_CYRILLIC_RUSSIAN, E_LATIN1_EURO, E_CYRILLIC_S_EUROPE, E_USCANADA_EURO, E_GERMANY_EURO, E_DENMARKNORWAY_EURO, E_FINLANDSWEDEN_EURO, E_ITALY_EURO, E_SPANISH_EURO, E_UK_EURO, E_FRANCE_EURO, E_INTL_EURO, E_ICELAND_EURO **Note**: On the Unicode encodings, a suffix of *LE* means Little Endian and *BE* means Big Endian and this refers to text storage. Refer to the ByteOrder property for specifying the storage of binary numbers.
ErrorHandler	Controls the processing of errors in the file. The valid choices are: • **Automatic** - (default) Transall will automatically handle and recover from errors. Transall will "throw" only critical errors that must be "caught" via an On Error Resume Next type statement. • **Manual** - Transall will ignore all errors and return errors to your Scripts or LogicTrees for handling.

Delimited File

Property	Meaning
Access	Controls the file access method. The valid choices are: • **Read** - (default) If this is a new file, Transall will create it with read access. • **Read Write** - If this is a new file, Transall will create it with read / write access.
FieldDelimiter	This is either a single character or a string of characters that Transall will look for in the data fields to locate each field's data value. The default for this is a double quote. If the incoming data has no double quotes delimiting a field's value, Transall will still correctly process the field.
FieldSeparator	This is either a single character or a string of characters that Transall will look for in the data records to locate each data field. The default for this is a comma.
FileName	Pathname for the file containing data records.
FirstRecordHeader	For source files, this property controls the processing of the first record in a file. You have these choices: • No - (default) Transall does not treat the first record as a header record. • Yes - Transall treats the first record as a header record. This makes Transall skip this record. It also enables the Import Header menu item in the delimited source File Assistant to automatically build the record layout from the header record. For destination files, this property determines if a header record is written. You have these choices: • No - (default) Transall does not write a header record. • Yes - Transall writes a header record before the first detail record that contains the field names wrapped in the field delimiter character.
LockMode	Controls the file access permission. The valid choices are: • **Write** - (default) Transall will set the file's access permission to lock out other processes from writing to the file while Transall has the file open for reading. • **Read** - Transall will set the file's access permission to lock out other processes from reading the file while Transall has the file open for reading. • **Read Write** - Transall will set the file's access permission to lock out other processes from having any access to the file while Transall has the file open for reading. • **Shared** - Transall will NOT set the file's access permission to lock out other processes from having access to the file while Transall has the file open for reading.

Property	Meaning
Mode	Controls the file input access. The valid choices are: • **Input** - (default) Transall will open the file for input access in “cooked” or translated mode. In this compatibility mode, carriage-return / line-feed combinations are converted to line-feed symbols. This yields a common compatibility with UNIX file systems. • **Binary Input** - Transall will open the file for input access in “raw” or untranslated mode. In this mode, carriage-return / line-feed combinations are NOT converted to line-feed symbols.
OpenMode	Controls the generation of logic to open a file. The valid choices are: • **Automatic** - (default) Transall will automatically generate logic to open this file for you. • **Manual** -Transall will generate logic to open this file but will not automatically call the logic. You must manually place a call to the generated logic in your Transall Scripts or LogicTrees to have Transall run the instructions that open this file. You might want to select **Manual** if you need to have extra control over when a file is opened by Transall. This may be the case if you have a need to open and close a file several times during a Transall run. By default, the **Automatic** open mode opens the file and holds the file open for the life of the Transall run, closing the file when Transall is ready to terminate.
Separator	This is a reference to a Transall resource (set up under **Tools>Options>Separators**) that is either a single character or a string of characters that Transall will look for in the file to locate each data record.
ByteOrder	Controls the processing of binary data in the file. The valid choices are: • **Any** - (default) Transall will assume any binary data being accessed in this file is in the native binary format of the platform Transall is running on. • **Little-Endian** -Transall will assume any binary data being accessed in this file is in little endian (least significant bits in) format and Transall will perform any binary manipulation required to process the data on the platform that Transall is operating on. **Note:** Data on the WIN32 platform is Little Endian. • **Big-Endian** -Transall will assume any binary data being accessed in this file is in big endian (most significant bits in) format and Transall will perform any binary manipulation required to process the data on the platform that Transall is operating on.

Property	Meaning
CharacterSet	Controls the processing of string data in the file. The valid choices are: • **Any** - (default) Transall will assume string data being accessed in this file is in the native character set of the platform Transall is running on. • **ASCII** - Transall will assume string data being accessed in this file is in the ASCII character set and Transall will perform any string manipulation required to process the data on the platform that Transall is operating on. • **EBCDIC** - Transall will assume string data being accessed in this file is in the EBCDIC character set and Transall will perform any string manipulation required to process the data on the platform that Transall is operating on. Or • **Unicode encodings** - UTF-8, UTF-16, UTF-16LE, UTF-16BE, UCS-2, UCS-2LE, UCS-2BE, UCS-4, UTF-32, UCS-4LE, UTF-32LE, UCS-4BE, UTF-32BE, UTS-6 • **Windows code pages** - W_CENTRAL_EUROPE, W_CYRILLIC, W_LATIN1, W_GREEK, W_LATIN5, W_HEBREW, W_ARABIC, W_BALTIC, W_VIETNAMESE, W_THAI, W_JAPANESE, W_KOREAN, W_S_CHINESE, W_T_CHINESE • **DOS code pages** - D_USLATIN, D_ARABIC1, D_GREEK, D_BALTIC, D_LATIN1, D_LATIN2, D_CYRILLIC, D_TURKISH, D_LATIN1EURO, D_PORTUGUESE, D_ICELANDIC, D_HEBREW, D_CANADIANFRENCH, D_ARABIC, D_NORDIC, D_CYRILLICRUSSIAN, D_GREEK2, D_THAI, D_ARABICASMO • **ISO code pages** - ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, ISO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, ISO_8859_11, ISO_8859_13, ISO_8859_14, ISO_8859_15, ISO_8859_16 • **Other code pages** - O_KOI8R, O_KOI8U, O_KOI8RU, O_KOI8UNI, O_BIG5, O_GB12345, O_GB2312, O_JIS0201, O_JIS0208, O_JIS0212, O_JOHAB, O_KSC5601, O_KSX1001, O_WANSUNG, O_GB18030 • **EBCDIC code pages** - E_DFXDEFAULT, E_USCANADA, E_LATIN5TURKISH, E_INTERNATIONAL, E_GREEK, E_HEBREW, E_ROECELATIN2, E_JAPANESEKATAKANA_EX, E_ARABIC, E_KOREAN_EX, E_CYRILLIC_RUSSIAN, E_LATIN1_EURO, E_CYRILLIC_S_EUROPE, E_USCANADA_EURO, E_GERMANY_EURO, E_DENMARKNORWAY_EURO, E_FINLANDSWEDEN_EURO, E_ITALY_EURO, E_SPANISH_EURO, E_UK_EURO, E_FRANCE_EURO, E_INTL_EURO, E_ICELAND_EURO • **Note**: On the Unicode encodings, a suffix of *LE* means Little Endian and *BE* means Big Endian and this refers to text storage. Refer to the ByteOrder property for specifying the storage of binary numbers.

Property	Meaning
ErrorHandler	Controls the processing of errors in the file. The valid choices are: • **Automatic** - (default) Transall will automatically handle and recover from errors. Transall will “throw” only critical errors that must be “caught” via an On Error Resume Next type statement. • **Manual** - Transall will ignore all errors and return errors to your Scripts or LogicTrees for handling.

COBOL File

Property	Meaning
Access	Controls the file access method. The valid choices are: • **Read** - (default) If this is a new file, Transall will create it with read access. • **Read Write** - If this is a new file, Transall will create it with read / write access.
FileName	Pathname for the file containing data records.
LockMode	Controls the file access permission. The valid choices are: • **Write** - (default) Transall will set the file's access permission to lock out other processes from writing to the file while Transall has the file open for reading. • **Read** - Transall will set the file's access permission to lock out other processes from reading the file while Transall has the file open for reading. • **Read Write** - Transall will set the file's access permission to lock out other processes from having any access to the file while Transall has the file open for reading. • **Shared** - Transall will NOT set the file's access permission to lock out other processes from having access to the file while Transall has the file open for reading.
Mode	Controls the file input access. The valid choices are: • **Input** - (default) Transall will open the file for input access in "cooked" or translated mode. In this compatibility mode, carriage-return / line-feed combinations are converted to line-feed symbols. This yields a common compatibility with UNIX file systems. • **Binary Input** - Transall will open the file for input access in "raw" or untranslated mode. In this mode, carriage-return / line-feed combinations are NOT converted to line-feed symbols.
OpenMode	Controls the generation of logic to open a file. The valid choices are: • **Automatic** - (default) Transall will automatically generate logic to open this file for you. • **Manual** -Transall will generate logic to open this file but will not automatically call the logic. You must manually place a call to the generated logic in your Transall Scripts or LogicTrees to have Transall run the instructions that open this file. You might want to select **Manual** if you need to have extra control over when a file is opened by Transall. This may be the case if you have a need to open and close a file several times during a Transall run. By default, the **Automatic** open mode opens the file and holds the file open for the life of the Transall run, closing the file when Transall is ready to terminate.

Property	Meaning
ByteOrder	Controls the processing of binary data in the file. The valid choices are: • **Any** - (default) Transall will assume any binary data being accessed in this file is in the native binary format of the platform Transall is running on. • **Little-Endian** -Transall will assume any binary data being accessed in this file is in little endian (least significant bits in) format and Transall will perform any binary manipulation required to process the data on the platform that Transall is operating on. **Note:** Data on the WIN32 platform is Little Endian. • **Big-Endian** -Transall will assume any binary data being accessed in this file is in big endian (most significant bits in) format and Transall will perform any binary manipulation required to process the data on the platform that Transall is operating on.

Property	Meaning
CharacterSet	Controls the processing of string data in the file. The valid choices are: • **Any** - (default) Transall will assume string data being accessed in this file is in the native character set of the platform Transall is running on. • **ASCII** - Transall will assume string data being accessed in this file is in the ASCII character set and Transall will perform any string manipulation required to process the data on the platform that Transall is operating on. • **EBCDIC** - Transall will assume string data being accessed in this file is in the EBCDIC character set and Transall will perform any string manipulation required to process the data on the platform that Transall is operating on. Or • **Unicode encodings** - UTF-8, UTF-16, UTF-16LE, UTF-16BE, UCS-2, UCS-2LE, UCS-2BE, UCS-4, UTF-32, UCS-4LE, UTF-32LE, UCS-4BE, UTF-32BE, UTS-6 • **Windows code pages** - W_CENTRAL_EUROPE, W_CYRILLIC, W_LATIN1, W_GREEK, W_LATIN5, W_HEBREW, W_ARABIC, W_BALTIC, W_VIETNAMESE, W_THAI, W_JAPANESE, W_KOREAN, W_S_CHINESE, W_T_CHINESE • **DOS code pages** - D_USLATIN, D_ARABIC1, D_GREEK, D_BALTIC, D_LATIN1, D_LATIN2, D_CYRILLIC, D_TURKISH, D_LATIN1EURO, D_PORTUGUESE, D_ICELANDIC, D_HEBREW, D_CANADIANFRENCH, D_ARABIC, D_NORDIC, D_CYRILLICRUSSIAN, D_GREEK2, D_THAI, D_ARABICASMO • **ISO code pages** - ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, ISO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, ISO_8859_11, ISO_8859_13, ISO_8859_14, ISO_8859_15, ISO_8859_16 • **Other code pages** - O_KOI8R, O_KOI8U, O_KOI8RU, O_KOI8UNI, O_BIG5, O_GB12345, O_GB2312, O_JIS0201, O_JIS0208, O_JIS0212, O_JOHAB, O_KSC5601, O_KSX1001, O_WANSUNG, O_GB18030 • **EBCDIC code pages** - E_DFXDEFAULT, E_USCANADA, E_LATIN5TURKISH, E_INTERNATIONAL, E_GREEK, E_HEBREW, E_ROECELATIN2, E_JAPANESEKATAKANA_EX, E_ARABIC, E_KOREAN_EX, E_CYRILLIC_RUSSIAN, E_LATIN1_EURO, E_CYRILLIC_S_EUROPE, E_USCANADA_EURO, E_GERMANY_EURO, E_DENMARKNORWAY_EURO, E_FINLANDSWEDEN_EURO, E_ITALY_EURO, E_SPANISH_EURO, E_UK_EURO, E_FRANCE_EURO, E_INTL_EURO, E_ICELAND_EURO • **Note**: On the Unicode encodings, a suffix of *LE* means Little Endian and *BE* means Big Endian and this refers to text storage. Refer to the ByteOrder property for specifying the storage of binary numbers.

Property	Meaning
Separator	This is a reference to a Transall resource (set up under **Tools>Options>Separators**) that is either a single character or a string of characters that Transall will look for in the file to locate each data record.
ErrorHandler	Controls the processing of errors in the file. The valid choices are: • **Automatic** - (default) Transall will automatically handle and recover from errors. Transall will "throw" only critical errors that must be "caught" via an On Error Resume Next type statement. • **Manual** - Transall will ignore all errors and return errors to your Scripts or LogicTrees for handling.

PPS File

Property	Meaning
Access	Controls the file access method. The valid choices are: • **Read** - (default) If this is a new file, Transall will create it with read access. • **Read Write** - If this is a new file, Transall will create it with read / write access.
FileName	Pathname for the file containing data records.
LockMode	Controls the file access permission. The valid choices are: • **Write** - (default) Transall will set the file's access permission to lock out other processes from writing to the file while Transall has the file open for reading. • **Read** - Transall will set the file's access permission to lock out other processes from reading the file while Transall has the file open for reading. • **Read Write** - Transall will set the file's access permission to lock out other processes from having any access to the file while Transall has the file open for reading. • **Shared** - Transall will NOT set the file's access permission to lock out other processes from having access to the file while Transall has the file open for reading.
Mode	Controls the file input access. The valid choices are: • **Input** - (default) Transall will open the file for input access in "cooked" or translated mode. In this compatibility mode, carriage-return / line-feed combinations are converted to line-feed symbols. This yields a common compatibility with UNIX file systems. • **Binary Input** - Transall will open the file for input access in "raw" or untranslated mode. In this mode, carriage-return / line-feed combinations are NOT converted to line-feed symbols.
OpenMode	Controls the generation of logic to open a file. The valid choices are: • **Automatic** - (default) Transall will automatically generate logic to open this file for you. • **Manual** -Transall will generate logic to open this file but will not automatically call the logic. You must manually place a call to the generated logic in your Transall Scripts or LogicTrees to have Transall run the instructions that open this file. You might want to select **Manual** if you need to have extra control over when a file is opened by Transall. This may be the case if you have a need to open and close a file several times during a Transall run. By default, the **Automatic** open mode opens the file and holds the file open for the life of the Transall run, closing the file when Transall is ready to terminate.

DESTINATION PROPERTIES

There follows a list of the allowable values for the unique properties for each kind of Destination component.

Fixed File

Property	Meaning
Access	Controls the file access method. The valid choices are: • **Read** - (default) If this is a new file (destination only), Transall will create it with read access. • **Read Write** - If this is a new file, Transall will create it with read / write access.
FileName	Pathname for the file containing data records.
LockMode	Controls the file access permission. The valid choices are: • **Write** - (default) Transall will set the file's access permission to lock out other processes from writing to the file while Transall has the file open for reading. • **Read** - Transall will set the file's access permission to lock out other processes from reading the file while Transall has the file open for reading. • **Read Write** - Transall will set the file's access permission to lock out other processes from having any access to the file while Transall has the file open for reading. • **Shared** - Transall will NOT set the file's access permission to lock out other processes from having access to the file while Transall has the file open for reading.
Mode	Controls the file input access. The valid choices are: • **Input** - (default) Transall will open the file for input access in "cooked" or translated mode. In this compatibility mode, carriage-return / line-feed combinations are converted to line-feed symbols. This yields a common compatibility with UNIX file systems. • **Binary Input** - Transall will open the file for input access in "raw" or untranslated mode. In this mode, carriage-return / line-feed combinations are NOT converted to line-feed symbols.

Property	Meaning
OpenMode	Controls the generation of logic to open a file. The valid choices are: • **Automatic** - (default) Transall will automatically generate logic to open this file for you. • **Manual** -Transall will generate logic to open this file but will not automatically call the logic. You must manually place a call to the generated logic in your Transall Scripts or LogicTrees to have Transall run the instructions that open this file. You might want to select **Manual** if you need to have extra control over when a file is opened by Transall. This may be the case if you have a need to open and close a file several times during a Transall run. By default, the **Automatic** open mode opens the file and holds the file open for the life of the Transall run, closing the file when Transall is ready to terminate.
Separator	This is a reference to a Transall resource (set up under **Tools>Options>Separators**) that is either a single character or a string of characters that Transall will look for in the file to locate each data record.
ByteOrder	Controls the processing of binary data in the file. The valid choices are: • **Any** - (default) Transall will assume any binary data being accessed in this file is in the native binary format of the platform Transall is running on. • **Little-Endian** -Transall will assume any binary data being accessed in this file is in little endian (least significant bits in) format and Transall will perform any binary manipulation required to process the data on the platform that Transall is operating on. **Note:** Data on the WIN32 platform is Little Endian. • **Big-Endian** -Transall will assume any binary data being accessed in this file is in big endian (most significant bits in) format and Transall will perform any binary manipulation required to process the data on the platform that Transall is operating on.

Property	Meaning
CharacterSet	Controls the processing of string data in the file. The valid choices are: • **Any** - (default) Transall will assume string data being accessed in this file is in the native character set of the platform Transall is running on. • **ASCII** - Transall will assume string data being accessed in this file is in the ASCII character set and Transall will perform any string manipulation required to process the data on the platform that Transall is operating on. • **EBCDIC** - Transall will assume string data being accessed in this file is in the EBCDIC character set and Transall will perform any string manipulation required to process the data on the platform that Transall is operating on. Or • **Unicode encodings** - UTF-8, UTF-16, UTF-16LE, UTF-16BE, UCS-2, UCS-2LE, UCS-2BE, UCS-4, UTF-32, UCS-4LE, UTF-32LE, UCS-4BE, UTF-32BE, UTS-6 • **Windows code pages** - W_CENTRAL_EUROPE, W_CYRILLIC, W_LATIN1, W_GREEK, W_LATIN5, W_HEBREW, W_ARABIC, W_BALTIC, W_VIETNAMESE, W_THAI, W_JAPANESE, W_KOREAN, W_S_CHINESE, W_T_CHINESE • **DOS code pages** - D_USLATIN, D_ARABIC1, D_GREEK, D_BALTIC, D_LATIN1, D_LATIN2, D_CYRILLIC, D_TURKISH, D_LATIN1EURO, D_PORTUGUESE, D_ICELANDIC, D_HEBREW, D_CANADIANFRENCH, D_ARABIC, D_NORDIC, D_CYRILLICRUSSIAN, D_GREEK2, D_THAI, D_ARABICASMO • **ISO code pages** - ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, ISO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, ISO_8859_11, ISO_8859_13, ISO_8859_14, ISO_8859_15, ISO_8859_16 • **Other code pages** - O_KOI8R, O_KOI8U, O_KOI8RU, O_KOI8UNI, O_BIG5, O_GB12345, O_GB2312, O_JIS0201, O_JIS0208, O_JIS0212, O_JOHAB, O_KSC5601, O_KSX1001, O_WANSUNG, O_GB18030 • **EBCDIC code pages** - E_DFXDEFAULT, E_USCANADA, E_LATIN5TURKISH, E_INTERNATIONAL, E_GREEK, E_HEBREW, E_ROECELATIN2, E_JAPANESEKATAKANA_EX, E_ARABIC, E_KOREAN_EX, E_CYRILLIC_RUSSIAN, E_LATIN1_EURO, E_CYRILLIC_S_EUROPE, E_USCANADA_EURO, E_GERMANY_EURO, E_DENMARKNORWAY_EURO, E_FINLANDSWEDEN_EURO, E_ITALY_EURO, E_SPANISH_EURO, E_UK_EURO, E_FRANCE_EURO, E_INTL_EURO, E_ICELAND_EURO • **Note**: On the Unicode encodings, a suffix of *LE* means Little Endian and *BE* means Big Endian and this refers to text storage. Refer to the ByteOrder property for specifying the storage of binary numbers.

Property	Meaning
ErrorHandler	Controls the processing of errors in the file. The valid choices are: • **Automatic** - (default) Transall will automatically handle and recover from errors. Transall will "throw" only critical errors that must be "caught" via an On Error Resume Next type statement. • **Manual** - Transall will ignore all errors and return errors to your Scripts or LogicTrees for handling.

Delimited File

Property	Meaning
Access	Controls the file access method. The valid choices are: • **Read** - (default) If this is a new file, Transall will create it with read access. • **Read Write** - If this is a new file, Transall will create it with read / write access.
FieldDelimiter	This is either a single character or a string of characters that Transall will look for in the data fields to locate each field's data value. The default for this is a double quote. If the incoming data has no double quotes delimiting a field's value, Transall will still correctly process the field.
FieldSeparator	This is either a single character or a string of characters that Transall will look for in the data records to locate each data field. The default for this is a comma.
FileName	Pathname for the file containing data records.
LockMode	Controls the file access permission. The valid choices are: • **Write** - (default) Transall will set the file's access permission to lock out other processes from writing to the file while Transall has the file open for reading. • **Read** - Transall will set the file's access permission to lock out other processes from reading the file while Transall has the file open for reading. • **Read Write** - Transall will set the file's access permission to lock out other processes from having any access to the file while Transall has the file open for reading. • **Shared** - Transall will NOT set the file's access permission to lock out other processes from having access to the file while Transall has the file open for reading.
Mode	Controls the file input access. The valid choices are: • **Input** - (default) Transall will open the file for input access in "cooked" or translated mode. In this compatibility mode, carriage-return / line-feed combinations are converted to line-feed symbols. This yields a common compatibility with UNIX file systems. • **Binary Input** - Transall will open the file for input access in "raw" or untranslated mode. In this mode, carriage-return / line-feed combinations are NOT converted to line-feed symbols.

Property	Meaning
OpenMode	Controls the generation of logic to open a file. The valid choices are: • **Automatic** - (default) Transall will automatically generate logic to open this file for you. • **Manual** -Transall will generate logic to open this file but will not automatically call the logic. You must manually place a call to the generated logic in your Transall Scripts or LogicTrees to have Transall run the instructions that open this file. You might want to select **Manual** if you need to have extra control over when a file is opened by Transall. This may be the case if you have a need to open and close a file several times during a Transall run. By default, the **Automatic** open mode opens the file and holds the file open for the life of the Transall run, closing the file when Transall is ready to terminate.
Separator	This is a reference to a Transall resource (set up under **Tools>Options>Separators**) that is either a single character or a string of characters that Transall will look for in the file to locate each data record.
ByteOrder	Controls the processing of binary data in the file. The valid choices are: • **Any** - (default) Transall will assume any binary data being accessed in this file is in the native binary format of the platform Transall is running on. • **Little-Endian** -Transall will assume any binary data being accessed in this file is in little endian (least significant bits in) format and Transall will perform any binary manipulation required to process the data on the platform that Transall is operating on. **Note:** Data on the WIN32 platform is Little Endian. • **Big-Endian** -Transall will assume any binary data being accessed in this file is in big endian (most significant bits in) format and Transall will perform any binary manipulation required to process the data on the platform that Transall is operating on.

Property	Meaning
CharacterSet	Controls the processing of string data in the file. The valid choices are: • **Any** - (default) Transall will assume string data being accessed in this file is in the native character set of the platform Transall is running on. • **ASCII** - Transall will assume string data being accessed in this file is in the ASCII character set and Transall will perform any string manipulation required to process the data on the platform that Transall is operating on. • **EBCDIC** - Transall will assume string data being accessed in this file is in the EBCDIC character set and Transall will perform any string manipulation required to process the data on the platform that Transall is operating on. Or • **Unicode encodings** - UTF-8, UTF-16, UTF-16LE, UTF-16BE, UCS-2, UCS-2LE, UCS-2BE, UCS-4, UTF-32, UCS-4LE, UTF-32LE, UCS-4BE, UTF-32BE, UTS-6 • **Windows code pages** - W_CENTRAL_EUROPE, W_CYRILLIC, W_LATIN1, W_GREEK, W_LATIN5, W_HEBREW, W_ARABIC, W_BALTIC, W_VIETNAMESE, W_THAI, W_JAPANESE, W_KOREAN, W_S_CHINESE, W_T_CHINESE • **DOS code pages** - D_USLATIN, D_ARABIC1, D_GREEK, D_BALTIC, D_LATIN1, D_LATIN2, D_CYRILLIC, D_TURKISH, D_LATIN1EURO, D_PORTUGUESE, D_ICELANDIC, D_HEBREW, D_CANADIANFRENCH, D_ARABIC, D_NORDIC, D_CYRILLICRUSSIAN, D_GREEK2, D_THAI, D_ARABICASMO • **ISO code pages** - ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, ISO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, ISO_8859_11, ISO_8859_13, ISO_8859_14, ISO_8859_15, ISO_8859_16 • **Other code pages** - O_KOI8R, O_KOI8U, O_KOI8RU, O_KOI8UNI, O_BIG5, O_GB12345, O_GB2312, O_JIS0201, O_JIS0208, O_JIS0212, O_JOHAB, O_KSC5601, O_KSX1001, O_WANSUNG, O_GB18030 • **EBCDIC code pages** - E_DFXDEFAULT, E_USCANADA, E_LATIN5TURKISH, E_INTERNATIONAL, E_GREEK, E_HEBREW, E_ROECELATIN2, E_JAPANESEKATAKANA_EX, E_ARABIC, E_KOREAN_EX, E_CYRILLIC_RUSSIAN, E_LATIN1_EURO, E_CYRILLIC_S_EUROPE, E_USCANADA_EURO, E_GERMANY_EURO, E_DENMARKNORWAY_EURO, E_FINLANDSWEDEN_EURO, E_ITALY_EURO, E_SPANISH_EURO, E_UK_EURO, E_FRANCE_EURO, E_INTL_EURO, E_ICELAND_EURO • **Note**: On the Unicode encodings, a suffix of *LE* means Little Endian and *BE* means Big Endian and this refers to text storage. Refer to the ByteOrder property for specifying the storage of binary numbers.

Property	Meaning
ErrorHandler	Controls the processing of errors in the file. The valid choices are: • **Automatic** - (default) Transall will automatically handle and recover from errors. Transall will "throw" only critical errors that must be "caught" via an On Error Resume Next type statement. • **Manual** - Transall will ignore all errors and return errors to your Scripts or LogicTrees for handling.

COBOL File

Property	Meaning
Access	Controls the file access method. The valid choices are: • **Read** - (default) If this is a new file, Transall will create it with read access. • **Read Write** - If this is a new file, Transall will create it with read / write access.
FileName	Pathname for the file containing data records.
LockMode	Controls the file access permission. The valid choices are: • **Write** - (default) Transall will set the file's access permission to lock out other processes from writing to the file while Transall has the file open for reading. • **Read** - Transall will set the file's access permission to lock out other processes from reading the file while Transall has the file open for reading. • **Read Write** - Transall will set the file's access permission to lock out other processes from having any access to the file while Transall has the file open for reading. • **Shared** - Transall will NOT set the file's access permission to lock out other processes from having access to the file while Transall has the file open for reading.
Mode	Controls the file input access. The valid choices are: • **Input** - (default) Transall will open the file for input access in "cooked" or translated mode. In this compatibility mode, carriage-return / line-feed combinations are converted to line-feed symbols. This yields a common compatibility with UNIX file systems. • **Binary Input** - Transall will open the file for input access in "raw" or untranslated mode. In this mode, carriage-return / line-feed combinations are NOT converted to line-feed symbols.
OpenMode	Controls the generation of logic to open a file. The valid choices are: • **Automatic** - (default) Transall will automatically generate logic to open this file for you. • **Manual** -Transall will generate logic to open this file but will not automatically call the logic. You must manually place a call to the generated logic in your Transall Scripts or LogicTrees to have Transall run the instructions that open this file. You might want to select **Manual** if you need to have extra control over when a file is opened by Transall. This may be the case if you have a need to open and close a file several times during a Transall run. By default, the **Automatic** open mode opens the file and holds the file open for the life of the Transall run, closing the file when Transall is ready to terminate.

Property	Meaning
ByteOrder	Controls the processing of binary data in the file. The valid choices are: • **Any** - (default) Transall will assume any binary data being accessed in this file is in the native binary format of the platform Transall is running on. • **Little-Endian** -Transall will assume any binary data being accessed in this file is in little endian (least significant bits in) format and Transall will perform any binary manipulation required to process the data on the platform that Transall is operating on. **Note:** Data on the WIN32 platform is Little Endian. • **Big-Endian** -Transall will assume any binary data being accessed in this file is in big endian (most significant bits in) format and Transall will perform any binary manipulation required to process the data on the platform that Transall is operating on.

Property	Meaning
CharacterSet	Controls the processing of string data in the file. The valid choices are: • **Any** - (default) Transall will assume string data being accessed in this file is in the native character set of the platform Transall is running on. • **ASCII** - Transall will assume string data being accessed in this file is in the ASCII character set and Transall will perform any string manipulation required to process the data on the platform that Transall is operating on. • **EBCDIC** - Transall will assume string data being accessed in this file is in the EBCDIC character set and Transall will perform any string manipulation required to process the data on the platform that Transall is operating on. Or • **Unicode encodings** - UTF-8, UTF-16, UTF-16LE, UTF-16BE, UCS-2, UCS-2LE, UCS-2BE, UCS-4, UTF-32, UCS-4LE, UTF-32LE, UCS-4BE, UTF-32BE, UTS-6 • **Windows code pages** - W_CENTRAL_EUROPE, W_CYRILLIC, W_LATIN1, W_GREEK, W_LATIN5, W_HEBREW, W_ARABIC, W_BALTIC, W_VIETNAMESE, W_THAI, W_JAPANESE, W_KOREAN, W_S_CHINESE, W_T_CHINESE • **DOS code pages** - D_USLATIN, D_ARABIC1, D_GREEK, D_BALTIC, D_LATIN1, D_LATIN2, D_CYRILLIC, D_TURKISH, D_LATIN1EURO, D_PORTUGUESE, D_ICELANDIC, D_HEBREW, D_CANADIANFRENCH, D_ARABIC, D_NORDIC, D_CYRILLICRUSSIAN, D_GREEK2, D_THAI, D_ARABICASMO • **ISO code pages** - ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, ISO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, ISO_8859_11, ISO_8859_13, ISO_8859_14, ISO_8859_15, ISO_8859_16 • **Other code pages** - O_KOI8R, O_KOI8U, O_KOI8RU, O_KOI8UNI, O_BIG5, O_GB12345, O_GB2312, O_JIS0201, O_JIS0208, O_JIS0212, O_JOHAB, O_KSC5601, O_KSX1001, O_WANSUNG, O_GB18030 • **EBCDIC code pages** - E_DFXDEFAULT, E_USCANADA, E_LATIN5TURKISH, E_INTERNATIONAL, E_GREEK, E_HEBREW, E_ROECELATIN2, E_JAPANESEKATAKANA_EX, E_ARABIC, E_KOREAN_EX, E_CYRILLIC_RUSSIAN, E_LATIN1_EURO, E_CYRILLIC_S_EUROPE, E_USCANADA_EURO, E_GERMANY_EURO, E_DENMARKNORWAY_EURO, E_FINLANDSWEDEN_EURO, E_ITALY_EURO, E_SPANISH_EURO, E_UK_EURO, E_FRANCE_EURO, E_INTL_EURO, E_ICELAND_EURO • **Note**: On the Unicode encodings, a suffix of *LE* means Little Endian and *BE* means Big Endian and this refers to text storage. Refer to the ByteOrder property for specifying the storage of binary numbers.
Separator	This is a reference to a Transall resource (set up under **Tools>Options>Separators**) that is either a single character or a string of characters that Transall will look for in the file to locate each data record.

Property	Meaning
ErrorHandler	Controls the processing of errors in the file. The valid choices are: • **Automatic** - (default) Transall will automatically handle and recover from errors. Transall will "throw" only critical errors that must be "caught" via an On Error Resume Next type statement. • **Manual** - Transall will ignore all errors and return errors to your Scripts or LogicTrees for handling.

Documaker File - Variable Replacement File (VRF)

Property	Meaning
FileName	Pathname for the file containing data records.
MessageFile	User-specified name of a file where messages are written during generation of the VRF file.
OpenMode	Controls the generation of logic to open a file. The valid choices are: • **Automatic** - (default) Transall will automatically generate logic to open this file for you. • **Manual** -Transall will generate logic to open this file but will not automatically call the logic. You must manually place a call to the generated logic in your Transall Scripts or LogicTrees to have Transall run the instructions that open this file. You might want to select **Manual** if you need to have extra control over when a file is opened by Transall. This may be the case if you have a need to open and close a file several times during a Transall run. By default, the **Automatic** open mode opens the file and holds the file open for the life of the Transall run, closing the file when Transall is ready to terminate.
WorkingDir	Pathname of the working directory when writing the VRF file occurs.
DmgrFmt(370)	Determines whether to generate a VLAM or Flat File EDL. The valid values are: • **True** - If set to true then, when Transall is running on the 370 platform, Transall will call the Documaker FP DMGRFMT and DMGVRFWR VDR APIs to create a VRF. These APIs create a VRF via calls to a VLAM-based EDL and RuleBase. • **False** - If set to false then, when Transall is running on the 370 platform, Transall will call the DMKVAE VDR API to create a VRF. This creates a VRF via calls to the new Flat File-based EDL and RuleBase.
ErrorHandler	Controls the processing of errors in the file. The valid choices are: • **Automatic** - (default) Transall will automatically handle and recover from errors. Transall will "throw" only critical errors that must be "caught" via an On Error Resume Next type statement. • **Manual** - Transall will ignore all errors and return errors to your Scripts or LogicTrees for handling.

Property	Meaning
FormChain	META, AFP, or DCD. This is the EDL chain type used at development time to “dump forms” for computing form DOT height and locating form tags for mapping in Transall.
FormsLibAutoAdd	Determines how to add the Electronic Document Library (EDL) listed in the Component Inspector. The valid values are: True - The script generated to open the VRF destination file automatically adds the EDL to the list used for building the VRF. False - The Transall developer must explicitly add all EDLs for use by the VRF via Scripts or LogicTree instructions.
FormsLibrary	This is the name of EDL used at development time to “dump forms” for computing form DOT height and locating form tags for mapping in Transall.
FormsLibraryType	This is the type of EDL to be used at development time to “dump forms” for computing form DOT height and locating form tags for mapping in Transall. The valid values are **Database** or **File**.
RuleBaseType	This is the type of the RuleBase to be used at run-time. The valid values are: • **Database** - Transall calls MRGUSER.W32 to create a VRF. • **File** - Transall calls DMKUSER.W32 to create a VRF.
TGAFile	This is a Tagcommander Export file that can be imported into at development time to predefine the available tag list.

Note For FpPlus data destinations, see *FpPlus Data Destination Details* on page 201.

REFERENCE FOR COMPONENT PROPERTIES - SQL

REFERENCE FOR COMPONENT PROPERTIES - ODBC

This section presents the allowable values for the unique properties for each kind of Source and Destination component. These values can be viewed by going to the Component Inspector, under the Properties tab, then find the properties listed below. By clicking in the box to the right of the property, will select the word and display a control, that when clicked on will display a drop down box, that will show the subcomponent's properties.

SOURCE PROPERTIES

There follows a list of the allowable values for the unique properties for each kind of Source component.

ODBC Data Source

Property	Meaning
ConnectionMode	Controls the generation of logic to connect to a data source. The valid choices are: • **Automatic** - (default) Transall will automatically generate logic to connect to this data source for you. • **Manual** - Transall will generate logic to connect to this data source but will not automatically call the logic. You must manually place a call to the generated logic in your Transall Scripts or LogicTrees to have Transall run the instructions that connect to this data source. You might want to select **Manual** if you need to have extra control over when Transall connects to a data source. This may be the case if you have a need to connect to and disconnect from a data source several times during a Transall run. By default, the **Automatic** connection mode connects to a data source and holds the connection open for the life of the Transall run (disconnecting when Transall is ready to terminate).
DataSource	The ODBC connection string for the ODBC data source. This field is populated via the ODBC Administrator dialog.
ErrorHandler	Controls the processing of errors in the file. The valid choices are: • **Automatic** - (default) Transall will automatically handle and recover from errors. Transall will "throw" only critical errors that must be "caught" via an On Error Resume Next type statement. • **Manual** - Transall will ignore all errors and return errors to your Scripts or LogicTrees for handling.
ODBCPrompt	Indicates whether Transall should prompt for missing information. The valid values are: • **True** - (default) Transall will prompt for missing information such as user ID and password when connecting to an ODBC database connection. • **False** - Transall will not prompt for missing information, but will instead throw an "unable to connect" error message. Transall will also retain more information when ODBCPrompt is set to False then when this is set to True. When this property is set to **True**, Transall only retains the name of the ODBC connection. When this property is set to **False**, however, Transall retains the name of the ODBC connection and any information provided to connect to the ODBC to access database schema information. This extra information generally includes user ID and password.

DESTINATION PROPERTIES

There follows a list of the allowable values for the unique properties for each kind of Destination component.

ODBC Data Source

Property	Meaning
ConnectionMode	Controls the generation of logic to connect to a data source. The valid choices are: • **Automatic** - (default) Transall will automatically generate logic to connect to this data source for you. • **Manual** - Transall will generate logic to connect to this data source but will not automatically call the logic. You must manually place a call to the generated logic in your Transall Scripts or LogicTrees to have Transall run the instructions that connect to this data source. You might want to select **Manual** if you need to have extra control over when Transall connects to a data source. This may be the case if you have a need to connect to and disconnect from a data source several times during a Transall run. By default, the **Automatic** connection mode connects to a data source and holds the connection open for the life of the Transall run (disconnecting when Transall is ready to terminate).
DataSource	The ODBC connection string for the ODBC data source. This field is populated via the ODBC Administrator dialog.
ErrorHandler	Controls the processing of errors in the file. The valid choices are: • **Automatic** - (default) Transall will automatically handle and recover from errors. Transall will "throw" only critical errors that must be "caught" via an On Error Resume Next type statement. • **Manual** - Transall will ignore all errors and return errors to your Scripts or LogicTrees for handling.
ODBCPrompt	Indicates whether Transall should prompt for missing information. The valid values are: • **True** - (default) Transall will prompt for missing information such as user ID and password when connecting to an ODBC database connection. • **False** - Transall will not prompt for missing information, but will instead throw an "unable to connect" error message. Transall will also retain more information when ODBCPrompt is set to False then when this is set to True. When this property is set to **True**, Transall only retains the name of the ODBC connection. When this property is set to **False**, however, Transall retains the name of the ODBC connection and any information provided to connect to the ODBC to access database schema information. This extra information generally includes user ID and password.

DESCRIBING AN ODBC-BASED SOURCE

After you complete the Add Source dialog, the Transall Editor opens the DataSource Assistant.

Creating and populating an ODBC-based Source component requires more steps than for other kinds of Source components. However, these components can include more powerful data specifications.

You need to describe the data that the Transall Application will read from it by constructing a standards-compliant, Structured Query Language (SQL) statement. This statement identifies the following:

- The tables, provided by the identified ODBC data source, containing the data that the component uses
- The data columns to be included
- `JOIN` operations to combine data from more than one table
- `WHERE` clauses to restrict the data to certain table rows
- `ORDER BY` clauses to identify on which columns (if any) to sort the set of actual data records returned

Fortunately, you needn't be an SQL expert to define the behavior of an ODBC-based Source or Destination, although it helps. Rather, to describe the component's behavior, you work with graphical representations of "SQL queries." The result of doing so is a visible, and editable, SQL statement that the Transall Editor constructs. When the Transall Application runs, it submits this SQL statement to the ODBC data source to read or write the actual data records.

Note Populating an ODBC-based Source means to define a "query" for a data record. This query is the functional equivalent of a Record subcomponent in other kinds of Sources.

The Transall Editor contains a control bar that provides access to, and allows modification of, SQL Filter and Join data.

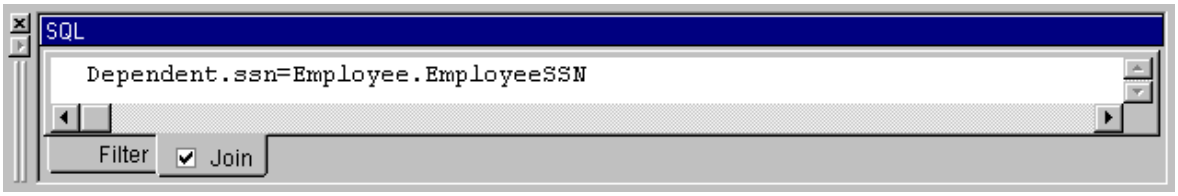


Figure 86: SQL Control Bar

By default, this control bar opens and closes along with the DataSource Assistant, but you can uncouple it by using **Tools>Settings** and unchecking **Automatically show SQL control bar**. You can manually display it by using **View>SQL Bar**. If the control bar contains Filter or Join expressions, you'll see a check mark on that tab.

Create the Source Component

For an ODBC-based Source, respond to the **Project>Add Source** dialog by typing the name of the ODBC data source, or by pressing the Browse button to select from among the ODBC data sources that are available to your workstation.

Note The target ODBC data source must be available to your workstation at the time you create the Source component. Otherwise, you cannot complete the definition of the component.

Define a Query

A **Query** is a subcomponent of an ODBC-based Source or Destination, just as a Record is a subcomponent in other kinds of Sources and Destinations. Each query represents one SQL statement that requests a certain result set, or stream, of actual data records be provided to the Transall Application. By defining more than one query for an ODBC-based Source, you can define more than one stream of actual data records for the same Source.

Note The DataSource Assistant for an ODBC-based Source is organized by record list that represent its Queries, just as the Assistants for other Sources are organized by record list that represent their Records.

Your next step is to define an SQL query. Right-click in the DataSource Assistant's Query List view to add a Query or use the **Resource>Add>Query** menu. This will open the Add Query dialog.

Figure 87 on page 161, shows the dialog, then name the Query, click on the Select Query icon, and then press OK.

In a given ODBC-based Source, for each Query that you define you will refer to one or more of the SQL tables provided by the actual ODBC data source. (You select which tables to include in a Query by opening an Add Table dialog, which is shown in Figure 88 on page 163.).

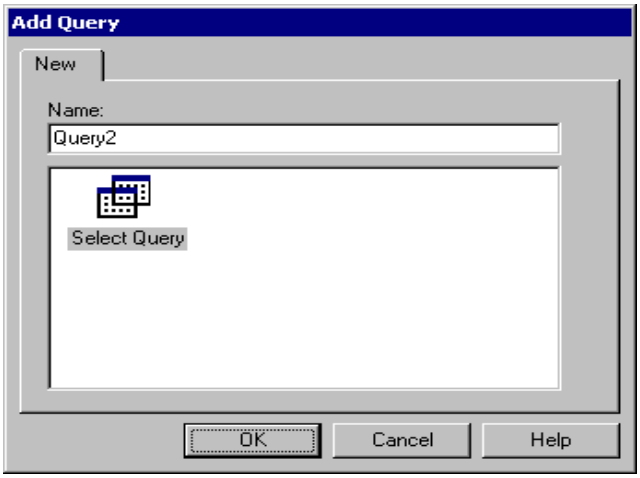


Figure 87: Add Query Dialog

Hint If you are populating an ODBC-based Destination component, you would select the Delete Query, Insert Query, or Update Query icon in the Add Query dialog because these icons represent operations that add, modify, or delete data records via a Destination.

QUERY PROPERTIES

In an open ODBC-based Source or Destination, go to the Component Inspector, under the Properties tab, then find the properties listed below. By clicking in the box to the right of the property, will select the word and display a control, that when clicked on will display a drop down box, that will show the Query subcomponent's properties, as follows:

Property	Meaning
Connect	Determines the type of handle created for executing SQL statements. The valid values are: • **PrivateCursor -** (default) Transall will generate logic to create a single database connection with a private statement handle for executing this SQL statement. This connection often provides the most capabilities while conserving resources with most ODBC drivers. • **Private -** Transall will generate logic to create a private database connection and statement handle for executing this SQL statement. This option should be used when more than one SQL statement needs to be active at the same moment in time and the ODBC driver being used does not support multiple active statements on a single database connection. • **Shared -** Transall will generate logic to create a shared database connection and a shared statement handle for executing this SQL statement. This option can be used when it is not necessary to have more than one SQL statement active at the same moment in time and system resources need to be conserved. **Note:** Not all SQL ODBC drivers support concurrent queries on a single database connection.
Format	Indicates whether to prebind SQL statements to Transall memory tables. The valid values are: • **Standard -** (default) Transall will prebind this SQL statement to Transall memory tables in the compile process for speed. • **Dynamic -** Transall will not prebind this SQL statement to Transall memory tables. This would be used when the SQL statement string needs to be built dynamically at execution time. Very few ODBC drivers require this option.
Type	Determines the type of SQL statement to generate. The only valid value is: • **Select -** Transall will generate a SELECT SQL statement.

Identify One or More Tables for the Query

To review, after you selected an actual ODBC data source for this new Source, the Transall Editor loaded ODBC-provided information about the data source's tables, synonyms, views, and system tables.

Your next step is to choose which tables provided by the ODBC data source contain the columns that contain the data relevant to this query. Click the right mouse button in the "Query Contents" view to display a pop-up menu, and select the **Add Table** command. This can also be done using the main menu **Resource>Add>Table**. This causes the Add Table dialog to appear, in which you select one or more tables that will be included in the query.

Figure 88 on page 163, shows the Add Table dialog.

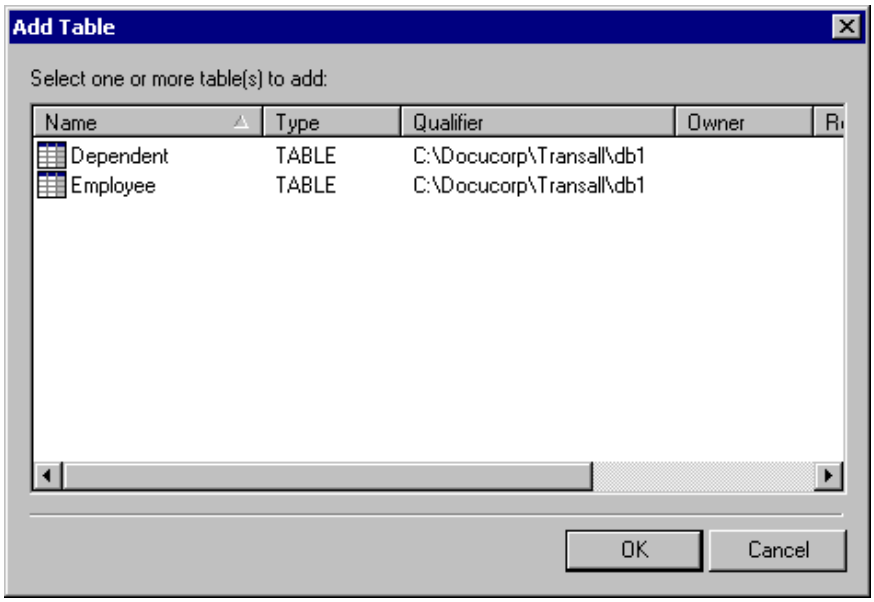


Figure 88: Add Table Dialog

After you select the tables, press OK.

Now the ODBC-based Source's DataSource Assistant contains graphical representations of each table you selected, as shown in *Figure 89 on page 164*.

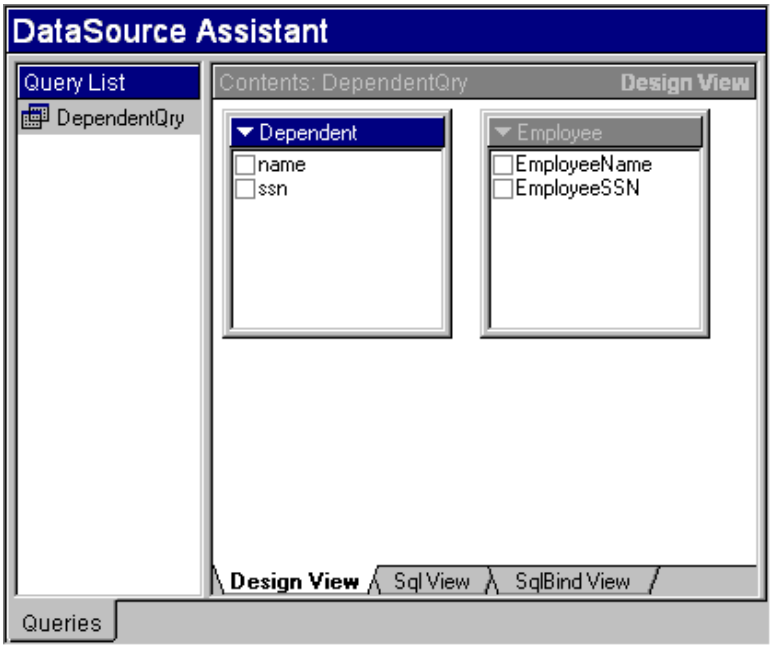


Figure 89: Tables Added to ODBC-Based Source

Selecting Table Columns

Note You can reduce or expand the amount of table information returned in the Add Table dialog by changing the selections for displaying the Schema. These options are available when you right-click in the Contents view and select "Schema Display".

When accessing an Excel file with no named ranges, you must have System Tables selected under Schema Display or you won't see any available items from which to select.

To identify the columns of data to be used in the SQL query, select one or more column names from the table representations shown in the DataSource Assistant.

An example is shown in Figure 90 on page 165, corresponding items have been selected in the both tables.

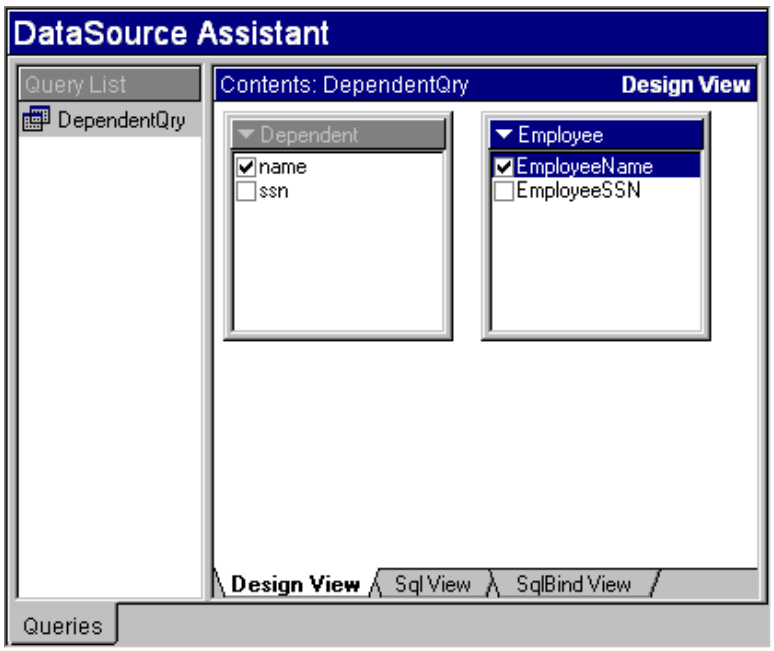


Figure 90: Columns Selected in Tables From ODBC Data Source

Selecting columns causes the Transall Editor to construct a `SELECT` statement for the SQL query that, when performed by the Transall Application, will provide data records for this Source. Click on the SQL View tab at the bottom of the DataSource Assistant to view the constructed statement.

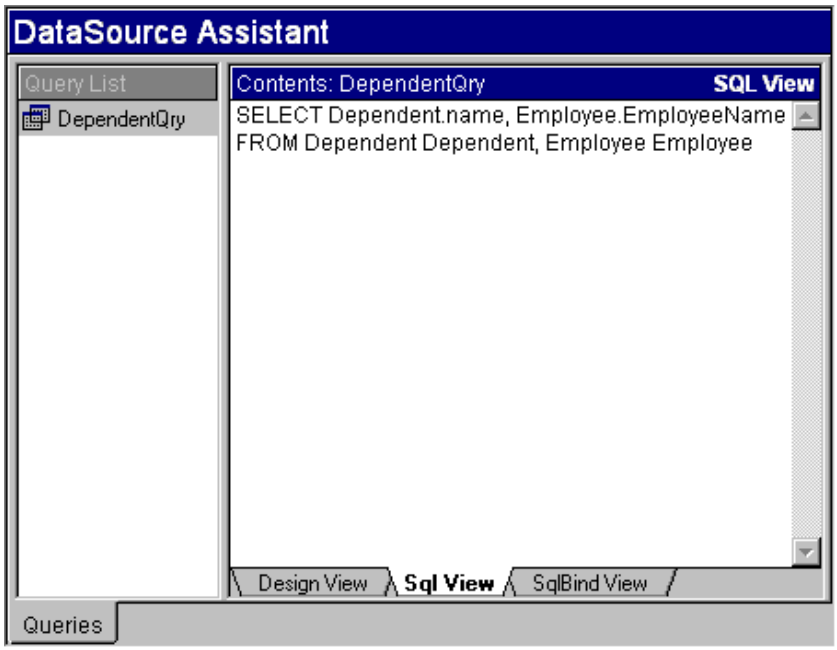


Figure 91: Columns Selected in Tables From ODBC Data Source

Defining Join Criteria for the Query

If you have included more than one table for a given query, you should define the SQL join criteria that describes their relationship. To do so, select the **Resource>Join Criteria** command. (You can also right-click in the Contents view and select the **Join Criteria** command from the pop-up menu.)

This causes the Transall Editor to display the SQL Join Expression Builder dialog, shown in *Figure 92 on page 166*.

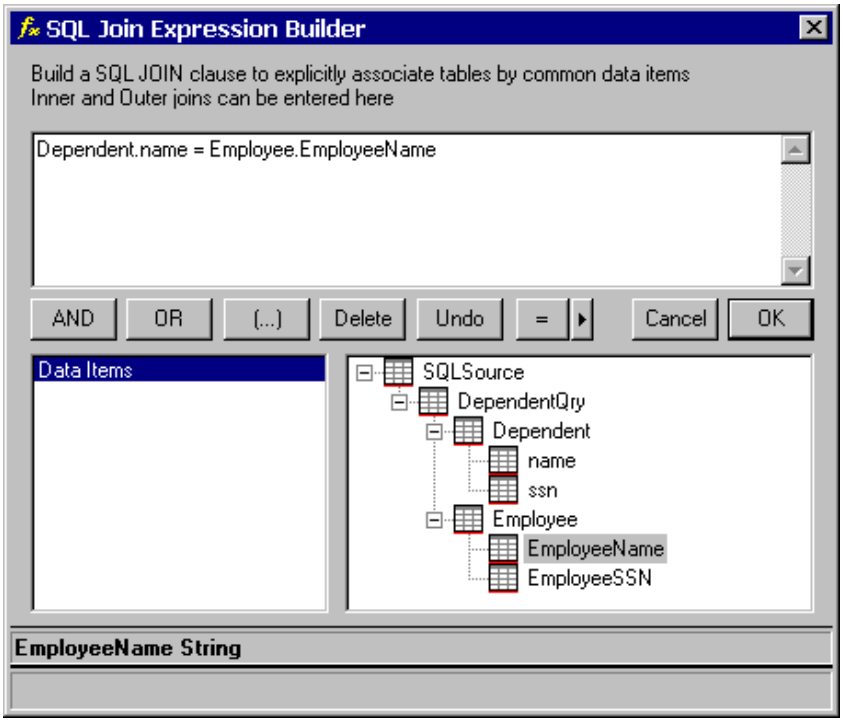


Figure 92: Defining Join Criteria

In the dialog's text block, construct the expression by select the data columns in the tree listing on the right view.

After you finish defining join criteria for this query, press OK.

For more information about the Expression Builder, see *Enhanced Expression Builder* on page 277.

Now the Transall Editor updates the Query's `SELECT` statement with a `WHERE` clause, shown in *Figure 93 on page 167*.

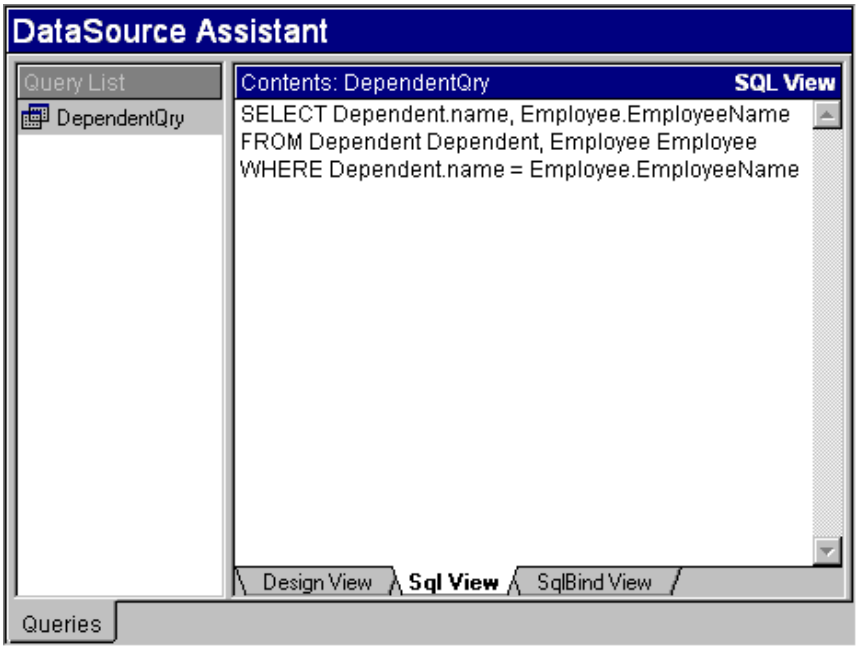


Figure 93: Updated SQL Statement for Query, New Join Criteria

Note If you code associations of table columns only, without a join keyword, Transall codes it as an implicit INNER JOIN and adds this criteria to the WHERE clause, as well as building the FROM clause from tables where you have selected columns. If, however, you code JOIN into this criteria, then you're explicitly coding the FROM clause and the Transall editor simply plugs this into the SQL statement. There's no syntax checking on Transall's part if you code JOIN, though you can still use "Verify Syntax" to have the ODBC driver check the syntax. You can still use "Filter Criteria" to further control the SELECT. The filter criteria will be in the WHERE clause.

Transall uses table name aliases to qualify column names. If there are no special characters, the alias name is the same as the table name; where special characters occur, Transall uses the underscore character (_) to make valid alias names (e.g., the "Order Detail" table has an alias of Order_Detail in Transall). This alias is defined to the SQL back end in the FROM clause as <table name> <alias>. If the table name has special characters, it must be wrapped in the quote character defined in the SQL back end:

```
FROM Products Products INNER JOIN `Order Details`
Order_Details
     ON Order_Details.ProductID = Products.ProductID
```

Defining Filter Criteria

For a given query, you can also restrict the set of actual data records by defining filter criteria. These criteria correspond to a `WHERE` clause in the Query's `SELECT` statement.

Right-click in the Contents view and select the **Filter Criteria** command from the pop-up menu.

This causes the Transall Editor to display the SQL Filter Expression Builder dialog, shown in *Figure 94 on page 168*.

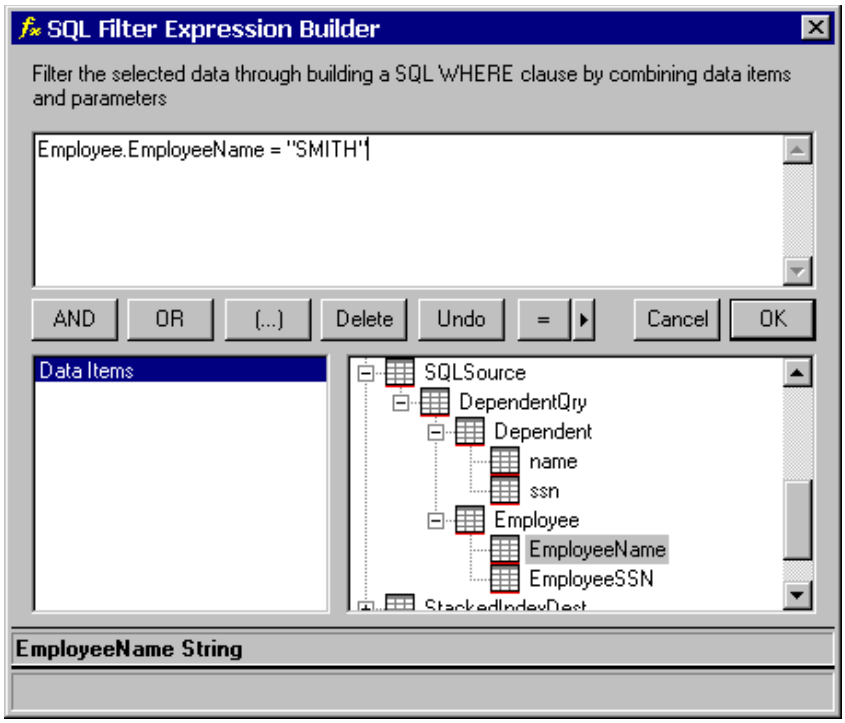


Figure 94: Defining Filter Criteria

In the dialog's text block, type or construct the expressions that identify the rows of data that you want the ODBC data source to provide in the tables already associated with this query.

After you finish defining filter criteria for this query, press OK.

This again causes the Transall Editor to update the `WHERE` clause in the Query's `SELECT` statement, shown in *Figure 95 on page 169*.

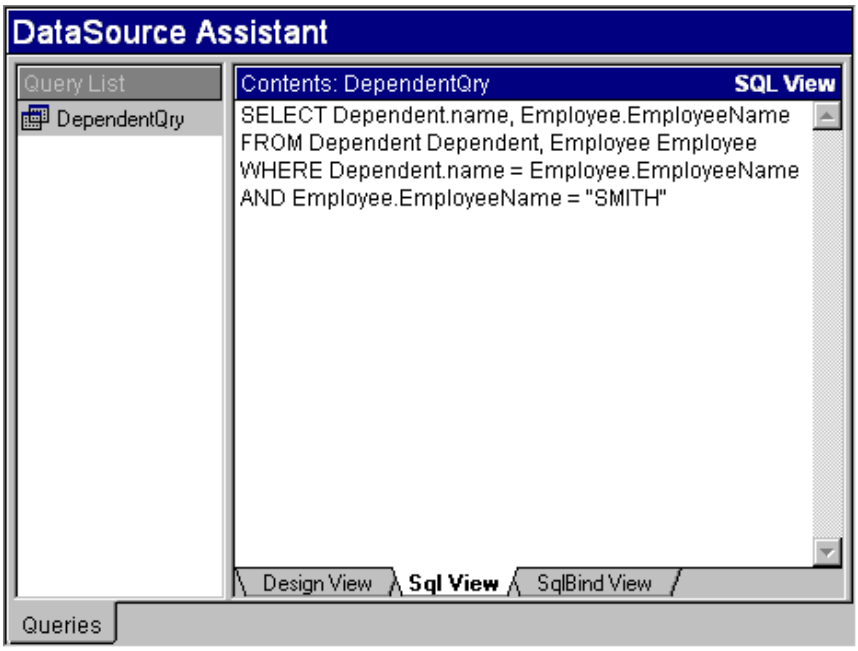


Figure 95: Updated SQL Statement for Query, New Filter Criteria

Define Sort Criteria

For a given Query, you can also define the sort order for the actual records that result from performing this Query for the specified tables provided by the actual ODBC data source. These criteria correspond to an `ORDER BY` clause in this Query's `SELECT` statement.

Right-click in the Contents view and select the **Sort Criteria** command from the pop-up menu.

This causes the Transall Editor to display the Sort dialog window, shown in *Figure 96*.

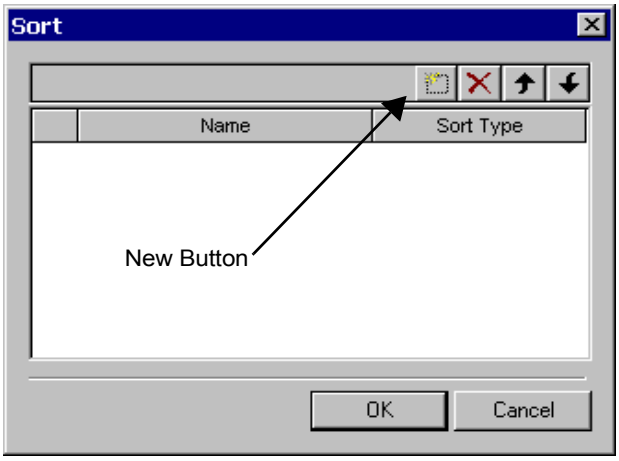


Figure 96: Add New Cell

1. Press the New button to add a row of cells.

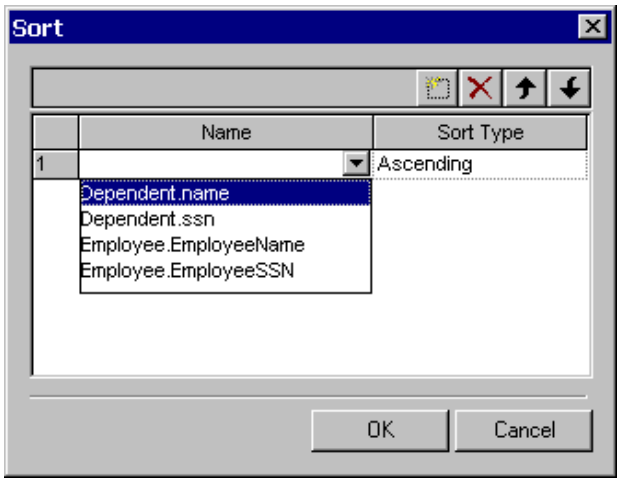


Figure 97: Add Row of Cells

2. On the new row click in the left cell to display a list of all columns for the selected tables in the Query, then select a column.

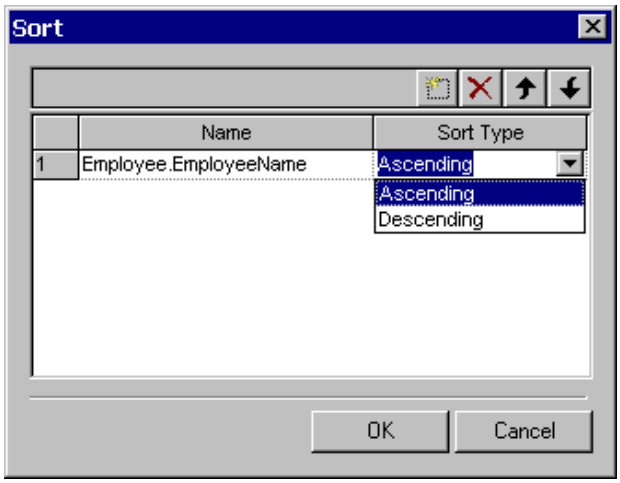


Figure 98: Select Tables and Sort Order

3. On the new row, click in the right cell to specify the sort order (ascending or descending) for the column you just selected.

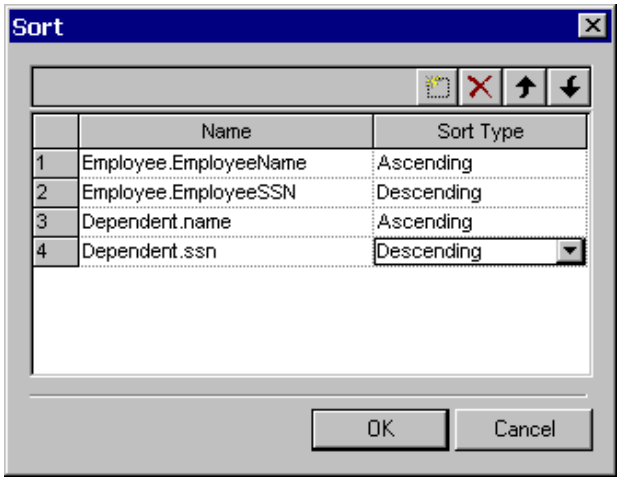


Figure 99: Add More Rows

4. Add more rows repeat steps 1, 2, and 3 in the dialog to specify the sort order of other columns that participate in the Query.

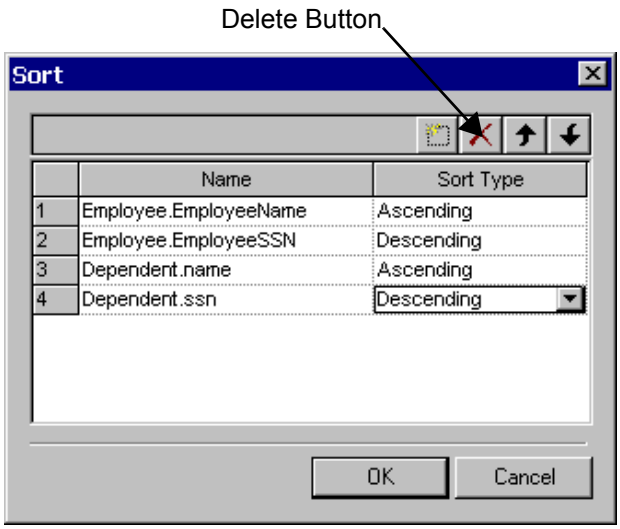


Figure 100: Delete a Sort

5. To delete a sort order specification in this dialog, click on the cell containing the row's number, and press the Delete button.

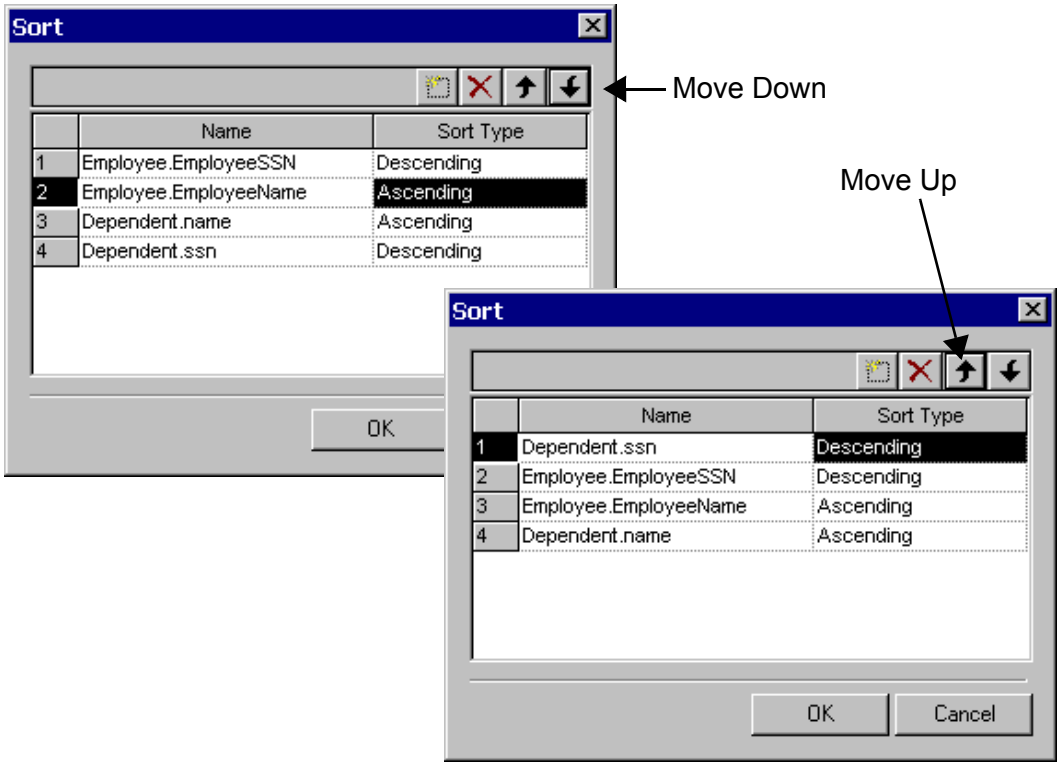


Figure 101: Change Order of List

6. To change the ordering of the list of sort order specifications, click on the cell containing the row's number, and press the Move Up or the Move Down button.
7. After you finish defining the sort criteria for this Query, press OK.

Now the Transall Editor inserts an `ORDER BY` clause in the Query's `SELECT` statement, shown in *Figure 102 on page 173*.

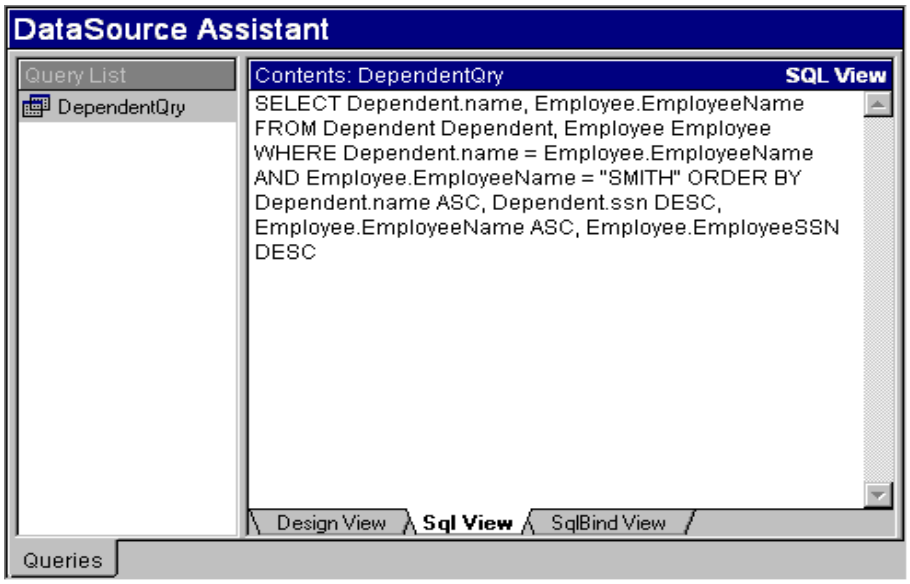


Figure 102: Updated SQL Statement, New Sort Criteria

DESCRIBING AN ODBC-BASED DESTINATION

When adding an ODBC-based Destination, you choose from Insert, Update, or Delete operations. In the Add Query dialog, these are presented as Insert Query, Update Query, and Delete Query icons.

You populate each query object as you do for an ODBC-based Source. You interact with the dialogs just as when describing a Select Query for an ODBC-based Source component.

However, when constructing the Insert Query, Update Query, or Delete Query objects, you can refer only to SQL tables (not views) and to only one SQL table per query object.

Query Properties

In an open ODBC-based Source or Destination, go to the Component Inspector, under the Properties tab, then find the properties listed below. By clicking in the box to the right of the property, will select the word and display a control, that when clicked on will display a drop down box, that will show the Query subcomponent's properties, as follows:

Property	Meaning
Connect	Determines the type of handle created for executing SQL statements. The valid values are: • **PrivateCursor** - (default) Transall will generate logic to create a single database connection with a private statement handle for executing this SQL statement. This connection often provides the most capabilities while conserving resources with most ODBC drivers. • **Private** - Transall will generate logic to create a private database connection and statement handle for executing this SQL statement. This option should be used when more than one SQL statement needs to be active at the same moment in time and the ODBC driver being used does not support multiple active statements on a single database connection. • **Shared** - Transall will generate logic to create a shared database connection and a shared statement handle for executing this SQL statement. This option can be used when it is not necessary to have more than one SQL statement active at the same moment in time and system resources need to be conserved. **Note:** Not all SQL ODBC drivers support concurrent queries on a single database connection.
Format	Indicates whether to prebind SQL statements to Transall memory tables. The valid values are: • **Standard** - (default) Transall will prebind this SQL statement to Transall memory tables in the compile process for speed. • **Dynamic** - Transall will not prebind this SQL statement to Transall memory tables. This would be used when the SQL statement string needs to be built dynamically at execution time. Very few ODBC drivers require this option.
Type	Determines the type of SQL statement to generate. The valid values are: • **Insert** - Transall will generate an INSERT SQL statement. • **Update** - Transall will generate an UPDATE SQL statement. • **Delete** - Transall will generate a DELETE SQL statement.

DETAILS ON SQL

SQL Bind Variables

There are three styles of bind variables available in Transall for passing Transall variable data to ODBC drivers.

Qualifier	Standard Format	Dynamic Format
Colon (:)	Passed as variable	Passed as a string
Caret (^)	n/a	Passed as a string
Question Mark (?)	n/a	Passed as a variable

Colon (:) - Available for queries with Standard or Dynamic format. With Standard format a variable prefixed with a colon is passed as a variable. When Format is dynamic colon prefixed variables are replaced in the SQL statement by their value. Variables of type String are wrapped in double quotes.

Caret (^) - Available for Dynamic format queries. Like colon prefixed bind variables, they are replaced by their value except string datatype variables are not wrapped.

Question Mark (?) - Available for Dynamic format queries. Variable names are stripped from the ? prefix and passed as parameters to OdbcExecuteDynam. The question marks are left in the SQL text as place holders for the ODBC driver. This style of binding is frequently used for Date-Time stamps as it supports dates out to milliseconds and doesn't require special formatting.

Example SQL statement using all three styles of bind variables:

```
SELECT Orders.ShippedDate, Order_Details.Quantity,
Products.ProductName, Order_Details.ProductID
FROM Orders Orders, `Order Details` Order_Details, Products
Products
WHERE Order_Details.OrderID = Orders.OrderID
AND Orders.ShippedDate > ?System.ShipDate
AND Products.ProductID IN ( ^System.ID )
AND Order_Details.ProductID = Products.ProductID
AND Order_Details.Quantity < :System.Quantity
```

Processing SQL statements on non-WIN32 platforms

When Transall is executing on UNIX, processing SQL is supported through the ISIDBLIB library of database interfaces. This library supports access to several popular DBMS from UNIX such as ORACLE and Sybase. This is the same library that Documaker uses to access EDLs and other resources when the publishing engine is running on UNIX.

Details for Accessing Databases from UNIX via SQL

Tranexe for UNIX allows you to interact with databases using a set of database access libraries written by Oracle. These libraries, referred to as ISIDBLIB, allow Oracle applications to access a range of databases using a common set of code. Using ISIDBLIB, Transall allows you to build an application on the PC, using ODBC to define databases, and to then run the application unchanged on UNIX.

Process Overview

To use Transall to access databases on an the UNIX platform carry out the following steps:

1. Create data source name (DSN) files on the UNIX machine which will contain information telling the Transall executable which database to use, which server to connect to, and the user id and password required for connecting to that server. Once defined, these files allow the user to refer to the database connection by using a single name.

```
;;  DSN -- Database Source Name file
;;

================================================================
==  This file defines connection information for connecting   ==
==  to the DB2_5 DB2 5.2 database using the ISIDBLIB DB2      ==
==  Version 5.0 driver, logging database activity to the      ==
==  file db2.log.                                             ==
================================================================

;;
;; Database driver information
;;
DRIVER=DB2/VERSION:5.0/TRACE:db2.log

;;
;;  Database name
;;
DB=DB2_5

;;
;;  User ID
;;
UID=db2user

;;
;;  Password
;;
PWD=db2pwd
```

Example DSN for DB2 Access

```
/==================================
/  DSN -- Database Source Name file
/==================================

================================================================
==     This file defines connection information for connecting  ==
==  to the ORACLE_7_1 database using the ISIDBLIB Oracle      ==
==  Version 2.3.2 driver, logging database activity to the    ==
==  file db.log.                                              ==
================================================================
;;
;; Database driver information
;;
DRIVER=Oracle/VERSION:2.3.2/TRACE:db.log

;;
;;  Database name
;;
DB=ORACLE_7_1

;;
;;  User ID
;;
UID=userid

;;
;;  Password
;;
PWD=password
```

Example DSN File for ORACLE Access

2. Configure the database connection on the PC. The ODBC data source name (DSN) used on the PC for identifying the database should be the same as the name used on UNIX.
3. Build the Transall executable on the PC.
4. Transfer the Transall executable from the PC to UNIX. This may be done in a variety of means, including direct file copy (for example, via diskette, FTP, NFS, or Samba). Whichever method you use, it is important to make certain that you are using a binary file copy.
5. The Transall executable (.TEX file) may then be run in the standard fashion on UNIX.

```
;;
;;  DSN -- Database Source Name file
;;
================================================================
==  This file defines connection information for connecting   ==
==  to the SYB_11 Sybase 11.1 database using ISIDBLIB the     ==
==  Sybase 11.1.1 driver, logging database activity to the    ==
==  file sybase.log.                                          ==
================================================================

;;
;; Database driver information
;;
DRIVER=SYBASE/VERSION:11.1.1/TRACE:sybase.log

;;
;;  Server name
;;
SRVR=SYBASE_11

;;
;;  Database name
;;
DB=SYB_11

;;
;;  User ID
;;
UID=testid

;;
;;  Password
;;
PWD=testid
```

Example DSN file for SYBASE Access

Creating DSN Files for UNIX

Before Tranexe may access a database, the Transall user needs to set up a configuration file that contains information necessary for Tranexe to locate and login to the database server. This file is often referred to as a data source name or DSN file. The DSN file is a text file consisting of a series of tags and values in a format similar to an INI file. The file should be given a name of *source_name*.DSN, where *source_name* is the name that will be used to refer to the database from within Transall. The file itself may contain blank lines, comment lines, and lines defining the tags required by the driver. Comment lines may begin with any character which is not alphanumeric. Common examples might include a semicolon (;), equals sign (=), or a forward slash (/). All text following the comment character on that line is ignored. The tags themselves are case sensitive, and must be entered in upper case.

Note that the **DRIVER=** tag contains sub-tags which are used in defining the driver connection. The sub-tags are separated by a forward slash (/). All other tags consist only of the tag itself and a single value.

Tranexe recognizes the following tags:

Tag	Meaning
DRIVER=	This tag identifies the database driver to use. It takes the form **DRIVER=***driver*/**VERSION:***version[*/**TRACE:***log_file]*. Where *driver* is the driver corresponding to the database which you wish to connect. Currently supported drivers are **DB2**, **SYBASE**, and **ORACLE**. *Version* is the version number of the supported ISIDBLIB driver. At the time of this printing, the current versions supported are DB2: 5.0, SYBASE: 11.1.1, and ORACLE: 2.3.2. For current version levels, check ISIDBLIB documentation. If you need a database log file for debugging purposes, append the optional string /**TRACE:***log_file* to the end of the tag, where *log_file* is the name of the log file. This should only be done if debugging is required, as creating a log file slows down program execution drastically.
SRVR=	This tag identifies the name of the database server. This tag is currently required only for SYBASE drivers.
DB=	This tag identifies the name of the database to access on the server.
UID=	This tag identifies the user name to use when connecting to the database server.
PWD=	This tag identifies the password to use when connecting to the server.

Sample DSN files for each database are illustrated in the examples below:

The DSN file must be in the directory from which the Transall executable will be run. If it is not there, the Tranexe will not be able to connect to the database. If more than one user will be running Tranexe, the administrator may wish to set up a centrally managed DSN file (or files) and have individual users place a link to the configuration file in the directory from which their Transall executable will be run.

Linux ODBC Support

Many Linux distributions now include the unixODBC package which provides support for a variety of ODBC drivers. Transall now allows users to use ODBC drivers on Linux, in conjunction with this package, to access databases. Both the unixODBC package and the specific ODBC driver needed to access the database must be properly installed and configured before you may use this feature. For details on acquiring and installing unixODBC and the drivers it supports, go to their home page at http://www.unixodbc.org.

Once you have installed an ODBC driver, it is used in the same fashion as a native database driver on Linux:

1. Create data source name (DSN) files on the UNIX machine which will contain information telling the Transall executable which database to use, which server to connect to, and the user id and password required for connecting to that server. Once defined, these files allow the user to refer to the database connection by using a single name.
2. Configure the database connection on the PC. The ODBC data source name (DSN) used on the PC for identifying the database should be the same as the name used on UNIX.
3. Build the Transall executable on the PC.
4. Transfer the Transall executable from the PC to UNIX. This may be done in a variety of means, including direct file copy (for example, via diskette, FTP, NFS, or Samba). Whichever method you use, it is important to make certain that you are using a binary file copy.
5. The Transall executable (.TEX file) may then be run in the standard fashion on UNIX.

The DSN file will need to use ODBC as the driver name. The value listed on the DB= field should be the name of the unixODBC DSN that you wish to connect to. The available unixODBC DSN values may be found by running a unixODBC utility, such as ODBCConfigure, or by examining the odbc.ini file on your system. The UID and PWD fields specify the user name and password respectively that are used to connect to the identified database. A sample ODBC .DSN file is shown below:

```
;;
;;  DB Connect info for ODBC -Oracle
;;
DRIVER=ODBC/VERSION:1.0/TRACE:odbc.log
DB=ORASRV1
SRVR=

;;
;;  User ID
;;
UID=user1

;;
;;  Password
;;
PWD=mypasswd
```

In normal instances, you should omit the "/TRACE:odbc.log" clause. Tracing should only be done as part of debugging a database problem, as it slows performance drastically.

Creating DSN Files for the PC

To access a database from the PC, it is necessary to configure an ODBC DSN. By using the same name for both the PC and UNIX data sources, it is possible to create applications on the PC, which will run unchanged on UNIX. When setting up the DSN, one of two methods may be used:

1. Use an ODBC driver to connect to the same database that will be used on UNIX, or a mirror of the database. Since the same database is being used on each platform, identical tables, data types and naming conventions will be used. This helps to ensure maximum compatibility across platforms.
2. Using a different ODBC driver and database, define copies of the database tables and data that you wish to access from UNIX, and use this DSN when building the Transall executable. Because of differences in naming conventions and data types that are supported by different databases, it is possible that varying results could be obtained. If connecting to the database to be used on UNIX is not an option, however, this may still be used without problem in most cases.

For specifics on the details of configuring the ODBC DSN for Windows, see documentation supplied by your database vendor and Microsoft.

Building the Transall Executable

Once database access has been established for the PC, the Transall executable (.TEX file) is built and tested in the standard fashion on the PC, using the ODBC DSN defined earlier. No additional steps are required in the build process for providing UNIX database access.

Transferring the Transall Executable to UNIX

Once the Transall executable (.TEX file) has been built, it must be transferred to the UNIX machine to be executed. There are currently many options available for transferring a file from an UNIX. Some of the more commonly used options include diskette, FTP, NFS, or Samba. While documenting each of these is beyond the scope of this manual, it is worth noting that many of these transfer methods provide methods to ensure that the file is transferred as binary data. Where available, these options should be used; the data will become corrupted if a text-based copy takes place.

DETAILS FOR ACCESSING DATABASES VIA JDBC

Transall now allows users with an existing JDBC environment to access the database using the existing JDBC drivers. To make use of this feature, you must already have an installed and working JDBC environment on the platform where your Transall project will be run. In addition, you must have a working Java runtime environment set up on the machine where the Transall project is to be run. Finally, you must set three environment variables which tell your Transall project how to properly connect to the Java environment:

- **TranJavaPath**—This variable should contain the full pathname of the tranjava.jar file which was installed with Transall.
- **JavaBaseDir**—This variable should contain the base directory where the Java runtime is installed (e.g., /usr/java/jre1.5.0_06).
- **JavaLibDir**—This variable should contain the .jar files needed by the Java runtime environment. Often, it is the "lib" subdirectory of the directory specified by JavaBaseDir.

JDBC support has not yet been added to the editor. To build a project for JDBC, you must build your Transall project to access the database in the standard fashion (i.e., using ODBC). Once the project is finished, change the "DataSource" entry in the Component Inspector bar for each Source or Destination that is to access a database using JDBC. The "DataSource" field should be typed as follows:

```
JDBC;JSN=JDBC_Source_Name
```

where *JDBC_Source_Name* refers to a file that describes the JDBC connection. When the Transall project is run, Transall will look in the current directory for a file with the name given and an extension of .JSN.

If the “DataSource” entry contained “JDBC;JSN=*GT_JDB*” when the program was running and attempted to connect to the database, it would open the file *GT_JDB.JSN* to find information about how to connect to the database server (see the following example).

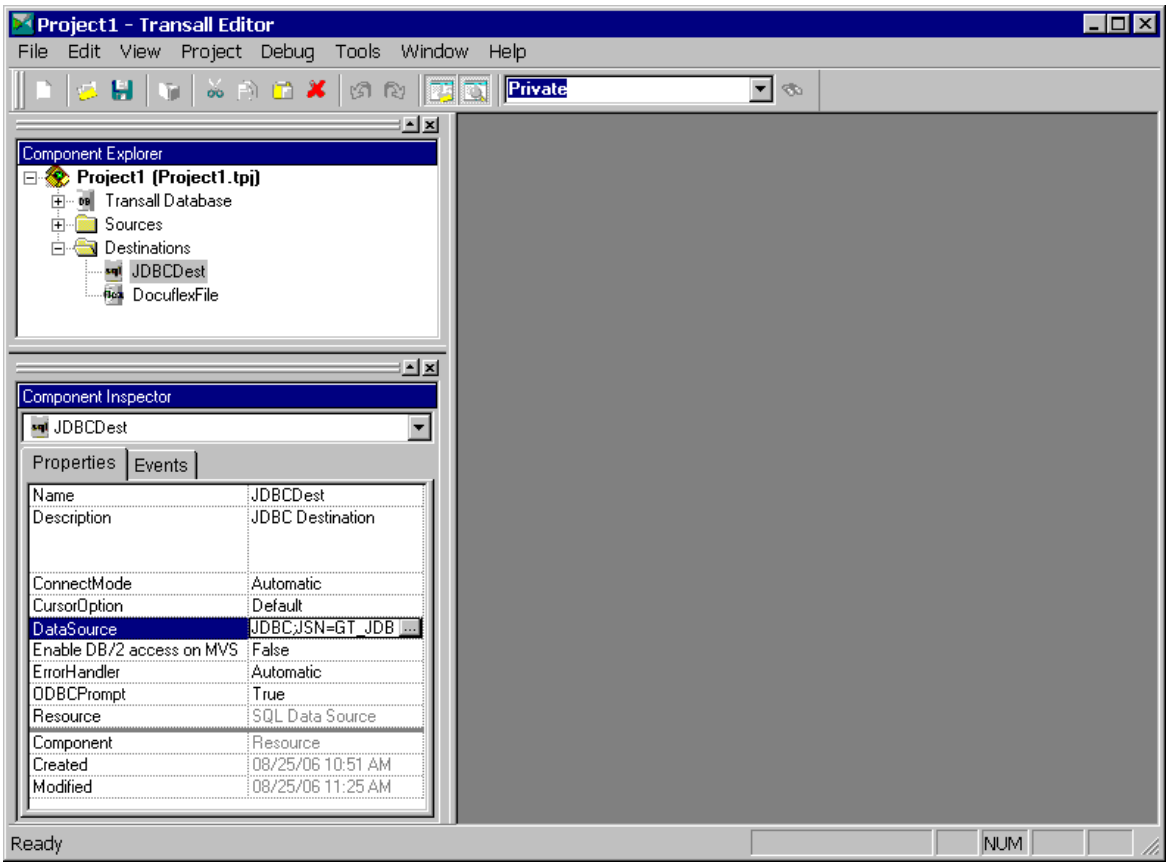


Figure 103: Sample DataSource

This approach is similar to how ODBC connections are handled on Unix platforms.

The JSN file must contain the following items:

- **DRIVER=jdbc_driver/VERSION:1.0[/TRACE:trace_filename]**
- **DB=database_location**
- **UID=user_name**
- **PWD=password**

Lines in the file which don’t begin with one of these values are treated as comments and ignored. It is recommended that the “/TRACE:filename” clause only be used when requested by Oracle Support in attempting to track down a problem, as tracing will drastically slow the performance of a running program. The values to supply for “jdbc_driver” and “database_location” will be specific to both the JDBC driver being used and the installation of the server.

Here is a sample Java data Source file for connecting to a DB2 server via JDBC:

```
;;
;; JSN - Java database Source Name file
;;

DRIVER=com.ibm.db2.jcc.DB2Driver/VERSION:1.0/TRACE:jdbc.log

DB=jdbc:db2://dal1db202:50000/dal1db202

;;
;; User ID
;;
UID=DB2Test

;;
;; Password
;;
PWD=password
```

For users who prefer not to have database user and password values in a file, Transall also allows the JSN to be specified via an environment variable at runtime, as with ODBC. To do this, specify the JSN as follows when building your project:

```
JDBC;JSN=$environment_variable_name
```

where *environment_variable_name* is the name of the environment variable that will contain the proper connection information at runtime. When the program is running and attempts to connect to the database, it will look in the named environment variable to find the database connection information. The environment variable must contain the same information that is required in the .JSN file. Each item in the environment variable should be separated by a semi-colon. For example, if the program was reading the database connection information from the environment variable *JDBC_INFO* to connect to the database from the example above, the variable *JDBC_INFO* should contain:

```
"DRIVER=com.ibm.db2.jcc.DB2Driver/VERSION:1.0/TRACE:jdbc.log;
DB=jdbc:db2://dal1db202:50000/dal1db202;
UID=DB2Test;PWD=password"
```

Mapping of ODBC Data Source Column Data types to Transall Record Field Data Types

When you create a Source or Destination that is based upon an ODBC data source, the Transall Editor automatically defines fields in the Source's or Destination's Record whose data type is based upon the data type of the selected ODBC data source's selected columns.

In this case, the Transall Editor performs the data type mapping summarized in the following table.

Table 1: Mapping of Data Types: ODBC Data Source to Transall

ODBC Data Source Datatype	Transall Datatype
BIGINT BINARY CHAR LONGVARBINARY LONGVARCHAR VARCHAR	String
INTEGER	Long
DATE TIME TIMESTAMP	Datetime
DECIMAL DOUBLE FLOAT NUMERIC REAL	Double
BIT TINYINT SMALLINT	Integer
Other	String

Docuflex Destination

OVERVIEW

Docuflex file is a data destination in Transall for feeding data to Oracle's Docuflex data publishing software. This destination consists of a set of Transall records arranged in a tree hierarchy where each record on the tree is called a "Node". The idea behind the Docuflex file destination is to arrange the data required for publishing a document via Docuflex into a hierarchical structure that makes it easy to drive the document publishing in Docuflex. Please see the Docuflex product documentation for a discussion of its extensive features.

SETTING UP A DOCUFLEX FILE DESTINATION

Select the **Project>Add Destination** menu item. From the Add Destination dialog, select Docuflex File. The Docuflex Assistant opens:

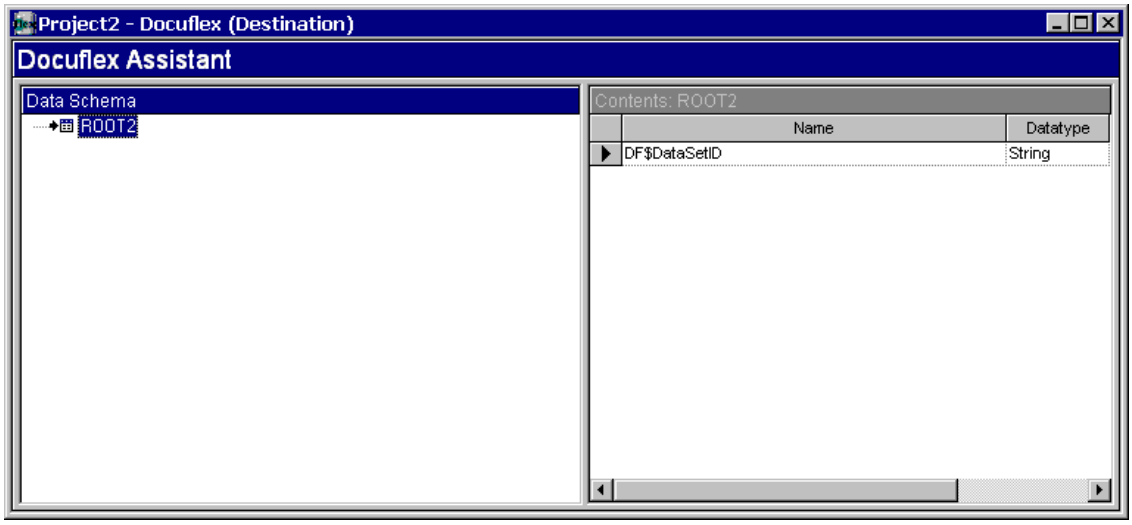


Figure 104: Docuflex Assistant

All Docuflex file destinations have a single main record node named ROOT#. Under this ROOT record node, you add record nodes in a hierarchy that represent the data needed to publish a document via Docuflex. To add a record node, click on the ROOT record node and then select the **Resource>Node Add** menu item (you can also right mouse button in the Data Schema view). This will add a new record node under the ROOT node.

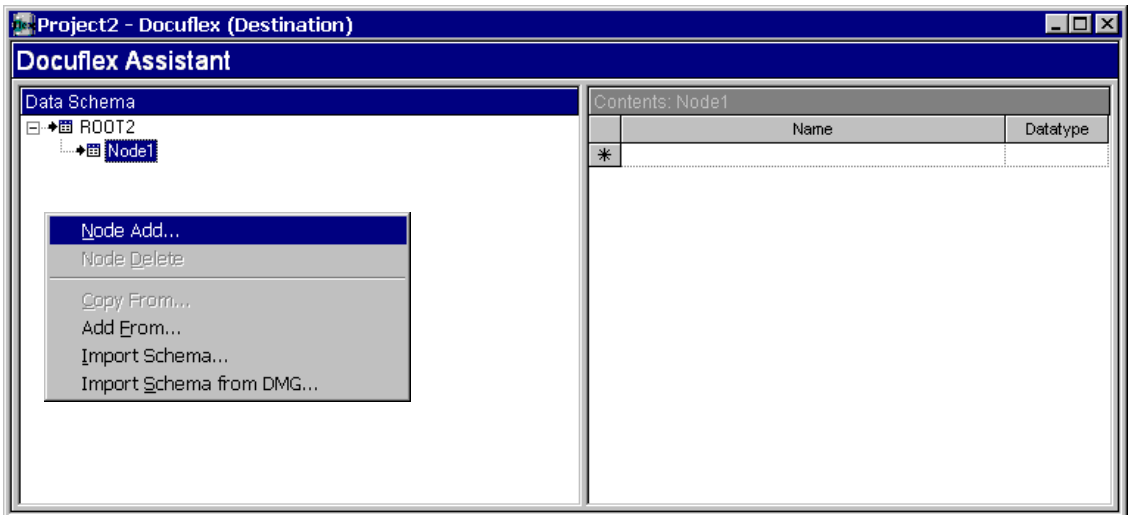


Figure 105: Adding a New Record under ROOT Record Node

To change the record node's name, update the name property in the Component Inspector. To add fields to the record node, enter the field name into the Contents view of the Docuflex Assistant. To add another record node to the hierarchy, click on the node that should be the parent to the new record node (this is the node that the new record node should be under) and select the **Resource>Node Add** menu item. Repeat this process to build out your data hierarchy for Docuflex. This data hierarchy is called a "Data Schema", it represents the data structure, types, and name that will be sent to Docuflex in a Docuflex input file.

Figure 106 on page 191, is an example of a complete Docuflex file destination:

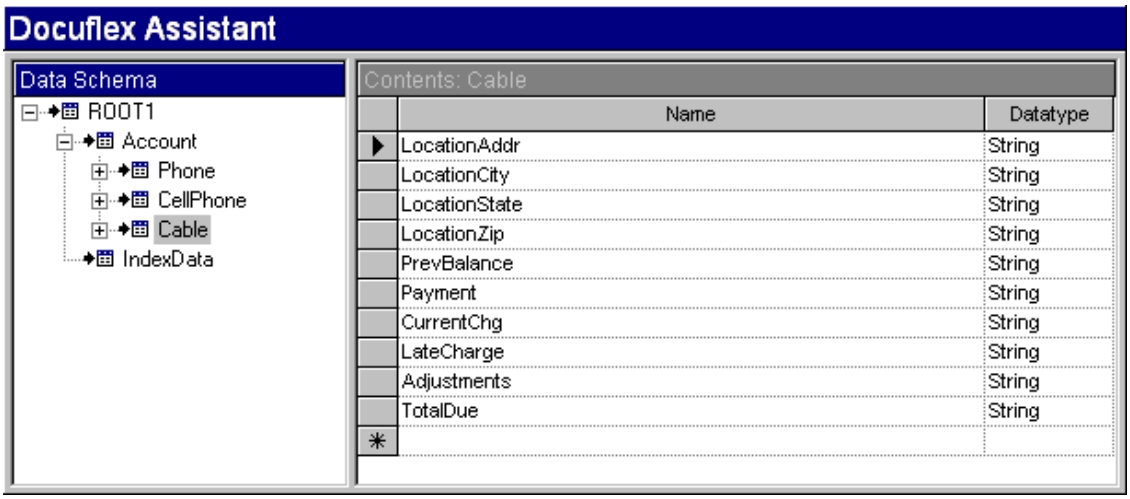


Figure 106: Docuflex Data Destination

If you already have a Docuflex Schema file that you want to use, you can select **Resource>Import Schema** and choose a Docuflex schema (**.DDS*) file. If you've integrated Transall with Documanage, select **Resource>Import Schema from DMG** and select a schema file from a Documanage Cabinet and Folder.

How Data Moves Through a Docuflex File Destination

Data moves through a Docuflex file destination in chunks called "DataSets". Each DataSet represents the data needed to publish one document (like a Contract or a Policy) via the Docuflex Data Publishing product. After all the data for a single document has been placed into a Docuflex file destination, the data is written to a Docuflex input file, all at once, via a LogicTree Output instruction. The Output instruction does two things. It writes a DataSet from the cached up data in the Docuflex file destination and it clears the Docuflex file destination's memory cache so it can be populated with another chunk or DataSet of data.

Because the Docuflex file destination is hierarchical, it is mapped slightly differently than other traditional data destinations such as delimited files. Maps that target Docuflex file destination record nodes have an extra property, "Auto Insert". When this property is set to Yes, Transall inserts a new row into the record node hierarchy in memory before mapping over the record fields. This is important because it enables Transall Maps to add new records to the hierarchy as they load fields into the records.

The logic that goes into populating a Docuflex file destination with data usually consists of several Map instructions at build out the data hierarchy from top to bottom.

Using *Figure 106* on page 191, as an example, a Policy record must be mapped before an Auto record can be mapped) and then an Output instruction that writes the data hierarchy DataSet to the Docuflex input file. Please see the documentation for Docuflex on input files for more information about how Docuflex processes data.

DEFINING A DATASET IDENTIFIER

The root record in a Docuflex destination has a field named DF$DataSetID that is used by the Docuflex publishing software to provide an identifier in messages during publishing. Use either a Map or an Execute instruction in the Logic instruction in the LogicTree to specify a meaningful value, such as account number, for the DF$DataSetID.

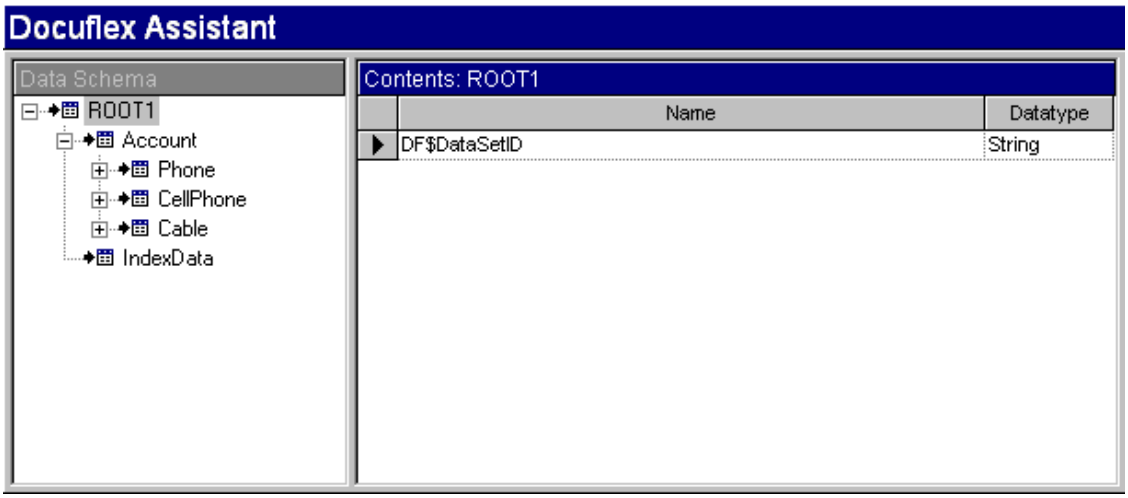


Figure 107: DataSet Identifier

USING A DOCUFLEX DESTINATION AS A DATA SOURCE

You can use Transall Docuflex Destinations as a Data Source to

- read the contents of a Docuflex Data File (DDF)
- extract the data contents from a Stacked Document Compound Document (DCD) file
- extract the data contents from an archived Docuflex DCD

To Use a Docuflex Destination as a Data Source

- Set the **AsSource** property of the DDF Destination to **True**:

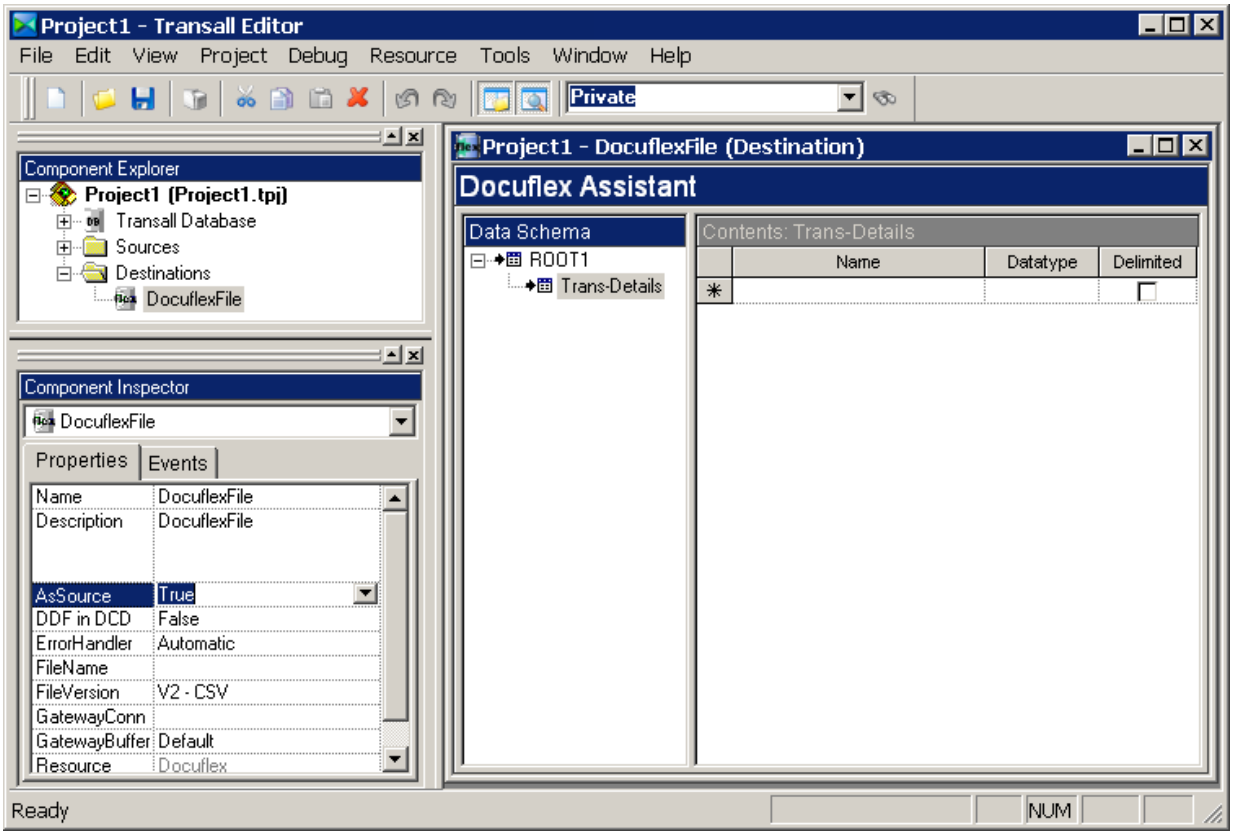


Figure 108: Set AsSource to True

If the file that will be read is not a DDF, but a DCD or stacked DCD, then also set the **DDF in DCD** property to **True**.

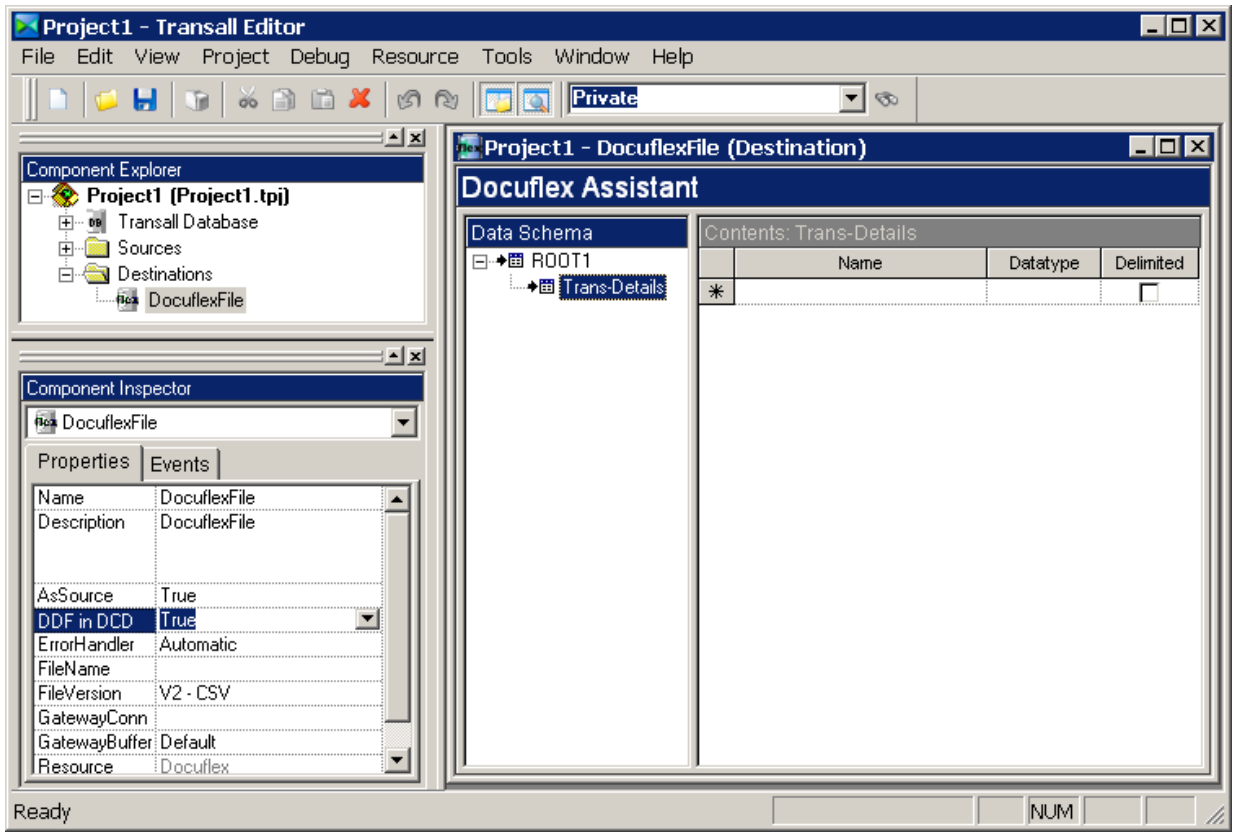


Figure 109: Set DDF in DCD to True

These settings make the Docuflex Destination definition available as a Data Source. The Docuflex Source acts like an XMLPlus Source. To use the Source, add a Walk on the Destination name to a LogicTree.

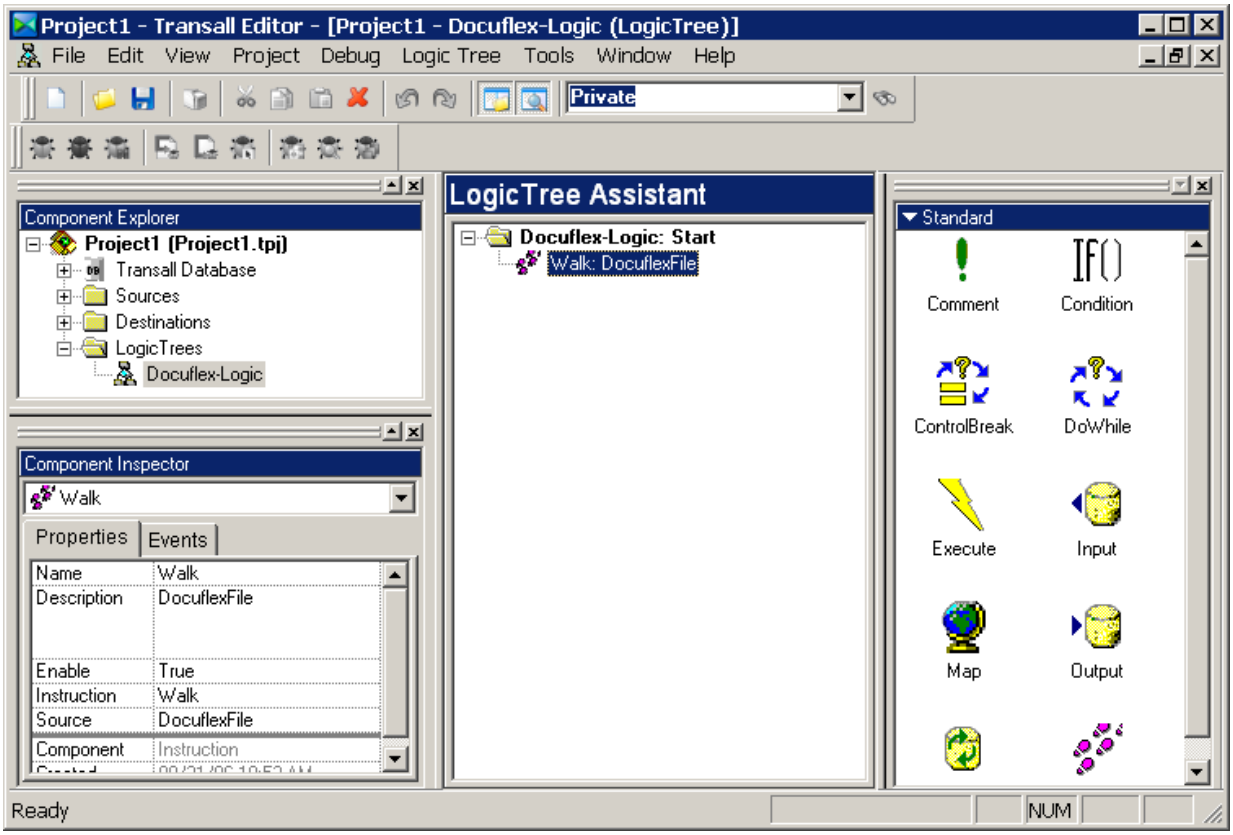


Figure 110: Add Walk on the Destination Name to a LogicTree

For each execution of the Walk instruction, Transall will load the next transaction from a DDF file, DCD, or Stacked DCD into the records defined for the Docuflex Destination. These records can then be traversed to process the data.

FP Plus Destination

Overview

Forms Publication Plus (FpPlus) is a data destination in Transall for feeding Documaker Forms Publishing (FP) software. Transall actually provides two kinds of FP data destinations. One is called “Documaker FP File” and the other is called “Documaker FP Plus”.

The Documaker FP File data destination is provided for use by Transall developers that are familiar with the Documaker Variable Data Reformatter (VDR) Application Program Interface (API) and wish to create a VRF via Transall by building logic that directly calls the detailed VDR API.

The Documaker FP Plus data destination is provided for use by Transall developers that are not familiar with the Documaker VDR API. The objective of the Documaker FP Plus data destination is to let the Transall developer concentrate on the business logic for selecting forms and tags that are required for various business scenarios vs. thinking about making calls to the Documaker VDR API. This data destination shields the Transall developer from the details of the VDR API while still providing powerful features such as automatic form DOT height counting. DOT height counting enables Transall to provide text flow control, assembling forms in the VRF that act as headers and footers in the output. This DOT counting ability also enables Transall to provide widow/orphan control to prevent page breaks from occurring at unwanted points in the output and other features. These special features, provided by the Documaker FP Plus data destination, would have to be custom coded by the Transall developer when using the Documaker FP File data destination.

FP Plus Built In Features

Headers and Footers - Forms that will appear as the first or the last on a page.

Keep - Attempts to keep a page break from happening between two forms or a block of forms.

Word Wrap - Transall will preprocess a tag’s or a block tag’s data to insert new line characters so that line breaks will occur between words. This feature becomes particularly powerful when combined with Overflow.

Overflow - Enables the tags of a form (the parent) to be mapped to a second form (the child) to handle tag data overflow conditions. When the length of data set to a parent's tag exceeds the Form's Tag length (or depth) the overflowing data will be sent by Transall to tags on the child form. If the overflow continues on the child form the child will be sent as many times as is required to transmit all the overflowing data.

Page Counting - Transall will maintain a page count and a total page count at two levels. The first is at the MergeSet and this page count works very similarly to the DMG.PAGE.COUNT tag provided by Documaker FP. The advantage of Transall's page count is that it can be used in business rules defined to FpPlus. The second page count is at the "section" level. Blocks of output can be identified as a section in FpPlus and Transall will maintain an individual page count and total page count for each section. This enables "Page # of #" type output for a section vs. the whole MergeSet.

How FP Plus Works

FpPlus is little different than other Data Destination types in Transall. Other Data Destinations have records defined to them that represent physical records of a file or rows in a SQL database table. The FpPlus data destination also has records but these do not represent physical records in a file or database. What these records represent is all the data that is needed to create a document (a MergeSet) in a VRF file. You can have many Records in an FpPlus data destination and these records can be arranged in a hierarchy where some of the records "own" child records. The record hierarchy feature of FpPlus can be used when processing data that has records connected by a parent child relationship.

For example, the records in an FpPlus destination for an auto insurance policy would hold all the data needed to map the tags on all the forms that are required to produce the auto policy. There would probably be a POLICY record with fields such as POLICY_NUMBER, PREMIUM, INSURED_NAME, INSURED_ADDRESS, etc. There would also probably be an AUTO record with fields such as VIN, MAKE, MODEL, GARAGE_ADDRESS, etc. The AUTO record may also have a child record DRIVER that holds a list of drivers for the auto with fields like NAME, AGE, etc. In this example, these three records will be inserted into the FpPlus Data Destination by any Data Maps used in your Transall application. The AUTO and DRIVER records should be inserted with care to sequence because Transall automatically connects child records to the last parent record inserted. So, in this example, let's say that the first AUTO has two drivers and the second only one, then the first AUTO would be inserted into the FpPlus destination followed by its two DRIVER records and then the second AUTO would be inserted followed by its one DRIVER.

Once the data has been inserted into the FpPlus data destination by normal Transall LogicTrees and Data Maps, a special Transall LogicTree, that is owned by the FpPlus data destination, is executed to process the data loaded in the destination's records. This special LogicTree is created by Transall for you when you define an FpPlus data destination to your Transall project and it is designed to hold business rules that select the forms required to produce the output you require. It is in this special LogicTree that special Documaker FpPlus LogicTree instructions can be used. These special instructions enable form selection, tag mapping, overflow definition, header and footer selection, etc. So when using an FpPlus data destination you will always have two LogicTrees, one LogicTree that performs data extract and loads data to the FpPlus Data Destination and then a second LogicTree that handles business rules for form selection and processing.

FpPlus Business Logic

As discussed in the preceding section, Transall will create a special LogicTree for you when you define an FpPlus data destination to your Transall project. When you look at this FpPlus Data Destination LogicTree you will see a topmost instruction "FpLayout". The FpLayout instruction tells the Transall compiler that all the instructions under it will be for an FpLayout that defines the business rules for form selection and assembly to build an FP document (a MergeSet) in an FP VRF file. Any Transall instructions can be used under an FpLayout instruction. Some Transall instructions can ONLY be used under an FpLayout. These instructions are: FpForm, FpAddTag, FpKeepOnSamePage, FpHeader, FpFooter, FpComment, FpDataGroup, FpDataHeader, and FpPageBreak.

The following are details on each instruction:

FpLayout - Designates a block of Transall logic that is dedicated to building a document (a MergeSet) in a VRF file. This instruction must be associated to an FpPlus Data Destination. FpLayout instructions can not be nested under other FpLayout instructions. More than one FpLayout instruction can be placed in a single LogicTree. Each FpLayout instruction will cause a new FP document (a MergeSet) to be placed in the resulting VRF file.

FpForm - Adds a FORM to the FP document (MergeSet) that will be written to the VRF file. When an FpForm instruction is added to the FpLayout LogicTree, Transall accesses the "current" Electronic Document Library (EDL) assigned to the FpPlus Data Destination that the FpLayout is associated with. Transall displays a searchable list of FORMs from the EDL that the Transall developer can select from. When a FORM is selected Transall scans the FORM for TAGs. Transall then displays a dialog for mapping data to TAGs defined as delete=yes. For TAGs found on the form that are defined as delete=no (e.g. are global), Transall adds these TAGs as fields to a special GlobalTag record that is defined in the FpPlus Data Destination's record list. This special record enables Transall to add delete=no TAGs only once to an FP document (MergeSet) and add delete=yes TAGs with their FORMs.

FpAddTag - Unconditionally adds a TAG to the FP document (MergeSet).

FpKeepOnSamePage - Attempts to keep a page break from occurring within a block of logic. Arrange instructions under an FpKeepOnSamePage to cause Transall to predict if a page break would occur while the logic is processing. Transall does this by pre-executing this logic at run-time. Before this pre-execution, the "state" of the Transall environment is saved. Then the logic under the FpKeepOnSamePage is executed in a pre-execution mode that does not write FORMs and TAGs to the FP document (MergeSet). After the pre-execution is complete the "state" of the Transall environment is restored. If a page break was hit during the pre-execution, a page break is forced to occur before the FpKeepOnSamePage is executed for real. If no page break occurred then the logic under the FpKeepOnSamePage instruction is processed normally.

FpHeader and FpFooter - Defines a block of instructions that are executed when Transall is writing FORMs as the first or last on a page. Arrange instructions under a FpHeader or FpFooter to define instructions that execute as the first or last for a page. More than one header or footer can be defined to a document. As each FpHeader and FpFooter is defined it replaces the last header or footer defined. To clear a header or footer, define an FpHeader or FpFooter instruction with no instructions nested under it.

FpComment - Places a comment in the FpLayout debugging output. FpPlus will write a debugging file that shows all the forms selected and all the TAGS and their values sent to the VRF file. The FpComment instruction places a comment in this file for debugging and testing purposes. (See the Debug File property of The FpPlus Data Destination to define a debug file.)

FpDataGroup - Defines a looping block of instructions. Arrange instructions under an FpDataGroup to define instructions that execute once for each row in a FpPlus record. This instruction loops on the record's rows starting with the first row and walking through to the last row. The FpDataGroup instruction will set the row "currency" for the record before running the block of instructions nested under the FpDataGroup. More than one FpDataGroup can be nested under each other to traverse parent child records that are several levels deep in the FpPlus Data Destination.

FpDataHeader - Defines a block of instructions that are executed only if a page break occurs in the FpDataGroup that is associated with the FpDataHeader. Arrange instructions under an FpDataHeader to define instructions that execute after page headers and only if a page break occurs while an associated FpDataGroup is processing. This instruction is great for adding "Continued" type headers to repeating data that repeats across more than one page.

FpPageBreak - Forces a page break to be processed by Transall. Note that this does not force a page break to occur in FP Documaker (Documaker). The best way to synchronize Transall's assumed page breaks to Documaker FP's physical page breaks is to define a special page break FORM defined as the last form in the Footer. This FORM should be defined as having two logical pages but only one physical page and a zero DOT height. Contact Oracle support if you need help creating this special page break form. The FpPageBreak instruction can also adjust the default max page DOT height used for pagination from the point of the Page break forward in the document (MergeSet) and reset Transall's section page numbering. If zero is passed for the page height on an FpPageBreak then Transall will make no adjustment to the default page height.

FPPLUS DATA DESTINATION DETAILS

Documaker FP Plus Data Destination Properties

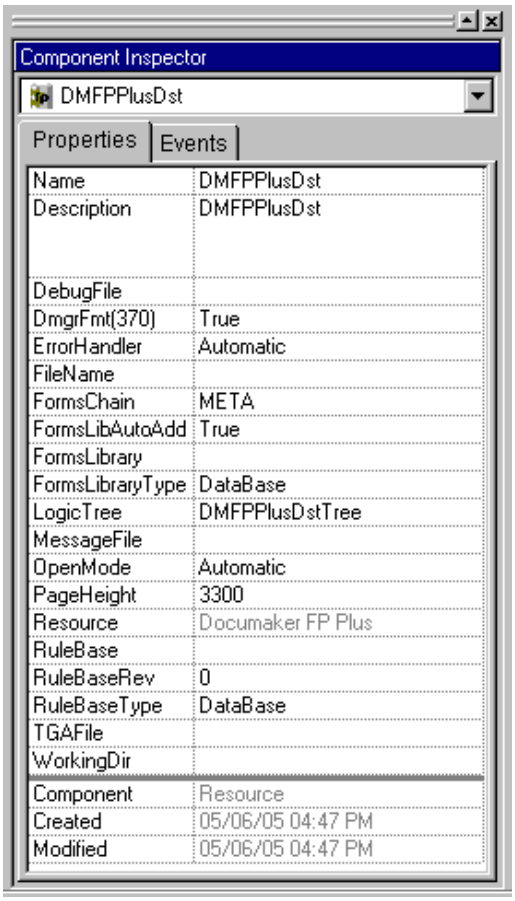


Figure 111: Documaker FP Plus Data Destination Properties

Documaker FP Plus data destination properties are shown in *Figure 111* on page 201.

***Name**:* *Name of the Documaker FP Plus data destination.*

***Description**:* *Comment text about the Documaker FP Plus data destination.*

***DebugFile**:* *Name of a file that will be created only when running Documaker FP Plus under the Transall debugger. This file helps explain Documaker FP Plus's pagination processing for debugging.*

***DmgrFmt(370)**:* *True or False. If set to true then when Transall is running on the 370 platform Transall will call the Documaker FP DMGRFMT and DMGVRFWR VDR APIs to create a VRF. These APIs create a VRF via calls to a VLAM based EDL and RuleBase. If set to false then when Transall is running on the 370 platform, Transall will call the DMKVAE VDR API to create a VRF. This creates a VRF via calls to the new "flat file" based EDL and RuleBase.*

***ErrorHandler**:* *Automatic or Manual. If set to Automatic, Transall will handle all recoverable errors itself and handle all unrecoverable errors by aborting with a message. If set to false Transall ignores all errors (not recommended) and the Transall user must write Transall script code to handle errors.*

***FileName**:* *This is the default name of the VRF file that will be created by this data destination.*

***FormsChain**:* *META, AFP, or DCD. This is the EDL chain type used at development time to "dump forms" for computing form DOT height and locating form tags for mapping in Transall.*

***FormsLibAutoAdd**:* *True or False. When True the EDL or EDLs entered in FormsLibrary are available at run time through VdrAddFormsLibrary calls in the open script.*

***FormsLibrary**:* *This is the Name of EDL used at development time to "dump forms" for computing form DOT height and locating form tags for mapping in Transall. Multiple EDLs may be entered, though only one can be active (open) at a time for any FP destination.*

***FormsLibrayType**:* *Database or File. This is the type of EDL to be used at development time to "dump forms" for computing form DOT height and locating form tags for mapping in Transall.*

LogicTree: *Name of the Documaker FP Plus Logic Tree.*

***MessageFile**:* *Name of a file that will receive messages from the DMKUSER.W32 VDR API or the DMKVAE API on the 370 platform. This parameter is ignored if Transall is running on the 370 platform and the*

	DmgrFmt(370) parameter is set to True. When this is the case messages from Documaker flow through the older DMGRFMT VDR API as documented in the Documaker 3.x Reference Guide.
***OpenMode**:*	*Automatic or Manual. When this is set to Automatic Transall to automatically prepare the Documaker FP Plus destination for receiving data. If this is set to Manual then the Transall user must explicitly execute the open commands for the data destination in the LogicTree.*
***PageHeight**:*	*Defaults to 3300. This is the initial DOT page height used by Documaker FP Plus for paginating forms written to the VRF.*
***Resource**:*	*Display only, this is the type of Transall resource.*
***RuleBase**:*	*Run time rule base, automatically set in the open script if not blank.*
***RuleBaseRev**:*	*Revision level of the named rule base.*
***RuleBaseType**:*	*Database or File. This is the type of the rule base to be used at run-time. If setting is placed to Database then Transall calls MRGUSER.W32 to create a VRF. If this is set to file then Transall calls DMKUSER.W32 to create a VRF.*
***TGAFile**:*	*This is a Tag Commander Export file that can be imported into Documaker FP Plus at development time to predefine the available tag list.*
***WorkingDir**:*	*Name of a directory that will be used to store temporary files (such as dumped form chains) by the DMKUSER.W32 VDR API. This parameter is ignored on the 370 platform. If blank it defaults to the current working directory.*

Documaker FP Plus Data Destination Record Properties

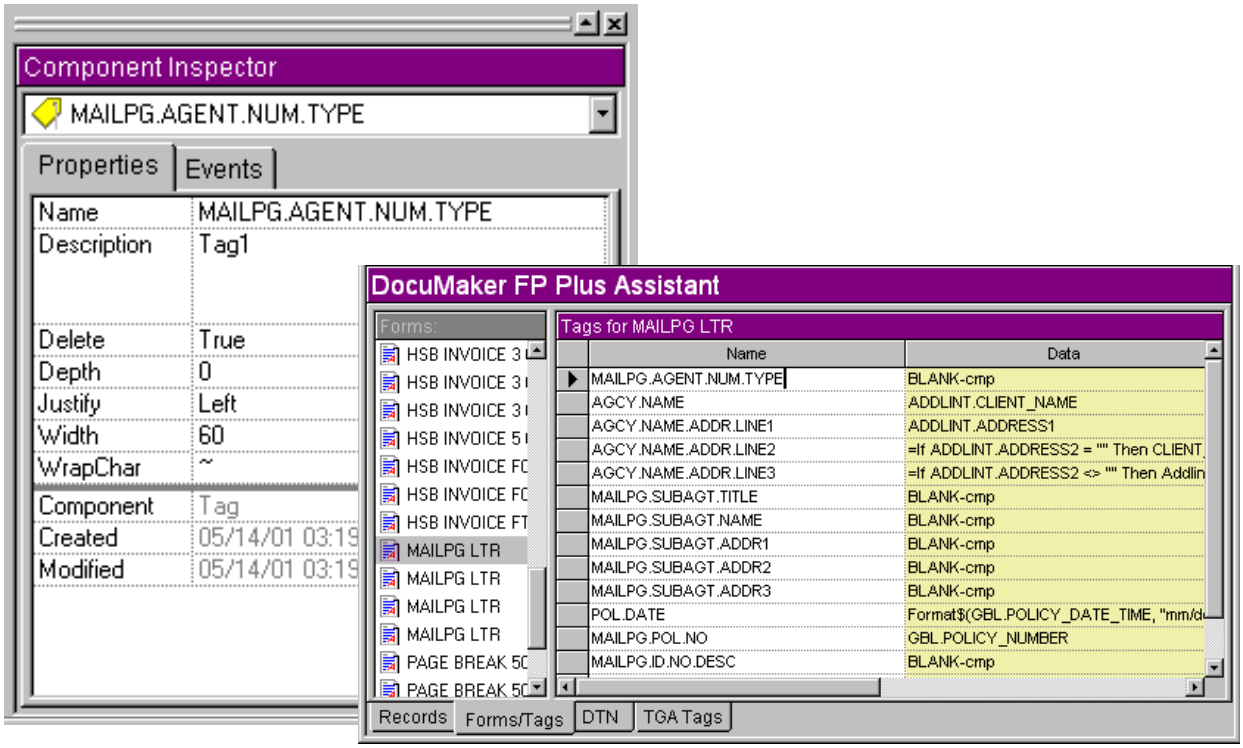


Figure 112: Documaker FP Plus Data Destination record properties

Documaker FP Plus data destination record properties are shown in *Figure 112* on page 204.

***Name**:* *Name of the record in the Documaker FP Plus data destination.*

***Description**:* *Comment text about the Documaker FP Plus record.*

Record properties:

***Name**:* *Name of the record to Transall (example the Transall name of the Policy Number).*

***Tag Name**:* *Name of a VRF tag that will be added to the VRF populated with the value from this record. Note this is an easy way to unconditionally populate tags in the MergeSet. This is most often used with delete=no tags.*

***Datatype**:* *The data type of the field Documaker FP Plus will use to hold this value in memory.*

***Format**:* *This is a formatting mask string that will be applied to the data when the field has a Tag Name defined and is written to the VRF. Note this value is most useful for formatting date fields.*

EXAMPLES

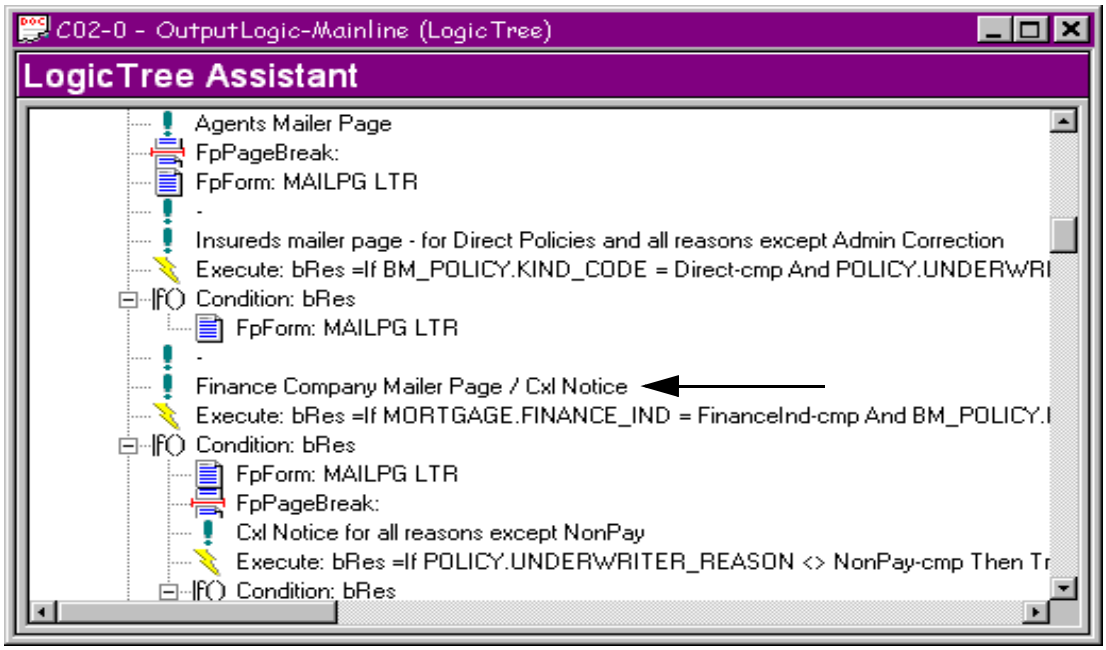


Figure 113: Example of a FpComment

FpComment - Is exactly what it says, a comment it is used to place a note, statement, or explanation about something pertaining to the program.

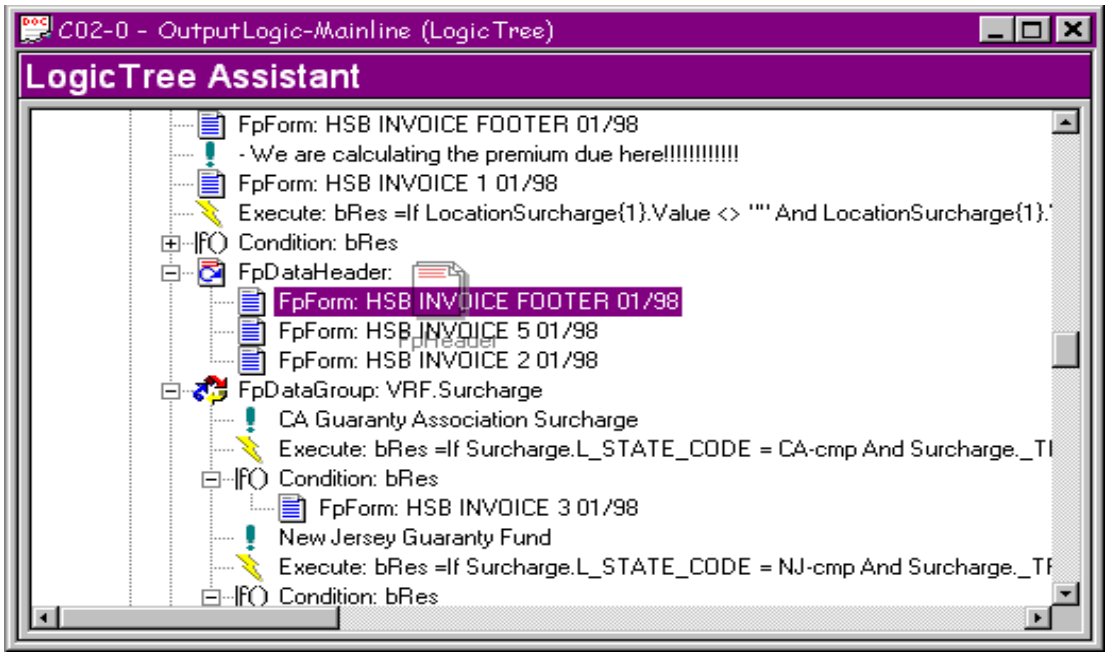


Figure 114: Example of Dropping an Instruction on to the Logic Tree

Note By holding down the SHIFT key, clicking on any of the instructions, and dragging it over the LogicTree Assistant, a blue line displays to show you where the instruction will be placed when you release the mouse button.

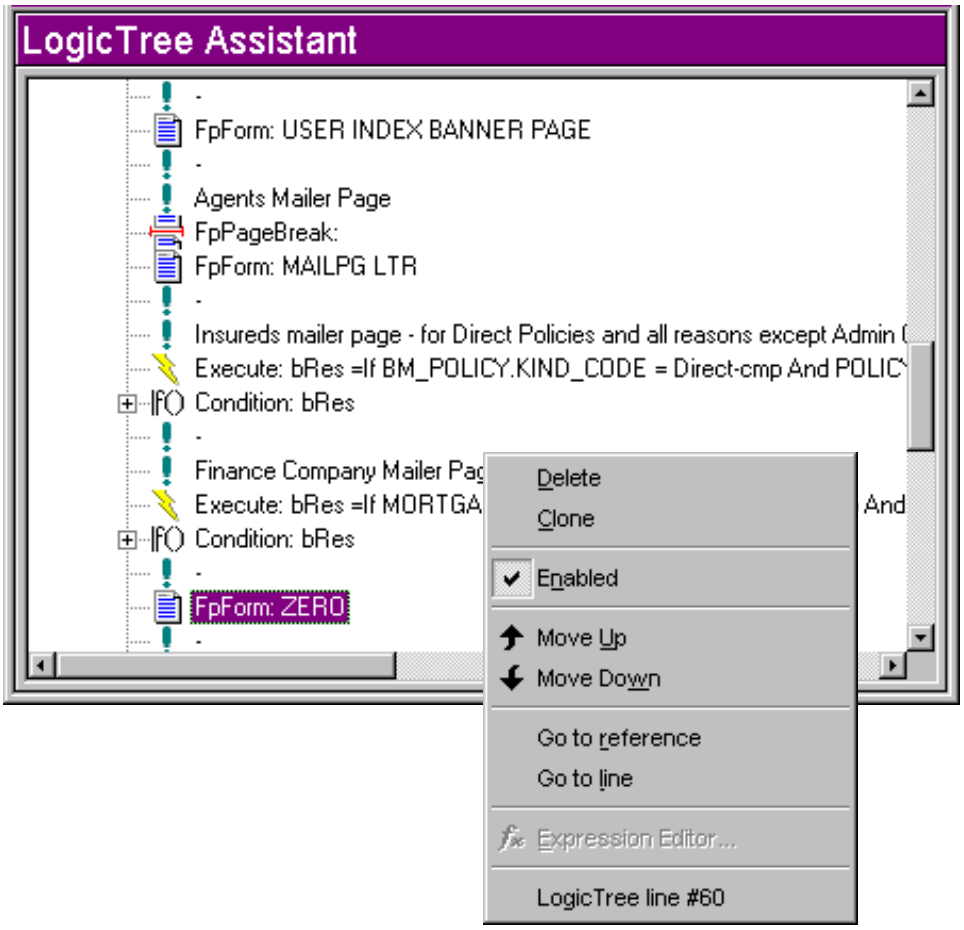


Figure 115: Right Mouse Menu

Note	Selecting an object in the LogicTree Assistant and clicking the right mouse button displays the menu above. This allows you to **Delete**, **Clone**, or even **Move** the selected object **Up** or **Down**. This same menu appears by going to Logic Tree on the main menu bar. You can also Move the object Up by pressing **CTRL+SHIFT+U**; Move the object Down by pressing **CTRL+SHIFT+W**.

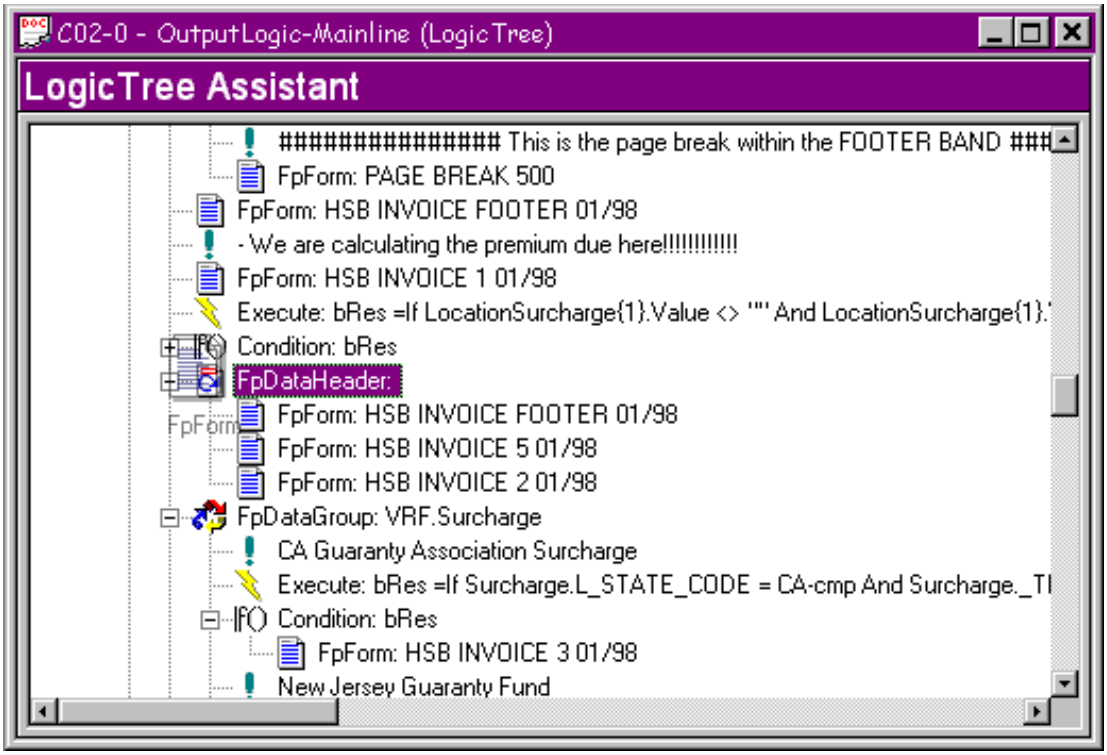


Figure 116: FpForm in LogicTree Assistant

FpForm - Add a form to the forms list.

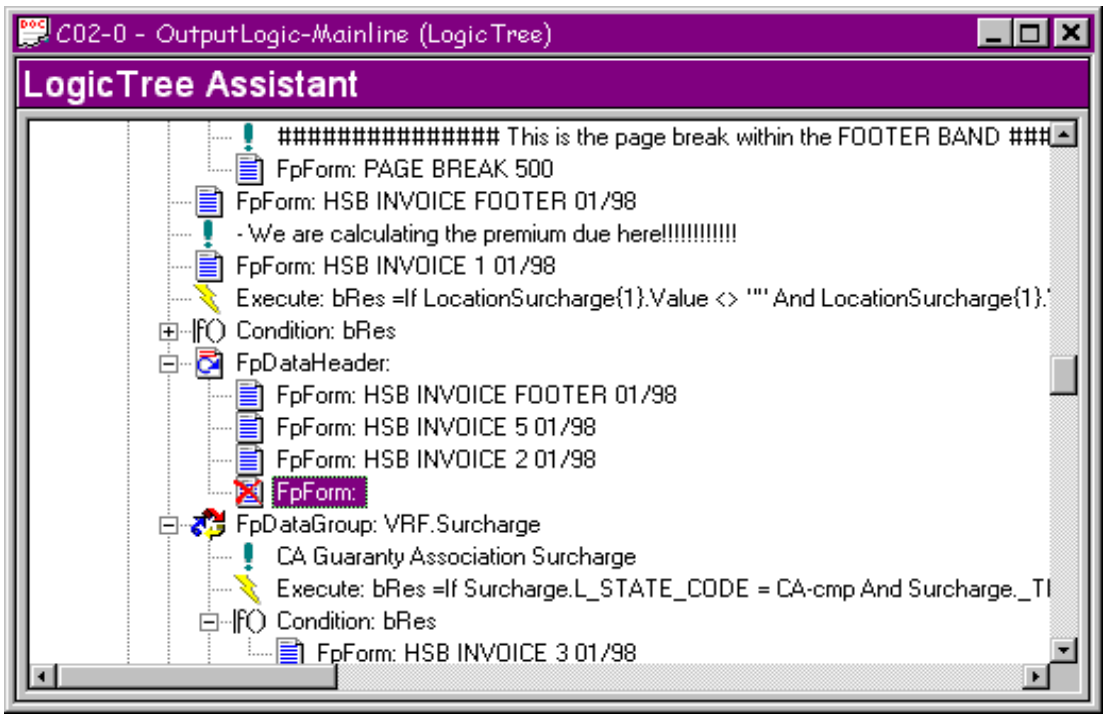


Figure 117: Red X and Red Text for Incomplete Setup

After being placed into the LogicTree Assistant, a red X appears on the left side and the text will appear in red, to show that an incomplete set up of the instruction.

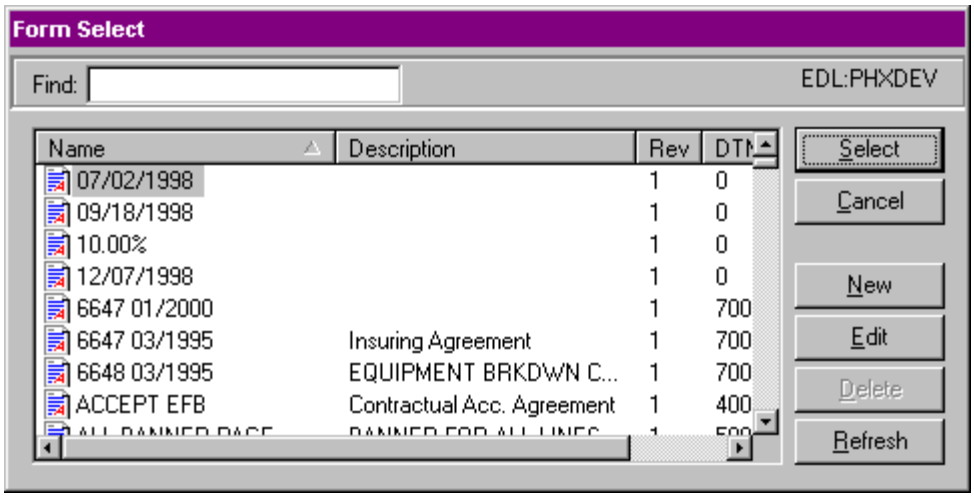


Figure 118: Electronic Document Library (EDL)

Double click on the highlighted FP Plus Form instruction and it will bring up the Electronic Document Library (EDL) that is a list of all the forms available in the project.

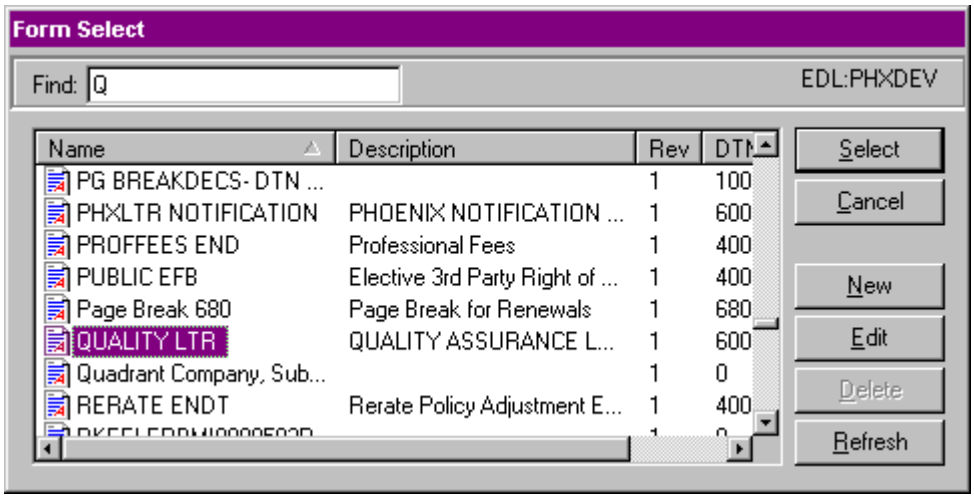


Figure 119: Find Form Using Form Select Dialog Box

In the Form Select dialog box, look up the form that is needed, using the Find feature. When the form has been found, highlight the form and click the Select button, Transall will dump the form to see if there are tags on that form.

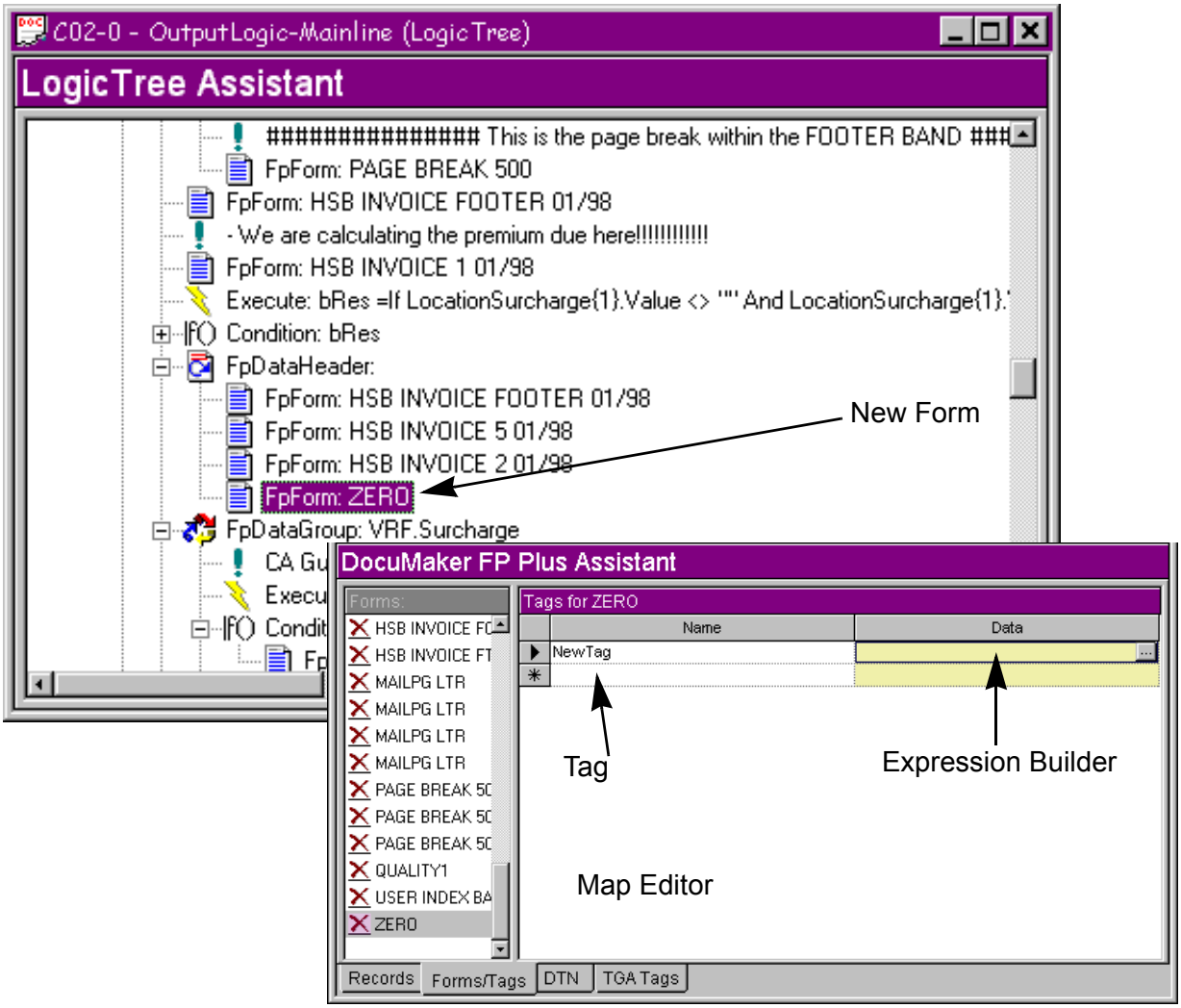


Figure 120: Documaker FP Plus Assistant

If the form contains tags, the Documaker FP Plus Assistant opens, displaying the tags. If the form has no tags, it will just be dumped into the LogicTree Assistant.

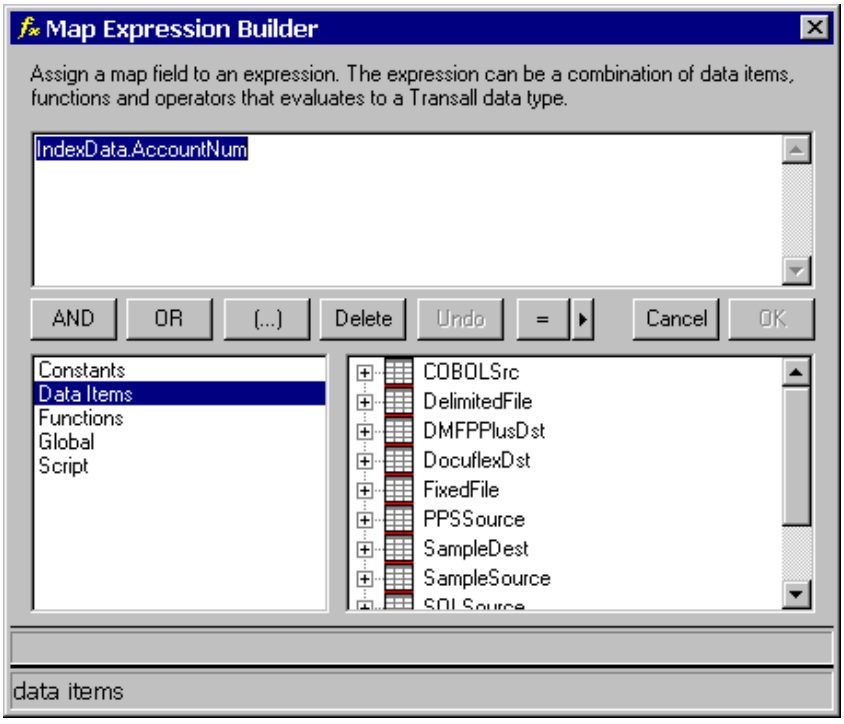


Figure 121: Map Expression Builder

You can add an expression or modify an existing expression by clicking the box with the three dots in the Expression Builder.

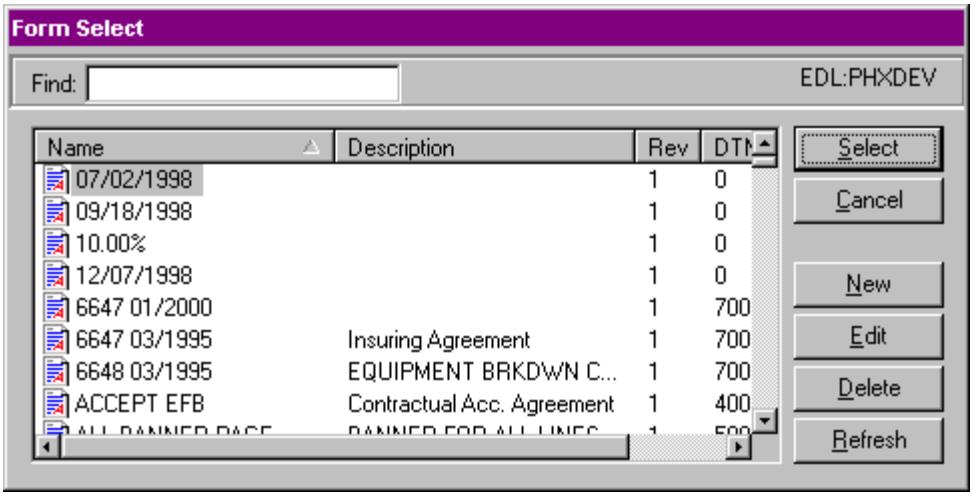


Figure 122: Form Select Dialog Box

If you create a new form placeholder in the Form Select dialog box, click the New button.

You have the ability to add your own user-based forms; there may be a time that a form is needed that is not in EDL. You can create a form, by naming and putting some default information in this form in Transall and what Transall will do is hold the form as an invalid form, as the form as not been validated against the EDL. This will allow you to build all your business logic and test that business logic. Transall does have a feature that can validate a form and the validation process go back over and checks to make sure the tags that were mapped, match the tag names that were put into the EDL.

Note You might create a new form placeholder while the EDL is being updated by your companies form people so you do not have to wait for the update to build business logic in Transall.

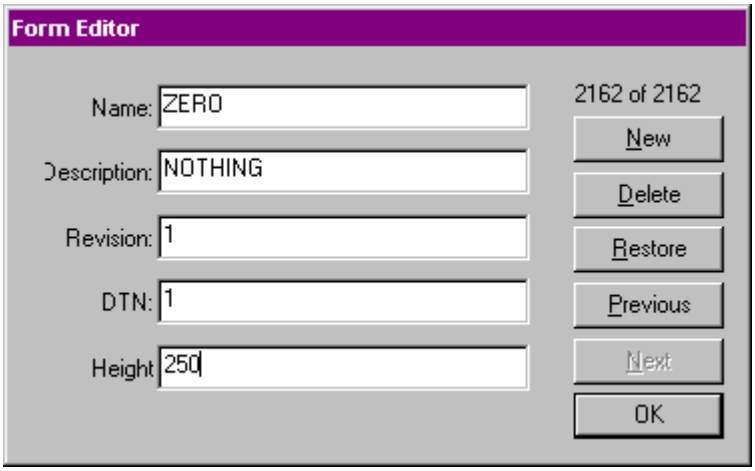


Figure 123: Form Editor

The Form Editor dialog box will open, input information and click OK.

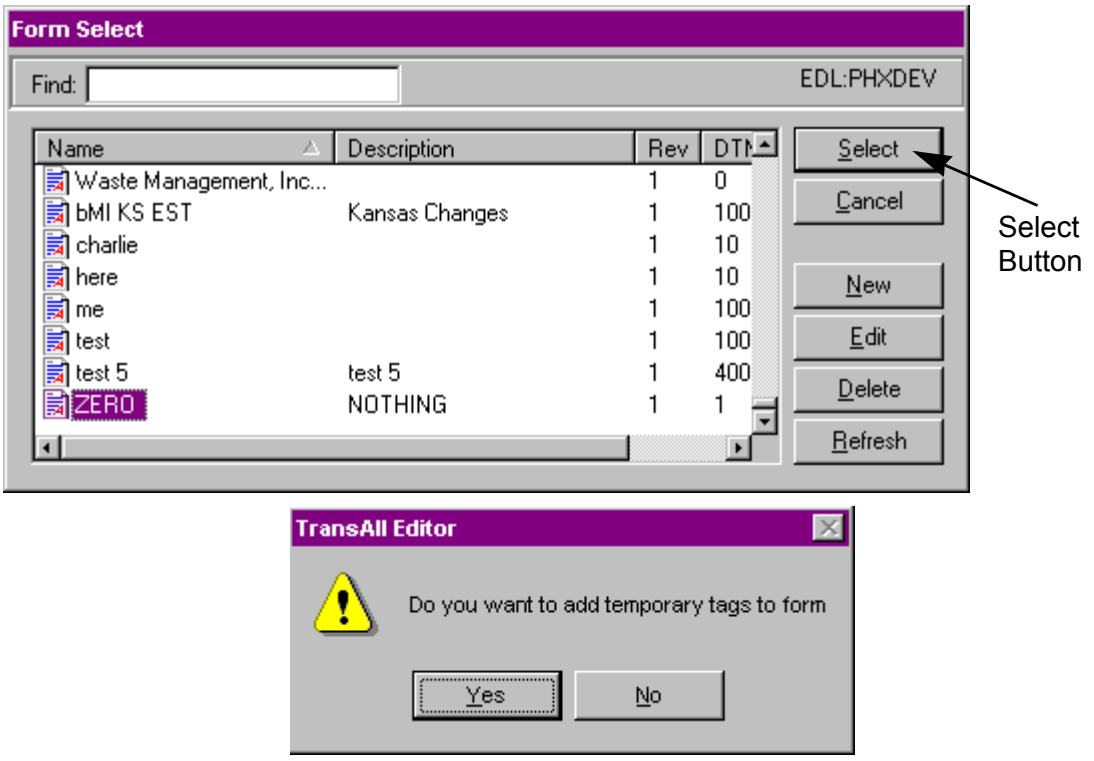


Figure 124: Add Tags to New Form

Highlight the new form and click Select and the Transall Editor will ask if you want to add temporary tags.

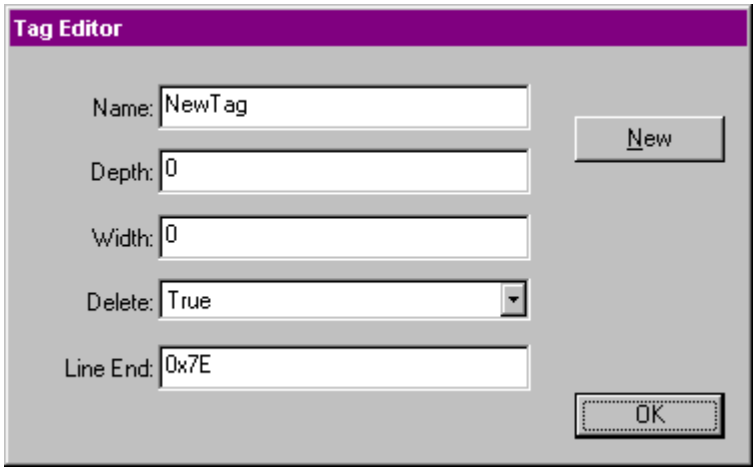


Figure 125: Tag Editor

If YES, to creating a new temporary Tag, the Tag Editor Dialog box will open, for information to be added. Input information, if you want to add another tag click the New button, if no new tags need to be added, click the OK button.

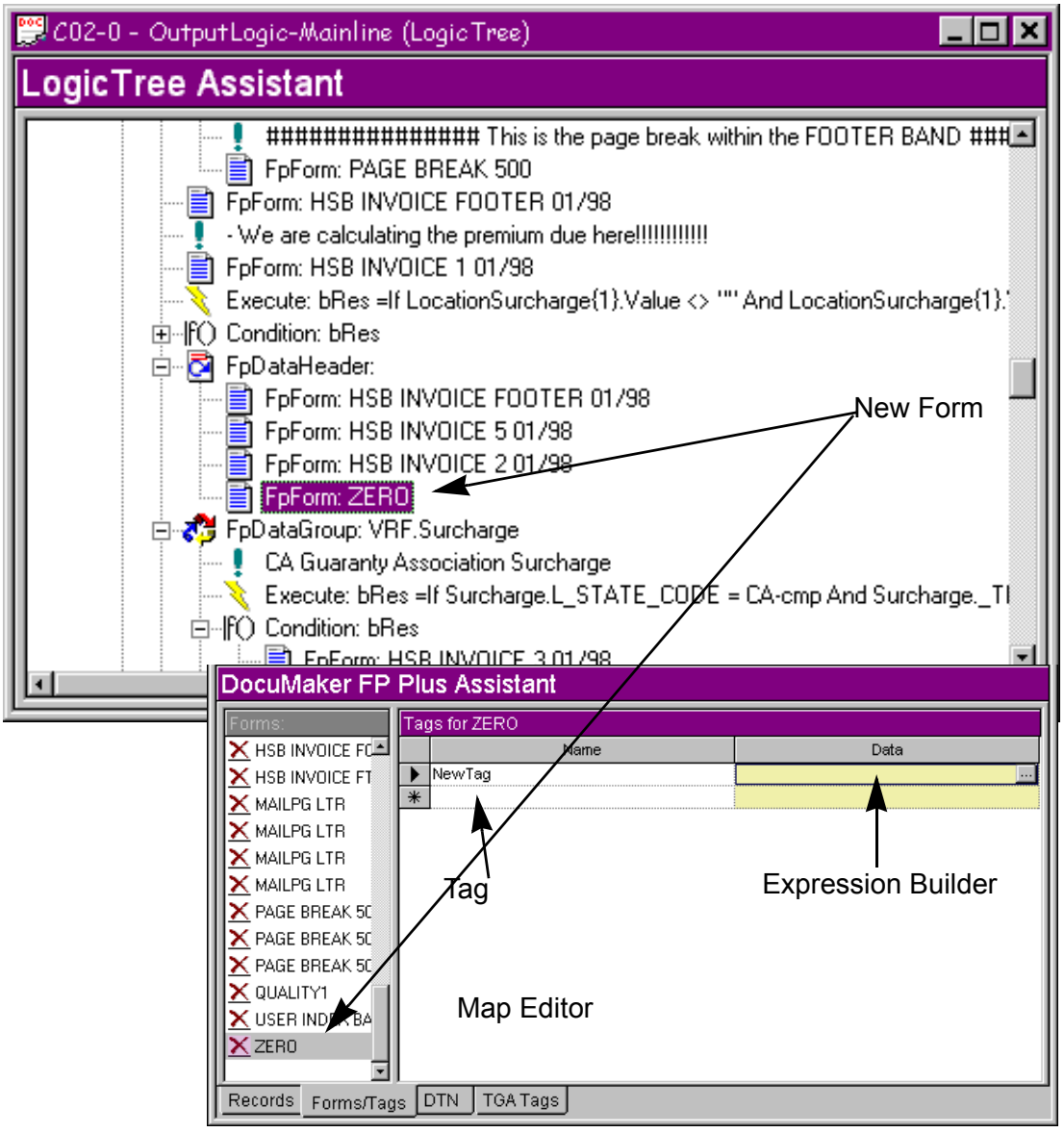


Figure 126: New Form with Tags

This is what a new form placeholder with tags would look like, note that new Form appears on the Forms list of the Documaker FP Plus Assistant and the LogicTree Assistant. In the Documaker FP Plus Assistant, this new Form placeholder will continue to have a red X indicating the form is not in the EDL. When the form is added to the EDL, you can use the form synchronize feature to validate the Form's information to the EDL.

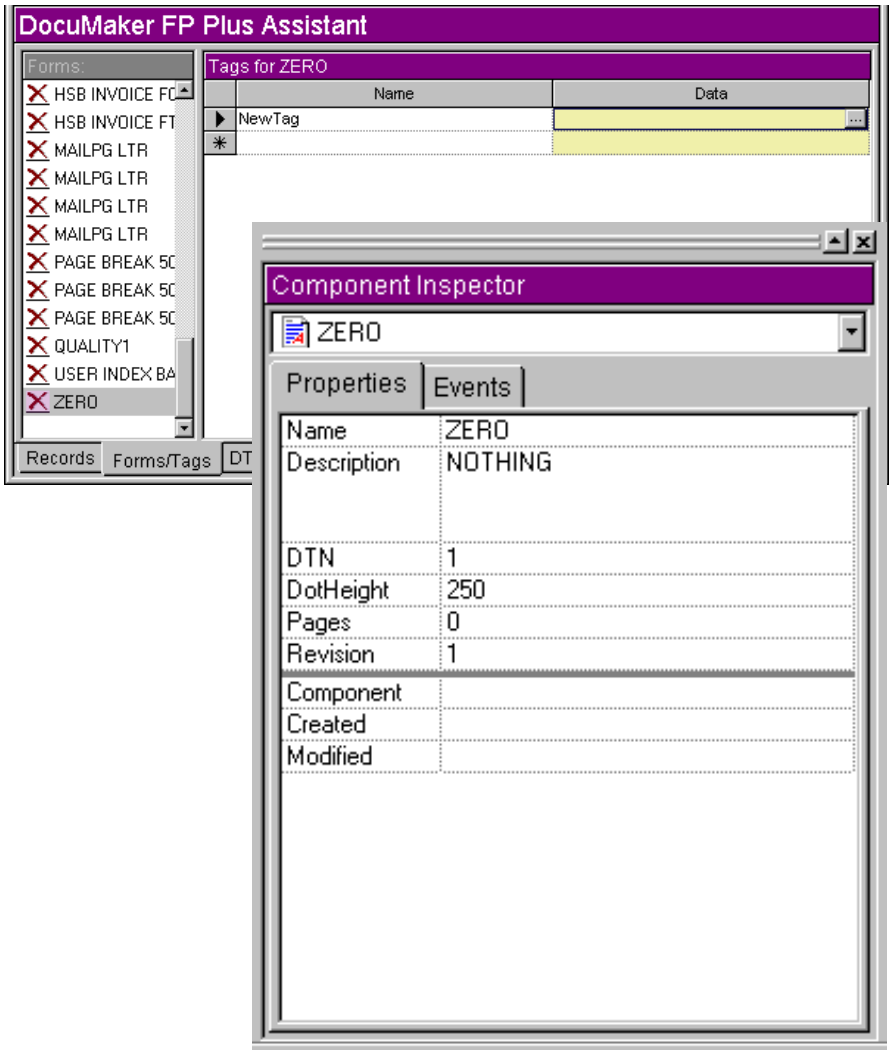


Figure 127: Form Properties

With the New Form Selected, this what the Form Properties would look like of the newly created form. These properties can be changed by clicking on the right side of the Component Inspector.

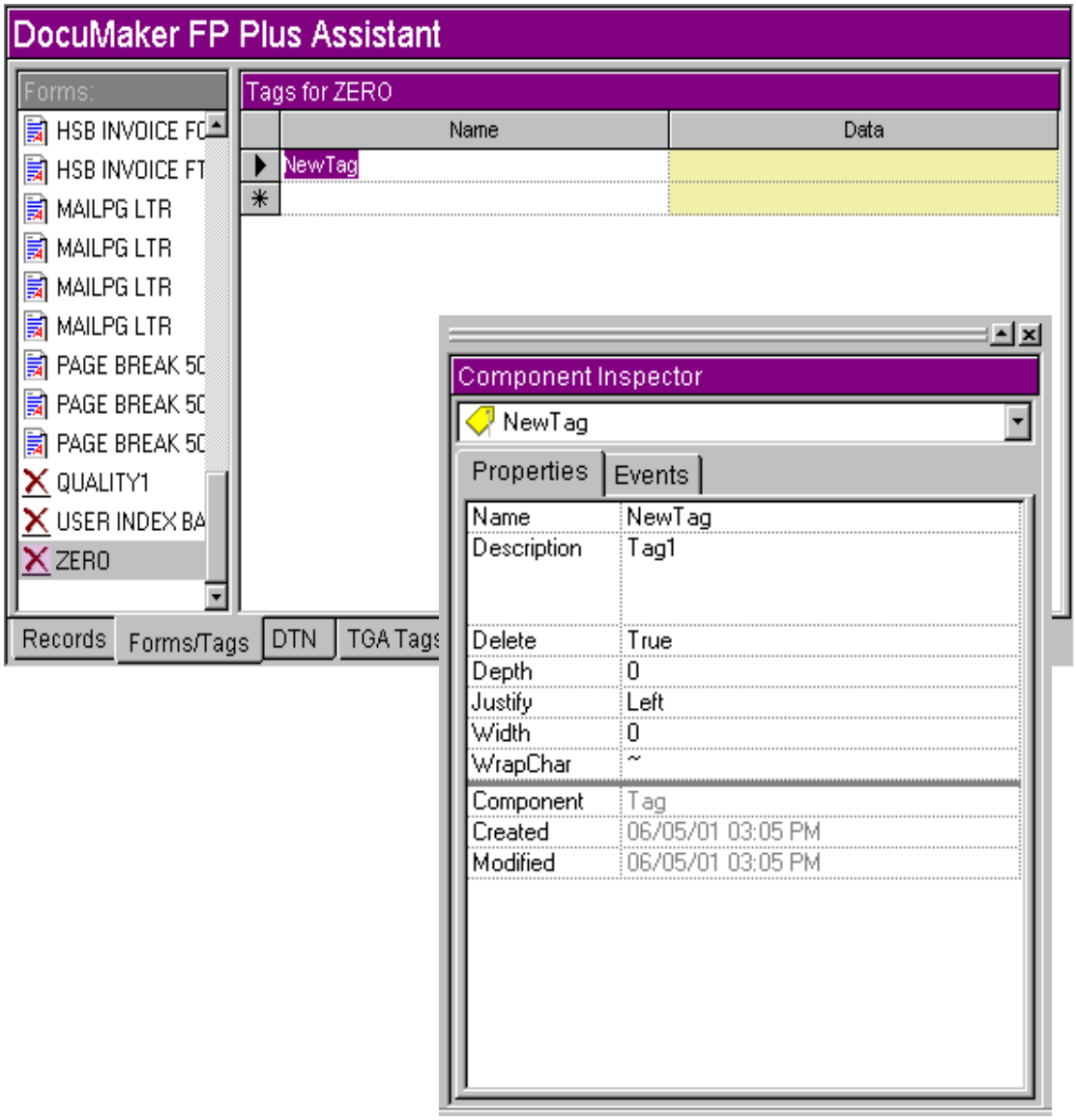


Figure 128: Tag Properties of New Tag

With the New Tag selected this is what the Tag Properties would look like in the Component Inspector. These properties can be changed by clicking on the right side of the Component Inspector

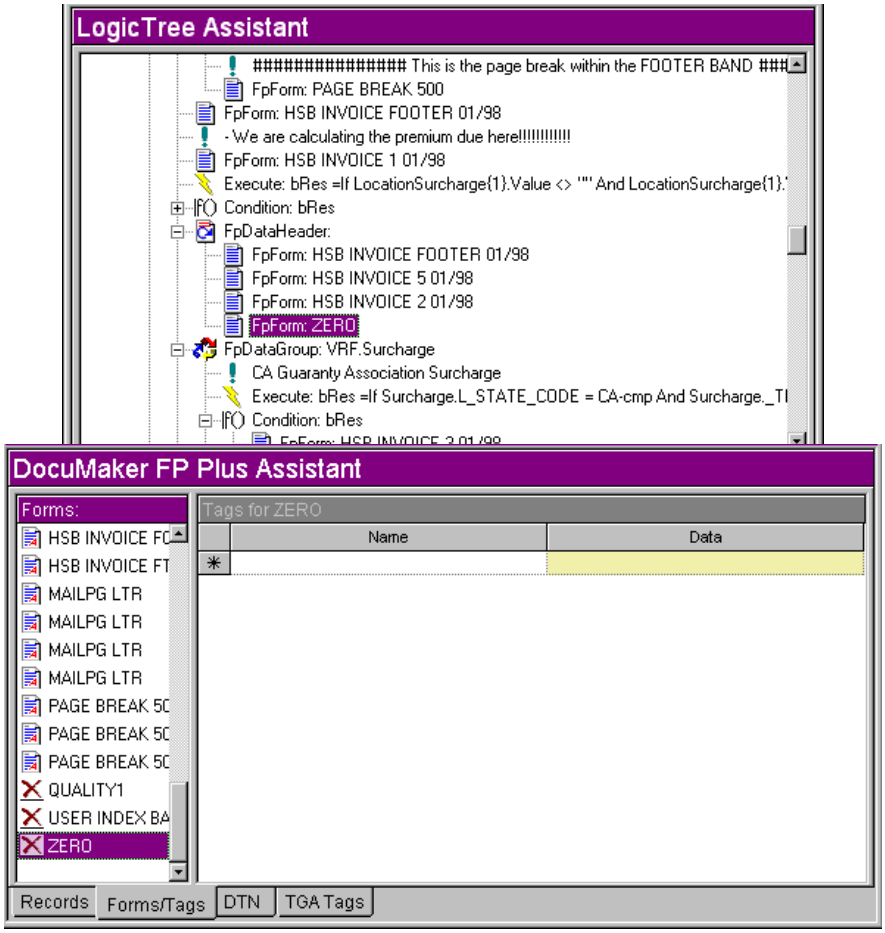


Figure 129: New Form without Tags

If NO, to creating new Tags, the new form will be added to the LogicTree Assistant, but note that new Form appears on the Forms list of the Documaker FP Plus Assistant, but has no tags.

You have the ability to add your own user-based forms; there may be a time that a form is needed that is not in EDL. You can create a form, by naming and putting some default information in this form in Transall and what Transall will do is hold the form as an invalid form, as the form as not been validated against the EDL. This will allow you to build all your business logic and test that business logic. Transall does have a feature that can validate a form and the validation process go back over and checks to make sure the tags that were mapped, match the tag names that were put into the EDL.

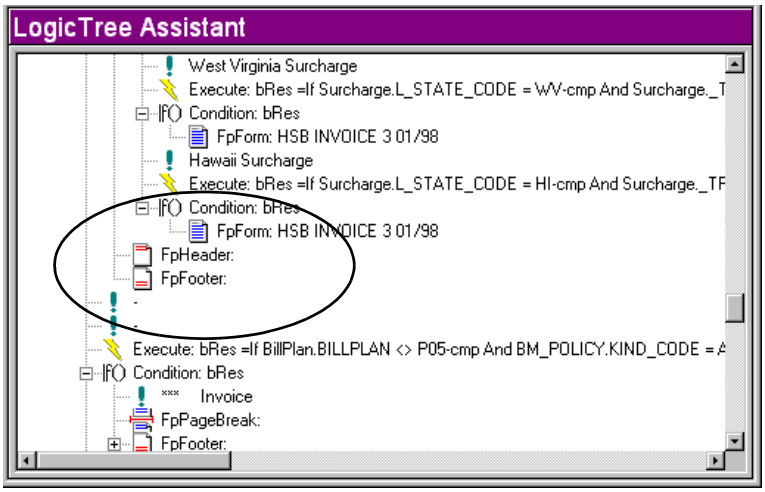


Figure 130: Example of FpHeader and FpFooter

FpHeader - When you drop a FPHeader into the project, you are telling the layout processor this list of forms under the FPHeader instruction is the first set of forms to be placed at the top of each page.

FpFooter - The FPFooter does the same thing that the FP Header does only it is a Footer and is a list of forms that gets transmitted at the bottom of each page. That does not mean that it will be transmitted to the bottom of the page, it only means that it will get transmitted as the last set of forms that landed on a page. When FP Plus realizes a page break has to happen, it is going to place the footer form list at the end of the list forms on the page, before the page break and that may physically be any place on the page. In fact, footers are not true page footers; they are really trailing forms on every page.

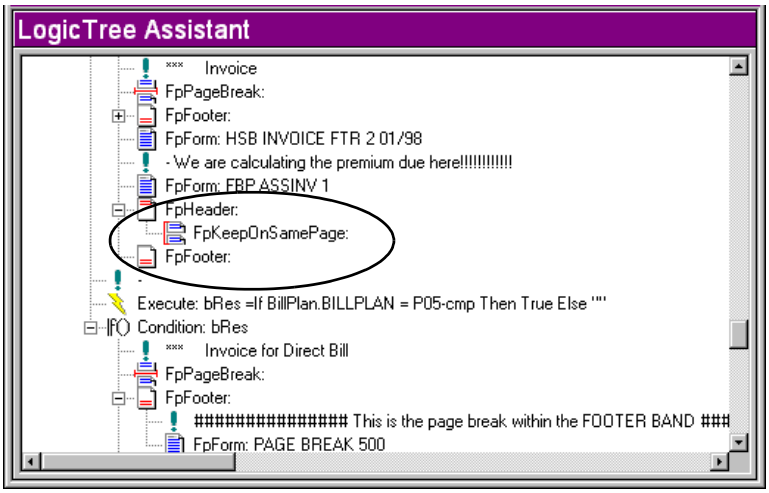


Figure 131: Example of FpKeepOnSamePage

FpKeepOnSamePage - Causes Transall to predict if a page break would happen in-between processing any of the forms under the FpKeepOnSamePage instruction. If a page break is predicted to happen, then Transall will cause the page break to happen in front of the forms in the Keep, so the forms will start on a new page and be more likely to be kept together.

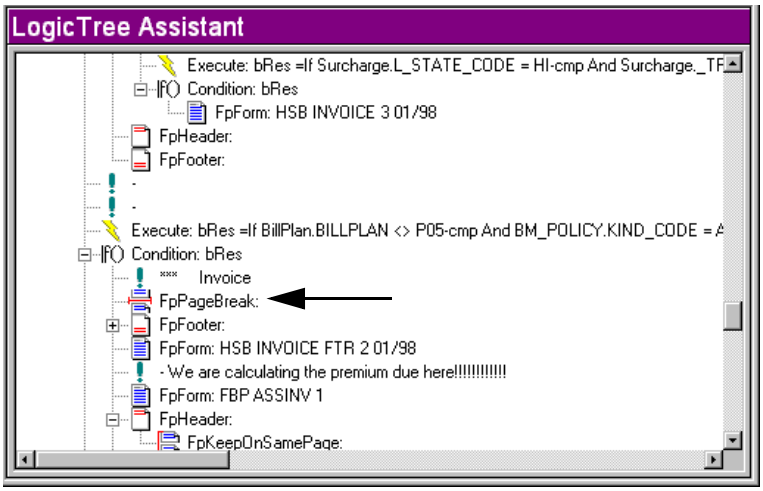


Figure 132: Example of FpPageBreak

FpPageBreak - This tells Transall to assume that a page break is going to happen right here. On a page break you can tell Transall to adjust the default dot height from that point forward in the document (Merge Set). The reason dot height adjustments might be required is to handle a change of page orientation or maybe a change in paper stock. Another use of FPPageBreak is to start a new section page count. Behind the scenes the Transall layout processor holds two kinds of page counting numbers, one is the total number of pages in the entire layout and the other way for the total number of pages in this section. By default there is one section, but every time you drop in a page break it can create a new section, which causes the section page count to go back to one, but does not effect the total page count.

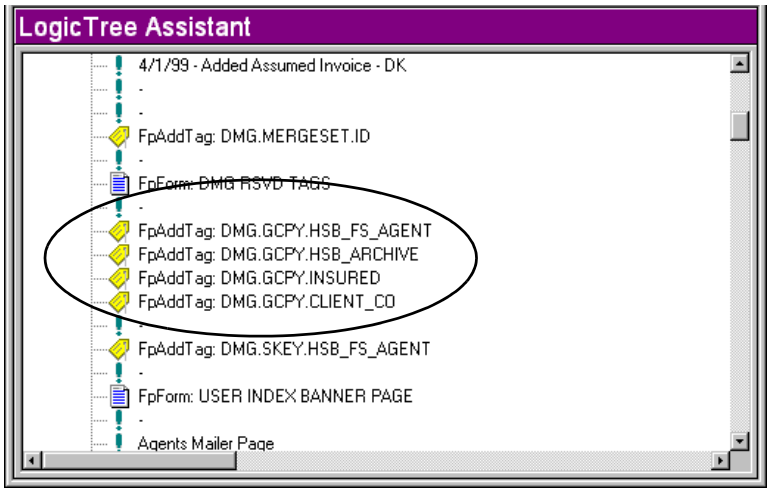


Figure 133: Example of FpAddTag

FpAddTag - There may a time when you may need to add a tag to the tag pool that does not have anything to do with a particular form. FPAddTag works the same way FPAddForm only it allows you to add a tag.

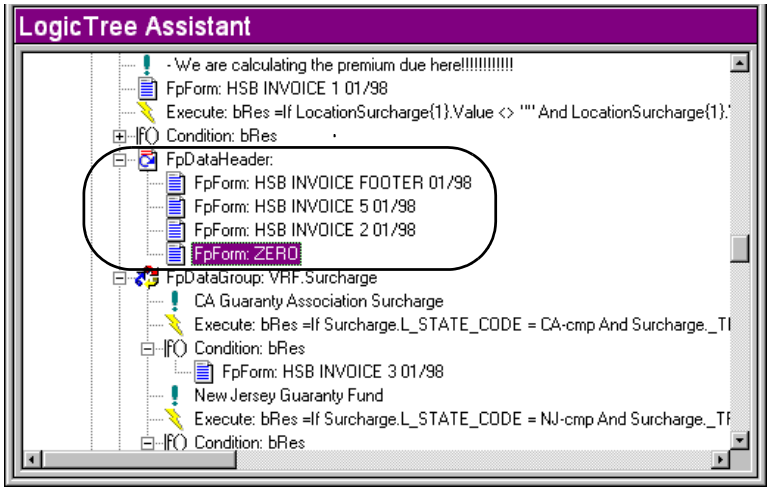


Figure 134: Example of FpDataHeader

FpDataHeader - A special header designed for a FpDataGroup. If a page break should happen while processing this Data Group, Transall will process this Data Header. The Data Header is only active while the Data Group is processing. The regular form Header will also be placed on the page along with the Data Header, something like a sub header. These Data Headers can be nested and you could have several Data Headers nested within several Data Groups.

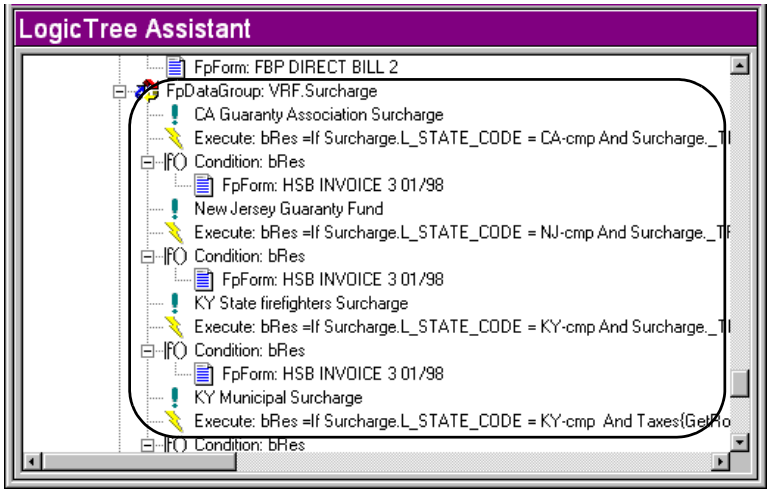


Figure 135: Example of FpDataGroup

FpDataGroup - An FPDataGroup tells the Transall layout to loop on a particular record, in the FPPlus data destination much like a Walk statement, and then perform the instructions underneath the FPDataGroup statement, once for each row in the record.

Scripted Data Sources and Destinations

OVERVIEW

Transall uses its Scripted Assistant to work with data sources and destinations that aren't supported by existing File, XML, or SQL options. This flexibility enables Transall to work with types of data that weren't anticipated by the authors.

Scripted data sources and destinations provide a framework of empty script subroutines, where you'll insert script, which will be called to perform operations on the custom-scripted sources and destinations. The operations supported are as follows:

- Open, Close, Prepare, GetNextRecord, and SetName at the source and destination level
- Read, Write, and Update operations at the record level

SCRIPTED ASSISTANT

The Generic File (Scripted) icon isn't normally displayed in the Add Source and Add Destination dialogs; you'll need to expand the list of available options.

To Access the Scripted Assistant

1. Select either **Project>Add Source** or **Project>Add Destination**.

Transall displays the Add dialog.

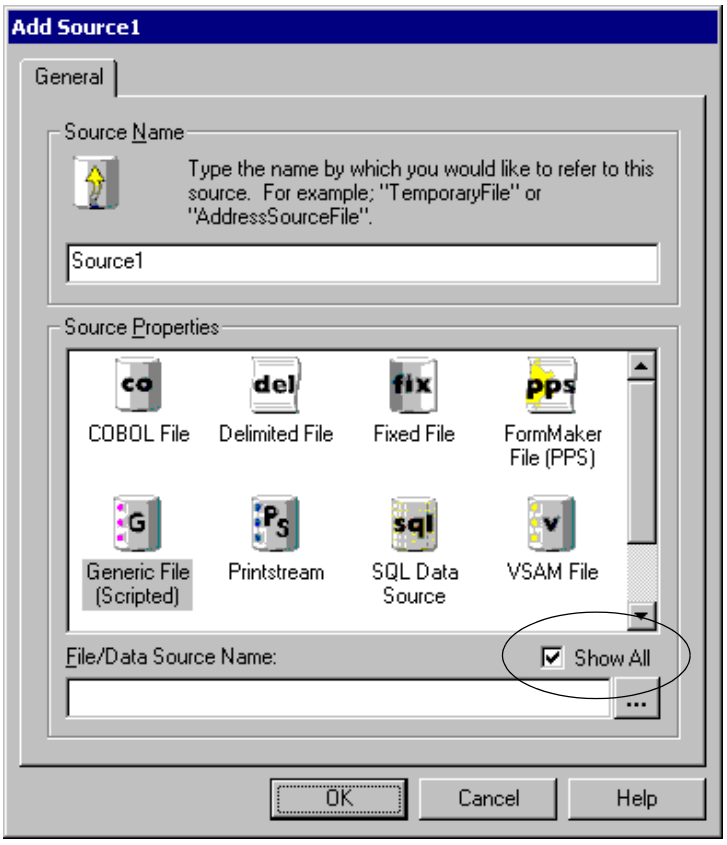


Figure 136: Add Dialog Box

2. Enable (check) **Show All**.

   Checking **Show All** displays infrequent data sources and destinations for your selection.

3. Select **Generic File (Scripted)**.

   Transall displays the Scripted Assistant.

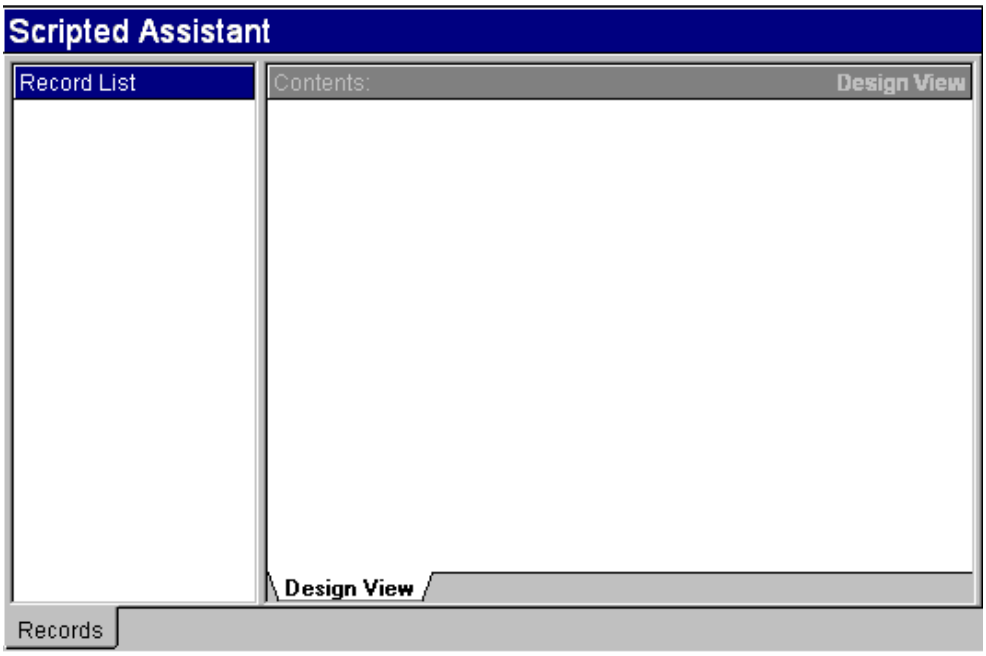


Figure 137: Scripted Assistant

Scripted sources and destinations work on a file input/output model. Transall attempts to make them operate as files so they are easier to use with existing File, XML, or SQL sources and destinations.

The Scripted Assistant builds a framework of empty script subroutines that you should fill with script to perform operations on the sources and destinations. These subroutines will then be called by Transall's LogicTrees and other data sources or destinations.

The operations supported by the framework of subroutines built by the Scripted Assistant are as follows:

- Open, Close, Prepare, GetNextRecord, and SetName at the source or destination level
- Read, Write, and Update at the record level

Like other ones, Scripted sources and destinations interact with Transall via records. These records act as a buffer to write to and read from scripted sources and destinations.

Scripted Sources and Destinations Operations (Events)

The following operations are supported at the data source or destination level.

Operation	Result
Open	Called to initialize a data source or destination.
Close	Called to de-initialize a data source or destination.
GetNextRecord	Called to get the next piece of data from a data source.
Prepare	Called before GetNextRecord to enable a data source to get ready to provide the next piece of data from the data source.
SetName	Not called directly by Transall. Provided so a name string can be set on the data source or destination. This string can hold connection information for the data source or destination.

The following operations are supported at the record level in a data source or destination.

Operation	Result
Read	Called to load a record in Transall with data from a data source.
Write	Called to write data from a record in Transall to a data destination.
Update	Called to replace data in a destination with data from a record in Transall.

Scripted Source and Destination Properties

The following properties are supported at the data source or destination level.

Property	Meaning
Name	The name of the data source. A data source name must be unique within a Transall project.
Description	A comment about the data source or destination that helps to document the Transall project.
FileName	The data string for the data source or destination. This string can hold connection information.
OpenMode	The valid choices are as follow: • **Automatic**—causes Transall to generate code to call the Open event (script) on the data source or destination when its records are referenced in a LogicTree. • **Manual**—prevents Transall from generating code to call the Open event (script) on the data source or destination.

The following properties are supported at the record level.

Property	Meaning
Name	The name of the record. A record name must be unique within a Transall project.
Description	A comment on the record that helps to document the Transall project.

XML Plus Data Source

OVERVIEW

Transall now supports a new Data Source called “XML Plus” that reads XML files and makes their data available to other Transall data sources and destinations. The XML Plus data source differs from the existing XML data source in its ease of use. The XML Plus source does a better job of automatically organizing the XML data to make it easier to use in Transall LogicTrees and data destinations. For most applications XML Plus is going to be the preferred method to read XML into Transall.

XML PLUS FEATURES

What makes the XML Plus source so ‘plus’ is its ability to scan a sample XML document and automatically organize the document's elements into a hierarchy of records and fields for use in Transall. What XML Plus does is look at the elements in the XML document; elements that have child elements are treated as records and elements that have no children are treated as fields on the records. The XML document is then presented as a hierarchy of records with fields that is used easily alongside traditional Transall data sources and destinations. Here is an example of a typical XML Plus source

:

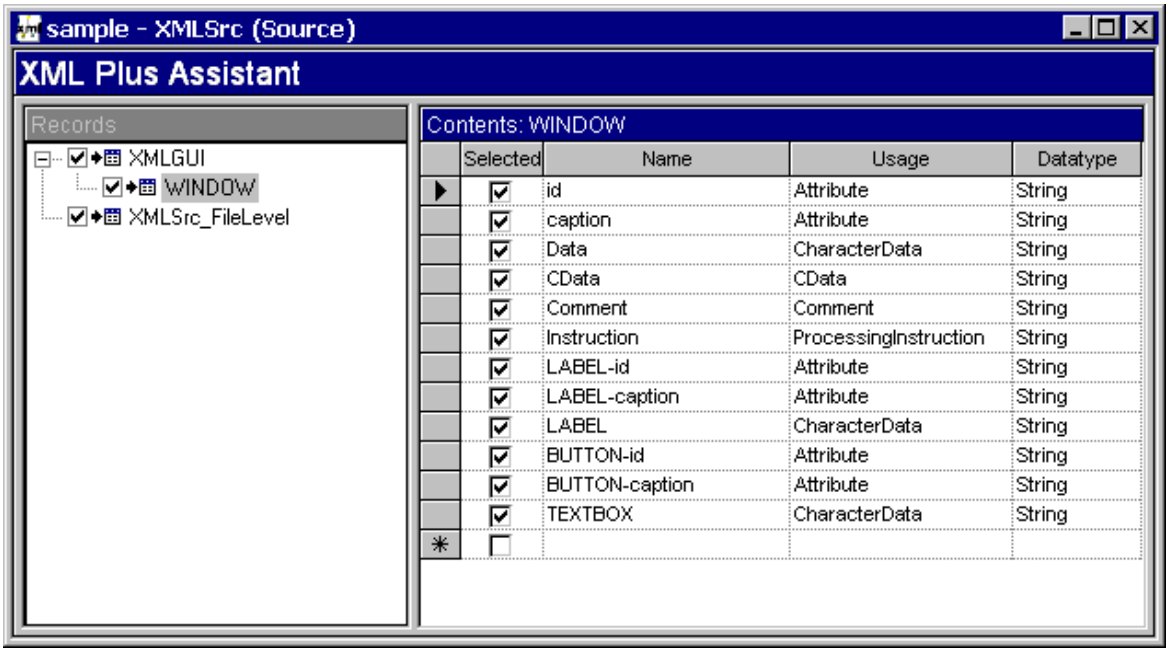


Figure 138: Sample XML Plus Source

In the XML Plus Assistant:

- Records are a hierarchical representation of the XML document's elements that have child elements.
- Fields are usually elements that have no children. In this case the field's value is populated from the character data for the element in the XML document. Transall also looks for XML Attributes, CData, Processing Instructions and Comments in the sample XML document provided for the data source. If any of these items are found for an element fields are created to capture the items.

Part of the setup process for XML Plus is selecting the records and fields that Transall will capture when processing the XML document. In the example the records and fields with a check mark will be captured and populated with data by the XML Plus data source. These records and fields will also be available in the various Transall helper tools for mapping data. Not all records need to be selected but all that are found in the sample XML document provided for the data source are selected by default. Only the records and fields that provide data to Transall data destinations and other logic used in the project need to be selected for capture by the data source.

ADDING RECORDS AND FIELDS

There are two ways to update an XML Plus data source with new records and fields. The first way is to rescan a sample XML document for the source. This can be done by updating the FileName property of the XML Plus data source to point to an updated sample XML document that is representative of the XML documents you will process with this data source and then select the **Resource>Refresh from file** menu item to have XML Plus rescan the XML document looking for new XML elements to create records and fields for in the XML Plus hierarchy of records. If the updated sample XML document does not contain all the XML elements that have records and fields defined in the XML Plus hierarchy, these records and fields are not removed by the rescan activity. Should you need to remove records or fields; records can be removed by clicking on the records and using the **Resource>Record Delete** menu item. Fields can be removed by selecting the field and pressing the **DELETE** key, or selecting the field row, clicking the right mouse button and selecting **Delete Row** from the menu.

If the sample XML document provided for the data source does not contain all the elements and data items that you want to add to the XML Plus hierarchy of records you can add records and fields manually. This is the second way to update an XML Plus data source with new records and fields. Records are added by opening the XML Plus data source's XML Plus Assistant. Click on an existing record that this new record will be placed under. Selecting the **Resource>Record Add** menu item to bring up the Add XML element dialog:

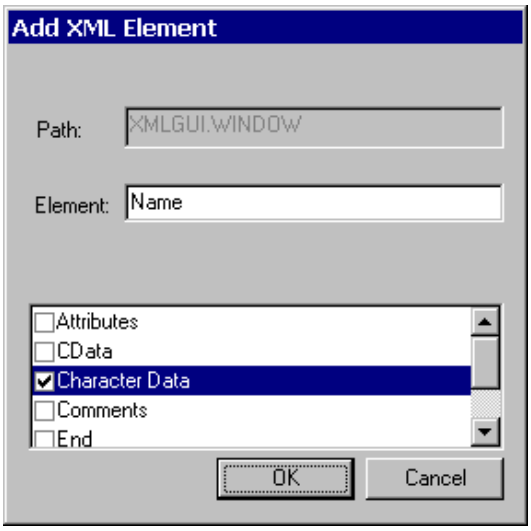


Figure 139: Add XML Element Dialog Box

Enter a name for the new record in the “Element” field and select the types of data you wish to capture for this element record. Character Data is selected by default. A field will be predefined in the record for each type of data you select in this dialog. The element name you enter will also be used to both name this record to Transall and to name the element that Transall should look for in the XML document to populate this record with data at run-time. Also note the value of the “Path” field in the Add XML Element dialog. This field shows the list of parent elements in the XML document's element hierarchy that Transall will search when scanning for data to populate this record. For example, the following example displays the element named “MyNewElement” that was added to the Transall XML Plus data source.

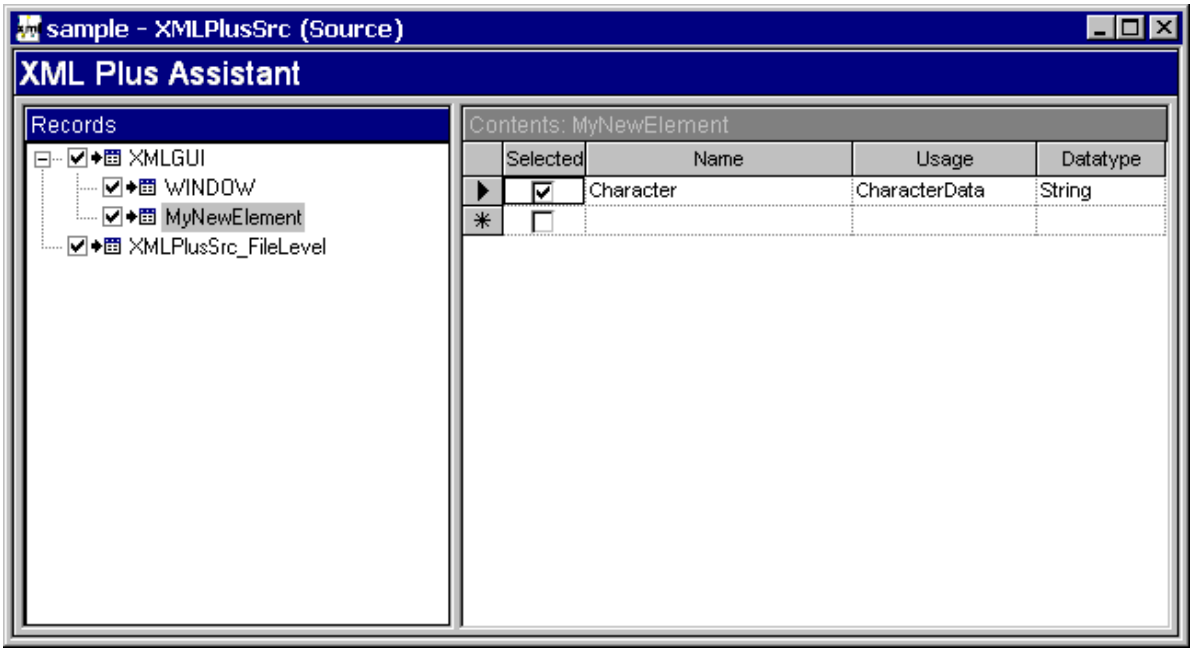


Figure 140: “MyNewElement” Add to the Transall XML Plus Data Source

At runtime, Transall will add a row to the MyNewElement record in the data source when it reads the XML document and locate an element named “MyNewElement” with character data under an element named “scanneddata”. The reason Transall looks under the scanneddata element for the MyNewElement element is because of the IdentifierValue property for the record.

Element IdentifierValue Properties

The IdentifierValue property lists the element hierarchy path that Transall will use when locating character data in an XML document to populate the MyNewElement record in XML Plus. In this case the value “scanneddata.MyNewElement” will cause Transall to look for character data under the “MyNewElement” element that is under the “scanneddata” element in the XML document.

Special “mask” characters can also be used in the IdentifierValue property of a record for added flexibility. These are the same characters that can be used in Like conditional statements in Transall. When the Masked property is set to true Transall uses 'Like' expression syntax to match an element's data in the XML element hierarchy path to a record in the XML Plus data source. For example, an IdentifierValue property of “*.MyNewElement” will map the data from an XML element named MyNewElement found anywhere in the XML hierarchy under the root element. This is because the '*' value in the Masked property means match one or more characters so 'Anything.MyNewElement' is a match.

FIELD USAGE

When Transall scans a sample XML document that is representative of the XML documents you will process in XML Plus it creates records to represent elements that have child elements and fields to represent child elements that have no children. All the fields automatically setup by XML Plus are configured to capture character data from the XML document. The usage property for these fields is set to “Element”. This means that the field will be populated with character data from a child element of the record's element whose name matches that of the field. There are several field usages that can be setup. Field usages other that element must be defined manually. The available field usages are:

- **Attribute**:Populated with data from an XML attribute on the record's element, whose name matches the name of the field.
- **CData**:Populated with data from XML CData under the record's element.
- **CharacterData**:Populated with XML Character Data under the record's element.
- **Comment**:Populated with XML Comment Data under the record's element.
- **Element**:Populated with XML Character Data from an element whose name matches the field name that is under the record's element.
- **ProcessingInstruction**:Populated with XML ProcessingInstruction Data under the record's element.

All field usages except “Element” cause XML Plus to map data from the element associated with a record to a field in the XML Plus record hierarchy. For example, setting up a field with a usage of “Attribute” causes XML Plus to map attribute data found in the XML document on the Record's element to a field in the XML Plus record hierarchy. In the case of setting up a field with a usage of “Element” XML Plus looks for an XML element under the record's element and maps the child element's character data to the field. Only fields with a usage of Element look down the XML hierarchy to a child element in the XML document under the record's element.

TRANSACTION BOUNDARIES

One element in the data source must be selected as the “transaction boundary” element. By default the transaction boundary element is selected as the root element in the source. This element plays a special role in telling Transall how much of the XML document to load in a single “transaction”. A transaction for XML Plus defines the amount of XML data loaded into the XML Plus record hierarchy each time a GetNextRecord event is processed on the XML Plus source. Transall uses a progressive XML parser to load data from the XML document to the record hierarchy. Because of this Transall does not have to read the whole XML document into memory at once. The document can be read in chunks. This is useful when processing large XML documents. In the case of the example, the “sqform” record makes sense to be the transaction record. Transall will then read the XML document into memory one sqform element at a time and populate the whole record hierarchy in the XML Plus source for each sqform element it finds in the source XML document. So if the source XML document has thousands or even millions of sqform elements, Transall can process them, one at a time. This feature makes it possible for XML Plus to process XML documents of unlimited size. An element does not need to be selected (checked) to be used as the transaction boundary.

XML PLUS AND THE LOGICTREE

The Transall LogicTree interacts with XML Plus through a Walk statement just like other data sources. What is different is that XML Plus reads a hierarchy of records into the data source for each loop through the walk statement (see the “Transaction boundaries” discussion in the preceding section). Once a transaction of data has been read in to the source, the source's record hierarchy can then be traversed to process the transaction data. The Transall LogicTree has a feature “Generate break instructions” that will cause it to generate instructions to process the various records of a data source. In the case of an XML Plus source, this feature generates a hierarchy of record Walk statements. These statements will loop through each record in the data source so it can be processed by the LogicTree. The following shows a LogicTree with a Walk instruction on the XML Plus data source we have been using as an example. The Walk instruction has had the “Generate break instructions” feature applied to generate a hierarchy of record Walk statements for our XML Plus data source:

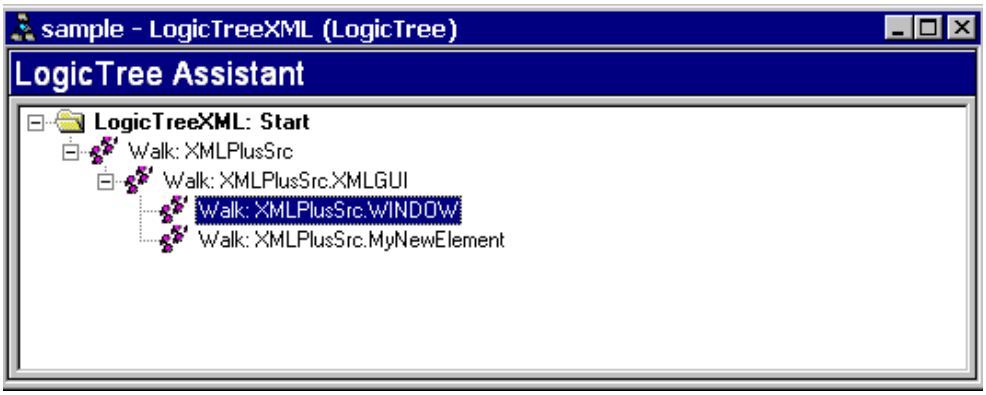


Figure 141: Generate Walk Statement

After the Walk statement has been setup on the data source and the “Generate break instructions” tool applied, other Transall instructions can be dropped in the LogicTree to perform processing on each record in the XML Source.

For example, let's say we want to write the XML data back out as a delimited file. To do this we add the delimited data destination to the Transall project. Use the record “Copy From...” feature to build records in the delimited data destination based on the records in the XML Plus source. The record Copy From feature also generates Maps that map the data from the XML Plus source to the Delimited data destination.

We then drop the Map and Output instructions on the LogicTree to write the delimited file as so:

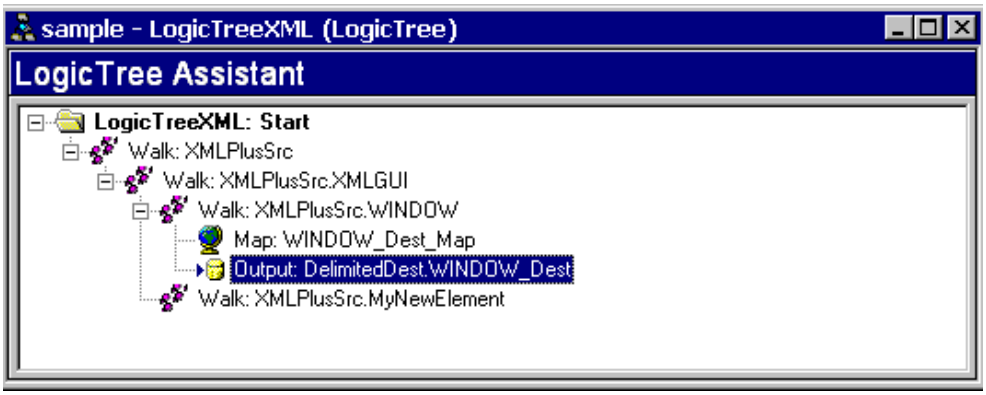


Figure 142: Delimited Output File

Note that instructions have now been placed under each Walk statement for the records in the XML Plus record hierarchy. These instructions will be executed once for each row in the hierarchy in the XML Plus data source. In this example these instructions write the XML data to a Delimited file destination and cause the Delimited file destination to write records to the delimited file.

XML PLUS SOURCE VS. XML SOURCE

XML Plus was developed to be easier to use than the original XML data source in Transall. The XML Plus data source organizes and loads the XML data into a hierarchy of Transall records. The original XML data source provides no such hierarchical organization. The original XML data source in Transall is really a direct implementation of SAX XML parsing. The original source “signals”, through record breaks, as each XML data item is parsed from the XML document. So signals are send for open and closing XML tags, String data, CData, Processing Instructions, etc. The XML Plus data source “signals” only when it has loaded a section of an XML document in to the data sources hierarchy of records. The size of the section, which can be the whole XML document, is set by the Transall developer by selecting a transaction element in the XML Plus data source. By reorganizing the XML data into a hierarchy of records and fields the XML data is much easier to work with.

For 99% of XML read processing, the XML Plus data source should be used. The original XML source can be used when the low-level SAX XML parsing nature of the source is more advantageous to use.

XML PLUS SOURCE COMPONENT PROPERTY DETAILS

XML PLUS SOURCE

Name: The name of the data source in Transall.

Description: Comment field.

Autoencoding: True or False. When True causes Transall to assume an encoding of the 'Encoding' property.

Encoding: Default encoding to be used when the Autoencoding is set to True and the XML document has no encoding defined.

FileAttributes: List of file-level XML data to be supported by this XML Plus source. The can be set to one or more of the following values: Comment, Inst, Markup, or Doctype. Each of these valuescause a record to be added to the XmlPlus source to collect Comment, ProcessingInstruction, Markup or Doctype XML tags that exist outside the root XML element in the source XML document.

ErrorHandler: Automatic or Manual. When set to 'Automatic' this causes Transall to automatically handle all normal processing errors on the XML source (such as end of file). When set to 'manual' this causes Transall to ignore all errors on the XML source leveling error checking to the Transall developer.

FileName: Name of the file that XML Plus should scan to build the record hierarchy for the data source. Also this is the default name of the file that will be opened by XML plus at run-time when process XML input files.

LimitSchemaRead: Number of tags to parse out when gathering information about an XML document's structure. Zero means no limit. This limits the amount of a sample XML document that Transall will load to determine the structure of the data in the XML document.

NamespaceSupport: True or False. If True this indicates that the parser should parse tags allocating them to their appropriate namespace in the XML document (the URL for that particular node) when processing a document containing namespaces. If this is False then the NameSpace parsing features are bypassed.

OpenMode: Automatic or Manual. When set to Automatic this causes Transall to automatically generate calls to the Open and Close events for the data source when the source is referenced in a Transall LogicTree.

Resource: Resource type. Always “XML Plus”.

Transaction: This is the name of the Transaction Boundary element. This element tells Transall how much of the XML document to load into the XML Plus record hierarchy each time a GetNextRecord event is processed on the XML Plus source. Transall read the XML document into memory one Transaction Boundary element at a time and populate the whole record hierarchy in the XML Plus source for each Transaction Boundary element it finds in the source XML document.

Validation: Always, Automatic, Never. This controls the level of XML validation Transall requests from the Xerces XML parser Transall is using. Always enforces the highest level of validation. Automatic enforces a lower level of validation where validation error can be ignored without stopping the XML processing process are ignored. Never performs no validation.

XML PLUS SOURCE RECORD

Name: The name of the record in Transall. Record names must be unique. Transall may change the name of a record from the Element name that the record represents to make it unique. Transall may also change the name of a record to make the name valid for Transall. Record names cannot contain spaces or special characters such as “)(*&^”.

Description: Comment field.

IdentifierValue: Lists the element herarchy path that Transall will use when locating data in an XML document to populate the record's fields.

Masked: True or False. When True this indicates that the IdentifierValue property should use Like syntax to match element hierarchy path to data in the XML document when populating the record's fields.

Selected: Yes or No. When Yes this indicates that this record's fields will be populated with data and available for reference in other Transall components (Maps, LogicTrees, Scripts). When No this indicates that this record's fields will NOT be available for reference in other Transall components and that the Transall script generator should not generate script for this record.

Component: Always “Record”

XML PLUS SOURCE RECORD FIELD

Name: The name of the record field in Transall. Field names must be unique on a record. Transall may change the name of a field to make the name valid for Transall. Field names cannot contain |spaces or special characters such as “)(*&^”.

Description: Comment field.

Data Type: This is the Transall datatype that will be used to hold the data read from the XML document for the field. See Transall DataTypes for more information on DataType details.

Identifier: Lists the element hierarchy path that Transall will use when locating data in an XML document to populate the record's fields. This property is available only for Usage=Element fields.

Selected: Yes or No. When Yes this indicates that this record field will be populated with data and available for reference in other Transall components (Maps, LogicTrees, Scripts). When No this indicates that this record field will NOT be available for reference in other Transall components and that the Transall script generator should not generate script for this field.

Usage: Element, Attribute, CData, CharacterData, Comment, None, or ProcessingInstruction. Element means the field is populated with XML Character Data from an element whose name matches the field name that is under the record's element. Attribute means the field is populated with data from an XML attribute on the record's element, whose name matches the name of the field. CData means the field is populated with data from XML CData under the record's element. CharacterData means the field is populated with XML Character Data under the record's element. Comment means the field is populated with XML Comment Data under the record's element. None means the field is not populated with XML data. ProcessingInstruction means the field is populated with XML ProcessingInstruction Data under the record's element.

Componet: Always “field”

Event-based XML Data Source

Overview of Event-based XML Support

Transall supports both reading and writing XML files. For reading XML files in event mode Transall utilizes a SAX API based XML parser, this type of XML parser is an “event” based parser vs. a “tree” based parser. A tree-based parser compiles the source XML document into an internal tree structure in memory and then allows an application to query and tree for information. This is how the XML Plus data source works. Conversely, an “event” based parser reports parsing events to the host application and does not compile an internal tree of XML source data. The events reported to the host application are things like “start document”, “start element”, “end element”, etc. Tree based XML parsers are useful for a wide range of XML applications, but tree parsers can put a great strain on system resources, especially if the XML document being processed is large.

How Event-based XML Parsing works in Transall

Transall interacts with XML data sources in a way very similarly to the way it interacts with multi-record file data sources. To Transall each XML element in an XML file is like a source record and each XML attribute on the XML elements are like source fields on a record. Through the Transall editor GUI you setup Transall data source records in an XML data source and associate these records with the names of XML elements found in the source file. Transall supports an import feature to help you scan a representative XML source file to acquire the names of XML elements and attributes that can appear in the XML data source document.

Setting up an Event-based XML Source

To setup an event based XML data source in Transall start the Transall editor (TRANEDIT.EXE) and select a Transall project for editing or start a new Transall project.

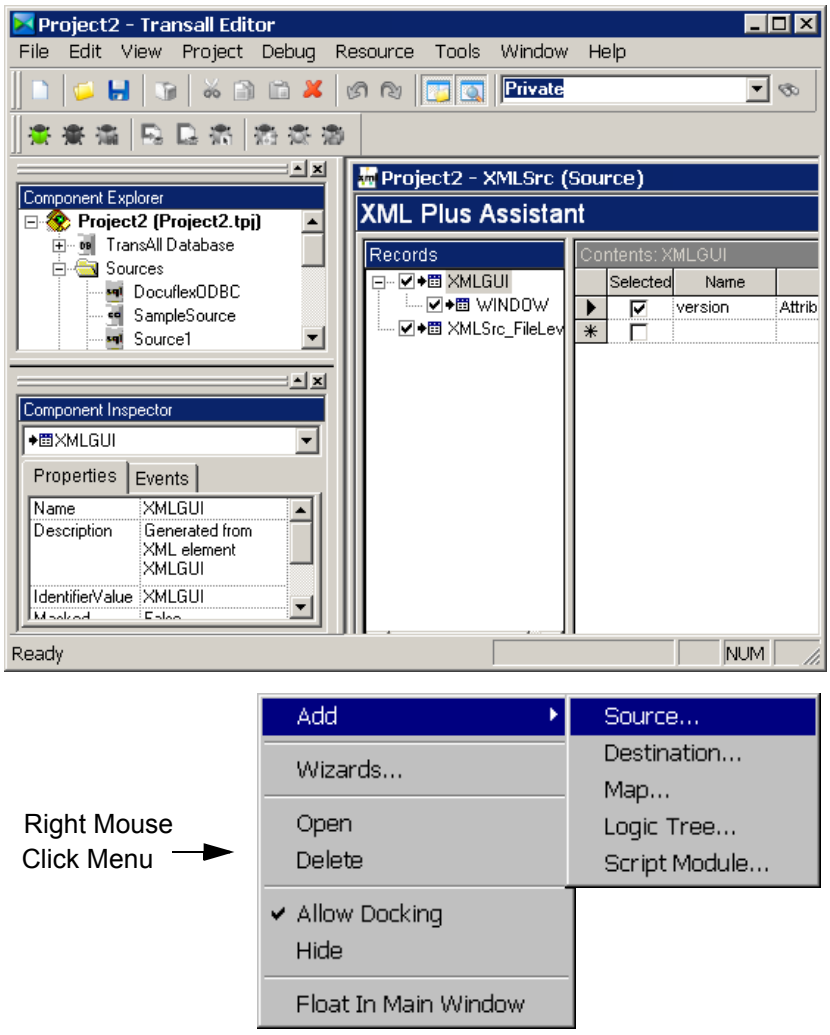


Figure 143: Project>Add Source and Add Menus

Click on the **Project>Add Source** menu item or right-mouse click in the Component Explorer to open the Add menu.

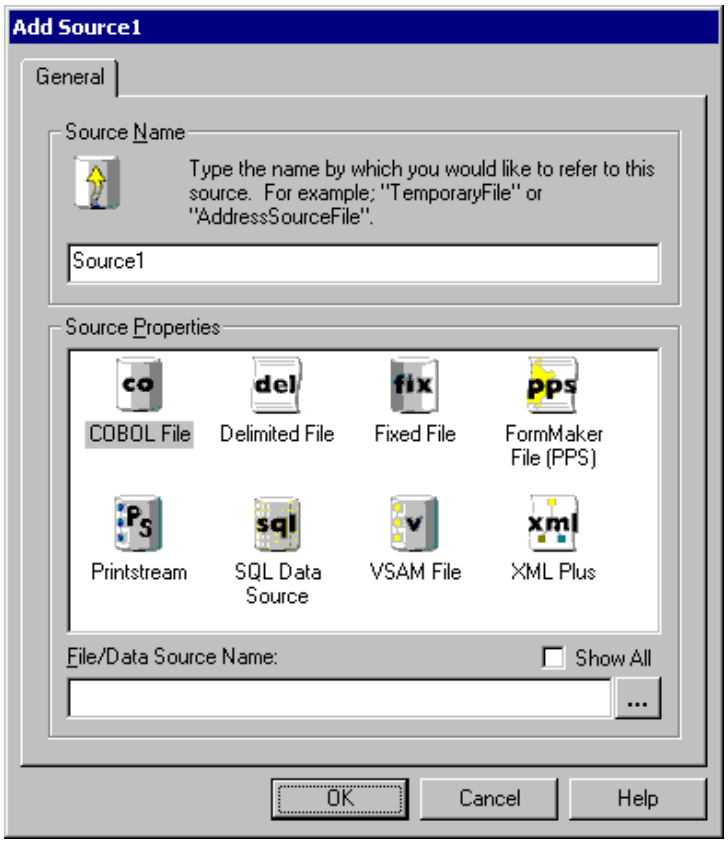


Figure 144: Add Source Dialog Window

This will display the Add Source dialog window. In the dialog window, update the Source Name to something meaningful for the data source.

Note You cannot use special characters or spaces in the name of a data source but you can use underscores "_" and dashes "-". It is often a good idea to suffix your data source names with a meaningful value like "-Src" or "-ScHo".

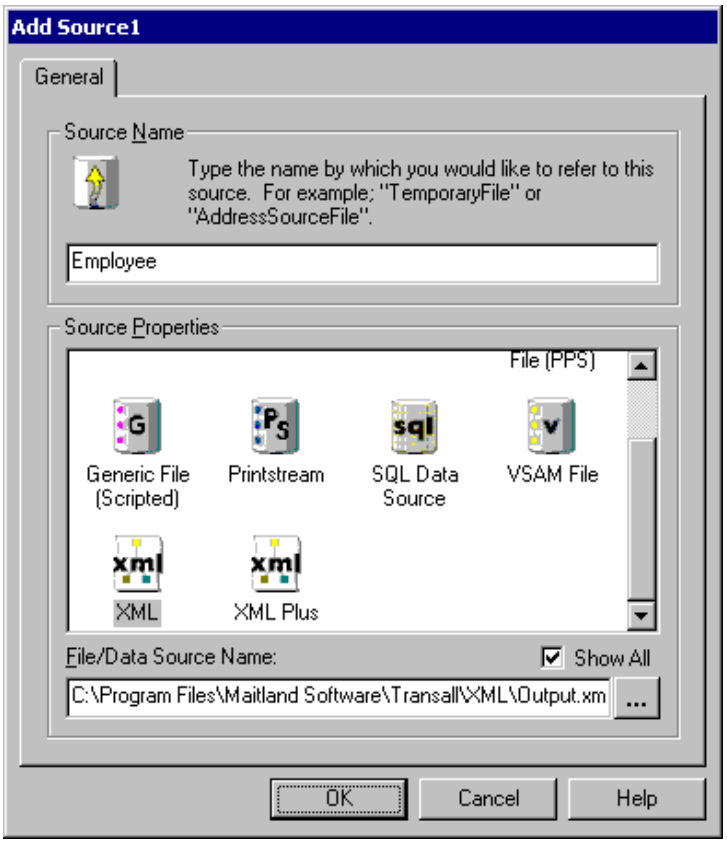


Figure 145: Completed Source Dialog Window

After typing in a source name, select the type of source from the Source Properties list. For XML data sources select “XML”. Last, select a file name for this source, either type in the file name or click on the button with three ellipses to display the Select File dialog window, from this dialog window select or type in a new file name. After selecting a file name click the “OK” button in the Add Source dialog window to complete the XML source creation.

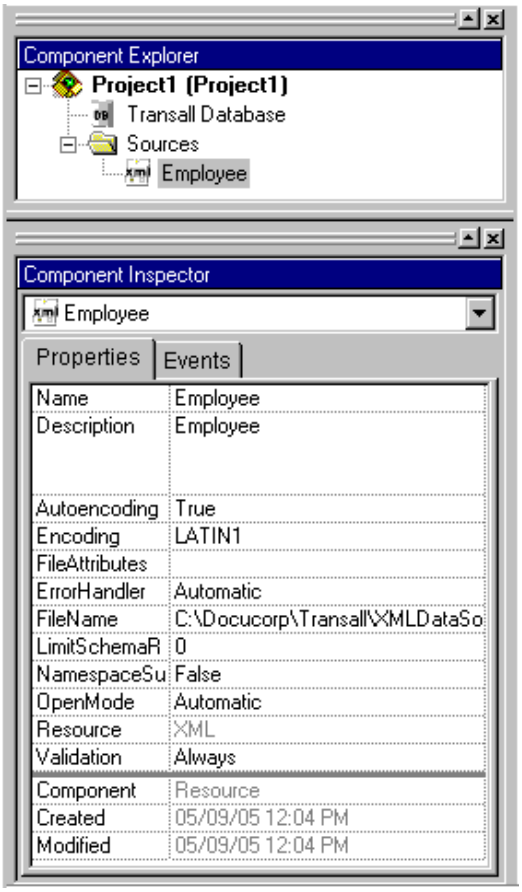


Figure 146: Component Explorer and Component Inspector

You will see your new data source listed in the Component Explorer under the Sources component branch. In addition, the Component Inspector will show the details of your new XML data source.

ADDING A RECORD FOR AN XML DATA SOURCE

To add a record type to a Transall XML data source, you must first locate or set up a new XML data source for the record.

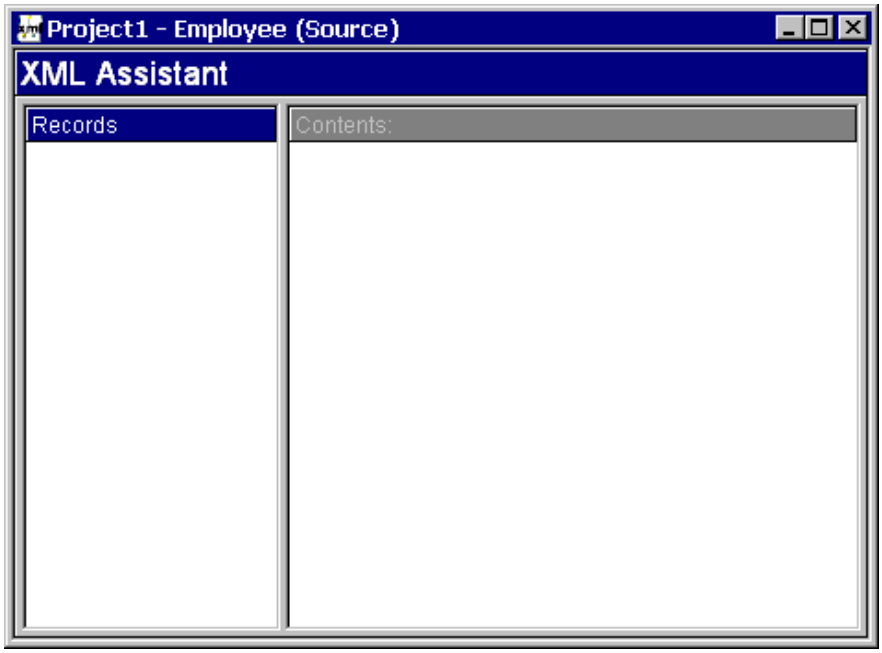


Figure 147: Data Source XML Assistant

Once an XML data source has been selected, open the XML Assistant by double clicking on the desired data source in the Component Explorer. Records can be added to an XML data source by either directly setting up the record or by importing record definitions from a representative XML document.

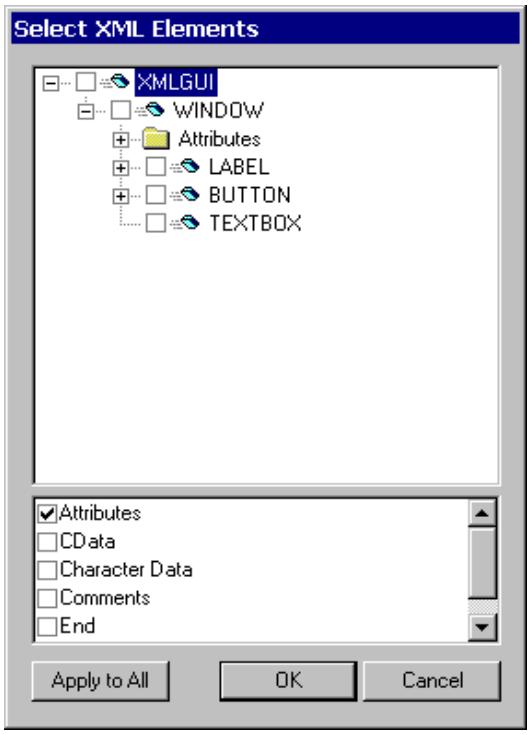


Figure 148: Select XML Elements Dialog

To import records from a representative XML document, select the **Resource>Record Import** menu item. This will display a Select XML Elements dialog that is populated with a tree showing the XML element hierarchy from the XML file selected for the XML data source. Right mouse click on elements in the hierarchy to select groups of elements or open up the XML element hierarchy tree. In the lower part of the Select XML Elements dialog select the types of data that Transall should collect from the XML document. Transall can collect the following data item types Attributes, CData, Character Data, Comments, and Processing Instructions. For each data item type, Transall sets up a record in the XML data source. These records work just like records from a multi-record type source file. When you build a LogicTree that reads this XML data source, you should set up Case statements under your Walk or Input statement for this XML data source. Transall will trigger a record found “Case” statement in the Transall LogicTree as it parses out the XML document.

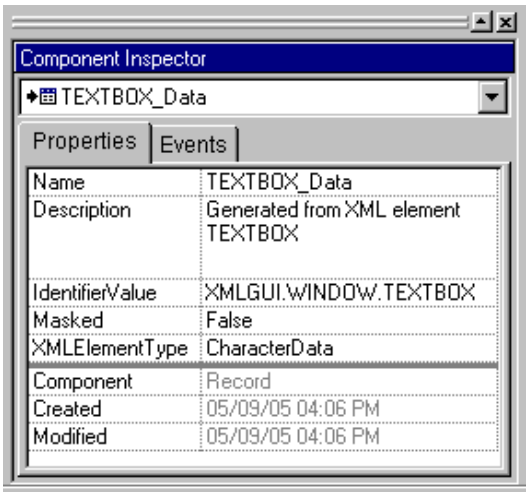


Figure 149: Component Inspector

After adding records to the XML data source, you can change the record names in the Transall Component inspector to a meaningful name for each record.

Note A record name cannot contain special characters or spaces but can contain underscores “_” and dashes “-”.

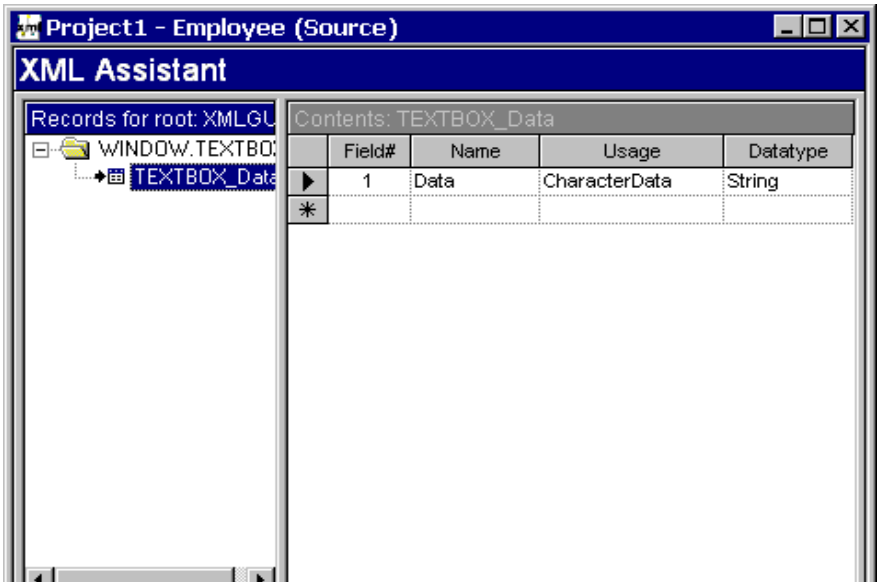


Figure 150: XML Assistant with Records

Once the records have been setup, you can add fields (or more fields) to the records. To add a field, enter the field name into the Name column of the field row with an asterisk in the row border. Press the tab key to move to the Usage column and select a usage type for the field. Press the tab key to move to the Datatype column and select a data type for the field. Setup one field on each record for each XML Attribute that Transall can expect to find in the XML document.

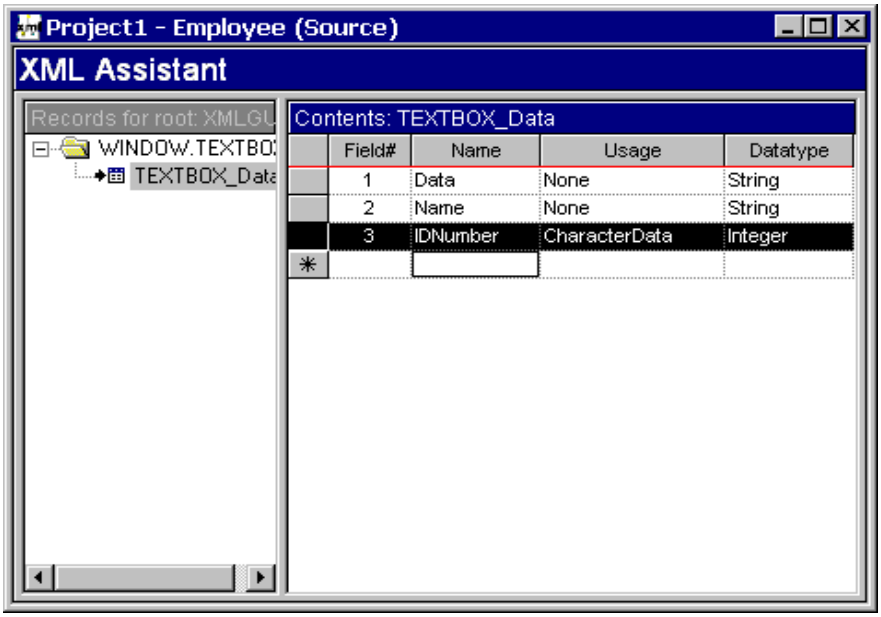


Figure 151: Moving a row in the XML Assistant

If you want to move a field up or down on the record you can do so by clicking in the gray area to the left of the field's number column in the XML Assistant. This will select the field's row in the XML Assistant. With the row selected, click and hold the left mouse button, then drag the field from the gray area to the left of the field's number column to a new location in the list of fields for the record. You will see a colored line in the list of record fields as you perform the drag that indicates where the field will go when you drop the field.

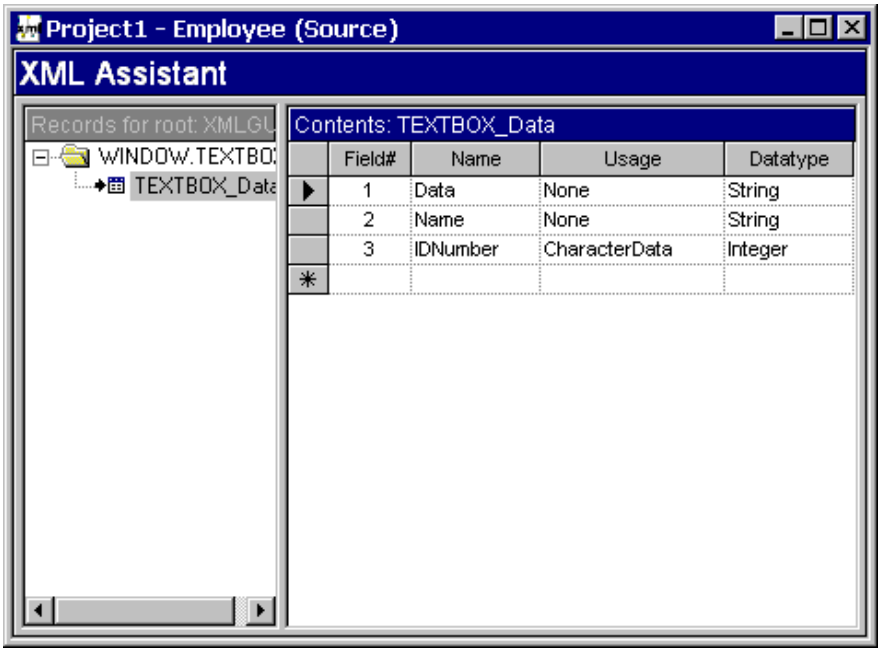


Figure 152: Record Selected in XML Assistant

After adding all the fields to the record and arranging the field order, click on the record name in the XML Assistant.

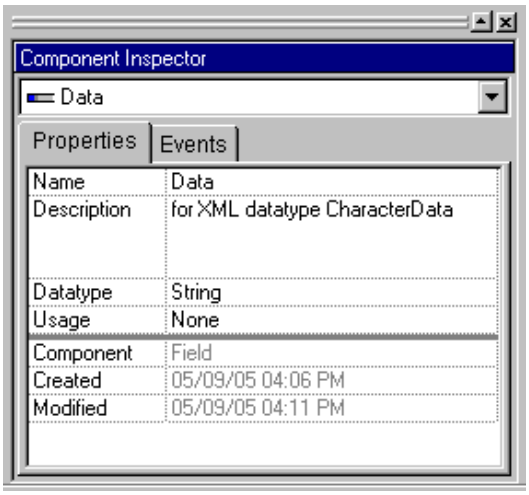


Figure 153: Record Properties in Component Inspector

This will cause the record's properties to be displayed in the Component Inspector. For XML data sources, a record ID field needs to be defined for each record. If the record was defined via an import from a representative XML document the record ID property will already be filled in with the name of the XML element that triggers this record. If the record was defined by hand, you can define a record ID by setting the IdentifierValue property in the Component Inspector. At run-time Transall will use this IdentifierValue value to recognize data from the XML file as belonging to this master record.

XML Data Destinations

SETTING UP AN XML DESTINATION

Transall interacts with XML destinations via a hierarchy of records that are setup in the XML destination. This record hierarchy in the XML destination describes the hierarchy of elements that can be written to the XML destination. A record hierarchy such as this is sometimes referred to as a Document Object Model (DOM) and in fact, the Transall data destination is DOM based.

To better illustrate how XML data destinations work we will contrast them to a more traditional Transall data destination, delimited files. For example, with a delimited file destination, you set up one record for each record type (i.e., grouping of fields) that Transall can write to the delimited file. The same is true for XML destinations, only XML files do not have records, they have a thing very much like a record called an Element. For the purpose of this discussion, records and elements are equivalent with one important difference: The elements in an XML file are in a hierarchy with one special “Root” element that is at the top of the hierarchy.

When you set up an XML destination in Transall, you set up one record for each XML element that Transall can write to the XML file. Now, with a delimited file, your LogicTree will probably perform a Map instruction to populate an output record buffer and then perform an Output instruction to write the record to the delimited file. XML files work the same way, except that XML is not written out one element at a time: The whole element hierarchy is written out all at once to form an XML file (sometimes called an XML document). Maps that target XML destination records have an extra property, “Auto Insert”.

The Auto Insert property indicates that Transall should automatically insert a row into the XML destination’s record hierarchy in memory before mapping over the data to the record. This lets a Map not only populate an XML element record but also build out the XML element data hierarchy by adding new rows to the XML destination’s cache of data. With an XML destination, you will often have many Map instructions that populate the XML hierarchy of data and one Output statement that flushes the data to an XML output file.

For example, let say we have an XML destination that has three elements, these elements are “Root”, “Parents” and “Children” and they are in a hierarchy that looks like the example below:

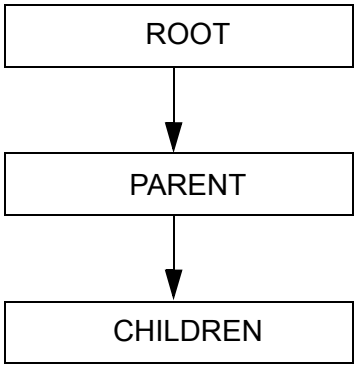


Figure 154: Example of Hierarchy

Therefore, the Root element can have many Parents elements and each Parent can have many Children elements.

Let’s also say that our actual data in the hierarchy looks like:

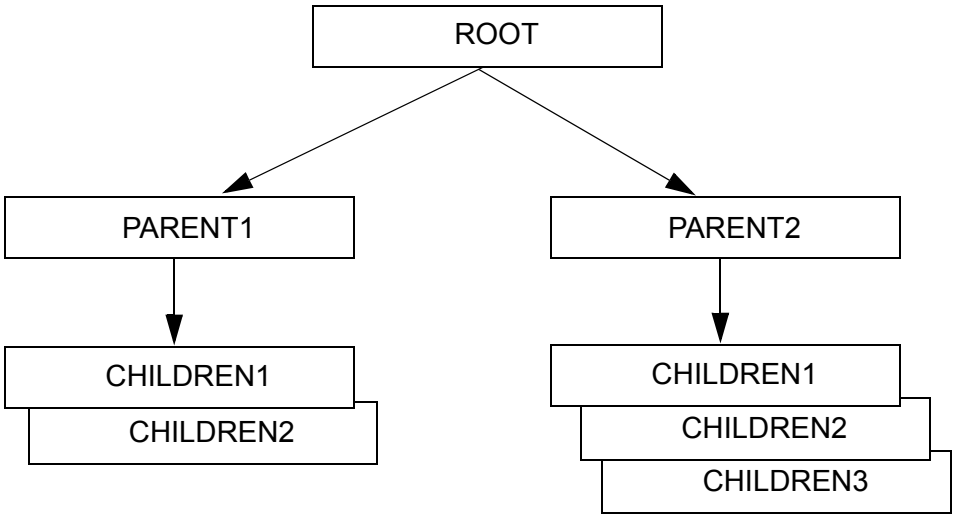


Figure 155: Example of XML Hierarchy

This hierarchy has one Root with two Parents. The first Parents row has two Children rows and the second Parents row has three Children rows. The LogicTree that moves data into the records for this XML destination will perform a Map of the first Parents element row then perform two Maps targeting the Children element rows. This sets up the first Parents and its two Children. Then the LogicTree will Map the second Parents element row and perform three Maps targeting the Children element rows. This second set of three Children element rows will be placed in the data hierarchy under the most recent Parent element row that was just inserted. Now Transall has One Root element with two Parents elements, the first Parents element has two Children elements and the second Parents element has three Children elements. Note that you do not have to Map the Root element. XML only allows one and only one row in the Root element so Transall pre-Maps this Element row for you.

After all the data has been placed in the XML data destination the destination can be instructed to write an XML document from the data via an Output statement. The Output statement will write the XML document and clear the data from the XML destination. It will not clear the DOM structure from memory, just the data in the DOM.

The Output instruction behavior for XML destinations brings another import difference between XML and other traditional data destinations to light. When Transall process an Output statement against an XML destination a whole, complete XML document is written. If the file that the XML destination is pointing to already has data in it that file is overwritten. So, XML destinations do not append to a file as each Output instruction is processed. XML destinations overwrite the output file as each Output instruction is processed. This is because the XML specification says it is illegal to have more than one XML document in an XML file.

ADDING A RECORDS TO AN XML DATA DESTINATION

To add a record type to a Transall XML data destination, you must first locate or set up a new XML data destination for the record. Once an XML data destination has been selected, open the XML Assistant by double clicking on the desired data destination in the Component Explorer. Records can be added to an XML data destination by either directly setting up the record or by importing record definitions from a representative XML document.

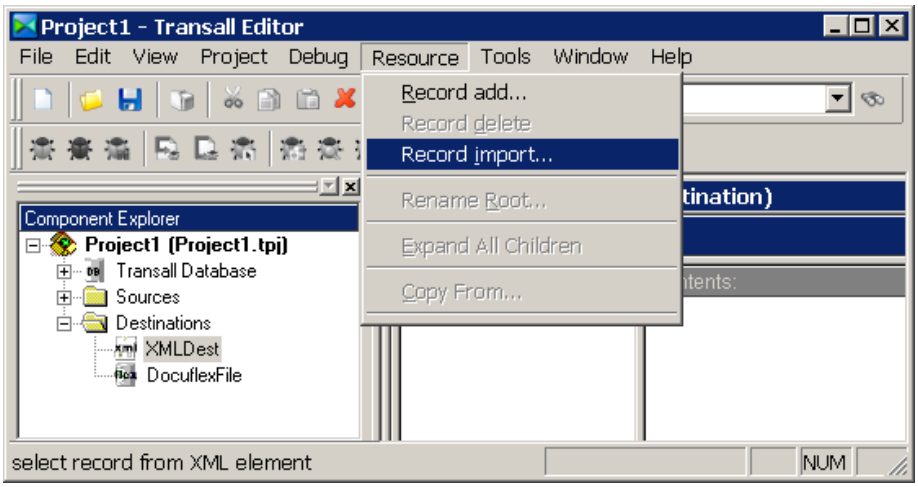


Figure 156: Resource>Record Import Menu

To import records from a representative XML document select the **Resource>Record Import** menu item.

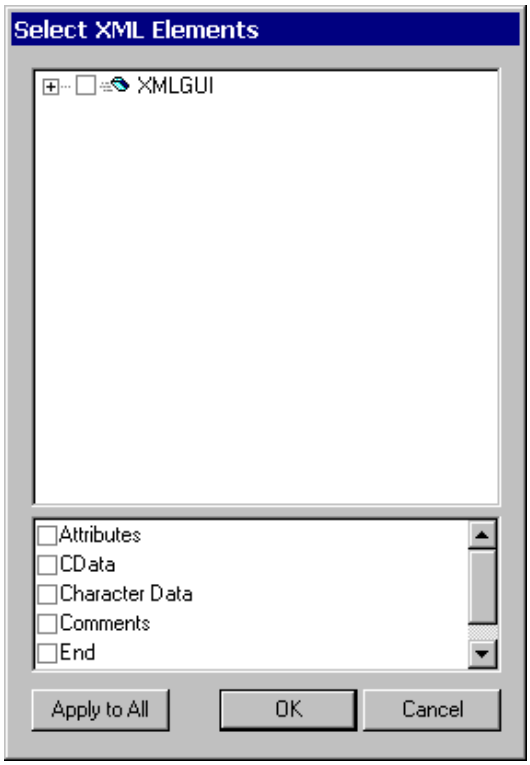


Figure 157: Select XML Elements Dialog Box

This will display a Select XML Elements dialog that is populated with a tree showing the XML element hierarchy from the XML file selected for the XML data destination.

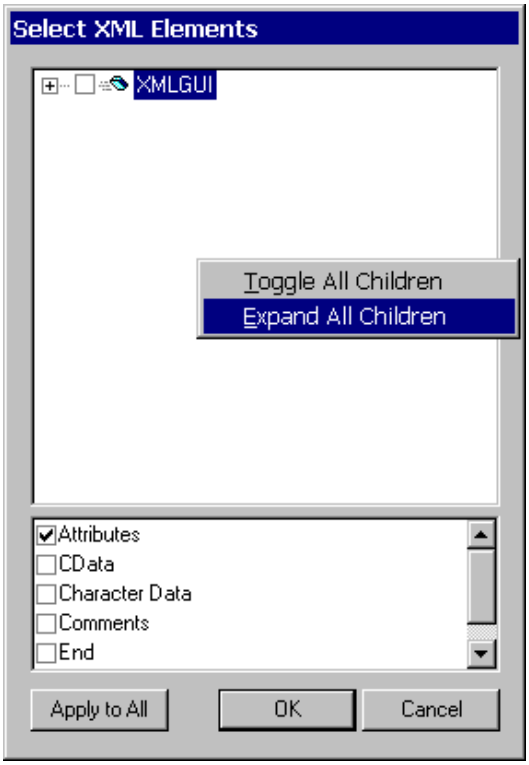


Figure 158: Select or Open Hierarchy Element Menu

Right mouse click on elements in the hierarchy to select groups of elements or open up the XML element hierarchy tree.

Note This dialog box can be made larger, by moving the mouse pointer to any dialog box edge, when the mouse pointer changes to a double headed arrow, click and hold, the left mouse button, and drag to the desired size, then release the mouse button.

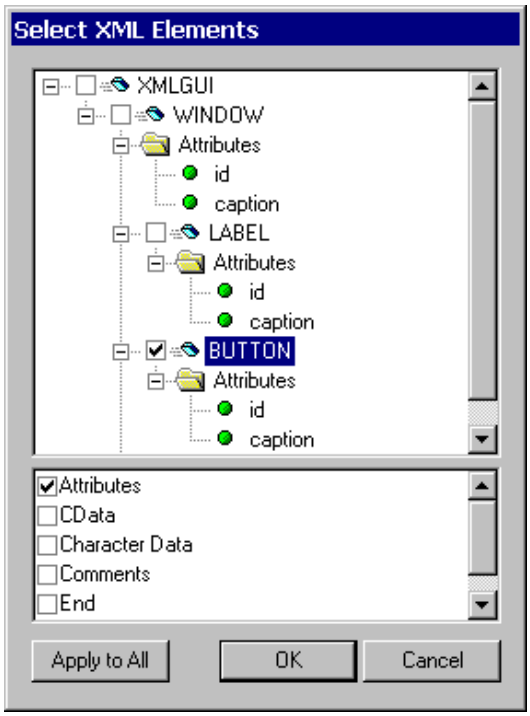


Figure 159: Hierarchy Elements Open and Selected

In the lower part of the Select XML Elements dialog select the types of data that Transall should buffer for the XML document. Transall can buffer the following data item types in an XML destination: Attributes, CData, Character Data, Comments, and Processing Instructions. For each data, item type Transall sets up a record in the XML data destination. You arrange these records in a tree that describes the hierarchy of elements that can be written to the XML destination. You can move items around in the hierarchy tree by dragging and dropping items to new locations in the tree. You can also change the record names in the Transall Component inspector to meaningful names for each record.

Note A record name cannot contain special characters or spaces but can contain underscores "_" and dashes "-".

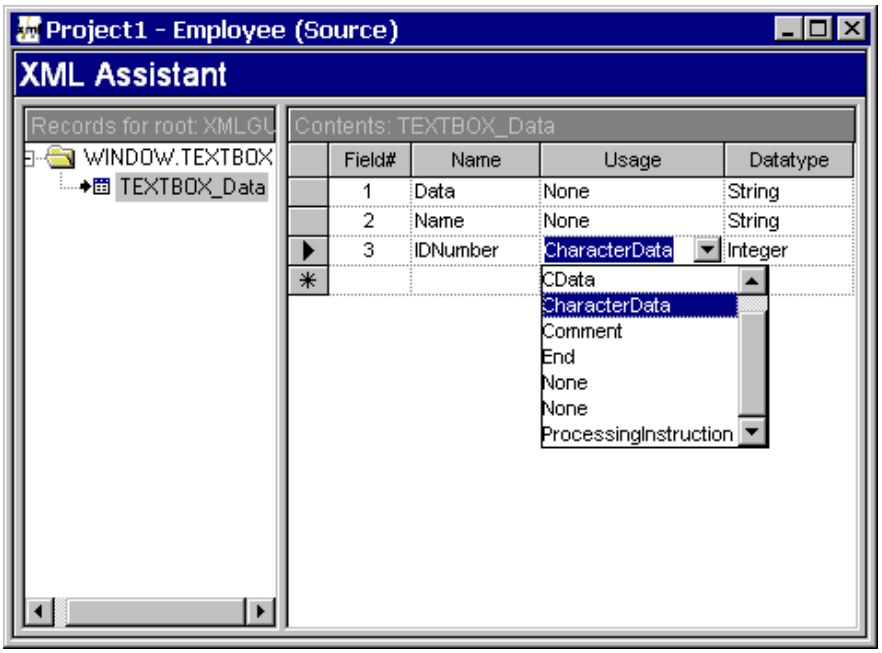


Figure 160: Adding Fields to Record

Once the records have been setup, you can add fields (or more fields) to the records. To add a field, enter the field name into the Name column of the XML Assistant on the field row with a star in the row border. Press the tab key to move to the Usage column and select a usage type for the field. Press the tab key to move to the Datatype column and select a data type for the field. Setup one field on each record for each XML Attribute that Transall is to write to the XML document.

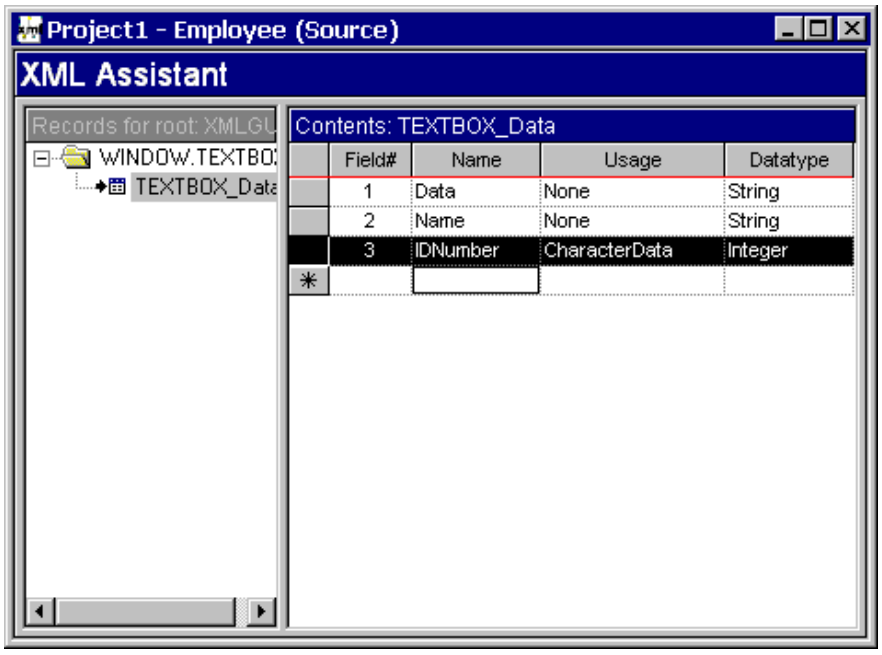


Figure 161: Moving a row in the XML Assistant

If you want to move a field up or down on the record you can do so by clicking in the gray area to the left of the field's number column in the XML Assistant. This will select the field's row in the XML Assistant. With the row selected, you can drag the field from the gray area to the left of the field's number column to a new location in the list of fields for the record. You will see a colored line in the list of record fields as you perform the drag that indicates where the field will go when you drop the field.

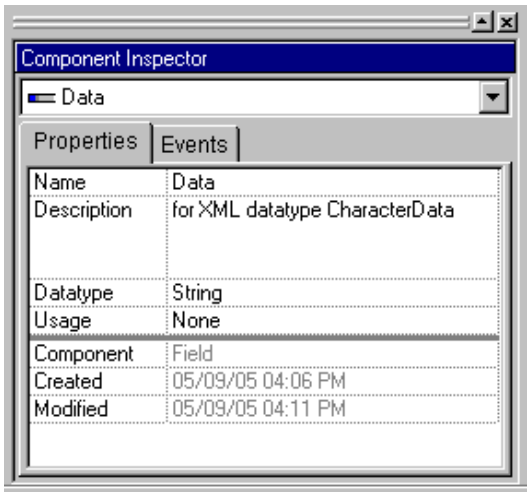


Figure 162: Record Properties in Component Inspector

After adding all the fields to the record and arranging the field order, click on the record name in the File Assistant. This will cause the record's properties to be displayed in the Component Inspector. For XML data destinations, a record ID field needs to be defined for each record. If the record was defined via an import from a representative XML document the record ID property will already be filled in with the name of the XML element that triggers this record. If the record was defined by hand, you can define a record ID by setting the IdentifierValue property in the Component Inspector. At run-time Transall will use this IdentifierValue value to send data from this record to the XML file.

Using Unicode

Both Docuflex and Transall let you process text information in Unicode. Unicode is a standard designed to allow text and symbols from the world's various writing systems to be consistently represented and manipulated. The Unicode support in Docuflex and Transall makes dealing with Unicode as seamless as possible.

Unicode support in Docuflex and Transall extends to all text information in the form of content, data, and logic control. It includes Unicode text processing by all the GUI controls that display information or allow information to be entered into the system. It also extends to all configuration files and all files in general, with support for writing Unicode values in both the names of files and in file contents.

Docuflex and Transall automatically *up-convert* non-Unicode text values to Unicode when reading them into the system, which enables them to be mixed with other Unicode text and seamlessly provides backward compatibility with legacy content and data sources.

Docuflex and Transall then automatically *down-convert* Unicode values into the desired format of output targets when writing information out of the system. This provides the significant benefits of Unicode to Docuflex and Transall users while minimizing Unicode's complexities.

Docuflex and Transall hold text information as Unicode by representing the information in memory in the Unicode Universal Character Set 2 (UCS-2LE) encoding format. When Docuflex and Transall interact with text information sources that are not already in a Unicode encoding format, the text is changed from its original format into a Unicode format as it is read into the system.

When Transall writes textual information into data files, the Unicode text values are converted into a format that is supported by the target file.

TRANSALL AND UNICODE

Transall carries all data defined as variable length strings in the same Unicode format as that used by Docuflex. However, data defined to Transall as fixed length strings is not carried as Unicode by default, but is treated as single-byte data. This lets Transall 12.1 projects that work with COBOL or undelimited data sources on single-byte structured data function as they did before support for (multi-byte) Unicode was introduced.

To further refine this behavior, support for two new string type declarations have been added. These make it possible to indicate the organization of variable or fixed length string declarations explicitly as either multi-byte Unicode or single-byte native data. These new data type declarations are **Unicode** and **Binary**.

Type	Description
Binary	You can use this data type declaration instead of String declarations when you want to carry variable length string data as a set of single byte native values (like was the case before the introduction of Unicode support). When assigning values from String data types to Binary data types, Transall assumes the code page of the target Binary string is ANSI (Windows 1252) and converts the Unicode values to single byte values as needed.
Unicode	Use the Unicode data type when you want to carry fixed length text data not as single byte values but as a fixed length string of multi-byte Unicode text. When assigning a Binary data type value to a Unicode string, Transall converts the binary data to Unicode assuming the data was from the Windows 1252 code page.

You can override the Binary to Unicode to Binary conversion behaviors using the following functions:

Function	Description
CUnicode	This function takes a Binary data type string and either an encoding or code page flag. The flag indicates the assumed source encoding or code page of the byte data when it is converted to Unicode.
CByte	This function takes a Unicode data type string and an encoding/code page flag. In this case the flag indicates the encoding or code page that the Unicode data should be converted to when it is converted to a Binary string.

You can use these values for the encoding or code page flag:

Unicode encodings	UTF-8, UTF-16, UTF-16LE, UTF-16BE, UCS-2, UCS-2LE, UCS-2BE, UCS-4, UTF-32, UCS-4LE, UTF-32LE, UCS-4BE, UTF-32BE, UTS-6
Windows code pages	W_CENTRAL_EUROPE, W_CYRILLIC, W_LATIN1, W_GREEK, W_LATIN5, W_HEBREW, W_ARABIC, W_BALTIC, W_VIETNAMESE, W_THAI, W_JAPANESE, W_KOREAN, W_S_CHINESE, W_T_CHINESE
DOS code pages	D_USLATIN, D_ARABIC1, D_GREEK, D_BALTIC, D_LATIN1, D_LATIN2, D_CYRILLIC, D_TURKISH, D_LATIN1EURO, D_PORTUGUESE, D_ICELANDIC, D_HEBREW, D_CANADIANFRENCH, D_ARABIC, D_NORDIC, D_CYRILLICRUSSIAN, D_GREEK2, D_THAI, D_ARABICASMO
ISO code pages	ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, ISO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, ISO_8859_11, ISO_8859_13, ISO_8859_14, ISO_8859_15, ISO_8859_16
Other code pages	O_KOI8R, O_KOI8U, O_KOI8RU, O_KOI8UNI, O_BIG5, O_GB12345, O_GB2312, O_JIS0201, O_JIS0208, O_JIS0212, O_JOHAB, O_KSC5601, O_KSX1001, O_WANSUNG, O_GB18030
EBCDIC code pages	E_DFXDEFAULT, E_USCANADA, E_LATIN5TURKISH, E_INTERNATIONAL, E_GREEK, E_HEBREW, E_ROECELATIN2, E_JAPANESEKATAKANA_EX, E_ARABIC, E_KOREAN_EX, E_CYRILLIC_RUSSIAN, E_LATIN1_EURO, E_CYRILLIC_S_EUROPE, E_USCANADA_EURO, E_GERMANY_EURO, E_DENMARKNORWAY_EURO, E_FINLANDSWEDEN_EURO, E_ITALY_EURO, E_SPANISH_EURO, E_UK_EURO, E_FRANCE_EURO, E_INTL_EURO, E_ICELAND_EURO

Data Sources and Destinations

Each of Transall's data sources and destinations can convert data automatically into and out of Unicode, as needed by the source or destination. This table outlines the Unicode support details for Transall.

Type of file	Description
Delimited	The File Assistant now shows extra values for the CharacterSet property. These values supply Transall with optional encoding information for data being read or written.
COBOL/Fixed/VSAM	The COBOL and fixed file sources and destinations assume all text data is byte oriented. To either place or retrieve Unicode encoded values in fields of these sources and destinations, you must use the CByte or CUnicode conversion functions For example, if you have a COBOL data source with a 10-character field of Display usage type named Field1 and this field is loaded with data from a file that contains double-byte UCS-2LE Unicode values, then the values can be decoded to Unicode and extracted from the field with an expression such as: `Target = CUnicode(Source.Field1, UCS-2LE)` You can use this expression in a Transall data map or as a standalone instruction. It tells Transall to treat the binary value in Source.Field1 as data that is encoded in the Unicode double-byte UCS-2LE format. The CUnicode function decodes the binary data into Transall's internal Unicode format and then assigns the decoded value to the Target variable as a Unicode value.
PPS	The file format used by PPS contains data Transall assumes is encoded in the Windows 1252 code page before it is converted to Transall's internal Unicode format.
PrintStream	The PrintStream data source contains data whose encoding is dictated by the print stream type. Transall automatically converts the data from the supported print stream types to Transall's internal Unicode format.
SQL data	SQL data sources and destinations now natively support Unicode through ODBC and JDBC database connectors. The legacy support for connecting to native database drivers has not been updated with support for transporting Unicode values via Transall. Instead these values are converted to and from single-byte ASCII or EBCDIC as needed by the legacy driver's APIs. The legacy native driver support continues to operate as it did before Unicode support was added to Transall.
XML	The XML source and destination already natively support transporting encoded Unicode values. Be sure, however, to select an encoding declaration such as UTF-8 for XML destinations that support transporting all possible Unicode values that Transall can write to the XML document. If you select an encoding declaration such as LATIN1, an error appears if an attempt is made to write a Unicode code point to the XML document that can not be supported by the LATIN1 encoding scheme.
Docuflex	The Docuflex data destination natively supports Unicode. Unicode values written to Docuflex data files (DDF) are encoded in UTS-6. Use either the Docuflex built-in data viewer or the DPAD tool to inspect Unicode encoded values in DDF files.
Documaker FP	The Documaker FP (Documerge VRF file) data destination assumes that all text data is byte-oriented EBCDIC. Transall automatically down-converts all Unicode values to EBCDIC for transmission in the VRF file. If Transall cannot down-convert a Unicode character value, it places a question mark (?) in the VRF file as a place holder for the character value that could not be converted to EBCDIC.

ASCII, EBCDIC and Unicode

There are no issues when you are up-converting to Unicode from legacy data sources in ASCII or EBCDIC encoding formats. The Unicode standard supports all the character points represented in these legacy encoding formats. There is, however, the possibility that Unicode encoded text may not successfully down-convert to a particular legacy encoding format such as ASCII or EBCDIC because of limitations in the physical structure of some legacy formats.

For example, Unicode data represented as UCS-2LE, or *double byte*, supports a universe of 65,536 distinct character points that are defined as the Unicode Basic Multilingual Plane (BMP). This Unicode encoding format specifies that each physical character is represented by two bytes. The ASCII and EBCDIC encoding formats specify that each physical character be represented by only one byte, meaning that the ASCII and EBCDIC encoding formats only define a universe 255 possible character points. Because of the limitations of these single-byte encoding formats, it is possible that character points (outside of English in this case) supported by Unicode can not be converted into the legacy encoding format. This is because legacy single-byte formats do not have the physical capacity to represent all the character points the Unicode encoding formats can represent.

When Docuflex and Transall encounter a problem converting from Unicode to a legacy format, a question mark (**?**) is placed in the converted results to represent the place holder of a character that could not be down-converted properly.

EDITING CONFIGURATION FILES WITH UNICODE

Use the DPAD file editing tool, included in version 12.1, to place hand-edited text into configuration files used by Docuflex and Transall, such as the INI and DDE files. DPAD displays and edits text in UCS2 mode, which lets you directly enter Unicode text, but saves the files in UTS-6 format for use by Docuflex or Transall.

In most cases you can update these files with Unicode values using the Docuflex GUI tools such as the Environment Manager. You only have to use DPAD to update these files if Unicode values need to be placed in them in ways that are unsupported by the GUI tools in Docuflex. If only standard ANSI (Windows 1252) data is in the files, then you can use any editor, such as Windows Notepad, to update the files outside of Docuflex.

To type Unicode character values into DPAD, or any of the GUI elements of Docuflex and Transall, you must enable Windows International support. This support includes a Windows feature called the Input Method Manager. This manager lets you remap the keys on a standard Windows keyboard for a particular locale. The remapping setup for a particular locale is called an Input Method Editor (IME) and it allows Unicode characters to be typed directly into all internationalized applications such as Docuflex and Transall from a standard Windows keyboard.

Transall Java Scripting Support

OVERVIEW

Transall now supports calling Java applications from Transall script in a way similar to Transall's support for calling ActiveX objects. Transall does not support Java script directly mixed in with Transall script, but it does support calling functions in Java applications from Transall script. Transall accomplishes this by loading the Java Virtual Machine (JVM) from Sun Microsystems and interacting with it through the Java Native Interface (JNI).

Transall can load a Java application to a JVM, pass and receive parameters with the application, call public subroutines in the application, and access public data members in the application.

Transall does not ship with the JVM from Sun Microsystems. This, along with the Java Run-Time Environment (JRE), can be downloaded from Sun at “http://java.sun.com/getjava”.

JAVA SUPPORT SCRIPT SYNTAX OVERVIEW

The following functions are provided by Transall to load and control Java applications from Transall script:

Function	Explanation
CreateJVM	Loads a JVM from Sun Microsystems and returns a handle to the Virtual Machine.
CreateJObject	Loads a Java application to a Virtual Machine, optionally passes parameters to the Java application's class constructor and returns a JObjectVariable handle to the Java application.
JObjectVariable.Method	This syntax using a JObjectVariable calls a method in a Java application. This syntax can also be used to retrieve a public member value from a Java application.
Set JObjectVariable = Nothing	Releases a Java application loaded on a JObjectVariable.
ReleaseJVM	Releases the JVM.
SetJavaVMOption	This syntax sets various options on the JVM, such as the default path from which to load Java Class and JAR modules.
SetJavaOption	This syntax sets various options on a Java object such as the path from which to load Java Class and JAR modules for a specific Java Object. This syntax must be used when using JAR files.

Example Script Calling a Java Application

Let's say we have a Java application that we want to call a method on. The following is the “Hello World” Java example from Sun Microsystems the Java[tm] Tutorial:

```
/**
 * The HelloWorldApp class implements an application that
 * simply displays “Hello World!” to the standard output.
 */
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println(“Hello World!”); //Display the string.
    }
}
```

We can use the following Transall script to load and call this Java application’s HelloWorld publish method:

```
Dim hJavaVM As Long
Dim JObjHellowWorld As JObject

'Load a Java Virtual Machine
hJavaVM = CreateJVM(“C:\Java\j2sdk1.4.0_01\lib”, “C:\Java\j2sdk1.4.0_01”)

’Set the Java Virtual Machine options
Set JavaVMOption(2, “C:\Java\MyClasses.jar”)

'Load a Java Application to the Virtual Machine
Set JObjHellowWorld = CreateJObject(hJavaVM, “Main-Class”)

'Call the Hello worldmain method
JobjHellowWorld.main()

'Release the Java Application
Set JObjHellowWorld = Nothing

'Release the Java Virtual Machine
ReleaseJVM(hJavaVM)
```

Java data types vs. Transall data types

Java supports a set of data types that overlap with Transall's native data types. The following table shows a mapping of Transall's primitive data types to Java's data types:

Transall Data Types	Java Data Types
String	java.lang.String
Integer	short
Long	int
Float	float
Double	double
JObject	java.util.Date

In order to use Java date-time values, a java.util.Date object must be created as a JObject in Transall. Just passing a Transall DateTime variable to a Java method expecting a java.util.Date will not work.

The following table shows a recommended mapping of Java data types back to Transall data types:

Java Data Types	Transall Data Type
java.lang.String	String
boolean (8bit unsigned)	Integer (16bit signed)
byte (8bit signed)	Integer (16bit signed)
char (16bit unsigned)	Long (32bit signed)
short (16bit signed)	Integer (16bit signed)
int (32bit signed)	Long (32bit signed)
long (64bit signed)	
float (32bit floating point)	Float (32bit Floating point)
double (64bit floating point)	Double (64bit floating point)
java.util.Date	JObject

To help convert Transall data to Java data Transall supports a “cast” syntax. Casting a value converts it from Transall's internal format to the format Java expects. For example, if I have a Transall Long variable 'lMyLong' that I want to pass to a Java Method 'MyJavaMethod' and this method expects the parameter to be a Java Long, I can cast the Transall variable in the script expression passing the value to Java, like so:

```
JavaObject.MyJavaMethod((JLong)lMyLong)
```

The '(JLong)' syntax preceding the lMyLong variable reference is the cast syntax. The cast will cause Transall to convert the lMyLong 32bit signed integer to a 64bit signed integer that the Java method MyJavaMethod expects as a parameter. Incorrect casting will result in an error thrown by the JVM. Here are some casting examples.

Java Data Type	Java Cast in Transall	Usage in Transall	Example
long	(JLong)	(JLong)Long	Dim lNumber As Long lNumber = 555-55-5555 hJava.getSSN((JLong)lNumber) or hJava.setSSN((Jlong)8)
int	(JInt)	(JInt)Integer (JInt)Long	Dim iNumber As Integer iNumber = 8 hJava.getEmployeeId((JInt)iNumber) or hJava.setEmployeeId((JInt)8)
byte	(JByte)	(JByte)String * 1	Dim sGender As String * 1 sGender=“F” hJava.gender = (JByte)sGender

Java Data Type	Java Cast in Transall	Usage in Transall	Example
char	(JChar)	(JChar)String * 1	Dim sMidInitial As String * 1 sMidInitial = "L" hJava.status = (JChar)sMidInitial
boolean	(JBool)	(JBool)Integer	DimiHasDependents As Integer iHasDependents="L" hJava.setDependentStatus((JBool)iHasDependents) or hJava.setDependentStatus((JBool)1)
short	(JShort)	(JShort)Integer	Dim iAge As Integer iAge = 29 hJava.setAge((JShort) iAge) or hJava. setAge ((JShort)29)
float	(JFloat)	(JFloat)Float	Dim fSalary As Float fSalary = 20,500.89 hJava.setSalary(fSalary) or hJava.setSalary((JFloat)20,500.89
double	NONE	Double	Dim nAvgSalary As Double nAvgSalary=20,500.812349 hJava.avgSalary(nAvgSalary)

Java Object Data Types

Transall supports non-primitive Java Platform Class data types by defining them as JObjects via the CreateJObject() function. For example, Transall allows the user to pass a string to Java by using the standard "String" Transall data type but, you would be unable to manipulate the value as a java.lang.String object. You can create a "java.lang.String" object by calling the CreateJObject() using "java.lang.String" as the sClassID. See the following example:

```
Dim hJavaVM As Long
Dim StringObj As JObject
Set JavaVMOption(2, "C:\Java\MyClasses.jar")
hJavaVM = CreateJVM("C:\Java\j2sdl1.4.0_01\lib",
"C:\Java\j2sdk1.4.0_01")

Set StringObj = CreateJObject(hJavaVM, "java.lang.String", "Init value for Java
string")

or

Dim sStringValue As String
sStringValue = "This is initial value of the Java string"
Set StringObj = CreateJObject(hJavaVM, "java.lang.String", sStringValue)
```

This technique of defining non-primitive Java data types as JObjects must be used to create Java arrays. For example:

```
Dim ZipCodeArray As JObject
Set JavaOption (2, "C:\Java\MyClasses.jar")
Set ZipCodeArray = CreateJObject(hJavaVM, "java.lang.reflect.Array", (JLong)75060,
(JLong)75061, (JLong)75094, (JLong)76006)
```

This example creates a Java array of Longs and initializes the array with the values 75060, 75061, 75094, and 76006. Here is a more complex example:

```
Dim StudentObj1 As JObject
Dim StudentObj2 As JObject
Dim StudentObj3 As JObject
Dim StudentObj4 As JObject
Dim ObjArray As JObject

Set JavaOption (2, "C:\Java\MyClasses.jar")
Set StudentObj1 = CreateJObject(hJavaVM, "classes.Student", "Wonder Woman")
Set StudentObj2 = CreateJObject(hJavaVM, "classes.Student", "Batman")
Set StudentObj3 = CreateJObject(hJavaVM, "classes.Student", "Spider-Man")
Set StudentObj4 = CreateJObject(hJavaVM, "classes.Student", "The Spawn")

Set ObjArray = CreateJObject(hJavaVM, "java.lang.reflect.Array", StudentObj1,
StudentObj2, StudentObj3, StudentObj4)
```

The preceding example creates a Java array of Java objects.

Java objects may also be used as wrappers for primitive data types. For instance, a Java object representing the "Long" primitive data type, would look like:

```
Dim LongObj As JObject
Set LongObj = CreateJObject(hJVM, "java.lang.Long", (JLong)lParam)
```

Running Java Class Applications

If the Java application you wish to control is located within a single Java class file you can call the "main" method of the main class after the Java application has been loaded to run it from Transall script. In this case the JVM must be initialized via the SetJavaVMOption function to set the classpath to the directory in which that class module is located. See the SetJavaVMOption function in *Transall Java Support Syntax Details* on page 267.

For example, if the Java class, "Enrollment" has the main method and is located in C:\Java\MyClasses directory, then the call to CreateJObject() without passing any value to the constructor would be:

```
Dim hJavaVM As Long
Set JavaVMOption(1, "C:\Java\MyClasses")
hJavaVM=CreateJVM("C:\Java\j2sdl1.4.0_01\lib")

SetJavaOption(1,"C:\Java\MyClasses")
Set hEnrollmentObj = CreateJObject(hJVM, "Enrollment")
```

Then to invoke the main method to enroll "Batman", the user would call the Main method as follows:

```
hEnrollmentObj.main()
```

This technique only works if all class files are located within a single Java .class file. If the application is located within a jar file, the user must utilize a separate calling convention.

Running Java Applications Located in JAR Files

When the Java application is located within a JAR file, you must hard-code the word "Main-Class" as the sClassID when calling CreateJObject() to load the application. Also, the JAR file must have been previously compressed using the "Main-Class" option to specify the class that contains the main method. The only object that needs to be loaded is the one that contains the main method. For example if the class "Enrollment" has the main method, then the call would look like this:

```
Dim hJavaVM As Long
Set JavaVMOption(1, "C:\Java\MyClasses")
hJavaVM=CreateJVM(("C:\Java\j2sdl1.4.0_01\lib")
SetJavaOption(2,"C:\Java\MyCMyAppclasses\classes.jar")
Set hEnrollmentObj = CreateJObject(hJavaVM, "Main-Class")
hEnrollmentObj.main()
```

The SetJavaOption function must be called to tell the JVM that this class is located within a JAR file. "Main-Class" tells the JVM to look for this option in the manifest file located within the JAR file and the subsequent call to "main" actually runs the application.

Get and set Java Object Field Values

A Java object public member field value can be retrieved via the following syntax:

```
FieldValue = hJavaObject.FieldName
```

A Java object public member field value can be set via the following syntax:

```
hJavaObject.FieldName = FieldValue
```

Transall Java Support Syntax Details

SetJavaVMOption (ByVal lOptionType As Long, ByVal sOptionValue As String) As Long

SetJavaVMOption sets various JVM startup options. For these options to be in effect SetJavaVMOption must be called prior to CreateJVM.

- lOptionType is a value indicating the type of option to set. The type codes are indicated below:

Option	Java Equivalent	Option Type Code	Description	Example
CLASSPATH	-cp -classpath	1	Sets the classpath (location of the user's classes). Several paths may be included separated by a semi-colon.	SetJavaVMOption(1,“C:\Java\classes”)
JARPATH	-jarpath	2	Sets the jarpath (location of the user's Java files in jar format). Several paths may be included separated by a semi-colon.	SetJavaVMOption(2,“C:\Java\classes\college.jar;C:\Java\MyApp\MyApp.jar”)
VERIFY	-verify	3	Specifies whether to run the byte code verifier on all loaded classes.	SetJavaVMOption(3,“TRUE”)
VERIFYREMOTE	-verifyremote	4	Runs the verifier on all code that is loaded into system via a classloader. This is the default for the interpreter.	SetJavaVMOption(4,“TRUE”)
NOVERIFY	-noverity	5	Turns verification off.	SetJavaVMOption(5,“TRUE”)
XMS	-Xms	6	Specifies how much memory is allocated for the heap when the JVM starts.	SetJavaVMOption(6, “1064”)
XMX	-Xmx	7	Specifies the maximum heap size (value in bytes, with a value greater than 1000) the Java interpreter will use for dynamically allocated objects and arrays.	SetJavaVMOption(7, “1064”)

- sOptionValue is the string value of the option to set.

CreateJVM (ByVal sJVMLibPath As String, ByVal sJREPath As String) As Long

CreateJVM loads a Java Virtual Machine (JVM) and returns a handle to it. CreateJVM will return a non-zero value for a successful JVM load and zero for failure.

- sJVMLibPath is the directory where the JVM library (jvm.lib) resides.
- sJREPath is the directory of the Java Run-Time Environment (JRE) and its accompanying folders.

SetJavaOption (ByVal lOptionType As Long, ByVal sOptionValue As String) As Long

SetJavaOption sets various Java object instantiation options. For these options to be in effect SetJavaOption must be called prior to CreateJObject.

- lOptionType is a value indicating the type of option to set. Options are:

Option	Java Equivalent	Option Type Code	Description	Example
CLASSPATH	-cp -classpath	1	Sets the classpath (location of the user's classes). Several paths may be included separated by a semi-colon.	SetJavaOption(1,"C:\Java\classes")
JARPATH	-jarpath	2	Sets the jarpath (location of the user's Java files in jar format). Several paths may be included separated by a semi-colon.	SetJavaOption(2,"C:\Java\classes \college.jar;C:\Java\MyApp\MyApp.jar")

- sOptionValue is the string value of the option to set.

CreateJObject (ByVal hJVM As Long, ByVal sClassID As String [, parameter, parameter,...]) As JObject

CreateJObject instantiates a Java object and returns a handle to the object instance.

- hJVM is a handle to a JVM returned from a call to CreateJVM.
- sClassID is the name of the Java class to load (instantiate). If the class is located within a package, then the sClassID value must be prefixed by the package name. For example, if the class "Student" was located within the package "classes", then the sClassID string should be "classes.Student".
- [, parameter, parameter, ...] are values passed to the class constructor.

ReleaseJVM (ByVal hJVM As Long)

ReleaseJVM releases the Java Virtual Machine (JVM).

- hJVM is a handle to the JVM returned from a call to CreateJVM.

Working with Maps

OVERVIEW

This chapter introduces how to use Map components to describe the origins of the data written into a data Destination's Record fields or Query columns.

Maps provide a convenient way to encapsulate the data-flow relationships and business rules that drive your Transall Application's outputs.

For each Destination component that you add to your Transall project, you should create at least one Map component. A Map component describes the origin of each piece of data that occupies a field in a data record that the Transall Application writes to a physical data destination. That is, you should create one Map for each Record subcomponent or Query subcomponent contained in each Destination component that your Transall Application uses as it runs.

Before writing physical data records to a data destination, the Transall Application must first place values into those records' fields. In many cases those values originate from fields in data records obtained from data sources whose properties are encapsulated in Source components. In other cases those values originate from fields in the rows of Tables found in the Transall Application's Transall Database or from the return values of functions coded in your project's Transall Script modules.

For more information about Tables and other components in the Transall Database, see *Working with the Transall Database* on page 279.

For more information about coding and working with Transall Scripts and Script Modules, see *Working with Transall Scripts and Script Modules* on page 339.

MAPS AND RECORDS

A Map always refers to one Record or Query subcomponent in some Destination. Thus, before you can create a Map your project must contain at least one Destination component that also contains at least one Record or Query subcomponent.

If a Destination contains more than one Record or Query subcomponent, you must create a different Map for each Record or Query that the Transall Application uses to write a physical destination data record. Each Map refers to the fields defined for a particular Record or to the columns defined for a particular Query.

If your Transall Application requires, you can create more than one Map for each Record or Query in a Destination. This allows the Transall Application to populate an output data record with data values from different sets of origins over the course of its execution.

The diagram in *Figure 163* on page 270 shows three Maps referring to two Records in a Destination.

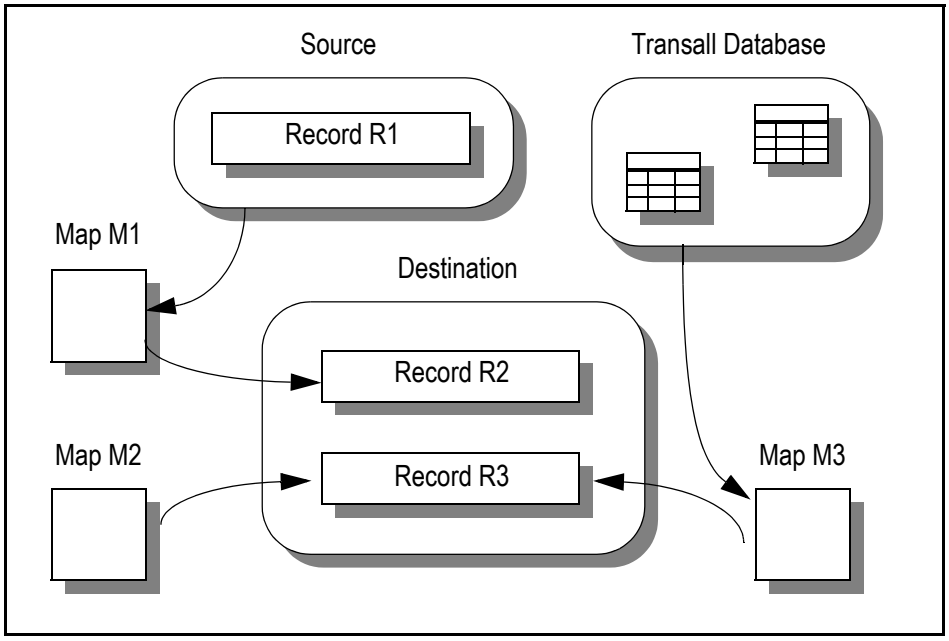


Figure 163: Relations of Maps, Records, and the Transall Database

In the diagram Map M1 refers to the Source's Record named R1 as an origin of data values to be moved into the Destination's Record named R2.

The diagram shows that Map M3 refers to a Table in the Transall Database as an origin of data values to be moved into Record R3 defined for the same Destination.

The diagram also illustrates the fact that more than one Map can refer to the same Record or Query in the same Destination. Specifically, Map M2 also refers to Record R3 in the Destination, but does not refer to any Record in any Source. This reflects another fact that a Map can refer to data values that are the product of calculations on constants or are the return values from one or more Transall Script routines. (You code Transall Script routines in your project's Script Modules, as described in *Working with Transall Scripts and Script Modules* on page 339.)

Finally, consider that the diagram presents a simplified scenario. That is, although a Map can refer to only one Record or Query in only one Destination, it can refer to more than one Record in one or more Sources as origins of the Destination Record's or Query's data values.

MULTIPLE MAPS FOR THE SAME DESTINATION

You might be required to create more than one Map that refers to the same Destination. For instance, the Destination might have more than one Record subcomponent, so you must define a distinct Map for each Destination Record.

Or, your Transall Application might use input data records referenced by different Source Record definitions to populate distinct sets of fields in the same Destination Record or Query. In this case, you define a Map for each distinct set of target fields in a given Destination Record or Query. When the Transall Application is ready to populate the actual output data record, a Logic Tree can perform in sequence each Map that describes origins of data values for the distinct sets of fields in that Destination Record.

Figure 164 on page 271, illustrates this scenario.

In the figure, Map M1 refers to fields in Source S1's Record R2 as the origin of the data values moved into two fields in Destination D1's Record R1. Map M2 refers to one field in Source S2's Record R3 as the origin of the data value moved into a distinct field in the same Destination Record.

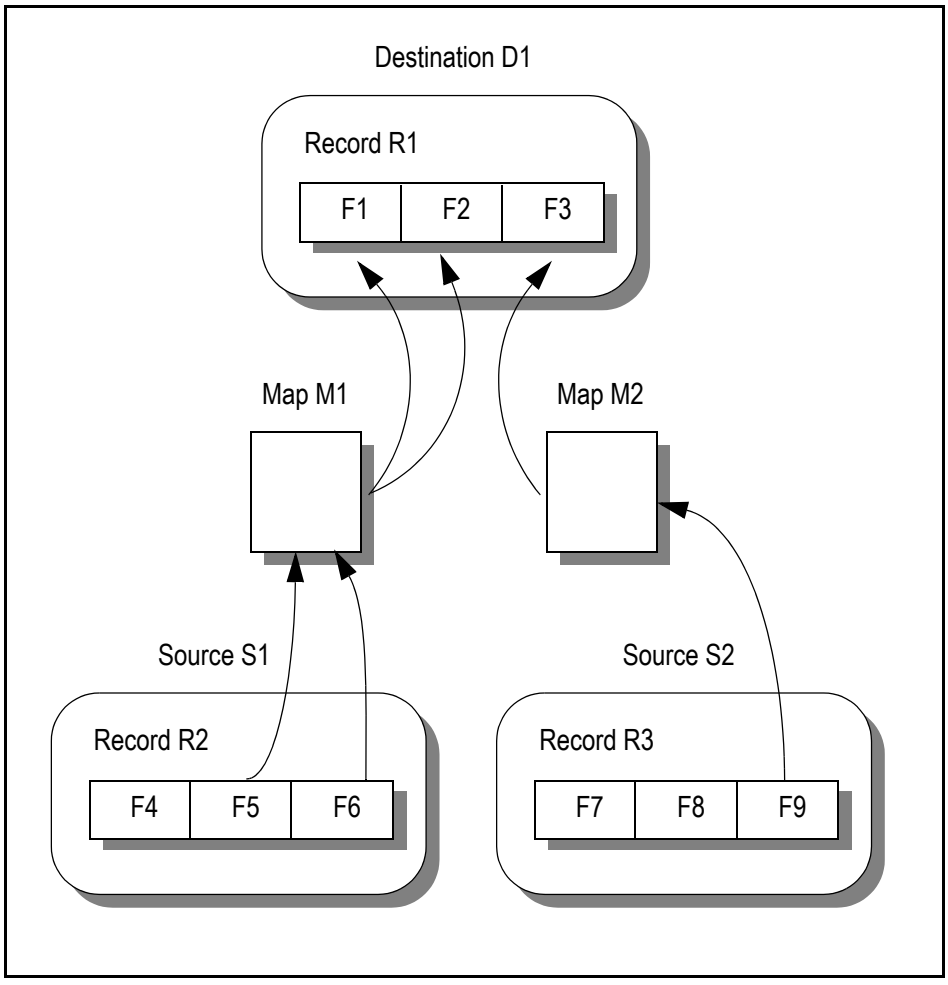


Figure 164: Using Two Maps to Populate Distinct Sets of Fields in One Destination Record

CREATING A MAP

To create a new Map, select the **Project>Add Map** menu command.

In the Add Map dialog.

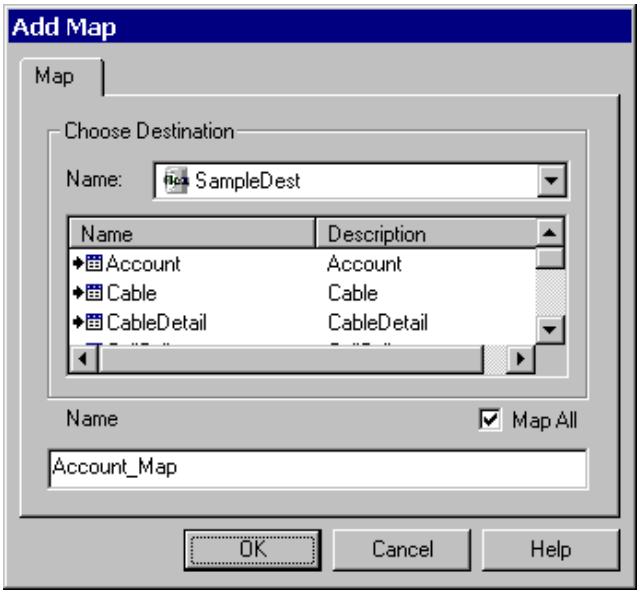


Figure 165: Add Map Dialog, for a Destination with More Than One Record Subcomponent

1. In the dialog's Choose Destination -- Name field:
   - Select a Destination by name, or
   - Select the Tables item

Tip If you want the Editor to generate automatically maps and names for all records in the chosen Destination, enable the **Map All** check box.

2. If you select a Destination, the dialog's second list shows the Records defined for that Destination.

   As shown in *Figure 165* on page 272, if more than one Record or Query name appears in the list, you must select one as the Map's "target." (For your convenience, the dialog lists all Record or Query subcomponents defined for the selected Destination.)

3. If you select the Tables item in the pop-down list, the dialog's second list shows the Tables currently defined in the project's Transall Database. Select one as the Map's "target".
4. Type a name for the new Map.
5. Click **OK** when finished.

INTERACTING WITH THE MAP ASSISTANT

After completing the Add Map dialog, the Map Assistant displays in the Transall Editor's workspace area, as shown in *Figure 166*.

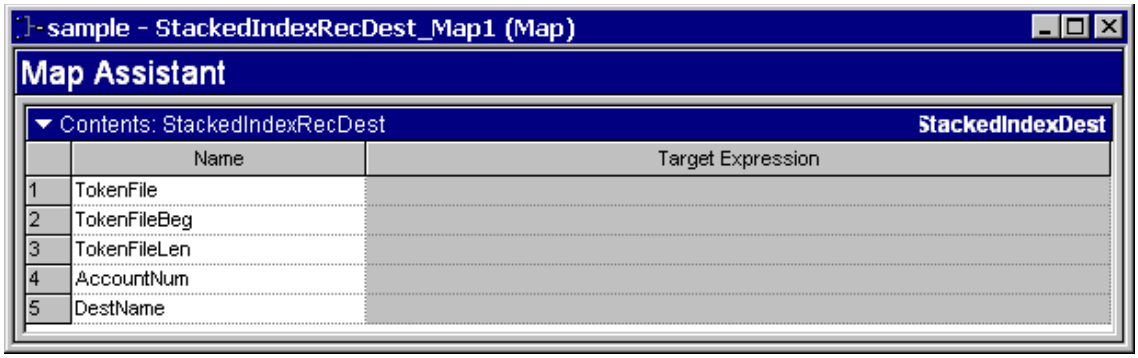


Figure 166: Map Assistant

This Assistant displays a Contents view containing a grid of two columns:

- Name column—lists the fields or columns in the selected Source Record, Source Query, or Table
- Target Expression column—lists the fields in the Map's target Destination Record or Query. If necessary, you can define an expression that represents how the Transall Application derives the field's or column's data value.

If you're describing the map to a COBOL Destination and you need to navigate to

- fields contained in a COBOL subgroup field
- COBOL fields that *redefine* another field

click , displayed on **Contents: *map_Name*** bar. Use the drop-down list to select a the necessary level of the hierarchy.

You can use the mouse to drag-and-drop the name of the Record field, Query column, or Table column from the Resource control bar (see *Using the Resource Control Bar* on page 275) to a Target Expression cell in the grid.

Figure 167 summarizes how.

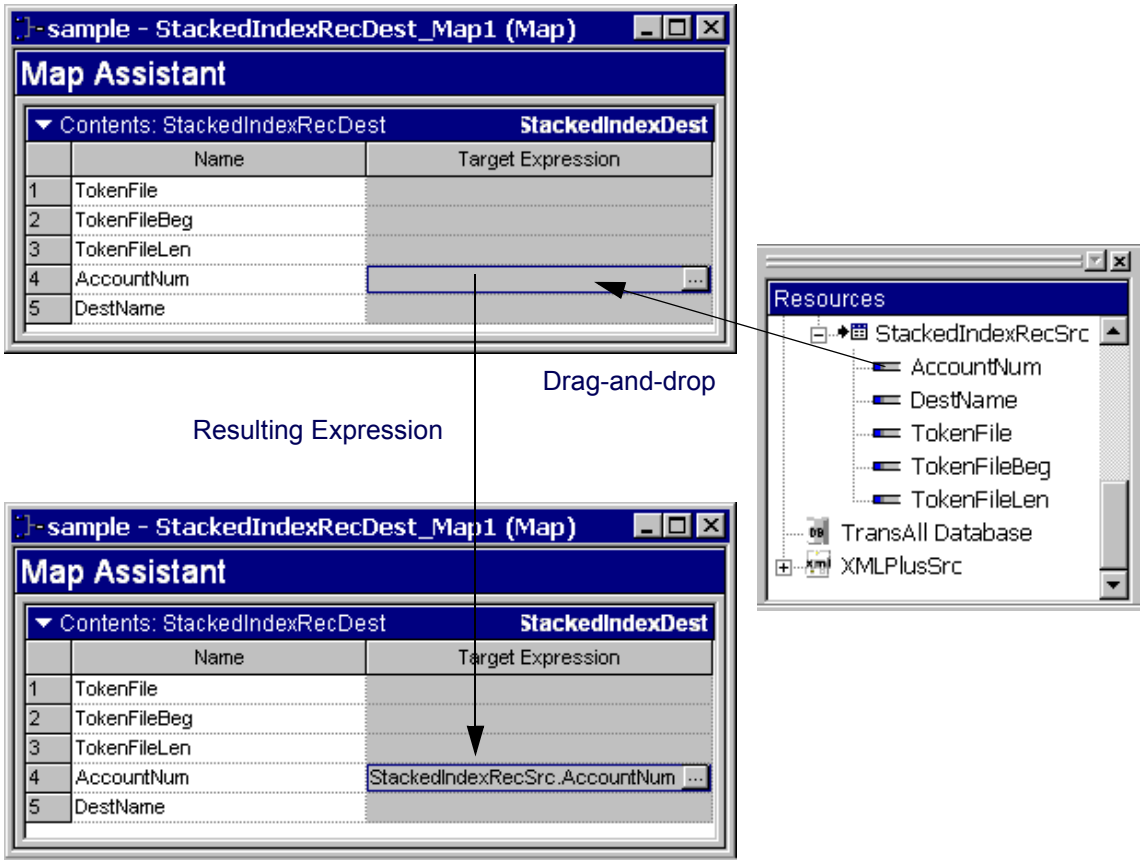


Figure 167: Populating a Destination Field Expression in a Map

You can edit the text in a Target Expression cell to form an expression. The expression describes how the Transall Application will manipulate the selected Source Record field's value, Query column's value, or Table column's value to become a new value that is stored in that Destination Record field or Query column.

When you finish describing the Map, close the Map Assistant.

Using the Resource Control Bar

The Resource Control Bar is automatically invoked and closed, by default, when a Map Assistant is opened and closed. You can change the Default behavior via **Tools>Settings** by and unchecking "Automatically show Resource control bar". Alternatively, you can toggle the Resource Control Bar via **View>Resource Bar**.

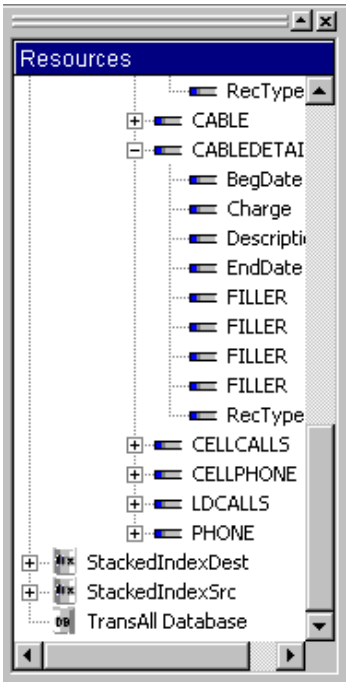


Figure 168: Resource Control Bar

The Resource control bar contains the following context menu items:

- Sorted—provides control concerning displayed resources. You can Sort the display of the tree listing in alphabetical order, or deselect Sort and display the tree in physical order.
- Flattened—provides control concerning displayed resources. You can Flatten the hierarchy of the tree listing, or deselect Flattened and display the expanded tree listing.

Note Each time you start the Transall Editor, it reverts to the default settings of Sorted=On and Flattened=Off. Toggling either of these menu items causes a repopulation of the entire control bar.

- Automap—If a Map Assistant has focus and some item in the Resource Control Bar is selected, then Automap is enabled. If you're mapping a Source record to a target record that uses a one-to-one relationship, Automap attempts to assign map expressions by matching the field names.
- Map Expression— achieves the same results as dragging-and-dropping from the Resource control bar to the Target Expression field of a record. First, select a Target Expression field in the Map Assistant; then, right-click a Resource and select **Map Expression**. The Resource is displayed in the Target Expression column.

INTERACTING WITH THE EXPRESSION BUILDER DIALOG

By clicking in a Target Expression cell, the Browse button appears as a small box with three dots in it, as shown in *Figure 169*.

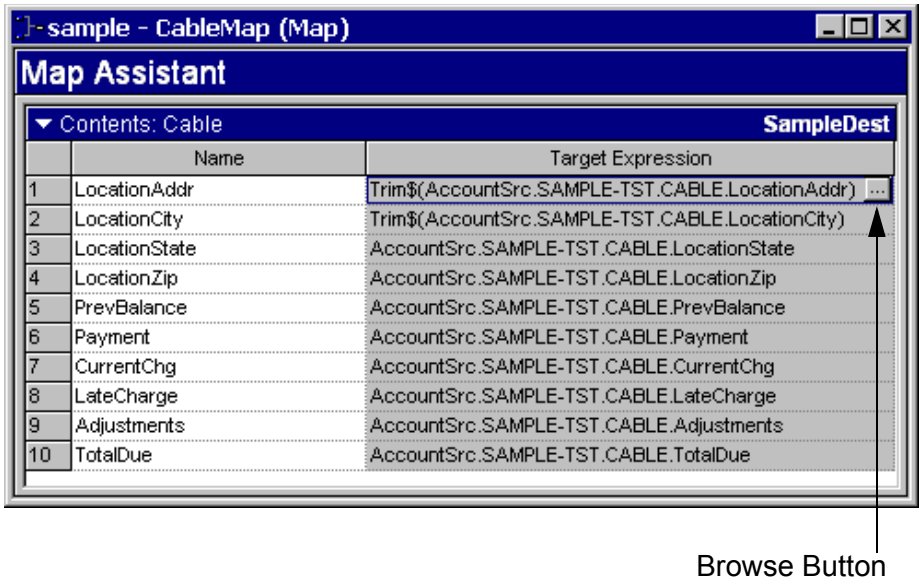


Figure 169: Map Assistant with Browse Button displayed

Clicking the Browse button will open the Expression Builder dialog, as shown in *Figure 170* on page 277.

Enhanced Expression Builder

The Transall Expression Builder has been redesigned and enhanced to be more usable. This same Expression Builder is also used in other Oracle products.

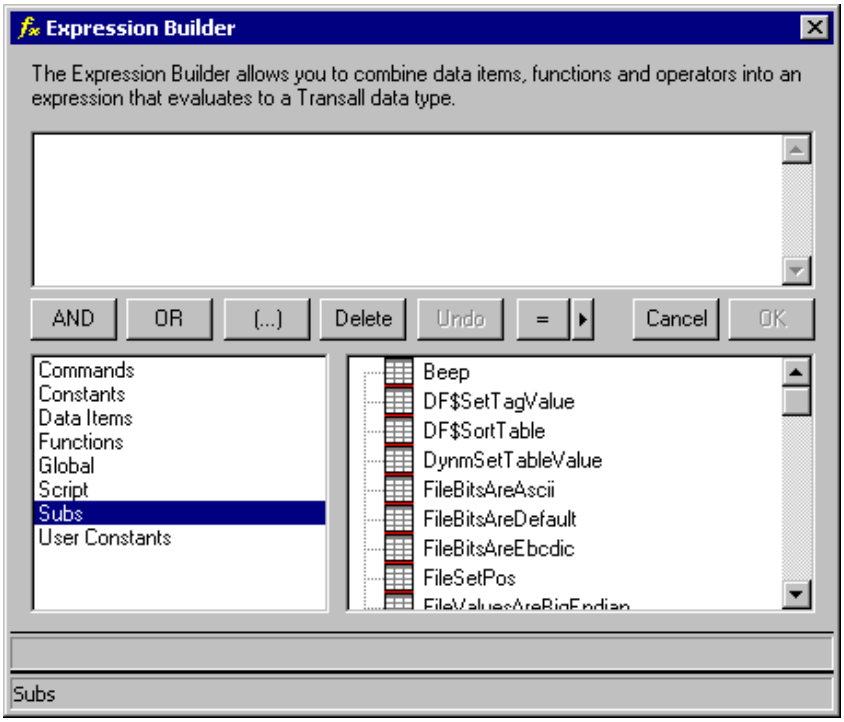


Figure 170: Transall Editor's Expression Builder Dialog

In the large text field at the top, you can compose an expression of any complexity in the Transall Script language. The expression specifies how to compute the value that the Transall Application places in the selected Destination Record field or Query column.

You can quickly and conveniently construct any expression you wish, because the dialog allows you to select from tree-structured lists of all Transall Script built-in functions and from lists of all Transall Script routines already defined in this project.

Hint After you expand a node in the tree, select a function or routine name, then drag and drop it in the large text field. Selecting a built-in function name causes a concise description of its syntax to appear at the bottom of the dialog.

Press OK when you finish composing the expression.

If you opened the Expression Builder while editing a Script in a Script Module component, the resulting expression or statement is inserted into the Script at the cursor position. Also, the Script editor corrects the capitalization of any Transall Script verbs, built-in functions, and other keywords that are inserted. However, misspellings of Transall Script verb, built-in functions, and other keywords are *not* corrected.

If you opened the Expression Builder while editing a component property, such as the Condition property of a DoWhile instruction in a Logic Tree, capitalization and misspellings of Transall Script verbs, built-in functions, and other keywords in the resulting expression are *not* corrected when the resulting expression or statement is stored in the property.

PERFORMING A MAP

The Transall Application performs a Map when it encounters a Map instruction in a Logic Tree. A Logic Tree's Map instruction directs the Transall Application to perform the data movements and calculations that are contained in the specified Map component.

The Transall Application can also perform a Map when the Map's Execute method is called from a Transall Script routine. A Map's Execute method is not accessible for editing among the component's built-in methods listed in the Component Inspector.

For more information, see *Working with Logic Trees* on page 287 and*Working with Transall Scripts and Script Modules* on page 339.

Working with the Transall Database

OVERVIEW

This chapter explains the purpose of a Transall project's Transall Database components—Tables and Sets—and how to create them. Transall has an internal database that allows you to store data temporarily, for the duration of the project's execution. To use the internal database, you define Tables and, optionally, Set relationships that mirror your storage requirements.

To perform its work, a Transall Application might need to create and work with data objects that do not change while the application runs or that contain intermediate results. To meet these kinds of needs, the Transall Application can create and work with components in its **Transall Database**. These components are called Tables and Sets.

For example, assume that part of your Transall Application's work includes using a table of values to look up information that determines which data to provide to a Destination. It might be more convenient (and efficient) for the Transall Application to load that table of values into its memory *once* and to remember it for the duration of the application's lifespan, rather than to reread those values from the same Source every time a lookup operation must take place as Transall is processing.

Each independent object in the Transall Database that contains data is called a **Table**. Each Table contains descriptions of one or more columns.

The Transall Database can also contain definitions of "parent-child" relations between the respective rows in a pair of Tables. Each of these definitions is called a **Set**.

TABLES

As its name implies, a Table contains data that is organized as rows and columns. Each data value occupies a *cell* in the Table. A Table contains intermediate data for the Transall Application to use and each Table is visible to all routines in the Transall Application.

You define a Table as a set of columns. If needed, you can define a Table with only one column or even no columns.

Note	A Table's cells contain data that conforms to the Transall Script data types only.

OPERATIONS ON TABLES

Your Transall Application *populates* a Table by adding one or more rows to it. Each added row can contain one data value for each of the Table's columns.

After a Table contains at least one row, a Transall Application can read the Table, beginning with its first row, as follows:

- Via a Walk instruction in a Logic Tree
- Using the verbs GetRow, InsertRow, UpdateRow, FindTblRow, FindSetRow, GetColSumSet, GetColSumTbl, GetRowCountSet, GetRowCountTbl, LookUpValSet, LookUpValTbl, and DeleteRow in a Transall Script routine

Resource Considerations for Tables

Your Transall Application should use Tables to store a limited set of information, as follows:

- Static data, such as a lookup table, that is loaded only once during the start-up of the Transall Application's run-time
- One logical data entity, such as the data that represents a policy or part, for any Source or Destination that is in use

As the Transall Application runs, the memory resource that its Tables require is limited only by the resources available to the machine running the Transall application.

ADDING A TABLE

After clicking on the Contents tab in the Transall Database Assistant, add a Table to the Transall Database by selecting the **Database>Add>Table** pull-down menu command. Alternatively, you can right-click on the window's background and select the **Add>Table** popup menu command.

In the Add Table dialog that displays next, enter a name for the new Table, select the Standard icon, and press OK.

Select the Standard icon to define the Table's columns yourself, using the same spreadsheet-style interface you have seen for Records in a Source or Destination.

Adding Columns to a Standard Table

Tip You can use the Copy From menu option to populate table contents from other tables, records, and queries.

A column describes the attributes of a piece of data, as does a field in a Record subcomponent of a Source or Destination.

In the Transall Database Assistant, highlight the new Table's name to display its contents in the right panel.

To add a column to the Table, click in the first cell in the Name column and enter a name. Press Tab to move to the next cell.

To complete the column definition, select an Access property value and data type property value for the column.

Figure 171 shows a Table's contents as you add a new column definition.

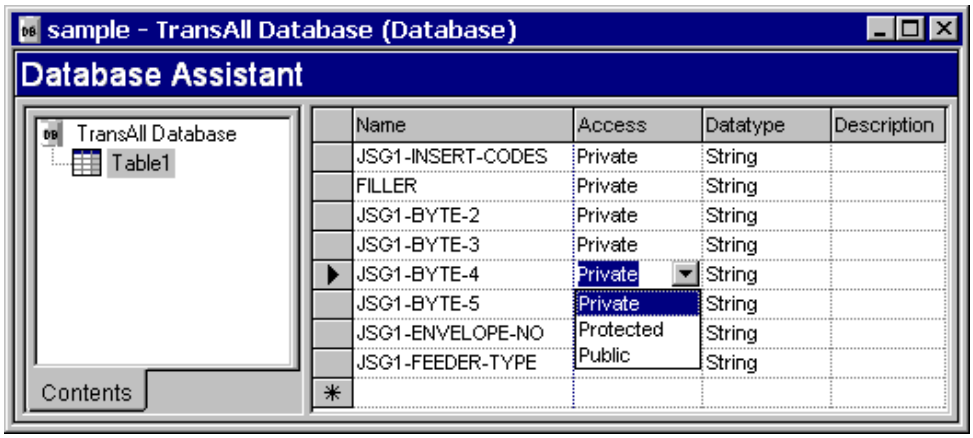


Figure 171: Database Assistant While Adding a Column Definition

After you finish defining the Table's columns, close the Database Assistant.

SETS

A Set describes a "parent-child" relationship between the rows in a pair of Transall Database Tables. A parent-child relationship means that each row in the "parent" Table is related to certain rows in the "child" Table.

For example, given two Tables, assume that the data in Table P describes a set of insurance policies written by the same agent, and the data in Table E describes a set of endorsements for those policies. Each row in Table P relates to none, one, or more than one row in Table E. This means that Table P has a parent-child relationship to Table E.

A Table can participate in more than one Set. The same Table can be both a parent and a child to other Tables.

For the details about how a Set relates the rows of two Tables, see *How a Set Establishes Relations between Rows* on page 282 later in this chapter.

To begin creating Tables and Sets in your Transall project, double-click on the Transall Database node in the Transall Editor's Component Explorer.

This opens the Transall Database Assistant, shown in *Figure 172*.

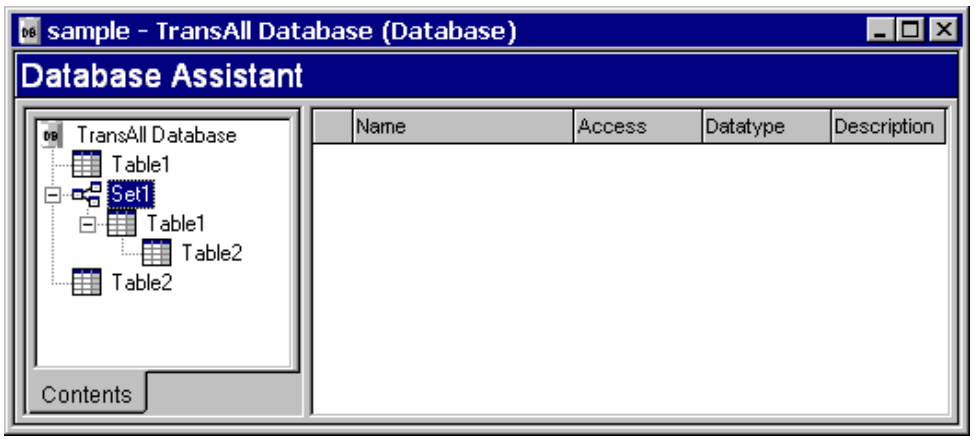


Figure 172: Transall Database Assistant

The Transall Database Assistant presents a tab on the left panel, labeled Contents:

- Clicking on the Contents tab presents the names of existing Tables and Sets. After clicking here, you can create a Table or Set.

ADDING A SET

After clicking on the Contents tab in the Transall Database Assistant, add a Set to the Transall Database by selecting the **Database>Add>Set** pull-down menu command. Alternatively, you can right-click on the window's background and select the **Add>Set** popup menu command.

In the Add Set dialog, enter a name for the Set and press OK. Set relationships are displayed in the database tree.

Complete the definition of the Set by editing its properties in the Transall Editor's Component Inspector. See *Reference for Component Properties* on page 285 in this chapter for a list of the allowable values for each property.

Tip You can assign the parent and child tables by dragging and dropping tables onto the Set nodes.

It is invalid to specify the same Transall Database Table in the Child Table and Parent Table properties of a Set. It is also invalid to name a nonexistent Table in the Child Table or Parent Table property of a Set. The Transall Editor notifies you of these discrepancies as it compiles the open project's components, after you select the **Project>Compile** or **File>Make .tex** pull-down menu command.

How a Set Establishes Relations between Rows

Often the Transall Application can use a row in one Transall Database Table to contain a copy of a data record just obtained from a Source or being prepared for output to a Destination. At other times, the Transall Application might be required to use two Tables, or more, to represent the set of data that relates to one entity.

For example, the Transall Application might be required to prepare a data record that incorporates both a “master” set of data about each employee and corresponding sets of “detail” data.

As represented in *Figure 173*, there can be more than one set of detail data for a given employee, such as the number of hours worked each week and the series of projects to which each employee has been assigned over time.

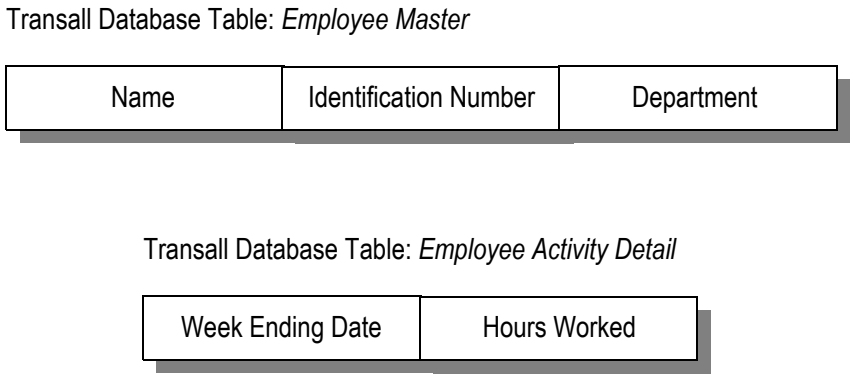


Figure 173: Master-Detail Data for an Employee

In this case:

- One Transall Database Table holds one row of master data for one employee.
- A second Table holds zero, one, or more than one rows of employee activity detail data for that employee. Each of these rows is related to the same row of master data.

Because each master (or “parent”) row can have none, one, or more than one corresponding detail (or “child”) row in one of the other Tables, a “parent-child” relation exists between the rows. One parent-child relation exists between the master data row and the rows in the second Table.

Figure 174 on page 283, depicts the relations between the row in the employee master Table and the rows in the employee activity detail Table.

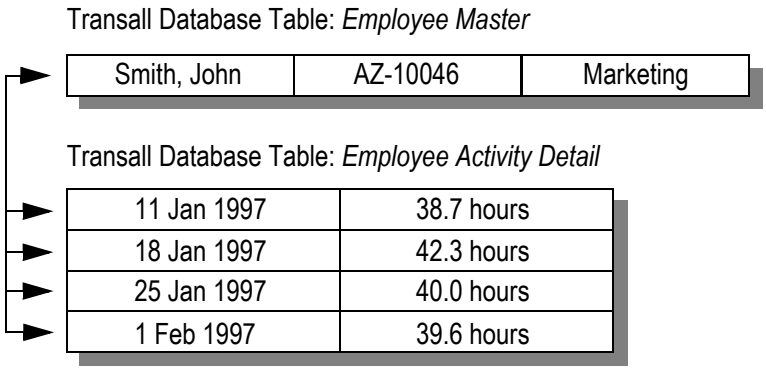


Figure 174: Related Rows for Parent Table and Child Table

Rows Become Related as They Are Inserted

As rows are added to the two Transall Database Tables that participate in a Set, the Transall Application establishes relations between the rows in the two Tables. The rows become related as follows:

1. The project contains a Set component that references the two Tables. The first Table referenced is the parent Table.
2. The Transall Application uses the Transall Script verb InsertRow to insert a new row in the parent Table. The new row is now the *current row* in that parent Table.
3. The Transall Application uses InsertRow to insert a row in the child Table. Transall causes the new row in the child Table to become related to the current row in the parent Table via the set component. The new row in the child Table is now that Table's current row. (This is significant because the child Table can also participate in another Set as parent Table.)

In this way each new row inserted in the child Table becomes related to the current row in the parent Table via the set.

Another row can be inserted in the parent Table at any time, and that new row becomes the current row in that Table.

Figure 175 depicts how new rows in the parent and child Tables become related.

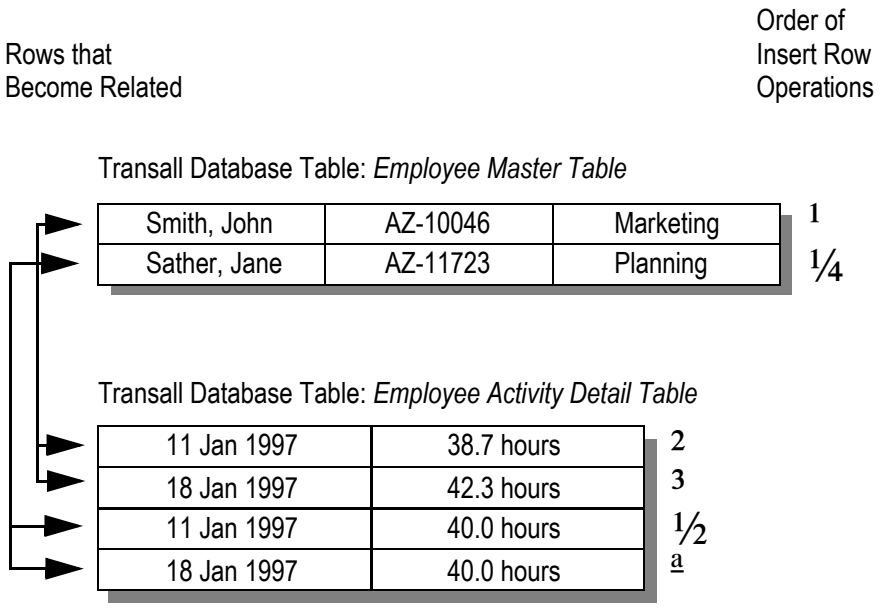


Figure 175: Connected Rows for Parent Table and Child Table

For these two Tables:

(Assume that both Tables in the Set are empty.)

1. The Transall Application adds a row to the parent Table, which becomes the current row in that Table.

2. The Transall Application adds one or more rows to the child Table. Because the two Tables are defined as a Set and because the first row in the parent Table is the current row, the next rows added to the child Table become related to that row in the parent Table.
3. The Transall Application adds another row to the parent Table, which becomes the current row in that Table.
4. The Transall Application adds one or more rows to the child Table. They become related to the second row in the parent Table.

Walking the Related Parent and Child Rows in a Set

After a Set's two Tables have been populated with rows, the Transall Application can process them in several ways.

To access the parent Table's first row and all its related rows in the child Table, the Transall Application can use the Transall Script verb Walk to obtain access to that row and all data in all rows in the child Table that are related.

For example, the Transall Application can perform this series of Transall Script statements:

```
...
walk ParentAndChildTableSet
...
```

Reference for Component Properties

The following is a list of the unique properties and their allowable values for each kind of Transall Database component.

Table Properties

Access: (Private, Protected, Public) If *Private*, the Table can be accessed only from member methods of this Transall Application; If *Protected*, the Table can be accessed from member methods and from inheriting classes of this Transall Application; If *Public*, the Table can be accessed from any method.

Alias: Optional, additional name for this Table

Set Properties

Child Table: Name of the child Table in the Set's parent-child relation.

Connection: (Automatic, Mandatory, Optional) If *Automatic*, the Set is automatically connected as rows are inserted into the member Tables; the Set can be disconnected. If *Mandatory*, the Set is automatically connected as rows are inserted into the member tables; the Set cannot be disconnected. If *Optional*, the Set is not automatically connected as rows are inserted into the member tables; the Set can be disconnected.

Delete Option: (Cascading, Manual) If *Cascading*, all child Table rows connected to a parent Table Row are deleted when the parent Row is deleted. If *Manual*, all child Table rows connected to the parent Table Row are disconnected from the set when the parent table Row is deleted.

Parent Table: Name of the parent Table in the Set's parent-child relation.

Sort Option: (Next, First, Last) If *Next*, new rows are inserted after the current row. If *First*, new rows are inserted at the beginning of the Set. If *Last*, new rows are inserted at the end of the Set.

Access: (Private, Protected, Public) Same as for Table component.

Working with Logic Trees

OVERVIEW

This chapter describes how Logic Trees encapsulate and express a Transall Application's high-level operations.

A Logic Tree is a Transall project component that contains a series of *instructions*. A Logic Tree expresses in a visual manner how the Transall Application reads, manipulates, and writes data via the project's Sources, Destinations, and Maps. A Logic Tree can also contain instructions that perform individual Transall Script verbs; this allows a Logic Tree to call another Logic Trees or any Transall Script routines.

For example, *Figure 176* shows a Logic Tree that contains a Walk instruction followed by the "nested" instructions Execute, Map, and Output.

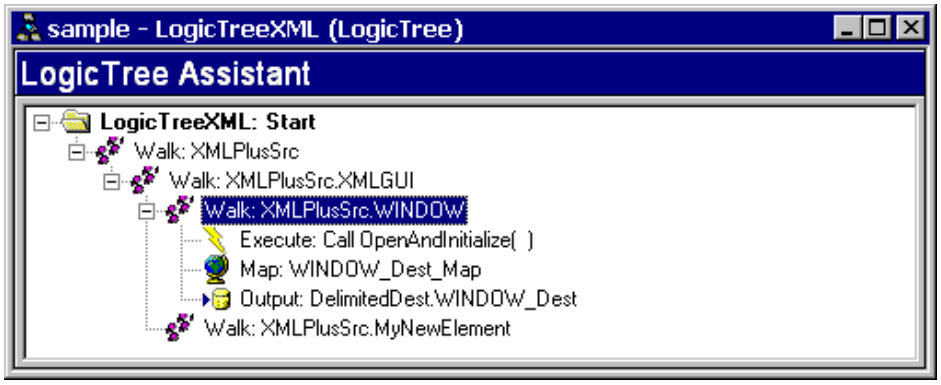


Figure 176: Simple Logic Tree

In this case, the Walk instruction directs the Transall Application to open and read all the records from a Source for reading. For each data record Read from the source the Transall Application performs, in top-to-bottom order, the instructions shown as "nodes" nested under the Walk instruction, as follows:

- The Execute instruction uses the Transall Script Call verb to invoke the Transall Script subroutine named `PrepareRecord`. Notice that the Logic Tree displays the value of this instruction's Description property, which the user has edited to form a meaningful comment.
- The Map instruction executes the named Map—that is, performs the calculations and data movements described in the Map component named WINDOW_Dest_Map to move data from Source records to Destination records.

- The Output instruction directs the Transall Application to physically write data records to the named Destination using the Destination's Record subcomponent named WINDOW_Dest.

Note A routine that is invoked via a Logic Tree's Call instruction must exist either as part of the same Transall Application or in software found outside the Transall Application but callable from it via a dynamic link library (DLL) call. In this case, `PrepareRecord` is a Transall Script subroutine found in the Script Module named ScriptModule1 in the open project.

This sample demonstrates that a significant body of data processing can be expressed in just a few Logic Tree instructions. Thus, without the knowledge and effort required for conventional programming you can construct one or more Logic Trees in your Transall project to form the "heart" of your Transall Application's activities.

For more information about Sources and Destinations, see *Working with Sources and Destinations* on page 115.

For more information about Maps, see *Working with Maps* on page 269.

For more information about the Transall Script programming language and Transall Script routines, see *Working with Transall Scripts and Script Modules* on page 339.

How a Logic Tree Works

A Logic Tree's instructions provide logic building blocks for basic input (the Input instruction), output (the Output instruction), loop on available input (the Walk instruction), loop on condition (the DoWhile instruction), data movement into Destination Record fields (the Map instruction), and branch on condition (the Condition instruction) capabilities.

The Input and Walk instructions cause the Transall Application to read one data record at a time from the specified Source into a buffer that is dedicated to that Source. The Output instruction causes the Transall Application to write one data record (subject to the specified Record subcomponent's format) to the specified Destination, using a buffer that is dedicated to that Record-Destination combination. The Transall Application initializes and manages all input and output buffers for you.

The Transall Application begins executing a Logic Tree at its first instruction. The Transall Application performs each Logic Tree instruction in sequence, until a Condition, ControlBreak, DoWhile, or Walk instruction is encountered.

After the last instruction in a Condition, ControlBreak, DoWhile, or Walk instruction's nested series is performed, the Transall Application performs the next sibling instruction for that Condition, ControlBreak, DoWhile, or Walk instruction.

After the Logic Tree's last instruction has been performed, control within the Transall Application passes to the Transall Script subroutine or function that called the Logic Tree.

ADDING A LOGIC TREE

To create a new Logic Tree, use the Transall Editor's **Project>Add Logic Tree** menu command. In the Add Logic Tree dialog, provide a name for the new Logic Tree, select the Standard type, and press OK.

As shown in *Figure 177*, the Transall Editor displays the LogicTree Assistant for the new Logic Tree and opens the Logic control bar that contains a set of icons.

Each icon represents a kind of instruction building block that you can add to the Logic Tree by dragging it into the LogicTree Assistant.

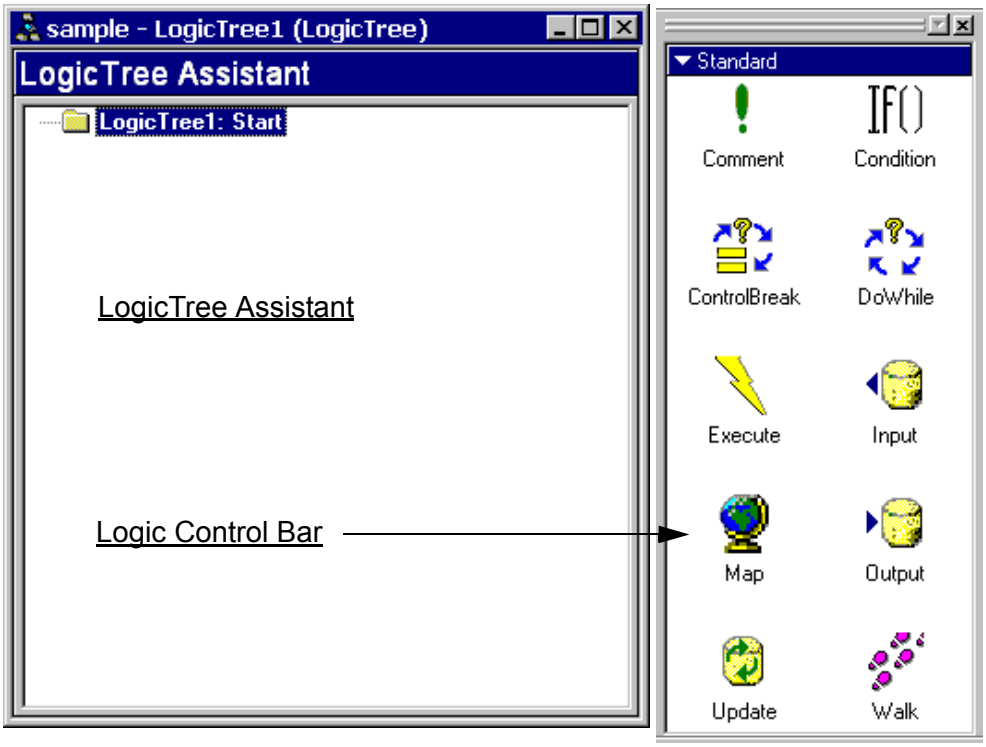


Figure 177: LogicTree Assistant and Logic Bar for New Logic Tree

Use the control in the Logic bar's top border to select from the following sets of LogicTree instructions:

- The **Standard** set of instructions includes the Comment, Condition, ControlBreak, DoWhile, Execute, Input, Map, Output, Update, and Walk instructions. Every new Logic Tree will use at least some of these instructions.
- The **Documaker FP** set of instructions includes the AddForm, AddFormsLibrary, AddTag, EndMergeSet, MergeSetBreak, SetEffectiveDate, SetRulebase, StartMergeSet, and SubmitVRF instructions. Select the Documaker FP set of instructions if you are coding this Logic Tree to produce output data records for a Documaker FP File (VRF) Destination.
- The **Documaker FP Plus** set of instructions includes the FpAddTag, FpComment, FpDataGroup, FpDataHeader, FpFooter, FpForm, FpHeader, FpKeepOnSamePage, FpLayout, and FpPageBreak instructions. Select the Documaker FP Plus set of instructions if you are coding this Logic Tree to produce output data records for a Documaker FP Plus (VRF) Destination.

Each Logic Tree instruction is described in the section entitled *Logic Tree Instructions* on page 292.

ADDING AN INSTRUCTION

To add the first new instruction, use the mouse to drag one of the instruction icons from the Logic bar to the Start tag so that the tag becomes highlighted.

Release the mouse button to insert the instruction, as shown in *Figure 178*.

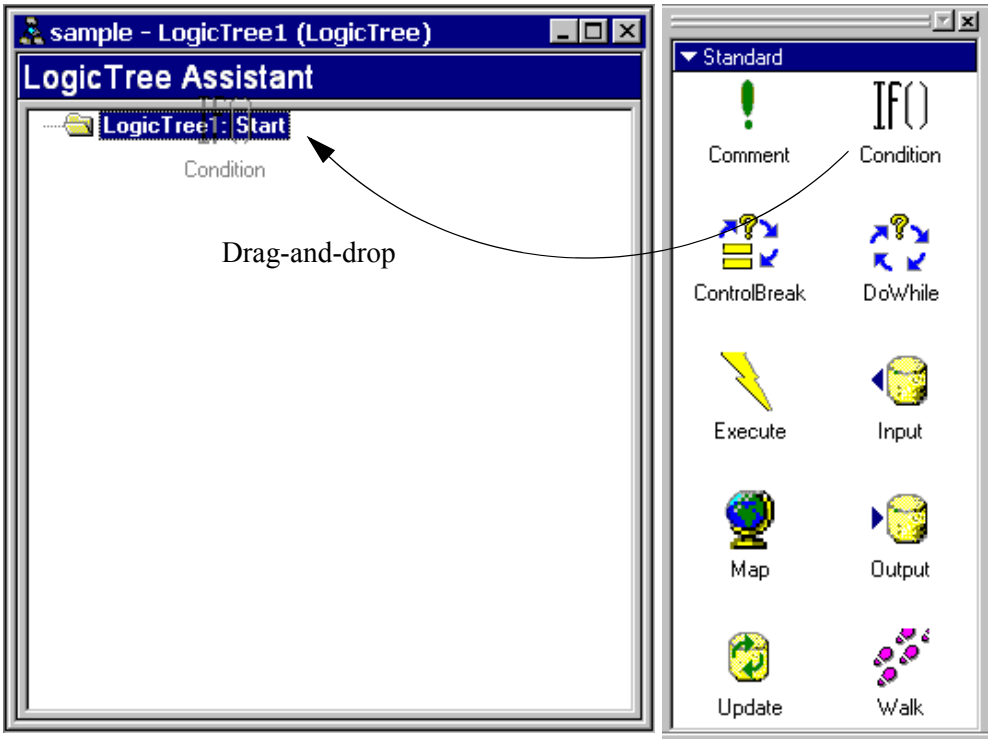


Figure 178: Dragging New Instruction into the LogicTree Assistant

When you drag and drop a new instruction, it is inserted *in a nested position* after the highlighted instruction upon which you dropped the instruction icon. After dropping the instruction, if you must adjust its order within its nested group, select **Logic Tree>Move Up** (or Move Down), or right-click on the instruction tag and select Move Up (or Move Down), or select the instruction tag and hold down the 'CTRL' key and hold down the left mouse button and drag the instruction to adjust its order.

When a Transall object is dropped on a logic tree instruction, it will either become associated with that instruction or cause a new instruction to be added as a child of the dropped-on instruction. For instance, dropping a Source file on a Walk instruction will set that instruction to Walk that Source file. Dropping the same Source file on a Condition instruction will create a Walk instruction as a child of the Condition (the default instruction is listed in bold).

- Source files: **Walk**, Input
- Destination files:* **Output**
- Destination Records: **Output**

- Maps: **Map**
- Database Tables: **Walk**, Input

Note * only destinations with a single record—destination records can also be dragged from the Resource Control Bar.

Notice that the new instruction appears in red text, to indicate that the instruction is not fully specified and therefore is *invalid*. You specify it further by editing the appropriate instruction properties in the Component Inspector or double-clicking on the instruction. For example, to complete a new Walk instruction, you must type or select the name of a Source component from which to read. (All other Walk instruction properties have default values.)

Note An invalid Logic Tree instruction prevents the Transall Editor from producing a Transall Application from the open project.

Depending on the kind of instruction, you can also specify a condition or expression that it should observe. To do so, open the Transall Editor's Expression Builder dialog. Double-click on the instruction to open the Expression Builder dialog. Filling in the Expression Builder dialog is explained in the section *Interacting with the Expression Builder Dialog* on page 276.

An instruction in a Logic Tree can have a **next sibling instruction**. This is the next instruction at the same nesting level.

Deleting an Instruction

To delete the selected instruction, select the Logic Tree > Delete command, or right-click on the instruction tag and select the Delete command.

Cloning an Instruction

You can insert a copy of the selected instruction in place. Select the Logic Tree > Clone command, or right-click on the instruction tag and select the Clone command, or select the instruction tag and hold down the 'Shift' key and hold down the left mouse button and drag to clone a copy of the instruction. The new instruction's properties have the same values as the selected instruction.

Notice that the new instruction appears in red text, to indicate that it is incomplete. You should edit the instruction's properties in the Component Inspector, completing its definition, before continuing.

Reordering Instructions

To change the order of instructions, select an instruction tag and select the Logic Tree > Move Up (or Move Down) command, or right-click on the instruction tag and select the Move Up (or Move Down) command or drag the instruction with the CTRL held down.

Enabling and Disabling Instructions

Each instruction in the Logic Tree is either *enabled* or *disabled*. An enabled instruction is ready to execute, as long as it is valid. When a disabled instruction occupies a node, it will not be executed when called.

In the Logic Tree pull-down menu, or in the pop-up menu for a highlighted instruction, a 'check mark' symbol beside the Enabled command, on the menu, indicates that the selected instruction is enabled. Select the **Logic Tree>Enabled** command, or right-click on the instruction tag and select the Enabled command, to "toggle" whether the selected instruction is enabled or not.

Logic Tree Instructions

A Logic Tree can include Standard instructions and Documaker instructions.

Standard Instructions

There are eight kinds of Standard instructions, as follows:

- **Condition**, or perform the nested series of instructions that follow based on the condition expression evaluating to "True". (not Zero)
- **ControlBreak**, or check for a "control break," which means a change in value from data record to data record for *either* a key field (property of the Record or Query subcomponent) or any of a list of Record fields or Query columns
- **DoWhile**, or perform the nested series of instructions that follow based on the "truth" of a condition expression. After performing the last nested instruction, if the condition expression is still "true" perform the same nested set of instructions again, and so on.
- **Execute**, or evaluate the specified Transall Script expression or perform the specified Transall Script statement
- **Input**, or read a data record from a Source
- **Map**, or perform the data movements described in the specified Map component
- **Output**, or write a data record to a Destination
- **Walk**, or for each data record obtained from a Source, perform the nested series of instructions that follow

You can also differentiate the Standard instructions, as follows:

- A Logic Tree's Input, Output, Walk, and Map instructions refer to data components (Sources, Destinations, Maps, and Tables) in the open project.
- The ControlBreak, Condition, and DoWhile instructions allow a nested series of instructions to be performed when a condition is met.
- The Execute instruction allows custom behavior to be expressed anywhere in the Logic Tree.

The following sections describe each Standard instruction in more detail.

CONDITION INSTRUCTION

The Condition instruction can be immediately followed by a nested series of instructions. The nested instructions are performed, in order, if the Condition instruction's condition expression produces a value that the Transall Application interprets as not equivalent to the Transall constant *False*. Conditions are *"True"* for any non zero/non null valve. Otherwise conditions are false and the Transall Application performs the Condition instruction's next sibling instruction in the logic tree.

After adding a Condition instruction, you must specify a Transall Script expression in its Condition property. You can enter this expression as follows:

- By double-clicking on the instruction node in the LogicTree Assistant to open the Transall Editor's Expression Builder.
- In the Component Inspector you can click anywhere in the Condition property row, then press the browse button to open the Transall Editor's Expression Builder.

Filling in the Expression Builder dialog is explained in the section, *Interacting with the Expression Builder Dialog* on page 258.

The Condition instruction's **condition expression** must be a valid Transall Script expression that produces a value that is either equivalent to the Transall Script constant *False or not equivalent to False*. Conditions that are not equivalent to False are true. Condition expressions often include a relational operator—greater than (>), less than (<), equals (=), greater than or equals (>=), and so on—or the onstants *True* or *False*.

Each of the following is a valid condition expression:

```
SourceRecord.Name = ""
DestinationRecord.TotalDeducts > 0.00
True
False
```

Hint A Condition instruction's condition expression can use a variable that was declared in a Transall Script statement that was invoked in an Execute instruction that has already been performed in the same Logic Tree. See the section, *Using Variables in Logic Tree Instructions* on page 304, for more information.

If a Condition instruction is *not* followed by a nested series of instructions, the Transall Application immediately performs the instruction's next sibling instruction, if it exists.

Optionally, in the Component Inspector you can type descriptive text about the Condition instruction in its Description property and this will be shown in the logic tree for better documentation.

CONTROLBREAK INSTRUCTION

The ControlBreak instruction is meaningful *only* if it is included in the nested series of instructions *under a Walk instruction on a file data source*. This is because the Walk instruction establishes a Source to read and therefore determines which Record subcomponents are candidates for driving a control break.

The ControlBreak instruction directs the Transall Application to perform the nested series of instructions that follow:

- if the value in the referenced Source Record's or Query's "identifier" field or column matches the last record read from the source, or
- if the value of any of the fields or columns in the ControlBreak instruction's BreakRecordFields property has changed from data record to data record for this record type.

For more information, see the section *Setting Up Control-Break Processing* on page 300, later in this chapter.

After adding a ControlBreak instruction, in the Component Inspector you must update these property values:

- BreakOnFirst - *Yes* (the default), to consider the referenced Source's first data record as a control break or *No*, not to consider the first data record as a control break
- BreakRecord - Name of a Record or Query subcomponent in the Source component named in the preceding Walk instruction

Optionally, in the Component Inspector you can edit the values of these properties:

- Description - Short text that describes the component
- BreakRecordField(s) - If this Logic Tree performs ControlBreak processing that is based on a change in value of a field (or fields) from data record to data record, use the Parameters dialog to specify those Source Record/Field pairs in this property. This ControlBreak instruction will be triggered when the Transall Application detects a change in value for *any* of the fields in the list you construct in the Parameters dialog.

To add or edit a value in the BreakRecordField(s) property, in the Component Inspector click on its property row and press the browse button.

This opens the Parameters dialog, as shown in *Figure 179* on page 295.

In the Parameters dialog, add a Record field name or Query column name to the BreakRecordField(s) property by clicking in its check box to the left. When finished, press OK.

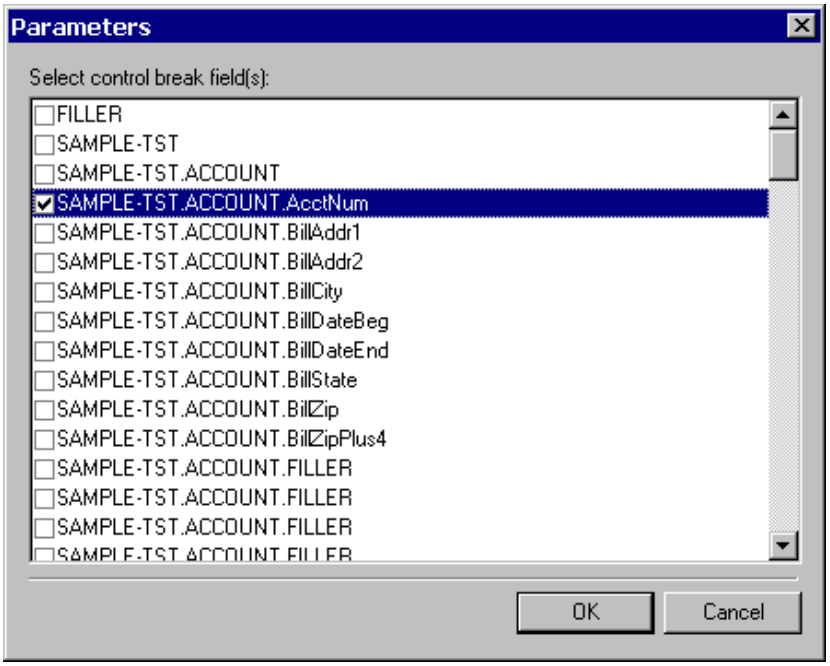


Figure 179: Parameters Dialog

Note If you specify more than one field in the BreakRecordField(s) property, control-break processing takes place when *any* of those fields' or columns' values changes from data record to data record.

Optionally, in the Component Inspector you can type descriptive text about the ControlBreak instruction in its Description property.

DoWhile Instruction

The DoWhile instruction can be immediately followed by a nested series of instructions. The nested instructions are performed, in order, if the Transall Application determines that the DoWhile instruction's condition expression produces a value that the Transall Application interprets as not equivalent to the Transall constant *False*. Otherwise, the Transall Application performs the DoWhile instruction's next sibling instruction.

After adding a DoWhile instruction, you must specify a Transall Script expression in the new instruction's Condition property. You can enter this expression as follows:

- By double-clicking on the instruction node in the LogicTree Assistant to open the Transall Editor's Expression Builder.
- In the Component Inspector you can click anywhere in the Execute property row, then press the browse button to open the Transall Editor's Expression Builder.

Filling in the Expression Builder dialog is explained in the section, *Interacting with the Expression Builder Dialog* on page 276.

Constructing a valid condition expression in the instruction's Condition property is explained in the section, *Condition Instruction* on page 293.

Optionally, in the Component Inspector you can type descriptive text about the DoWhile instruction in its Description property.

EXECUTE INSTRUCTION

The Execute instruction directs the Transall Application to evaluate a Transall Script expression or to perform a Transall Script statement.

Use the Expression Builder dialog to specify the expression or statement. To open the Expression Builder, double-click on the Execute instruction's node, or in the Component Inspector select the Execute property row and press the browse button.

Filling in the Expression Builder dialog is explained in the section, *Interacting with the Expression Builder Dialog* on page 276.

You can use an Execute instruction to call another Transall Script subroutines or functions, or to call another Logic Tree.

To call another Logic Tree, you must specify a Transall Script statement of this form:

`call` *`LogicTreeName`*`_Execute`

where *`LogicTreeName`* represents the name of the Logic Tree to be called. That is, specify the routine whose name is the name of the Logic Tree with the suffix `_Execute`. When you produce the Transall Application, for each Logic Tree in the project the Transall Editor automatically generates a Transall Script subroutine with a name of this form.

Note All Transall components generate scripts that are callable. To see the available scripts select the desired component and view the component's Events in the Component Inspector view.

Optionally, in the Component Inspector you can type in descriptive text about the Execute instruction in its Description property.

INPUT INSTRUCTION

The Input instruction directs the Transall Application to obtain the next data record from the specified Source. Specify the Source in the instruction's Source property.

Optionally, in the Component Inspector you can type in descriptive text about the Input instruction in its Description property.

MAP INSTRUCTION

The Map instruction directs the Transall Application to perform the calculations and data movements contained the specified Map component in the open project.

After adding a Map instruction, in the Component Inspector type or select the name of a Map component in the new instruction's Map property.

Optionally, in the Component Inspector you can type in descriptive text about the Map instruction in its Description property.

For more information on scripts, see *Built-In Component Methods* on page 346.

OUTPUT INSTRUCTION

The Output instruction directs the Transall Application to write data records in the specified Destination using *one* of its Record or Query subcomponents.

Note Query subcomponents could update or delete records when processing a write statement.

Because an Output instruction refers to only one Record or Query in a target Destination, it is valid (and indeed necessary) to include more than one Output instruction, each referring to a different Destination-Record or Destination-Query combination, to write data records that are based on different Record or Query subcomponents for the same target Destination.

After adding an Output instruction, in the Component Inspector type or double click on the output instruction to select a fully qualified destination Record or Query subcomponent name in the new instruction's Output property. That is, select or enter a name with this form:

DestinationName.*RecordName*
or
DestinationName.*QueryName*

where *DestinationName* is the name of a Destination component in the open project, *RecordName* is the name of a Record subcomponent in that Destination, and *QueryName* is the name of a Query subcomponent in that Destination.

Optionally, in the Component Inspector you can type in descriptive text about the Output instruction in its Description property.

WALK INSTRUCTION

The Walk instruction directs the Transall Application to perform the nested series of instructions that follow, in order, for each data record obtained from the specified Source.

After adding a Walk instruction, in the Component Inspector type or select the name of a Source component in the new instruction's Source property.

The Walk instruction encapsulates the programmatic behavior described in this pseudocode:

```
while ( [Another Data Record is Available from Source] is True )
   repeat this set of instructions {
      /* Operations performed for each record */
      ... }
```

When there are no more data records available from the specified Source, the Transall Application performs the Walk instruction's next sibling instruction.

If a Walk instruction is *not* followed by a nested series of instructions, the Transall Application reads, in turn and one at a time, each available data record from the specified Source until reaching the end of available records for the source, then immediately steps to the next sibling instruction, if it exists.

Optionally, in the Component Inspector you can type in descriptive text about the Walk instruction in its Description property.

TESTING IN THE LOGICTREE

Testing in the LogicTree enables testing breakpoints to be set directly in the Transall LogicTree GUI. Breakpoints can also be set in Transall script code. Transall will step between breakpoints in the LogicTree GUI and breakpoints in script at the same time. This enhancement helps keep the Transall developer working in the LogicTree to test business logic in Transall vs. having to test via the script code Transall generated for the business logic instructions in the LogicTree GUI. Here is a screen shot of a LogicTree with break points set in a testing session:

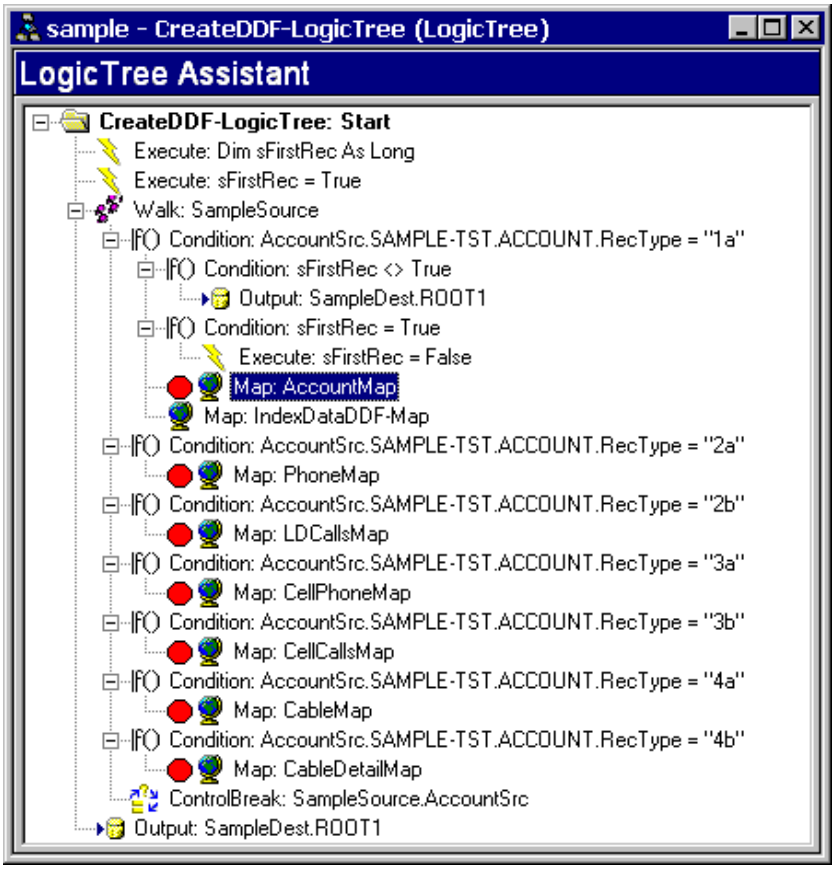


Figure 180: Sample Testing in a LogicTree

DOCUMAKER FP INSTRUCTIONS

A Logic Tree can contain instructions that perform the basic functions of a Variable Data Reformatter (VDR) program. A VDR is a custom program interface between an application and the Documaker fp product. Its purpose is to filter an application's system data into a format understood by Documaker fp.

The output of the VDR is a Variable Replacement File (VRF). The VRF is used as an input file to Documaker fp. Documaker fp then uses the VRF to produce printable Document Packages containing the application data.

For more information about the purpose of these instructions, see the description of writing a VDR program in your Documaker fp product documentation.

There are eight kinds of Documaker fp instructions, as follows:

- **AddForm**, related to VDR's AddExplicitForm
- **AddFormsLibrary**, an initialization operation that must be used before any Merge Sets are created; must be used before the SetRulebase instruction. Corresponds to calling the AddFormsLibrary routine in Documaker fp's MRGUser library.
- **AddTag**, to write a tag name and its data when the tag is not found in the Rulebase Tag Table. Corresponds to calling the AddTag routine in Documaker fp's MRGUser library.
- **EndMergeSet**, to delimit one Merge Set from another. Corresponds to calling the EndMergeSet routine in Documaker fp's MRGUser library.
- **SetEffectiveDate**, to set the effective date for forms from the EDL. Corresponds to calling the SetEffective Date routine in Documaker fp's MRGUser library.
- **SetRulebase**, to set the Rulebase name and revision level from which to obtain merging rules. This is an initialization operation that must be used before any Merge Sets are created; must be used after the AddFormsLibrary instruction. Corresponds to calling the SetRulebase routine in Documaker fp's MRGUser library.
- **StartMergeSet**, to initialize a Documaker fp Merge Set. Corresponds to calling the StartMergeSet routine in Documaker fp's MRGUser library.

STARTING APPLICATION EXECUTION IN A LOGIC TREE

When a Transall Application starts it begins processing in the Logic Tree marked as the "Primary Tree". Only one tree can be marked as Primary, see the Logic Tree properties shown in the Component Inspector.

A Logic Tree can also be called using a Transall Script statement of this form:

```
call LogicTreeName_Execute
```

where *LogicTreeName* represents the name of the Logic Tree. Specify the routine whose name is the name of the Logic Tree with the suffix `_Execute`. When you produce the Transall Application, the Transall Compiler automatically creates, includes, and compiles a Transall Script subroutine with the appropriate name for each Logic Tree in the open project.

A Logic Tree can be called by a Transall Script subroutine or function or by another Logic Tree.

SETTING UP CONTROL-BREAK PROCESSING

Control-break processing means that the Logic Tree performs additional instructions when it detects a record that matches the record ID of a record type for a data source or a change from one input data record to another data record in one or more Source Record fields or Query columns.

You can set up two kinds of control-break processing in a Logic Tree:

- *Identifier field processing* - Change in processing due to the detection of a Source Record's or Query's "identifier" field in an input data record. Set up this processing for a Source that refers to a data source from which the Transall Application can read a stream of data records whose organization varies from data record to data record. That is, the Source contains more than one Record/Query subcomponent.

The Record/Query's identifier field is named in its Identifier property. The value that identifies a given Record/Query "type" is specified in its Identifier Value property. The identifier field's value indicates to the application which Record/Query should format the current input data record. During input of data records from the data source, the Transall Application automatically compares the identifier field's value in the record to the Identified Value properties setup on the records defined to the source.

- *Break fields processing* - (For a Source Record/Query that has an "identifier" field specified) Change in processing due to a change in the value of any of one or more "break" fields in consecutive input data records of the same type. This processing can also pertain to a Source with Records/Queries that define an identifier field. The break fields are identified in a ControlBreak instruction that follows a Walk instruction.

During input of data records from the data source, the Transall Application automatically compares the values of all the break fields in the previous and current data records. When the previous and current values differ for *any* of the break fields, the Logic Tree performs the nested instructions under the ControlBreak instruction.

Each of these kinds of control-break processing requires that the Logic Tree contain a Walk or Input instruction. The Walk instruction refers to a Source that must contain at least one Record or Query subcomponent in which the Identifier property specifies the name of a field/column in that Record/Query.

EXAMPLE OF IDENTIFIER FIELD CONTROL-BREAK PROCESSING

The Logic Tree shown in *Figure 181* performs control-break processing by examining an identifier field named REC-TYPE that is defined in each of the Record subcomponents (named REC-TYPE-A and REC-TYPE-B) for the Source named UNPREDICTABLE-INPUT:

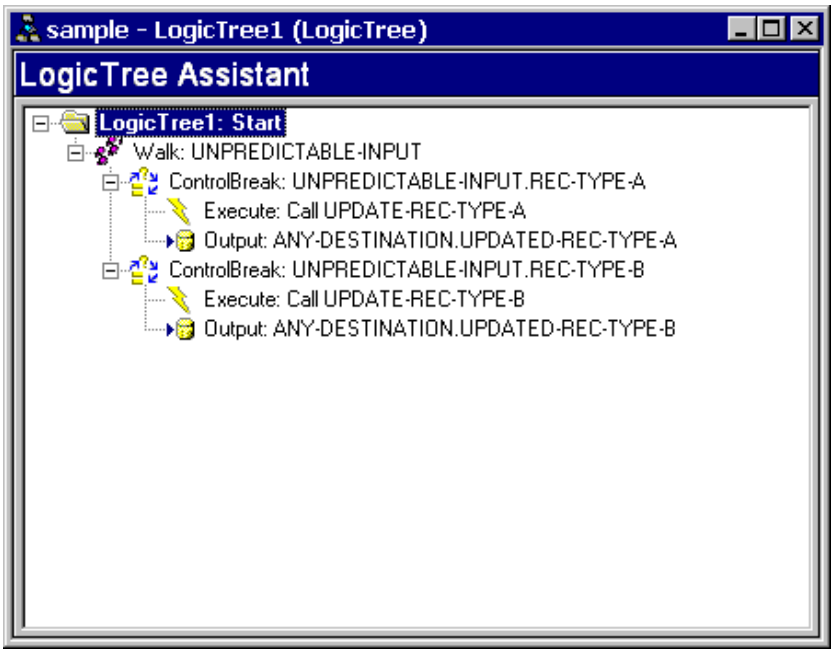


Figure 181: Logic Tree that Performs Identifier Field Control-Break Processing

The Identifier property in the REC-TYPE-A Record names the REC-TYPE field; the Identifier property in the REC-TYPE-B Record also names the REC-TYPE field.

The IdentifierValue property for Record REC-TYPE-A is “A”, and that for Record REC-TYPE-B is “B”.

The Transall Application performs this example Logic Tree as follows:

1. The Walk instruction causes the Transall Application to read input data records from the Source named UNPREDICTABLE-INPUT.
2. At the next ControlBreak instruction the Transall Application makes note of the Source Record/Query that the instruction references, then examines the portion of the input data record that corresponds to that Record’s/Query’s identifier field.
3. If the Transall Application determines that Record REC-TYPE-A’s IdentifierValue property’s value (“A”) is found in the portion of the current data record that corresponds to field named in that Record’s Identifier property (the REC-TYPE field in the REC-TYPE-A Record), the Transall Application next performs the nested series of instructions (Execute, then Output) that follows this ControlBreak instruction. If not, the Transall Application next performs this ControlBreak’s instructions next sibling instruction.

4. The next sibling instruction of the first ControlBreak instruction is another ControlBreak instruction. Because it is possible for some input data records from UNPREDICTABLE-INPUT might conform to REC-TYPE-B instead, the author also included a second ControlBreak instruction. The Transall Application performs its nested series of instructions if the current data record contains a "B" in the portion of the data record that corresponds to the REC-TYPE field. If not, the Transall Application next performs this instruction's next sibling instruction.
5. Because there is no next sibling instruction for the second ControlBreak instruction, the Transall Application loops back to the Walk instruction and reads another data record from the UNPREDICTABLE-INPUT data source.

EXAMPLE OF BREAK FIELDS CONTROL-BREAK PROCESSING

The Logic Tree shown in *Figure 181* on page 301, can be modified to allow the Transall Application to respond to a variance in certain field values in consecutive data records that have the same record type.

The modified Logic Tree is shown in *Figure 182*.

The modifications are:

- In each of the two ControlBreak instructions, the BreakRecordFields property names at least one (non-identifier) field found in that Record/Query.
- An additional Execute instruction is performed after each of the two ControlBreak instructions, to respond to the variance in the values of the break fields in the current data record versus the previous data record.

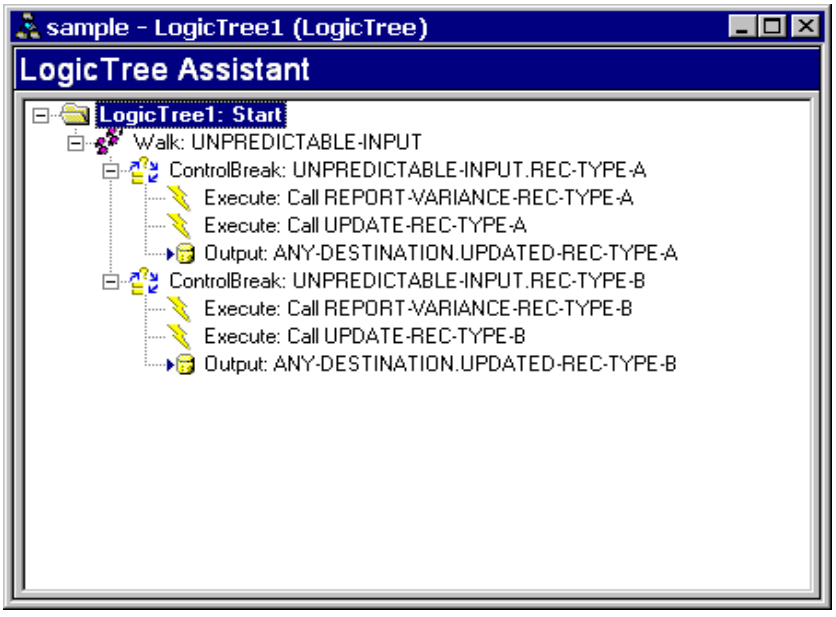


Figure 182: Logic Tree that Performs Break Fields Control-Break Processing

As in the previous example, the Identifier property in the REC-TYPE-A Record names the REC-TYPE field; the Identifier property in the REC-TYPE-B Record also names the REC-TYPE field.

As in the previous example, the IdentifierValue property for Record REC-TYPE-A is “A”, and that for Record REC-TYPE-B is “B”.

Specifying one or more (non-identifier) fields in the ControlBreak instruction’s BreakRecordFields property causes the Transall Application to perform the nested series of instructions when both of the following are true:

- The current data record’s identifier field value matches that defined for one of the Records/Queries for the Source named in the previous Walk instruction.
- The values of at least one of the field(s)/column(s) names in the ControlBreak instruction’s BreakRecordFields property differ in the current data record versus the previous data record.

This behavior allows the Logic Tree to discriminate among input data records of the same record “type” but whose field values vary from data record to record.

The Transall Application performs this example Logic Tree as follows:

1. The Walk instruction causes the Transall Application to read input data records from the Source named UNPREDICTABLE-INPUT.
2. At the next ControlBreak instruction the Transall Application makes note of the Source Record/Query that the instruction references, then examines the portion of the input data record that corresponds to that Record’s/Query’s identifier field.
3. The Transall Application determines whether Record REC-TYPE-A’s IdentifierValue property value (“A”) is found in the portion of the current data record that corresponds to field named in that Record’s Identifier property (the REC-TYPE field in the REC-TYPE-A Record). If so, the Transall Application next compares the value of each field named in the ControlBreak instruction’s BreakRecordFields property in the current data record versus the previous Type A data record. If the values for any of the named break fields differ in the current data record, the Transall Application performs the nested series of instructions (Execute, Execute, then Output) that follows this ControlBreak instruction. If not, the Transall Application next performs this ControlBreak’s instructions next sibling instruction.
4. The next sibling instruction of the first ControlBreak instruction is another ControlBreak instruction. The Transall Application processes it just as it did the first ControlBreak instruction, except that control breaks if the identifier field value is “B”.
5. Because there is no next sibling instruction for the second ControlBreak instruction, the Transall Application loops back to the Walk instruction and reads another data record from the UNPREDICTABLE-INPUT data source.

USING VARIABLES IN LOGIC TREE INSTRUCTIONS

Within the same Logic Tree a Condition or DoWhile instruction's condition expression can refer to a Transall Script variable that was declared via an Execute instruction that the Logic Tree has already performed. This allows the Logic Tree to make processing decisions based on data, such as counters, that is not strictly found in any project component.

For example, to use a counter that is initialized with a value read from a particular field in an input data record, you could code the Logic Tree shown in *Figure 183*.

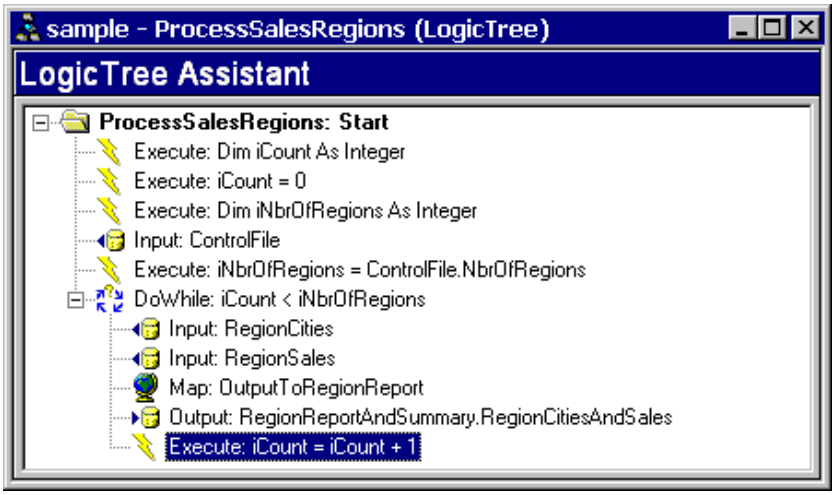


Figure 183: Logic Tree That Uses Variables

In the example:

- A Logic Tree named ProcessSalesRegions processes input data from two Sources (RegionCities and RegionSales), uses a Map to move data from the input data records to an output data record, and writes to the RegionCitiesAndSales Record of the RegionReportAndSummary Destination.
- The variable iCount, which is declared and initialized in consecutive Execute instructions at the beginning of the Logic Tree's execution, counts how many items (in this case, sales regions) have been processed.
- The variable iNbrOfRegions is declared at the beginning of the Logic Tree's execution.
- The Logic Tree's DoWhile instruction causes processing to loop based on a comparison of the values of iCount and iNbrOfRegions.
- After a number of input data records equal to iNbrOfRegions have been processed, the DoWhile instruction no longer loops. Because the DoWhile is the last instruction in the Logic Tree's first nesting level, the Logic Tree ends its execution when the DoWhile's condition expression evaluates to *False*.

Debugging and Deploying Transall Applications

OVERVIEW

This chapter describes how to use the Transall Editor to compile the open project and produce a Transall Executable. Also described is how to use the Transall Editor to debug a running Transall Executable. The chapter's final sections describe how to deploy a production-ready Transall Executable.

When you believe that your Transall project contains a sufficient combination of Sources, Destinations, Maps, Logic Trees, and (optionally) Tables and user-written Scripts, you are ready to use the Transall Editor to produce an executable Transall Application that incorporates all those components' definitions and instructions.

You can produce a Transall Application that is prepared either for debugging or for release:

- If prepared for debugging, the Transall Application contains extra information to support debugging and execution in a controlled manner in the Transall Integrated Development Environment.
- If prepared for release, the Transall Application contains minimal information to support debugging and can only be executed in its intended deployment (or "production") computing environment.

COMPILING VERSUS BUILDING

The activity of producing a Transall *Application* for debugging is called *compiling*. The activity of producing a Transall Application for release is called *building*.

Settings in the **Project>Settings** dialog determine whether the Transall Editor next produces a debug or release version of the Transall Application. This dialog also presents other settings that determine whether the Transall Editor produces other auxiliary files during its compiling or building activity. The section *Editing Build Settings* on page 306 describes how to prepare these settings.

- Using the **Project>Compile** menu command produces a debug version of the Transall Application. The section *Using the Debug Tab* on page 313, describes how to compile a debug version of the Transall Application.

- Using the **File>Make .tex** menu command produces a release version of the Transall Application. The section, *Editing Build Settings* on page 306, describes how to build a release version of the Transall Application.

FILES PRODUCED WHEN COMPILING A TRANSALL APPLICATION

When you compile or build a Transall Application, the Transall Compiler creates these files:

- *project*`.TEX` - P-code file; for execution by the Transall product's Transall Host facility (created for a successful compile)
- *project*`.TLB` - Type library file; for use only by ActiveX Automation (optionally created for a successful compile)

where *project* is the name found in the Name property of the open project's Project component.

The Transall Compiler leaves these files in the directories whose paths you specify in settings, as described in section, *Editing Build Settings* on page 306.

Always when you compile or build, the Transall Editor creates this file:

- *project*`.TSC` - Source file; contains all Transall Script source code that is used to produce the Transall Application

USING SOURCE CONTROL WITH TRANSALL FILES

Two files, *project*`.TSC` and *project*`.TPJ`, are appropriate files for version control by Transall developers.

The *project*`.TSC` file contains all the Transall Script source code that was used to create the *project*`.TEX` file.

The *project*`.TPJ` file contains all the persistent information about a Transall project in a binary form.

The `.TSC` file can be source compared to find changes in Transall applications from version to version.

EDITING BUILD SETTINGS

Two sets of **build settings** affect how the Transall Editor next produces a Transall Application:

- Debug settings
- Release settings

When the open project is saved, the current values of these settings are saved in the project's `.TPJ` file.

The Project Settings dialog contains several sections that are displayed under selectable tabs. Common to all tabs are the selectable buttons labeled *Debug* and *Release*. Use these buttons to determine whether the Transall Editor next produces a debug or release version of the Transall Application.

Use the *project* Settings command to specify several default Transall options for use when compiling and debugging your project. The Project Settings dialog and all of its properties are used to manage "project-related" information. This information is saved and reloaded with each project.

To Specify Project Settings

- Select **Project>*project* Settings**.

  The Project Settings dialog displays.

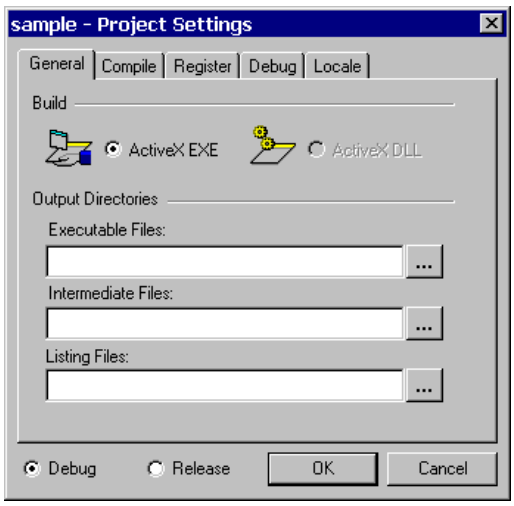


Figure 184: Project Settings Dialog Box

USING THE PROJECT SETTINGS DIALOG AND TABS

The Project Settings dialog box contains several tabs. You complete a setting by specifying general operating parameters in panels under these tabs.

To Use a Settings Tab

- Do any of the following:

To specify	Go to this topic
The type of build and directories for ancillary files	*Using the General Tab* on page 308
Compiling and debugging options	*Using the Compile Tab* on page 309
Registration settings for the Transall Application	*Using the Register Tab* on page 312
Settings for debugging the Transall Application	*Using the Debug Tab* on page 313
Various formatting symbols common to your geographic location	*Using the Locale Tab* on page 319

To Save or Close the Project Settings Dialog

- Do one of the following:

To	Do this
Apply the specifications you've provided and return to Transall	Click **OK**.
Return to Transall without applying the specifications you've provided	Click **Cancel**.

Using the General Tab

Use the first tab in the Project Settings dialog box to customize the default settings of Transall, including:

- Controlling whether the Build command produces an ActiveX executable or dynamic link library (DLL).
- Specifying the directory paths to contain various files produced during the build process.

To Display the General Panel

- If the General panel isn't already showing in the Project Settings dialog box, click on the **General** tab.

  The General panel displays.

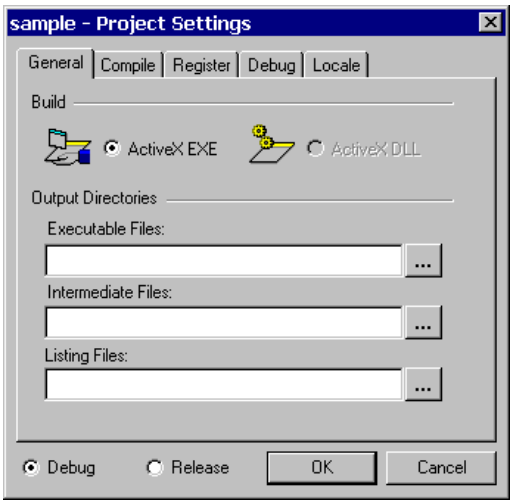


Figure 185: General Settings in the Project Settings Dialog

Because the General panel contains multiple group boxes of specifications, this guide provides a separate topic for each group.

If you need to specify the	Go to
Settings for use during the Build	*To Specify the Build Settings* on page 309
Directory paths for various files produced during the Build	*To Indicate the Supporting Directory Paths* on page 309

If you need to specify the	Go to
Saving or closing of the General tab panel	*To Save or Close the Project Settings Dialog* on page 308

To Specify the Build Settings

- Do one of the following:

If you want to	Do this:
Produce an out-of-process ActiveX executable	Click **ActiveX EXE**.
Produce an in-process ActiveX dynamic link library (DLL). (Currently, Transall only supports building a Transall Application that is an ActiveX executable.)	Click **ActiveX DLL**

To Indicate the Supporting Directory Paths

1. In the Executable Files text box, type the directory path for the Transall Application executable file, or click ... to select the folder.
2. In the Intermediate Files text box, type the directory path for other intermediate files that are by-products of the build process, or click ... to select the folder.
3. In the Listing Files text box, type the directory path for listing files that summarize the results of the build process, or click ... to select the folder.

Using the Compile Tab

Use the second tab in the Project Settings dialog box to modify how the Transall Editor produces the Transall Application's executable file and other auxiliary files.

To Display the Compile Panel

- If the Compile panel isn't already showing in the Project Settings dialog box, click on the **Compile** tab.

  The Compile panel displays.

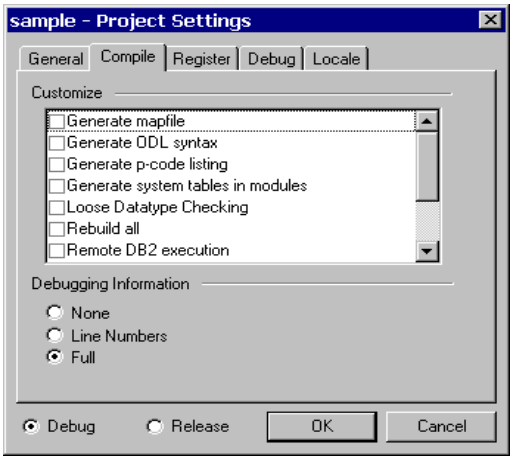


Figure 186: Compile Settings in the Project Settings Dialog

Because the Compile panel contains multiple group boxes of specifications, this guide provides a separate topic for each group.

If you need to	Go to
Customize the settings for compiling	*To Customize the Compiling Options* on page 310
Indicate the desired debugging information	*To Indicate the Level of Debugging Information* on page 311
Compile the project for debugging	*To Compile the Project* on page 311
Save or close the Compile tab panel	*To Save or Close the Project Settings Dialog* on page 308

To Customize the Compiling Options

- In the Customize group box, select as many of the following as necessary:

Option	Meaning	
Generate mapfile	whether to provide details about the internal symbols used in the Transall Application	
Generate ODL syntax	whether to support applications that call the Transall Application by means other than ActiveX calling interfaces	
Generate p-code listing	whether to produce a listing that aids in debugging a Transall Application	
Generate system tables in modules	whether to build internal system tables that describe the client tables' setup in the Transall database (e.g., SYSTABLES, SYSELEMENTS, and SYSSETS)	
	SYSTABLES	contains one row for each table in the Transall database. Each row contains one field, NAME, that contains the name of the client table
	SYSELEMENTS	contains one row for each field in the Transall database:
		TBNAME—The name of the client table
		NAME—The name of the field
		SQLTYPE—The datatype name for the field
		PRECISION—The datatype's number of digits to the left of the decimal or total length for the field
		SCALE—The datatype's number of digits to the right of the decimal for the field
		NULLFLG—Flag indicating if the field supports nulls
	SYSSETS	contains one row for each Set relationship defined in the Transall database:
		NAME—The name of the client Set
		PARENT—The name of the owning table
		CHILD—The name of the child table
		MANOPTFLG—Flag indicating if this Set is Mandatory or Optional
		ORDERBYFLG—Flag indicating the sort order
		CASCADEFLG—Flag indicating if this set will cascade deletes from the Parent table to the child table
Loose Datatype Checking	tells the compiler to generate code to convert automatically the data types of variables on-the-fly so you won't receive a "mismatched data type" compile time error	
Rebuild all	whether to rebuild the Transall Application even if it's already newer than the Transall source code from which it was built	
Remote DB2 execution	whether to build components that enable the Transall Application to access DB2	

Option	Meaning
Suppress linking	whether to create only a *project*.TOB file for the open project
Suppress startup banner	whether the Transall Application should display the Transall Compiler product's banner when it is started
Suppress typelib generation	whether to build a typelib (*.tlb) for the Transall Application
Verbose messages	whether to generate extra messages from the build process

To Indicate the Level of Debugging Information

- In the Debugging Information group box, do one of the following:

If you want to	Do this:
exclude debugging information from the Transall Application	Click **None**.
only include source code line numbers in the Transall Application	Click **Line Numbers**.
include full debugging information in the Transall Application	Click **Full**.

To Compile the Project

Compiling the open project allows you to determine whether there are certain kinds of errors and omissions in the contents of your project components.

- Select **Project>Compile** from the Transall Editor's menu bar.

  Compiling the open project means:

  a. The Transall Editor automatically generates a set of Transall Script source code routines and definitions and calls the Transall Compiler to compile the script to p-code. The code in these routines implements the instructions and data definitions contained in the open project's components.

  b. The Transall Compiler translates all Transall Script routines associated with the open project, that is, generated *and* user-written routines into a compact form, called p-code, which can be executed by the Transall Host run-time library facility.

As the Transall Editor compiles the open project's components, it displays information about its progress in its Output Control Bar. Errors found during compilation are reported using messages that display in this bar. To open the Output bar, select the **View>Output Bar** command.

Note Before the program compiling takes place, the Transall Editor checks whether the open project has ever been saved. If not, the Transall Editor opens a SaveAs dialog in which you must specify a name for the open project's .TPJ file. However, using this dialog to save the project for the first time does not cause the project's own component properties to be updated. Therefore, after the compile completes, you should manually update the project component's Name property, so that it is the same as the name of the project's .TPJ file.

Figure 187 on page 312, shows the location and contents of the Output bar after selecting the **Project>Compile** menu command.

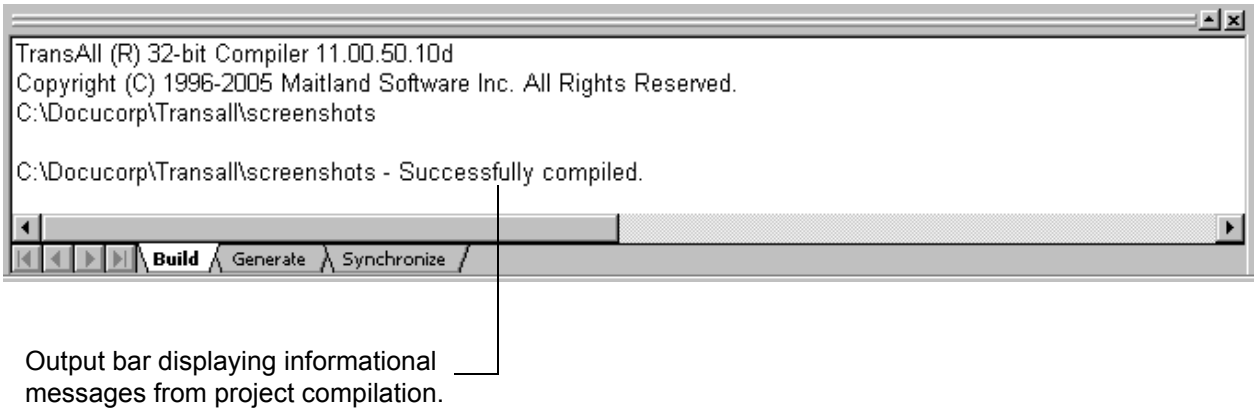


Figure 187: Output Bar in the Transall Editor

Using the Register Tab

Use the third tab in the Project Settings dialog box to specify settings that modify how the Transall Editor registers the Transall Application's executable file in the Windows Registry.

To Display the Register Panel

- If the Register panel isn't already showing in the Project Settings dialog box, click on the **Register** tab.

  The Register panel displays.

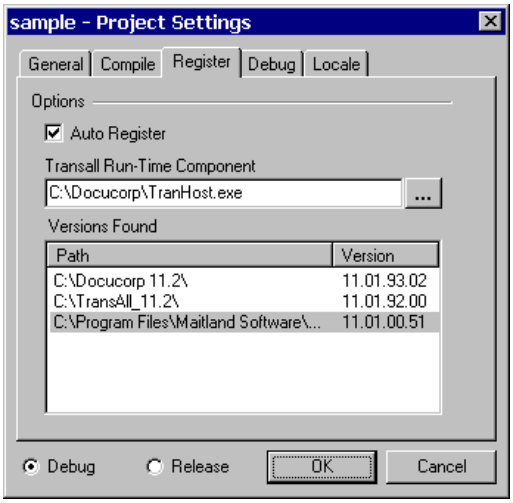


Figure 188: Register Settings in the Project Settings Dialog

Because the Register panel contains multiple group boxes of specifications, this guide provides a separate topic for each group.

If you need to	Go to
Automatically register the Transall Application as an ActiveX server	*To Register Automatically the Transall Application* on page 313
Indicate the desired version of the Transall Host to use	*To Indicate the Transall Host* on page 313

If you need to	Go to
Save or close the Register tab panel	*To Save or Close the Project Settings Dialog* on page 308

To Register Automatically the Transall Application

- Enable the Auto Register check box if you want the Transall Editor to register the resulting Transall Application's *project*.TEX file as an ActiveX server on your workstation.

Tip When building a **debug** version of the Transall Application, the Auto Register check box is *unchecked* by default. In order to test and debug your Transall Application as an ActiveX server on your own workstation, you must register it on your workstation and update your workstation's Registry as the Transall Application changes.

When building a **release** version of the Transall Application, the Auto Register check box is also *unchecked* by default. This state reflects the expectation that the Transall Application is not necessarily being deployed on your workstation and therefore probably must be registered on another system.

To Indicate the Transall Host

- In the Transall Run-Time Component text box, type the directory path of the Transall Host executable file (tranhost.exe) to use when running the resulting *project*.TEX file, or click ... to select the folder.

  -or-

  Select a Transall Host version by double-clicking an item in the Versions Found list box.

  When you open the Project Settings dialog, the Transall Editor automatically populates the Versions Found list after searching the directories listed in the PATH environment variable for your workstation. By selecting from the Versions Found selection list, you can build a Transall Application that will be executed, for instance, by the version of Transall Host that is installed on your workstation rather than the one installed elsewhere on your network.

Using the Debug Tab

Use the fourth tab in the Project Settings dialog box to debug a debug version of a Transall Application. This action requires the Transall Editor to be already running and to have opened the Transall Application's corresponding project. After the Transall Editor is running in this manner, the Transall Application can be started for debugging by

- another ActiveX-enabled client application as an ActiveX server
- the Transall Editor in command-line mode
- the Transall Editor in offline mode
- another EXE that calls the Transall DLL API

To Display the Debug Panel

- If the Debug panel isn't already showing in the Project Settings dialog box, click on the **Debug** tab.

  The Debug panel displays.

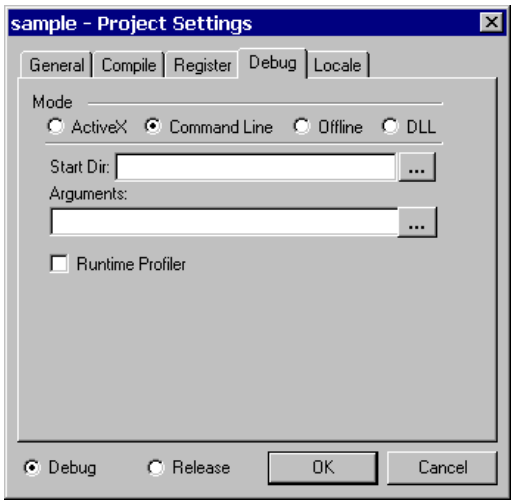


Figure 189: Debug Settings in the Project Settings Dialog

Because the Debug panel contains multiple group boxes of specifications, this guide provides a separate topic for each group.

If you need to	See this:
Debug in ActiveX mode	*To Specify ActiveX Debugging* on page 315
Debug in Command Line mode	*To Specify Command Line Debugging* on page 315
Debug in Offline mode	*Specifying Offline Debugging* on page 317
Debug in DLL mode	*To Specify Transall DLL API Debugging* on page 319
Save or close the Debug tab panel	*To Save or Close the Project Settings Dialog* on page 308

To Specify ActiveX Debugging

- Select **ActiveX** if the Transall Application will be started for debugging by the Transall Editor in ActiveX mode.

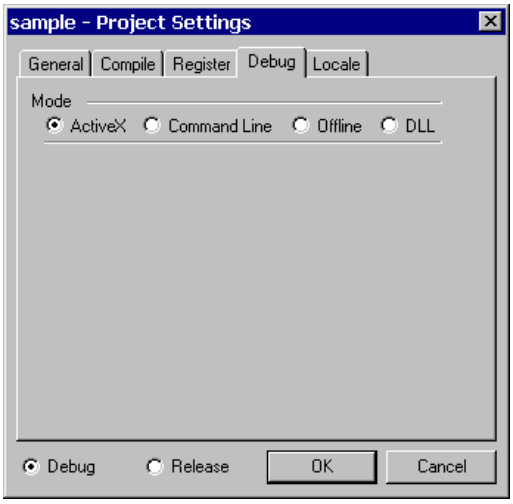


Figure 190: Select ActiveX

This option allows you to start the Transall Application and leave it to wait for an ActiveX Client Application to call it (e.g., a Visual Basic Application).

To Specify Command Line Debugging

1. Select **Command Line** if the Transall Application will be started for debugging by the Transall Editor in command-line mode.

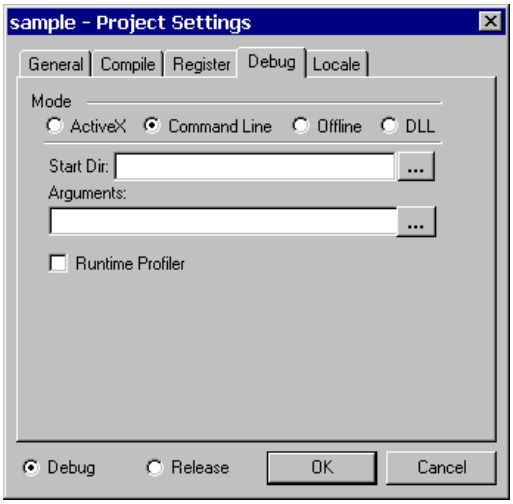


Figure 191: Select the Command Line Radio Button

2. In the Start Dir text box, type the data path of the working directory where Transall should try to locate Tranexe. (This option is similar to using the "Start In" option on a Windows shortcut.)

   -or-

   Click ... to select the folder.

3. In the Arguments text box, type the arguments (if any) to pass to the Transall Application.

   -or-

   Click ... to select the Transall Script, declared as `Public`, in the Transall Application where debuggable execution begins.

   It's important that you type text in the Arguments text box in a particular manner, as summarized in Figure 192.

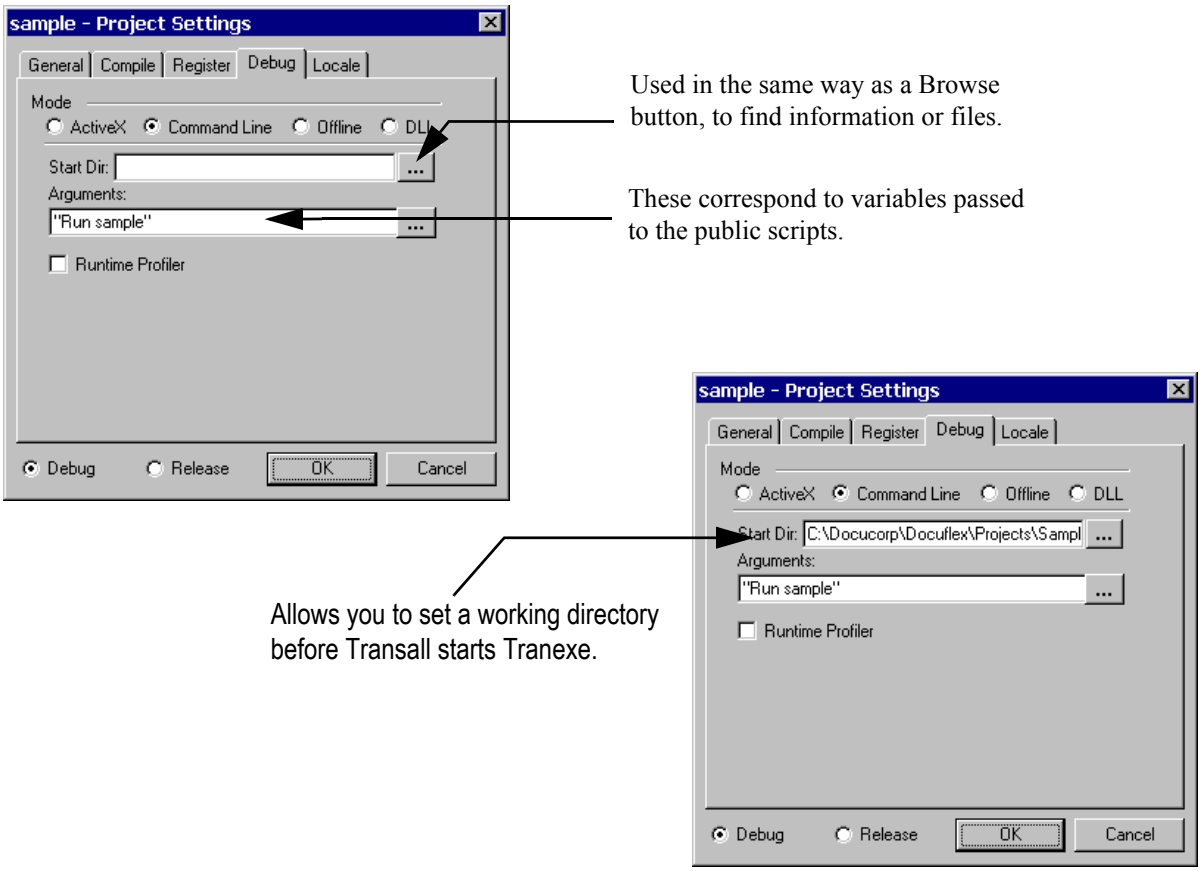


Figure 192: Command-line Debug Settings in the Project Settings Dialog

Notice in the figure that a string argument with an embedded blank space must be enclosed in double-quotes. Otherwise, when running, the Transall Application automatically performs the appropriate type conversion on literal values passed to any of its public Scripts via this mechanism.

4. Enable the Runtime Profiler checkbox to display a profile report in the Transall IDE after a debugging session execution completes. The TranExe module also supports passing the "-prof <filename>" parameter directly on the command line, enabling profiler support when executing outside the IDE. This parameter causes a profile report listing to be generated in the passed <filename>. Transall runs much slower than normal when profiling. Here is an example of the profile report:

```
Transall 11.01.00.01 profile report for 'AlliedDa.tex' Wed Oct 12 18:18:46 2005

Top 100 line numbers consuming CPU time, seconds time consumed:

4177, 0.438968
5575, 0.246649
4484, 0.176921
4173, 0.070266
4175, 0.059781
  :
  :
  :

Top 100 most executed line numbers, exe count:

4169, 9381
4171, 9381
4173, 9381
4175, 9381
4177, 9381
  :
  :
  :
```

The profile report is in two sections. The first section lists up to 100 of the most time-consuming lines of script from the project, sorted from most time-consuming to least time-consuming. The second section lists up to 100 of the most-executed lines of script from the project, sorted from most-executed to least-executed.

Looking at the first entry of the consuming CPU time report from the example, there is "4177, 0.438968," this is saying that script line number 4177 consumed 0.438968 seconds, which was the most time consumed by any single script line for the run. To see the script at this line in the Transall project, open the project in the Transall editor that produced the TEX that was profiled, select the "Debug/External Runtime Error" menu item, and enter 4177 for the line number. This will bring line 4177 into focus so you can see the line in question.

The Transall project must be complied with at least debug line number support enabled for profiling support to be available.

Specifying Offline Debugging

Transall supports debugging Transall Applications that can't execute under the IDE debugger (e.g., Transall applications executing on non-WIN32 platforms, such as UNIX). Transall's Offline debugger is an after-the-fact, log-based, post-mortem debugger and won't work in these situations.

On WIN32 platforms, Transall supports debugging a Transall application as it executes—"live". On non-WIN32 platforms and on WIN32 platforms where the Transall IDE isn't installed, Transall supports creating a log file of events for debugging an application remotely. This debugging log file can then be "played back" on a workstation with the Transall IDE installed to see what happened when the application ran.

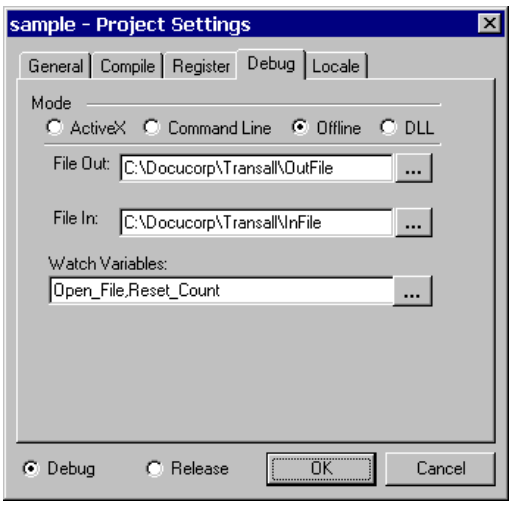


Figure 193: Offline Debug Settings in the Project Settings Dialog

To Specify Offline Debugging

1. Select **Offline** if the Transall Application will be started for debugging by the Transall Editor in Offline mode.
2. In the File Out text box, type the name of the file to receive offline debugging instructions from the Transall Editor.
3. In the File In text box, type the name of the file created by Transall when the application was run in offline debugging mode.
4. In the Watch Variables text box, include all the breakpoints you want set and any variables you want to be able to "watch" in the Transall debugger's Variable Watch window.

Offline debugging mode is triggered by passing two special parameters to the Transall Host at run-time:

- /din <ddorfilename from File Out>
- /dout <ddorfilename from File In>

These parameters pass the name of the "in" and "out" files to the Transall Host. Note that the "in" file from the Editor goes with the "/dout" parameter to Transall at run-time. The "out" file from the Editor goes with the "/din" parameter to Transall because the "out" file from the Editor is the input file to the Host. You must make available this "out" file to the Transall Host running in offline debugging mode. If you have to copy the file in order for it to be available to the Transall Host, be sure to make a binary copy with no character conversion.

If you're using offline debugging via MVS JCL, you need to include two DD statements in your JCL for the debug files. Also, you need to pass the "/din <dd>" and "/dout <dd >" parameters in the JCL's PARM= statement to enable offline debug mode and tp provide the names of the DD statements to Transall.

When you're ready to begin an offline debugging session, see *To Perform Offline Debugging* on page 319.

To Perform Offline Debugging

1. Complete all the text boxes for Offline debugging, and then click **OK**.
2. Select **Debug>Start** to start the debug session.
3. Click **Write Debug Instructions** in the Offline Batch Debugging dialog.
4. Copy (via a binary copy) the "out" file to a place where the offline Transall Host has access. The "out" file is created by the Editor and specified in the "File Out" text box on the Debug panel of the Project Settings dialog.
5. Run the Transall application with the **/din** and **/dout** parameters and MVS DD statements.
6. Copy (via a binary copy) the "/dout" file to a place accessible to the Transall Editor. The "/dout" file is created by the Transall application execution. The file name must match the file specified in the "File In" text box on the Debug panel of the Project Settings dialog.
7. Select **Debug>Start** to display the Offline Batch Debugging menu.
8. Click **Playback Offline Session** to step through the breakpoints and see the debug information captured during the offline debug execution.

To Specify Transall DLL API Debugging

1. Select **DLL** if the Transall Application will be started for debugging by the Transall DLL API.
2. In the Executable text box, type the name of and parameters to the EXE that will call the Transall DLL API.

   -or-

   Click ... to select the file.

   The Transall Editor will start the named executable file. This file enables Transall to debug calls to the DLL API. Only an executable started by Transall in this manner can debug DLL API calls to Transall. See Deploying Transall Applications on page 326 for details on the Transall DLL API.

Using the Locale Tab

Use the fifth tab in the Project Settings dialog box to specify the major language of your geographical region and whether to use the default Windows formatting symbols for currency, dates, and times.

If you click Dynamic, Transall uses the default symbol value for the Locale that is selected through the Windows Control Panel, regardless of what is selected in the Editor.

If you want to override the default Value for a given Symbol, you can leave the Dynamic check box blank and enter the desired value.

To Display the Locale Panel

- If the Locale panel isn't already showing in the Project Settings dialog box, click on the **Locale** tab.

  The Locale panel displays.

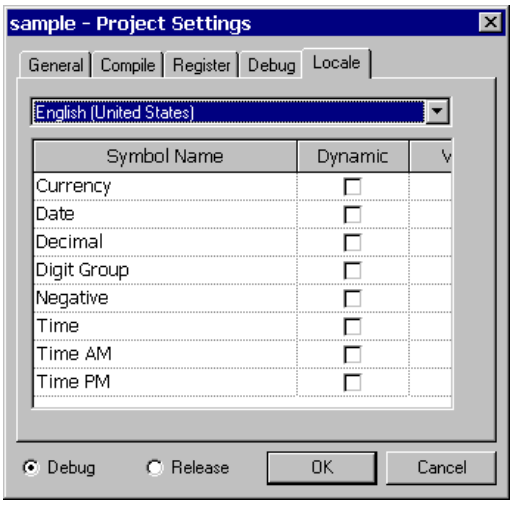


Figure 194: The Locale Panel

Because the Locale panel contains multiple group boxes of specifications, this guide provides a separate topic for each group.

If you need to	Go to this topic
Specify the language common to your geographic region	*To Specify the Regional Language* on page 320
Indicate dynamic loading of common formatting symbols for date, time, and currency	*To Specify Dynamic Loading of Formatting Symbols* on page 321
Specify custom values for the formatting symbols	*To Specify Custom Formatting Symbol Values* on page 321
Save or close the Locale tab panel	*To Save or Close the Project Settings Dialog* on page 308

To Specify the Regional Language

- In the Language drop-down list box, select the language common to your geographic region.

Transall saves the language option *for this workstation*. If you access this project from another workstation, be sure to verify the language.

To Specify Dynamic Loading of Formatting Symbols

Depending on the language you choose on your workstation, Windows contains formatting symbols common to that region for currency, date, and time fields. If you click the Dynamic option, you can't override the symbol value: Transall loads the symbol from the Windows Registry when you compile and run your project.

- For each formatting symbol, enable the **Dynamic** check box if you want to load the value stored in the Windows Registry.

To Specify Custom Formatting Symbol Values

- If you want to use a symbol other than the default, leave the **Dynamic** check box blank and enter the desired Value.

RUNNING A TRANSALL APPLICATION IN DEBUG MODE

An important part of ensuring that your Transall Application is behaving as intended is to watch it in action, but in a controlled fashion. This is called *debugging*.

You can prepare a Transall Application for debugging at the time you produce it. After doing so, you can also use the Transall Editor as a debugging environment. After a debug version Transall Application exists, you can direct the Transall Editor to intercept control of its execution the next time it is started.

After the Transall Application's execution is under the control of the Transall Editor, you can direct the Transall Application to precede one source-code statement at a time, or to proceed from one *breakpoint* to another. You can also view and update the contents of data variables as the Transall application runs.

To use the Transall Editor in this manner, you first must take these steps:

- Produce a debug version of the open project's Transall Application.
    - Set a breakpoint in the source code of at least one of the Transall Application's Transall Script routines.

    -or-

    Set a breakpoint in at least one of the Transall Application's LogicTree routines.

Note By setting one or more breakpoints in a LogicTree routine, you can watch the flow of your logic without having to view the underlying source code.

COMPILE SETTINGS FOR PRODUCING A DEBUG VERSION

To produce a debugging version of the open project's Transall Application, you must set the proper Transall Editor compile settings.

1. Select **Project>*project* Settings**.
2. Select the **Compile** tab.

3. In the Debugging information group box, click either **Line Numbers** or **Full**.
4. Click the **Debug** option button.
5. Click **OK** to save the settings.
6. Select **Project>Compile** or press **F7**, to compile the open project and produce a debugging version of the Transall Application.

BREAKPOINTS IN TRANSALL

When the debug version Transall Application executes under the control of the Transall Editor, you can cause it to pause at a particular source code or LogicTree statement. This is called *setting a breakpoint*.

To set a breakpoint within the Transall Application, open a Script Module or LogicTree component, set the cursor in the pertinent statement, and select the Debug > Toggle Breakpoint command or press F9.

You can set a breakpoint on any statement in

- a user-written Transall Script subroutine or function.
- any built-in Transall Script method including component methods.
- a LogicTree routine.

For more information about component methods, see the section *Built-In Component Methods* on page 328.

Figure 195, shows a Script Module's Contents view for the Transall Script routine named CreateCPD after the user has set a breakpoint.

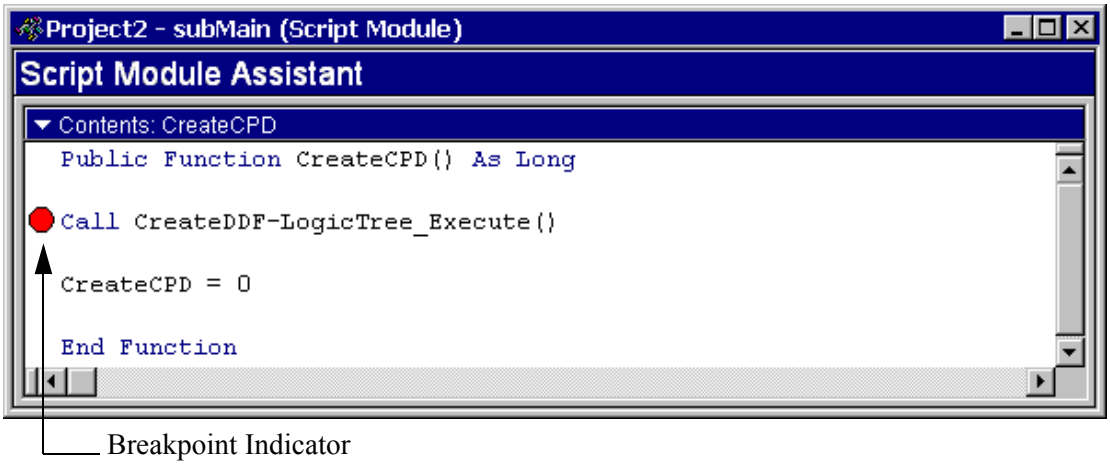


Figure 195: Breakpoint in Script Module Contents View

Figure 196 shows a LogicTree's Contents view for the LogicTree routine named Start after the user has set a breakpoint.

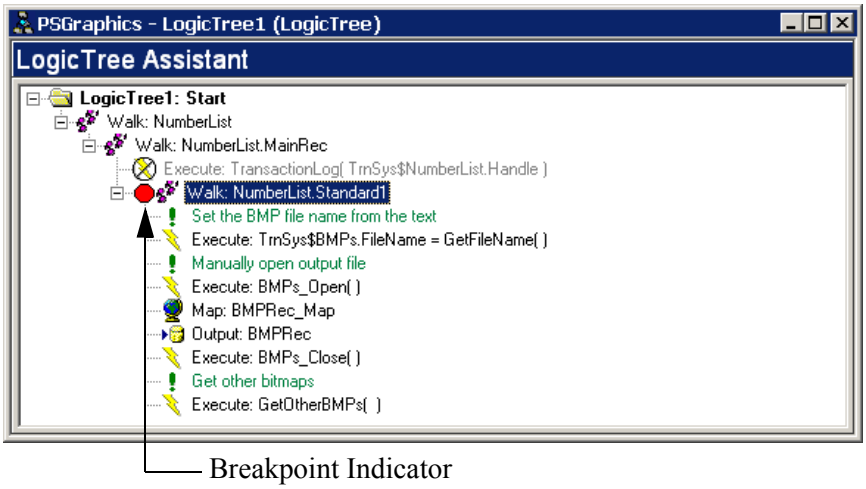


Figure 196: Breakpoint in LogicTree Contents View

When the Transall Application's execution has paused, you can examine:

- Values of variables defined within the paused Transall Script subroutine or function (that is, whose lexical scope is *local*)
- Values of variables defined in the Declarations sections of the project's Script Modules (that is, whose lexical scope is *global*)
- The names of Transall Script routines that have been called before reaching the paused subroutine or function

For example, if you know that one Transall Script routine in your Transall Application calls another, you can set a breakpoint on a significant statement in the calling routine and on one in the called routine. When the Transall Editor gains control of the running Transall Application, you can allow the execution to pause at the calling routine's breakpoint then continue the execution until it reaches the called routine's breakpoint. In this manner, you can verify that the Transall Application is behaving as expected within a certain context of execution.

OPERATING TRANSALL UNDER DEBUGGING CONTROL

To operate a debug version of a Transall Application under the control of the Transall Editor, follow these steps:

1. Select **Debug>Go**, or press **F5**, to direct the Transall Editor to start or to intercept control of the Transall Application when it is next started. (Notice that the Transall Editor now presents its debugging interface; "[Run]" status appears in the Transall Editor window's title bar.

Note If the application is not an ActiveX application, then the F5 key starts the run time execution. If the application is ActiveX, the F5 key just loads the ActiveX module and tells it to wait for a call to one of its public methods (subroutines).

If this is an ActiveX debug session, use an external, ActiveX-enabled application to call a public routine that has been compiled into the Transall Application.

The Transall Editor gains control of the Transall Application's execution and causes it to pause at the first statement encountered where a breakpoint is set.

2. As the Transall Application executes, use Debug menu commands or Debug toolbar buttons to step from one source-code statement to the next, or from breakpoint to breakpoint. Control bars in the Transall Editor's debug interface present local and automatic variables, specified "watch" variables, and the names of the routines presently on the Transall Application's routine call stack.
3. To end debugging before the Transall Application completes its execution, select **Debug>Stop Debugging**, press **CTRL+F5**, or press the Stop Debugging button on the Debug toolbar.

You can individually configure the Transall Editor's three debugging control bars. With the mouse cursor over one of these bars, right-click for a pop-up menu that allows you to hide the bar, allow it to float within the Transall Editor, or allow it to dock on one side of the Transall Editor.

After the Transall Editor gains control of the Transall Application's execution, when paused at any statement you can:

- Select **Debug>Step Into**, or press F8, to proceed one source-code statement at a time. If the next statement calls another routine within the Transall Application, that routine's source code is opened and execution continues to the first statement in that routine.
- Select **Debug>Step Over**, or press Ctrl+F8, to proceed one source-code statement at a time. If the next statement calls another routine within the Transall Application, execution pauses at the next statement in the calling routine *after* the calling statement.

When you stop debugging the Transall Application, the Transall Editor terminates the Transall Application and reverts to its default, component-oriented interface. When the Transall Application terminates in this manner, it returns an error condition message to its caller in the COM-based execution environment.

Hint Transall "remembers" the look of the editor while debugging and changes to that look each debugging session.

Viewing Debug Variables in the Variables and Watch Bars

The Debug Variables bar offers two tabs to select which set of variable names and values to display:

- Auto - Variables in the current and previous line of execution
- Local - Variables in this routine only

While the Transall Application's execution is stopped, change the value of a variable shown in the Debug Variables bar by clicking on the variable's value cell and enter a new value.

As shown in Figure 197, stopping execution in a Transall Script function routine causes the Debug Variables bar to display a variable with the function's name; this variable represents the function's return value.

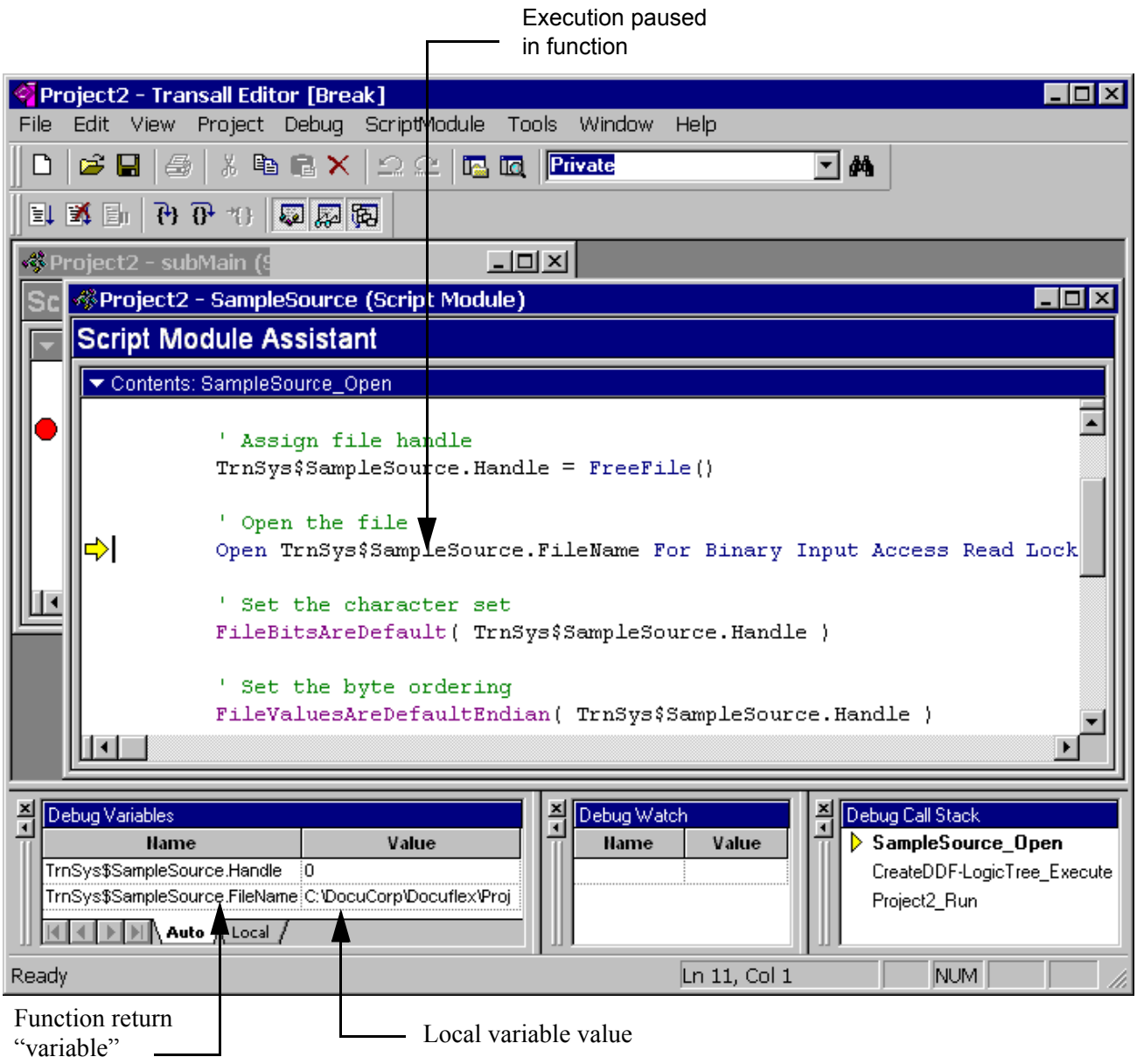


Figure 197: Local Variable Values Visible in Debug Variables Bar

In the Debug Watch bar, click in a Name cell and enter the name of a variable. This variable's value will be displayed in the value cell when it is within the current execution scope. That is, variables defined in any Script Module's Declarations section are always in scope, but a routine's local variables (those defined by a Transall Script Dim statement) are only in scope, and therefore only have a value, while that routine is being executed.

Changing Display Scope for Variables

Click on an item in the Debug Call Stack bar to display the values of variables that exist and are in scope for that routine's execution context.

Building a Transall Application for Release

When you are ready to produce a Transall Application for use in a production system environment, you can edit the Project Settings dialog that controls the setting for a release version of the Transall Application. To do so:

1. Select **Project>*project* Settings** to open the Project Settings dialog.

   Update any relevant settings under the dialog's General, Compile, and Register tabs.

2. Select the **Compile** tab.
3. In the Debugging Information group box, click **None**.
4. Click the **Release** option button at the bottom of the dialog.
5. Click **OK** to close the dialog.
6. Select **File>Make *project*.tex** to compile the open project. This action produces a release version of the Transall Application.

Deploying Transall Applications

Transall applications can be deployed and executed several ways:

- Windows command line
- Batch (*.BAT) files

  The Windows command line and batch file support enables you to start a Transall application and pass it parameters via Windows batch files or the Windows command line. The Transall application will run a single script to conclusion and terminate.

- ActiveX automation
- DLL Application Programming Interface (API)

  ActiveX automation and DLL API support enables you not only to run a Transall application script, but also to call several scripts in a single "session" while the Transall application maintains its "state" (connections to data sources such as files and SQL databases) across the calls. This is a more sophisticated way to control a Transall application from another application.

- Under Oracle's Internet Document Server (IDS)

  Running Transall under Oracle's IDS facility:

  - enables an IDS request type to be defined that loads a Transall application (whose name is provided in an IDS attachment variable), and
  - executes a script in the application, with the script's name also being provided in an IDS attachment variable.
- as a Windows Service

WINDOWS COMMAND LINE

Transall applications can be run directly from the Windows command line. To start a Transall application from the Windows command line, select the Windows Start button and then click on **Run…**. In the "Open" text box, enter the command to start Transall using the same syntax that is used to run Transall applications in batch files.

BATCH (*.BAT) FILES

Transall applications can be controlled via Windows batch files. These are files whose extension ends in .BAT and they contain a script of Windows command interpreter commands (such as copy, mkdir, and ren). Please see your Windows documentation for the Windows Command Interpreter for more information on the commands that can be included in Windows Command BAT files. Transall applications can be executed as part of Windows .BAT files with the following syntax:

TRANEXE *app.tex StartScriptName [parameter] [parameter] [parameter] […]*

where:

Parameter	Meaning
app.tex	Name of a Transall executable. This is a file produced by the Transall compiler with an extension of .TEX.
StartScriptName	Name of the script subroutine that Transall should execute.
parameter	Value to be passed to the script. Parameters passed on the command line are matched to the *StartScriptName* script's parameters by order. The first *parameter* goes to the first *StartScriptName* script parameter and the second *parameter* goes to the second *StartScriptName* script parameter and so on.

After Transall has completed executing the *StartScriptName* script, it will set the .BAT variable **ERRORLEVEL** to the value returned by the *StartScriptName* script if the script was a Function returning a numeric value such as an Integer, Long or Double. If the *StartScriptName* script returns a String then **ERRORLEVEL** will be set to the numeric value of the string as if the string was processed by a Val function. If the *StartScriptName* script was a Sub or a function returning a DateTime then **ERRORLEVEL** will be set to zero for a successful run and to 1001 if a critical error occurred that was not handled by an On Error Resume processing in the Transall application.

ACTIVEX AUTOMATION

Transall applications can be run as ActiveX Automation Servers from ActiveX enabled Automation clients such as Microsoft Visual Basic and Microsoft Office applications. Transall supports both early (through a Type Library *.TLB file) and late (through iDispatch) ActiveX Automation binding. Please see the detailed documentation available from your ActiveX Automation client software (VB or Office) for details on controlling ActiveX Automation severs.

Before Transall applications can be controlled via ActiveX Automation they must be "registered" on the workstation as available ActiveX Automation servers. The following command line syntax can be used to register Transall applications:

TRANHOST *app.tex* /Register

Where *app.tex* is the name of a Transall executable. This is a file produced by the Transall compiler with an extension of .TEX. If your ActiveX Automation client software is Microsoft Visual Basic the following are the general steps required to control Transall applications as ActiveX Automation servers:

Early Binding (advantage slightly faster call to Transall, disadvantage must set up a reference to your Transall project's TypeLib TLB file in VB).

1. Set a VB reference to the Transall project's TypeLib TLB file.
2. In your VB code, define a variable of type ***AppName.*TransObject** where *AppName* is the internal name of your Transall application.
3. In your VB code make the following call to load your Transall application into memory:

   Set *Step2varName* = **New** *AppName.*TransObject
4. Run public **Sub** scripts in your Transall application:

   Call *Step2varName.PublicScriptName([parm][, parameter][…])*

   Run public **Function** scripts in your Transall application:

   RetVar = Step2varName.PublicScriptName([parm][, parameter][…])

Special early binding notes:

- Microsoft Visual Basic holds the Transall TypeLib TLB open while you are in VB. If you recompile your Transall application while VB is open, and you have your Transall application setup to regenerate the TypeLib on compile, you will get a compile error that Transall could not write the TypeLib. You will need to close the VB project to get VB to release the Transall TypeLib.

- The Transall application loads into memory on the third step call to "*Step2varName* = **New** *AppName.*TransObject". The Transall application will stay in memory until the *Step2varName* is assigned to **Nothing** or the variable passes out of scope (see VB documentation on VB variable scoping). Since Transall stays in memory and holds its memory "state" across calls to its Public scripts, it is a good idea to set up your Transall application's Sources and Destinations with the "Manual" OpenMode property. Then setup a Public script that Opens the Sources and Destinations and another that Closes them. Have your VB code call the Open script after the "*Step2varName* = **New** *AppName.*TransObject" call. This will enable Transall to hold its connections to data sources and destinations while you make several calls to Transall Public scripts. This can be much more efficient than having each Public script automatically open and close connections to the data sources and destinations each time they are called.

Late Binding (advantage do not need to set up a reference to your Transall project's TypeLib TLB file in VB).

1. In your VB code define a variable of type **Object**.
2. In your VB code make the following call to load your Transall application into memory:

   Set *Step1varName* = **CreateObject("***AppName.*TransObject**")**
3. Run public **Sub** scripts in your Transall application:

   Call *Step1varName.PublicScriptName([parm][, parameter][...])*

   Run public **Function** scripts in your Transall application:

   RetVar = Step1varName.PublicScriptName([parm][, parameter][...])

Special late binding notes:

- The Transall application loads into memory on the second step call to "**Let** *Step1varName* = **CreateObject("***AppName.*TransObject**")**". The Transall application will stay in memory until the *Step1varName* is assigned to **Nothing** or the variable passes out of scope (see VB documentation on VB variable scoping). Since Transall stays in memory and holds its memory "state" across calls to its Public scripts, it is a good idea to set up your Transall application's Sources and Destinations with the "Manual" OpenMode property. Then setup a Public script that Opens the Sources and Destinations and another that Closes them. Have your VB code call the Open script after the "**Let** *Step1varName* = **CreateObject("***AppName.*TransObject**")**" call. This will enable Transall to hold its connections to data sources and destinations while you make several calls to Transall Public scripts. This can be much more efficient than having each Public script automatically open and close connections to the data sources and destinations each time they are called.

DLL APPLICATION PROGRAMMING INTERFACE (API)

Transall applications can be run as Dynamically Loaded Libraries (DLL) from applications that can load DLLs such as Microsoft Visual Basic and Microsoft Visual C/C++. Transall installs with an "API" subdirectory. In this directory are three files:

trandynm.bas - Microsoft Visual Basic Declare syntax for Transall DLL Application Programming Interface (API).

trandynm.h - C declaration syntax for Transall DLL API.

trandynm.lib - Static library for WIN32 compatible linkers that defines the Transall DLL API.

There is also a fourth file that is required for the DLL API. This file is found in the Transall install directory

trandynm.w32 - WIN32 DLL that implements the Transall DLL API.

To call Transall applications as a DLL from VB perform the following steps:

1. Include the trandynm.bas file in your VB application's file list. This will define the Transall DLL API functions to your VB application.
2. In your VB code setup a Global variable of type Long. This will receive a handle to your Transall application when it is loaded as a DLL. For this discussion I am calling this variable "hTex".
3. In your VB code make the following call to load your Transall application into memory:

   iOk = **LoadTransallTex(***sTexFileName***,** *hTex***)**

   iOk is the name of an Integer variable that receives the results of the call.
   sTexFileName is the name of the Transall executable *.TEX to load.
   hTex will receive a handle value to the loaded TEX. Note, more than one TEX can be loaded at once. Create a separate Global variable of type Long to receive the handle of each loaded TEX.

   This and all functions for the Transall DLL API return Zero to indicate no error and non-Zero to indicate error.
4. In your VB code call one or more of the following to load each parameter to the Transall Public script that you want to call via the DLL API:

   iOk = **SendParmString(***hTex*, *StringParmValue*, **Len(***StringParmValue***))**
   iOk = **SendParmLong(***hTex*, *LongParmValue***)**
   iOk = **SendParmDouble(***hTex*, *DoubleParmValue***)**
   iOk = **SendParmDateTime(***hTex*, *Year*, *Month*, *Day*, *Hour*, *Minute*, *Second*, *Millisecond***)**

   Repeat these calls to load each parameter to the Transall Public script that you want to call via the DLL API.

5. In your VB code make the following call to run a script in your Transall application:

   iOk = **ExeMethod(**hTex, ScriptName)

6. In your VB code call one or more of the following to retrieve the value of ByRef parameters passed to the Transall Public script called in Step 5:

   StringParmRetLen = **Len(***StringParmRet***)**
   iOk = **GetParmString(***hTex*, *ParmIndex*, *StringParmRet*, *StringParmRetLen***)**
   iOk = **GetParmLong(***hTex*, *ParmIndex*, *LongParmRet***)**
   iOk = **GetParmDouble(***hTex*, *ParmIndex*, *DoubleParmRet***)**
   iOk = **GetParmDateTime(***hTex*, *ParmIndex*, *Year*, *Month*, *Day*, *Hour*, *Minute*, *Second*, *Millisecond***)**

   Repeat these calls to retrieve the value of each ByRef parameter to the Transall Public script that called in Step 5.

Note You get retrieve the return value of a script by passing on ParmIndex value of zero.

7. Repeat steps 4 – 6 to call Public functions via the DLL API.

8. In your VB code make the following call to unload your Transall application from memory:

   iOk = **UnloadTransallTex(***hTex***)**

Special DLL notes:

- Calling Transall applications via the DLL API provides some of the flexibility of ActiveX Automation without requiring that the Transall application be registered.
- The Transall application loads into memory on the third step call to LoadTransallTex. The Transall application will stay in memory until the eighth step call to UnloadTransallTex. Since Transall stays in memory and holds its memory "state" across calls to its Public scripts, it is a good idea to set up your Transall application's Sources and Destinations with the "Manual" OpenMode property. Then setup a Public script that Opens the Sources and Destinations and another that Closes them. Have your VB code call the Open script after the LoadTransallTex call. This will enable Transall to hold its connections to data sources and destinations while you make several calls to Transall Public scripts. This can be much more efficient than having each Public script automatically open and close connections to the data sources and destinations each time they are called.

IDS INTERFACE

Transall applications can run as "rules" under Oracle's Internet Document Server (IDS). To setup a Transall application to run as an IDS rule, update the IDS DOCSERV.INI with the following entries:

```
[ REQTYPE:SomeName ]
   function        = tranrule->Run
```

These entries identify **Tranrule.dll** (for Windows, and **tranrule.so** for UNIX) as a loadable module that implements a "Run" interface conforming to the IDS calling convention (see the IDS documentation for more information on the details of rule module interfaces).

Note For DFLXRULE, TRANRULE, and CPYRULE to run correctly with Oracle's Internet Document Server 2.0 (and later) on the Linux platform, you must update the export definition for LD_PRELOAD by adding this file to the preload list:

/usr/lib/libstdc++-libc6.2-2.so.3

Include this change in the script you use to start IDS.

The Run interface in the Tranrule module retrieves the value of the "TEXFILE" and "TEXSCRIPT" attachment variables from IDS when Run is invoked. It expects the TEXFILE variable to contain a fully-qualified path and file name to a Transall executable module that will be loaded by Tranrule. The Run interface also expects the TEXSCRIPT variable to contain the name of a Public script in the executable module loaded by TRANEXE. This Public script must be set up as a Subroutine taking no parameters. The Run interface in Tranrule will then execute the Public script named in TEXSCRIPT.

Tranrule is invoked by an Init, RunForward, RunReverse, and Term call from IDS. Since Tranrule responds to all IDS calls, we have to use a script, such as IDS-Run-Example(), to determine which IDS call we act on.

Note TrnSys$IdsRunEvent, TrnSys$IdsInit, TrnSys$IdsRunForward, TrnSys$IdsRunReverse, and TrnSys$IdsTerm are new reserved variables.

Option 1:

```
Public Function IDS-Run-Example() As Long

    If TrnSys$IdsRunEvent = TrnSys$IdsInit Then

        'Init

    ElseIf TrnSys$IdsRunEvent = TrnSys$IdsRunForward Then

        'Run Forward
        '* Call some Logictree to do work here *

    ElseIf TrnSys$IdsRunEvent = TrnSys$IdsRunReverse Then

        'Run Reverse

     ElseIf TrnSys$IdsRunEvent = TrnSys$IdsTerm Then

        'Term

    End If

End Sub
```

Option 2:

```
Public Sub IDS-Run-Example()

    Static lCallCount As Long

    lCallCount = lCallCount + 1

    If lCallCount = 1 Then

        'Init

    Else If lCallCount = 2 Then

        'Run Forward
        '* Call some Logictree to do work here *

    Else If lCallCount = 3 Then

        'Run Reverse

    Else If lCallCount = 4 Then

        lCallCount = 0
        'Term

    End If

End Sub
```

When running under Tranrule and IDS, Transall script can call several special functions to retrieve information from the IDS environment:

- **DSIAddAttachVar**(<*long queue flag*>, <*String Variable Name*>, <*String Return*>)

  Use this function to push values into any IDS queue. The <*long queue flag*> parameter can be set to either **TrnSys$DSI_INPUTQUEUE** or **TrnSys$DSI_OUTPUTQUEUE** to select the input or output queues.

- **DSILocateAttachVar**(<*long queue flag*>, <*String Variable Name*>, <*String Return*>)

  Use this function to retrieve values from any IDS queue. The <*long queue flag*> parameter can be set to either **TrnSys$DSI_INPUTQUEUE** or **TrnSys$DSI_OUTPUTQUEUE** to select the input or output queues.

- **GetExtEnvString**(<*String Variable Name*>)

  Use this function to retrieve a value from the environment in which Transall is running. When running under IDS, this will be the IDS input queue. This function works when Transall is running under IDS, Docuflex Entry, Docuflex batch processing, or any environment in which Transall can run it. Using this function to retrieve external variable values keeps your Transall script from needing to know that it is running under IDS, Docuflex, or some other environment.

- **IDS_Get**(<*String Variable Name*>)

  Use this function to retrieve values from the IDS input queue. This function returns the value retrieved as a string.

- **IDS_Put**(<*String Variable Name*>, <*String Value*>)

  Use this function to push values into the IDS output queue.

Transall applications loaded to Tranrule under an IDS instance retain their "state" across calls to Tranrule.

Example A Transall application running under Tranrule can connect to a database as part of the initial IDS invocation—and stay connected—so processing will be as fast as possible for subsequent invocations of the Transall application under IDS.

The best way to accomplish this type of database connection pooling is to perform database connections, and any other onetime processing that is statefull, as part of the Transall application's OnCreate event script. Once connected, the database connection(s) can then be accessed via Transall LogicTrees and other Transall facilities normally. These connections made in the OnCreate script will stay connected across subsequent IDS Tranrule invocations.

WINDOWS SERVICE

Transall applications can be run as Windows Services. To run a Transall application as a Service, it must be "installed" as a Service. Transall Service installation is performed from the Windows command line via the Transall Command Line Host Tranexe application. Tranexe supports both installing and removing Transall applications as Windows services via the "`-instserv`" and "`-remserv`" command line options. The full syntax for installing a Transall application as a service is as follows:

```
tranexe -instserv <Tex file name>[ <Service Start
Public Script>]
```

where:

Parameter	Meaning
<Tex file name>	Name of Transall executable
<Service Start Public Script>	(Optional) Name of a public script where service processing starts

For example, the Transall installation includes a sample Transall service application called "DirWatch." This application is a sample shell application for watching a directory on a Windows server. When files or sub directories are added to the watched directory, this application will locate each new file and directory and act upon it. For the sample DirWatch, the application just moves the files from the watched directory to another directory. In a real application, using the techniques shown in the DirWatch sample, Transall could be used to start other applications (like a Oracle publishing application) or perform other processing that is triggered by the appearance of files in the watched directory.

The sample DirWatch application can be installed as a service by opening a Windows command prompt, changing to the directory where the DirWatch example resides, and running the following command line:

```
tranexe -instserv DirWatch
```

The results will look something like this in the command prompt window

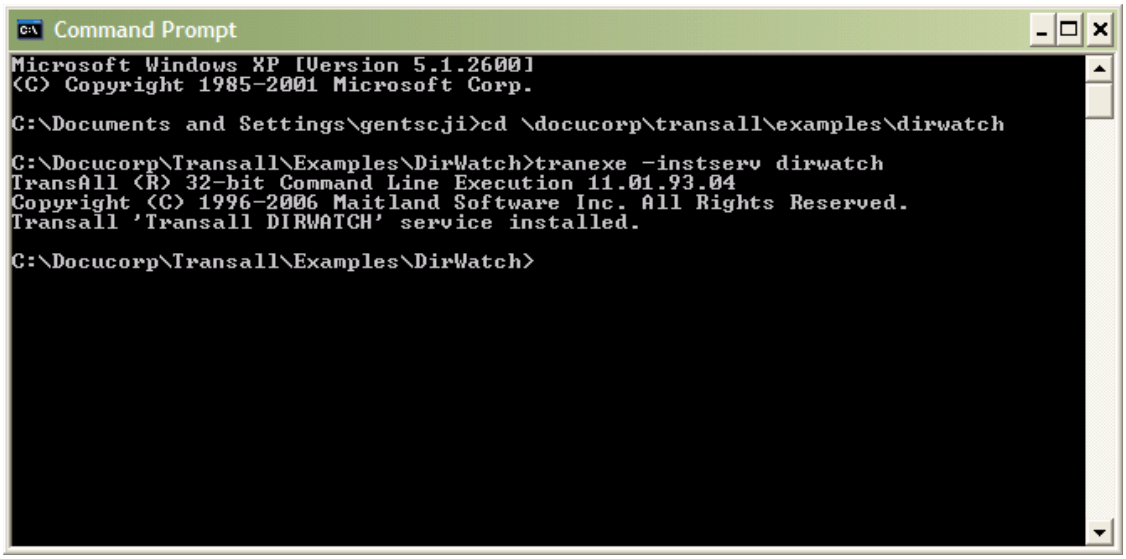


Figure 198: Results of the dirwatch Command

The DirWatch Transall application is installed as a Windows Service with these properties:

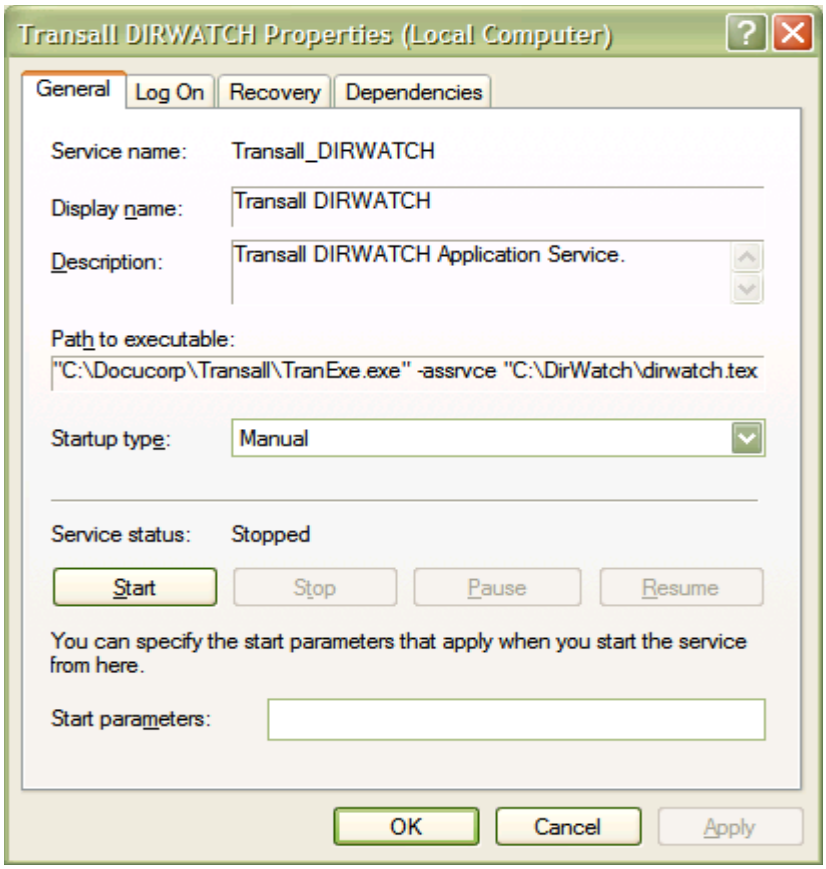


Figure 199: DirWatch Application Windows Service

To remove the DirWatch Transall application as a Windows Service, open Windows command prompt, change to the directory where the DirWatch example resides, and run the following command line:

```
tranexe -remserv DirWatch
```

Writing to the System Events Logs from a Transall Application Running as a Service

Once a Transall application is installed and running as a Windows Service, output from the application that would have gone to the Windows Command console now goes to the Windows event log. Any calls to **WriteConStdErr(<Message String>)** go to the Windows Application event log as Error events, and calls to **WriteConStdOut(<Message String>)** go to Windows Application event logs as either Information events or Warning events (depending on the content of the Message String).

If the Message String starts with “W:”, like

```
"W: Trans #2653 checking #0566209133 failed check
digit testing."
```

then this message is logged as a Warning.

If the Message String does not start with a “W:”, like

```
"Processed 1,121,385 transactions."
```

then the message is logged as Information.

Working with Transall Scripts and Script Modules

OVERVIEW

This chapter describes how to construct a Transall Script and how to collect Scripts in a Script Module component.

For some applications, the programmability provided by your Transall project's Maps and Logic Trees is sufficient. For other applications, where additional business rules require a larger body of custom processing, you can construct reusable sequences of instructions called **Scripts**.

Each Script contains statements in the Transall Script programming language. Each Script's first statement "declares" the Script as either a **subroutine**, which returns no value to its caller, or a **function**, which returns a value to its caller. Each Script can use its own set of variables and can contain any number of Transall Script statements.

The *Transit Script Language Syntax, in Appendix A: Statement Syntax* on page 393, describes all statements and expressions available in the Transall Script language.

A Script can call, and be called by, other Scripts that are contained in your Transall project. (A common use of a Script is to provide custom capabilities for the Transall built-in component methods listed in section, *Appendix A: Built-In Component Methods* on page 346.)

A Script can also be called by an Execute instruction in a Logic Tree component in your Transall project.

Scripts are not themselves components, but must be contained in components called Script Modules. A **Script Module** can contain one or more Scripts. Each Script Module serves as a container for a related set of Scripts.

In addition to one or more Scripts, a Script Module contains a **Declarations section** where you use Transall Script statements to declare

- external routines that are called by any of this Script Module's contained Scripts.
- Transall working Storage Tables that are used by Scripts contained in this Script Module.

CREATING A SCRIPT MODULE

To add a new Script Module to your project, select the **Project>Add Script Module** command from the Transall Editor's menu bar. In the Add Script Module dialog enter a name for the new Script Module and click **OK**.

The Transall Editor opens the Script Module Assistant for the new Script Module in the workspace area.

As shown in *Figure 200*, by default a new Script Module's Script Module Assistant presents the Module's Declarations section.

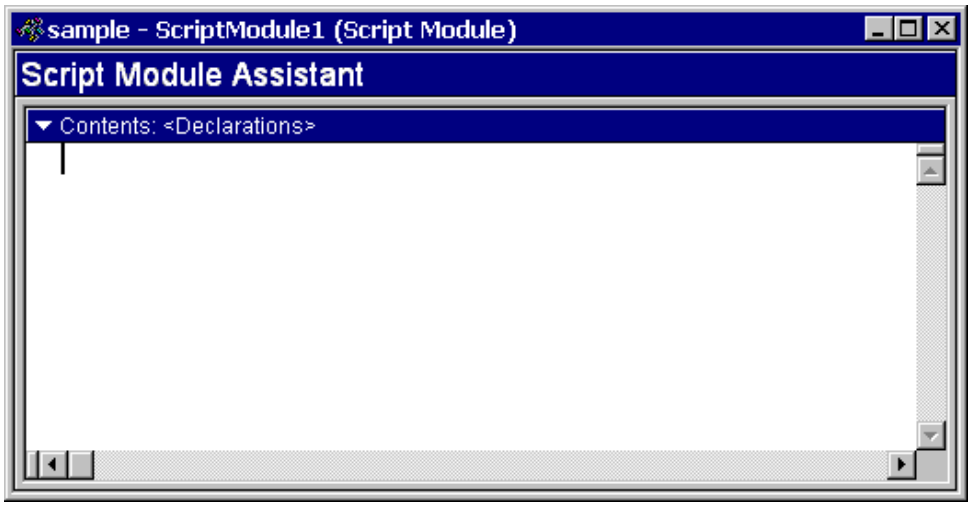


Figure 200: Empty Declarations Section of a New Script Module

CODING THE DECLARATIONS SECTION

Figure 201, shows a Declarations section that contains valid Transall Script language declarations for two external routines that are called by the Scripts contained in this Script Module.

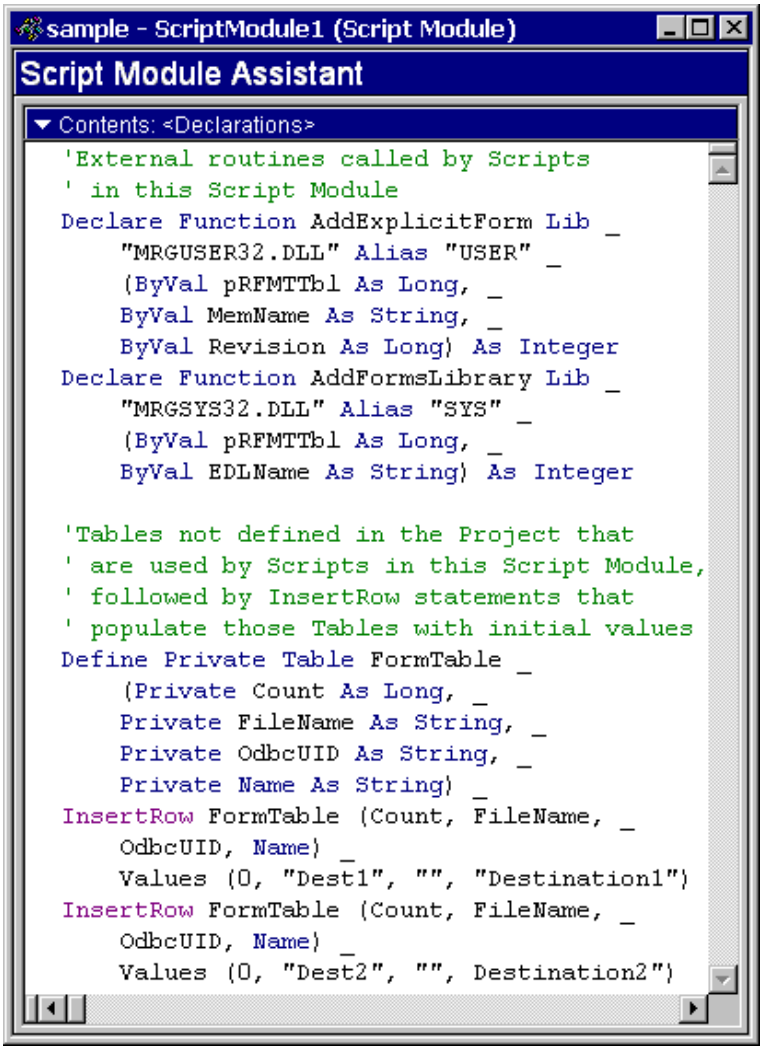


Figure 201: Declarations Section That Declares External Routines and Noncomponent Table

This Declarations section also illustrates how to declare and populate rows for a Table that the Scripts in this Script Module can use. This kind of Table is called a **noncomponent Table**, because its definition is not stored in the Transall project as a distinct component. The final two InsertRow statements populate two rows of that Table. (This is the only valid use of the InsertRow verb outside the context of a Script function or subroutine.)

Hint The underline (_) character indicates the continuation of a Transall Script statement on the next line.

The *Transit Script Language Syntax,* in *Appendix A: Statement Syntax* on page 393, describes the syntax for the Declare Function, Define Table, and InsertRow statements in the Transall Script language.

Adding a Script

To add a Script to the open Script Module, select the **Script Module>Script Add** menu command from the Transall Editor menu bar, or right-click on the background of the Script Module Assistant and select the **Script Add** pop-up menu command.

This opens the Add Script dialog, shown in *Figure 202*.

Figure 202: Add Script dialog

To Add a Script

1. In the **Name** text box, type a name for the Script.
2. In the **Type** group box, select whether it is a subroutine or function Script.
3. If the Script is a function, select the return value's Transall Script data type.
4. Accept the default **Access** property value of Private. (Select Public only if you are adding a Script that will be called from a routine outside your Transall Application, such as via ActiveX Automation.)
5. Click **OK** when finished.

   After completing the dialog, the Transall Editor displays the Contents view of the Script Module Assistant.

   As shown in *Figure 203* on page 343, the Contents view now displays the Transall Script statements that define a subroutine: **Sub** and **End Sub**.

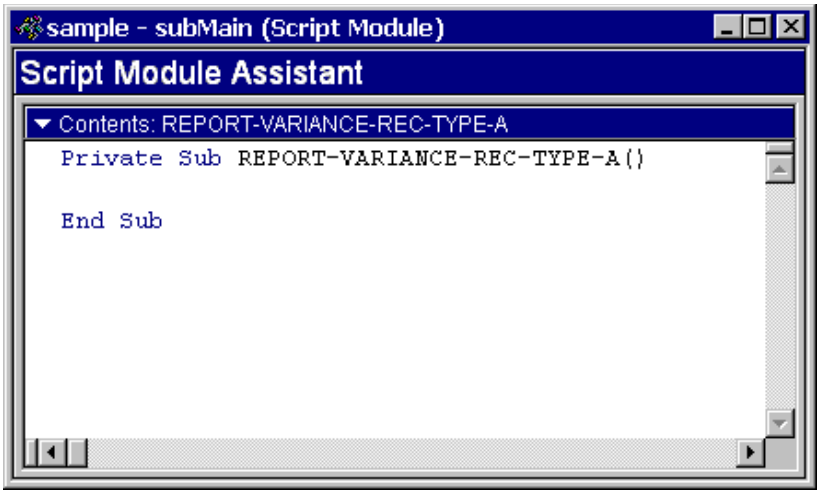


Figure 203: Default Appearance of Script Module Assistant for New Script

6. Add Transall Script statements to the body of the new routine between the **Sub** and **End Sub** statements. The *Transit Script Language Syntax*, in *Appendix A: Statement Syntax* on page 393, describes all statements and expressions available in the Transall Script language.

Tip The context menu contains a command called Auto Formatting. Enabled by default, you can toggle automatic formatting of script code. In general, you should avoid using keywords as variable names. If you must, however, you can disable automatic formatting to keep the Editor from changing the case of words to make them match a Transall keyword (e.g., “name” would be changed to “Name” with Auto Formatting enabled).

7. If you want to construct a statement quickly and insert it at the cursor location in the Script Module Assistant, you can use the Transall Editor’s Expression Builder dialog. The section, *Appendix A: Interacting with the Expression Builder Dialog* on page 258 describes how to construct and format expressions. To open the Expression Builder dialog when editing a Script, right-click in the background of the Script Module Assistant and select the **Expression Builder** pop-up menu command.

8. If you need to create variables or constants, you can use the Transall Editor’s Variable and Constant Builder. To open the Variable and Constant Builder when editing a Script, right-click in the background of the Script Module Assistant and select the **Variable Builder** pop-up menu command.

   As shown in *Figure 204* on page 344, the dialog assists in the creation of local and global variables, global constants, and local static variables.

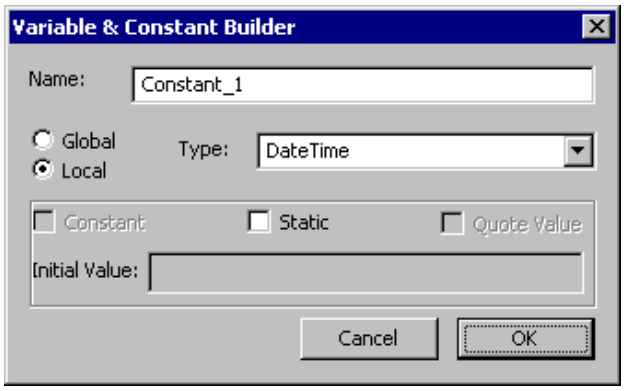


Figure 204: Variable & Constant Dialog Box

The Variable and Constant Builder is an alternative to hand-coding variables and constants. It provides some protections, in that it disallows constants in local scripts, places globals in a user declaration area, and generates the appropriate syntax, but essentially, it's just enforcing the compiler constraints.

For more information about global constants, see *Expression Syntax* on page 443.

To Delete a Script

- Select the Script **Module>Script Delete** menu command from the Transall Editor menu bar

  -or-

  Right-click in the background of the Script Module Assistant and select the **Script Delete** pop-up menu command.

Editing a Script's Source Code

The Script Module Assistant for a Transall Script subroutine or function provides a simple text editor. A cursor indicates where text can be inserted.

The editor provides these keystrokes:

- Move cursor by character: Up, Down, Left, Right
- Move cursor by word: Ctrl-Left, Ctrl-Right
- Move cursor to Top, Bottom of routine: Home, End

To search for a series of characters in the routine, select the **Edit>Find** menu command.

Alternatively, to search for a series of characters, type a search term (or select an existing search term) in the **Find control bar** and press **RETURN**.

As shown in *Figure 205*, each new search string that you enter in the Find control bar is added to a selectable list.

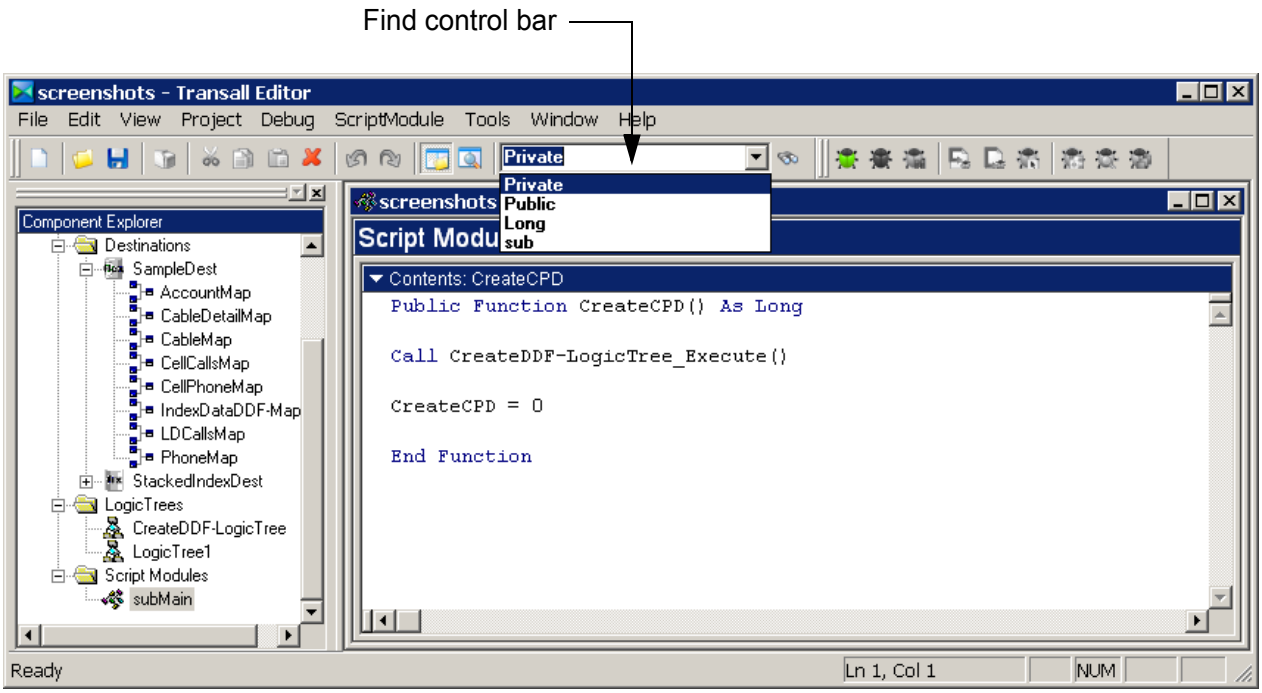


Figure 205: Selectable List of Search Strings in Toolbar's Find Control Bar

To search and replace characters in the routine, select the **Edit>Replace** menu command.

The editor also uses color to distinguish Transall Script keywords, user-specified identifiers, constants, and comments, as follows:

- Transall Script keywords appear in *blue*.
- Transall Script extended keywords, such as built-in functions, appear in *purple*.
- User-specified identifiers, constants, and component names appear in *black*.
- Comments appears in *green*.

Viewing the List of Scripts in a Script Module

After you add Scripts to a Script Module, click the control on the Contents view title bar to open a drop-down list of the names of the Scripts contained in the Script Module.

Figure 206 shows the appearance of this list.

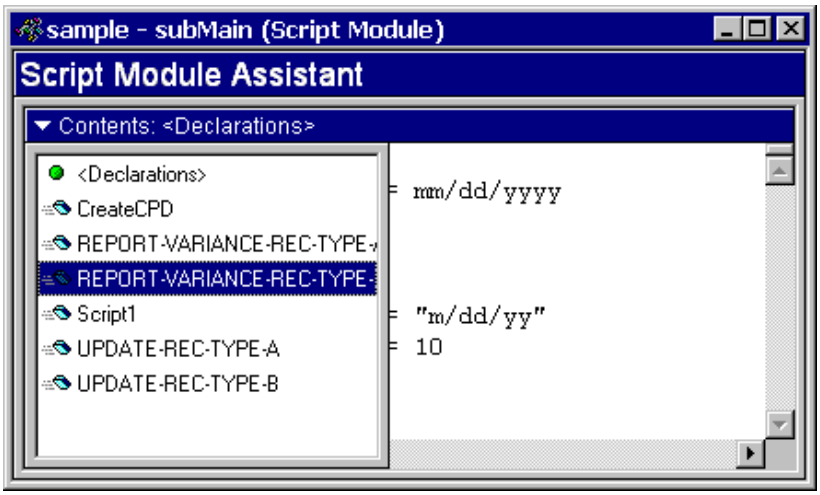


Figure 206: Contents List for a Script Module

Typing the first character of a script name moves you to that entry, a second similar keystroke highlights the next item starting with that character. Use the **Enter** key to open the highlighted script.

BUILT-IN COMPONENT METHODS

Transall provides many built-in component methods that you can call from your own scripts to provide special-purpose behavior in your Transall Application. Some of these methods are user defined and some are generated automatically. Each method is a Transall Script function or subroutine.

For example, for any file-based Source component there are `OnOpenBefore` and `OnOpenAfter` methods that you can customize. One is automatically called before the Transall Application attempts to open the Source, and the other is automatically called after that attempt.

After selecting a component in the Transall Editor's Component Explorer, select the Events tab in the Component Inspector to display the available built-in methods for that component. Double-click on a method name in the list to open a Script Module Assistant where you can edit 'On' events, denoted with a lighting bolt. You can not edit generated events denoted with a building block.

Table 2: Transall Built-in Component Methods lists all the built-in methods whose source code the Transit Editor presents for you to customize.

Table 2: Transall Built-in Component Methods

Component / Method Signature	Description	Access / Function or Sub	Return Type
Table			
OnDelete( )	Called when a record is deleted	Private Function	Integer

Table 2: Transall Built-in Component Methods (Continued)

Component / Method Signature	Description	Access / Function or Sub	Return Type
OnInsert( )	Called when a record is inserted	Private Function	Integer
OnUpdate( )	Called when a record is deleted	Private Function	Integer
Map			
Map (ByVal */FileNumber* As Long)	Map	Private Sub	Void
OnMap (ByVal */FileNumber* As Long)	Called before Map execution	Private Sub	Void
Logic Tree			
Execute( )	Execute	Private Function	Integer
OnExecute( )	Called before Execute	Private Sub	Void
Source			
OnReadAfter (ByVal */FileNumber* As Long)	Called after Read	Private Sub	Void
OnReadBefore (ByVal */FileNumber* As Long)	Called before Read	Private Sub	Void
Read (ByVal */FileNumber* As Long)	Read a record	Private Function	Integer
File-based Source			
GetNextRecord( )	Input Record from Source	Private Function	Integer
OnGetNextRecordBefore( )	Called before a record is read	Private Sub	Void
Destination			
OnWriteBefore (ByVal */FileNumber* As Long)	Called before Write	Private Sub	Void
OnWriteAfter (ByVal */FileNumber* As Long)	Called after Write	Private Sub	Void
Write (ByVal */FileNumber* As Long)	Write a Record	Private Function	Integer

Table 2: Transall Built-in Component Methods (Continued)

Component / Method Signature	Description	Access / Function or Sub	Return Type
File-Based Source or Destination			
Close( )	Close file	Private Function	Integer
OnCloseAfter (ByVal */FileName* As String)	Called after Close	Private Sub	Void
OnCloseBefore (ByVal */FileNumber* As Long)	Called before Close	Private Sub	Void
OnOpenAfter (ByVal */Handle* As Long, ByVal */FileName* As String)	Called after Open	Private Sub	Void
OnOpenBefore (ByRef */FileName* As String)	Called before Open	Private Sub	Void
Open( )	Open record	Private Function	Integer
SetParameters (ByRef */ResourceName* As String)	Changes default values for file name and attributes	Public Sub	Void
ODBC-based Source or Destination			
Connect( )	Establish an ODBC connection	Private Function	Integer
Disconnect( )	End an ODBC connection	Private Function	Integer
OnConnectAfter( )	Called before Connect	Private Sub	Void
OnConnectBefore( )	Called before Connect	Private Sub	Void
OnDisconnectAfter( )	Called after Disconnect	Private Sub	Void
OnDisconnectBefore( )	Called before Disconnect	Private Sub	Void
SetParameters (ByRef */ResourceName* As String, ByRef *sUID* As String, ByRef */PWD* As String)	Changes default values for file name and attributes	Public Sub	Void

Table 2: Transall Built-in Component Methods (Continued)

Component / Method Signature	Description	Access / Function or Sub	Return Type
ODBC-based Source or Destination with Query			
Prepare( )	Readies a SQL statement for execution	Private Function	Integer
Execute( )	Executes a prepared SQL statement	Private Function	Integer

Project Sharing

OVERVIEW

Transall project sharing enables top-level components of Transall projects such as sources, destinations, maps, logic trees, scripts, Transall database tables and sets to be shared between projects. The sharing is supported one of two ways, copy or link. With a copy share, the desired components are copied to the target project. These components are then part of that project exactly as if the Transall user had entered them. With a link share, the desired components are included in the target project but they are not available for editing in the target project. When sharing items from another project via a link means that, you are relying on the source project for the definitions and maintenance of these items. A link is maintained in the target project to the shared components from the source. These linked components can still be edited in the source project and the updates can be absorbed into the target project via a link synchronize feature.

Transall tries to minimize sharing problems by disallowing duplicate names and forcing destinations to be included with shared maps. It is incumbent on the user to assure the shared items have the resources they need. For example, Transall logic trees may reference many items. If shared as a link all items referenced by the logic tree must also be shared as linked items.

PROJECT SHARING DETAILS

Share menu item: Under the File menu is the Share menu item, which displays a dialog for selecting the source project and the Transall items to share.

Choose the source project by typing in the project name or by clicking the ellipsis, button and using the file open dialog. If you manually enter the project name, you must leave the edit field for the project to open. A tree of top-level items from the source project is displayed below the project name.

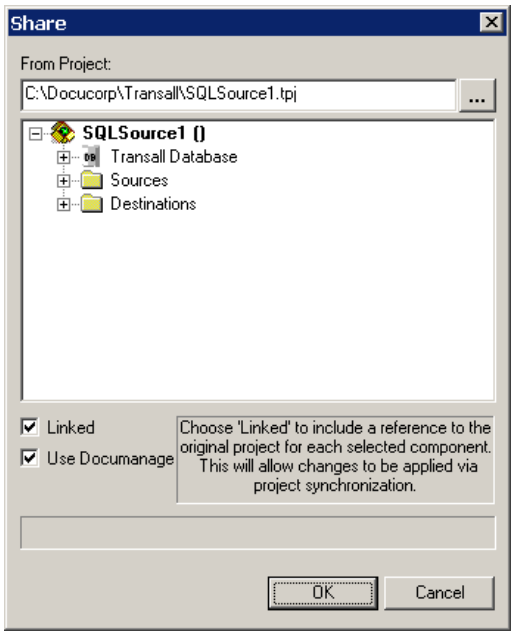


Figure 207: Linking to a Project

Select items by clicking the check boxes to the left of each item, or right click for a popup menu that allows all children of the highlighted node to be selected. The *Linked* checkbox determines whether the item(s) are shared as a link or copy. A project can share any number of links and/or copied items from one or more source projects. The *Use Documanage* checkbox means that you want to select the item(s) from a Documanage Cabinet/Folder (i.e., check Use Documanage and then click the browse button).

Source project items for sharing that are grayed out indicate that the item name, or a key sub item name, already exists in the destination project, and sharing is disallowed. To include grayed out items into the target project you must rename the target project items causing the name collision.

Synchronize menu item: Under the Project menu is a sub menu Synchronize, which has three menu items, Datasource, Links, and DMG Report.

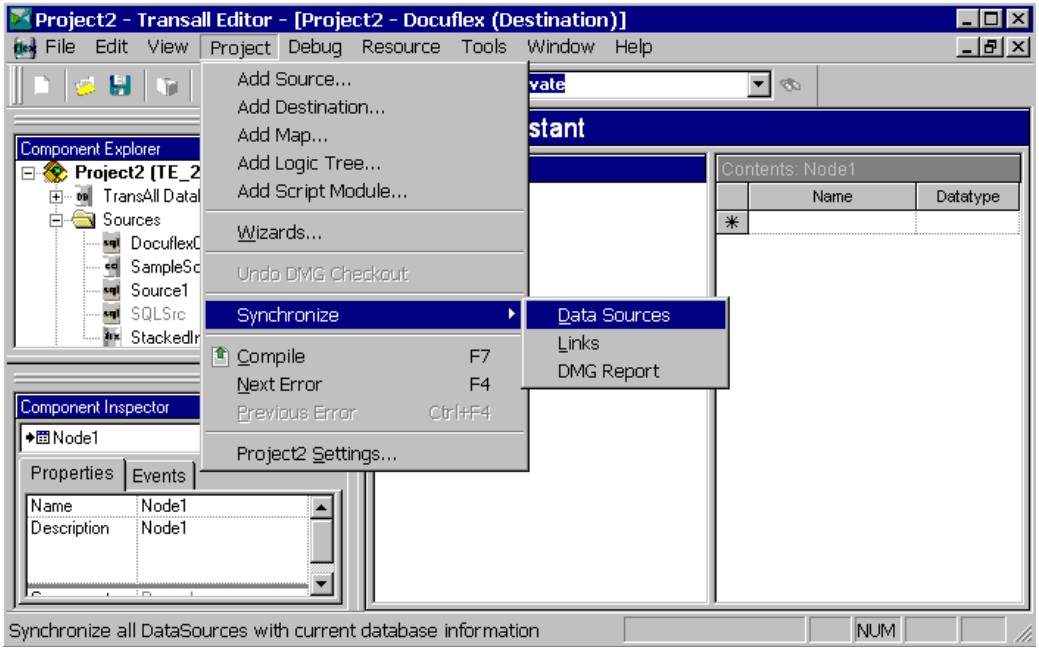


Figure 208: Synchronize Menu

Data sources that are linked into a project cannot be synchronized in the target project. The source project must be opened to perform the datasource synchronize. **Links** synchronizes all links and displays the results in the Output Bar under the Synchronize tab.

Figure 209: Synchronize Links Display Window

DMG Report indicates which Documanage documents will supply link requirements.

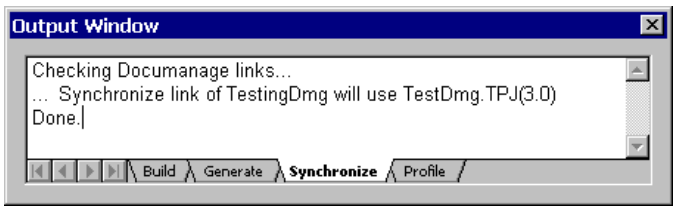


Figure 210: Documanage Links

Links menu item: Under the View menu is menu item Links.

It displays a list box of all the links for the current project.

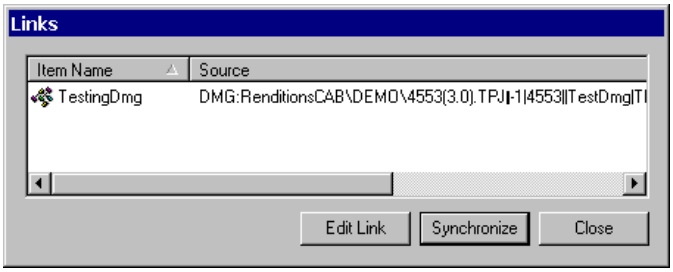


Figure 211: Links Displays Box

Any links from Documanage will appear as "DMG:Cabinet\Folder\id(MajVer.MinVer).ext. Clicking **Edit Link** displays the Select Documanage Document dialog and clicking **Synchronize** retrieves the listed version from Documanage. For all other links, double-clicking a row or clicking the **Synchronize** button will synchronize just that one link.

Managing Transall Applications

TRANSALL HOW TO'S:

CREATING A TRANSALL PROJECT

1. To setup a new Transall project, start the Transall editor (TRANEDIT.EXE).

   The editor will optionally open the last project you had been working on or it will open a default new project.

2. Select **File>New** to have Transall start a new project called "Project1".

Hint When you create a new project, it will be named "Project1".

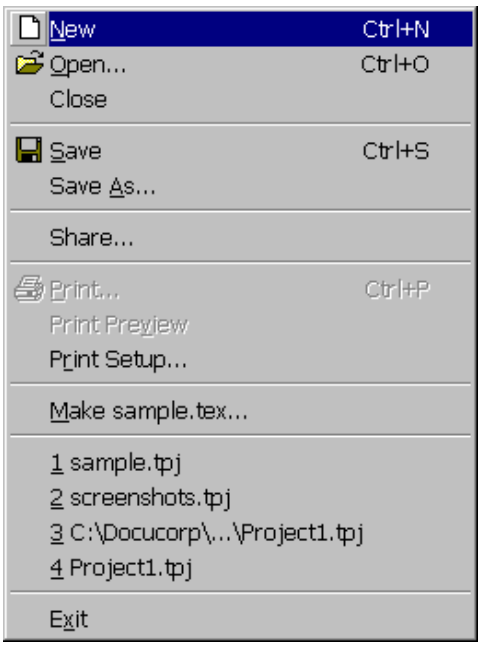


Figure 212: File Menu

3. Open the Component Explorer (if it is not already open) by selecting the **View>Component Explorer** menu item.

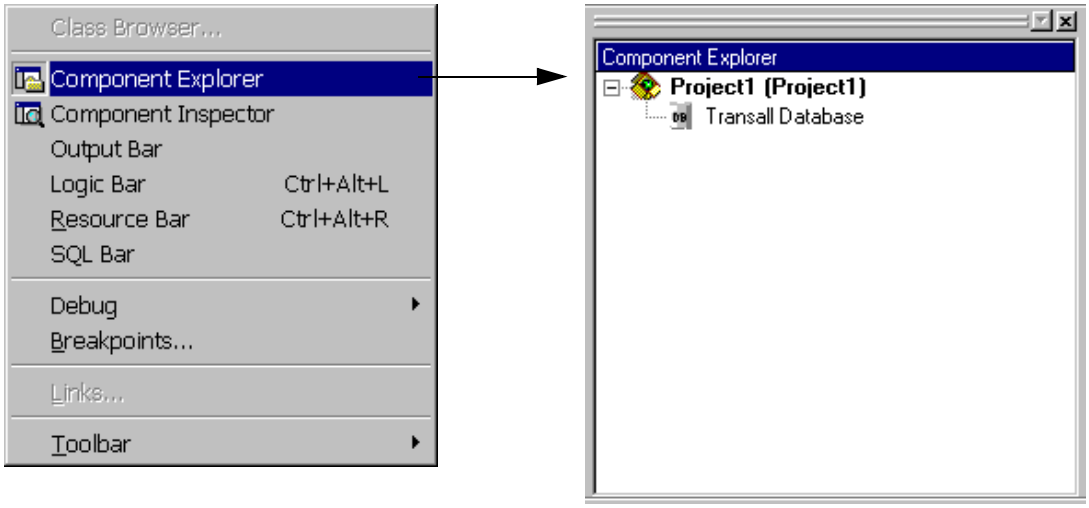


Figure 213: View Menu and Component Explorer

In the Component Explorer control bar, you'll see a Tree View of your Transall project. The top node in the tree is the project node and it displays the project's internal name and external file name. After selecting **File>New** to create a new project in the editor, the project's internal name will be Project1 and the external name will be (Project1).

4. Select the **File>Save** menu item to save and name your project.

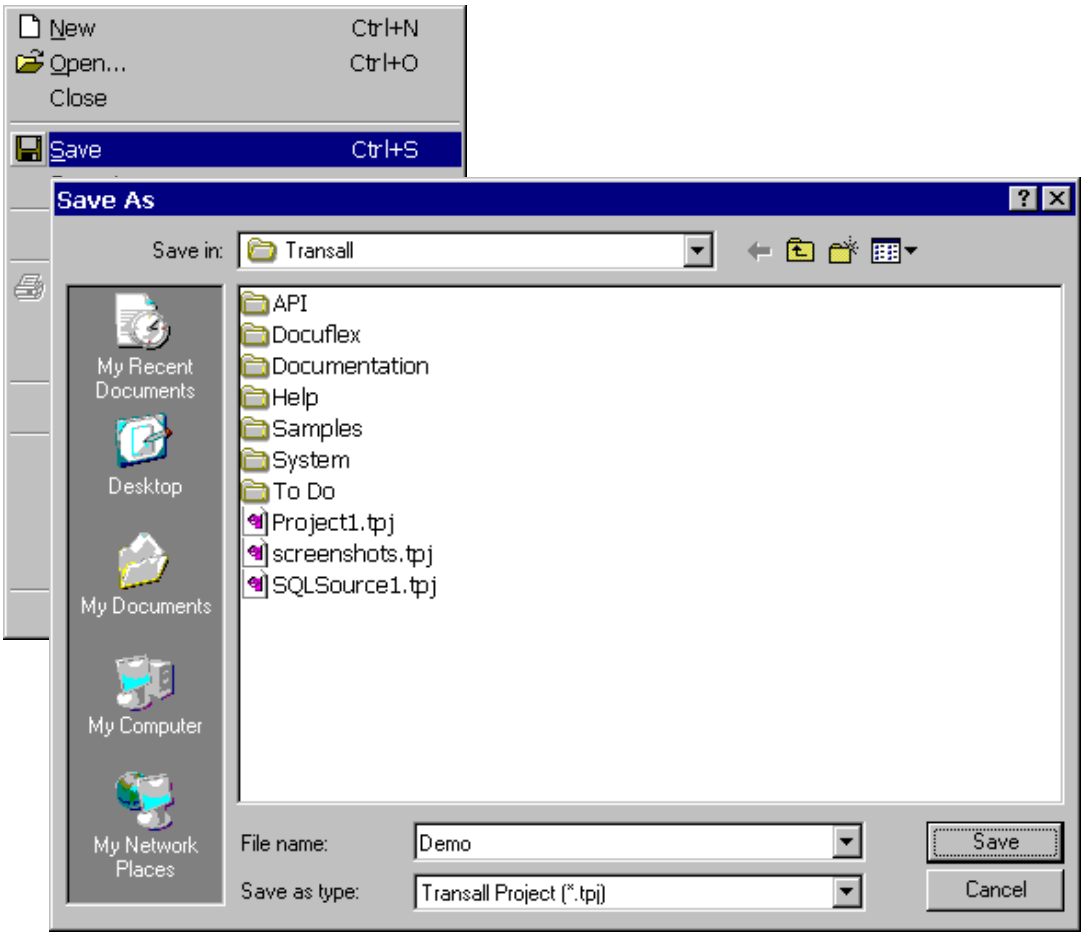


Figure 214: File>Save Menu and Save As Dialog Window

5. From the dialog window, type in a file name for the project. Since this is the first time the project is being saved, the file name typed in here will become both the internal and external name for the Transall project.

   Let's assume you typed, "Demo" for a Save As file name. Transall will update the Component Explorer control bar to reflect that the Transall project now has an internal name of "Demo" and an external name (Demo.tpj). The difference between the internal and external name for a Transall project is that the **external** name of a project is the name of the project file (TPJ file extension) in which the Transall project will be saved. The **internal** name is the name of the executable to which the project will compile when compiled (built) by the Transall editor. The Transall editor is also referred to as the Transall Integrated Development Environment (IDE).

6. Click **Save** to save the project.

OPEN AN EXISTING TRANSALL PROJECT

1. To open an existing Transall project, start the Transall editor (TRANEDIT.EXE).

   The editor will optionally open the last project you had been working on or it will open a default new project.

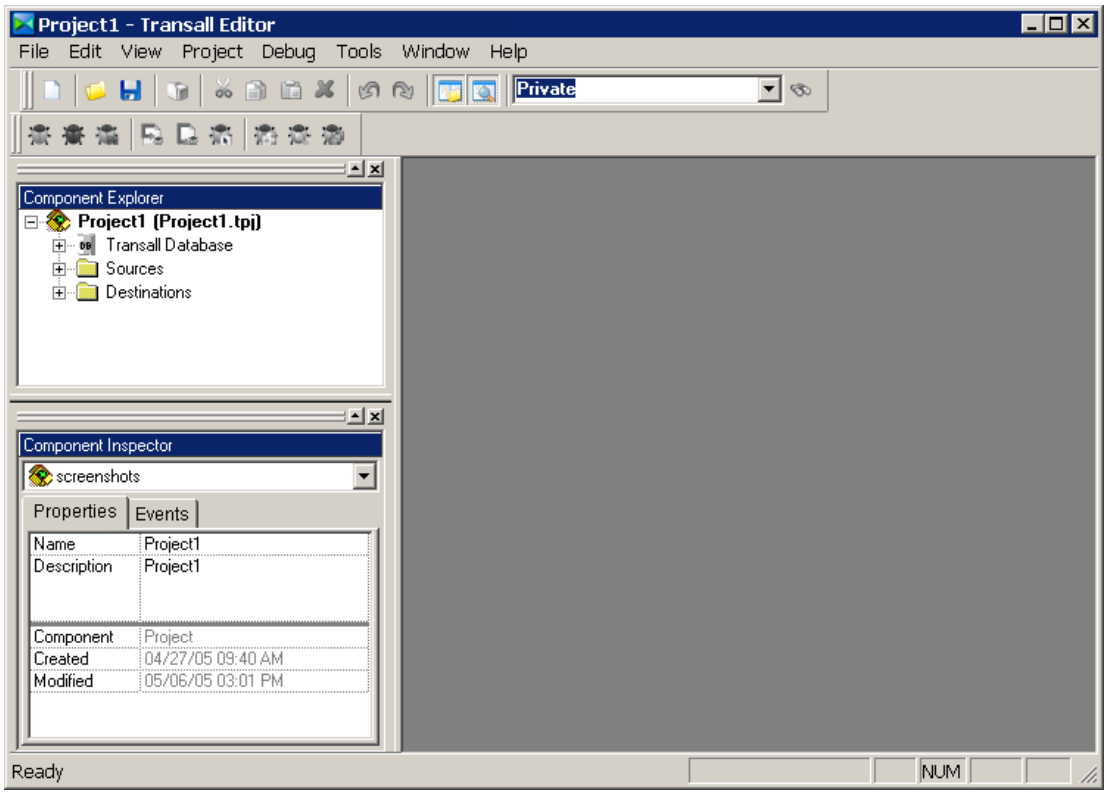


Figure 215: Opening Transall

2. If the editor opens the last project you were working on but that is not the project you need to edit, select the **File>Open** menu item.

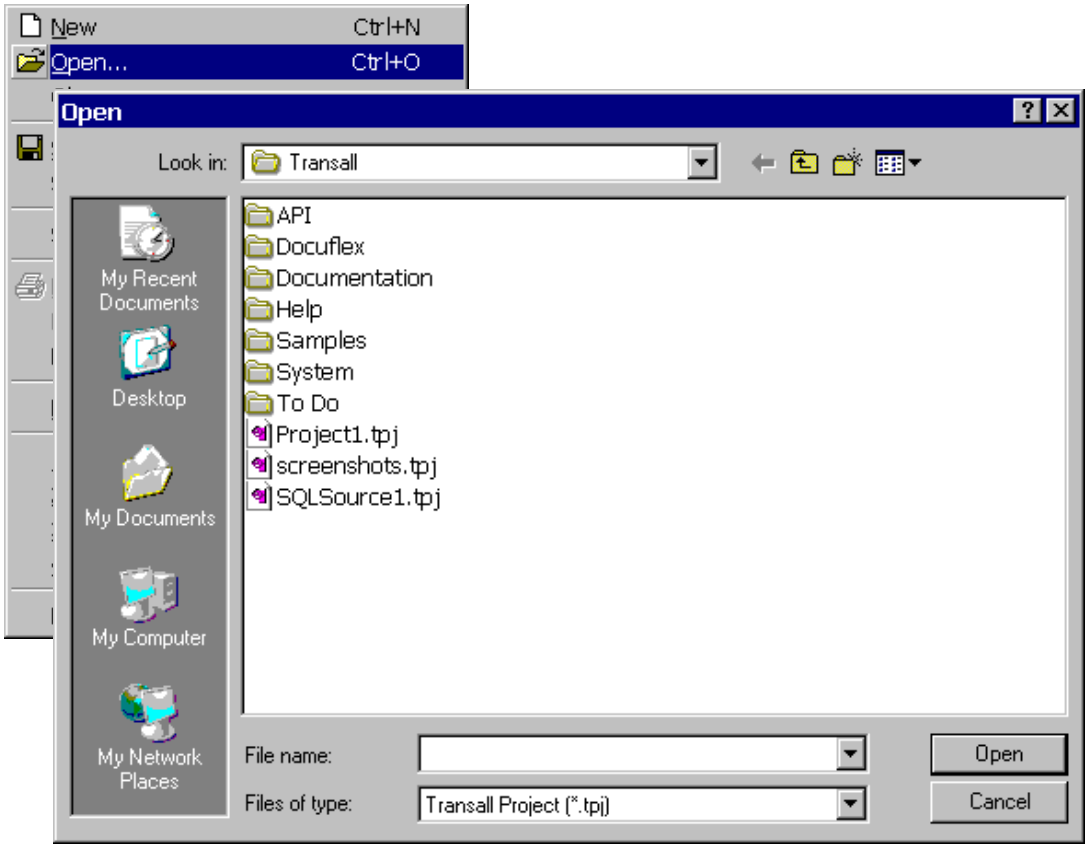


Figure 216: File>Open Menu and Open File Dialog Window

3. From the dialog window, select the Transall project you want to edit and click **Open**.

ADJUSTING CONTROL BARS AND WINDOWS

1. To change the way the Transall Editor shows you a project's contents, select the View menu item.
2. From the View menu item, you can turn the Component Explorer and Component Inspector on or off. These control bars display parts of a Transall project. You might want to turn them on or off to have more screen space in which to work.

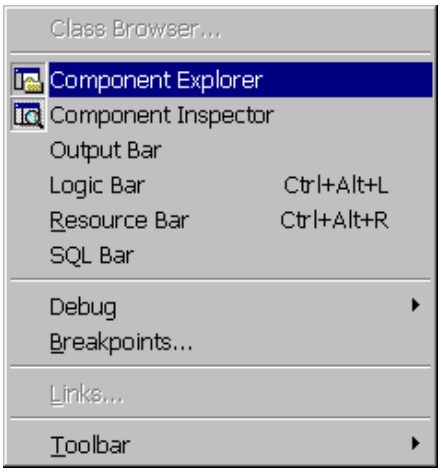


Figure 217: View Menu

3. From the **View>Toolbar** menu item, you can turn on and off toolbars and use the Customize option to adjust the buttons available on the editor's toolbars.

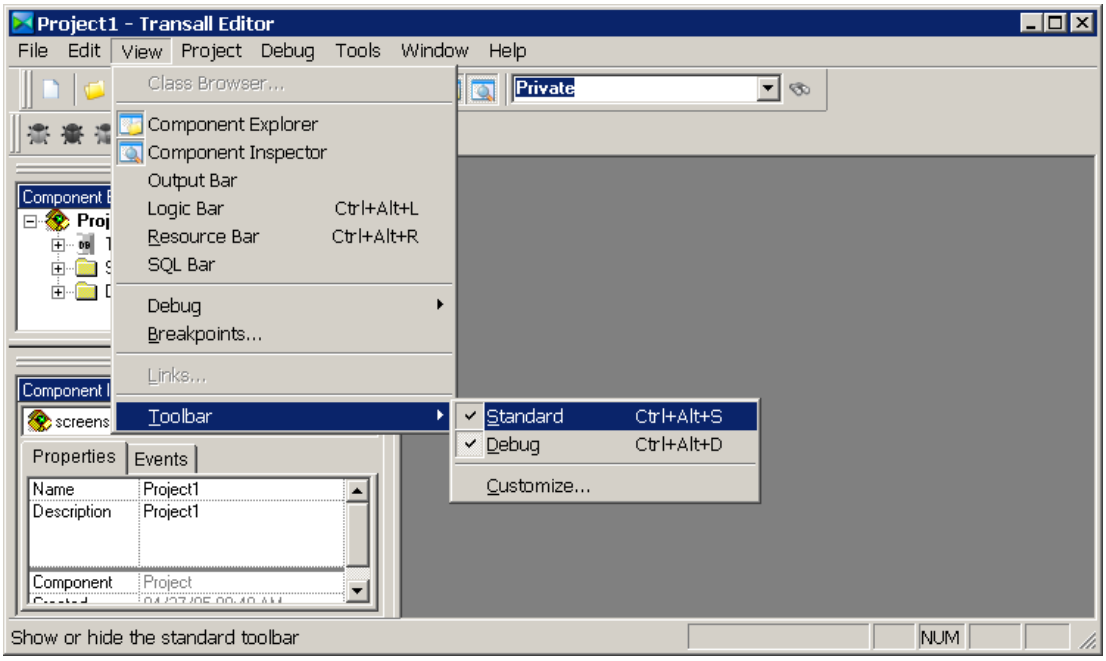


Figure 218: View>Toolbar Menu

4. From the Window menu item, you can switch between Assistant windows that are open in the Transall editor. You can also arrange the windows or close all the windows from the Window menu item.

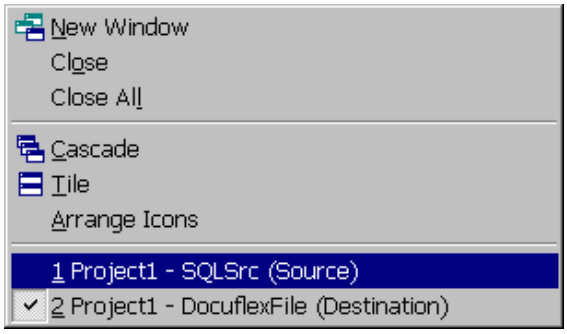


Figure 219: Window Menu

5. To open Editor Assistant windows, double-click on the project component that you want to edit in the Component Explorer view.

   This will open a window specialized for editing the Transall project component you selected.

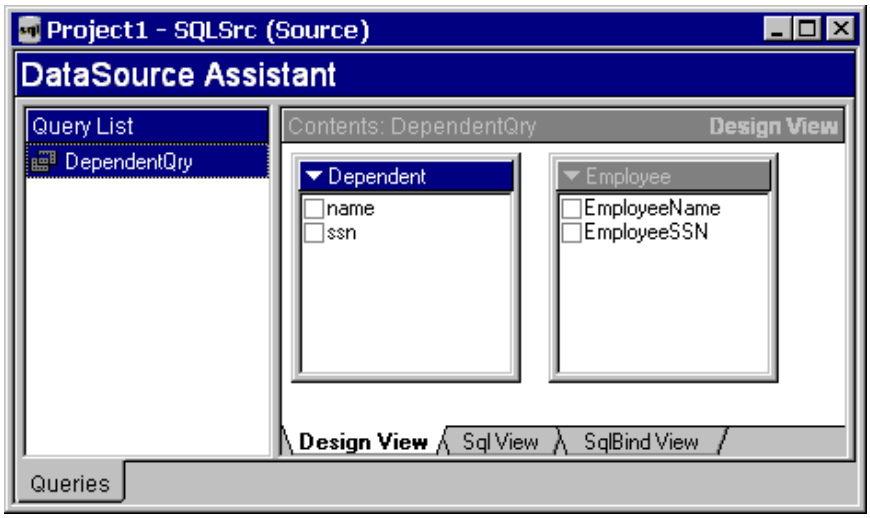


Figure 220: Assistant Window

SETTING UP AN SQL DATA SOURCE OR DESTINATION (ODBC CONNECTION)

To setup a SQL data source, which is an ODBC connection, start the Transall editor (TRANEDIT.EXE) and select a Transall project for editing or start a new Transall project

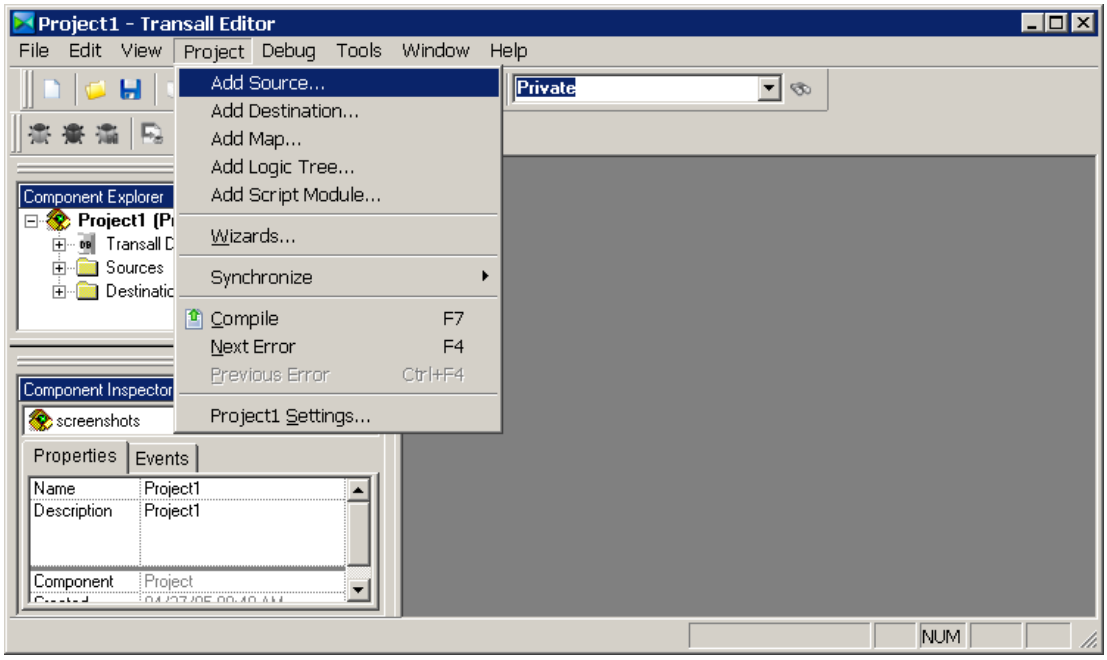


Figure 221: Project>Add Source Menu

Click on the **Project>Add Source** menu item.

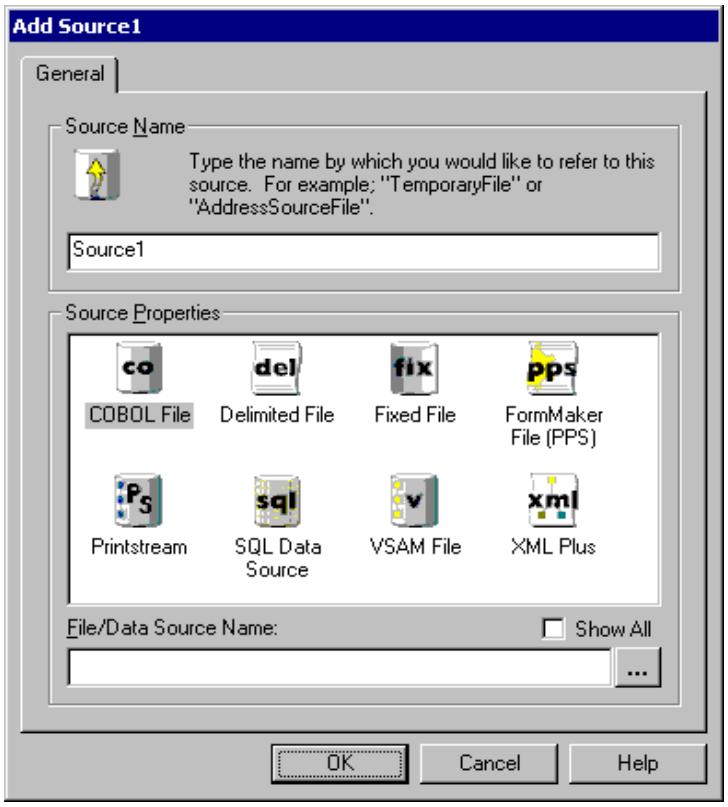


Figure 222: Add Source Dialog Window

This will display the Add Source dialog window.

Note In the dialog window, update the Source Name to something meaningful for the data source. You cannot use special characters or spaces in the name of a data source, but you can use underscores "_" and dashes "-". It is often a good idea to suffix your data source names with a meaningful value, like "-Src" or "-ScHo".

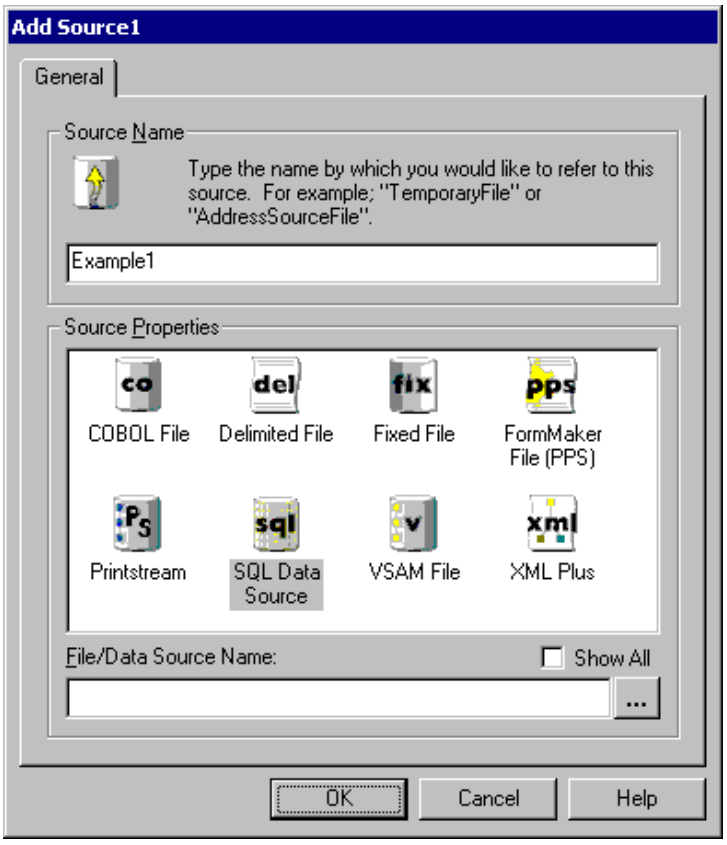


Figure 223: Source Properties List

After typing in a source name, select the type of source from the Source Properties list. For SQL sources select "SQL Data Source".

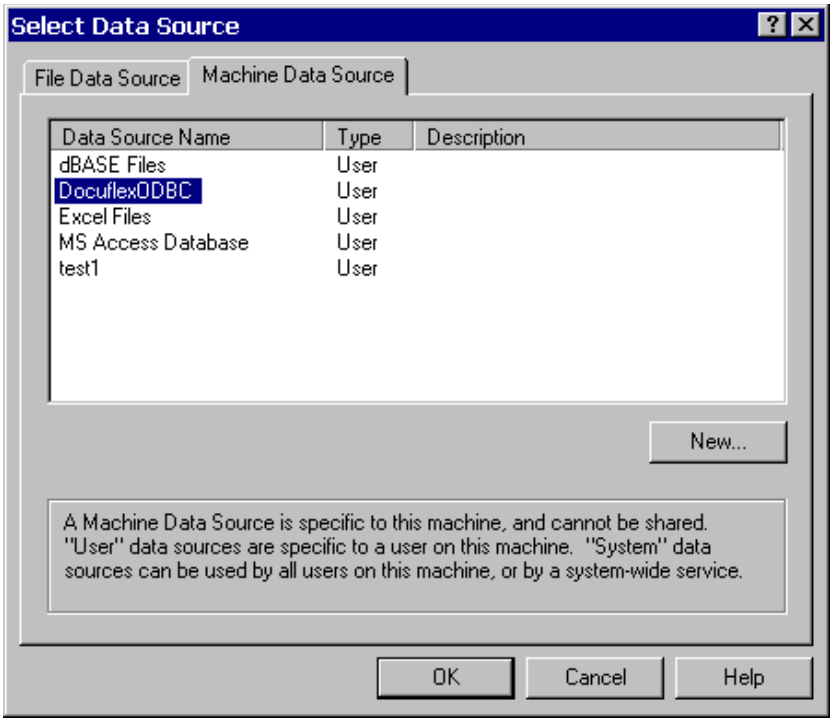


Figure 224: Select Data Source Dialog Window

Now select an ODBC data connection for this source. In the File/Data Source Name window, either type in the ODBC connection name or click on the button with three ellipses to display the Select Data Source dialog window, from this dialog window select or create an ODBC connection.

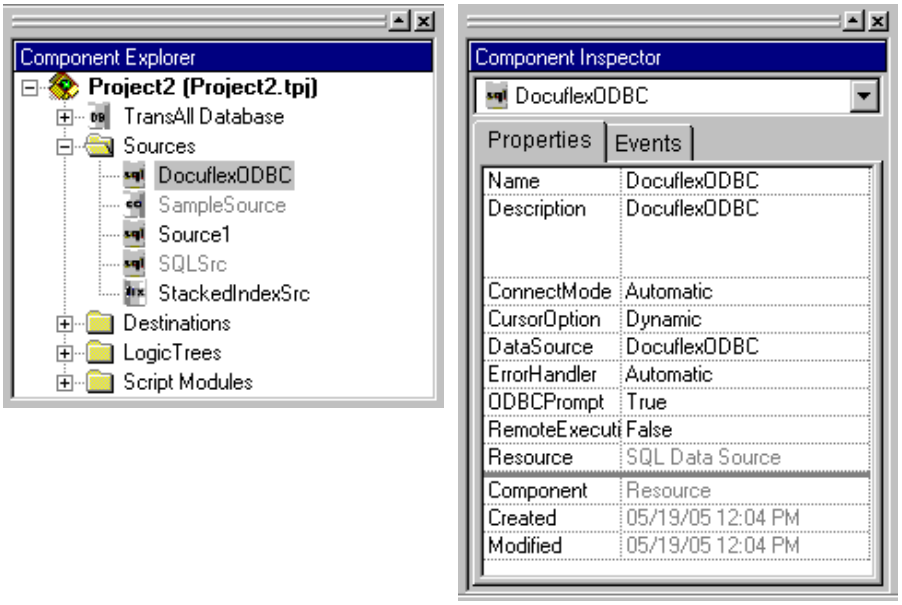


Figure 225: Component Explorer and Component Inspector Windows

After selecting an ODBC connection, click the “OK” button to complete the SQL source creation. You will see your new data source listed in the Component Explorer under the Sources component branch. Also, the Component Inspector will show the details of your new SQL data source. The Component Inspector will show extra details or properties that were not available for setting in the Add Source dialog window.

These extra properties include the following:

ConnectionMode:

Automatic - *(default) Transall will automatically generate logic to connect to this datasource for you.*

Manual - *Transall will generate logic to connect to this datasource but will not automatically call the logic. You must manually place a call to the generated logic in your Transall Scripts or LogicTrees to have Transall run the instructions that connect to this data source. You might want to select Manual if you need to have extra control over when a data source is connected to by Transall. This may be the case if you have a need to connect and disconnect several times to a data source during a Transall run. By default, the Automatic connection mode connects to a data source and holds the connection for the life of the Transall run disconnecting when Transall is ready to terminate.*

ErrorHandler:

Automatic - *(default) Transall will automatically handle and recover from errors. Transall will “throw” only critical errors that must be “caught” via an On Error Resume Next type statement.*

Manual - *Transall will ignore all errors and return errors to your Scripts or LogicTrees for handling.*

ODBC Prompt:

True - *(default) Transall will prompt for missing information such as user ID and password when connecting to an ODBC database connection.*

False - *Transall will not prompt for missing information such as user ID and password when connecting to an ODBC database connection but will instead throw an unable to connect error message. Transall will also retain more information when ODBC Prompt is set to False then when this is set to True. When this property is set to, True Transall only retains the name of the ODBC connection. When this property is set to False Transall retains the name of the ODBC connection and any information provided to connect to the ODBC connection to access database schema information. This extra-retained information generally includes user ID and password.*

Adding an SQL Query (Statement) to an SQL Data Source or Destination

To add an SQL query, which is an SQL statement; to a Transall SQL data Source or Destination, you must first locate or setup a new SQL data Source or Destination for the query. Once an SQL data Source or Destination has been selected, open the data Source or Destination's Assistant window by double clicking on the desired data Source or Destination in the Component Explorer.

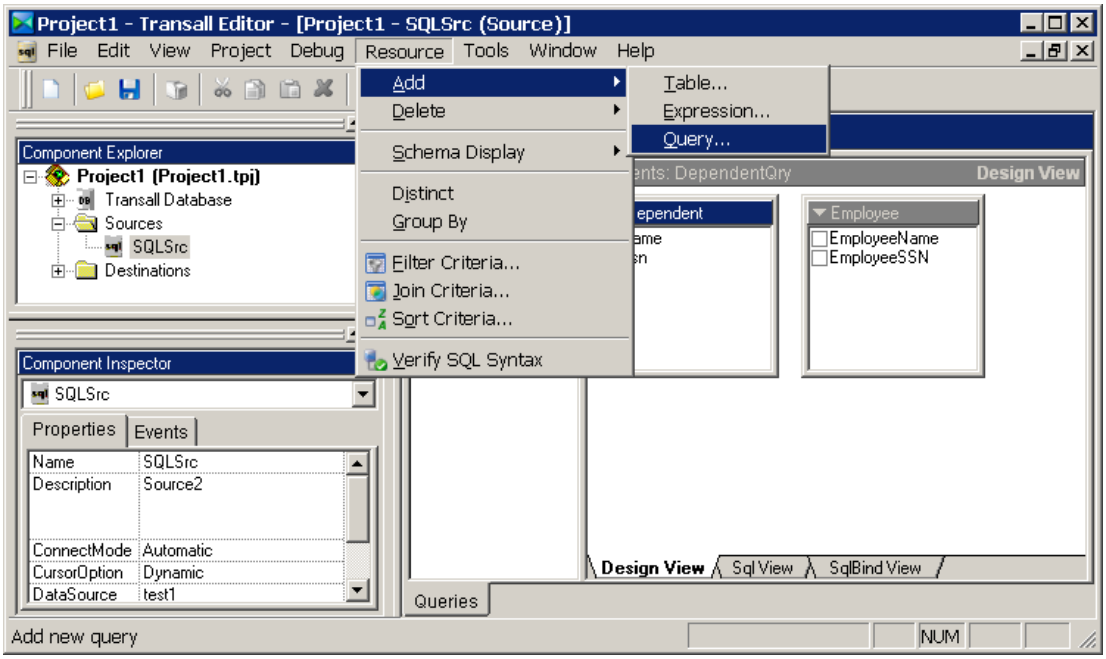


Figure 226: Resource>Add>Query Menu

With the data Source or Destination's Assistant window open, select the **Resource>Add>Query** menu item.

Note If the **Resource>Add** menu doesn't display the options as active, place the mouse pointer in the DataSource Assistant and click the left mouse button once. This action makes the Assistant the target focus. Now, selecting the **Resource>Add** menu displays active options. You can also add a query by placing the mouse pointer in the Query List window, then click and release the right mouse button. This will open a menu for adding a query.

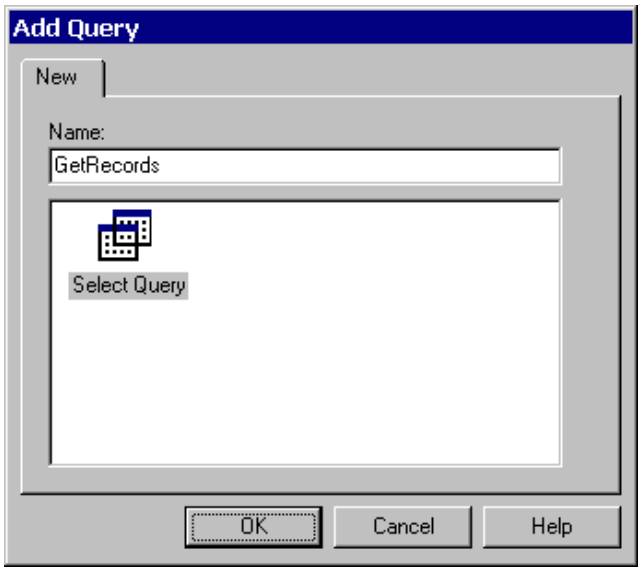


Figure 227: Add Query Dialog Window

This will display the Add Query dialog window. In the dialog window, enter a meaningful name for your query. Press the "OK" button to complete the query addition.

Note The name cannot contain special characters or spaces but can contain underscores "_" and dashes "-".

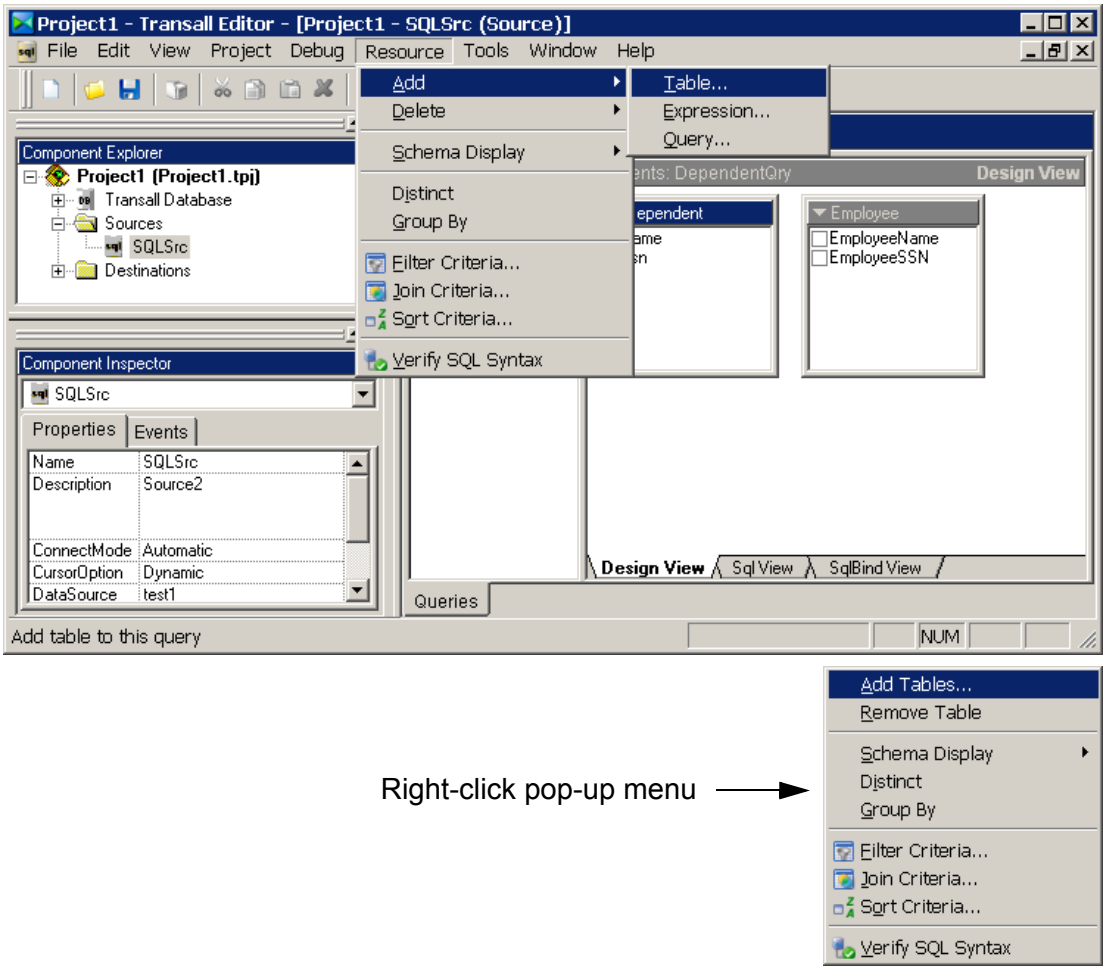


Figure 228: Add Tables Menu

After adding a query, you will see the query listed in the data source or destination's DataSource Assistant in the Query List view. At this point, you have created a placeholder for a database query. To define the query's contents, click on the new query's name in the Query List view. Then click on the **Resource>Add>Table** menu item or right mouse click in the Contents view and select **Add Tables**.

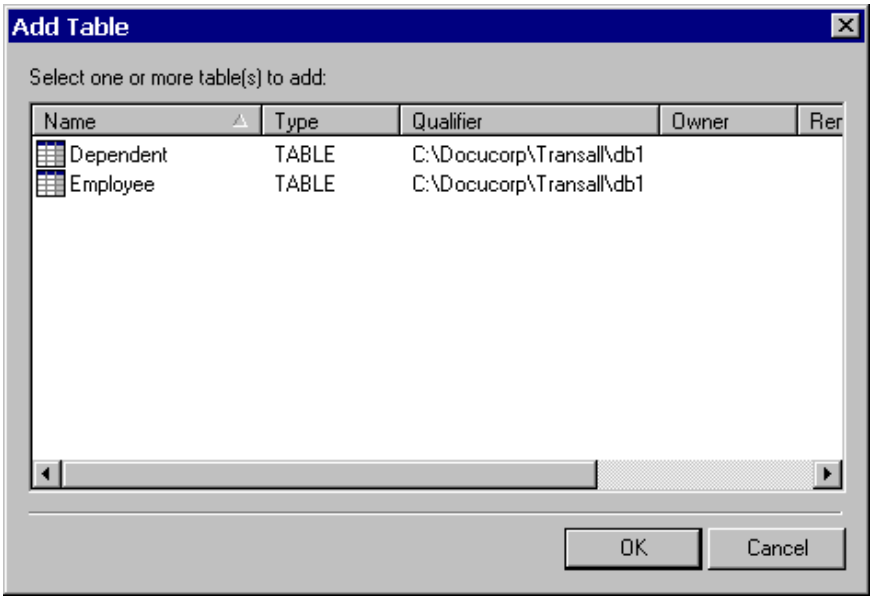


Figure 229: Add Tables Dialog Window

This will display the Add Table dialog window. This dialog window lists all the tables and views available in the database that the SQL data source or destination is connected to via ODBC. Select a table or view or a set of tables and views to be included in the database query. Press the "OK" button to complete the table add.

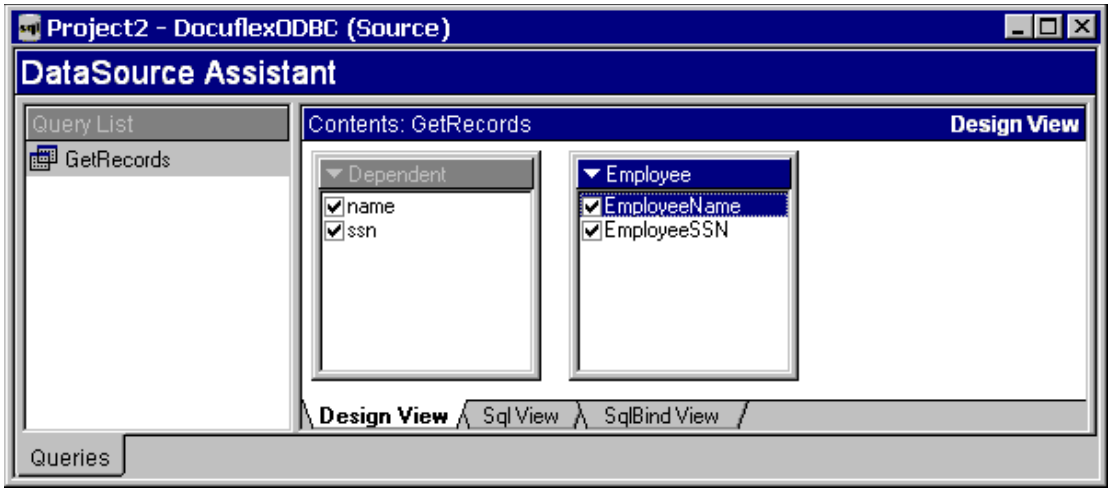


Figure 230: DataSource Assistant with Query added

After the tables and views have been selected, they will appear in the Data source Assistant's editor window as little "Table" windows. One little Table window will appear for each table and view selected. These little windows show the fields available from the table or view. Click on the fields in the table windows to place a check mark next to the fields that you want returned for a data source or sent for a data destination. At this point, you have created a SQL statement. This will be a Select statement for data sources and either an Insert, Update or a Delete statement for data destinations. You can further qualify the SQL statement with a Where clause, an OrderBy clause or a Distinct directive from the Resource menu.

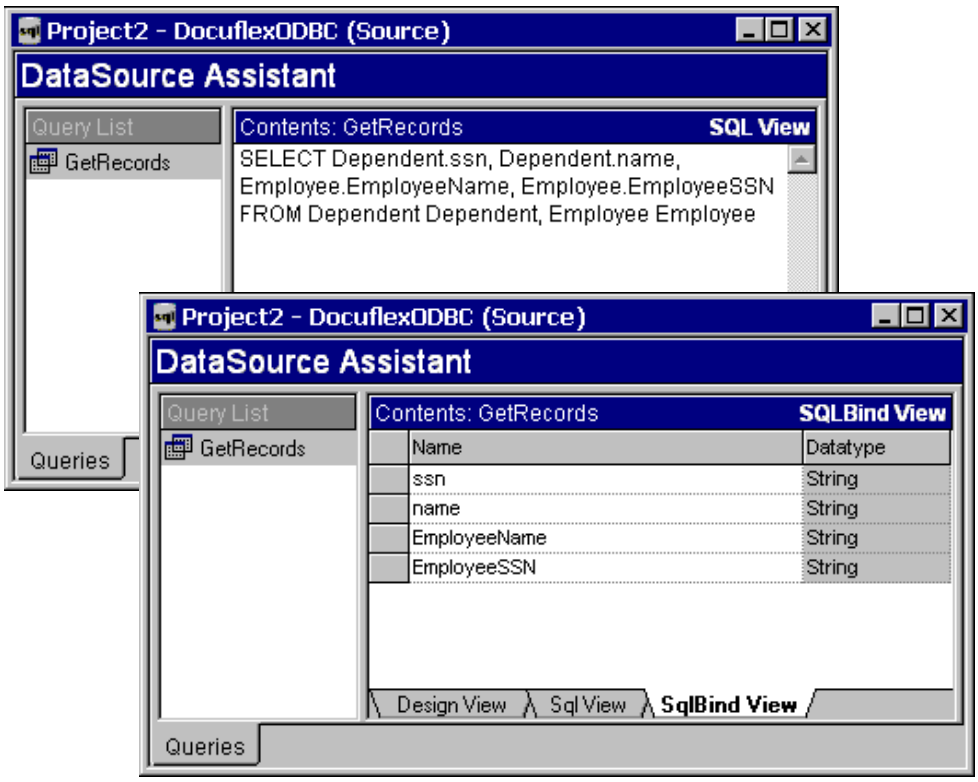


Figure 231: Views of SqlView and SqlBind View Tabs

If you want to view the SQL syntax that Transall is building for you, it is available in the Sql View tab of the DataSource Assistant. The SqlBind View of the DataSource Assistant shows a list of the fields being accessed by the SQL statement and how they are being "bound" to a Transall memory table that is acting as a data value buffer for the SQL statement.

After SQL Query is built, the Component Inspector will show extra details or properties that were not available for setting in the Add Query dialog window.

These extra properties include the following:

Command:

PrepareAndExecute *- (default) Transall will generate logic to run the SQL statement via separate Prepare and Execute ODBC calls.*

ExecuteDirect *- Transall will generate logic to run the SQL statement via a single ExecuteDirect ODBC call.*

Connect:

PrivateCursor *- (default) Transall will generate logic to create a single database connection with a private statement handle for executing this SQL statement. This connection often provides the most capabilities while conserving resources with most ODBC drivers.*

Private *- Transall will generate logic to create a private database connection and statement handle for executing this SQL statement. This option should be used when more than one SQL statement needs to be active at the same moment in time and the ODBC driver being used does not support multiple active statements on a single database connection.*

Shared *- Transall will generate logic to create a shared database connection and a shared statement handle for executing this SQL statement. This option can be used when it is not necessary to have more than one SQL statement active at the same moment in time and system resources need to be conserved.*

Format:

Standard *- (default) Transall will prebind this SQL statement to Transall memory tables at compile time for speed.*

Dynamic *- Transall will not prebind this SQL statement to Transall memory tables. This would be used when the SQL statement string needs to be built dynamically at execution time. Very few ODBC drivers require this option.*

TableQualifier:

none *- (default) Transall will not generate a SQL statement with table qualified syntax.*

Owner *- Transall will generate a SQL statement with table qualified syntax.*

Type:

Select *- (sources only) Transall will generate a SELECT SQL statement.*

Insert *- (destinations only) Transall will generate an INSERT SQL statement.*

Update -	*(destinations only) Transall will generate an UPDATE SQL statement.*
Delete -	*(destinations only) Transall will generate a DELETE SQL statement.*

Have One SQL Statement Reference the Results of Another

When setting up SQL queries in Transall it can be desirable to have one SQL statement reference a result field of another statement or another data source. This is not the same as a table join in the query itself. Instead, this is a situation when you want to use data from another query, maybe even another data source altogether as part of a SQL statement's where clause. For example, this technique would be used when using the results of reading a file to load the where clause of a SQL query. Transall supports "binding" data values from other data sources to a SQL query through the Transall expression builder. When using the expression builder to construct a where clause, the builder will use two types of syntax depending on the type of data selected to be part of the where expression. If the data selected is from one of the tables included in the SQL Select then Transall will use regular SQL syntax to build the where expression. This is because a reference to the tables included in the SQL Select is really a table join. If the data selected is from outside the query, like from another query or from another data source then Transall will use data binding syntax to construct the where expression. The difference in the syntax is a colon ":" placed just in front of the referenced data item. The colon indicates that this data item is being "bound" to the SQL statement from outside the SQL statement. Bound variable references are resolved to data values when the SQL statement is executed at run-time.

SETTING UP A DELIMITED FILE DATA SOURCE

To setup a delimited file data source, which is a file whose records and fields are delimited by some value (usually carriage-return/line-feeds for records and commas for fields), start the Transall editor (TRANEDIT.EXE) and select a Transall project for editing or start a new Transall project.

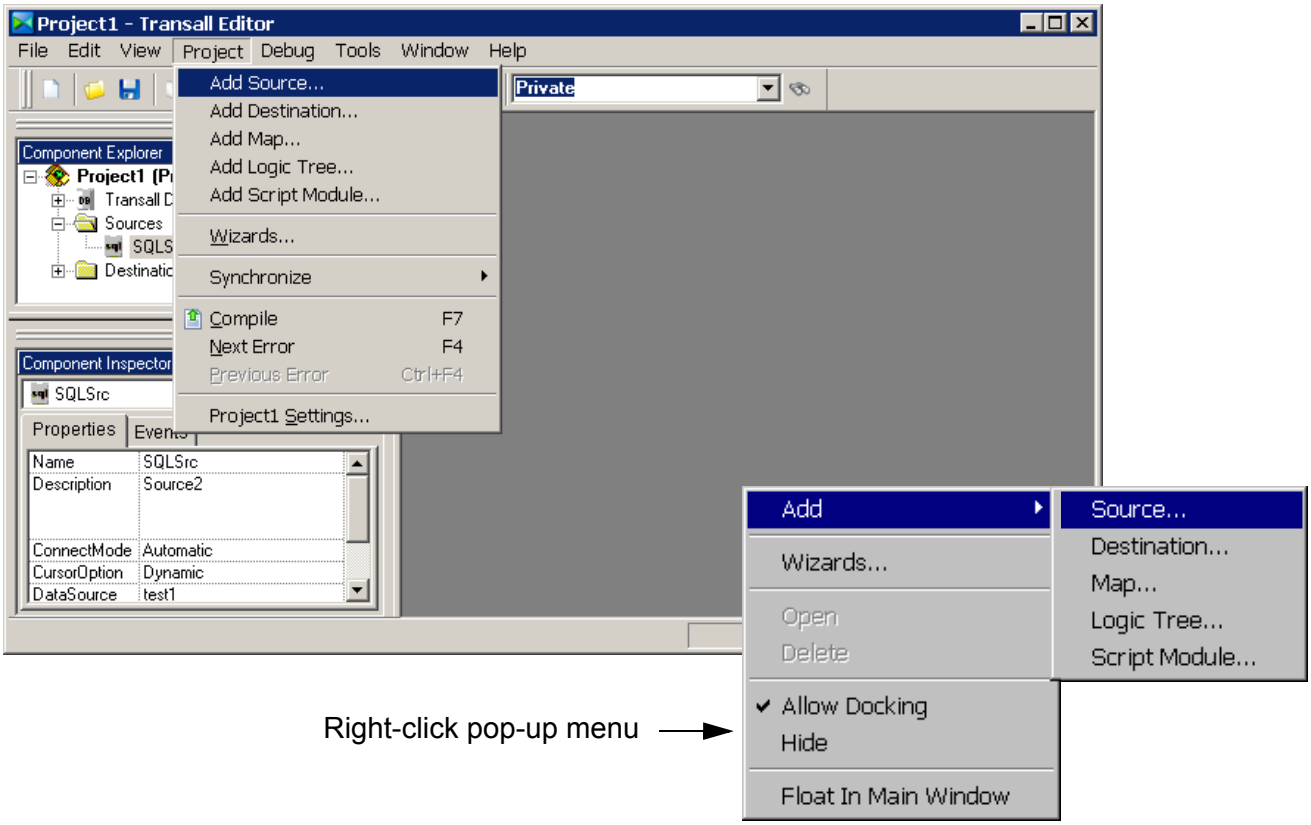


Figure 232: Add Source Menu

Click on the **Project>Add Source** menu item. This will display the Add Source dialog window.

Note By right mouse clicking and releasing in the Component Explorer will display a menu that will allow adding a source.

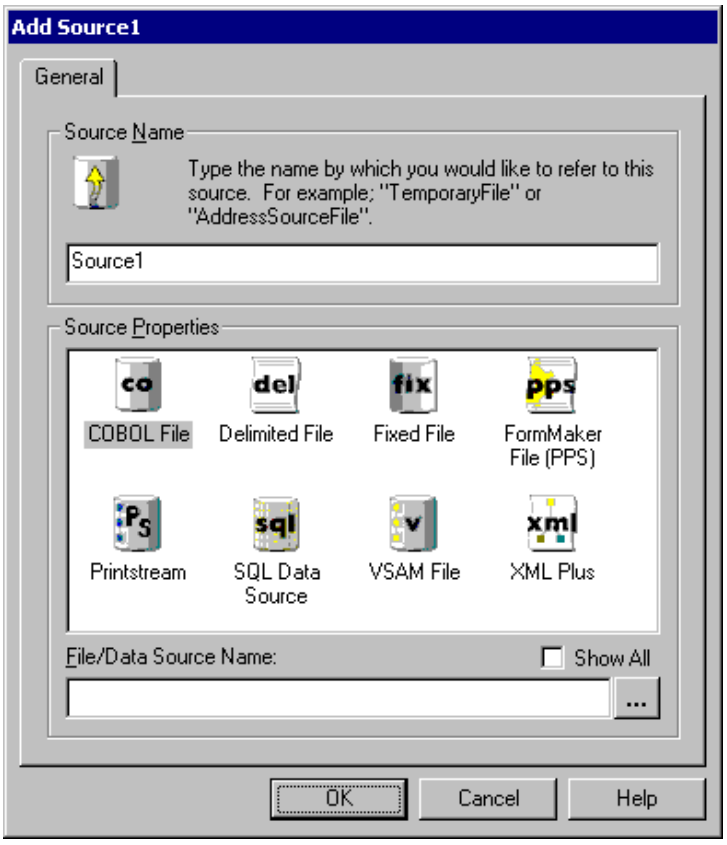


Figure 233: Add Source Dialog Window

In the dialog window, update the Source Name to something meaningful for the data source.

Note You cannot use special characters or spaces in the name of a data source but you can use underscores "_" and dashes "-". It is often a good idea to suffix your data source names with a meaningful value like "-Src" or "-ScHo".

After typing in a source name, select the type of source from the Source Properties list. For delimited files, select “Delimited File”. Last, select a file name for this source. Either type in the file name or click on the button with three ellipses to display the Select File dialog window, from this dialog window either select or type in a new file name. After selecting a file name click the “OK” button in the Add Source dialog window to complete the delimited file source creation.

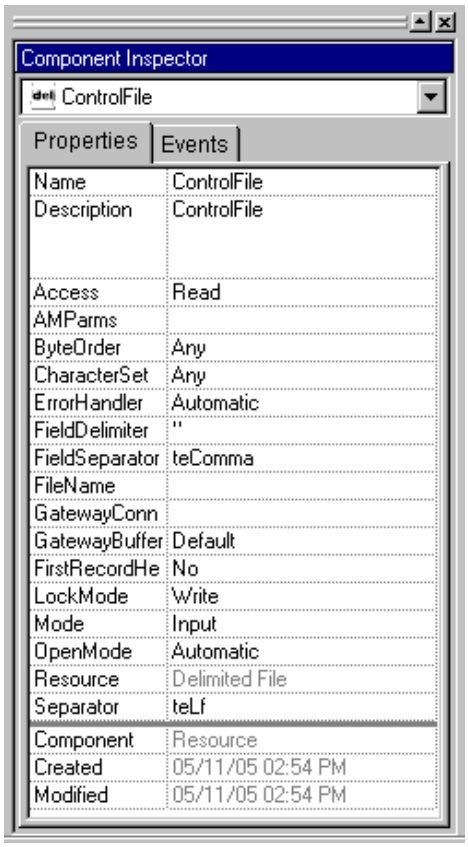


Figure 234: Component Inspector with Properties Shown

You will see your new data source listed in the Component Explorer under the Sources component branch. Also, the Component Inspector will show the details of your new delimited file data source. The Component Inspector will show extra details or properties that were not available for setting in the Add Source dialog window.

These extra properties include the following:

Access:

Read *-* *(default) If this is a new file Transall will create it with read access.*

Read Write *-* *If this is a new file Transall will create it with read/write access.*

ByteOrder:

Any *-* *(default) Transall will assume any binary data being accessed in this file is in the native binary format of the platform Transall is running on.*

Little-Endian - *Transall will assume any binary data being accessed in this file is in little endian (least significant bits in) format and Transall will perform any binary manipulation required to process the data on the platform that Transall is operating on. Note: data on the WIN32 platform is Little Endian.*

Big-Endian - *Transall will assume any binary data being accessed in this file is in big endian (most significant bits in) format and Transall will perform any binary manipulation required to process the data on the platform that Transall is operating on.*

CharacterSet:

Any - *(default) Transall will assume string data being accessed in this file is in the native character set of the platform Transall is running on.*

ASCII - *Transall will assume string data being accessed in this file is in the ASCII character set and Transall will perform any string manipulation required to process the data on the platform that Transall is operating on.*

EBCDIC - *Transall will assume string data being accessed in this file is in the EBCDIC character set and Transall will perform any string manipulation required to process the data on the platform that Transall is operating on.*

Unicode encodings *One of these encodings is specified: UTF-8, UTF-16, UTF-16LE, UTF-16BE, UCS-2, UCS-2LE, UCS-2BE, UCS-4, UTF-32, UCS-4LE, UTF-32LE, UCS-4BE, UTF-32BE, UTS-6*

Windows code pages *One of these code pages is specified: W_CENTRAL_EUROPE, W_CYRILLIC, W_LATIN1, W_GREEK, W_LATIN5, W_HEBREW, W_ARABIC, W_BALTIC, W_VIETNAMESE, W_THAI, W_JAPANESE, W_KOREAN, W_S_CHINESE, W_T_CHINESE*

DOS code pages *One of these code pages is specified: D_USLATIN, D_ARABIC1, D_GREEK, D_BALTIC, D_LATIN1, D_LATIN2, D_CYRILLIC, D_TURKISH, D_LATIN1EURO, D_PORTUGUESE, D_ICELANDIC, D_HEBREW, D_CANADIANFRENCH, D_ARABIC, D_NORDIC, D_CYRILLICRUSSIAN, D_GREEK2, D_THAI, D_ARABICASMO*

ISO code pages *One of these code pages is specified: ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, ISO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, ISO_8859_11, ISO_8859_13, ISO_8859_14, ISO_8859_15, ISO_8859_16*

Other code pages *One of these code pages is specified: O_KOI8R, O_KOI8U, O_KOI8RU, O_KOI8UNI, O_BIG5,*

O_GB12345, O_GB2312, O_JIS0201, O_JIS0208, O_JIS0212, O_JOHAB, O_KSC5601, O_KSX1001, O_WANSUNG, O_GB18030

EBCDIC code pages *One of these code pages is specified: E_DFXDEFAULT, E_USCANADA, E_LATIN5TURKISH, E_INTERNATIONAL, E_GREEK, E_HEBREW, E_ROECELATIN2, E_JAPANESEKATAKANA_EX, E_ARABIC, E_KOREAN_EX, E_CYRILLIC_RUSSIAN, E_LATIN1_EURO, E_CYRILLIC_S_EUROPE, E_USCANADA_EURO, E_GERMANY_EURO, E_DENMARKNORWAY_EURO, E_FINLANDSWEDEN_EURO, E_ITALY_EURO, E_SPANISH_EURO, E_UK_EURO, E_FRANCE_EURO, E_INTL_EURO, E_ICELAND_EURO.*

Note On the Unicode encodings, a suffix of *LE* means Little Endian and *BE* means Big Endian and this refers to text storage. Refer to the ByteOrder property for specifying the storage of binary numbers.

ErrorHandler:

Automatic *-* *(default) Transall will automatically handle and recover from errors. Transall will "throw" only critical errors that must be "caught" via an On Error Resume Next type statement.*

Manual *- Transall will ignore all errors and return errors to your Scripts or LogicTrees for handling.*

FieldDelimiter:

This is either a single character or a string of characters that Transall will look for in the data fields to locate each field's data value. The default for this is a double quote. If the incoming data has no double quotes delimiting a field's value, Transall will still correctly process the field.

FieldSeparator:

This is either a single character or a string of characters that Transall will look for in the data records to locate each data field. The default for this is a comma.

FirstRecordHeader:

No *-* *(default) Transall will not treat the first record as a header record.*

Yes *-* *Transall will treat the first record as a header record causing Transall to skip this record.*

LockMode:

Write *-* *(default) Transall will set the file's access permission to lock out other processes from writing to the file while Transall has the file open for reading.*

Read *-* *Transall will set the file's access permission to lock out other processes from reading the file while Transall has the file open for reading.*

Read Write *-* *Transall will set the file's access permission to lock out other processes from having any access to the file while Transall has the file open for reading.*

Shared *-* *Transall will NOT set the file's access permission to lock out other processes from having access to the file while Transall has the file open for reading.*

Mode:

Input *-* *(default) Transall will open the file for input access in "cooked" or translated mode. In this compatibility mode, carriage-return/line-feed combinations are converted to just line-feed symbols. This yields a common compatibility with UNIX file systems.*

Binary Input *-* *Transall will open the file for input access in "raw" or untranslated mode. In this mode, carriage-return/line-feed combinations are NOT converted to just line-feed symbols.*

OpenMode:

Automatic *-* *(default) Transall will automatically generate logic to open to this file for you.*

Manual *-* *Transall will generate logic to open to this file but will not automatically call the logic. You must manually place a call to the generated logic in your Transall Scripts or LogicTrees to have Transall run the instructions that open this file. You might want to select Manual if you need to have extra control over when a file is opened by Transall. This may be the case if you have a need to open and close a file several times during a Transall run. By default the Automatic open mode opens the file and holds the file open for the life of the Transall run, closing the file when Transall is ready to terminate.*

Separator:

*This is a reference to a Transall resource (setup under **Tools>Options>Separators**) that is either a single character or a string of characters that Transall will look for in the file to locate each data record.*

Why There Are Multiple Records for Some File Data Sources or Destinations

Each record defined to a delimited file data source or destination describes a grouping of fields that can occur in the file. Many files have multiple record types (i.e. many different groupings of fields that might appear in them). For Transall to process each record type, a corresponding record needs to be setup in the delimited file data source or destination. Therefore, if you have a file with three record types, you would setup a delimited file data source or destination with three records, one for each record type. When Transall is reading records from a file it will also look for a special field in each record that serves as a record type identifier. For files that have only one record type, no identifier needs to be defined, since all the records are the same. But, for files that have many record types, Transall needs a field to act as a record identifier that tells Transall at run-time what record type it has read so that Transall can map the file record's data to the correct Transall record format for use by your Transall logic.

Adding a Record Type to a Delimited File Data Source or Destination

To add a record type to a Transall delimited file data source or destination you must first locate or setup a new delimited file data source or destination for the record. Once a delimited file data source or destination has been selected, open the data source or destination's Assistant window by double clicking on the desired data source or destination in the Component Explorer

.

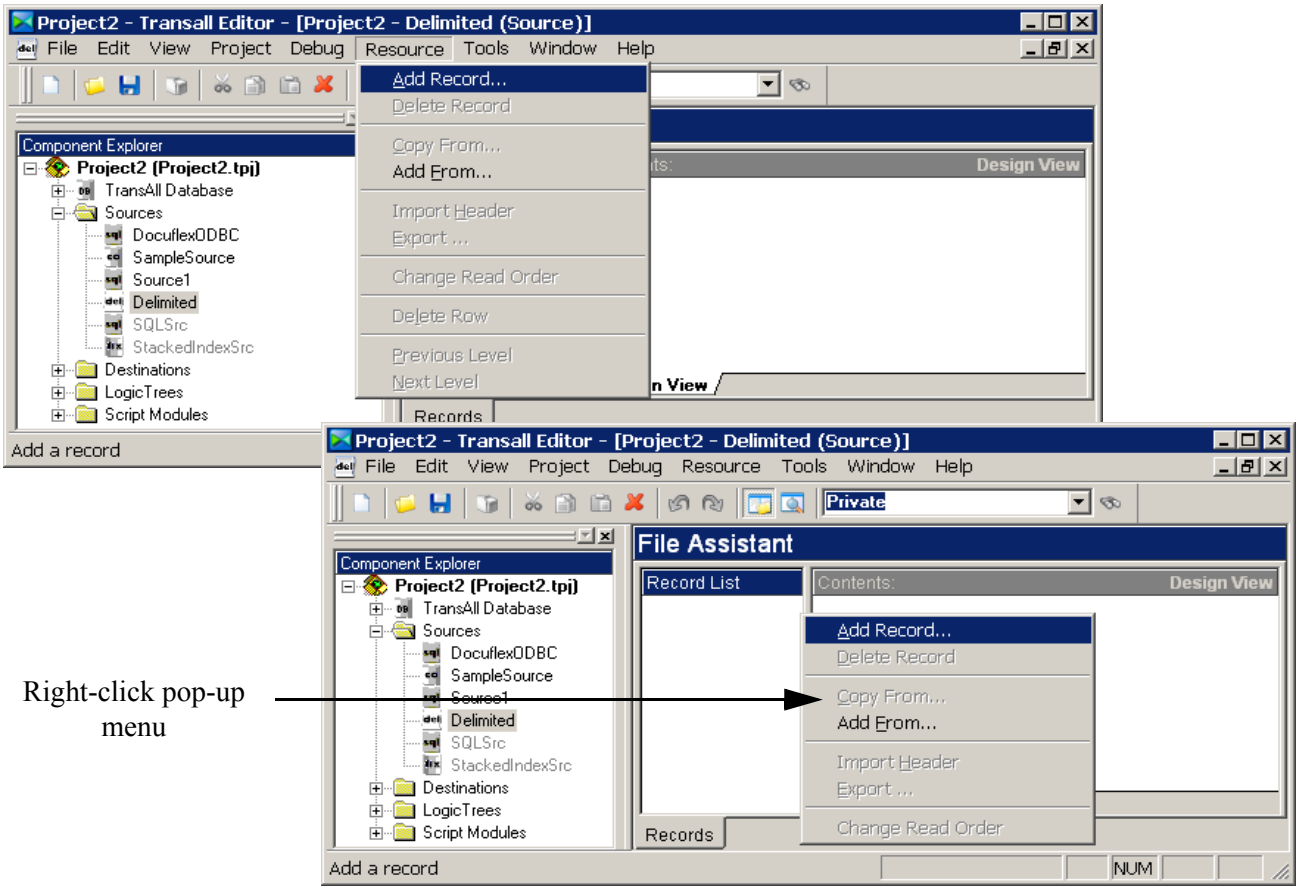


Figure 235: Add Record Menus

With the Assistant window open, select the **Resource>Add Record** menu item or right mouse click in the Records List of the File Assistant.

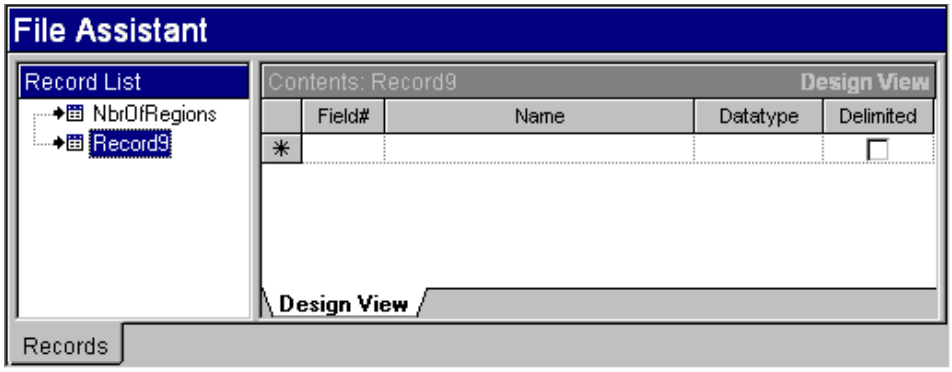


Figure 236: Adding New Record

This will add an empty record to the delimited file's definition.

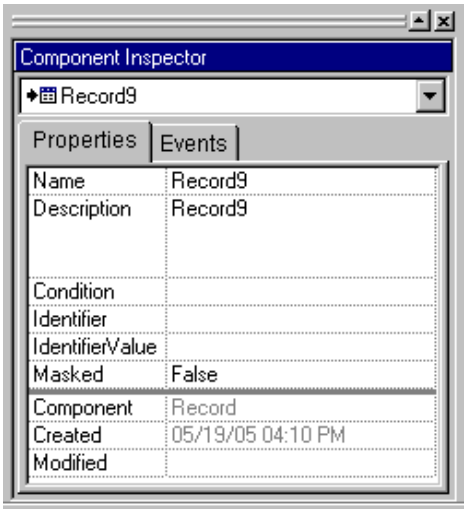


Figure 237: Properties Tab on Component Inspector

After adding a record, change its name on the Properties tab of the Component Inspector to a meaningful name for the record.

Note The name cannot contain special characters or spaces but can contain underscores "_" and dashes "-".

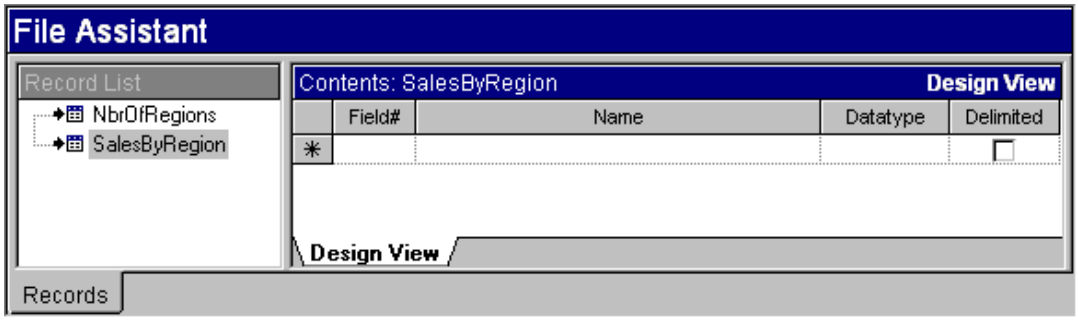


Figure 238: Adding Name Field to New Record

Once the record has been setup you can start adding fields to the record. To add a field, enter the field name into the Name column of the File Assistant on the field row with a star in the row border.

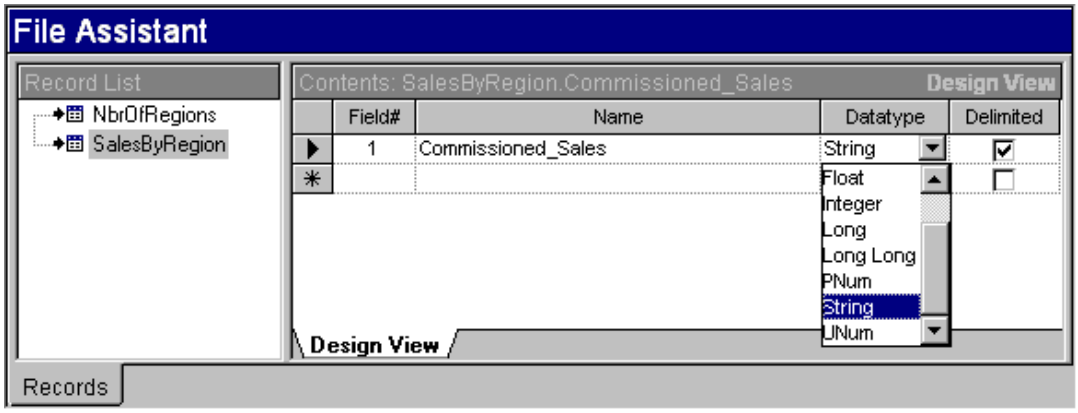


Figure 239: Adding Datatype to New Record

Press the tab key to move to the Datatype column and select a data type for the field. Press the tab key again to set or clear the Delimited indicator for the field.

Note The delimited indicator tells Transall to respect or ignore the FieldDelimiter value set at the data source or destination level for this field. If this value is set to true (checked) and the field's data does not have a field delimiter, Transall will still processing the field's data correctly so it is generally best to leave this field checked. Setup one field for each field in the record type that Transall can expect in the file. The fields need to be in the same order as the fields in the file.

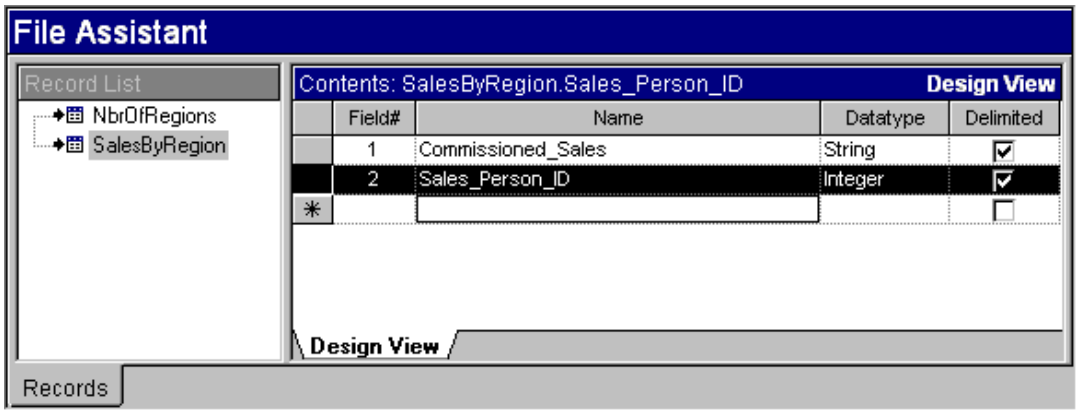


Figure 240: Selecting Field Row

If you need to move a field up or down you can do so by clicking in the gray area to the left of the field's number column in the File Assistant. This will select the field's row in the File Assistant.

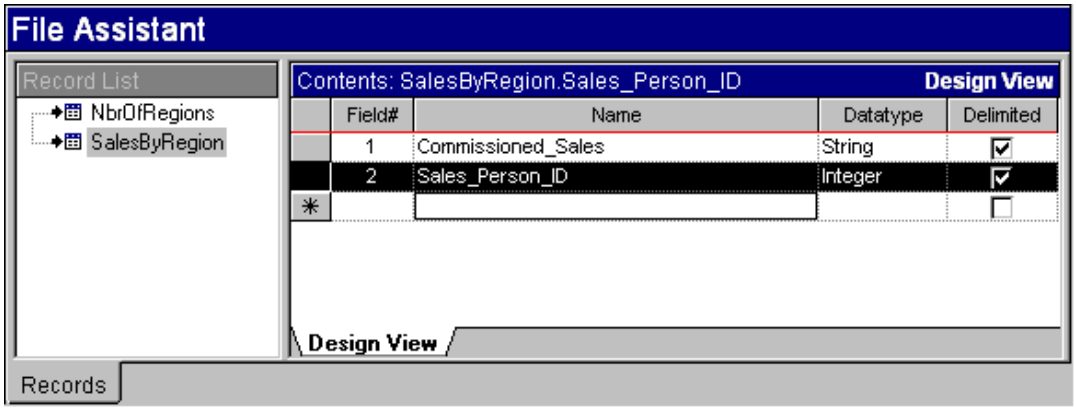


Figure 241: Move Field Row to New Position

With the row selected, you can drag the field from the gray area to the left of the field's number column to a new location in the list of fields for the record. You will see a colored line in the list of record fields as you perform the drag that indicates where the field will go when you drop the field.

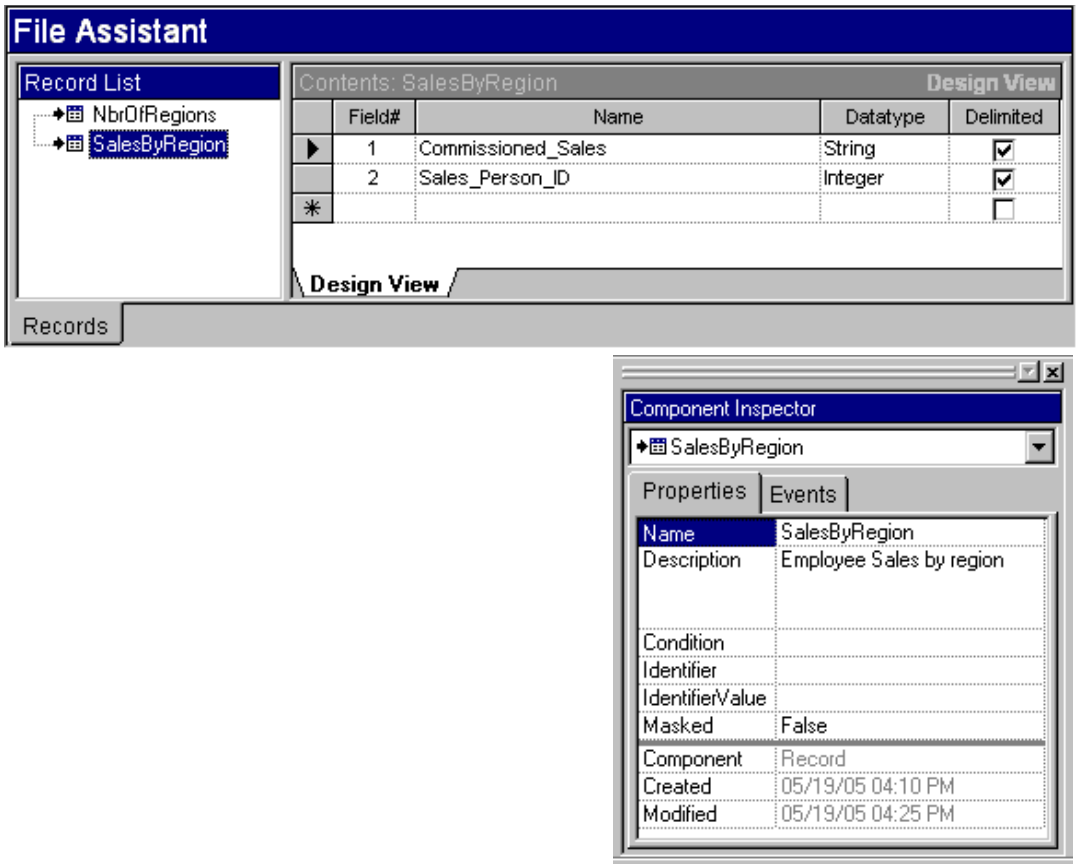


Figure 242: Select Field Row to View Properties in Component Inspector

After adding all the fields to the record and arranging the field order, click on the record name in the File Assistant. This will cause the record's properties to be displayed in the Component Inspector.

If the data source or destination has more than one record, a record ID field needs to be defined. To define a record ID field, select a field in the Identifier field property of the Component Inspector and then set the IdentifierValue property in the Component Inspector. At run-time Transall will look in this Identifier field for the IdentifierValue value to recognize data from the file as belonging to this record.

Transall Threaded Data Manager

INTRODUCTION

This document describes the Transall Threaded Data Manager (TDM) facility, a component of the Transall product suite that enables horizontal process scaling for Docuflex and Transall applications in support of high-volume publishing and data processing systems. The TDM enables multi-process concurrency, coordination, and load balancing for the Docuflex and Transall product suites.

The TDM enables multiple Docuflex and Transall applications running Transall's Host execution engine to cooperatively work together by managing sequential file read and write Input/Output from Docuflex and Transall processes to a set of data files shared by the processes and managed by the TDM.

This facility has been implemented on the Windows and UNIX platforms supported by Transall.

COMPONENTS

The TDM consists of the following components:

Component	Explanation
trandman	The TDM server component that manages sequential file read and write Input/Output over a TCP/IP socket to a set of data files shared by Docuflex and Transall processes.
tdmwait	An application that enables a main script's processing to wait while detached processes it started all run at the same time (as illustrated in the following batch file).

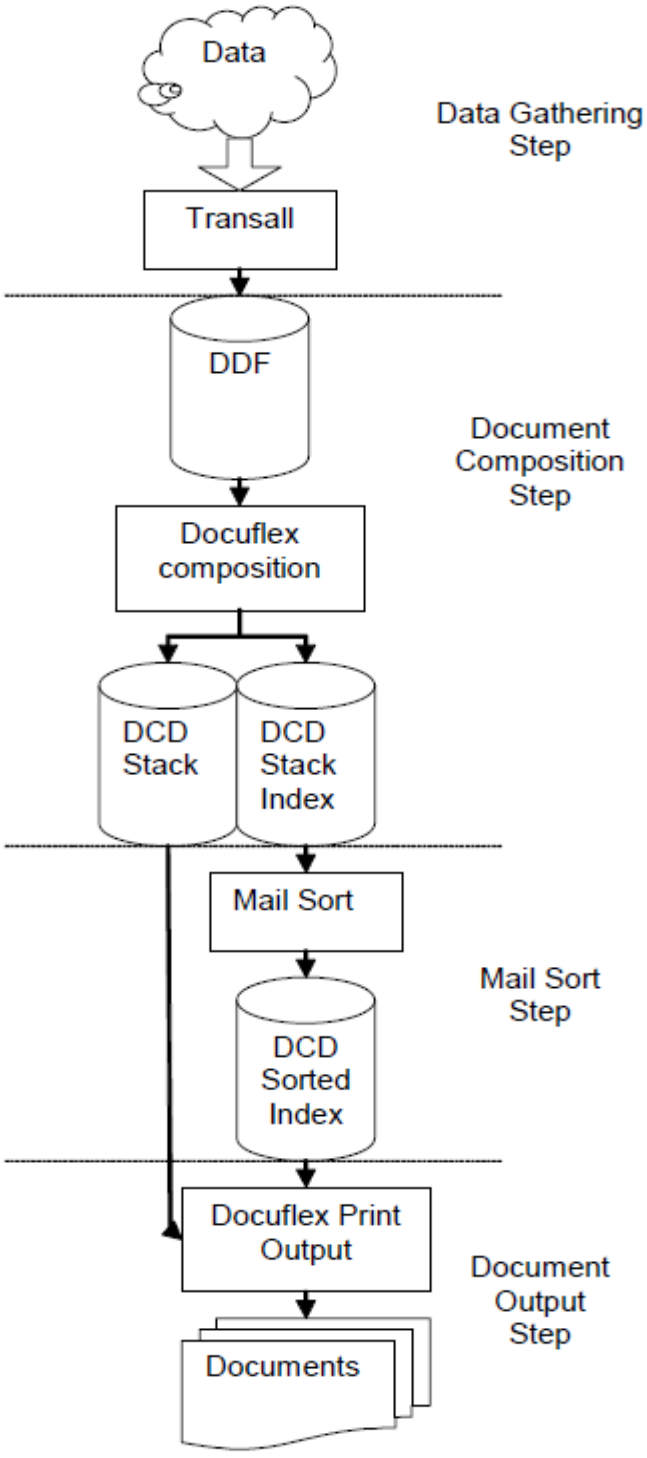
```
rem Start four concurrently-running copies of batch dflx32
start dflx32 tdm=::job
start dflx32 tdm=::job
start dflx32 tdm=::job
start dflx32 tdm=::job
rem Wait while the four running copies of batch dflx32 do their work
tdmwait ::job dflxout
```

tdmjobs	An administrative application that queries the **trandman** component to provide a list of active connections to the TDM server.
tdmterm	A program that asks a TDM server (trandman) to terminate when all processing is complete on the server. You can use this facility in batch processing, that utilizes the TDM, to accomplish the following: • start the TDM server at the top of the batch process • stop the TDM server at the end of the batch process before the process exits

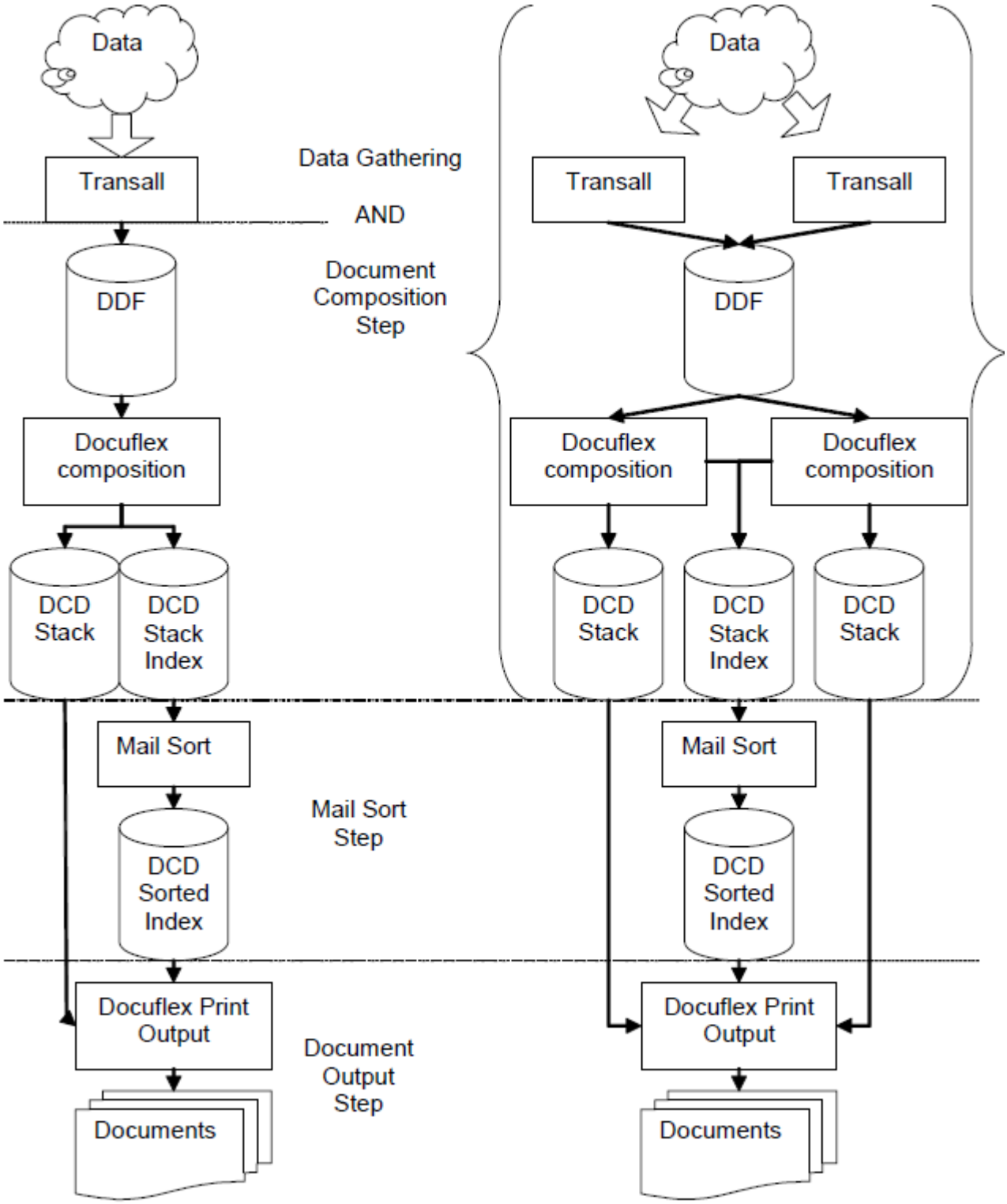
Component	Explanation
Transall Host	The Transall Host execution applications: **tranexe**, **tranhost, trandymn**, and **tranrule**. The Transall Host also includes the Docuflex batch processing applications that imbed the Transall host: **dlfx32**, **dflxux**, and **dflxout**. All these applications are now enabled to have their sequential file read and write Input/Output managed by the TDM server.

EXAMPLES

The Docuflex publishing process usually has the following steps, that are performed in order, in a single “thread” of processing:


Data
Data Gathering Step
Transall
DDF
Document Composition Step
Docuflex composition
DCD Stack
DCD Stack Index
Mail Sort
Mail Sort Step
DCD Sorted Index
Docuflex Print Output
Document Output Step
Documents

With the TDM facility, these steps can be processed by multiple copies of Transall or Docuflex concurrently (e.g., the data gathering process “thread” can run at the same time as the Document Composition process “thread”)


Data
Transall
DDF
Docuflex composition
DCD Stack
DCD Stack Index
Data Gathering
AND
Document Composition Step
Data
Transall
Transall
DDF
Docuflex composition
Docuflex composition
DCD Stack
DCD Stack Index
DCD Stack
Mail Sort
DCD Sorted Index
Mail Sort Step
Mail Sort
DCD Sorted Index
Docuflex Print Output
Documents
Document Output Step
Docuflex Print Output
Documents

CONNECTING TRANSALL AND DOCUFLEX TO THE TDM

To enable Transall and Docuflex to use the TDM you must:

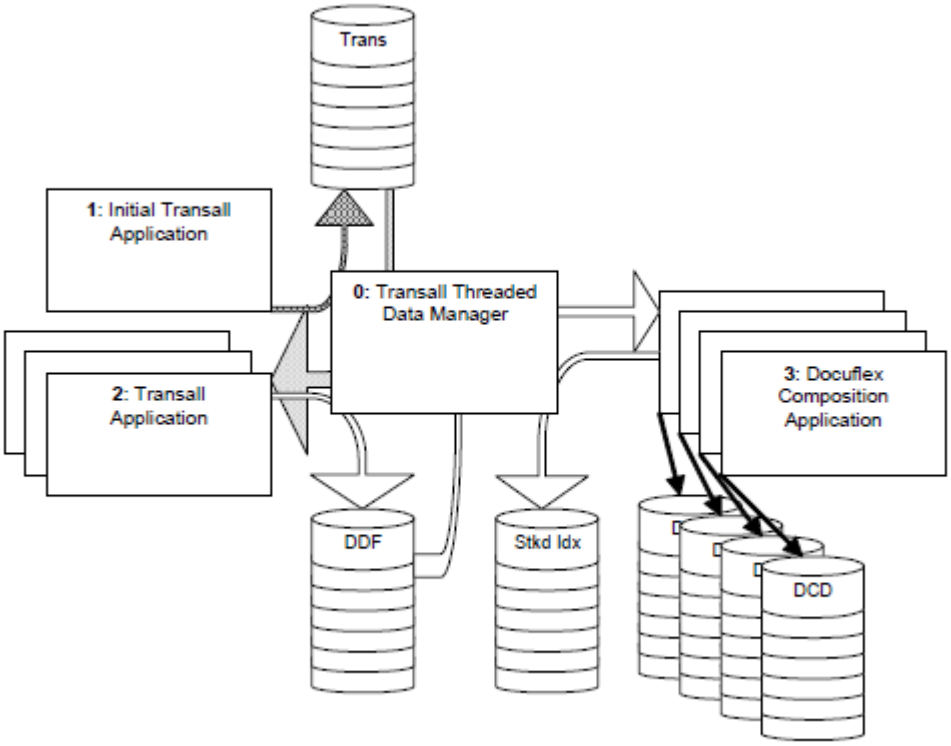
- set a "ThreadedDatMgrSupt" property to "Enabled" for the Docuflex DDF Data Destination in Transall (see *Setting up Transall Projects to use the TDM* on page 401).
- pass a new command line parameter, "/TDM ::", to the Transall Command Line host (see *Command-Line Reference* on page 395).
- pass a new command line parameter, "TDM=::", to Docuflex composition (see *Command-Line Reference* on page 395).

Note These are the minimum updates needed to enable TDM cooperative processing in existing Transall and Docuflex applications.

ADVANCED EXAMPLE OF DATA GATHERING AND DOCUMENT COMPOSITION

Here is another, more robust, example showing how a high-volume Docuflex client might implement the TDM to maximize data gathering and document composition throughput. The numbers in the process boxes represent the startup order for the job's processes.

:



Each of the job's processes will be started "almost simultaneously" from a command script or batch file driving the job. These processes start almost simultaneously because each process will launch using a "start with no wait" command option from the command script.

For this example, it is assumed that Process 0, the TDM facility, is already running as a service or background task.

The batch job starts with Process 1 by starting an initial Transall application process. This application runs a light-weight Transall data extract to create a file that has one record in it for each Docuflex document to be processed by this job. This initial application process does not gather all the details for each Docuflex document; it instead only gathers "key" information for each document.

The batch job continues with Process 2—this process may (and probably will) start before Process 1 completes, because Process 1 was started with a "no wait" command option. This option causes the batch script not to wait for an applications started by the script to complete before proceeding to the next instruction in the script. Process 2 launches three copies of a Transall application that each concurrently read from the file being written to by Process 1. These applications, started by Process 2, gather all the details for each Docuflex document. They work concurrently to write these details to a single Docuflex Data File (DDF) for use by Docuflex composition. None of these copies of Transall, started by Process 2, will read the same record from the file being written to by Process 1, because the file is managed by the TDM facility.

Next, Process 3 of the batch job launches four copies of Docuflex composition. Again, these may start before the Transall applications started by Process 2 have completed because the Transall applications were started with a "no wait" command option. Each Docuflex composition copy will concurrently read from the DDF file being written by the Transall applications started by Process 2. The Docuflex applications will each receive unique document records from the DDF file because the file is managed by the TDM facility. Each Docuflex composition copy will write composed documents to its own "stacked" Document Compound Document (DCD) file. These DCD files are not managed by the TDM facility, but a single stacked index file for the composition job is, and the TDM will keep the records written to the shared stacked index file from being written on top of each other as the four copies of Docuflex composition run concurrently.

This configuration allows all eight applications, started by the three batch job processes, to run concurrently in a load balanced system.

SAMPLE BATCH COMMAND SCRIPT

Here is a sample Windows batch command script that implements the prior advanced example:

```
@ REM Define parameter file locations
@
@ SET TDM=server_name:5650:AsampleJob
@
@ SET DFLXINI1=c:\DocuCorp\Docuflex\Projects\docuflx1.dde
@ SET DFLXINI2=c:\DocuCorp\Docuflex\Projects\docuflx2.dde
@ SET DFLXINI3=c:\DocuCorp\Docuflex\Projects\docuflx3.dde
@ SET DFLXINI4=c:\DocuCorp\Docuflex\Projects\docuflx4.dde
@
@ REM Start Initial Transall process
@
@ SET TRA=c:\DocuCorp\Docuflex\Projects\sample\samptrst.tex
@ SET DDF=C:\docucorp\docuflex\Projects\sample\sampdata.ddf
@ SET TFL=C:\docucorp\docuflex\Projects\sample\samptran.txt
@
Start "c:\program files\maitland software\tranexe" /TDM %TDM% %TRA% start %TFL%
@
@ REM Give initial process 5 seconds to get running
@
"c:\program files\maitland software\tdmwait" -q -w 5
@
@ REM Start Data Gathering Transall process
@
@ SET TRD=c:\DocuCorp\Docuflex\Projects\sample\samptrdf.tex
@
Start "c:\program files\maitland software\tranexe" /TDM %TDM% %TRD% start %TFL% %DDF%
Start "c:\program files\maitland software\tranexe" /TDM %TDM% %TRD% start %TFL% %DDF%
Start "c:\program files\maitland software\tranexe" /TDM %TDM% %TRD% start %TFL% %DDF%
@
@ REM Start Docuflex processes
@
@ SET DDP=c:\DocuCorp\Docuflex\Projects\sample\sampdflx.ddp
@
Start "c:\docucorp\docuflex\dflx32" dde=%DFLXINI1% ddp=%DDP% ddf=%DDF% tdm=%TDM%
Start "c:\docucorp\docuflex\dflx32" dde=%DFLXINI2% ddp=%DDP% ddf=%DDF% tdm=%TDM%
Start "c:\docucorp\docuflex\dflx32" dde=%DFLXINI3% ddp=%DDP% ddf=%DDF% tdm=%TDM%
Start "c:\docucorp\docuflex\dflx32" dde=%DFLXINI4% ddp=%DDP% ddf=%DDF% tdm=%TDM%
@
@
@ REM Wait for processes to complete
@
"c:\program files\maitland software\tdmwait" -q %TDM%
```

where:

Variable	Meaning
TDM	Connection string to the TDM.
DFLXINI#	Docuflex environment setting file
TRA	Name of the Transall initial data gathering application
TFL	Name of the initial file created by Step 1 that has one record in it for each Docuflex document to be processed by this job. This file contains "key" information for each document
TRD	Name of the Transall detailed data gathering application
DDF	Name of the Docuflex Data File

Variable	Meaning
DDP	Name of the Docuflex Project to be executed by the Docuflex composition process

NAMED JOBS IN THE TDM

The connection string that accesses the TDM is formatted with three sections that are separated by colons—Network address, port number, and Job Name.

Example The connection string "*server_name*:5650:JobOne" means to look for an instance of the TDM running on the computer at the IP address pointed to by the "*server_name*" Domain Name Service (DNS) name on port 5650 and mark this connection with the name "JoOne".

Naming connections enables the **tdmwait** facility to pause a command script or batch file, based on the connections to the TDM with a particular name. Let's say you have two processes—one for producing letters and another for producing statements. Let's also say it's desirable to have both processes run at the same time through the same instance of the TDM. By setting the TDM connection string for both jobs with different Job Name identifiers (e.g., "letters" and "statements"), the command script or batch files used to run the processes can have calls to **tdmwait** that only wait on the TDM connections for their work. Both processes for producing letters and statements can run concurrently, but independently, through a single instance of the TDM.

There is also a facility called **tdmjobs** that lists job names and the number of connections for each job name running on the TDM, so you can check the status of processing as jobs run.

COMMAND-LINE REFERENCE

TRANDMAN

Enter the following syntax on the command line:

```
TRANDMAN [/option -option ...] [port#
default=5650]
```

where:

Option	Meaning
-D	Debug mode, lists server actions to console for debugging support (causes server to run very slowly).
-INSTSERV	Windows only, installs TDM as a Windows Service.
-L	Opens the TCP/IP socket port on the 'localhost' IP address for the workstation.
-Q	Suppress display of TRANDMAN startup banner.
-REMSERV	Windows only, removes the TDM if it was installed as a Windows Service.
-T	Terminate execution after all clients disconnect.
-W	Wait latency time (default = 4 seconds).

TDMWAIT

Enter the following syntax on the command line:

```
TDMWAIT [/option -option ...] [JobName …]
```

where:

Option or Parameter	Meaning	
-A	Wait on all jobs connected to TDM (ignore JobName parameter).	
-W	Wait a minimum number of seconds.	
-Q	Suppress display of TDMWAIT startup banner.	
JobName	A TDM connection string in the format of: [ServerName]:[PortNumber]:[JobName] -or- JobName where:	
	ServerName	Optional parameter with Domain Name or IP address of a server where the TDM server is running.
	PortNumber	Optional parameter with Port Number on the server on which the TDM is listening.
	JobName	Optional parameter with Job Name of connections of which to wait for completion.

TDMJOBS

Enter the following syntax on the command line:

```
TDMJOBS [/option -option ...] [JobName …]
```

where:

Option or Parameter	Meaning	
-Q	Suppress display of TDMJOBS startup banner.	
JobName	A TDM connection string in the format of: [ServerName]:[PortNumber]:[JobName] -or- JobName where:	
	ServerName	Optional parameter with Domain Name or IP address of a server where the TDM server is running.
	PortNumber	Optional parameter with Port Number on the server on which the TDM is listening.
	JobName	Optional parameter with Job Name of connections for which to list the connection count.

TDMTERM

Enter the following syntax on the command line:

```
TDMTERM [/option -option ...][server][:port]
```

where:

Option or Parameter	Meaning
-Q	Suppress display of TDMTERM startup banner.
server	Domain Name or IP address of a server where the TDM server is running.
port	Port Number on the server on which the TDM is listening.

Transall

Enter the following syntax on the command line:

```
TRANEXE [/tdm
[ServerName]:[PortNumber]:[JobName]]
```

where:

tdm	A TDM connection string in the format of: [ServerName]:[PortNumber]:[JobName] where:	
	ServerName	Optional parameter with Domain Name or IP address of a server where the TDM server is running.
	PortNumber	Optional parameter with Port Number on the server on which the TDM is listening.
	JobName	Optional parameter with Job Name of connections of which to wait for completion.

Docuflex

Enter one of the following syntax statements on the command line:

```
DFLXUX tdm=ServerName:PortNumber:JobName
```

-or-

```
DFLX32 tdm=ServerName:PortNumber:JobName
```

where:

tdm	A TDM connection string in the format of: [ServerName]:[PortNumber]:[JobName] where:	
	ServerName	Optional parameter with Domain Name or IP address of a server where the TDM server is running.
	PortNumber	Optional parameter with Port Number on the server on which the TDM is listening.
	JobName	Optional parameter with Job Name of connections of which to wait for completion.

STARTING THE TDM SERVER IN AUTHENTICATION MODE

The Threaded Data Manager server may optionally be started in authentication mode. When the server is started in authentication mode, Transall and Docuflex client applications must provide the proper authentication string to connect to the server. In addition, the TDMTerm program must provide the authentication string to stop the server. The authentication mode is backward compatible with existing projects, when they are run with newer versions of Transall, so that existing projects do not need to be recompiled to take advantage of this feature.

Authentication mode is enabled on the server either via the **–A** command line option (see Command-Line Reference on page 395) or by setting the TDM_AUTH environment variable to the authentication string before starting the process. Client applications enable authentication mode by either appending "***:authentication_string***" to the connect string or by setting the TDM_AUTH environment variable to the authentication string before starting the application.

The command line references for Trandman, and TDMTerm need to be modified as shown in the following sections.

TRANDMAN

Enter the following syntax on the command line:

```
TRANDMAN [/option -option ...] [port#
default=5650]
```

where:

Option	Meaning
-A AuthCode	Authorization code. All applications must use authorization to connect.
-D	Debug mode, lists server actions to console for debugging support (causes server to run very slowly).
-INSTSERV	Windows only, installs TDM as a Windows Service.
-L	Opens the TCP/IP socket port on the 'localhost' IP address for the workstation.
-Q	Suppress display of TRANDMAN startup banner.
-REMSERV	Windows only, removes the TDM if it was installed as a Windows Service.
-T	Terminate execution after all clients disconnect.
-W	Wait latency time (default = 4 seconds).
port#	Port Number on which the Threaded Data Manager should listen.

TDMTERM

Enter the following syntax on the command line:

```
TDMTERM [/option -option
...][server][:[port][:authCode]]
```

where:

Option or Parameter	Meaning
-Q	Suppress display of TDMTERM startup banner.
server	Domain Name or IP address of a server where the TDM server is running.
port	Port Number on the server on which the TDM is listening.
authCode	(Optional) Authentication code required by server for access.

The Transall and Docuflex command line options in this section need to be changed as shown in the following sections.

Transall

Enter the following syntax on the command line:

```
TRANEXE [/tdm
[ServerName]:[PortNumber]:[JobName][:Authenticati
on]]
```

where:

tdm	A TDM connection string in the format of: [ServerName]:[PortNumber]:[JobName][:Authentication] where:	
	ServerName	Optional parameter with Domain Name or IP address of a server where the TDM server is running.
	PortNumber	Optional parameter with Port Number on the server on which the TDM is listening.
	JobName	Optional parameter with Job Name of connections of which to wait for completion.
	Authentication	Optional authentication string required to connect to server.

Docuflex

Enter one of the following syntax statements on the command line:

```
DFLXUX tdm=ServerName:PortNumber:JobName
```

-or-

```
DFLX32 tdm=ServerName:PortNumber:JobName
```

where:

tdm	A TDM connection string in the format of: [ServerName]:[PortNumber]:[JobName][Authentication] where:	
	ServerName	Optional parameter with Domain Name or IP address of a server where the TDM server is running.
	PortNumber	Optional parameter with Port Number on the server on which the TDM is listening.
	JobName	Optional parameter with Job Name of connections of which to wait for completion.
	Authentication	Optional authentication string required to connect to server.

SETTING UP TRANSALL PROJECTS TO USE THE TDM

A new “ThreadedDatMgrSupt” property has been added to Transall Delimited, Fixed, and COBOL file type Data Sources and Destinations. This property defaults to “Disabled”. To enable a Transall Data Source or Destination to run through the TDM, change the property value for ThreadedDatMgrSupt to “Enabled”. This causes the Transall editor to generate extra code in the Open event script for Data Sources or Destinations to connect to the TDM upon opening the file.

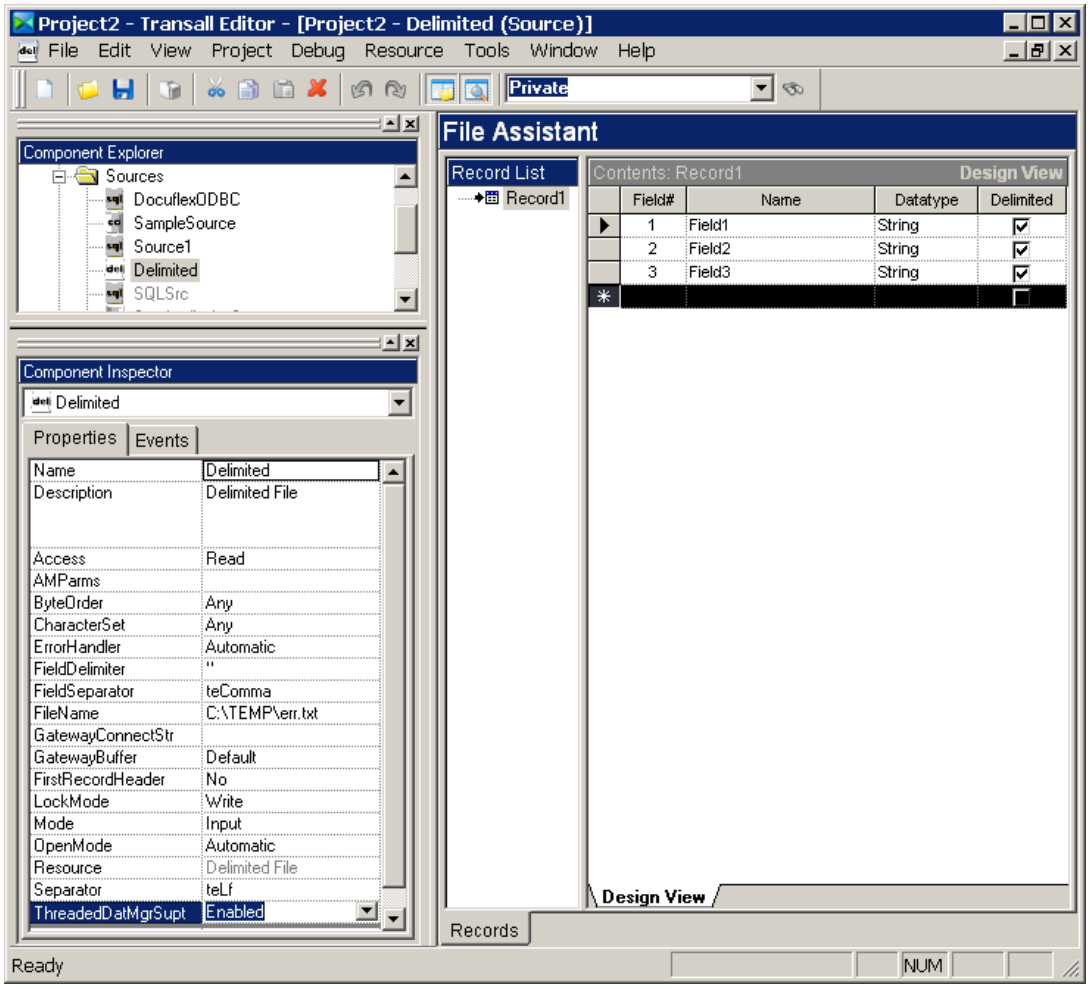


Figure 243: Set ThreadedDatMgrSupt to “Enabled”

Files connected to the TDM continue to work like files that aren’t connected to the TDM (with some restrictions): files that are accessed through the TDM may **not explicitly seek to an offset in the file**. The TDM provides sequential access, either read or write, to files; however, it doesn’t at this time provide random access to files. Because of this limitation, only Data Sources and Destinations with one record type defined are available to have their ThreadedDatMgrSupt property set to “Enabled”.

When the ThreadedDatMgrSupt property is set to "Enabled," the Transall editor generates script in the Open events of Sources and Destinations that will connect access for the Source or Destination to be performed by the TDM, using one of two new Transall system fields.

- TrnSys$TdmConnectCmdLine—this system constant field is populated with the value of the string that was passed to Transall (or Transall running under Docuflex) on the TDM command line parameter.
- TrnSys$*SourceName*.TdmConnectString—this system field is generated for each Data Source or Destination that has its ThreadedDatMgrSupt property set to "Enabled," where *SourceName* is the Data Source or Destination name in Transall.

To have Transall connect a Data Source or Destination to the TDM, either the TDM command line parameter must be passed to Transall, or the new TrnSys$*SourceName*.TdmConnectString field must be populated with a TDM connection string before the Open event script is executed for the Data Source or Destination.

If both the TrnSys$*SourceName*.TdmConnectString and the TrnSys$TdmConnectCmdLine are provided, the TrnSys$*SourceName*.TdmConnectString will be used to connect to the TDM for the Data Source or Destination.

When setting the TDM connection string for an individual Data Source or Destination, a good place to populate the TrnSys$*SourceName*.TdmConnectString value is in the **OnOpenBefore** event of the Data Source or Destination.

The TDM connection string consists of the following parameters:

```
[ServerName]:[PortNumber]:[JobName][Authentication]
```

where:

Parameter	Meaning
ServerName	Optional parameter with Domain Name or IP address of a server where the TDM server is running.
PortNumber	Optional parameter with Port Number on the server on which the TDM is listening.
JobName	Optional parameter with a Job Name label for this connection to the TDM.
Authentication	Optional authentication string required to connect to server.

The TDM connection string can be as simple as "::" which means to use the default

- Server Name (e.g., the current workstation where Transall is running)
- Port Number (e.g., defaults to 5650)
- Job Name (e.g., defaults to nothing)
- Authentication (e.g., uses TDM_AUTH environment setting)

The JobName parameter is important because it tells the TDM that the files open on this connection are part of a job under this name. Using a JobName enables batch command scripts to use the **tdmwait** facility to wait for all connections labeled with a particular job name to complete before proceeding to the next step in the batch command script. To pass just a job name in the TDM connection string, the string will look like "::AJobName". This means to use the default Server Name, the default Port Number, and "AJobName" as the Job Name string.

Transall Gateway

OVERVIEW

The Transall Gateway is a server process which runs on a remote machine and allows
Transall, running on a local machine to access files on the remote machine, even if there is no shared filesystem between them (for instance, between a PC and z/OS). The Gateway server may be run on the following platforms:

- AIX
- Linux
- Solaris
- Windows
- z/OS

A Transall Gateway Server can also be accessed by processes running on any of the above platforms. The Transall Gateway Server runs in either multi-threaded mode (AIX, Linux, Solaris, Windows) or in multi-tasking mode (z/OS) to service requests from remote clients. Clients using the Gateway client interface, which is built into Transall, can carry out an assortment of basic file processing functions on the host Gateway Server where the Gateway is running, including:

- Open files for I/O
- Close files
- Read data
- Write data
- Seek within open files (where supported by platform)
- Get file position via “tell” (where supported by platform)
- Use the “getpos” to obtain the current file position (all platforms)
- Use “setpos” to return to a previous file position (all platforms)
- Remove files
- Get file information (where supported by platform)
- Query system type (Window, Unix, z/OS) where host is running

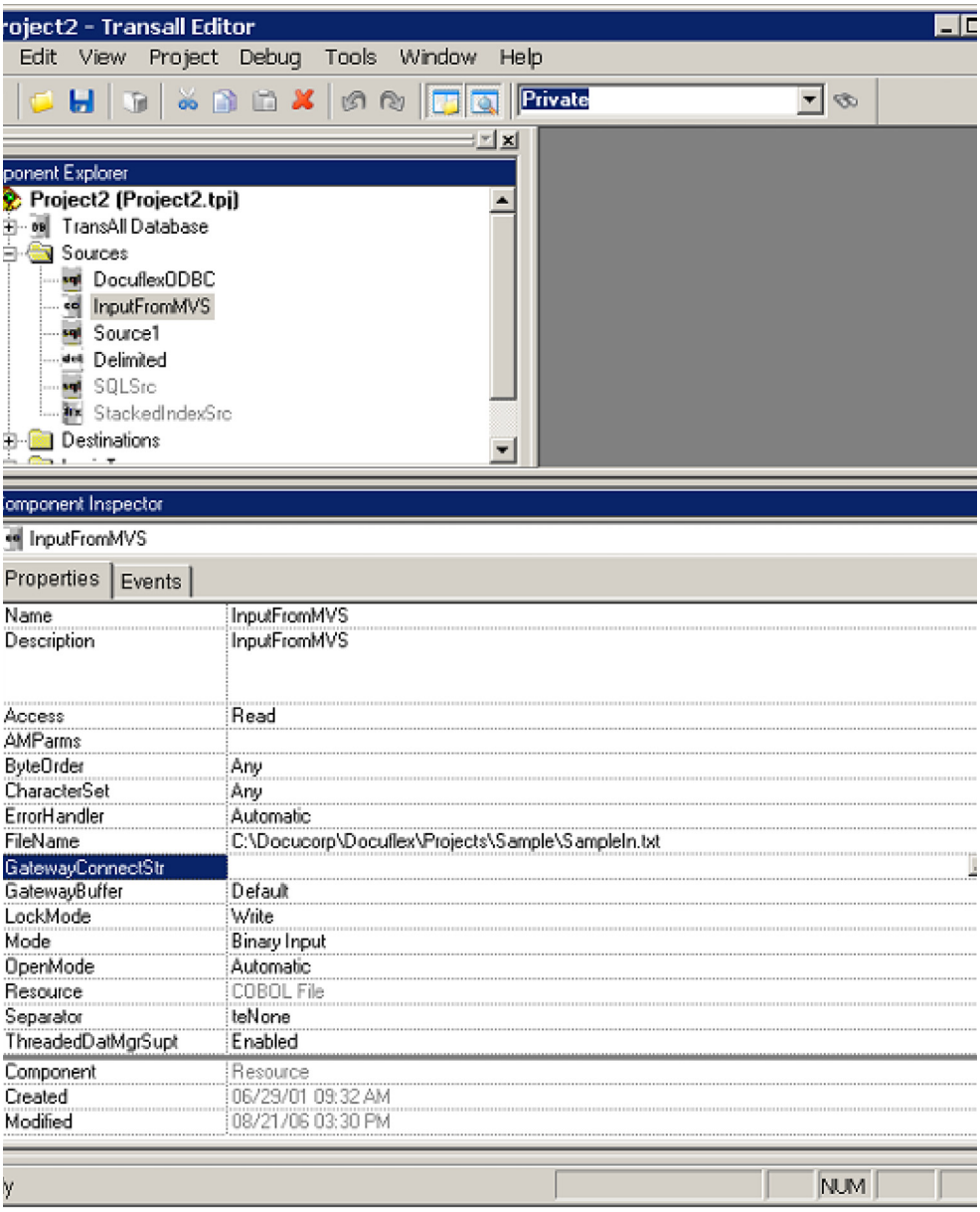
In addition, if the Gateway Server is running on z/OS, the following additional functions are available:

- Open a file using z/OS specific attributes (RECFM, LRECL, VSAM, etc.)
- Search a VSAM file for a record (KSDS)
- Retrieve a record from a VSAM file
- Insert a record into a VSAM file
- Replace a record in a VSAM file
- Delete a record from a VSAM
- Remove a VSAM file

TRANSALL INTERFACE

Transall's scripting syntax and GUI for the Open file statement allow the author of the Transall application to specify a “Gateway” for use in accessing a file remotely. The properties for file data sources and destinations use the property “cx” to identify the Gateway Server that will be used to access the file on the remote system.

The syntax for the open command also provides support for an optional Gateway




```
name:
| Fast |
Open filename [ At gateway [ With | Safe |
Buffering]] ...
| No |
```

The filename may be either a quoted string or a string variable that will contain the name of the file to open when the script is run. Note that if you are opening a file on a z/OS system via a Gateway, the filename specified is not a DDName, but is a fully qualified name, such as “USERID.TEST.FILE” or “USER.NOTES(NOTE1)”. The gateway parameter names a Transall Gateway Server that is used to access the file on the remote system by a Gateway identifier whoes final value will be passed on the Transall command line via a “/gate” option (see: Accessing Files Through a Transall Gateway with Tranexe). The Gateway name may be specified as either a

quoted string, or as a string variable that will contain the name of the Gateway when the script is run. The “With...Buffering” clause is an advanced option that may be used to control how data is buffered as it is read from the Gateway. “Fast” buffering mode allows for faster transfers of data, but is not reliable if the file is going to be accessed randomly, or it is not a binary file. “Safe” buffering is slightly slower, but is reliable for either text or binary files. If the “With...Buffering” clause is excluded, the file is opened in “Safe” mode if it is a “text” file and “Fast” mode if it is a “binary” file. Valid Open statements include things like:

```
Open Source1.FileName At "Billing" For Binary Input
Access Read Lock Write As Source1.Handle
```

(Opens the binary file at the Gateway Server running on the “Billing” machine with fast buffering by default)

```
Open Source1.FileName At "Sales" For Input Access
Read Lock Write As Source1.Handle
```

(Opens the text file at the Gateway on the “Sales” machine with safe buffering by default)

```
Open Source1.FileName At "Logging" With No
Buffering For Input Access Read Lock Write As
Source1.Handle
```

(Opens the text file at the Gateway on the “Logging” machine with no read buffering

Accessing Files Through a Transall Gateway with Tranexe

The command line of Tranexe has been modified to accept the option: “/gate”. The “/gate” option should be followed by the Gateway connect string. The Gateway connect string consists of multiple parts: the name of the Gateway (as specified in the appropriate “Open” statements in the Transall program), optionally, the address of the Gateway, if it is not the same as the name of the Gateway, and the port number to which the Gateway server is listening. The address value can contain the TCP/IP address value of the Server on which the Transall Gateway is running, or the Server’s Domain Name System (DNS) name. The connect string is specified as either:

`Name:port`

or

`Name/address:port`

Using the format which uses a separate name and address allows you to change the location of the server that you are reading or writing from without having to change the Transall programs themselves. This can be handy in the event that you want to carry out the same task at various remote servers, or simply in the case that a machine has been retired and a new machine with a new address is replacing it. When your program is run, and attempts to open a file via a Gateway, Tranexe will see if that Gateway was identified on the command line. If so, the program will use

the Gateway name and address specified to attempt the file open on the remote server. If that name is not found, or no "/gate" option was given, Tranexe will attempt to open the file locally. This option allows the developer to write scripts that will run unchanged on multiple platforms, accessing files that reside on only one platform.

For instance, a script could be written which reads data from a VSAM file on the mainframe and uses the data to write a report to a PC file. By using gateway-style file opens for both the input and output files, the same file would run on the PC—accessing the VSAM data through a gateway— and on the mainframe—writing the report to a file on the PC.

The following example illustrates running a script which opened its input and outputfiles through gateways:

●
●●

```
' Open the VSAM file
Open TrnSys$Source1.FileName At "MVSGATE" For VSAM
Input AMParms:
keylen=20,keyoff=0 As TrnSys$Source1.Handle
```

●●●

```
'Open the PC file
Open TrnSys$Destination1.FileName At "PCGATE" For
Binary Output Access
Write Lock Write As TrnSys$Destination1.Handle
```

●●●

The following command:

```
tranexe /gate MVSGATE:2301 sample1.tex Run
```

will run the sample1.tex program, attempting to open the VSAM file via the Gateway server on the "MVSGATE" machine via port 2301. Since the "PCGATE" gateway address wasn't identified on the Tranexe command line, the output file would be written onto the local machine.

Running the same program on z/OS with the following JCL:

```
//JS010 EXEC PGM=TRANEXE,COND=(0,NE),
// PARM='SAMPLE1 /gate PCGATE:2001 Run'
```

opens the VSAM file locally (since no address for the "MVSGATE" was specified) and would attempt to create the output file via the Gateway server running on the "PCGATE" machine via port 2001.

RUNNING THE GATEWAY SERVER

Prior to starting the Gateway Server application, you should verify that TCP/IP is installed and running on your server machine. In addition you should set aside a TCP/IP service port to be used by the Gateway server. You may need the administrator of the machine to provide a port number to you. Once you have verified that TCP/IP is operational and you have a service port to use, you are ready

to run the Gateway server. The Gateway server is a started from the command line and takes only one command line parameter, the service port number. If running the server on a Windows or UNIX platform, you would simply type:

```
trgateip port
```

where “port” is the number of the TCP/IP port that the server should listen to for connections from remote Gateway clients. If running the server on z/OS, your JCL might look like:

```
//JS010 EXEC PGM=TRGATEIP,COND=(0,NE),
// PARM='2345'
```

Statement Syntax

<numeric value> = Abs (<numeric expression >)

Returns the absolute value of a number.

<numeric ASCII value> = Asc (<string expression >)

Returns the ASCII code value of the first character in the string expression.

<numeric value> = Atn (<numeric expression >)

Returns the arctangent of a number.

Beep

When executed, this function causes a beep.

CAny(<Any>)

Attempts to convert data to any data type.

Call <method sub/function name> ([<expression>,<expression>...])

Call a method ignoring return value.

<datetime> = CDate (<string expression>)

Convert string expressions to datetime data types.

<double> = CDbl (<string expression>)

Convert string expressions to double data types.

ChDir <string path>

Changes the Current directory but not the default drive.

ChDrive <string drive>

Changes the current drive.

Choose(<index>, <choice1>[, ... <choice n>])

Returns item #<index> from the list passed in. For example, if the value of index was 2, the second item from the list would be returned. Note that the index must be a numeric value that can be assigned to an integer. If the index passed in is less than one, or exceeds the number of choices passed in, a value of Null is returned. Choose() evaluates *every* choice in the list, even though only one is returned, so you should be careful to avoid unwanted side effects.

<string> = Chr$ (<numeric expression ASCII value>)

Returns a one character string for the ASCII value.

ClassClassRefClass {GUID} (See **LibGUID**)

(Global Only) Define the Global Unique ID (GUID) for the Transit Class ActiveX Object.

<long> = CLng (<string expression>)

Convert string expressions to long data types.

<Long Long> = CLngLng(<string expression>)

Converts values to Long Long.

<integer> = CInt (<string expression>)

Convert string expressions to integer data types.

Close [#]<file number>

Close file.

CloseDocumentSet(<String FileName>)

Closes a DDF file and flushes all buffered file I/O for a DDF to the file on disk. If the DDF is written to again after issuing a CloseDocumentSet, it will truncate and overwrite the DDF file contents. This function should be used when calling Transall as a server, such as when Transall is creating DDF files under Docupresentment transactions or via calls to Transall running as an ActiveX server.

<COBOL Numeric> = CNum(<String>)

Convert numeric string to the COBOL Numeric types of UNum (unpacked numeric) or PNum (packed numeric).

<numeric value> = Cos (<numeric expression >)

Returns the cosine of an angle.

Set *objectvariable* = CreateObject("*progID*", *["servername"]*)

The *progID* argument should be a fully qualified class name to the object being created; for example, "MyProject.TransObject". The optional *servername* argument can be used to create an object on a remote machine across a network. This argument takes the Machine Name portion of a share name. For example, with a network share named "\\MyServer\Public", the *servername* argument would be "MyServer."

<string path for current drive> = CurDir (<string drive>)

Returns current path.

CvtCurrencyToEuro(*amount, exchange_rate)*

In compliance with EC Regulation 1103/97, this function takes an amount of a national currency and an exchange rate with the Euro (based on 1 Euro) and converts the national currency amount to Euros. The original amount, the exchange rate, and the amount returned are all UNum data types. The amount returned is rounded to the nearest cent (.105 rounds to .11).

CvtCurrencyFromEuro(*amount, exchange_rate, digits_on_right_of_decimal*)

In compliance with EC Regulation 1103/97, this function takes an amount in Euros, an exchange rate with the Euro (based on 1 Euro), and the number of digits to the right of the decimal to retain in the result, and converts the amount from Euros to the new currency. The original amount, the exchange rate, and the amount returned are all UNum data types.

CvtCurrencyToCurrency(*amount, exchange_rate_for_amount, rate_for_new_currency, digits_on_right_of_decimal*)

In compliance with EC Regulation 1103/97, this function converts one currency into another, based on the exchange rate of each currency with one Euro. The function returns the converted amount rounded to the number of decimal places specified by the caller. The original amount, the exchange rates and the amount returned are all UNum data types. For example, to convert $25.00 to Swiss Francs, accurate to 4 decimal places, when the exchange rate for Dollars was $1.1901 for 1 Euro and the exchange rate for Swiss Francs was 1.5864 Swiss Francs for 1 Euro, you would enter

```
Dim SwissFrancs As UNum( 10, 4, "S")

SwissFrancs = CvtCurrencyToCurrency( 25.00, 1.1901, 1.5864, 4)
WriteConStdOut( "$25.00 converts to " & SwissFrancs & " Swiss Francs" &
_CRLF_)
```

DataManConnect(<hFile>, <String DNS or IP name> ,<Long Type>, <Long Lrl>, <String Value Delimiter>, <String Record Delimiter>)

Connects a file handle to the Transall Threaded Data Manager (TDM). On successful connection all file IO to this handle is redirected to the TDM.

DataManConnectDDF(<String DDF Name>, <String DNS or IP name>)

Connects a DDF file name to the Transall Threaded Data Manager (TDM). On successful connection all file IO to this DDF is redirected to the TDM.

<datetime> = Date

Returns the current system date.

<datetime > = DateAdd (<string interval mask expression>, <double number of intervals expression>, <datetime expression>)

Returns a datetime computed from the passed datetime value with the passed time interval added.

String interval mask	Value
yyyy	Year.
q	Quarter.
m	Month.
y	Day of year. *
d	Day. *
w	Weekday. *
ww	Week.
h	Hour.
n	Minute.
s	Second.
* = All same interval.	

<double> = DateDiff (<string interval mask>, <datetime>, < datetime >)

Returns a double containing the number of time intervals between the two passed date times.

String interval mask	Value
yyyy	Year.
q	Quarter.
m	Month.
y	Day of year. *
d	Day. *
w	Weekday. *
ww	Week.
h	Hour.
n	Minute.
s	Second.
= All same interval	

variable = DateSerial(year, month, day)

Builds a date value, where:

year is an integer value for the year
month is an integer value for the month (1-12)
day is an integer value for the day

Note that expressions returning integers may be used for the values, and that values outside the accepted range move forward or backward as appropriate:

DateSerial(1990, 7, 31 + 1) returns the date for August 1,1990
DateSerial(2001, 1, -1) returns the date for December 31, 2000

<integer> = Day (<datetime>)

Returns the day.

<access specifier> Declare Function <name> Lib <filename.dll> [Alias "<name>"] ([ByVal] <Parm> As <Data Type>,...) As <Data Type>

(Global Only) Define external functions in DLLs

Parameter	Value
Access Specifier	The valid choices are as follows: • **Public**—Items are accessible from any method. • **Protected**—Items are accessible from member methods and inheriting classes. • **Private**—Items are accessible from member methods.
Alias	External name of function in DLL. By default, the external and internal name of the DLL function is identical.
ByVal	Indicates the value should be passed by value on the stack rather than by pointer.
Data Type	The valid choices are as follows: • **Integer**—16-bit signed integer. • **Long**—32-bit signed integer. • **Long Long**—64-bit signed integer. • **String**—BSTR up to 2 GB in length. • **DateTime**—Date/Time structure in the form of CCYY/MM/DD.HH.MI.SS.MMMMMMMMM. • **Float**—32-bit (4-byte) floating-point number. • **Double**—64-bit (8-byte) floating-point number. • **Any**—Disables datatype checking.

<access specifier> Declare Sub <name> Lib <filename.dll> [Alias "<name>"] ([ByVal] <Parm> As <Data Type>,...)

(Global Only) Define external subroutine in DLLs

Parameter	Value
Access Specifier	The valid choices are as follows: • **Public**—Items are accessible from any method. • **Protected**—Items are accessible from member methods and inheriting classes. • **Private**—Items are accessible from member methods.
Alias	External name of function in DLL. By default, the external and internal name of the DLL function is identical.
ByVal	Indicates the value should be passed by value on the stack rather than by pointer.
Data Type	The valid choices are as follows: • **Integer**—16-bit signed integer. • **Long**—32-bit signed integer. • **Long Long**—64-bit signed integer. • **String**—BSTR up to 2 GB in length. • **DateTime**—Date/Time structure in the form of CCYY/MM/DD.HH.MI.SS.MMMMMMMMM. • **Float**—32-bit (4-byte) floating-point number. • **Double**—64-bit (8-byte) floating-point number. • **Any**—Disables datatype checking.

<Any> = Decode(<Any ExamineVal>, <Any TestVal1>, <Any ReturnVal1>[, ... <Any Test n>,<Any ReturnValn>][,<Any DefaultVal]))

Evaluates each test value and returns the corresponding ReturnVal if the test value matches the value to examine. If there is no match, the default value is returned if one is provided, otherwise a null value is returned.

Define <access specifier> Set <parent table name> <child table name> <connection option> [Cascading] <sort option>

(Global Only) Define a set between tables in the Transit object's database.

Parameter	Value
Access Specifier	The valid choices are as follows: • **Public**—Items are accessible from any method. • **Protected**—Items are accessible from member methods and inheriting classes. • **Private**—Items are accessible from member methods.
Connect	The valid choices are as follows: • **Automatic**—Set is automatically connected as rows are inserted into member tables. Set can be disconnected. • **Mandatory**—Set is automatically connected as rows are inserted into member tables. Set can NOT be disconnected. • **Optional**—Set is not automatically connected as rows are inserted into member tables. Set can be disconnected. • **Cascading**—All child table rows connected to the parent are deleted when the parent is deleted.
Sort	The valid choices are as follows: • **Next**—New rows are insert after the current row. • **First**—New rows are inserted at the top of the set. • **Last**—New rows are inserted at the end of the set.

Define <access specifier> Table <table name> (<access specifier> <element name>
As <Data Type> [, <access specifier> <element name> As <Data Type>…])

(Global Only) Define tables in the Transit object's database.

Parameter	Value
Access Specifier	The valid choices are as follows: • **Public**—Items are accessible from any method. • **Protected**—Items are accessible from member methods and inheriting classes. • **Private**—Items are accessible from member methods.
Data Type	The valid choices are as follows: • **Integer**—16-bit signed integer. • **Long**—32-bit signed integer. • **Long Long**—64-bit signed integer. • **String**—BSTR up to 2 GB in length. • **DateTime**—Date/Time structure in the form of CCYY/MM/DD.HH.MI.SS.MMMMMMMMM. • **Float**—32-bit (4-byte) floating-point number. • **Double**—64-bit (8-byte) floating-point number. • **PNum(dig left[, dig right[, "S" or "U"]])**—Packed zoned-decimal field. • **UNum(dig left[, dig right[, "S" or "U"]])**—Zoned-decimal field. • **<User-defined type>**—See Type statement.

DefineEvent <event name>(<Parm> As <Data Type>[, ...])

Defines an event and the number of parameters that it should expect.

DeleteAllRows <Table or Set>

Deletes all rows from a table or a set in the Transall internal database.

DeleteRow <Table Name>

Delete the current row in a table in the Transit object's database.

DeleteSetting base_key, section [, key]

Removes a value or group of values from the registry. If the "key" value is supplied, only that specific value will be deleted. If not, the entire section will be deleted. As with GetSetting and SaveSetting, the "base_key" may either be an existing base value, such as:

HKEY_CLASSES_ROOT
HKEY_CURRENT_CONFIG
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS,HKEY_PERFORMANCE_DATA
HKEY_DYN_DATA

or simply a user-defined base. If one of the "HKEY" values is not used, one will be built by appending the user-defined base name to the following value:

HKEY_CURRENT_USER\Software\Transall Program Settings\....

<Integer> DF$GetFirstRow(<Table Name>)

Sets the current row of a table to the first row under the table's current parent in the Docuflex schema.

<Integer> DF$GetLastRow(<Table Name>)

Sets the current row of a table to the last row under the table's current parent in the Docuflex schema.

<Integer> DF$GetNextRow(<Table Name>)

Sets the current row of a table to the next row under the table's current parent in the Docuflex schema.

<String> = DF$GetPathmapValue(<Pathmap string expression>)

This function returns the string value of a pathmap from the Docuflex environment.

<Integer> DF$GetPriorRow(<Table Name>)

Sets the current row of a table to the prior row under the table's current parent in the Docuflex schema.

<Long> DF$GetRowCount(<Table Name>)

Gets the count of table rows under the table's current parent in the Docuflex schema.

<Long> DF$GetRowNumber(<Table Name>)

Gets the row number of the table's current row under the table's current parent in the Docuflex schema.

<String> DF$GetTagValue(<Tag Name String>)

Retrieves a value from the Docuflex tag pool.

<Bool> = DF$GetVirtualFlag()

Returns True if running under ISITAVM host for Docuflex and Docuflex is running in extended virtual memory mode.

<Long Count> = DF$GetVirtualPages()

When running under ISITAVM host for Docuflex and Docuflex is running in extended virtual memory mode returns the number of composed pages that are retained in memory.

<Integer> DF$IsRowFirst(<Table Name>)

Tests if the table's row currency is the first row under the table's current parent in the Docuflex schema.

<Integer> DF$IsRowLast(<Table Name>)

Tests if the table's row currency is the last row under the table's current parent in the Docuflex schema.

<Integer> DF$IsRowMiddle(<Table Name>)

Tests if the table's row currency is NOT the first or the last row under the table's current parent in the Docuflex schema.

<Integer> DF$IsTableEmpty(<Table Name>)

Tests if the table has any rows under the table's current parent in the Docuflex schema.

<Long> DF$SetCurrentRow(<Table Name>, <Row Number Long>)

Sets the current row number of a table under the table's current parent in the Docuflex schema.

<String> DF$SetTagValue(<Tag Name String>, <Tag Value String>)

Sets a value in the Docuflex tag pool.

<Bool> = DF$SetVirtualFlag(<Bool VirtFlag>)

Sets indicator to Docuflex that it should be running in either True, extended virtual memory mode, or False, not in extended virtual memory mode, and returns value that this flag was set to before the call to this function.

<Long Count> = DF$SetVirtualPages(<Long VirtPages>) As Long

When running under ISITAVM host for Docuflex and Docuflex is running in extended virtual memory mode sets the number of composed pages that are retained in memory, and returns setting that existed prior to this call.

<Bool> DF$SortTable(<Table.Col Name>[, <Descending Bool>])

Sorts the rows of a table under the table's current parent in the Docuflex schema.

DF$Walk <Table> [<Long start at expression>]

Repeats a block of statements once for each row in a Docuflex schema database table.

Dim <varname> As <Data type>

Defining local variables:

Parameter	Value
Data Type	The valid choices are as follows: • **Integer**—16-bit signed integer. • **Long**—32-bit signed integer. • **Long Long**—64-bit signed integer. • **String**—BSTR up to 2 GB in length. • **String * <string length>**—BSTR up to 2 GB in length. • **DateTime**—Date/Time structure in the form of CCYY/MM/DD.HH.MI.SS.MMMMMMMMM. • **Float**—32-bit (4-byte) floating-point number. • **Double**—64-bit (8-byte) floating-point number. • **PNum(dig left[, dig right[, "S" or "U"]])**—Packed zoned-decimal field. • **UNum(dig left[, dig right[, "S" or "U"]])**—Zoned-decimal field. • **<User-defined type>**—See Type statement. • **Any**—Variant datatype.

Global variables are defined outside the scope of any function (traditionally before the first function or subroutine is defined) and are visible to every function or subroutine within the program:

```
Dim visible As Integer
         *
         *
         *
  Public Sub Run()
     visible = 1
     Call showVis()
  End Sub

  Public Sub showVis()
     WriteConStdOut( "Visible = " & visible & _CRLF_)
  End Sub
```

<string name of a file or directory> = Dir$ [(<filespecString>[,attrmask])]

Returns name of a file or directory that matches a specified pattern and file attribute. Also can return a volume label.

<DmgObject Level> = DmgCabOpenLevel(<DmgObject Cabinet>, <Long Level>)

Open the particular level of Cabinet.

<Long ErrCode> = DmgDocAddRendition(<DmgObject Doc>, <DmgObject Doc2>)

Add Doc and Doc2 to the same rendition set. Doc should not be checked out.

<Long ErrCode> = DmgDocCheckIn(<DmgObject Doc>, <String VersionComment>, <Long CheckInType>)

Check in Doc's changes. CheckInType should be one of DmgDocCheckIn constants.

<Long> = DmgDocCheckInMajor

Constant for DmgDocCheckIn. Check in as major document version.

<Long> = DmgDocCheckInMinor

Constant for DmgDocCheckIn. Check in as minor document version.

<Long> = DmgDocCheckInSame

Constant for DmgDocCheckIn. Check in as same document version.

<DmgObject CheckedOutDoc> = DmgDocCheckOut(<DmgObject Doc>, <DmgObject DestFolder>, <DateTime DueDate>, <String CheckOutReason>)

Check out Doc for modification.

<Long ErrCode> = DmgDocDelRendition(<DmgObject Doc>)

Remove Doc from its current rendition set. Doc should not be checked out.

<Long ErrCode> = DmgDocGetAppProps(<DmgObject Doc>, <String SubCategory Return>, <String Status Return>, <DateTime Date Return>, <String RenditionKey Return>)

Get Doc's application properties.

<Long ErrCode> = DmgDocGetCheckOutProps(<DmgObject Doc>, <Long IsCheckedOutHandle Return>, <String AltCab Return>, <Long AltID Return>, <String CheckedOutBy Return>, <String CheckedOutDesc Return>, <DateTime DueDate Return>)

Get Doc's check out properties.

<Long ErrCode> = DmgDocGetCMProps(<DmgObject Doc>, <Long Approved Return>, <Long Released Return>, <Long Obsolete Return>)

Get Doc's content management state.

<Long ErrCode> = DmgDocGetFile(<DmgObject Doc>, <String DiskFileName>)

Download Doc to local file DiskFileName

<Long ErrCode> = DmgDocGetGenProps(<DmgObject Doc>, <String Label Return>, <String Type Return>, <String Category Return>, <String Description Return>, <String Author Return>)

Get Doc's general properties.

<Long ErrCode> = DmgDocGetSpecifier(<DmgObject Doc>, <Long DocID Return>, <Long MajorVer Return>, <Long MinorVer Return>)

Get Doc's properties that form its document specifier.

<Long ErrCode> = DmgDocGetUserProps(<DmgObject Doc>, <String Keyword1 Return>, <String Keyword2 Return>, <String DocFlag1 Return>, <String DocFlag2 Return>)

Get user-definable properties.

<Long ErrCode> = DmgDocInitialCheckIn(<DmgObject Doc>, <String FileName>)

Check in new Doc. FileName's content's will be uploaded and saved.

<DmgObject Cabinet> = DmgDocOpenRendition(<DmgObject Session>, <DmgObject Doc>, <String RenditionKey>, <DmgObject RenditionDoc Return>)

Open RenditionDoc who has RenditonKey and is in Doc's rendition set.

<Long ErrCode> = DmgDocSetAppProps(<DmgObject Doc>, <String Subcategory>, <String Status>, <DateTime Date>, <String RenditionKey>)

Set application properties. Doc should be checked out.

<Long ErrCode> = DmgDocSetCMProps(<DmgObject Doc>, <Long Approved>, <Long Released>, <Long Obsolete>)

Set content management state. Set state to True, False, or -1 for don't-change. Doc should not be checked out.

<Long ErrCode> = DmgDocSetFile(<DmgObject Doc>, <String DiskFileName>)

Upload DiskFileName's contents to Doc. Doc should be checked out.

<Long ErrCode> = DmgDocSetGenProps(<DmgObject Doc>, <String Label>, <String Type>, <String Category>, <String Description>, <String Author>)

Set Doc's general properties. Doc should be checked out.

<Long ErrCode> = DmgDocSetUserProps(<DmgObject Doc>, <String Keyword1>, <String Keyword2>, <String DocFlag1>, <String DocFlag2>)

Set user-definable properties. Doc should be checked out.

<Long ErrCode> = DmgDocUndoCheckOut(<DmgObject Doc>)

Check in Doc without saving changes.

<DmgObject Doc> = DmgFoldCreateDoc(<DmgObject Folder>)

Create new doc in folder. Doc is considered checked out. Use DmgDocInitialCheckIn to save document.

<Long ErrCode> = DmgFoldSave(<DmgObject Folder>)

Save changes to a new folder.

<Long ErrCode> = DmgGetError(<String Message Return>)

Get the last error code and error message. Error code is 0 when there was no error.

<DmgObject Folder> = DmgItemCreateFolder(<DmgObject Item>)

Create new folder as a child of Item. You will need to set properties and use DmgFoldSave.

<DmgObject Project> = DmgItemCreateProject(<DmgObject Item>)

Create new project as a child of Item. You will need to set properties and use DmgFoldSave.

<Long ErrCode> = DmgItemDelete(<DmgObject Item>)

Permanently delete Item. If a doc, all of its versions are deleted. If a folder, it should have no children.

<DmgObject ChildItem> = DmgItemGetFirstChild(<DmgObject Item>, <Long ItemType Return>, <String Label Return>)

Get the first child of Item along with its label and type. ItemType will be one of DmgObjType constants.

<DmgObject NextItem> = DmgItemGetNextSibling(<DmgObject Item>, <Long ItemType Return>, <String Label Return>)

Get the next sibling of Item along with its lable and type. ItemType will be one of DmgObjType constants.

<Long> = DmgListCategoryStatus

Constant for DmgObjOpenList. Statuses available for category.

<Long> = DmgListCategorySubcategory

Constant for DmgObjOpenList. Subcategories available for category.

<Long> = DmgListCategoryUserFlag1

Constant for DmgObjOpenList. Possible values for UserFlag1 for category.

<Long> = DmgListCategoryUserFlag2

Constant for DmgObjOpenList. Possible values for UserFlag2 for category.

<Long> = DmgListDocRendition

Constant for DmgObjOpenList. Rendition key for each document in the same rendition set as the document.

<Long> = DmgListDocVersion

Constant for DmgObjOpenList. Major.Minor pairs for each version of the document.

<Long Count> = DmgListGetCount(<DmgObject List>)

Get the number of items in the list.

<String Item> = DmgListGetItem(<DmgObject List>, <Long Index>)

Get the Index'th item from List. Index is zero-based.

<Long> = DmgListHandleExtProp

Constant for DmgObjOpenList. Extended properties for object.

<Long> = DmgListHandleProp

Constant for DmgObjOpenList. Normal properties for object.

<Long> = DmgListProjectBranch

Constant for DmgObjOpenList. Project's possible branch names.

<Long> = DmgListProjectTeamMember

Constant for DmgObjOpenList. Members in the team that can advance project.

<Long> = DmgListSessionCabinet

Constant for DmgObjOpenList. Cabinets available in session.

<Long> = DmgListSessionCategory

Constant for DmgObjOpenList. Categories available in session.

<Long> = DmgListSessionWorkflow

Constant for DmgObjOpenList. Workflows available in session.

<Long ErrCode> = DmgObjClose(<DmgObject Object Return>)

Frees resources associated with an Object. Every 'Open' call needs a close.

<Long Handle> = DmgObjGetHandle(<DmgObject Object>)

Get the dmg_api handle for Object. Used when Transall's API is insufficient.

Use of this function should be rare. It is intended to provide the ability to create temporary workarounds. General use of this function should be considered unsupported.

<DmgObject Prop> = DmgObjGetProp(<DmgObject Object>, <Long PropType>, <String PropName>)

Get named property. PropType should be one of DmgPropType constants.

<DmgObject List> = DmgObjOpenList(<DmgObject Obj>, <Long Type>)

Open a list. See DmgList constants.

<Long> = DmgObjTypeDocument

Constant for DmgItemGetFirstChild & DmgItemGetNextSibling. Item is a document.

<Long> = DmgObjTypeFolder

Constant for DmgItemGetFirstChild & DmgItemGetNextSibling. Item is a folder.

<Long> = DmgObjTypeProject

Constant for DmgItemGetFirstChild & DmgItemGetNextSibling. Item is a project.

<Long> = DmgProjAll

Constant for DmgSessOpenProjects. All pending and suspended projects.

<Long ErrCode> = DmgProjCheckIn(<DmgObject Project>[, <String BranchName>])

Check in Project. If BranchName is specified, it will advance Project.

If there is only a single branch and BranchName is specified, then the actual value of BranchName is ignored and the project is advanced down the only path.

<Long ErrCode> = DmgProjCheckOut(<DmgObject Project>)

Check out a project for modification and advancement.

<Long ErrCode> = DmgProjForward(<DmgObject Project>, <String Username>)

Forward check out ownership of project to team member. Project should be checked out.

<Long> = DmgProjPending

Constant for DmgSessOpenProjects. Pending projects.

<Long> = DmgProjPending

Constant for DmgSessOpenProjects. Suspended projects.

<Long> = DmgProjPending

Constant for DmgSessOpenProjects. Suspended projects.

<Long ErrCode> = DmgProjResume(<DmgObject Project>)

Resume suspended project.

<Long ErrCode> = DmgProjSuspend(<DmgObject Project>, <String Comment>, <DateTime TillDate>)

Suspend project until resume or TillDate.

<String Value> = DmgPropGetQualifiedName(<DmgObject Prop>)

Get the qualified name of a property. Useful for DmgSessPerformQuery.

<String Value> = DmgPropGetValue(<DmgObject Prop>)

Get the value of property.

<Long> = DmgPropTypeBasic

Constant for DmgObjGetProp. Basic property.

<Long> = DmgPropTypeExt

Constant for DmgObjGetProp. Extended property.

<Long ErrCode> = DmgPropSetValue(<DmgObject Prop>, <String Value>)

Set the value of property.

<TaskID Long> = DmgSessGetTaskID(<DmgObject Session>, <String Workflow>, <String TaskDesc>)

Lookup the task id for task. Returns 0 if not found or no projects are in task.

<DmgObject Session> = DmgSessOpen(<String Profile>)

Open a session using Profile. This connects to a Documanage server.

<DmgObject Cabinet> = DmgSessOpenCabinet(<DmgObject Session>, <String CabinetName>)

Open cabinet. DmgItem functions are not useful on returned cabinet.

<DmgObject Category> = DmgSessOpenCategory(<DmgObject Session>, <String CategoryName>)

Open category.

<DmgObject Cabinet> = DmgSessOpenDoc(<DmgObject Session>, <String CabinetName>, <Long DocID>, <Long MajorVer>, <Long MinorVer>, <DmgObject Doc Return>)

Open the document with document specifier. This is the fasted way to open a document from a previous session.

Use -1 for the major and/or minor version numbers for the latest version of that type.

<DmgObject Cabinet> = DmgSessOpenProjects(<DmgObject Session>, <String Workflow>, <Long TaskID>, <Long State>)

Open the projects in task. See DmgProj constants for State.

Use DmgClose on Cabinet

<DmgObject Cabinet> = DmgSessPerformQuery(<DmgObject Session>, <String CabinetName>, <String FolderFilter>, <String DocFilter>, <String DocCategory>, <String DocPropFilter>)

Perform a query. Use DmgItemGetFirstChild and DmgItemGetNextChild to iterate results.

Use DmgClose on Cabinet.

Do

<statements...>
Loop [While(<expr>)]

Allows you to code a looping construct which (optionally) tests for the looping condition each time, at the end of the sequence of code in the loop.

Do Walk <Table name> or <Set name>

<statements...>
Loop

Loop processing once for each table row in a table or set in the Transit object's database.

Do While {<condition>}

<statements...>
Loop

Loop processing while condition is "True".

<Long resultcode> = DSIAddAttachVar(<long queue flag>, <String Variable Name>, <String Return>)

Sets a value in the Oracle Internet Document Server (IDS) environment. Note, the queue flag should be set to either TrnSys$DSI_INPUTQUEUE or TrnSys$DSI_OUTPUTQUEUE)

<Long resultcode> = DSILocateAttachVar(<long queue flag>, <String Variable Name>, <String Return>)

Retrieves a value from the Oracle Internet Document Server (IDS) environment. Note, the queue flag should be set to either TrnSys$DSI_INPUTQUEUE or TrnSys$DSI_OUTPUTQUEUE)

<any> = DynmGetTableValue(<string table name>, <string column name>)

Returns a value from a Transall database table dynamically.

End Function

Ends a Transall Function script.

End SendEvent

Indicates that an event should be handled by whatever routine was handling it prior to when the corresponding "SendEvent" statement was issued. This allows you to handle specific events differently in different situations, or only during the course of certain routines.

Enum <name>

<enum_name> = <value>
  *
  *
  *

End Enum

Allows you to assign names to integer values and assign a datatype name to the collection of values.

```
Enum Dwarf
    Sneezy  = 1
    Dopey   = 2
    Grumpy  = 3
    Sleepy  = 4
    Bashful = 5
    Doc     = 6
    Happy   = 7
End Enum
       *
       *
       *

Public Sub example()
    Dim  Actor1 As Dwarf

    Actor1 = Bashful
```

<any> = Environ(<string environment variable>)

Returns the value of the passed environment variable.

<string> = Environ$(<string environment variable>)

Returns the value of the passed environment variable.

<long> = Err

Error code from last instruction (usually 0 or 1001).

<string> = Err.Description

Error message from last instruction.

Error [<Message String>,] [<Number>]

Examples:

`Error 1234`—terminates process with a 1234 reported error message, returns 1001 as error result (see "TerminateApp" to kill processing with a return value).

`Error "Stopped processing because database is down."`—terminates process with a string reported error message, returns 1001 as error result.

`Error "Stopped processing because database is down.", 1234`—terminates process with a string reported error message, returns 1234 as error result from processing.

<boolean integer> = Even(<Any>)

Returns non-zero if the expression equates to an Even number.

Exit Do

Unconditional break out of loop processing.

Exit Function

Unconditional method exit.

Exit Sub

Unconditional method exit.

<numeric value> = Exp (<numeric expression >)

Returns e (the base of natural logarithms) raised to a power.

<boolean integer> = FileBitsAreAscii (<lFileHandle as Long>)

Causes Transall to read and write a file using a binary coding scheme of ASCII.

<boolean integer> = FileBitsAreDefault (<lFileHandle as Long>)

Causes Transall to read and write a file using the default binary coding scheme of the operating environment in which Transall is executing.

<boolean integer> = FileBitsAreEbcdic (<lFileHandle as Long>)

Causes Transall to read and write a file using a binary coding scheme of EBCDIC.

FileConcat <string expression source>, <string expression destination>

Concatenates the contents of one file to another.

FileCopy <string expression source>, <string expression target>

Copies the contents of a file.

<boolean integer> = FileValuesAreBigEndian (<lFileHandle as Long>)

Causes Transall to read and write binary values to a file ordering the bytes of binary value fields as Big Endian (big end in).

<boolean integer> = FileValuesAreDefaultEndian (<lFileHandle as Long>)

Causes Transall to read and write binary values to a file ordering the bytes of binary value fields in the default order of the operating environment in which Transall is executing.

<boolean integer> = FileValuesAreLittleEndian (<lFileHandle as Long>)

Causes Transall to read and write binary values to a file ordering the bytes of binary value fields as Little Endian (little end in).

<boolean integer> = FindRowSet (<set>, <table.col>,<value>[,<boolean start top>)

Returns non-zero value found and table currency changed.

<boolean integer> = FindRowTbl (<table.col>,<value>[,<boolean start top>)

Returns non-zero value found and table currency changed.

<numeric value> = Fix (<numeric expression >)

Returns the integer portion of a number. If the number is negative, Int returns the first negative integer greater than or equal to number.

For <counter> = <start> To <end> [Step <increment>]

*
*
*

Next [counter]

Executes all code between the "For" statement and the "Next" statement and then increments the "<counter>" variable by "<increment>". Control then continues back to the top of the loop and this continues until the value of the counter variable exceeds "<end>". If no step value is supplied, a value of 1 is assumed. The increment value may be positive or negative. If a negative value is used, the value of "<start>" should be larger than "<end>".

```
-------------------------------------------------
For Counter = 1 To 5
       WriteConStdOut( "  " & Counter & _CRLF_)
    Next Counter
-------------------------------------------------
```

produces:

1

2

3

4

5

while

```
-------------------------------------------------
For Counter = 5 To 1 Step -1
       WriteConStdOut( "  " & Counter & _CRLF_)
    Next Counter
-------------------------------------------------
```

produces:

5

4

3

2

1

In addition, For…Next loops may be nested, as long as all inner loops are completed before closing the outer loops.

```
-------------------------------------------------
Dim Row As Integer
    Dim Col As Integer

    For Row = 1 To 5
      WriteConStdOut( "  " & Row & ".)  ")
      For Col = 0 To 4
         WriteConStdOut( Chr$( Col + Asc("A")) & "  ")
      Next Col
      WriteConStdOut( _CRLF_)
    Next Row
-------------------------------------------------
```

produces:

1.) A B C D E

2.) A B C D E

3.) A B C D E

4.) A B C D E

5.) A B C D E

<string> = Format$ (<string expression> [,<string expression format mask>])

Returns a string formatted according to symbols contained in a format mask.

<Long> Fp$PageDotCount <DOT Position>

Returns the current DOT position in an FpPlus data destination's logic tree.

<Long> Fp$PageNumber <Page Number>

Returns the current page number in an FpPlus data destination's logic tree.

<Long Fp$SectionPageNumber <Page Number>

Returns the current section page number in an FpPlus data destination's logic tree.

<Long> Fp$TotalPages <Page Count>

Returns the total number of pages in an FpPlus data destination's logic tree.

<Long> Fp$TotalSectionPages <Page Count>

Returns a section's total number of pages in an FpPlus data destination's logic tree.

<Long> FpPlusGetRfrmtHandle(<FpPlus Handle>)

Retrieves the internal handle to the VRF reformatter so Vdr* functions can be used against an FpPlus data destination.

<next available file number> = FreeFile ()

Returns the next file number available for use with the Open statement.

<Access Specifier> Function <name> ([[Optional] [ByVal|ByRef] <Parm> As <Data Type>,... [= <default>]]) As <Data Type>

<statements...>
End Function

Parameter	Value
Access Specifier	The valid choices are as follows: • **Public**—Items are accessible from any method. • **Protected**—Items are accessible from member methods and inheriting classes. • **Private**—Items are accessible from member methods.
Optional	Preface the parameter definition with the keyword "Optional" and supply a default value to the parameter(s).
ByVal	Indicates the value should be passed by a copy value on the stack rather than by a pointer back to the original value.
ByRef *default *	Indicates the value should be passed by a reference pointer back to the original value rather than by a copy value on the stack.
Data Type	The valid choices are as follows: • **Integer**—16-bit signed integer. • **Long**—32-bit signed integer. • **Long Long**—64-bit signed integer. • **String**—BSTR up to 2 GB in length. • **DateTime**—Date/Time structure in the form of CCYY/MM/DD.HH.MI.SS.MMMMMMMMM. • **Float**—32-bit (4-byte) floating-point number. • **Double**—64-bit (8-byte) floating-point number. • **PNum(dig left[, dig right[, "S" or "U"])**—Packed zoned-decimal field. • **UNum(dig left[, dig right[, "S" or "U"])**—Zoned-decimal field. • **Any**—Variant datatype.
= <default>	Preface the parameter definition with the keyword "Optional" and supply a default value to the parameter(s).

Using optional default values:

```
Public Function Sum3(Optional ByVal First As Long = 0, _
                     Optional ByVal Second As Long = 0, _
                     Optional ByVal Third As Long = 0) As Long
    Sum3 = First + Second + Third
End Function
        •
        •
        •
Dim lExeRes          As Long

lExeRes = Sum3()
  WriteConStdOut( "The sum of nothing is " & lExeRes & _CRLF_)
lExeRes = Sum3(1)
  WriteConStdOut( "The sum of 1 is " & lExeRes & _CRLF_)
lExeRes = Sum3(1,2)
  WriteConStdOut( "The sum of 1 and 2 is " & lExeRes & _CRLF_)
lExeRes = Sum3(1,2,3)
  WriteConStdOut( "The sum of 1 and 2 and 3 is " & lExeRes & _CRLF_)
```

When run, this would produce:

The sum of nothing is 0
The sum of 1 is 1
The sum of 1 and 2 is 3
The sum of 1 and 2 and 3 is 6

Note that once an optional parameter is defined, all subsequent parameters must be defined with default values as well.

<double sum value> = GetColSumSet(<Column Name>, <Set Name>)

Computes the sum of the values for a field from all rows in a Transall set.

<double sum value> = GetColSumTbl(<Column Name>)

Computes the sum of the values for a field from all rows in a Transall table.

<String> = GetExtEnvString(<String Variable Name>)

Retrieves a value from the external environment. Works consistently under all Transall execution modes.

Set *objectvariable* = GetObject([*pathname*] [, *progID*])

The *pathname* argument can be the path to an existing file, an empty string, or Null. If it is Null, then *progID* is required. Specifying the path to an existing file causes GetObject to create an object using the information stored in the file. Using an empty string for the first argument causes GetObject to act like CreateObject - it will create a new object of the class whose programmatic identifier is progID. The following table describes the results of using GetObject:

If the ActiveX application is loaded	Result
Set A = GetObject(Null, "MyProject.TransObject")	An existing TransObject reference is assigned to A.
Set A = GetObject("", "MyProject.TransObject")	The ActiveX application (MyProject) is started and its reference is assigned to A.

If the ActiveX application is not loaded	Result
Set A = GetObject(Null, "MyProject.TransObject")	An error occurs.
Set A = GetObject("", "MyProject.TransObject")	The ActiveX application (MyProject) is started and its reference is assigned to A.

GetRow Set< First/Last/Prior/Next> In <Set Name>

Set currency pointer for a child table in a set in the Transit object's database by walking the set.

GetRow Table< First/Last/Prior/Next> <Table Name>

Set currency pointer for a table in the Transit object's database by walking the physical table.

<long row count> = GetRowCountSet(<Set Name>)

Returns the count of rows in a Transall set.

<long row count> = GetRowCountTbl(<Table Name>)

Returns the count of rows in a Transall table.

<long row count> = GetRowNumberSet(<Set Name>)

Returns the number of the current row in a Transall set counting from the beginning of the set.

<long row count> = GetRowNumberTbl(<Table Name>)

Returns the number of the current row in a Transall table counting from the beginning of the table.

GetSetting(base_key, section, key [, value])

Reads a value from the registry key formed by joining the "base_key", "section", and "key" values together. The "base_key" may either be an existing base value or simply a user-defined base. If one of the "HKEY" values is not used, one will be built by appending the user-defined base name to the value

HKEY_CURRENT_USER\Software\Transall Program Settings\....

Note that, optionally, a default value may be provided to be returned if no value exists for the specified key. If no default is provided, a Null string is returned for undefined keys. Like the similarly-named Visual Basic function, this function only returns string values.

For example, to return to the current version of the Transall Editor:

GetSetting("HHKEY_LOCAL_MACHINE","SOFTWARE\Oracle Internaltional\Transall 12.1","LastInstallVersion","none")

<any> = GetVal (<string>)

Converts a string number with punctuation to a variant number without punctuation.

GoTo <label>

Branch to a line label within a procedure.

GoSub label

Causes the program execution to switch to the named label. When a **Return** statement is encountered (see below), program execution will return to the line following the **GoSub** statement. **GoSub** calls may be nested:

```
Dim Count As Integer

      Count = 10
      GoSub Countdown
      WriteConStdOut( "BLASTOFF!!" & _CRLF_)
      GoTo TheFinalFrontier

    Countdown:
      If ( Count > 0) Then
         WriteConStdOut( "  " & Count & _CRLF_)
         Count = Count - 1
         GoSub Countdown
      End If

      Return

    TheFinalFrontier:
      WriteConStdOut( "That was cool!" & _CRLF_)
```

produces:

10
9
8
7
6
5
4
3
2
1
BLASTOFF!!
That was cool!

<String> = Hex$(<Long>)

Returns a string that represents the hexadecimal value of a decimal argument

<integer> = Hour (<datetime>)

Returns the hour.

<String> = IDS_Get(<String Variable Name>)

Retrieves a value from the Oracle Internet Document Server (IDS) input queue environment.

IDS_Put(<String Variable Name>, <String Value>)

Sets a value in the Oracle Internet Document Server (IDS) output queue environment.

IidGUID {<GUID>}

(Global Only) Define the "root" Interface Global Unique ID (GUID) for the Transit class ActiveX object.

InsertRow <Table Name> (<Column> [,<Column>...]) Values (<constant> [,< constant>…])

(Global Only) Insert a row at compile time into a table in the Transit object's database.

InsertRow <Table Name> [([<Column>,<Column>...]) Values ([<expression>,<expression>…])]

Insert a row into a table in the Transit object's database

<long> = Instr(<numeric expression start>,<string expression to search>, <string expression looking for>, [<boolean case insensitive>])

Returns the index offset of a substring inside a larger string (the first character has an index of one)

<Long> InstrRev(<StringCheck>, <StringMatch>[, <Integer start> [, <Bool case in-sensitive>]])

Returns the position of an occurrence of the StringMatch string within the StringCheck string, looking from the end of the StringMatch string.

<numeric value> = Int (<numeric expression >)

Returns the integer portion of a number. If the number is negative, Int returns the first negative integer less than or equal to number.

<boolean integer> = IsDate(<Any>)

Returns non-zero if the expression is a valid date time.

<boolean integer> = IsNumeric(<Any>)

Returns non-zero if the expression is a number or can be processed as a number.

<boolean integer> = IsNull (<expression >)

Returns non-zero if expression evaluates to null.

<boolean integer> = IsNull(<Any>)

Returns non-zero if the expression is Null.

<boolean integer> = IsSetCurrent(<set>)

Returns non-zero if the set has a current row.

<boolean integer> = IsSetEmpty (<set>)

Returns non-zero if the set is empty.

<boolean integer> = IsTableCurrent (<table>)

Returns non-zero if the table has a current row.

<boolean integer> = IsTableEmpty (<table>)

Returns non-zero if the table is empty.

<String> = Join(<array>[, <separator string>])

Accepts an array, and optionally a String value. If no string value is provided, it defaults to a space (" "). The function builds a string, concatenating the elements of the array and separating them with the String value provided, and returns the newly-built String value. Note that if an empty string value is specified for the separator (""), then the elements of the array are concatenated with no intervening characters.

Jump [**Case**] <*testexpression*>
[**Case** <*Constant-1*>
[statementblock-1]]
[**Case** <*Constant-2*>
[statementblock-2]]
[**Case Like** <*String Constant-3*>
[statementblock-3]]
[**Case Like** <*String Constant-4*>
[statementblock-4]]
[**Case Else**
[statementblock-n]]
End Jump

The Jump Case statement functions like the Visual Basic "Select Case" statement, but requires constants for the Case statement blocks and also supports a "Like" operator on the Case statements. The expressions following Case statements under a Jump statement must be constant data references. Jump statements will tend to execute faster than Select Case statements when the number of Case blocks is greater than four.

Kill <String Expression File Name>

Deletes a file.

<string> = LCase$(<string expression>)

Returns a string with all letters converted to lowercase.

<string> = Left$(<string expression>, <numeric expression start>)

Returns characters from left side of a string.

<length of string> = Len (<string expression>)

Returns the number of characters in a string.

Let <Variable Expression>

Assigns the value of an expression to a variable.

LibGUID {<GUID>}

(Global Only) Define the Type Library Global Unique ID (GUID) for the Transit Class ActiveX object.

<numeric value> = Log (<numeric expression >)

Returns the natural logarithm of a number.

<Return value as any datatype> = LookupValSet (<set name>,
<Lookup In Column Name>,<Search For Value Expression>,
<Return From Column Name>[, <Boolean set lookup row current>])

Looks up an encode value in a Transall set and returns a decode value.

<Return value as any datatype> = LookupValTbl (<Lookup In Column Name>,
<Search For Value Expression>, <Return From Column Name>
[, <Boolean set lookup row current>])

Looks up an encode value in a Transall table and returns a decode value.

<string> = LTrim$(<string expression>)

Returns a string with leading (left most) spaces removed.

<string> = Mid$(<string expression>, <numeric expression start>
[, < numeric expression length>])

Returns characters from middle of a string.

<long> = Millisecond (<datetime>)

Returns the milliseconds.

<integer> = Minute (<datetime>)

Returns the minute.

MkDir <string path>

Creates a new directory

<integer> = Month (<datetime>)

Returns the month.

Name <oldname> As <newname>

Renames file <oldname> to <newname>. Due to differences in file systems, applications that use this may not be portable.

<datetime> = Now

Returns the current system date time.

<null value> = Null

Equates to Any datatype as a Null value.

Oct(<expression>)

Returns a string representation of the indicated value in octal (base 8). For example, Oct(7) returns "7" and Oct(8) returns "10".

<Database connection handle> = OdbcCommit (<Database connection handle expression>)

Commits an ODBC database transaction.

<Database connection handle> = OdbcConnect (<expression ODBC data source string>,<expression User ID string>,<expression password string>)

Establishes connections to an ODBC driver and an ODBC data source.

<Database connection handle> = OdbcConnectCsr (<expression ODBC data source string>,<expression User ID string>,<expression password string>)

Establishes connections to an ODBC driver and an ODBC data source. If a database connection already exists using the same Data source string, User ID and Password only a new cursor is defined on the existing database connection.

OdbcDisconnect (<Database connection handle expression>)

Closes the connection associated with a specific ODBC connection handle.

<Database connection handle> = OdbcDriverConnect (<expression ODBC connection string>)

Establishes connections to an ODBC driver and an ODBC data source.

<Database connection handle> = OdbcDriverConnectCsr (<expression ODBC connection string>)

Establishes connections to an ODBC driver and an ODBC data source. If a database connection already exists using the same connection string only a new cursor is defined on the existing database connection.

<Database connection handle> = OdbcDriverConnectPrompt (<expression ODBC connection string>)

Establishes connections to an ODBC driver and an ODBC data source and will prompt for additional connect information (if necessary).

<Database connection handle> = OdbcDriverConnectPromptCsr (<expression ODBC connection string>)

Establishes connections to an ODBC driver and an ODBC data source and will prompt for additional connect information (if necessary). If a database connection already exists using the same connection string only a new cursor is defined on the existing database connection.

OdbcExecute (<Database connection handle expression>)

Executes a SQL statement prepared via OdbcPrepare.

OdbcExecuteDirect(<Database connection handle expression >, <Constant String SQL Statement>)

Prepares and executes a SQL statement directly against an ODBC data connection (usually used to execute stored procedures).

OdbcExecuteDynam(<Database connection handle expression> [, Bound Input Variables…])

Executes a SQL statement prepared via OdbcPrepareDynam.

OdbcFetch (<Database connection handle expression>)

Fetches a row of data from a SQL result set created by OdbcExecute or OdbcRun.

<Long varname> = OdbcGetConnectionOptionLong (<Database connection handle expression>,<Integer Option expression>)

Gets options that govern aspects of ODBC connections. See ODBC Driver doc for values.

<String varname> = OdbcGetConnectionOptionString (<Database connection handle expression>,<Integer Option expression>)

Gets options that govern aspects of ODBC connections. See ODBC Driver doc for values.

OdbcGetErrorInfo (<Database connection handle expression >, <Error Message string varname>,<Error State string varname>,<ODBC Error return code integer varname>,<Native data source error return code long varname>)

Returns ODBC error or status information.

<Long varname> = OdbcGetStmtOptionLong (<Database connection handle expression>,<Integer Option expression>)

Gets options related to an statement on an ODBC connection. See ODBC Driver doc for values.

<String varname> = OdbcGetStmtOptionString (<Database connection handle expression>,<Integer Option expression>)

Gets options related to an statement on an ODBC connection. See ODBC Driver doc for values.

<Long varname> = OdbcGetSqlRowCount (<Database connection handle expression>)

Gets the number of rows affected by the last UPDATE, INSERT, or DELETE statement run on an ODBC connection.

OdbcMoreResults(<Database Handle>[,<Variable>...>])

Tests for and binds data buffers to secondary result-sets returned by SQL stored procedures.

OdbcPrepare (<Database connection handle expression >, <Constant String SQL Statement>)

Prepares a SQL string for ODBC execution.

OdbcPrepareDynam(<Database connection handle expression >, <Variable String SQL Statement>[, Bound Output Variables…])

Prepares a SQL string for ODBC execution.

OdbcRollback (<Database connection handle expression >)

Rolls back an ODBC database transaction.

OdbcRun (<Database connection handle expression >, <Constant String>)

Prepares and Executes a SQL string via ODBC.

OdbcSetConOptAutoCommit (<Database connection handle expression>, <Boolean expression>)

Sets transaction auto commit option on or off. The default is determined by the ODBC driver but is usually on.

OdbcSetConOptIsoReadCommited (<Database connection handle expression>)

Sets the transaction's record locking isolation level to READ COMMITTED for a database connection. This causes the server to acquire a shared lock while reading a row into a cursor and free the lock immediately after reading the row. Because a shared lock request is blocked by an exclusive lock, the cursor should be prevented from reading a row that another task has updated but not yet committed.

OdbcSetConOptIsoReadRepeatable (<Database connection handle expression>)

Sets the transaction's record locking isolation level to REPEATABLE READ for a database connection. This causes the server to acquire a shared lock while reading a row into a cursor and to hold the lock for the life of the SQL transaction. This works the same as OdbcSetConOptIsoReadRepeatable.

OdbcSetConOptIsoReadUncommited (<Database connection handle expression>)

Sets the transaction's record locking isolation level to READ UNCOMMITTED for a database connection. This causes the server to request no locks while reading rows into a cursor. This means that cursors can be populated with values that have been updated but not yet committed thereby bypassing all of locking transaction control mechanisms on the SQL server.

OdbcSetConOptIsoSerializable (<Database connection handle expression>)

Sets the transaction's record locking isolation level to REPEATABLE READ for a database connection. This causes the server to acquire a shared lock while reading a row into a cursor and to hold the lock for the life of the SQL transaction. This works the same as OdbcSetConOptIsoSerializable.

OdbcSetConOptIsoSerializable(<Database Handle>)

Sets the record locking isolation level to SQL_TXN_SERIALIZABLE for a database connection.

OdbcSetConOptReadOnly (<Database connection handle expression>, <Boolean expression>)

Sets the access mode option to read only or to read write. The default is determined by the ODBC driver but is usually read write.

OdbcSetConOptTrace (<Database connection handle expression>, <Boolean expression>)

Sets statement tracing option on or off. The default is determined by the ODBC driver but is usually off.

OdbcSetConOptTraceFile (<Database connection handle expression>, <String file name expression>)

Sets statement tracing option on and identifies the name of the file that trace information will be written to. The default file name is determined by the ODBC driver but it is usually "SQL.LOG".

OdbcSetConnectionOptionLong (<Database connection handle expression>, <Integer Option expression>,<Long Value expression>)

Sets options that govern aspects of ODBC connections. See ODBC Driver doc for values.

OdbcSetConnectionOptionString (<Database connection handle expression>, <Integer Option expression>,<String Value expression>)

Sets options that govern aspects of ODBC connections. See ODBC Driver doc for values.

OdbcSetPrepareOpt <Integer> (<Database Handle>,<Bool>)

Toggles SQL Prepare optimizations for a database connection and returns prior value of switch.

OdbcSetPrepareOptGlobal(<Bool>) <Integer>

Toggles global prepare optimization features and returns prior value of switch.

OdbcSetStmtOptionLong (<Database connection handle expression>, <Integer Option expression>,<Long Value expression>)

Sets options related to an statement on an ODBC connection. See ODBC Driver doc for values.

OdbcSetStmtOptionString (<Database connection handle expression>, <Integer Option expression>,<String Value expression>)

Sets options related to an statement on an ODBC connection. See ODBC Driver doc for values.

OdbcStatementCloseResult(<Database Handle>)

Closes and releases the result-set of a SQL statement.

OdbcStatementDrop(<Database Handle>)

Drops any compiled statement, closes and releases any result-set on a database handle.

<boolean integer> = Odd(<Any>)

Returns non-zero if the expression equates to an Odd number.

On Error Goto 0

Causes run-time errors to be honored.

On Error GoTo <label>

If an error occurs before the end of the current function or subroutine, control is passed to the designated label.

On Error Resume Next

Causes run-time errors to be ignored.

Open <filename> [For <mode>] [NoCreate] [NoReplace] [Access <access>] [<lock mode>] [AMParms: <AMParm, AMParm>…] As [#]<filenumber> [Len=<reclength>]

Open a file.

Parameter	Value
Mode	The valid choices are as follows: • **Append**—Opens file for sequential output in text (translated) mode. Sets initial file pointer to end of file. Creates file if does not exist. • **Binary Append**—Opens file for sequential output in binary (un-translated) mode. Sets initial file pointer to end of file. Creates file if does not exist. • **Output**—Opens file for sequential output in text (translated) mode. Sets initial file pointer to beginning of file. File contents are destroyed. Creates file if does not exist. • **Binary Output**—Opens file for sequential output in binary (un-translated) mode. Sets initial file pointer to beginning of file. File contents are destroyed. Creates file if does not exist. • **Input**—Opens file for sequential input in text (translated) mode. Sets initial file pointer to beginning of file. • **Binary Input**—Opens file for sequential input in binary (un-translated) mode. Sets initial file pointer to beginning of file. • **Binary**—Opens file for random input and output in binary (un-translated) mode. Sets initial file pointer to beginning of file. Creates file if does not exist. • **Random**—Opens file for random input and output in text (translated) mode. Sets initial file pointer to beginning of file. Creates file if does not exist. • **VSAM Append Random**—Opens VSAM file for random input and output in binary (un-translated) mode. Records can be retrieved or inserted, but not replaced or deleted. Records can be inserted at any point in the file, not just at the end. Creates file if does not exist. • **VSAM Append**—Opens a VSAM file for random output in binary (un-translated) mode. Records can be inserted, but not retrieved, replaced, or deleted. Records can be inserted at any point in the file, not just at the end. Creates file if does not exist. • **VSAM Input**—Opens VSAM file for random input in binary (un-translated) mode. Records can be retrieved, but not replaced, deleted, or inserted. If the file has not been loaded, the open will fail. • **VSAM Random**—Opens VSAM file for random input and output in binary (un-translated) mode. All file operations are permitted. If the file has not been loaded, the open will fail.

Parameter	Value
	• **VSAM Output**—Opens VSAM file for output in binary (un-translated) mode. If the file contains any records, they are erased. If the file is not defined as REUSABLE and it contains any records, the open will fail. This open mode enables records to be added to the file, but not to be retrieved, updated, or deleted. • **VSAM Output Random**—Opens VSAM file for output in binary (un-translated) mode. If the file contains any records, they are erased. If the file is not defined as REUSABLE and it contains any records, the open will fail. This open mode enables records to be added to the file, but not to be retrieved, updated, or deleted. • **NoCreate**—(Not applicable to VSAM) File must already exist to be opened. Overrides file creation for open modes Append, Output, Binary and Random. • **NoReplace**—(Not applicable to VSAM) File must NOT already exist to be opened. Overrides file content replace for open modes Append, Output, Binary and Random.
Access	(Not applicable to VSAM) The valid choices are as follows: • **Read**—File will be left read only if this process created file. • **Write**—File will be left write only if this process created file (same as Read Write). • **Read Write**—File will be left available for read and write if this process created file.
Lock Mode	(Not applicable to VSAM) The valid choices are as follows: • **Shared**—File will be available for all access to other process. • **Lock Read**—File will be unavailable for reading to other process while open. • **Lock Write**—File will be unavailable for writing to other process while open. • **Lock Read Write**—File will be unavailable for reading and writing to other process while open.

AMParm	(Not applicable to non-mainframe environments) The valid choices are as follows:
	• **recfm=f/v/u**—Operating system record format:
	  • **f—fixed**
	  • **v—variable**
	  • **u—undefined**
	• **Reclen=nnn|x**—Operating system record length.
	• **blksize=nnn**—Operating system block size.
	• **keylen=nnn**—VSAM key offset.
	• **keyoff=nnn**—VSAM key offset.
	• **org=value**—File organization:
	  • **PS—File is an ordinary sequential file, such as a CMS disk file, a tape file, or a CMS spool file. To read the directory, specify the value PS for a file that is a PDS.**
	  • **OS—File is an OS format file under CMS, such as a filemode 4 file or a file on an OS disk.**
	  • **PO—File is a partitioned data set, or a CMS MACLIB or TXTLIB. Under systems supporting PDSEs, the file can be either a regular PDS or a PDSE.**
	  • **PDS—File is a regular (non-PDSE) PDS.**
	  • **PDSE—File is a PDSE.**
	  • **KS—File is a VSAM KSDS.**
	  • **ES—File is a VSAM ESDS (limited support).**
	  • **BYTE—File is an OpenEdition HFS file.**
	• **print=yes|no**—File destined to be printed.
	• **page=nnn**—Line per page with print=yes.
	• **pad=no|null|blank**—File padding.
	• **trunc=yes|no**—Effect of output before end of file.
	• **grow=yes|no**—Controls whether new data can be added to file.
	• **order=seg|random**—Sequential or Random file access.
	• **bufnd=nnn**—Number of Data I/O buffers VSAM is to use.
	• **bufni=nnn**—Number of Index I/O buffers VSAM is to use.
	• **bufsize=nnn**—Maximum number of bytes of storage to be used by VSAM for file data and index I/O buffers.
	• **Len:nnn**—File buffer length. Default is 4096k if value not given or is zero.

<string header data> = PpsExpReadNextHdr$ (<file number>)

Read next header record from a PPS export file formatted with headers.

PpsExpReadVarData #<file number>, <string PPS variable field name>, <varname>[,<string PPS variable field name>, <varname>...]

Read variable data lines from a PPS export file.

PSSaveAsDCD(hPrintStream, szFilename)

This routine saves the current transaction to a file in DCD format.

Raise GeneralFailure(<text>)

The GeneralFailure event is raised by the Transall runtime engine whenever a runtime error occurs which has not already been handled by an "On Error" statement.

RaiseEvent <event name>[(<parm>[, ...])]

The "RaiseEvent" statement activates a previously-defined event. Events which are not directed to a handling routine by a "SendEvent" will simply be ignored.

Randomize

Initializes the random-number generator.

ReadFixed [#]<file number>, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>[, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>, …]

Read returns fewer bytes than requested at end of file.

<boolean integer> = ReadPrivateProfileRemove(<string filename>)

Drops Transall's cache of INI data for a single file. Returns True if file was cached.

ReadPrivateProfileRemoveAll()

Drops Transall's cache of INI file data.

<string> = ReadPrivateProfileSection(<string section name>, <string file name>)

Returns all values from a section of an INI formatted file (like WIN32 API GetPrivateProfileSection). Note, INI files are cached.

<long> = ReadPrivateProfileSetCacheSize(<long number of files to cache>)

Sets the number of INI files that Transall will cache. Returns old cache value.

<string> = ReadPrivateProfileString(<string section name>, <string key name>, <string default value>, <string file name>)

Returns a value from an INI formatted file (like WIN32 API GetPrivateProfileString). Note, INI files are cached.

ReadVariable #<file number>, <record separator>, <element separator>, <element delimiter>, <varname>, <varname>...

Read delimited records.

Replace$(<String>, <SearchString>, <ReplacementString>)

Returns a copy of a string after replacing search string values with replacement string values.

Resume [0]

When issued inside an error handler, returns control to the line that created the error condition.

Resume Next

When issued inside an error handler, returns control to the line FOLLOWING the one that created the error condition.

Return

Returns control to the line following the GoSub (see above) which called it. Note that attempting to execute a Return statement, which was not called via a GoSub, will produce a runtime error.

<string> = Right$(<string expression>, <numeric expression start>)

Returns characters from right side of a string.

RmDir <string path>

Removes an existing directory.

Rnd (<numeric expression seed value>)

Returns a random number.

Round(a[, B])

Uses BANKERS rounding to round a to the nearest B decimal places. If not supplied, B defaults to zero. BANKERS rounding is the type of rounding used by Visual Basic's Round() function. BANKERS rounding always rounds values of .5 to the nearest *even* integer. For instance 1.5 rounds to 2, but 2.5 also rounds to 2. This rounding is symmetric, in that the sign of a number does not affect the magnitude of the value that it rounds to. For instance, 2.5 and -2.5 round to 2 and -2, respectively. For more information about this see: http://support.microsoft.com/support/kb/articles/Q196/6/52.ASP

Round45(a[, B])

Uses 4/5 or arithmetic rounding to round a to the nearest B decimal places. If not supplied, B defaults to zero. Arithmetic is the type of rounding most commonly taught and used, where anything below .5 rounds down, and values of .5 and above round up in magnitude. With arithmetic rounding, 1.5 rounds to 2, 2.5 rounds to 3, and 2.4 rounds to 2. This function rounds symmetrically, so that 2.5 and -2.5 round to 3 and -3, respectively.

Round45A(a[, B])

Uses 4/5 or arithmetic asymmetric rounding to round a to the nearest B decimal places. If not supplied, B defaults to zero. This function behaves in a manner similar to the Round45() function, except that this function includes the sign of the number when deciding whether to round up or down. As with the Round45() function 1.5 rounds to 2, 2.5 rounds to 3, and 2.4 rounds to 2. For negative values, however, -2.4 rounds to -2 and -2.5 also rounds to -2, respectively.

<string> = RTrim$(<string expression>)

Returns a string with trailing (right most) spaces removed.

SaveSetting base_key, section, key, value

Saves a string value to the registry key formed by joining the "base_key", "section", and "key" values together. The "base_key" may either be an existing base value, such as:

HKEY_CLASSES_ROOT
HKEY_CURRENT_CONFIG
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS,HKEY_PERFORMANCE_DATA
HKEY_DYN_DATA

or simply a user-defined base. If one of the "HKEY" values is not used, one will be built by appending the user-defined base name to the following value:

HKEY_CURRENT_USER\Software\Transall Program Settings\....

<integer> = Second (<datetime>)

Returns the second.

Seek [#]<filenumber>, <position>

Determines the read/write position of a file.

<current read/write location offset> = Seek (<numeric expression file>)

Returns the read/write position of a file.

SendEvent <event name> To <handler routine name>

The "SendEvent" statement is used to direct an event to a particular subroutine which will handle the event. An event must be defined before it may be used.

"SendEvent" statements may be nested, allowing you to handle an event differently depending upon other conditions within your program.

Before control is passed to a handler routine by a "RaiseEvent" statement, the handler for the event is set to the previous event handler. Upon return from the routine, the event handler is restored. For instance if event "GeneralFailure" is currently being directed to routine "A", and prior to that it was handled by Transall, and a "GeneralFailure" exception is raised, before passing control to routine "A", Transall will reset the "GeneralFailure" event back to what was handling it before (Transall in this case). Once routine "A" has returned, Transall will restore routine "A" as the handler for a "GeneralFailure" event. The reason for this is twofold. First, it allows the user-defined event handler to pass control to the system-defined event handler, if necessary. Second, it helps to prevent infinite looping situations from occurring in cases where your event handling routine generates (perhaps inadvertently) the event that it is handling, such as a "GeneralFailure" event.

Events may be defined to have optional parameters. Note that if optional values are specified by a handler routine, they are ignored. The only place that optional values may be specified for an event is in the event definition itself. (This is because it is impossible for the compiler to know which event handler might be active at runtime.)

SendEvent GeneralFailure To <handler routine name>

The event handler for a GeneralFailure event is passed one parameter, a String which contains the error message for the error.

If you raise a GeneralFailure event within your program that is not processed by an event handler, the program will display the message passed to it, and exit. If you pass no parameter to the event, the default parameter of the last error message is used.

SetAttr <filename>, <attribute>

Changes the file system attributes for the indicate file. The attribute may be a hard coded value or a sum of one or more of the following constants:

TRAN_UNIX_SUID
TRAN_UNIX_SGID
TRAN_UNIX_RWXU
TRAN_UNIX_RUSR
TRAN_UNIX_WUSR
TRAN_UNIX_XUSR
TRAN_UNIX_RWXG
TRAN_UNIX_RGRP
TRAN_UNIX_WGRP
TRAN_UNIX_XGRP
TRAN_UNIX_RWXO
TRAN_UNIX_ROTH
TRAN_UNIX_WOTH
TRAN_UNIX_XOTH
TRAN_WIN32_ARCHIVE
TRAN_WIN32_SYSTEM
TRAN_WIN32_HIDDEN
TRAN_WIN32_READONLY
TRAN_WIN32_NORMAL

You may add constants from different operating systems if you wish, to allow the function to provide similar functionality on either platform.

SetNull (<variable>)

Sets the value of a variable to null.

SetTblBufInfo(<Table Name>, <Long MaxRowsToKeep>, <Long RowsToFlush>)

Sets the maximum number of rows to be kept in memory for a Transall database table and sets the number of rows to be flushed to disk when the maximum is exceeded for a single table in Transall. These values default to 2GB and 10,000.

SetTblDefaultBufInfo(<Long MaxRowsToKeep>, <Long RowsToFlush>)

Sets the maximum number of rows to be kept in memory for all Transall database tables and sets the number of rows to be flushed to disk when the maximum is exceeded for all tables in Transall. These values default to 2GB and 10,000.

SetTblSwapName(<String SwapFileName>)

Sets the name of the swapfile that Transall should use should table rows need to be flushed to disk when the maximum number of rows is exceeded.

SetTblSwapSize(<Long SizeInBytes>)

Sets the maximum size of the swapfile that Transall will create should table rows need to be flushed to disk when the maximum number of rows is exceeded. When new swap files must be created the file name is appended with a number.

<numeric value> = Sgn (<numeric expression >)

Returns an integer indicating the sign of a number.

<numeric value> = Shell(<string expression command> [, <integer WindowStyle>])

This function runs an executable program as a separate process. A value of zero is returned if an error occurred. On Windows platforms, an integer value indicating the style of the window of the created process may optionally be specified. On non-Windows platforms, this value is ignored. Please note that certain commands, batch files, or shell scripts may require that they be passed as a command to a system command shell, such as ksh, bash, or cmd.exe to function properly.

<numeric value> = Sin (<numeric expression >)

Returns the sine of an angle.

SortSet(<column name>, <set name>[,<boolean descending sort>])

Sorts all rows in a set by a column.

SortTable (<column name>[,<boolean descending sort>])

Sorts all rows in a table by a column.

SortTableMulti(Tbl.column, descend [,tbl.column2, descend2 ...])

Allows sorting of Transall tables by more than one column. For each column named, you must specify a value to indicate whether the column will be sorted in ascending or descending order. A non-zero value indicates that the specified column will be sorted in descending order.

Space(<count>)

Returns a string consisting of the indicated number of spaces.

<numeric value> = Sqr (<numeric expression >)

Returns the square root of a number.

Static <varname> As <type> [Init <value>]

Causes a variable to be defined whose contents do not change between function or subroutine invocations. In this manner they are similar to a global variable, but unlike a global variable, the variable name is visible only to the function or subroutine in which the variable is declared. Using the optional "Init <value>" clause causes "<value>" to be stored in the variable before the variable is first used.

```
-----------------------------------------------
Public Sub Run()
        Dim Count As Integer

        Count = 10

        While( Count > 0)
            Call CountOff()
            Count = Count - 1
        Wend
    End Sub


    Public Sub CountOff()
        Static Count As Integer Init 1

        WriteConStdOut( "   " & Count & _CRLF_)
        Count = Count + 1
    End Sub
-----------------------------------------------
```

produces:

1
2
3
4
5
6
7
8
9
10

StrComp(<string1>, <string2>[, <case_insensitive>])

Compares two strings. It returns -1 if string1 comes before string2, 0 if the two strings are equal, or 1 if string1 comes after string2. "<case_insensitive>", if used, should be either 0 to indicate a case sensitive compare (the default) or 1 to indicate that case should be ignored when comparing the strings.

<string> = String$(<numeric expression>,<string expression>)

Returns a string consisting of the string expression repeated numeric expression times.

StrReverse(<string>)

Returns a string which is identical to the string passed in, except the order of the characters is reversed.

<Access Specifier> Sub <name> ([[Optional] [ByVal] <Parm> As <Data Type>,...
[= <default>]])

<statements...>
End Sub

Method definition

Parameter	Value
Access Specifier	The valid choices are as follows: • **Public**—Items are accessible from any method. • **Protected**—Items are accessible from member methods and inheriting classes. • **Private**—Items are accessible from member methods.
Optional	Preface the parameter definition with the keyword "Optional" and supply a default value to the parameter(s).
ByVal	Indicates the value should be passed by a copy value on the stack rather than by a pointer back to the original value.
ByRef *default *	Indicates the value should be passed by a reference pointer back to the original value rather than by a copy value on the stack.
Data Type	The valid choices are as follows: • **Integer**—16-bit signed integer. • **Long**—32-bit signed integer. • **Long Long**—64-bit signed integer. • **String**—BSTR up to 2 GB in length. • **DateTime**—Date/Time structure in the form of CCYY/MM/DD.HH.MI.SS.MMMMMMMMM. • **Float**—32-bit (4-byte) floating-point number. • **Double**—64-bit (8-byte) floating-point number. • **PNum(dig left[, dig right[, "S" or "U"]])**—Packed zoned-decimal field. • **UNum(dig left[, dig right[, "S" or "U"]])**—Zoned-decimal field. • **<User-defined type>**—See Type statement.
= <default>	Preface the parameter definition with the keyword "Optional" and supply a default value to the parameter(s).

Using optional default values:

```
Public Sub Sub( Optional ByVal name As type = <default>)
  *
  *
  *

  Public Sub Count( Optional ByVal First As String = "Uno", _
                    Optional ByVal Second As String = "Dos", _
                    Optional ByVal Three  As String = "Tres")

        *
        *
        *
  Call Count()          ' Just like Call Count("Uno", "Dos", "Tres")
  Call Count( 1)        ' Just like Call Count( 1, "Dos", "Tres")
  Call Count( 1, 2)     ' Just like Call Count( 1, 2, "Tres")
  Call Count( 1, 2, 3)
```

Note that once an optional parameter is defined, all subsequent parameters must be defined with default values as well.

Switch(<expr 1>, <choice1>[, ... <expr n>, <choice n>])

Similar to the Choose() function, it evaluates each expression in the list and returns the choice corresponding to the first expression that evaluates to TRUE. The expression may be a variable, a function call, or an expression such as "value < 7". Note however, that if the expression uses an operator, the expression should be enclosed in parentheses, such as "(value < 7)". Switch() evaluates *every* choice and expression in the list, even though only one is returned, so you should be careful to avoid unwanted side effects.

<numeric value> = Tan (<numeric expression >)

Returns the tangent of an angle.

<datetime> = Time

Returns the current system time.

TimeSerial(<hour>, <minute>, <second>)

Returns a DateTime value reflecting the indicated time. Note that relative values may be used for the hour, minute, or second values. Negative values indicate that much time before of that hour, minute or second. For example, TimeSerial(3, -15, 0) indicates 15 minutes before 3:00, or 2:45.

TransAllBeginTrans

This causes all Transall database working storage tables to begin transaction logging.

TransAllEndTrans

This causes all Transall database working storage tables to end transaction logging (i.e. commit).

TransAllRollbackTrans

This causes all Transall database working storage tables to return to the state they were in when TransAllBeginTrans was last called.

TransLogTransMessage(<String Message>)

Writes a general message to the Transall Recap Log File.

<string> = Trim$(<string expression>)

Returns a string with both leading (left most) and trailing (right most) spaces removed.

<Bool> = TrnSys$InDocuflex

Returns True when running under the ISITAVM Transall host for Docuflex.

<Bool> = TrnSys$InTranDynm

Returns True when running under the TRANDYNM Dynamic Load Module Transall host.

<Bool> = TrnSys$InTranExe

Returns True when running under the TRANEXE Batch Command Line Transall host.

<Bool> = TrnSys$InTranHost

Returns True when running under the TRANHOST ActiveX COM Transall host.

<Bool> = TrnSys$InTranRule

Returns True when running under the TRANRULE Transall host for IDS.

TrnSys$LocaleDefCurrencySym <String>

Returns the default currency symbol returned by Transall from the Format$ command.

TrnSys$LocaleDefDateSym <String>

Returns the default date separator symbol returned by Transall from the Format$ command.

TrnSys$LocaleDefDecimalSym <String>

Returns the default decimal symbol returned by Transall from the Format$ command.

TrnSys$LocaleDefDigitGroupSym <String>

Returns the default digit grouping symbol returned by Transall from the Format$ command.

TrnSys$LocaleDefNegativeSym <String>

Returns the default negative symbol returned by Transall from the Format$ command.

TrnSys$LocaleDefTimeAmSym <String>

Returns the default AM time symbol returned by Transall from the Format$ command.

TrnSys$LocaleDefTimePmSym <String>

Returns the default PM time symbol returned by Transall from the Format$ command.

TrnSys$LocaleDefTimeSym <String>

Returns the default time separator symbol returned by Transall from the Format$ command.

TrnSys$LocaleSetCurrencySym (<String New Value>)

Sets the currency symbol returned by Transall from the Format$ command.

TrnSys$LocaleSetDateSym (<String New Value>)

Sets the date separator symbol returned by Transall from the Format$ command.

TrnSys$LocaleSetDecimalSym (<String New Value>)

Sets the decimal symbol returned by Transall from the Format$ command.

TrnSys$LocaleSetDigitGroupSym (<String New Value>)

Sets the digit grouping symbol returned by Transall from the Format$ command.

TrnSys$LocaleSetNegativeSym (<String New Value>)

Sets the negative symbol returned by Transall from the Format$ command.

TrnSys$LocaleSetTimeAmSym (<String New Value>)

Sets the AM time symbol returned by Transall from the Format$ command.

TrnSys$LocaleSetTimePmSym (<String New Value>)

Sets the PM time symbol returned by Transall from the Format$ command.

TrnSys$LocaleSetTimeSym (<String New Value>)

Sets the time separator symbol returned by Transall from the Format$ command.

<Bool> = TrnSys$OnAIX

Returns True when running on AIX

<Bool> = TrnSys$OnLinux

Returns True when running on Linux

<String> = TrnSys$OnPlatform

Returns a string such as "Windows", "AIX", "Linux", and "Solaris"

<Bool> = TrnSys$OnSolaris

Returns True when running on Solaris

<Bool> = TrnSys$OnUnix

Returns True when running on Any Unix platform

<Bool> = TrnSys$OnWin32

Returns True when running on 32bit Windows

<Number> = True

Can be used in conitional expressions or to set a variable's value True.

[<access specifier>] Type <user type name> [Packed] <elementname> [(subscripts)]
As <Data Type> [<elementname> [(subscripts)] As <Data Type>]

Defines a user-defined data type

<long> = UBound(<array>)

Returns the upper boundary of the array.

VERSION <0-9999>[.<0-9999>[.<0-9999>[.<0-9999>]]]

Such as:

VERSION 3.6.0.24

VERSION 2.17

Note that there should be no other text (aside from leading spacing) on the line as the VERSION statement, and that any non-specified values default to zero. In addition, new options were added to the TranCC and TranExe commands to support the VERSION statement:

For the TranCC command, the "/AIB" (Auto-Increment Build number) option was added. Whenever this option is used, the compiler will increment the version number found in the source file (if no VERSION was specified, it defaults to 1.0.0.0) and writes the new VERSION statement into the source file.

For the TranExe command, the "/info" option was added. This not only allows the user to see the user to see the version number that was supplied by the VERSION statement, but also shows the version of TranCC that was used to build the executable, and the names of public methods that are defined in the executable.

Weekday (<DateTime>)

Returns an value between 1 (Sunday) and 7 (Saturday) that indicates the day of the week for the datetime argument.

With <base name>

<statements...>
End <base name>

The "With" statement allows you to specify the prefix of structure variables so that you do not have to retype the base:

```
Private Type Categories Packed
  a As String
  b As String
  c As String
  d As String
End Type

 *
 *
 *

Dim FileFolder As Categories

With FileFolder
   .a = "Apple"
   .b = "Banana"
   .c = "Candy"
   .d = "Dessert"
End With
```

is the same as typing:

```
Private Type Categories Packed
  a As String
  b As String
  c As String
  d As String
End Type

 *
 *
 *

 FileFolder.a = "Apple"
 FileFolder.b = "Banana"
 FileFolder.c = "Candy"
 FileFolder.d = "Dessert"
```

Note also that it is possible to nest one "With" block inside another:

```
Private Type Categories Packed
  a As String
  b As String
  c As String
  d As String
End Type
```

While <expr>

<statements...>
Wend

The While...Wend construct allows you to build a loop in which the looping condition is tested each time, before the code is executed.

<boolean integer> = WriteConStdErr (<string expression>)

Returns False if the value of the string expression is written to the console's standard error output. Returns True if write failed.

<boolean integer> = WriteConStdOut (<string expression>)

Returns False if the value of the string expression is written to the console's standard output. Returns True if write failed.

WriteFixed [#]<file number>, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>[, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>, …]

Write fixed length records.

Parameter	Value
Const String Data-type	The valid choices are as follows: • **DISPLAY**—Character data. • **ZONED DECIMAL**—Numeric data. • **8-bit**—1 byte. • **16-bit**—2 bytes (WORD). • **32-bit**—4 bytes (DWORD or FLOAT). • **64-bit**—Eight binary bytes (DOUBLE). • **COMP [SYNC]**—Binary, 2 bytes for 1 through 4 digits, 4 bytes for 5 – 9, 8 bytes for 10 – 18. • **COMP-1 [SYNC]**—4 Bytes floating point (FLOAT). • **COMP-2 [SYNC]**—8 Bytes floating point (DOUBLE). • **COMP-3 [SYNC]**—Packed decimal. • **COMP-4 [SYNC]**—Same as COMP.
Const String Sign	The valid choices are as follows: • **S**—Signed. • **L**—Signed leading. • **T**—Signed trailing. • **U**—Unsigned.

WriteVariable #<file number>, <record delimiter>, <element delimiter>, <data delimiter>, <expression> [, <expression>...]

Write variable length records

WriteXmlDocumentSet(<root table>, <string file name>,<string DTD information>)

Writes an XML document from a Transall database hierarchy.

WriteXmlDocumentSet(<Table>, <File Name String>, <Declaration File Name String>)

Writes an XML document from a Transall database hierarchy.

<string> = UCase$(<string expression>)

Returns a string with all letters converted to uppercase.

UpdateRow <Table Name> ([<Column>,<Column>...]) Values ([<expression>,<expression>…])

Update the current row in a table in the Transit object's database

<double> = Val (<string>)

Converts a string to a double.

<Error code As Integer> = VdrAddExplicitForm (<pRFMTTbl As Long>, <MemName As String>, < Revision As Long>)

Adds one explicit form and revision level to the list of forms used for this merge set.

<Error code As Integer> = VdrAddFormsLibrary (<pRFMTTbl As Long>, <EDLName As String>)

Supplies the name of an EDL to be search by the merge assembly engine while resolving form names listed the VRF.

<Error code As Integer> = VdrAddTag (<pRFMTTbl As Long>, <TagName As String>, <TagData As String>, <DataLen As Integer>)

Writes a Tag name and its data directly to the VRF.

<Error code As Integer> = VdrBeginReformatter (<pRFMTTbl As Long>, <UserIDName As String>)

Establishes communications with the reformatter system.

<Error code As Integer> = VdrBuildMergeSet (<pRFMTTbl As Long>, <DataBuf As String>, < BufLen As Long>)

Passes the merge record to the reformatter system.

VdrCallMrgUserOnWin32 <boolean value expression>

True indicates that Transall should call MRGUSER.W32 (database rulebase) and False indicates that Transall should call DMKUSER.W32 (filebase rulebase) to perform VDR operations on non 370 platforms.

<integer error code> = VdrCloseVRF (<pRFMTTbl As Long>,<VRFName As String>)

Closes one VRF file at a time.

<integer error code> = VdrCloseVRFs (<pRFMTTbl As Long>)

Closes all VRF files.

<long return code> = VdrDmgrfmtGetReasonCode(<pRFMTTbl As Long>)

Returns the RFCB-REASON-CODE value from the RBCB-CONTROL-BLOCK when Transall is calling DMGRFMT

<long return code> = VdrDmgrfmtGetReturnCode(<pRFMTTbl As Long>)

Returns the RFCB-RETURN-CODE value from the RBCB-CONTROL-BLOCK when Transall is calling DMGRFMT

<Old Value As String > = VdrDmgrfmtSetAllowMissingRulebase(pRFMTTbl As Long, sAllowMissingRulebaseFlag As String)

Sets the RFCB-ALLOW-MISSING-RULEBASE value in the RBCB-CONTROL-BLOCK when Transall is calling DMGRFMT

<Old Value As String > = VdrDmgrfmtSetAlternateParmList(pRFMTTbl As Long, sAlternateParmListFlag As String)

Sets the RFCB-ALTERNATE-PARM-LIST value in the RBCB-CONTROL-BLOCK when Transall is calling DMGRFMT

<Old Value As String > = VdrDmgrfmtSetEndMergeSet(pRFMTTbl As Long, sEndMergeSetFlag As String)

Sets the RFCB-END-MERGESET value in the RBCB-CONTROL-BLOCK when Transall is calling DMGRFMT

<Old Value As String > = VdrDmgrfmtSetMaxMessageLevel(pRFMTTbl As Long, sMessageLevel As String)

Sets the RFCB-MAX-MESSAGE-LEVEL value in the RBCB-CONTROL-BLOCK when Transall is calling DMGRFMT

<Old Value As String > = VdrDmgrfmtSetNoEffDateMsg(pRFMTTbl As Long, sNoEffDateMsgFlag As String)

Sets the RFCB-NO-EFF-DATE-MSG value in the RBCB-CONTROL-BLOCK when Transall is calling DMGRFMT

<Old Value As String > = VdrDmgrfmtSetSuppressImplictForms(pRFMTTbl As Long, sSuppressImplictFormsFlag As String)

Sets the RFCB-SUPPRESS-IMPLICIT-FORMS value in the RBCB-CONTROL-BLOCK when Transall is calling DMGRFMT

<Old Value As String > = VdrDmgrfmtSetSuppressSysPrint(pRFMTTbl As Long, sSysPrintFlag As String)

Sets the RFCB-SUPPRESS-SYSPRINT value in the RBCB-CONTROL-BLOCK when Transall is calling DMGRFMT

<Old Value As String > = VdrDmgrfmtSetVrfAllocDDname(pRFMTTbl As Long, sDDname As String)

Sets the RFCB-VRF-DDNAME value in the RBCB-CONTROL-BLOCK when Transall is calling DMGRFMT

<Old Value As String > = VdrDmgrfmtSetWriteExplictForms(pRFMTTbl As Long, sWriteRfcbFlag As String)

Sets the RFCB-WRITE-EXPLICIT-FORMS value in the RBCB-CONTROL-BLOCK when Transall is calling DMGRFMT

<Old Value As String > = VdrDmgrfmtSetWriteExplictForms(pRFMTTbl As Long, sWriteExplictFormsFlag As String)

Sets the RFCB-WRITE-EXPLICIT-FORMS value in the RBCB-CONTROL-BLOCK when Transall is calling DMGRFMT

<Old Value As String > = VdrDmgrfmtSetWriteRfcb(pRFMTTbl As Long, sWriteRfcbFlag As String)

Sets the RFCB-WRITE-RFCB value in the RBCB-CONTROL-BLOCK when Transall is calling DMGRFMT

<Error code As Integer> = VdrEndMergeSet (<pRFMTTbl As Long>)

Marks the end of the merge record.

<Error code As Integer> = VdrEndReformatter (<pRFMTTbl As Long>)

Ends communications with the reformatter program.

<Error code As Integer> = VdrSetEffectiveDate (<pRFMTTbl As Long>, <EffDate As String>)

Sets the effective date of form images to be selected in the EDL.

<Error code As Integer> = VdrSetJobDescription (<pRFMTTbl As Long>, <JobDescription As String>)

Allows the user to specify a unique job description for a job submitted to the Documaker queue.

<Error code As Integer> = VdrSetMessageFile (<pRFMTTbl As Long>, <PathFileName As String>)

Sets the path and file name to where the Documaker messages with be written.

<Error code As Integer> = VdrSetProductionDefinition (<pRFMTTbl As Long>, <ProdDef As String>)

Sets the name of the production definition to use for immediate merging.

<Error code As Integer> = VdrSetRulebase (<pRFMTTbl As Long>, <Rulebase As String>, <RevLvl As Long>)

Sets the rulebase name and revision level from which to retrieve the merging rules.

<Error code As Integer> = VdrSetVRFFile (<pRFMTTbl As Long>, <VRFName As String>)

Sets the path and file name of the VRF file to be created.

<Error code As Integer> = VdrSetWorkDirectory (<pRFMTTbl As Long>, <Directory As String>)

Sets the directory for temporary work files.

<Error code As Integer> = VdrStartMergeSet (<pRFMTTbl As Long>)

Marks the start of a merge set in the VRF file.

<Error code As Integer> = VdrSubmit (<pRFMTTbl As Long>, <ProductionDef As String>, <VDRpathAndFile As String>)

Submits a VRF to the Documaker merge server processing queue.

VsamDelete [#]<file number>

Deletes the current record from the VSAM file. After a deletion of a record, the file is positioned to the next record in the file.

VsamInsert [#]<file number>, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>[, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>, …]

Adds a new record to the VSAM file. After a successful VsamInsert the file is positioned to the record following the one inserted.

VsamRead [#]<file number>, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>[, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>, …]

Retrieves the next record from the VSAM file. The record is retrieved for update (if the file's open mode permits writing). For a file with duplicate keys, records with the same key are always retrieved in the order in which they were added to the file.

VsamReadPrior [#]<file number>, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>[, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>, …]

Retrieves the prior record from the VSAM file. The record is retrieved for update (if the file's open mode permits writing). For a file with duplicate keys, records with the same key are always retrieved in the order in which they were added to the file, not in the reverse of this order because of this being a prior read request.

VsamReplace [#]<file number>, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>[, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>, …]

Replaces the current record of the VSAM file. Replacement of the record does not change the file position. However, the updated record is no longer current and must be retrieved again before another update.

VsamSearch [#]<file number>, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>[, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>, …]

Searches forward in the VSAM file for a record matching the generic key specification. The first record found with a key greater than or equal to the key specified is considered to match.

VsamSearchBack [#]<file number>, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>[, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>, …]

Searches backward in the VSAM file for a record matching the generic key specification. The first record found with a key greater than or equal to the key specified is considered to match.

VsamSearchBackExact [#]<file number>, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>[, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>, …]

Searches backward in the VSAM file for a record matching the exact key specification.

VsamSearchExact [#]<file number>, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>[, <const string data-type>, <number places to left of decimal>, <number places to right of decimal>, <const string sign>, <any data type varname>, …]

Searches forward in the VSAM file for a record matching the exact key specification.

<integer> = Year (<datetime>)

Returns the year.

XmlClose(<long file handle>)

Closes an XML file handle.

XmlDeclareColAttribute(<table column>, <string attribute name>)

Declares that a table column contains XML element attribute data.

XmlDeclareColCDATA(<table column>)

Declares that a table column contains XML string data.

XmlDeclareColComment(<table column>)

Declares that a table column contains XML comment data.

XmlDeclareColProcInstr(<table column>)

Declares that a table column contains XML processing instructions.

XmlDeclareColStringData(<table column>)

Declares that a table column contains XML string data.

XmlDeclareTableElement(<table>, <string element name)

Declares that a table contains XML element data.

<boolean integer> = XmlEncodingTranscode(<boolean integer encode output>)

Tells Transall to encode XML data to the default encoding scheme as it is written.

XmlGetDeclaration (<long file handle>, <long attribute type>, <string attribute name>, <string variable attribute value> [,<long attribute type>, <string attribute name>, <string variable attribute value>…])

Reads XML declaration information from an XML document.

Parameter	Value
Attribute Type	The valid choices are as follows: • **8**—XML Declaration Information. • **9**—Document Type Declaration information.

XmlGetDTD(<long file handle>, <long attribute type>, <string element name>, <string attribute name>, <string variable value> [,<long attribute type>, <string element name>, <string attribute name>, <string variable value>…])

Reads data from an XML document's DTD, if one exists.

Parameter	Value
Attribute Type	The valid choices are as follows: • **11**—Element declaration. • **12**—Attribute declaration. • **13**—Entity declaration. • **14**—Notation declaration.

XmlGetElementData(<long file handle>, <long attribute type>, <string attribute name>, <string variable value> [,<long attribute type>, <string attribute name>, <string variable value>…])

Reads data from an XML document.

<boolean integer success> = XmlGetElementData (<long XML file handle>, <long XML item type>, <string name of XML item>, <any variable to receive XML file data>[,<long XML item type>, <string name of XML item>, <any variable to receive XML file data>]...)

Retrieves the XML data from an XML file.

Parameter	Value
Attribute Type	The valid choices are as follows: • **0**—Root element. • **1**—Element Data. • **2**—Character Data. • **3**—CDATA. • **4**—Processing Instruction. • **5**—Comment. • **6**—End. • **15**—Attribute Data. • **16**—Attribute Data Iteration–first attribute, second attribute, ….

<long result> = XmlGetNextElement (<long file handle>, <variable string XML element name>, <long XML chunk type>)

Reads the next chunk of information from an XML document.

Parameter	Value
Result	The valid choices are as follows: • **0**—OK. • **1**—Warning – See Err.Description. • **2**—Recoverable error – See Err.Description. • **3**—Unrecoverable error – See Err.Description.
XML Chunk Type	The valid choices are as follows: • **1**—Root. • **2**—Character Data. • **3**—CDATA. • **4**—Processing Instruction. • **5**—Comment. • **6**—Element End.

<long file handle> = XmlOpen (<string filename>, <string encoding value>, <long validation flag>, <long name space flag>, <long autoencoding boolean>)

Opens an XML file for reading and returns a handle to the file.

Parameter	Value

Encoding Value	The valid choices are as follows:
	• **DEFAULT or spaces**—Used when Auto Encoding enabled.
	• **Available encoding schemes:**
	  • **ISO-8859-1**
	  • **UTF-8**
	  • **UTF-16**
	  • **LATIN1**
	  • **UCS-4**
	  • **WINDOWS-1252**
	  • **IBM1140**
	  • **IBM037**
	  • **USASCII**

Validation Flag	The valid choices are as follows: • **0**—No Validation. • **1**—Always Validate. • **2**—Automatically Validate.
Name Space Flag	The valid choices are as follows: • **0**—No namespace processing. • **1**—Support namespace processing.
Autoencoding Boolean	The valid choices are as follows: • **0**—No autoencoding. • **1**—Auto encode–parser uses original encoding for document.

XmlSetEncoding(<String Encoding>)

Sets the encoding character set for an XML document.

<boolean integer> = XmlWriteWhiteSpace(<boolean integer include whitespace>)

Tells Transall to write XML files with whitespace for readability.

<long old value> = XmlWriteWhiteSpace(<boolean flag>)

Causes Transall to write white space characters when writing XML documents.

WEB ACCESS FUNCTIONS

The following functions give your Transall application the ability to send and receive information from web services. Two sets of functions are provided. The simple functions provide the ability to retrieve a web page or post information to a web service and retrieve the results as a single function call. In each case, the data may be stored in either a file or a string. For clients that require more control of the data exchange or are looking to carry out more than one exchange in a single connection, there is a set of advanced functions.

Note These functions do not work with a proxy server currently.

For the function descriptions, the following terms are applied.

URL : The complete description of a remote HTTP resource in the form:

http://server[:port][/resource...]

The "http://" prefix is required. The port number (":port") specification is optional. If not included, it defaults to port 80 for the simple functions. For the advanced functions, it defaults to the value provided by the user in a previous call to **Http_Port**, or to 80 if no default port was specified.

ResponseCode

A numeric return code within the ranges is specified in the HTTP specifications, with values of 200-299 indicating some degree of success. In addition, Transall has added two other code values: A value of 998 indicates a data transmission error and 999 indicates a file I/O error.

SIMPLE FUNCTIONS

<ResponseCode> = HttpGetString(<URL As String>, <ByRef Response As String>)

This function issues a request to get the resource specified by the URL from the remote server and stores the response from the server into the provided string. The response code from the request is returned to the user.

<ResponseCode> = HttpGetFile(<URL As String>, <FileName As String>)

This function issues a request to get the resource specified by the URL from the remote server and stores the response from the server into the named file. The response code from the request is returned to the user.

<ResponseCode> = HttpPostFileToString(< URL As String>, <FileName As String>, <ByRef Response As String>)

This function posts the contents of the named file to the server named by the URL. The response from the server is stored into the provided string, and the response code is returned to the user.

<ResponseCode> = HttpPostFileToFile(< URL As String>, <SourceFileName As String>, <DestFileName As String>)

This function posts the contents of the named file to the server named by the URL. The response from the server is stored into the named file, and the response code is returned to the user.

<ResponseCode> = HttpPostStringToString(< URL As String>, <SendData As String>, <ByRef Response As String>)

This function posts the contents of the provided string to the server named by the URL. The response from the server is stored into the provided string, and the response code is returned to the user.

<ResponseCode> = HttpPostStringToFile(< URL As String>, <SendData As String>, <DestFileName As String>)

This function posts the contents of the provided string to the server named by the URL. The response from the server is stored into the named file, and the response code is returned to the user.

ADVANCED FUNCTIONS

<Bool> = Http_Connect(<ByRef Handle As Long>, <URL As String>)

This function allocates a connection handle to the named resource and stores it into the handle provided. A non-zero value is returned if this call was successful. The remote server is not contacted until a data transmission is requested. Note that if subsequent requests are for a different resource on the same server, this function may be called without first calling Http_Close. If the remote server supports pipelining, the request will be sent without having to close the connection and then reallocate a new one.

<Boolean> = Http_IsConnected(<Handle As Long>)

This function returns a non-zero value if the provided handle is currently connected to a remote server.

Http_Timeout(<Handle As Long>, <Timeout As Long>)

This function sets the default timeout value (in milliseconds) to wait before quitting when sending or receiving data.

Http_Port(<Handle As Long>, <DefaultPort As Long>)

This function sets the default port number to use if none is specified in the URL. The default value is 80.

Http_Close(<Handle As Long>)

This function closes the connection on the specified handle.

Note: It may not always be necessary to close a handle before opening a new connection, if the connection would be made to the same server. For more information see Http_Connect.

HttpSetOutputHeader(<Handle As Long>, <HeaderName As String>, <Value As String>)

This function sets an HTTP header value that will be transmitted along with all subsequent Http_Getxx and Http_Postxx requests. If the value provided is an empty string, the header is removed from the list of transmitted headers. Header fields are similar to environment variables -- they are used to provide additional information and are accessed by name, returning a value.

HttpGetResponseHeader(<Handle As Long>, <HeaderName As String>, <ByRef Value As String>)

This function checks the header fields of the last response and stores the value of the named header into the provided string, it the header value is available. Header fields are similar to environment variables -- they are used to provide additional information and are accessed by name, returning a value.

<ResponseCode> = Http_GetString(<Handle As Long>, <ByRef Response As String>)

This function issues a request to the remote server and stores the response from the server into the provided string. The response code from the request is returned to the user.

<ResponseCode> = Http_GetFile(<Handle As Long>, <FileName As String>)

This function issues a request to the remote server and stores the response from the server into the named file. The response code from the request is returned to the user.

<ResponseCode> = Http_PostFileToFile(<Handle As Long>, <SourceFileName As String>, <DestFileName As String>)

This function posts the contents of the named file to the remote server. The response from the server is stored into the named file, and the response code is returned to the user.

<ResponseCode> = Http_PostFileToStr(<Handle As Long>, <SourceFileName As String>, <ByRef Response As String>)

This function posts the contents of the named file to the remote server. The response from the server is stored into the provided string, and the response code is returned to the user.

<ResponseCode> = Http_PostStrToFile(<Handle As Long>, <SourceMsg As String>, <DestFileName As String>)

This function posts the contents of the provided string to the remote server. The response from the server is stored into the named file, and the response code is returned to the user.

<ResponseCode> = Http_PostStrToStr(<Handle As Long>, <SourceMsg As String>, <ByRef Response As String>)

This function posts the contents of the provided string to the remote server. The response from the server is stored into the provided string, and the response code is returned to the user.

CONDITIONAL SYNTAX

Conditional processing.

```
If <cond> Then <exp> ElseIf  <cond> Then  <exp> Else <exp>
```

```
If <condition> Then
<statements...>
ElseIf <condition> Then
<statements...>
Else <expression>
<statements...>
End If
```

If **Not**(A < B)

B = **Not** A

"Is" conditional operator for comparing object references.

```
Dim A As Object
   Dim B As Object
   Set C = A
      *
      *
      *
   Set A = CreateObject( "Some.Application")
   Set B = CreateObject( "SomeOther.Application")
   Set C = A
      *
      *
      *
   If Not A Is B Then
      WriteConStdOut( "A and B are different" & _CRLF_)
   Else
        WriteConStdOut( "A and B are the same?" & _CRLF_)
   End If
```

"In" conditional.

This conditional returns true if the left value is contained in the set of values to the right of the conditional. The list may contain either variables or constants.

```
B = 0
 If A In ( B, 2, 4, 6, 8, 10) Then
   WriteConStdOut( A & " is an even integer between 0 and 10." & _CRLF_)
 End If
```

Select...Case control structure.

```
Select Case B
    Case A,7,Is > 10
       WriteConStdOut( "It's A or 7 or > 10!!!" & _CRLF_)

    Case Len( Mid$( "Funky", 2, 1))
       WriteConStdOut( "It's 1" & _CRLF_)

    Case Else
       WriteConStdOut( "It's none of the above!" & _CRLF_)

  End Select
```

Conditional Assignment.

```
<variable> =If <cond> Then <exp> ElseIf  <cond> Then <exp> Else <exp>
```

Operator	Value
=	Equal
<>	Not Equal
<	Less than
<=	Less than or Equal
>	Greater than
>=	Greater than or Equal

LIKE CONDITIONAL OPERATOR SYNTAX

Used to compare two strings.

Syntax

result = *string* **Like** *pattern*

The **Like** operator syntax has these parts:

Part	Description
result	Required; any numeric variable.
string	Required; any string expression.
pattern	Required; any string expression conforming to the pattern-matching conventions described in Remarks.

Remarks

If *string* matches *pattern*, *result* is **True**; if there is no match, *result* is **False**.

Results in string comparisons are based on a sort order that is derived from the internal binary representations of the characters.

Built-in pattern matching provides a versatile tool for string comparisons. The pattern-matching features allow you to use wildcard characters, character lists, or character ranges, in any combination, to match strings. The following table shows the characters allowed in *pattern* and what they match:

Characters in *pattern*	Matches in *string*
?	Any single character.
*	Zero or more characters.
#	Any single digit (0-9).
[*charlist*]	Any single character in *charlist*.
[!*charlist*]	Any single character not in *charlist*.

A group of one or more characters (*charlist*) enclosed in brackets (**[ ]**) can be used to match any single character in *string* and can include almost any character, including digits.

Note To match the special characters left bracket (**[**), question mark (**?**), number sign (**#**), and asterisk (*****), enclose them in brackets. The right bracket (**]**) can't be used within a group to match itself, but it can be used outside a group as an individual character.

By using a hyphen (**-**) to separate the upper and lower bounds of the range, *charlist* can specify a range of characters. For example, `[A-Z]` results in a match if the corresponding character position in *string* contains any uppercase letters in the range A-Z. Multiple ranges are included within the brackets without delimiters.

The meaning of a specified range depends on the binary character ordering of the system the code is running on.

Other rules for pattern matching

- An exclamation point (**!**) at the beginning of *charlist* means that a match is made if any character except the characters in *charlist* is found in *string*. When used outside brackets, the exclamation point matches itself.
- A hyphen (**-**) can appear either at the beginning (after an exclamation point if one is used) or at the end of *charlist* to match itself. In any other location, the hyphen is used to identify a range of characters.
- When a range of characters is specified, they must appear in ascending sort order (from lowest to highest). `[A-Z]` is a valid pattern, but `[Z-A]` is not.
- The character sequence `[]` is considered a zero-length string ("").

EXPRESSION SYNTAX

• <variable> = <expression/method function>

Variable assignment.

Operator	Value
=	Assignment operator.
+ – * / % ^	Add, Subtract, Multiply, Divide, Modulus, Exponent operators
( )	Expression grouping
&	Convert to string and concatenate operator
Hexadecimal constants	Count = &H2A + 0x200
Octal constants	Count = &052
Built-in constants for common character sequences	• _CRLF_—Carriage Return/Line Feed • _CR_—Carriage Return • _LF_—Line Feed • _TAB_—Tab • _BS_—Backspace • _QUOTE_—Double Quote • _FF_—Form Feed `WriteConStdOut( _QUOTE_ & "Wow!" & _QUOTE_ & " he exclaimed." & _CRLF_)`

Examples	
x = (y + 3)	x is assigned the value of y + 3
Call s(arg1, arg2)	Subroutine call
z = (f(x) + 3)	A function-call expression
str = “Jim” & (f(x) + 3) & p	str is assigned the result of the string concatenation

• = Assignment

Assigns the value of an expression to a variable.

Syntax:

variable = valueexpression

• + Addition

Sum two numbers or concatenate two strings.

Syntax:

result = operand1 + operand2

- **-** **Subtraction**

Finds the difference between two numbers or to indicate the negative value of an operand.

Syntax:

result = operand1 + operand2

- ***** **Multiplication**

Multiplies two numbers.

Syntax:

*result = operand1 * operand2*

- */* **Division**

Divides two numbers.

Syntax:

result = operand1 / operand2

- **%** **Modulus**

Divides two numbers and returns only the remainder.

Syntax:

result = operand1 % operand2

- **Mod** **Modulus**

Divides two numbers and returns only the remainder.

Syntax:

result = operand1 Mod operand2

- **^** **Exponent**

Used to raise a number to the power of an exponent.

Syntax:

result = number ^ exponent

- **&** **Concatenate**

Forces string concatenation of two operands.

Syntax:

result = operand1 & operand2

FORMATTING SYNTAX

The following table shows the characters you can use to create user-defined numerical formats and the meaning of each:

Character	Meaning
Null string	Display the number with no formatting.
0	Digit placeholder. Display a digit or a zero. If there is a digit in the expression being formatted in the position where the 0 appears in the format string, display it; otherwise, display a zero in that position. If the number being formatted has fewer digits than there are zeros (on either side of the decimal) in the format expression, leading or trailing zeros are displayed. If the number has more digits to the right of the decimal separator than there are zeros to the right of the decimal separator in the format expression, the number is rounded to as many decimal places as there are zeros. If the number has more digits to the left of the decimal separator than there are zeros to the left of the decimal separator in the format expression, the extra digits are displayed without modification.
#	Digit placeholder. Display a digit or nothing. If there is a digit in the expression being formatted in the position where the # appears in the format string, display it; otherwise, display nothing in that position. This symbol works like the 0 digit placeholder, except that leading and trailing zeros aren't displayed if the number has the same or fewer digits than there are # characters on either side of the decimal separator in the format expression.
@	Digit placeholder. Display a digit or a space. If there is a digit in the expression being formatted in the position where the @ appears in the format string, display it; otherwise, display a space in that position. If the number being formatted has fewer digits than there are @ characters (on either side of the decimal) in the format expression, leading or trailing spaces are displayed. If the number has more digits to the right of the decimal separator than there are @ characters to the right of the decimal separator in the format expression, the number is rounded to as many decimal places as there are @ characters. If the number has more digits to the left of the decimal separator than there are @ characters to the left of the decimal separator in the format expression, the extra digits are displayed without modification.
.	Decimal placeholder. The decimal placeholder determines how many digits are displayed to the left and right of the decimal separator. If the format expression contains only number signs to the left of this symbol, numbers smaller than 1 begin with a decimal separator. If you want a leading zero to always be displayed with fractional numbers, use 0 as the first digit placeholder to the left of the decimal separator instead. The actual character used as a decimal placeholder in the formatted output depends on the Number Format specified in the International section of the Microsoft Windows Control Panel. For some countries, a comma is used as the decimal separator.
%	Percentage placeholder. The expression is multiplied by 100. The percent character (%) is inserted in the position where it appears in the format string.
,	Thousand separator. The thousand separator separates thousands from hundreds within a number that has four or more places to the left of the decimal separator. Standard use of the thousand separator is specified if the format contains a comma surrounded by digit placeholders (0 or #). Two adjacent commas or a comma immediately to the left of the decimal separator (whether or not a decimal is specified) means "scale the number by dividing it by 1000, rounding as needed." You can scale large numbers using this technique. For example, you can use the format string "##0,," to represent 100 million as 100. Numbers smaller than 1 million are displayed as 0. Two adjacent commas in any position other than immediately to the left of the decimal separator are treated simply as specifying the use of a thousand separator. The actual character used as the thousand separator in the formatted output depends on the Number Format specified in the International section of the Control Panel. For some countries, a period is used as the thousand separator.
:	Time separator. The time separator separates hours, minutes, and seconds when time values are formatted. The actual character used as the time separator depends on the Time Format specified in the International section of the Control Panel.
/	Date separator. The date separator separates the day, month, and year when date values are formatted. The actual character used as the date separator in the formatted output depends on Date Format specified in the International section of the Control Panel.

Character	Meaning
- + $ ( ) space	Display a literal character. To display a character other than one of those listed, precede it with a backslash (\) or enclose it in double quotation marks (" ").
****	Display the next character in the format string. Many characters in the format expression have a special meaning and can't be displayed as literal characters unless they are preceded by a backslash. The backslash itself isn't displayed. Using a backslash is the same as enclosing the next character in double quotation marks. To display a backslash, use two backslashes (\\). Examples of characters that can't be displayed as literal characters are the date- and time-formatting characters (a, c, d, h, m, n, p, q, s, t, w, y, and /:), the numeric-formatting characters (#, 0, %, E, e, comma, and period), and the string-formatting characters (@, &, <, >, and !).
"ABC"	Display the string inside the double quotation marks. To include a string in *fmt* from within Visual Basic, you must use Chr(34) to enclose the text (34 is the ANSI code for a double quotation mark).

A format expression for numbers can have from one to four sections separated by semicolons.

If you use	The result is
One section only	The format expression applies to all values.
Two sections	The first section applies to positive values and zeros, the second to negative values.
Three sections	The first section applies to positive values, the second to negative values, and the third to zeros.
Four sections	The first section applies to positive values, the second to negative values, the third to zeros, and the fourth to Null values.

The following example has two sections: the first defines the format for positive values and zeros; the second section defines the format for negative values.

"$#,##0;($#,##0)"

If you include semicolons with nothing between them, the missing section is printed using the format of the positive value. For example, the following format displays positive and negative values using the format in the first section and displays "Zero" if the value is zero.

"$#,##0;;\Z\e\r\o"

Some sample format expressions for numbers are shown below. (These examples all assume that Country is set to United States in the International section of the Control Panel.) The first column contains the format strings. The other columns contain the output that results if the formatted data has the value given in the column headings.

Format (*fmt*)	Positive 5	Negative 5	Decimal .5	Null
Null string	5	-5	0.5	
0	5	-5	1	
0.00	5.00	-5.00	0.50	

#,##0	5	-5	1	
#,##0.00;;;Nil	5.00	-5.00	0.50	Nil
$#,##0;($#,##0)	$5	($5)	$1	
$#,##0.00;($#,##0.00)	$5.00	($5.00)	$0.50	
0%	500%	-500%	50%	
0.00%	500.00%	-500.00%	50.00%	
0.00E+00	5.00E+00	-5.00E+00	5.00E-01	
0.00E-00	5.00E00	-5.00E00	5.00E-01	

The following table shows the characters you can use to create user-defined date/ time formats and the meaning of each:

Character	Meaning
c	Display the date as ddddd and display the time as ttttt, in that order. Only date information is displayed if there is no fractional part to the date serial number; only time information is displayed if there is no integer portion.
d	Display the day as a number without a leading zero (1-31).
dd	Display the day as a number with a leading zero (01-31).
ddd	Display the day as an abbreviation (Sun-Sat).
dddd	Display the day as a full name (Sunday-Saturday).
ddddd	Display a date serial number as a complete date (including day, month, and year) formatted according to the Short Date setting in the International section of the Windows Control Panel. The default Short Date format is m/d/yy.
dddddd	Display a date serial number as a complete date (including day, month, and year) formatted according to the Long Date setting in the International section of the Control Panel. The default Long Date format is mmmm dd, yyyy.
w	Display the day of the week as a number (1 for Sunday through 7 for Saturday.)
ww	Display the week of the year as a number (1-53).
m	Display the month as a number without a leading zero (1-12). If m immediately follows h or hh, the minute rather than the month is displayed.
mm	Display the month as a number with a leading zero (01-12). If m immediately follows h or hh, the minute rather than the month is displayed.
mmm	Display the month as an abbreviation (Jan-Dec).
mmmm	Display the month as a full month name (January-December).
q	Display the quarter of the year as a number (1-4).
y	Display the day of the year as a number (1-366).
yy	Display the year as a two-digit number (00-99).
yyyy	Display the year as a four-digit number (100-9999).
h	Display the hour as a number without leading zeros (0-23).
hh	Display the hour as a number with leading zeros (00-23).

Character	Meaning
n **nn**	Display the minute as a number without leading zeros (0-59). Display the minute as a number with leading zeros (00-59).
s **ss**	Display the second as a number without leading zeros (0-59). Display the second as a number with leading zeros (00-59).
ttttt	Display a time serial number as a complete time (including hour, minute, and second) formatted using the time separator defined by the Time Format in the International section of the Control Panel. A leading zero is displayed if the Leading Zero option is selected and the time is before 10:00 A.M. or P.M. The default time format is h:mm:ss.
9[99999999]	Display one to nine positions of milliseconds.
AM/PM	Use the 12-hour clock and display an uppercase AM with any hour before noon; display an uppercase PM with any hour between noon and 11:59 PM.
am/pm	Use the 12-hour clock and display a lowercase AM with any hour before noon; display a lowercase PM with any hour between noon and 11:59 PM.
A/P	Use the 12-hour clock and display an uppercase A with any hour before noon; display an uppercase P with any hour between noon and 11:59 PM.
a/p	Use the 12-hour clock and display a lowercase A with any hour before noon; display a lowercase P with any hour between noon and 11:59 PM.
AMPM	Use the 12-hour clock and display the contents of the 1159 string (s1159) in the WIN.INI file with any hour before noon; display the contents of the 2359 string (s2359) with any hour between noon and 11:59 PM. AMPM can be either uppercase or lowercase, but the case of the string displayed matches the string as it exists in the WIN.INI file. The default format is AM/PM.

The following are examples of user-defined date and time formats:

Format	Display
m/d/yy	12/7/58
d-mmmm-yy	7-December-58
d-mmmm	7 December
mmmm-yy	December 58
hh:mm AM/PM	08:50 PM
h:mm:ss a/p	8:50:35 p
h:mm	20:50
h:mm:ss	20:50:35
m/d/yy h:mm	12/7/58 20:50

The following table shows the characters you can use to create user-defined string formats and the meaning of each:

Character	Character Meaning
@	Character placeholder. Display a character or a space. If there is a character in the string being formatted in the position where the @ appears in the format string, display it; otherwise, display a space in that position. Placeholders are filled from right to left unless there is an ! character in the format string. See below.
&	Character placeholder. Display a character or a space. If there is a character in the string being formatted in the position where the @ appears in the format string, display it; otherwise, display a space in that position. Placeholders are filled from right to left unless there is an ! character in the format string. See below.
<	Force lowercase. All characters are displayed in lowercase format.
>	Force uppercase. All characters are displayed in uppercase format.
!	Force placeholders to fill from left to right instead of right to left.

FORMAT FUNCTIONS

Transall supports special formatting tokens that act as format functions. The syntax rules for these format functions are as follows; To use the functions, a function identifier must be the first token on the format string. Format functions can be combined and the results will reflect the application of the combined format functions in the order they were listed in the format string. Format functions can not be combined with standard format masks. The two are mutually exclusive.

Format function details:

Formatting Function	Result
"* alphabetic"	Displays a number as a lowercase alphabetic character. Format$(2, "* alphabetic") will result in "b".
"* ALPHABETIC"	Displays a number as an uppercase alphabetic character. Format$(2, "* ALPHABETIC") will result in "B".
"* Caps"	Capitalizes the first letter of each word. Format$("julie tanner", "* Caps") will result in "Julie Tanner".
"* CardText"	Displays a number in cardinal text. Format$(1234.56, "* CardText") will result in "one thousand two hundred thirty five". Note that floating point numbers get rounded and processed as integer numbers.
"$#,##0.00;($#,##0.00)"	Displays a number with a currency symbol. Format$(1234.56, "$#,##0.00;($#,##0.00))" will result in either $1,234.56 or, in the case of a negative number, ($1,234.56).
"#,##0.00;(#,##0.00)"	Displays a number with a specified number of decimal places. Format$(1234.56, "#,##0.00;(#,##0.00))" will result in either 1,234.56 or, in the case of a negative number, (1,234.56).
"* DollarText"	Displays a number in dollar text. Format$(1234.56,"* DollarText") will result in "one thousand two hundred thirty four and 56/100".
"* FirstCap"	Capitalizes the first letter of the first word. Format$("weekly report on sales.", "* FirstCap") will result in "Weekly report on sales".
"* Hex"	Displays a number in hexadecimal format. Format$(458, "* Hex") will result in "1CA".

Formatting Function	Result
"* Lower"	Makes all letters lowercase. Format$("DOCUMENTATION IS FUN","* Lower") will result in "documentation is fun".
"#,##0;(#,##0)"	Displays a number as an integer. Format$(1234.56, "#,##0;(#,##0))" will result in either 1,235 or, in the case of a negative number, (1,235). Note that floating point numbers get rounded and processed as integer numbers.
"* Ordinal"	Displays a number in ordinal Arabic text. Format$(30, "* Ordinal") will result in "30th".
"* OrdText"	Displays a number in ordinal text. Format$(1234.56,"* OrdText") will result in "one thousand two hundred thirty-fifth". Note that floating point numbers get rounded and processed as integer numbers.
"* Repeat *c*"	*Use this function for publishing footnotes denoted with a symbol instead of a number.* Format$(DCP.FOOTNOTE.NUMBER, "* Repeat *") will result in footnote symbols of *, **, ***, and so on, as the tag's value increases.
"* roman"	Displays a number in lowercase Roman text. Format$(1234.56,"* roman") will result in "mccxxxv". Note that floating point numbers get rounded and processed as integer numbers.
"* ROMAN"	Displays a number in uppercase Roman text. Format$(1234.56,"* ROMAN") will result in "MCCXXXV". Note that floating point numbers get rounded and processed as integer numbers.
"* SelectChar *cccc*"	*Use this function for publishing footnotes denoted with a symbol instead of a number, where the numeric value being formatted acts as an index to select one of the characters.* Format$(DCP.FOOTNOTE.NUMBER, "\SelectChar @#$%") will result in footnote symbols of @, #, $, and % as the tag's value increases.
"* SYMBOL"	Displays a single character, in either the ASCII character set on WIN32 and UNIX. Format$(130, "*SYMBOL") will result in "é". Note that tloating point numbers are rounded and processed as integer numbers. Also, hexadecimal values are supported with the format "0x*n*" or "0X*n*", where the hexadecimal number *n* is preceded by "0x" or "0X" (a zero followed by the letter "x" or "X").
"* Trim"	Trims spaces from both right and left of a string. Format$(" This is a test ","* Trim") will result in "This is a test".
"* LTrim"	Trims spaces from the left of a string. Format$(" This is a test ","* LTrim") will result in "This is a test ".
"* RTrim"	Trims spaces from the right of a string. Format$(" This is a test ","* RTrim") will result in " This is a test".
"* Upper"	Capitalizes all letters. Format$("docuflex rules!","* Upper") will result in "DOCUFLEX RULES!".

Format functions can be combined:

Formatting Function	Result
"* Ordinal * Upper"	Displays a number in upper case ordinal arabic text. For example, Format$(30, "* Ordinal * Upper") will result in "30TH".

SENDING SMTP EMAIL MESSAGES

Transall supports sending email in SMTP format. Transall supports two different mechanisms for sending mail. One is handy when sending just a few messages, while the other is more efficient when a larger number of items need to be sent. For each method you need to have the following information:

1. The SMTP address of your email server. This is frequently in the form "smtp.yourdomain.com".
2. A valid email address for connecting to the server.
3. Optionally a port number for connecting to the server. (Most non-encrypted SMTP servers listen to port 25, which is the default.)

When sending no more than a few messages, you may choose to use the single call method. These routines allow you to transfer the contents of a file or string as a text message or an HTML message, or to send a file as a text message with the file attached. These calls are listed below:

```
MailSendFile( < SendTo>, <UserID>, < SMTPServer>, <Subject>,  <FileName>[[, <MessageType>][, <HideRecipients>]])
MailSendMsg( <SendTo>, <UserID>, <SMTPServer>, <Subject>,  <MessageText>[[, <MessageType>,][ <HideRecipients>]])
MailSendFileAsAttachment( <SendTo>, <UserID>, <SMTPServer>, <Subject>,  <Message>, <FileName>[, <HideRecipients>])
```

Each of these routines, when called, will connect to the server, send the file in the manner requested and disconnect from the server. Each returns a non-zero value if the message was successfully transferred to the server. Each routine accepts an optional value "HideRecipients" which may be set to a non-zero value if the resulting email should not contain the addresses of each person receiving the mail. This may be desirable when transferring a file to a group of individuals who do not want their email addresses made public.

If you are transferring many different files, then it may be advantageous to use the second method for sending messages. In this method, a single connection to the email server is made and kept open until all files have been delivered to the SMTP server. Once a message has been transferred to and accepted by the server, it is the responsibility of the server to deliver the message, even if the connection should be dropped before it is closed by the server. When using this method, it is also possible (if desired) to name the sender of a message as being different from the "UserID" and to change the sender of the mail between messages. The multiple call functions are listed below:

```
MailConnect( <UserID>, <SMTPServer>[, <port>])
MailNewSender( <handle>, <Sender>)
MailFile( <handle>, <SendTo>, <Subject>, <FileName>[[, <MessageType>][, <HideRecipients>]])
MailMsg( <handle>, <SendTo>, <Subject>, <Message>[[, <MessageType>][, <HideRecipients>]])
MailFileAsAttachment( <handle>, <SendTo>, <Subject>, <Message>, <FileName>[, <HideRecipients>])
MailDisconnect( <handle>)
```

It is possible to send multiple emails between issuing the MailConnect() and MailDisconnect(), and the name of the sender can optionally be set or changed at any time by calling the MailNewSender function. Once the sender name is set, it remains in effect for that mailer until it is changed again. By default, the sender is the same as the UserID value. Each function returns a non-zero value if it was carried out successfully. Whenever the MailDisconnect function is called, no more mail can be sent on that handle until a new MailConnect command is issued to reconnect to the server.

In addition to the functions listed above, there is one other function, **MailGetError**([<handle]), that may be called without a handle to get the last error which occurred using a single call function, or with a handle to get the last error which occurred on that handle.

The following examples show the mail functions in use. The first three examples use the single call method to send a file as a text message, send a file attached to a message, and send the contents of a string as an HTML message.

Example 1: Single Call Mail Functions

```
|-------------------------------------------------------------
|  This example sends a text message from a file
|-------------------------------------------------------------
Public Sub singleCallMailText( SendTo As String, UserID As String, SMTPServer As String)

   If (MailSendFile( SendTo, UserID, SMTPServer, "File as text", "file.txt") = 0) Then
      WriteConStdOut( "Mailing msg:  " & MailGetError() & _CRLF_)
   End If

End Sub
```

Example 2: Single Call Mail Functions

```
|-------------------------------------------------------------
|  This example sends a file as an attachement in a message
|-------------------------------------------------------------
Public Sub singleCallMailAtch( SendTo As String, UserID As String, SMTPServer As String)

   If (MailSendFileAsAttachment( SendTo, UserID, SMTPServer, "Attachment", _
         "See attachment", "File.doc") = 0) Then
      WriteConStdOut( "Mailing msg:  " & MailGetError() & _CRLF_)
   End If

End Sub
```

Example 3: Single Call Mail Functions

```
|-----------------------------------------------------------------
|  This example sends an HTML message
|-----------------------------------------------------------------
Public Sub singleCallMailHtml( SendTo As String, UserID As String, SMTPServer As String)

   If (MailSendMsg( SendTo, UserID, SMTPServer, "Msg as HTML", _
                           "<em>Italic</em> <b>Bold</b>", TRAN_MAIL_HTML) = 0) Then
      WriteConStdOut( "Mailing msg:  " & MailGetError() & _CRLF_)
   End If

End Sub
```

This next example carries out a similar task, but does so using the multiple call method. The routine below sends the contents of a file as an HTML message, sends a file attached to a message, and sends the contents of a string as a text message.

Example 4: Single Call Mail Functions

```
|-------------------------------------------------------------------------
|  This example sends an HTML message, an attachment and a text message
|-------------------------------------------------------------------------
Public Sub multipleCallMail( SendTo As String, UserID As String, SMTPServer As String)
   Dim RC As Long
   Dim hMail As Long

   hMail = MailConnect( UserID, SMTPServer)

   If ( hMail <> 0) Then
      RC = MailFile( hMail, SendTo, "File as HTML", "file.htm", TRAN_MAIL_HTML)

      If ( RC <> 0) Then
         RC = MailFileAsAttachment( hMail, SendTo, "Attachment", _
                                    "See attachment", "File.doc")

         If ( RC <> 0) Then
            RC = MailMsg(hMail, SendTo, "Text Msg", "<em>Italic</em> <b>Bold</b>")
         End If
      End If

      If ( RC = 0) Then
         WriteConStdOut( "Mailing error!  " & MailGetError( hMail) & _CRLF_)
      End If

      MailDisconnect( hMail)
   Else
      WriteConStdOut( "Mailing error!  " & MailGetError() & _CRLF_)
   End If

End Sub
```

SMTP EMAIL FUNCTIONS

You can use these functions to send SMTP email messages:

<Boolean integer> = MailSendFile(<String SendTo>, <String UserID>, <String SMTPServer>, <String Subject>, <String FileName> [[, <MessageType>][, <Long HideRecipients>])

This function is used to send the contents of a file as a text message. The file is not reformatted in any manner. This function attempts to log onto the named server using the given user ID, send the text file as a text message with the indicated subject to the recipient(s) and disconnect from the server. If successful, a non-zero value is returned. The various fields are described below.

Field	Description
SendTo	This is list of one or more recipients, with each email address separated by a comma (“,”)
UserID	This is a valid email address for sending email from the SMTP server.
SMTPServer	This is the name of the SMTP server, for example “smtp.ourserver.com”.
Subject	This is the entry for the “Subject” field of the email.
FileName	This is the name of the file to be transmitted.
MessageType	This indicates how the message should be displayed when it is read. Valid types are TRAN_MAIL_TEXT and TRAN_MAIL_HTML. This defaults to TRAN_MAIL_TEXT.
HideRecipients	This is an optional value. If you wish to hide the email addresses of the recipients of the mail in the message that is sent, then set this field to a non-zero value. By default, recipient addresses are displayed.

<Boolean integer> = MailSendMsg(<String SendTo>, <String UserID>, <String SMTPServer>, <String Subject>, <String MessageText> [[, <MessageType>][, <Long HideRecipients>]])

This function is used to send the contents of a file as an HTML message. The file is not reformatted in any manner. This function attempts to log onto the named server using the given user ID, send the file as an HTML message with the indicated subject to the recipient(s) and disconnect from the server. If successful, a non-zero value is returned. The various fields are described below.

Field	Description
SendTo	This is list of one or more recipients, with each email address separated by a comma (“,”)
UserID	This is a valid email address for sending email from the SMTP server.
SMTPServer	This is the name of the SMTP server, for example “smtp.ourserver.com”.
Subject	This is the entry for the “Subject” field of the email.
MessageText	This is the text of the message to be sent. It may consist of one or more lines separated by _CRLF_ and may be either simple text or HTML.
MessageType	This indicates how the message should be displayed when it is read. Valid types are TRAN_MAIL_TEXT and TRAN_MAIL_HTML. This defaults to TRAN_MAIL_TEXT.
HideRecipients	This is an optional value. If you wish to hide the email addresses of the recipients of the mail in the message that is sent, then set this field to a non-zero value. By default, recipient addresses are displayed.

<Boolean integer> = MailSendFileAsAttachment(<String SendTo>, <String UserID>, <String SMTPServer>, <String Subject>, <String Message>, <String FileName>[, <Long HideRecipients>])

This function is used to send a file as an attachment to a text message. The file is not reformatted in any manner. This function attempts to log onto the named server using the given user ID, send the file as an attachment to a text message with the indicated subject to the recipient(s) and disconnect from the server. If successful, a non-zero value is returned. The various fields are described below.

Field	Description
SendTo	This is list of one or more recipients, with each email address separated by a comma (“,”)
UserID	This is a valid email address for sending email from the SMTP server.
SMTPServer	This is the name of the SMTP server, for example “smtp.ourserver.com”.
Subject	This is the entry for the “Subject” field of the email.
FileName	This is the name of the file to be transmitted.
Message	This is a text message to include when a file is transferred as an attachment. It may consist of only one line, or of many lines separated by _CRLF_.
HideRecipients	This is an optional value. If you wish to hide the email addresses of the recipients of the mail in the message that is sent, then set this field to a non-zero value. By default, recipient addresses are displayed.

<Long Handle> = MailConnect(<String UserID>, <String SMTPServer>[, <Long port>])

This function is used to start a new mail session with the SMTP server. This call identifies the server to connect to, the userid to be used for sending mail, and optionally a port number to connect to on the server (the default value is 25). If successful, a non-zero value is returned. This call must be successfully made before any mail can be sent to the server using the multiple call family of functions. The various fields are described below.

Field	Description
UserID	This is a valid email address for sending email from the SMTP server.
SMTPServer	This is the name of the SMTP server, for example “smtp.ourserver.com”.
Port	This is an optional value, used to indicate the port on the SMTP server to connect to. This defaults to 25.

MailNewSender(<Long Handle>, <String Sender>)

This function sets the value to use as the sender of subsequent email messages. This function is not required. If it is not used, the value of the “UserID” field of the MailConnect function will be used as the sender. Once this call has been made, the value provided will be used as the sender of all subsequent messages until either MailNewSender is called again, or the MailDisconnect function is called.

Field	Description
Handle	This is the handle returned by a prior call to MailConnect.

Field	Description
Sender	This is the value to display as the sender of the message. It is not required to be a valid email address, but some mail filters may flag it as SPAM if it is not.

<Boolean integer> = MailFile(<Long Handle>, <String SendTo>, <String Subject>, <String FileName>[[, <MessageType>][, <Long HideRecipients>]])

This function is used to send the contents of a file as a text message. The file is not reformatted in any manner. This function attempts to send the text file as a text message with the indicated subject to the recipient(s). If successful, a non-zero value is returned and the mail is scheduled to be delivered by the server when the MailConnect command is issued, or if the connection is broken. The various fields are described below.

Field	Description
Handle	This is the handle returned by a prior call to MailConnect.
SendTo	This is list of one or more recipients, with each email address separated by a comma (“,”)
Subject	This is the entry for the “Subject” field of the email.
FileName	This is the name of the file to be transmitted.
MessageType	This indicates how the message should be displayed when it is read. Valid types are TRAN_MAIL_TEXT and TRAN_MAIL_HTML. This defaults to TRAN_MAIL_TEXT.
HideRecipients	This is an optional value. If you wish to hide the email addresses of the recipients of the mail in the message that is sent, then set this field to a non-zero value. By default, recipient addresses are displayed.

<Boolean integer> = MailMsg(<Long Handle>, <String SendTo>, <String Subject>, <String MessageText>[[, <MessageType>][, <Long HideRecipients>]])

This function is used to send the contents of a string as a message. The message is not reformatted in any manner. This function attempts to send the message with the indicated subject to the recipient(s). If successful, a non-zero value is returned and the mail is scheduled to be delivered by the server when the MailDisconnect command is issued, or if the connection is broken. The various fields are described below.

Field	Description
Handle	This is the handle returned by a prior call to MailConnect.

Field	Description
SendTo	This is list of one or more recipients, with each email address separated by a comma (“,”)
Subject	This is the entry for the “Subject” field of the email.
FileName	This is the name of the file to be transmitted.
MessageText	This is the text of the message to be sent. It may consist of one or more lines separated by _CRLF_ and may be either simple text or HTML.
MessageType	This indicates how the message should be displayed when it is read. Valid types are TRAN_MAIL_TEXT and TRAN_MAIL_HTML. This defaults to TRAN_MAIL_TEXT.
HideRecipients	This is an optional value. If you wish to hide the email addresses of the recipients of the mail in the message that is sent, then set this field to a non-zero value. By default, recipient addresses are displayed.

<Boolean integer> = MailFileAsAttachment(<Long Handle>, <String SendTo>, <String Subject>, <String Message>, <String FileName>[, <Long HideRecipients>])

This function is used to send the contents of a file as a file attachment to a text message with the indicated subject to the recipient(s). The file is not reformatted in any manner. If successful, a non-zero value is returned and the mail is scheduled to be delivered by the server when the MailDisconnect command is issued, or if the connection is broken. The various fields are described below.

Field	Description
Handle	This is the handle returned by a prior call to MailConnect.
SendTo	This is list of one or more recipients, with each email address separated by a comma (“,”)
Subject	This is the entry for the “Subject” field of the email.
Message	This is a text message to include when a file is transferred as an attachment. It may consist of only one line, or of many lines separated by _CRLF_.
FileName	This is the name of the file to be transmitted.
HideRecipients	This is an optional value. If you wish to hide the email addresses of the recipients of the mail in the message that is sent, then set this field to a non-zero value. By default, recipient addresses are displayed.

<Boolean integer> = MailDisconnect(<Long Handle>)

This function tells the SMTP server to deliver all messages which were previously transferred by one of the multiple call functions and then disconnects from the server. Once this call has been made, no further emails can be sent on this handle until another call to MailConnect has been issued. If successful, a non-zero value is returned. The various fields are described below.

Field	Description
Handle	This is the handle returned by a prior call to MailConnect.

<Message> = MailGetError([<Long Handle>])

This function returns the text of the last error message on the given handle. If no handle is provided, it returns the text of the last error message for one of the single line mail functions.

Field	Description
Handle	This is the handle returned by a prior call to MailConnect.

RECAP LOG FILE

The Recap Log File is a new logging feature added to Transall to recap a Transall project that was run.

Below is an example of a Transall project sample log:

```
--------------------------------------------------------------------------------
Transall
Script Module: CreateAce

Transaction Error Report - System timestamp: Thu Feb 19 10:16:32 2004
Some Transaction Name: Warning!
Some Transaction Name: Error!
Electric Meter 77634: W: No reading.
Electric Meter 77634: E: Negative reading value.
==> Trans    read:    2
==> Trans written:    2
==> Warning count:    2
==> Error   count:    2
End of Transaction Error Report - System timestamp: Thu Feb 19 10:16:32 2004
Elapsed Time: 0 seconds
--------------------------------------------------------------------------------
```

This log was produced by this script:

```
<Start>
TransLogOpen("C:\test.log", "Script Module: CreateAce")
TransLogTransRead("Some Transaction Name:")
TransLogWarning("Warning!")
TransLogError("Error!")
TransLogTransWrite()
TransLogTransRead("Electric Meter 77634:")
TransLogWarning("W: No reading.")
TransLogError("E: Negative reading value.")
TransLogTransWrite()
TransLogClose()
<End>.
```

Note You should add a call to "TransLogOpen" at the start of your Transall project.

Add a call to "TransLogTransRead" so it gets called before each transaction gets written, and add a call to "TransLogTransWrite" so it gets called after each transaction gets written. These calls update the transaction counters for transactions read and written. You should be sure to pass something that logically identifies a transaction to the "TransLogTransRead" call, like an Account Number or a Client Name.

You should also add calls to "TransLogWarning" or "TransLogError" so they can report any problems that occurred during the Transall run in the log.

Finally, you need to make a call to "TransLogClose" at the end of your Transall project to write final transaction count numbers to the log and close the log file.

LIMITATIONS

- When operating Transall on Sun Solaris, you can't create a VRF for use with Docuflex.

EC REGULATION 1103/97

Transall now includes enhanced script support for three internal functions that comply with Articles 4 and 5 of EC Regulation 1103/97, which govern all manner of currency conversion.

- **CvtCurrencyToEuro**(*amount, exchange_rate)*
- **CvtCurrencyFromEuro**(*amount, exchange_rate, digits_on_right_of_decimal*)
- **CvtCurrencyToCurrency**(*amount, exchange_rate_for_amount, rate_for_new_currency, digits_on_right_of_decimal*)

These functions are included for clients who either are based in Europe or have financial dealings with Europe, and are involved in the following possible scenarios:

- You price goods and services according to your national currency, while your customer's Member State uses the Euro.
- Your Member State uses the Euro, while your customer uses his national currency.
- Both you and your customer use your own national currency.

INDEX

A
About Docuflex Studio command 114
Abs 411
absolute value 411
ActiveX automation technology 34
Add Source 83
adding Record subcomponents 124
adjusting control bars and windows 360
AIX
    installation 26
    setting up the environment 26
API (Application Program Interface) 197
Arctangent 411
Asc 411
ASCII 411
Assistants in the workspace 46
Atn 411
Authentication Mode 398
Auto Formatting command 343

B
Beep 411
Big Endian 426
breakpoints
    discussed 322
    setting 322
build settings, editing 306
building a Transall Application for release 326
building vs. compiling 305
built-in component methods
    discussed 49
    in Component Inspector Events tab 48
    list of 346

C
Call 411, 464
Call Stack window 79
CAny 411
Cascade command 113
CDate 411
CDbl 411
ChDir 411
ChDrive 411
Choose 411

Chr$ 411
CInt 412
Class 411, 434
ClassRefClass {GUID} 411
Clear All Breakpoints command 105
Clipboard
CLng 411
CLngLng 412
Close 412
Close (File) command 60
Close (Window) command 113
Close All command 113
CloseDocumentSet 412
Closing a project 60
CNum 412
COBOL copybooks, populating a COBOL Source component 126
coding a Script Declarations section 341
Compile command 107, 108
Compile Errors command 78
Compile panel
compiling a Transall Application for debugging 311
compiling vs. building 305
Component
Component Explorer 72
Component Explorer command 72
Component Explorer control bar 46
Component Inspector 73
Component Inspector control bar
Component Properties command 73
components
condition expressions 293
Condition instruction 293
Conditional syntax 461
control bars, discussed 45
ControlBreak instruction
control-break processing
    break fields processing
        discussed 300
        example of 302
    discussed 300
    Identifier field processing
        discussed 300
        example of 301
    requires Walk instruction 300
    using Identifier field of Record subcomponent 118
Convert 464

Copy command 65
Cos 412
Cosine 412
CreateObject 412
creating
    Destination components 120
    Logic Tree components 289
    Map components 272
    Script Module components 340
    Source components 120
    Transall projects 44
Creating a project 58
CurDir 412
Current directory 411
current drive 411, 412
Current Path 412
Cut command 64
CvtCurrencyFromEuro 412
CvtCurrencyToCurrency 412
CvtCurrencyToEuro 412

D

data records 117
Database, working with the Transall 279
Datatype
Date 413, 468
Date/Time 415
DateAdd 413
DateDiff 414
DateSerial 415
Day 413, 414, 415
Debug
Debug menu
    Clear All Breakpoints 105
    contents 97, 103
    Demand Break 104
    External Runtime Error 106
    Go 103
    Run to Cursor 105
    Start 103
    Step Into 104
    Step Over 104
    Stop 103
    Toggle Breakpoint 105
Debug toolbar 81
debugging

debugging and deploying Transall Applications
Declarations section
    coding 341
    in Script Modules 339
Declare Function 415
Declare Sub 416
Deebug menu
    Break 104
Define Set 417
Define Table 418
DefineEvent 418
Delete command 66
DeleteAllRows 418
DeleteRow 418
DeleteSetting 418
Deleting
    a component 66
deleting
    Record subcomponents 125
    script from a Script Module 344
Demand Break command 104
Destinations
    creating 120
    discussed 115
    exporting text files 115
    file-based 116
    ODBC-based 173
    scripted data 221
    SQL properties reference 158
    traditional files properties reference 143
DF$GetFirstRow 418
DF$GetLastRow 418
DF$GetNextRow 418
DF$GetPathmapValue 419
DF$GetPriorRow 419
DF$GetRowCount 419
DF$GetRowNumber 419
DF$GetTagValue 419
DF$IsRowFirst 419
DF$IsRowLast 419
DF$IsRowMiddle 419
DF$IsTableEmpty 419
DF$SetCurrentRow 419
DF$SetTagValue 419
DF$SortTable 419
DF$Walk 419

Dictionaries
displaying 109, 110, 111
Dim 419
Dir$ 420
Displaying
DMG Report 100
Do 424
Do Walk 424
Do While 424
docking control bars and toolbars 51
Docuflex Destination
data movement 191
Data Schema 190
DataSets 191
discussed 189
Docuflex Assistant 189
ROOT record node 190
setting up 189
Docuflex Studio Help command 114
Documaker Forms Publishing (FP) 197
Documaker FP Plus (VRF) destination 95
Documanage
sharing projects 62, 352
Document menu
contents 83, 89
DoWhile instruction 295
DynmGetTableValue 425

E
EBCDIC 426
EC Regulation 1103/97 481
Edit menu
editing
EDL
Electronic Document Library 208
setting the effective date for forms 453
Else
conditional assignment 462
conditional processing 461
ElseIf
conditional assignment 462
conditional processing 461
email messages 474
End Function 425
End SendEvent 425
Enum 425

Environ 425
Environ$ 425
Err 425
Err.Description 425
Error code As Integer 453
Even 426
Event-based XML Source
    how parsing works 237
    overview 237
    setting up 237
Events tab (Component Inspector) 48
examples
Execute instruction 296
executing
    Logic Tree components 288
    Map components 278
Exit command 63
Exit Do 426
Exit Function 426
Exit Sub 426
Exiting the Editor 63
Exp 426
exporting text files 115
Expression Builder dialog 276
expression syntax 464
Extraneous Properties 69

F

fields, subcomponents of records 117
file attribute 420
file exporting 115
File Menu
    Share 62
File menu
file, Recap Log 480
FileBitsAreAscii 426
FileBitsAreDefault 426
FileBitsAreEbcdic 426
FileConcat 426
FileCopy 426
files produced from building or compiling 306
FileValuesAreBigEndian 426
FileValuesAreDefaultEndian 426
FileValuesAreLittleEndian 426
filter criteria in SQL tables of Query subcomponents 168
Find command 66

Find control bar 344
FindRowSet 426
FindRowTbl 426
Fix 426
Flat File data source 91
floating control bars and toolbars 51
For To 427
Form Editor dialog 212
Form Select dialog
format options
Format$ 427
formatting syntax
Forms Publication Plus (Fp Plus) 197
FP Plus Destination
    FpAddTag 199
    FpComment 200
    FpDataGroup 200
    FpDataHeader 200
    FpForm 199
    FpHeader and FpFooter 200
    FpKeepOnSamePage 200
    FpLayout 199
    FpPageBreak 200
  destination properties 202
  Documaker FP File 197
  Documaker FP Plus 197
  Logic Tree
    business rules 199
    data extract 199
  overview 197
  record hierarchy feature 198
  record properties 204
  special Transall LogicTree 199
Fp$PageDotCount 427
Fp$PageNumber 428
Fp$PlusGetRfrmtHandle 428
Fp$SectionPageNumber 428
Fp$TotalPages 428
Fp$TotalSectionPages 428
FreeFile 428
Function 428
functions, in Scripts 339

G

Gallery view (PG Chart Types)
General panel

discussed 308
displaying 308
General view (PG Chart Options)
discussed 97, 98
GetColSumSet 429
GetColSumTbl 429
GetObject 429
GetRow Set 431
GetRow Table 431
GetRowCountSet 431
GetRowCountTbl 431
GetRowNumberSet 431
GetRowNumberTbl 431
GetSetting 431
GetVal 431
Go 103
GoSub 432
GoTo 431

H
Help menu
About Docuflex Studio 114
Docuflex Studio Help 114
Hex$ 432
Hour 413, 414, 432

I
Identifier fields of Record subcomponent 118
If 461
IidGuid 432
importing COBOL copybooks into COBOL Source components 126
Input instruction 296
InsertRow
Global 432
Local 432
Installing Transall
UNIX
copying files 26
running the install script 27
setup 26
WIN32
discussed 21
steps for installing 21
Instr 433
InstrRev 433
instructions, next sibling instruction in LogicTree 291

Int 426, 433
Interface Global Unique ID (GUID) 432
IsDate 433
IsNull 433
IsNumeric 433
IsSetCurrent 433
IsSetEmpty 433
IsTableCurrent 433
IsTableEmpty 433

J

Java Database Connectivity (JDBC)
    Details for accessing databases 184
    integrating access to data 35
JDBC *see* Java Database Connectivity (JDBC)
Join 433
join criteria in SQL tables of Query subcomponents 166

K

Kill 433

L

LCase$ 434
Left$ 434
Len 434, 438, 439
length of string 434
Let 434
LibGUID 434
Like conditional operator syntax 462
Links
    command 97
Linux
    installation 26
    setting up the environment 26
Little Endian 426
Locale panel (Document>Settings)
Log 434
Logarithm 434
Logic Bar 76
Logic Tree components
Long Long 412
LookupValSet 434
LookupValTbl 434
Lowercase 434, 469
LTrim$ 434

M
managing Transall Applications
Map instruction in Logic Tree 296
mapping data types between ODBC source and Transall fields 187
Maps
    Map Assistant
        Expression Builder 276
        Resource Bar 275
    Map components
        creating 272
        explained 269
        for each Record or Query in Destination 269
        Map Assistant 273
        performing 278
menu bar 45
Method 446
Mid$ 434
Millisecond 435
Minute 413, 414, 435
MkDir 435
Month 413, 414, 435

N
Name As 435
naming a project 44
New command 58
next sibling instruction 291
noncomponent Tables 341
Now 435
NULL 435

O
Oct 435
ODBC *see* Open Database Connectivity (ODBC)
OdbcCommit 435
OdbcConnect 435
OdbcConnectCsr 435
OdbcDiscconect 435
OdbcDriverConnect 435
OdbcDriverConnectCsr 435
OdbcDriverConnectPrompt 435
OdbcDriverConnectPromptCsr 436
OdbcExecute 436
OdbcExecuteDirect 436
OdbcExecuteDynam 436
OdbcFetch 436

OdbcGetConnectionOptionLong 436
OdbcGetConnectionOptionString 436
OdbcGetErrorInfo 436
OdbcGetSqlRowCount 436
OdbcGetStmtOptionLong 436
OdbcGetStmtOptionString 436
OdbcMoreResults 436
OdbcPrepare 436
OdbcPrepareDynam 436
OdbcRollback 436
OdbcRun 436
OdbcSetConnectionOptionLong 437
OdbcSetConnectionOptionString 437
OdbcSetConOptAutoCommit 437
OdbcSetConOptIsoReadCommited 437
OdbcSetConOptIsoReadRepeatable 437
OdbcSetConOptIsoReadUncommited 437
OdbcSetConOptIsoSerializable 437
OdbcSetConOptReadOnly 437
OdbcSetConOptTrace 437
OdbcSetConOptTraceFile 437
OdbcSetPrepareOpt 437
OdbcSetPrepareOptGlobal 437
OdbcSetStmtOptionLong 437
OdbcSetStmtOptionString 437
OdbcStatementCloseResult 438
OdbcStatementDrop 438
Odd 438
On Error
    GoTo 438
    Goto 0 438
    Resume Next 438
Open 428, 438
Open Database Connectivity (ODBC)
    integrating access to data 35
    mapping data types to Transall fields 187
Opening an existing project 59
Output Bar 75
Output instruction 297
Overflow Definitions
    adding 84

P

Paste command 65
performing Map components 278, 296
PG charts

Chart Options dialog
General view 97, 98
Chart Types dialog
Gallery view 88, 101
PPS 440
PpsExpReadNextHdr$ 439
PpsExpReadVarData 440
production definition 453
Project
Project menu
Project Settings
Project Settings command 307
Project Settings dialog
Debug settings 313
Project Sharing 351
project sharing
projects
Properties tab
in Component Inspector pane 48
PSSaveAsDCD 440

Q

Quarter 413, 414
Query subcomponents
creating 160
defining SQL filter criteria for 168
defining SQL join criteria for 166
defining SQL sort criteria for 169
for ODBC-based Sources and Destinations 160
identifying SQL tables and columns 161
properties for ODBC Sources 174
properties for ODBC-based Sources 161
selecting columns from selected SQL tables 164
selecting SQL tables for 163

R

Raise GeneralFailure 440
RaiseEvent 440
Randomize 440
Random-Number 440, 441
ReadFixed 440
ReadPrivateProfileRemove 440
ReadPrivateProfileRemoveAll 440
ReadPrivateProfileSection 440
ReadPrivateProfileSetCacheSize 440
ReadPrivateProfileString 440

ReadVariable 440
rearranging control bars and toolbars 51
Recap Log file 480
Record subcomponents
Redo command 64
Register panel
registering Transall Application as ActiveX server 313
renaming Transall components globally 54
Repeating your actions 64
Replace command 67
Replace$ 440
Resource control bar 275
Resume Next 440
Return 440
Reversing your actions 64
Right$ 440
RmDir 441
Rnd 441
Round 442
Round45 442
Round45A 442
RTrim$ 442
Rulebase 453
running a Transall Application in debug mode 321

S

Save As command 60
Save command 60
SaveSetting 442
Saving a project
Script Module components
Scripted Destination
scripts
Second 413, 414, 442
Seek 442
Select All command 66
SendEvent 443
SendEvent GeneralFailure To 443
separator options 120
Set 417
Set components
SetAttr 443
SetNull 443
Setting up the UNIX environment 26
Settings
Sgn 444

Share 62
Shell 444
Sin 444
Sine 444
SMTP email messages 474
sort criteria in SQL tables of Query subcomponents 169
SortSet 444
SortTable 444
SortTableMulti 444
Source components
source control with Transall files 306
Space 444
SQL *see* Structured Query Language (SQL)
Sqr 444
square root 444
standard error 450
standard output 451
Standard toolbar 80
Start command 103
statement syntax 411
Static As 445
status bar 45
Stop command 103
StrComp 445
String$ 445
StrReverse 445
Structured Query Language (SQL)
Sub 446
subroutines 339
Sun
    installation 26
    setting up the environment 26
Switch 447
Synchorinze
    DMG Report 100
Synchorinze menu
    View Queries 99
Synchornize menu
    View Links 98
Synchronize menu
    Data Sources 97
    Links 97
system requirements
    AIX 17
    Linux 19
    Sun Solaris 19

Windows 17

T
Table components
describing data 38
discussed 279
establishing relationships with Sets 282
properties reference 285
resource considerations for 280
Tag Editor
creating a new tag 212
Tan 447
TDMJOBS 396
TDMTERM 397
TDMWAIT 396
Text
finding 66, 68, 71
replacing 67
selecting all 66
Then 461
Threaded Data Manager
Time 447
Time Format 466, 469
TimeSerial 447
To Copy Text to the Clipboard 65
To Move Text to the Clipboard 65
Toggle Breakpoint command 105
tool bars 45
Toolbars menu
Tools menu
TRANDMAN 395, 398
Transall
Transall “how-to”
Transall Application
Transall Applications 31
Transall Compiler 40
Transall Database 279
Transall Editor
Transall executable 40
Transall Host
Transall Java scripting support
Transall Script language
Transall Scripts, noncomponent Tables in 341
Transall UNIX
copying files 26
running the install script 27

setting up the environment 26
TransAllBeginTrans 447
TransAllEndTrans 447
TransAllRollbackTrans 447
Trim$ 447
TrnSys$LocaleDefCurrencySym 448
TrnSys$LocaleDefDateSym 448
TrnSys$LocaleDefDecimalSym 448
TrnSys$LocaleDefDigitGroupSym 448
TrnSys$LocaleDefNegativeSym 448
TrnSys$LocaleDefTimeAmSym 448
TrnSys$LocaleDefTimePmSym 448
TrnSys$LocaleDefTimeSym 448
TrnSys$LocaleSetCurrencySym 448
TrnSys$LocaleSetDateSym 448
TrnSys$LocaleSetDecimalSym 448
TrnSys$LocaleSetDigitGroupSym 448
TrnSys$LocaleSetNegativeSym 448
TrnSys$LocaleSetTimeAmSym 448
TrnSys$LocaleSetTimePmSym 448
TrnSys$LocaleSetTimeSym 448
True 449
Type Statement 449

U
UBound 449
UCase$ 451
UpdateRow 451
Uppercase 451, 469
User Layouts
User Layouts command 78
Using the General Tab 308
Using the Tools>Options Dialog Box

V
Val 452
variable assignment 464
Variables Window 79
VDR (Varible Data Reformater) 197
VdrAddExplicitForm 452
VdrAddFormsLibrary 452
VdrAddTag 452
VdrBeginReformatter 452
VdrBuildMergeSet 452
VdrCallMrgUserOnWin32 452
VdrCloseVRF 452

VdrCloseVRFs 452
VdrDmgrfmtGetReasonCode 452
VdrDmgrfmtGetReturnCode 452
VdrDmgrfmtSetAllowMissingRulebase 452
VdrDmgrfmtSetAlternateParmList 452
VdrDmgrfmtSetEndMergeSet 453
VdrDmgrfmtSetMaxMessageLevel 453
VdrDmgrfmtSetNoEffDateMsg 453
VdrDmgrfmtSetSuppressImplictForms 453
VdrDmgrfmtSetSuppressSysPrint 453
VdrDmgrfmtSetVrfAllocDDname 453
VdrDmgrfmtSetWriteExplictForms 453
VdrDmgrfmtSetWriteRfcb 453
VdrEndMergeSet 453
VdrEndReformatter 453
VdrSetEffectiveDate 453
VdrSetJobDescription 453
VdrSetMessageFile 453
VdrSetProductionDefinition 453
VdrSetRulebase 453
VdrSetVRFFile 453
VdrSetWorkDirectory 453
VdrStartMergeSet 453
VdrSubmit 453
VERSION 449
View menu
viewing
VSAM
VsamDelete 454
VsamInsert 454
VsamRead 454
VsamReadPrior 454
VsamReplace 455
VsamSearch 455
VsamSearchBack 455
VsamSearchBackExact 455
VsamSearchExact 455

W

Walk instruction
Watch Window 79
Week 413, 414
Weekday 413, 414, 449
While 450
WIN32, installing Transall for 21
Window menu

With 450
Wizards
Working with Maps 269
Working with the Transall Database
Working with Transall Scripts and Script Modules
workspace area 53
WriteConStdErr 450
WriteConStdOut 450
WriteFixed 451
WriteVariable 451
WriteXmlDocumentSet 451

X
XML Destination
XML Plus data source 90
XML Plus Destination 95
XML Plus Source
XML Source, adding a Record 242
XmlClose 455
XmlDeclareColAttribute 455
XmlDeclareColCDATA 455
XmlDeclareColComment 455
XmlDeclareColProcInstr 455
XmlDeclareColStringData 455
XmlDeclareTableElement 455
XmlEncodingTranscode 456
XmlGetDeclaration 456
XmlGetDTD 456
XmlGetElementData 456
XmlGetNextElement 457
XmlOpen 458
XmlSetEncoding 458
XmlWriteWhiteSpace 458

Y
Year 413, 414, 455