



DEVELOP 10 TIMES FASTER



Don't forget to visit our site (www.windev.com) on a regular basis to find out whether upgraded versions are available.

Email address of Free Technical Support: freetechnicalsupport@windev.com

This documentation is not contractually binding. Modifications may have been made to the software since this guide was published. See the **online help**.

All product names or other trademarks mentioned in this publication are registered trademarks of their respective owners.
© PC SOFT 2014: This publication may not be reproduced in part or in full without the express consent of PC SOFT.

TABLE OF CONTENTS

PART 1 : CODE EDITOR

1. OVERVIEW	29
1.1 The code editor	29
1.2 Features linked to the code input	31
1.2.1 Code coloring	31
1.2.2 Automatic completion	31
1.2.3 Code wizard	31
1.2.4 Assisted input of functions	31
1.2.5 Help	31
1.2.6 Code history	32
1.2.7 Code check	32
1.2.8 Automatic indent	32
1.2.9 Translating the messages	32
1.2.10 Managing the breakpoints	33
1.2.11 Inserting specific processes	33
1.2.12 Other features	33

PART 2 : BASIC SYNTAX OF WLANGUAGE

1. INTRODUCTION	37
1.1 Some characteristics of WLanguage	37
1.2 Programming in server code and in browser code	38
2. THE VARIABLES	39
2.1 The simples types	39
2.1.1 Principle	39
2.1.2 Types of variables	39
2.1.3 Declaring a simple type	40
2.1.4 The different types of integer	40
2.1.5 The currency type	41
2.1.6 The numeric type	41
2.1.7 The real type	41
2.1.8 The String type	41
2.1.9 The Buffer type	42
2.1.10 The date type	42
2.1.11 The time type	43
2.1.12 The DateTime type	43
2.1.13 The Duration type	43
2.1.14 The Variant type	44
2.2 Operations available for the dates, times and durations, ...	44

2.3 Managing the NULL value	44
2.3.1 Null and the queries	45
2.3.2 Null and the variants	46
2.3.3 Null and the numeric values.....	46
2.3.4 Null and the WLanguage functions.....	46
2.3.5 Null and the dynamic objects.....	46
2.4 Constants	46
2.5 The advanced types	47
2.5.1 Simple Array	47
2.5.2 Dynamic array	48
2.5.3 Fixed array	49
2.5.4 Associative array.....	51
2.5.5 Composite variable	51
2.5.6 Structure.....	52
2.5.7 Dynamic structure	54
2.5.8 Automation object.....	54
2.5.9 Dynamic Automation object.....	55
2.5.10 Data source	56
2.5.11 File description	57
2.5.12 Link description	58
2.5.13 Item description.....	59
2.5.14 Font.....	60
2.5.15 Connection	60
2.5.16 Queue	61
2.5.17 List	62
2.5.18 Stack.....	62
2.6 Local variables/global variables	64
2.6.1 Global variables	64
2.6.2 Local variables.....	65
2.7 Rule for variable scope	65
3. OPERATORS	66
<hr/>	
3.1 Overview	66
3.2 Logical operators	66
3.2.1 Overview	66
3.2.2 Rules	66
3.2.3 Notes.....	66
3.3 Arithmetic operators	67
3.3.1 Overview	67
3.3.2 Calculation rules	67
3.3.3 Notes.....	67
3.4 Binary operators	67
3.4.1 Binary operators.....	67
3.4.2 Shift operators	68
3.4.3 Operator for direct access.....	68
3.5 Comparison operators	69
3.5.1 Overview	69
3.5.2 Details	69

3.6 Operators on character strings	70
3.6.1 Flexible equality and very flexible equality	70
3.6.2 The [[and]] operator.....	71
3.6.3 Operators on the character strings and UNICODE	71
3.6.4 Position in a character string	71
3.6.5 Functions of WLanguage	71
3.7 Operator on address	72
3.8 Indirection operators	72
3.9 Various operators	73
3.9.1 The brackets	73
3.9.2 The square brackets.....	74
3.9.3 The comma.....	74
3.9.4 The semicolon	74
3.9.5 Colon	74
3.9.6 The dot	74
3.9.7 The double dot.....	74
3.9.8 Triple dot	74
3.9.9 The double slash.....	74

4. WLANGUAGE STATEMENTS **75**

4.1 Composite statements	75
4.1.1 LOOP statement.....	75
4.1.2 GOTO statement	76
4.1.3 FOR statement.....	76
4.1.4 FOR EACH/FOR ALL statement, file browsing of data	77
4.1.5 FOR EACH/FOR ALL statement, parsing strings	79
4.1.6 FOR EACH/FOR ALL statement, control browsing.....	80
4.1.7 FOR EACH/FOR ALL statement, array browsing.....	81
4.1.8 FOR EACH/FOR ALL statement, browsing associative arrays	82
4.1.9 SWITCH statement	83
4.1.10 IF statement	83
4.1.11 WHILE statement	84
4.2 Simple statements	85
4.2.1 CONTINUE statement.....	85
4.2.2 RETURN statement.....	86
4.2.3 RESULT statement	87
4.2.4 BREAK statement.....	87

5. RESERVED WORDS **89**

5.1 External	89
5.2 MyWindow	90
5.3 MyPage	90
5.4 MySource	91
5.5 Modulo	91
5.6 MySelf	91
5.7 MyPopupControl	92
5.8 MyReport	93

5.9 MyFile	93
5.10 MyParent	93
5.11 STOP (calling the debugger)	94
6. PROCEDURE AND FUNCTION	95
<hr/>	
6.1 Overview	95
6.2 Global and local procedure/function	95
6.2.1 Definition	95
6.2.2 Global procedure	95
6.2.3 Local procedure	97
6.3 Set of procedures	98
6.3.1 Definition	98
6.3.2 Creating a set of procedures	98
6.3.3 Importing a set of procedures	98
6.4 Declaring a procedure/a function	98
6.4.1 Syntax	98
6.4.2 Exiting from a procedure	98
6.4.3 Returning a result	98
6.5 Calling a procedure/function	99
6.6 Parameter of a procedure/function	99
6.6.1 Type of the parameters	99
6.6.2 Passing parameters	100
6.6.3 Optional parameter	101
6.6.4 Procedure with a variable number of parameters	101
6.7 Overload a WLanguage function	101
6.7.1 Definition	101
6.7.2 How do I proceed?	101
6.7.3 Differentiating between the WLanguage function and the custom function	101
6.8 Prototype overload	102
6.8.1 Overview	102
6.8.2 How do I proceed?	102
6.8.3 Managing the overload at run time	102
7. MANAGING EXCEPTIONS	104
<hr/>	
7.1 Overview	104
7.1.1 Displaying a custom message	104
7.1.2 Exception mechanism	104
7.2 Mechanism of general exceptions	104
7.2.1 Overview	104
7.2.2 Declaration syntaxes	105
7.2.3 Declaring several processes of general exceptions	105
7.2.4 General notes	105
7.2.5 Special case	106
7.3 Mechanism of specific exceptions	106
7.3.1 Overview	106
7.3.2 Declaration syntaxes	106



7.4 Mechanism of automated exceptions	106
7.4.1 Implementation	106
7.4.2 Running the error process ("CASE EXCEPTION:" in the code)	107
7.4.3 Running a procedure of exception process	107
7.5 Functions for managing the exceptions	107
8. OBJECT-ORIENTED PROGRAMMING (OOP)	108
8.1 Overview of OOP (Object Oriented Programming)	108
8.2 OOP concepts	108
8.2.1 Class	108
8.2.2 Object	108
8.2.3 Constructor and destructor	109
8.2.4 Inheritance	109
8.2.5 Data encapsulation	109
8.3 Class, members and methods	109
8.3.1 Declaration of the class	109
8.3.2 Declaration of members	110
8.3.3 Declare the constants	110
8.3.4 Declaring the methods	110
8.3.5 Creating and declaring properties	111
8.4 Constructor and destructor	112
8.4.1 Constructor of the class	112
8.4.2 Constructor of the base classes and of the members	112
8.4.3 Destructor method	112
8.5 Object	113
8.5.1 Declaring an object	113
8.5.2 The members of an object	113
8.5.3 The methods of an object	113
8.5.4 Lifespan of an object	113
8.6 Dynamic instantiation of an object	114
8.6.1 Declaring a dynamic object	114
8.6.2 Instantiating a dynamic object	114
8.6.3 Freeing a dynamic object	114
8.7 Class inheritance	114
8.7.1 Syntax	115
8.7.2 Redefining the methods	115

PART 3 : MANAGING WINDOWS, PAGES AND CONTROLS

1. MANAGING THE WINDOWS	119
1.1 Overview	119
1.2 Functions for managing the windows	119
1.3 MDI functions	121
1.4 Functions for managing the menus	121

2. MANAGING THE PAGES	123
2.1 Overview	123
2.2 Functions for managing the pages	123
2.3 Functions for managing the menus	124
3. MANAGING THE "BACK" BUTTON IN A PAGE	125
3.1 Overview	125
3.1.1 Two methods can be used to manage the browser "Back" button.....	125
3.1.2 Example of desynchronization	125
3.2 Preventing from using the "Back" button	126
3.2.1 Operating mode	126
3.2.2 Implementation	126
3.3 Managing the synchronization	126
3.3.1 Overview	126
3.3.2 Default synchronization	126
3.4 Synchronization by programming	127
4. COMMUNICATING WITH THE USER	128
4.1 Overview	128
4.2 The standard dialog boxes	128
4.3 The advanced dialog boxes	129
4.3.1 Overview	129
4.3.2 Directive questioning.....	129
4.3.3 Immediate input.....	129
4.3.4 Message database.....	129
4.4 Customizing the dialog boxes	129
4.4.1 Customizing the system information windows.....	130
4.4.2 Stop customizing the system information windows	130
4.4.3 Multilingual dialog boxes	130
4.5 Automatically close the dialog boxes	130
4.6 Advanced communication with the user	131
4.6.1 Overview	131
4.6.2 The available WLanguage functions	131
4.6.3 Managing a dialog via cells.....	131
4.6.4 Managing a dialog with pages.....	131
4.6.5 Managing a dialog via YesNo and OKCancel	132
4.7 Functions for managing the dialog boxes	132
5. MANAGING DRAG AND DROP	133
5.1 What is "Drag and Drop"?	133
5.2 Automatic drag and drop	133
5.2.1 WinDev: The controls affected by the automatic drag and drop	133
5.2.2 WebDev: The controls affected by the automatic drag and drop	133
5.2.3 Configuring the default "Drag and Drop"	134

5.3 Programmed "Drag and Drop"	134
5.3.1 Principle	134
5.3.2 Programming.....	134
5.4 "Drag and Drop" from the explorer	135
5.4.1 Principle	135
5.4.2 Functions specific to "Drag and Drop" from the explorer	135
5.4.3 Programming.....	135
5.5 Functions for managing Drag and Drop	136
6. MANAGING THE CONTROLS	137
<hr/>	
6.1 Overview	137
6.2 General functions for handling the controls	137
7. PROPERTIES OF WINDOWS, PAGES AND CONTROLS	139
<hr/>	

PART 4 : STANDARD FUNCTIONS

1. HANDLING THE NUMERIC VALUES	151
<hr/>	
1.1 Overview	151
1.2 The matrices	151
1.2.1 Definition	151
1.2.2 Handling the matrices	151
1.3 The statistics	151
1.4 Financial calculations	152
1.5 Functions for managing the numeric values	152
1.5.1 Miscellaneous functions	152
1.5.2 Binary functions.....	153
1.5.3 Matrix functions	153
1.5.4 Financial functions.....	154
1.5.5 Statistical functions	154
2. HANDLING THE CHARACTER STRINGS	155
<hr/>	
2.1 Handling the content of a string	155
2.2 WinDev and the Unicode format	155
2.2.1 What is the UNICODE format	155
2.2.2 WinDev and Unicode.....	155
2.3 Handling character strings in Pocket PC	156
2.4 Functions for managing the character strings	156

3. HANDLING THE DATES AND TIMES	159
3.1 Overview	159
3.2 Different methods for handling the dates and times	159
3.3 Handling dates/times found in each edit control	159
3.4 Functions for managing the dates and times	159
4. HANDLING THE CHARTS	161
4.1 Overview	161
4.2 The different types of charts	161
4.2.1 The "Pie" charts	161
4.2.2 The "Column" charts	161
4.2.3 The "Line" charts.....	161
4.2.4 The "Scatter" charts	162
4.2.5 The "Stock" charts	162
4.2.6 3D charts	162
4.3 How do I create charts?	163
4.3.1 Creating the charts in the editors.....	163
4.3.2 Creating the charts by programming.....	163
4.3.3 Charts and threads.....	163
4.3.4 Default values of a chart.....	163
4.4 Functions for managing the charts	164
5. HANDLING THE MEMORY ZONES	166
6. HANDLING THE EXTERNAL FILES	167
6.1 Overview	167
6.2 Handling the content of external files	167
6.3 Handling files	167
6.4 Handling disks and their directories	167
6.5 Functions for managing the external files	168
7. SHARED MEMORY ZONES	171
7.1 Overview	171
7.2 How do I proceed?	171
7.2.1 Creating a shared memory zone.....	171
7.2.2 Finding out whether a shared memory zone already exists.....	171
7.2.3 Handling the content of a shared memory zone by programming.....	171
7.2.4 WLanguage functions.....	172
7.3 Dialog between several applications	172
7.3.1 Automatic notification of modifications.....	172
7.3.2 Manual synchronization.....	172

7.4 Naming the shared memory zones	173
7.4.1 Managing the share mode.....	173
7.4.2 Correspondence between the name provided to fMemOpen and the opening in C.....	173
7.5 Functions for managing the shared memory zones	173
8. PRINTING IN WLANGUAGE	174
<hr/>	
8.1 Overview	174
8.2 Principle for printing in WLanguage	174
8.2.1 Step 1: Configuring the print parameters.....	174
8.2.2 Step 2: Creating print fonts.....	175
8.2.3 Step 3: Printing.....	175
8.3 Print functions	177
8.4 PDF functions	179
9. MANAGING WINDOWS	180
<hr/>	
9.1 Overview	180
9.2 Functions for managing the registry	180
9.3 Functions for managing the clipboard	181
9.4 Functions for managing the recycle bin	181
9.5 Functions for managing the mouse	181
9.6 Functions for checking the spelling	182
9.7 Speech recognition functions	182
9.8 Functions for managing the serial and parallel ports	182
9.9 Functions for managing Twain devices	183
9.10 USB functions	183
9.11 MCI functions	184
9.12 Service functions	184
9.13 System functions	185
9.14 Miscellaneous Windows functions	186
9.15 Windows event	187
9.16 Java functions	188
9.17 Miscellaneous WinDev/WebDev functions	188
9.18 Functions for managing the executables	192
9.19 Hasp functions	192
9.20 Functions for DDE management	193
9.21 Functions for managing the applications with "live update"	193
9.22 Functions for managing the networks	194
9.23 Functions for managing the SNMP protocol	194
9.24 Functions for managing the projects	194
9.25 Functions for managing the scheduler	195

10. ENABLING AN APPLICATION	196
10.1 Overview	196
10.2 How do I proceed?	196
10.2.1 Principle	196
10.2.2 Implementation in the client application	196
10.2.3 Implementation in the application of the provider.....	196
10.3 Functions for managing the activation keys	196
11. HANDLING YOUR XLS FILES	197
11.1 Overview	197
11.2 Method 1: Dynamically handling the XLS and XLSX files	197
11.3 Method 2: Reading the Excel files (kept for backward compatibility)	198
11.4 WLanguage functions	198
11.5 XLS functions	199
12. THE ARCHIVES	200
12.1 Overview	200
12.2 Handling the archives	200
12.3 The single-part and multi-part archives	201
12.3.1 Overview	201
12.3.2 Principle	201
12.3.3 Examples.....	201
12.4 Archiving functions	202
13. BURNING A CD OR A DVD	203
13.1 Overview	203
13.2 Burning a CD/DVD	203
13.3 Burn functions	203
14. FUNCTIONS FOR ACCESSING A POCKET PC	205
15. FUNCTIONS SPECIFIC TO WINDEV MOBILE	207
15.1 WinDev Mobile and SIM cards	207
15.1.1 Overview	207
15.1.2 Required configuration	207
15.1.3 Operating mode in GO mode and at run time	207
15.1.4 WLanguage functions.....	207
15.2 Pocket keyboard	207
15.2.1 Overview	207
15.2.2 WLanguage functions.....	208

PART 5 : WEB SPECIFIC FEATURES

1. UPLOADING FILES	211
1.1 Overview	211
1.2 Implementing the upload in a WebDev site	211
1.2.1 Available elements	211
1.2.2 Uploading a file in a page	212
1.2.3 Displaying the image to upload	212
2. DOWNLOADING FILES	213
2.1 Overview	213
2.2 Implementing file download in a WebDev site	213
2.2.1 Using the description window of controls (button, link,...)	213
2.2.2 Programming	213
2.2.3 Forcing the file download	213
3. COOKIES	214
3.1 Overview	214
3.2 What is a cookie made of?	214
3.3 WebDev and the management of cookies	214
3.3.1 Available elements	214
3.3.2 Writing a cookie on the computer of the Web user	214
3.3.3 Reading a cookie on the computer of the Web user (server code and browser code)	215
3.4 Checking the management of cookies in a WebDev site	215
4. VALIDITY OF THE SITE PAGES	216
4.1 Overview	216
4.2 How do I proceed?	216
4.2.1 Defining the validity period of pages	216
4.2.2 Deleting the validity period	216
4.2.3 Configuring the Windows application server to manage the validity period	216
4.2.4 Configuring the Linux application server to manage the validity period	217
5. INCLUDING JAVASCRIPT FILES OR A WEB RESOURCE	218
5.1 Overview	218
5.2 How do I proceed?	218
5.2.1 Including Javascript files (.js)	218
5.2.2 Including external resources in the site	218
5.2.3 Automatic detection of the encoding	219
5.3 Handling external Javascript objects from WLanguage	219

6. AJAX	220
6.1 Overview	220
6.2 Automatic and immediate AJAX	220
6.2.1 Overview	220
6.2.2 Processes that can use AJAX automatically	221
6.2.3 Elements and characteristics that can be automatically used by AJAX	221
6.2.4 Specific features	223
6.3 Programmed AJAX	223
6.3.1 Overview	223
6.3.2 Functions for AJAX management	223
6.3.3 Procedures that can be called by AJAX	224
6.3.4 WLanguage functions useless in AJAX	224
7. VISTA GADGETS	225
7.1 Overview	225
7.2 Creating the gadget	225
7.2.1 The steps	225
7.2.2 Generating the gadget	225
7.3 Programming the Vista gadgets	226
7.3.1 Overview	226
7.3.2 The different types of pages	226
7.3.3 The WLanguage functions	227
8. USING OFFLINE SITES	228
8.1 Overview	228
8.2 Implementation	228
8.2.1 Defining the resources to cache	228
8.2.2 Configuration of the server	228
8.2.3 Programming technique	228
8.2.4 Running the test of the site	229
8.3 Access in local mode to a database (SQLite)	229
8.3.1 How do I manage a local database?	229
8.3.2 The SQL functions	230
8.3.3 Saving the data of an offline site locally	230
9. LOCAL STORAGE	232
9.1 Overview	232
9.2 The WLanguage functions	232

10. SSL: SECURED TRANSACTIONS	233
10.1 Overview	233
10.2 Implementing the secured transactions with the SSL protocol	233
10.3 Obtaining an SSL certificate for IIS2 (2.0 or later)	233
10.3.1 Step 1: Creating a certificate	233
10.3.2 Step 2: Requesting a certificate	234
10.3.3 Step 3: Certification of the certificate on the server	234
10.4 Inserting secure transactions (SSL) into a WebDev site	235
10.4.1 Principle	235
10.4.2 Implementation	235
10.4.3 Going back to standard mode (non-secured transaction) in the current browser	235
11. JSON	236
11.1 Overview	236
11.2 Getting information in JSON format	236
11.2.1 Operating mode	236
11.2.2 Example for using the JSONExecute function	236
11.2.3 Example for using the JSONExecuteExternal function	237
11.3 Creating pages that return JSON data	237
11.3.1 Case of a page called by JSONExecute	237
11.3.2 Case of a page called by JSONExecuteExternal	237

PART 6 : COMMUNICATION

1. THE COMMUNICATION	241
1.1 Communication with WinDev/WebDev	241
1.2 Communication with WinDev Mobile	242
2. COMMUNICATE BY EMAILS	243
2.1 Overview	243
2.2 Managing the emails	243
2.3 Synchronous/Asynchronous mode (WebDev only)	243
2.4 Manage the emails via the POP3/SMTP protocols	244
2.4.1 Overview of the POP3/SMTP protocols	244
2.4.2 Principle	244
2.5 Manage the emails with the IMAP protocol	247
2.5.1 Overview of the IMAP protocol	247
2.5.2 Using the IMAP protocol	247
2.6 Managing emails with "Simple MAPI" (WinDev and WebDev)	247
2.7 Manage the emails with CEMAPI (WinDev Mobile only)	250
2.8 Reading and writing an email	250
2.9 Functions for managing the emails	250

3. ACCESSING LOTUS NOTES AND OUTLOOK 252

3.1 Access to Lotus Notes 252

- 3.1.1 Overview252
- 3.1.2 Method for accessing Lotus Notes252
- 3.1.3 Quick method if you do not want to access the documents252
- 3.1.4 Handling the data.....252

3.2 Access to Outlook 252

- 3.2.1 Overview252
- 3.2.2 Method for accessing Outlook.....252
- 3.2.3 Sending and receiving emails253
- 3.2.4 Version of Outlook253
- 3.2.5 Handling the data.....253

3.3 Lotus Notes and Outlook functions 253

- 3.3.1 Email functions for Lotus Notes.....253
- 3.3.2 Email functions for Outlook.....254
- 3.3.3 Task functions255
- 3.3.4 Appointment functions255
- 3.3.5 Contact functions256
- 3.3.6 Group functions256
- 3.3.7 Notes functions257

4. GOOGLE 258

4.1 Managing the Google contacts 258

- 4.1.1 Overview258
- 4.1.2 How do I manage the Google contacts?258
- 4.1.3 How do I retrieve a Google contact?258
- 4.1.4 How do I modify or delete the Google contacts?258
- 4.1.5 Functions for managing the Google contacts259

4.2 Managing the Google calendars 259

- 4.2.1 Overview259
- 4.2.2 How do I manage a Google calendar?259
- 4.2.3 How do I retrieve a Google calendar and its elements?260
- 4.2.4 How do I add, modify or delete events in a Google calendar?260
- 4.2.5 Functions for managing the Google calendars261

4.3 Using the service for managing the Google Picasa photo albums 262

- 4.3.1 Overview262
- 4.3.2 How do I proceed?262
- 4.3.3 Functions for managing the Picasa albums263

4.4 Managing the Google documents 264

- 4.4.1 Overview264
- 4.4.2 How do I manage the Google documents?264

4.5 Using the Google Maps service 264

- 4.5.1 Overview264
- 4.5.2 How do I proceed?265
- 4.5.3 Other services265
- 4.5.4 Functions for managing the Google maps.....265

5. SALESFORCE	266
5.1 Use the Salesforce service	266
5.1.1 Overview	266
5.1.2 How do I proceed?	266
5.2 Salesforce functions	266
6. RSS STREAM	268
6.1 Overview	268
6.2 How do I proceed?	268
6.3 Functions for managing the RSS streams	268
7. LDAP SERVER	269
7.1 Overview	269
8. WINDEV AND TELEPHONY	270
8.1 Overview	270
8.2 Managing the incoming calls	270
8.2.1 The different steps	270
8.2.2 Example	271
8.3 Managing the outgoing calls	272
8.3.1 The different steps	272
8.3.2 Handling an outgoing call	273
8.4 Telephony functions	273
9. MANAGING THE SMSs	275
9.1 Overview	275
9.2 The SMS structure	275
9.2.1 Overview	275
9.2.2 The variables of the SMS structure	275
9.2.3 Reading and deleting the SMSs found on Smartphone	275
9.2.4 Different types of numbers	276
9.2.5 Operating mode in GO mode and at run time	276
9.3 WLanguage functions	276
10. SENDING FAXES	277
10.1 Overview	277
10.2 Configuring the "fax server"	277
10.2.1 Configuring the current computer	277
10.2.2 Configuring the fax server in Windows 2000	277
10.2.3 Configuring the fax server in Windows XP	277
10.3 Application or site for sending faxes	278
10.3.1 Sending a fax from an application or from a site	278
10.3.2 Sending a fax created with the report editor	278

10.4 Configuring the fax server by programming	278
10.4.1 Options of the fax server	278
10.4.2 Tips	279
10.5 Functions for managing the faxes	279
11. RETRIEVING THE HTML PAGES	280
<hr/>	
11.1 Overview	280
11.2 HTTP functions	280
12. MANAGING FILES ON INTERNET	281
<hr/>	
12.1 Overview	281
12.1.1 Uploading and downloading files via WinDev FTP or RPC: the rules to follow	281
12.1.2 Other features	281
12.2 Detailed use of WinDev FTP/RPC	281
12.2.1 Step 1: Establishing a connection to a WinDev RPC/FTP server.....	281
12.2.2 Step 2: Transmitting a file to a WinDev FTP server	281
12.2.3 Step 3: Retrieving a file from a WinDev FTP server.....	282
12.2.4 Step 4: Closing a connection to a WinDev RPC/FTP server	282
12.3 Net functions	283
13. COMMUNICATING WITH AN FTP SERVER	284
<hr/>	
13.1 Handling files on a RPC server	284
13.1.1 Overview	284
13.1.2 FTP	284
13.1.3 Principle	284
13.1.4 Relative path/Absolute path.....	284
13.1.5 Example	285
13.2 FTP functions	285
14. MANAGING THE SOCKETS	286
<hr/>	
14.1 Overview	286
14.1.1 Different possibilities	286
14.1.2 Example	286
14.2 Client WinDev application/Client WebDev site	286
14.2.1 Principle of a client application or client site.....	286
14.2.2 Transmission mode of information	287
14.3 WinDev "Simplified Server" application	287
14.3.1 The simplified server.....	287
14.3.2 Transmission mode of information	288
14.4 Standard socket server	288
14.4.1 The standard socket server	288
14.4.2 Transmission mode of information	289
14.5 Socket functions	290

15. MANAGING THE BLUETOOTH KEYS	291
15.1 Overview	291
15.2 How do I proceed?	291
15.3 Which keys should be used?	291
15.4 The Bluetooth and OBEX functions	292
15.4.1 Bluetooth functions	292
15.4.2 OBEX functions	292
16. MANAGING THE THREADS	293
16.1 Overview	293
16.1.1 From simple management to advanced management of threads	293
16.1.2 Example	293
16.2 Principle for using the threads	293
16.2.1 Simple management of threads	293
16.2.2 Characteristics of the threads	293
16.2.3 Access to the existing elements and HFSQL context	294
16.2.4 Limits of the processes performed by the thread	294
16.3 Managing the semaphores in the threads	294
16.3.1 Principle	294
16.3.2 Implementing a semaphore	295
16.3.3 A limited semaphore: the critical section	296
16.4 Managing the mutexes in the threads	296
16.4.1 Principle	296
16.4.2 How do I implement a mutex?	296
16.4.3 The functions for managing the mutexes	297
16.5 Synchronizing the threads via signals	297
16.5.1 Simple management of signals	297
16.5.2 Advanced management of signals	298
16.6 Managing the opening of a WinDev window in a secondary thread	298
16.6.1 Opening a window from a secondary thread	298
16.6.2 Example	298
16.7 Functions for managing the threads	299
17. SOAP	301
17.1 Overview	301
17.1.1 WinDev/WebDev and the SOAP protocol	301
17.1.2 Example	301
17.2 Running procedures on a SOAP server	301
17.2.1 Principle	301
17.2.2 The SOAP structure	302
17.3 Creating and installing a WinDev SOAP server application	302
17.3.1 Principle	302
17.3.2 How do I create an application SOAP server	303
17.4 SOAP functions	304

18. XML WEBSERVICES	305
18.1 Importing XML Webservices	305
18.1.1 Overview	305
18.1.2 Importing a Webservice into a project	305
18.1.3 Updating the description of a Webservice	305
18.1.4 Properties of a Webservice	305
18.1.5 Using a Webservice imported into the project	306
18.1.6 Distributing a WinDev application that uses a Webservice	307
18.2 Generating an XML Web service	307
18.2.1 Overview	307
18.2.2 How do I make a Webservice available?	308
18.2.3 Generating a Webservice	308
18.2.4 Deploying a Webservice	308
19. XML	311
19.1 Managing XML documents	311
19.1.1 Overview	311
19.1.2 Definition	311
19.1.3 Principle	311
19.1.4 Using a string variable	311
19.1.5 Using an xmlDocument variable	312
19.2 Managing the XSD	312
19.2.1 Overview	312
19.2.2 Importing an XSD file into a project	313
19.2.3 Using an imported description in the project	313
19.3 Functions for managing the XML documents	313
20. .NET ASSEMBLIES	315
20.1 Overview	315
20.1.1 Definition	315
20.1.2 WinDev and .NET	315
20.2 Conditions for using a .NET assembly	315
20.2.1 Installing the .NET framework	315
20.2.2 Defining the .NET security level	315
20.2.3 Making the DLLs required to run the .NET assembly accessible	316
20.3 Creating a .NET assembly from WinDev	316
20.4 Creating a .NET assembly accessible by COM	317
20.4.1 Overview	317
20.4.2 Creating a .NET assembly accessible by COM from a WinDev project	317
20.5 Creating the setup program of a .NET assembly	317
20.6 Using .NET assemblies in a WinDev application	317

PART 7 : MANAGING THE DATA FILES

1. MANAGING DATA FILES	321
1.1 HFSQL data file management	321
1.1.1 Create a data file	321
1.1.2 Opening and closing data files	321
1.1.3 Managing HFSQL files larger than 2GB	322
1.1.4 Managing the keys	322
1.1.5 Managing the composite keys	324
1.1.6 Accessing the data file items	325
1.1.7 Schemas: the form mode and the table mode	330
1.1.8 Moving and positioning in a data file	330
1.2 The data files in FoxPro xBase format	331
1.2.1 Overview	331
1.2.2 Using the native xBase/FoxPro access	331
1.2.3 Importing the structure of the files	331
1.2.4 Important programming points	331
1.3 The data files found on a mobile device (Pocket PC, iOS, Android)	333
1.3.1 Handling a HFSQL database	333
1.4 Data files specific to Windows Mobile	333
1.4.1 Handling a CEDB database (Pocket PC only)	333
1.4.2 Functions for handling a CEDB database	335
2. ADVANCED FEATURES	336
2.1 Managing aliases	336
2.1.1 Several physical files with identical logical description	336
2.1.2 Several logical files linked to a single physical file	336
2.1.3 Functions for managing aliases	336
2.1.4 Create an alias on what?	337
2.1.5 Characteristics of an alias	337
2.1.6 Handling the alias file and its items	337
2.2 Managing NULL in HFSQL	337
2.2.1 How to manage the Null value in one of your items?	337
2.2.2 How can I use the NULL value in my applications?	338
2.3 Protecting and encrypting data files	339
2.3.1 The protection methods	339
2.3.2 Managing encrypted files	339
2.4 Managing an identifier	339
2.4.1 Automatic management	339
2.4.2 Manual management	340
2.4.3 Checking the uniqueness of a key	340
2.5 Automatic check of referential integrity	340
2.5.1 Benefit of the check of referential integrity	340
2.5.2 Definitions	340
2.5.3 The different types of links	341
2.5.4 Programming the automatic check of referential integrity	341

2.6 Managing the "memo" files	343
2.6.1 Text memo and binary memo	343
2.6.2 Image, sound, OLE and other binary memos	343
2.7 Reassigning data files	344
2.7.1 Benefit	344
2.7.2 Modifying the storage directory	344
2.7.3 Modifying the name of a data file	344
2.7.4 Keeping trace of reassignments	344
2.8 Full-text search and index	347
2.8.1 Overview	347
2.8.2 How do I perform a "full-text" search?	347
2.8.3 How do I create a full-text index?	347
2.8.4 How do I perform a full-text search?	347
2.8.5 Analyzing the result of a "full-text" query	349
2.8.6 Managing the full-text indexes by programming	349
2.9 Transactions	350
2.9.1 What is a transaction?	350
2.9.2 Knowing how to use the transactions according to your requirements	350
2.9.3 Principles	351
2.9.4 Handling the transactions by programming	351
2.9.5 Managing the special cases	352
2.9.6 Advanced management	353
2.10 Logs	353
2.10.1 General points	353
2.10.2 Implementing the log process	354
2.10.3 Files created when implementing the log process	355
2.10.4 WDLLog: Tool for log management	355
2.10.5 Handling the logs by programming	356
2.11 Automatic modification of data files	356
2.11.1 Principle	356
2.11.2 When is the automatic data modification required?	357
2.11.3 Performing the automatic data modification	357
2.11.4 Notes	358
2.12 Creating dynamic (or temporary) files	358
2.13 Retrieving the structure of the HFSQL files found in an analysis	359
2.14 Speeding up processes and optimizing an application or a site	359
2.14.1 Management of transactions	359
2.14.2 Managing the log	359
2.14.3 Managing the memos	360
2.14.4 Managing the ".REP"	360
2.14.5 Management of replication	360
2.14.6 The management of triggers	360
2.14.7 The management of RPC	360
2.15 ODBC driver on HFSQL	360
2.15.1 Overview	360
2.15.2 Configuration	360

2.16 ODBC on HFSQL via J++ and JDBC	361
2.16.1 Overview	361
2.16.2 Setup	361
2.16.3 Configuration.....	361
2.16.4 Use	361
2.17 OLE DB provider for HFSQL	362
2.17.1 Overview	362
2.17.2 Setup	362
2.17.3 Configuration.....	362
3. SQL LANGUAGE	364
<hr/>	
3.1 Overview	364
3.2 SQL language HFSQL	364
3.2.1 Overview.....	364
3.2.2 SQL commands that can be used with HFSQL	364
3.2.3 Functions for running queries	365
3.3 SQL language and other databases	366
3.3.1 Overview.....	366
3.3.2 Query created in the query editor.....	366
3.3.3 Query created by programming.....	366
4. MANAGING THE FILE LOCKS	368
<hr/>	
4.1 Overview of locks	368
4.1.1 Locking files.....	368
4.1.2 What is a lock?	368
4.1.3 When should you lock and what should you lock?	368
4.1.4 WLanguage and the locks	368
4.2 Managing locks	369
4.2.1 Example illustrating the need for locks.....	369
4.2.2 Structure of locks.....	369
4.2.3 Dead lock (inter locking)	369
4.3 The available lock modes	370
4.3.1 Single-user mode	370
4.3.2 Multi-user mode	372
4.3.3 The possible locks in multi-user mode	372
4.4 Assisted management of HFSQL errors	373
4.4.1 Principle.....	373
4.4.2 Standard operating mode	373
4.4.3 Customization.....	375
4.4.4 Disabling the assisted management (WinDev and WebDev)	376
5. THE HFSQL VIEWS	377
<hr/>	
5.1 Overview of HFSQL views	377
5.2 Benefits of views	377
5.3 Handling views	377

5.4 Creating HFSQL views	378
5.4.1 Choosing the items of a view	378
5.4.2 Choosing the initial sort item of the view	378
5.4.3 Selecting records from the view	379
5.4.4 Union operations between several views	379
6. HFSQL TRIGGERS	380
6.1 Overview	380
6.1.1 Definition	380
6.1.2 Benefits of triggers	380
6.2 How to create and handle triggers?	380
6.2.1 Functions for handling the triggers	380
6.2.2 Handling triggers	381
7. THE DATA REPLICATION	382
7.1 Overview of the replication	382
7.1.1 Overview	382
7.1.2 Vocabulary specific to the replication	382
7.1.3 Note	382
7.2 Implementing the universal replication	383
7.2.1 Activation	383
7.2.2 Declaring the master database	383
7.2.3 Declaring the subscriber databases	383
7.3 Replication between heterogeneous databases	383
7.4 Limitations	384
8. MANAGING FILES IN "BACK OFFICE"	385
8.1 Overview	385
8.2 The update by email	385
8.3 The remote access to HFSQL	385
8.3.1 Definition	385
8.3.2 Details of the three use modes	385
8.4 The data replication	386
8.4.1 Overview	386
8.4.2 Implementing the replication	386
9. ACCESSING EXTERNAL DATABASES	387
9.1 Overview	387
9.2 Specific features	387
9.3 Functions for managing the external databases	387

10. HFSQL FUNCTIONS 388

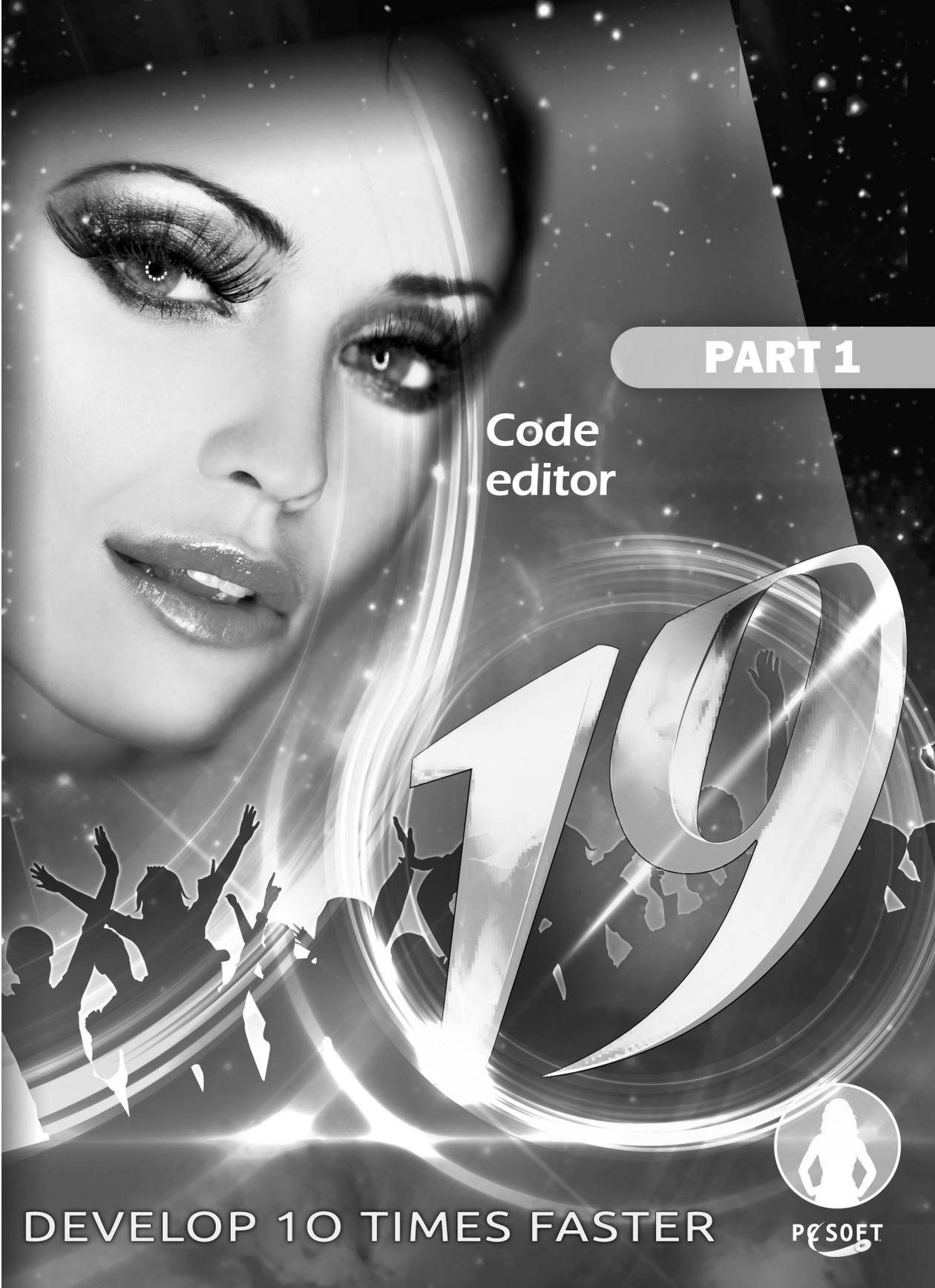
11. HFSQL PROPERTIES 394

12. SQL FUNCTIONS 398

12.1 Details of functions 399

- 12.1.1 ELT 399
- 12.1.2 EXTRACTVALUE 399
- 12.1.3 LEFT 399
- 12.1.4 RIGHT 399
- 12.1.5 SPLIT_PART 399
- 12.1.6 MID, SUBSTR and SUBSTRING 400
- 12.1.7 LTRIM 400
- 12.1.8 RTRIM 401
- 12.1.9 TRIM 401
- 12.1.10 REPLACE 401
- 12.1.11 REVERSE 402
- 12.1.12 TRANSLATE 402
- 12.1.13 CONCAT 402
- 12.1.14 STRING_AGG 402
- 12.1.15 LPAD 402
- 12.1.16 REPEAT 402
- 12.1.17 RPAD 402
- 12.1.18 SPACE 403
- 12.1.19 LOWER 403
- 12.1.20 LCASE 403
- 12.1.21 UCASE 403
- 12.1.22 UPPER 403
- 12.1.23 LEN and LENGTH 403
- 12.1.24 INSTR 404
- 12.1.25 FIELD 404
- 12.1.26 PATINDEX 404
- 12.1.27 POSITION 404
- 12.1.28 COUNT 404
- 12.1.29 AVG 405
- 12.1.30 MAX 405
- 12.1.31 MIN 405
- 12.1.32 SUM 405
- 12.1.33 BOTTOM 405
- 12.1.34 TOP 406
- 12.1.35 ASCII 406
- 12.1.36 SOUNDEX, SOUNDEX LIKE 406
- 12.1.37 SOUNDEX2, SOUNDEX2 LIKE 406
- 12.1.38 ADD_MONTHS 406
- 12.1.39 LAST_DAY 406
- 12.1.40 MONTHS_BETWEEN 407
- 12.1.41 NEW_TIME 407
- 12.1.42 NEXT_DAY 407

12.1.43 ROUND.....	407
12.1.44 SYSDATE	407
12.1.45 TRUNC.....	407
12.1.46 COALESCE.....	407
12.1.47 GREATEST.....	407
12.1.48 LEAST.....	407
12.1.49 NVL, IF_NULL, IS_NULL.....	407
12.1.50 DECODE	407
12.1.51 CASE	408
12.1.52 MATCH AGAINST.....	408
12.1.53 MD5.....	409
12.1.54 SHA and SHA1.....	409
13. HFSQL CLIENT/SERVER	410
13.1 Overview	410
13.2 Implementing a Client/Server application	410
14. HFSQL CLIENT/SERVER FUNCTIONS	411



PART 1

**Code
editor**

10

DEVELOP 10 TIMES FASTER



PC SOFT

1. OVERVIEW

WinDev, WebDev and WinDev Mobile are development tools that allow you to create projects and to manage the elements created with the WinDev, WebDev and WinDev Mobile language: WLanguage.

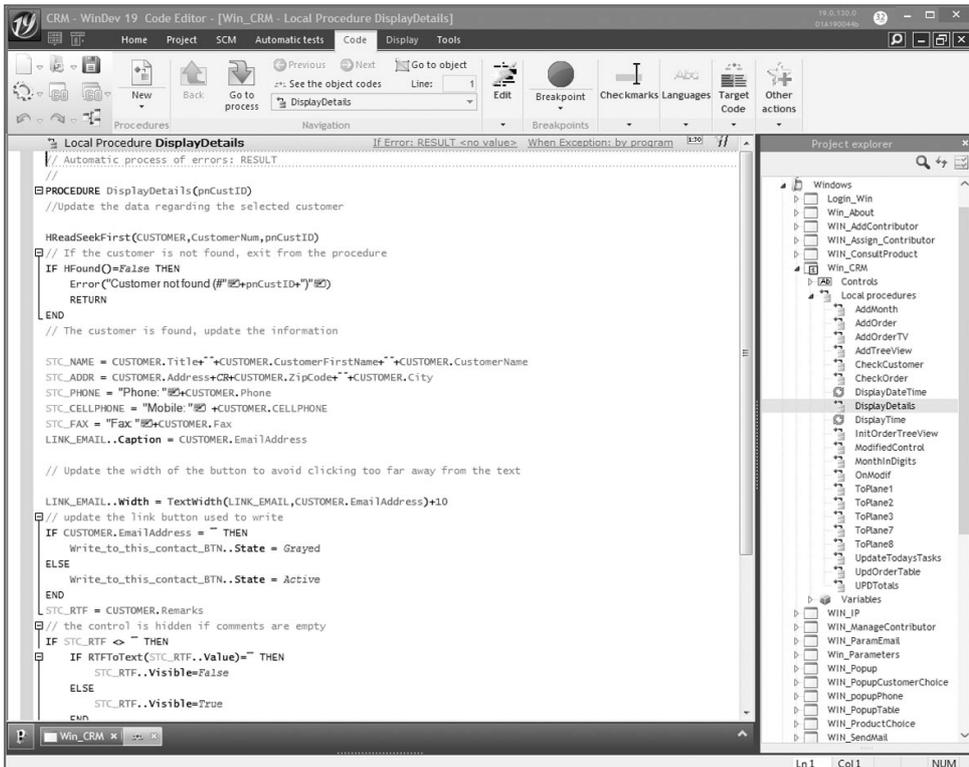
WLanguage is an integrated 5GL. The WLanguage functions allow you to create processes for each project, window, page and control.

These processes are entered in a dedicated source editor (also called code editor). This editor can be directly accessed from WinDev, WebDev or WinDev Mobile. It enables you to access all the processes of an object.

1.1 The code editor

The code editor allows you to enter all the processes in WLanguage (the programming language included in WinDev, WebDev and WinDev Mobile). It is used to enter the source code:

- of controls,
- of windows,
- of pages,
- of reports
- of local and global procedures,
- of classes and methods, ...



The different processes displayed in the code editor are separated by an horizontal bar containing the caption of the code. The corresponding WLanguage code must be entered below the bar.

A Ribbon menu

The options of the code editor are grouped in a menu that is shaped like a ribbon.



1.  WinDev/WebDev/WinDev Mobile button (top left). This button is used to display the "About" information for the current product.
2.  Displaying toolbars from previous versions (compatibility mode).
3.  Displaying drop-down menus from previous versions (compatibility mode).
4. Quick access buttons. These buttons correspond to the most often used options:



- New, Open,
- Save,
- Go of project,
- Go of window or page, ...

5. Ribbon panes. These panes are used to access the different options.



Several types of panes are visible:

- the "current" pane. The name of the pane is displayed with an orange line.
- the popup panes specific to the current element (the name of the pane is displayed in orange).
- the available panes (name of the pane in white).

6. Group of options. A group gathers the different options of the ribbon by theme.



7.  Group button. Button found at the extremity of a group used to perform a specific action (display the options of the current element, display the help, ...).

8. Ribbon options. Several types of options are available in the ribbon:

- Buttons to click
- Options to check.
- Button with arrow used to expand the options.



2 types of buttons with arrow are available:

- the buttons with arrow used to expand a menu.

- the buttons with arrow used either to expand a menu (click on the arrow), or to perform a default action (click on the button icon).

1.2 Features linked to the code input

This paragraph presents some interesting features of the code editor. See the online help for more details.

1.2.1 Code coloring

The source code entered is automatically colored to make the processes more readable. To modify the colors and fonts, use the "Coloring" tab in the options of code editor. To display the options of the code editor, on the "Home" pane, in the "Environment" group, expand "Options" and select "Options of code editor".

1.2.2 Automatic completion

The **automatic completion** proposes an assisted input of WLanguage code in your applications.

This completion is available for the WLanguage commands, the procedures, the variables, the data files, the items of data files, the windows, the controls, the type of variables, ...

This completion reduces the risk of typos and saves a lot of time: no need to search for the exact name of a global variable or control, it is automatically proposed!

The completion is used to:

- **fill the current input** by proposing a list of possible values (function name, property name, ...). All the values proposed in the list contain the word currently typed.
The content of the list is refreshed whenever a modification is performed.
For example, when typing the characters "OKA", all the functions, controls and so on, starting with these characters are suggested, ... containing these characters are proposed.
- **propose the value of the possible parameters** according to the WLanguage function used.
For example, when using the function named **HRead**, the first expected parameter corresponds to the name of a data file. The automatic completion will only propose the data files of the analysis associated with the current project.
- **select the combination of constants for a parameter.**
To do so, select the requested options (with the arrow keys) and press the SPACE key to validate these options.
All selected options are automatically included in the code during the validation (ENTER key).
- **start the code wizards.** These wizards are used to enter the different parameters of the function via

simple questions.

The <Wizard> option signals the presence of a wizard. To start the wizard, select <Wizard>.

- **start the existing resources for the function used:** example, code brick, ...
To start a resource, all you have to do is select it.

1.2.3 Code wizard

Several WLanguage functions include code wizards. These wizards help you type the different parameters of the function. Simply follow the instructions given by the different screens of the wizard to get the correct syntax. The use of these code wizards is optional.

For each function that includes a wizard, a list is opened when typing the function name. All you have to do is select <Wizard> from this list to start the corresponding code wizard.

1.2.4 Assisted input of functions

The following help mechanism simplifies the input of WLanguage functions in the code editor:

- **A visualization of the syntax currently typed via several tooltips.** This help is displayed when the name of the function is entirely typed. If several syntaxes are available for the current function, arrows appear in the different tooltips.
To modify the active syntax, press [Alt] + [Right Arrow] or [Alt] + [Left Arrow].
- **A tooltip displays the description of the current parameter (parameter tooltip).** This tooltip appears:
 - when the mouse cursor is located on the position of a parameter to type (the mouse cursor hovers the syntax in the tooltip).
 - if [Alt + F1] is used on a parameter that is already typed.
- **A tooltip displays the description of the result (result tooltip).** This tooltip appears:
 - when the mouse cursor is located on the name of the function (when typing a new function).
 - if [Alt + F1] is used on the name of a function that is already typed.

1.2.5 Help

The help can be accessed from the code editor by pressing the [F1] key. If a function is selected, the corresponding help is displayed. You have the ability to perform "cut-paste" operations from the help to the program.

This help is available:

- on Internet: the help pages of the version currently sold are updated on a regular basis. You benefit from the comments from the WinDev, WebDev, WinDev Mobile developer community.
- locally: an updated help is available for each new version or upgrade of the product.

1.2.6 Code history

The management of the history is used to:

- store the entire code of each modified process.
- restore an existing code that was deleted or modified thereafter.

To manage the code history of your project:

1. Display the code editor options: on the "Home" pane, in the "Environment" group, expand "Options" and select "Options of code editor".
2. In the "Code" tab, check "Save the history of code modifications".

This management of the history creates a "History" sub-directory in the directory of the project. This directory contains the following sub-directories:

- WDP: history of project processes
- WDW: history of window processes
- WDC: history of class modifications
- WDE: history of report processes
- WDR: history of query processes
- WDG: history of processes of set of procedures
- ...

In each sub-directory, ".FIC", ".NDX" and ".MMO" files corresponding to the name of windows, project, ... contain the history. These files can take a large amount of space after a little while. They can be:

- deleted in the Windows explorer directly.
- cleared. To do so, on the "Code" pane, in the "Other actions" group, expand "History" and select "Clear the history".

Caution: The code history cannot be displayed anymore if these files are deleted or cleared.

1.2.7 Code check

The code is checked in real time. The errors and warnings are immediately underlined when typing.

The "Compilation errors" pane is used to find out the type of problem that occurred.

Several types of messages can be displayed in this pane:

- the **warnings**, displayed in orange, notify you of potential code inconsistencies. These warnings do not block the execution of the process.
- the **errors**, displayed in red, point out errors in the code. These errors block the execution of the code.
- the **information**, displayed in black, explains the compiler choices or proposes potential advice for improving the code.
- the **GUI compilation errors**, displayed in red, present the potential problems found in the interfaces whenever the window or the page is saved or whenever the project is recompiled.

1.2.8 Automatic indent

When typing conditional statements (such as "Switch", "For" and "While" for example), the code is automatically indented to highlight the structure of the loops used in the program.

As soon as the first statement is typed, the "END" keyword is automatically inserted and the cursor is positioned **in the conditional statement**.

To automatically indent a section of code (code copy for example), on the "Code" pane, in the "Edit" group, click "Auto re-indent".

1.2.9 Translating the messages

When developing a multilingual application, WinDev gives you the ability to translate the different messages displayed.

The window for typing translated messages can be called from any message to translate. To do so:

- press [CTRL]+[T]
- on the "Code" pane, in the "Languages" group, expand "Translate the strings" and select "Translate the messages".

Each translated message corresponds to a "resource". The same "resource" can be used several times, in different processes.

An icon and a number are displayed on the right of the translated message. The number specifies the number of translations typed for the current message.

Note: The development of multilingual applications is presented in details in the online help.

1.2.10 Managing the breakpoints

When running the application test in the editor, the breakpoints are used to automatically start the debugger from a given line. See the online help for more details.

1.2.11 Inserting specific processes

For each element of the project, WinDev proposes:

- default processes (control initialization, button click, ...).
- specific processes (mouse hovering a control, right click on a window, ...). These special processes are displayed as icons in the element's code window. To insert a process specific to the use of the mouse or keyboard, click one of the available icons.

1.2.12 Other features

Programming charter

A programming charter is used to automatically prefix the name of all the variables and the name of all the project elements (window, report control, class, ...). This automatic prefixing is used to easily identify and/or find an element of a given type.

See the online help for more details.

Highlighting the modified code

The modified code lines are immediately identified by the presence of colored bars in front of the code lines.

Only the code modifications performed since the current element was opened are taken into account:

- the code lines preceded by a blue bar have been saved.
- the code lines preceded by an orange bar have not been saved yet.

When the element is closed, the colored bars disappear automatically.

Correcting the entered code

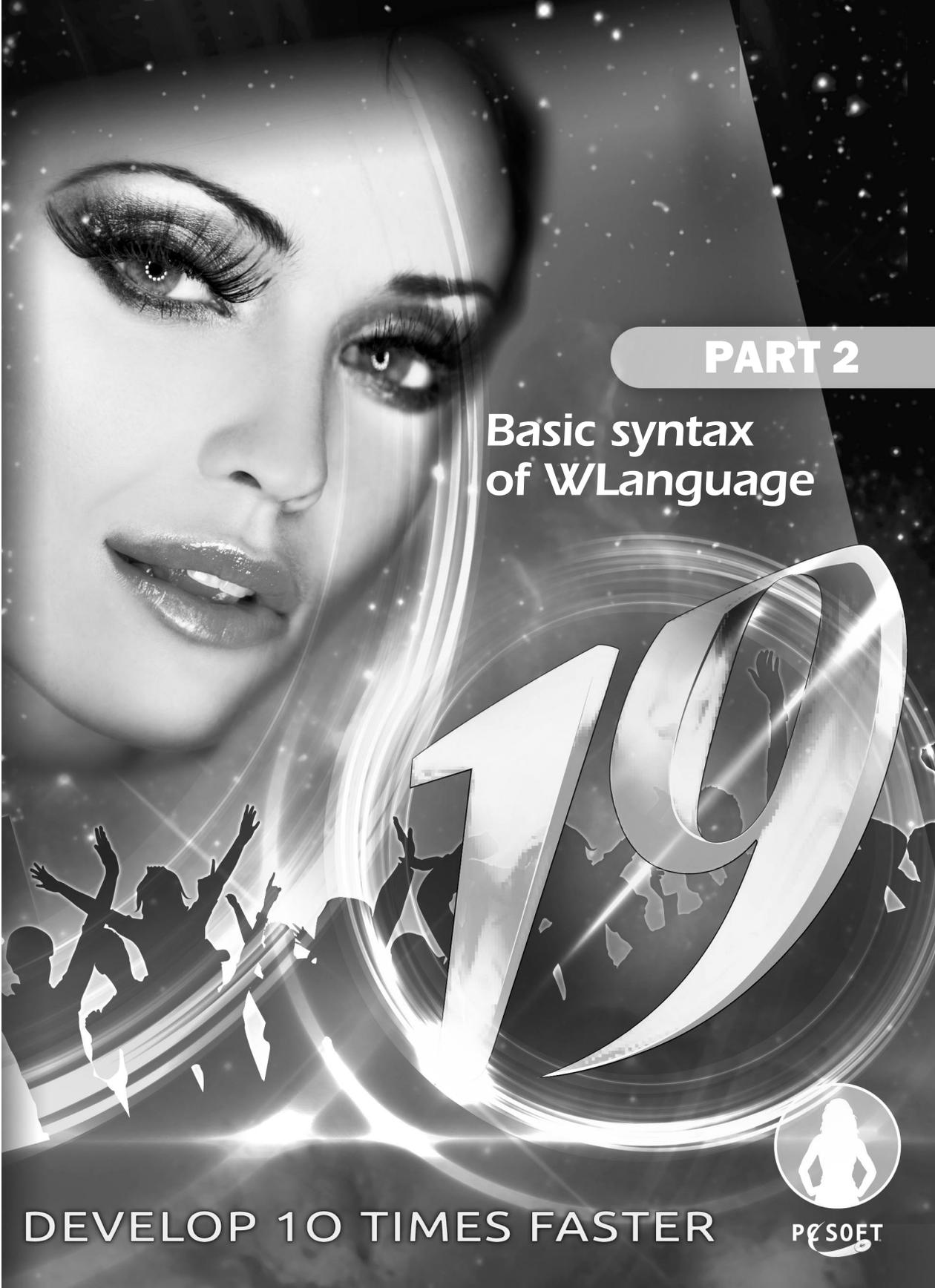
The code editor proposes several tools for correcting the code:

- **the automatic corrector of project elements:** when the name of a variable or project element is not recognized, the popup menu of the code editor now proposes a list of possible corrections. This list presents the name of the variables and/or project elements that may correspond.
- **the spelling checker** for the comments and the character strings.

Positioning marks

The positioning marks are used to "tag" specific code lines. These marks are used to quickly browse the code from a positioning mark to another one.

See the online help for more details.



PART 2

**Basic syntax
of WLanguage**

10

DEVELOP 10 TIMES FASTER



1. INTRODUCTION

WLanguage is an integrated 5GL. The WLanguage functions allow you to create processes for each project, window, page and control.

These processes are entered in a dedicated source

editor (also called code editor). This editor can be directly accessed from WinDev, WebDev or WinDev Mobile. It allows you to access all the processes of an object.

1.1 Some characteristics of WLanguage

- The WLanguage functions can be written in lowercase characters, in uppercase characters or in any combination of uppercase-lowercase characters. The code editor recognizes both. The code editor displays the various WLanguage keywords in different colors. This feature is very useful to check that no error was made when entering the names of functions.
- With WLanguage, you can practice event-driven programming without having to manage it. For each object created by WinDev, WebDev or WinDev Mobile, you have the ability to associate one or more processes that will be run on the action of a given event. You do not have anything to manage, you only have to enter each process. In WebDev, the processes can be entered:
 - in server code: the process will be run on the server.
 - in browser code: the process will be run in the browser displayed on the computer of the Web user (the WLanguage code entered in browser code is automatically translated into Javascript).For the edit controls, you have the ability to define:
 - a process that is run when initializing the control,
 - a process that is run when entering into the control,
 - a process that is run when exiting from the control,
 - a process that is run during the exit when the control is modified,
 - other processes linked to other events that you can be directly added in the code editor.
- WLanguage is made of keywords that represent:
 - functions,
 - properties,
 - preset constants,
 - statements for declaring variables,
 - statements for declaring functions and procedures,
 - keywords,
 - state variables, ...
- WLanguage supports the object paradigm.
- The WLanguage functions have a name that illustrates the use of the function.
- All the WLanguage keywords (functions, preset constants, properties, ...) also exist in French.
- A debugger is available for WLanguage. It can be called from the beginning of the project, by programming or when running the test of an element (window, page, ...) or the test of a project.

1.2 Programming in server code and in browser code

This chapter presents the syntax of WLanguage. Each WLanguage feature is:

- accessible in server code only: the feature will be run by the server,
- accessible in browser code only: the feature will be run by the browser on the computer of the Web user,
- accessible in server code and in browser code.

Notes:

- To program in server code, you must use WLanguage.
 - To program in browser code, you can use:
 - WLanguage that will be automatically translated into JavaScript,
 - JavaScript directly.
- In the code editor, bars of different color are used to differentiate between the server code and the browser code. By default, the following colors are used:
 - yellow for the server code,
 - green for the browser code written in WLanguage,
 - blue for the browser code written in JavaScript.
 - A process can be run on the server from a process performed in browser code.
Example:

```
// Browser code for modifying and  
exiting from a COMBO  
PageSubmit("", "BUTTON1")  
// Calls the server code of click on  
BUTTON1
```

2. THE VARIABLES

WLanguage proposes two types of variables:

- the simple types, that can be declared directly
- the advanced types, that include one or more variables of simple type.

2.1 The simples types

2.1.1 Principle

A variable is defined by its name and by its type.

The type of the variable defines the values that can be taken by the variable, its memory footprint and the available operations.

Reminder: Each type is identified by a WLanguage keyword. These keywords are reserved words.

2.1.2 Types of variables

The "simple" types of WLanguage are as follows:

- **Boolean:** type recommended for all the logical operations where only two alternatives are possible:

- True (value different from 0)
- False (value equal to 0)

Example: b is boolean

- **Integer:** type recommended for the calculations performed on integer values.

For the advanced calculations, WLanguage proposes different types of integers.

Example: b is int

- **Currency:** type recommended for the calculations performed on real values that require a high precision on decimal places, such as currency values.

A currency manages 24 significant digits (up to 18 digits for the integer part and up to 6 digits for the decimal part). The precision is precise to 6 decimals.

See "The currency type", page 41 for more details.

Example: MyCurrency is currency

- **Numeric:** type recommended for the calculations performed on real values that require a high precision on decimal places. A numeric manages 38 significant digits (up to 32 digits for the integer part and up to 6 digits for the decimal part).

Example: MyCurrency is numeric

- **Real:** type recommended for the simple calculations

performed on real values.

A real manages 15 significant digits but the precision of the decimals is not guaranteed. To perform precise calculations, use the "Currency" type.

For advanced calculations, WLanguage proposes different types of reals.

See "The real type", page 41 for more details.

Example: VAT is real

- **String:** type recommended to manage the characters and the character strings.

With this type, there is no need to declare the length of the string. This one can freely change when using the variable.

For an advanced management of character strings (mainly for the Windows APIs), WLanguage proposes different types of strings.

See "The String type", page 41 for more details

- **Buffer:** type recommended to write code that can be shared between WinDev and WinDev Mobile.

- **Date, Time, DateTime, Duration:** types recommended to manage the dates and times.

The "Date" type and the "Time" type allow you to easily handle the dates and times, and to manage the conversions almost automatically.

See "The date type", page 42 and "The time type", page 43 for more details

The "DateTime" type allows you to manage a date and a time at once. See "The DateTime type", page 43 for more details.

The "Duration" type allows you to easily manage the differences between times or dates. See "The Duration type", page 43 for more details.

- **Variant:** type recommended to store any other simple type of WLanguage.

The Variant type allows you to manage the NULL value. See "The Variant type", page 44 for more details.

2.1.3 Declaring a simple type

Declaring one or more variables

<Variable name> is <Variable type>

or

<Name of variable 1>, <Name of variable 2>

are <Variable type>.

```
CustomerName is string
Counter is int
Price is real
i,j,k are int
```

Declaring and initializing a variable

<Variable name> is <Variable type> = <Value>

OR

<Variable name> is <Variable type>

<Variable name> = <Value>

Note: When several variables of the same type are declared and initialized on the same line, only the last variable is initialized.

```
CustomerName is string
CustomerName = "Dupond"
Counter is int = 7
i,j,k are int = 21
// Only k is initialized to 21
```

Limit values for the types of data

If a variable is assigned with a value that exceeds the limits of its type:

- an error message is generated when running the test of a window, page or project.
- no error message signals the capacity overflow when the program is run. The value of the variable is incorrect.

To find out the limit values for the different types of data, see the "Limits.WL" file found in the "Personal\Extern" directory of WinDev/WebDev. This file can be opened by any text editor.

To simplify your tests, the limit values for the different data types have been assigned to constants in the "Limits.WL" file. To include these constants in your application, use the following code line in your project:

```
Extern "Limits.WL"
```

The "Limits.WL" file will be automatically sought in the "Personal\Extern" directory of WinDev/WebDev. There is no need to copy the "Limits.WL" file into the project directory.

Limitations: Declaring variables with the same name

Several variables with the same name cannot be used in the same process (regardless of the type of these variables).

If variables with the same name are declared in different processes of the project, the rule for variable scope applies (see page 65).

2.1.4 The different types of integer

WLanguage proposes different types of integers:

	Value included between
Integer	-2 x 109 and 2 x 109 (roughly)
1-byte integer	-128 and 127 (inclusive)
2-byte integer	-32.768 and 32.767 (inclusive)
4-byte integer	-2 x 109 and 2 x 109 (roughly)
8-byte integer	-9 x 1018 and 9 x 1018 (about)
Unsigned integer	0 and 4 x 109
Byte	0 and 255 (inclusive) This type is equivalent to the unsigned 1-byte integer.
Unsigned integer 1-byte	0 and 255 (inclusive)
Unsigned integer 2-byte	0 and 65.535 (inclusive)
Unsigned integer 4-byte	0 and 4 x 109 (roughly)
Unsigned integer 8-byte	0 and 18 x 1018
System integer	automatically adapts to the size generated by the system (4 bytes in 32 bits, 8 bytes in 64 bits)

Note: All the advanced types (different from the "Integer" type) are recommended when using the Windows APIs.

Default value

An "Integer" variable that is declared but not initialized is equal to 0.

2.1.5 The currency type

A currency variable is a real coded on 10 bytes. A currency can have up to 23 significant digits (17 digits for the integer part and 6 digits for the decimal part) and it can take a value included between -604 462 909 807 314 587.353 087 and +604 462 909 807 314 587.353 087.

The **currency** type uses a different coding system that does not trigger any rounding errors.

For a better precision, we recommend that you use the numeric type (38 significant digits).

For other calculations, we recommend that you use reals, which are faster.

Default value

A "Currency" variable that is declared but not initialized is equal to 0.

Binary coding of the reals

All the reals have the same number of significant digits (15).

If the result of a calculation performed on reals involves more than 15 significant digits, this result will be automatically rounded to 15 significant digits. The precision of the result will be less than the precision of a calculation performed with the currency type.

Calculations on currencies

All the calculations that use at least one **currency** will be performed in **currency** format. For a better precision of the result, all the operands will be automatically converted into **currency**.

External language

The currency type is not available in external language.

2.1.6 The numeric type

The numeric type is used to contain integer or decimal numbers by specifying if necessary:

- the number of digits in the integer part.
- the number of digits in the integer part and in the decimal part.

A numeric can be used:

- to declare a simple variable,
- as element of an array,
- as element of a composite variable,
- as element of a structure,
- as element of a class,

A variant can contain a numeric.

2.1.7 The real type

WLanguage proposes two types of reals:

- Real (8-byte real): A real can include up to 15 significant digits. The precision of the decimals is not guaranteed. To perform precise calculations, use the "Currency" type. Minimum value: 1.7×10^{-308} , Maximum value: $1.7 \times 10^{+308}$
- 4-byte real: A 4-byte real can include up to 6 significant digits. The precision of the decimals is not guaranteed. To perform precise calculations, use the "Currency" type. Minimum value: 3.4×10^{-38} , Maximum value: $3.4 \times 10^{+38}$.

Default value

A "Real" variable that is declared but not initialized is equal to 0.

Binary coding of the reals

All the reals contain the same number of significant digits (15).

If the result of a calculation performed on reals involves more than 15 significant digits, this result will be automatically rounded to 15 significant digits. To get a better precision, use the currency type.

2.1.8 The String type

WLanguage proposes different types of character strings.

The most common types of strings are as follows:

- Character: Character coded on 1, 2 or 4 bytes depending on the string management mode and the runtime platform.
- Ansi string: Dynamic size string containing only characters in ANSI format.
- Unicode string: dynamic size string containing only characters in UNICODE format.
- String: Dynamic size string.

The "string" type is specific to WLanguage. With this type, there is no need to declare the length of the string. This one can freely change when using the variable.

- Buffer: Binary memory zone. Used to write a code that can be shared between WinDev and WinDev Mobile regarding the operations performed on rough data. See "The Buffer type", page 42 for more details.

Other types of strings are also available:

- **String on:** Fixed-length string, ending with a binary 0 (like in C). The specified size corresponds to the maximum number of characters in the string.
This type of string is used to create compatible WinDev/WinDev Mobile code when calling APIs found on the two platforms.
- **ASCIIZ string on:** String ending with a binary 0 (like in C).
The size of an ASCIIZ string cannot exceed 2 GB. The length given to the ASCIIZ string must be equal to its current length plus 1 (for the binary zero).
Type not available in WinDev Mobile.
- **Fixed string on:** Fixed-size string.
The length of a fixed string cannot exceed 2 GB. The string is filled with:
 - 0 if the variable is not assigned yet
 - space characters if necessary if the variable is assigned (similar to the "string" type of Basic).
Type not available in WinDev Mobile.
- **Pascal string on:**
String preceded by a byte that specifies the length (like in Pascal). This byte is not accessible. For example, `String[1]` represents the first character of the string and not its length.
The length of a Pascal string cannot exceed 255 characters. The size given to the Pascal string must be equal to the size of the string.
Type not available in WinDev Mobile.
- **UNICODE string on:**
Fixed-size string containing characters in UNICODE format.

Notes:

- The "String on" type must be used to send input/output parameters to a Windows API.
- All the advanced types (different from the "String" type) are available for compatibility with the other programming languages (Turbo Pascal Windows, C, Visual Basic Windows, ...) and for the Windows APIs.

Default value

- A "String" or "String on" variable that is declared but not initialized corresponds to an empty string ("").
- A "Buffer" variable that is declared but not initialized is empty.
- A "Buffer on" variable that is declared but not initialized is filled with 0.

String type

WinDev Mobile and WinDev do not offer the same types of strings. See the online help for more details.

Passing a string in parameter to a procedure

A "String" variable can be passed in parameter to a procedure.

Caution: If the string is a "Fixed String", the space characters must be deleted. For example:

```
MyString is fixed string of 30
```

```
MyString = "WinDev is great"
// Delete the space characters
MyString = NoSpace(MyString)
CountLetter(MyString)
// CountLetter is procedure
```

2.1.9 The Buffer type

The buffer type corresponds to a binary memory zone. This type allows you to develop code that handles the binary format and to share this code between a WinDev application and a WinDev Mobile application.

The buffer type manages no specific end marker and it is used to store the binary 0.

Two types of variables are available:

- **Buffer:**
This type is used to handle a memory zone whose size is dynamic: it is automatically adapted to the content of the buffer.
- **Buffer on:**
This type is used to handle a memory zone whose size (in bytes) is defined during the compilation. This is an advanced type used to perform specific operations in memory, to use some Windows APIs.

2.1.10 The date type

The Date type enables you to easily handle the dates. This type allows you to manage the conversions almost automatically (*StringToDate*, *DateToString*, *IntegerToDate*, *DateToInteger*, ...).

This type can be used:

- to retrieve and handle the HFSQL items in Date format.
- to retrieve and handle the Date edit controls.
- in the WLanguage functions used to manage the dates.
- to perform calculations on dates.

Note: Several properties can be used with the Date type, so you can retrieve a piece of the date for instance.

Default value

By default, a Date variable is initialized with today's date (system date). To defined the default value assigned to the Date variables, use **DateTimeByDefault**.

```
StartDate is date = 1205
EndDate is date = "20011231"
```

```
// 12/31/2001
TodaysDate is date = Today()
```

The properties that can be used on the dates

See the online help to find out the properties that can be used on the dates.

```
StartDate is date = "20011201"
StartDate..Year +=5 // Adds 5 years
// Modifies the month
StartDate..Month = 5
// Calculates the end (30 days
later)
EndDate = StartDate
EndDate..Month++
EndDate..Day--
```

Range of dates

The Date type allows you to manage the dates included between 01/01/0001 and 12/31/9999.

2.1.11 The time type

The Time type is used to manage times. This type allows you to manage the conversions almost automatically (**TimeToString**, **TimeToInteger**, ...)

This type can be used:

- to retrieve and handle the HFSQL items in Time format,
- to retrieve and handle the edit controls in Time format,
- in the WLanguage functions used to manage times,
- to perform calculations on the times (difference, addition, ...).

Properties that can be used on times:

See the online help to find out the properties that can be used on the times.

```
StartTime is Time="20011201"
// Adds 5 hours
StartTime..Hour +=5
// Modifies the number of minutes
StartTime..Minutes = 5
```

Limits:

The Time type can be used to manage the hours found between 00:00 and 23:59. The precision is up to the millisecond.

2.1.12 The DateTime type

The DateTime type allows you to manage a date and a time as a single object. The DateTime type can be used to perform calculations (subtraction, addition, ...) on the dates, times and durations.

```
StartDate is DateTime
StartDate = "200112311524"
// 12/31/2001 at 15:24
```

Properties that can be used on the DateTime variables

See the online help to find out the properties that can be used on the DateTime variables.

```
StartDate is DateTime
StartDate = "200112011530"
StartDate..Year +=5 //Adds 5 years
// Modifies the month
StartDate..Month = 5
// Displays the date
Info("Date" + StartDate..Date)
// Displays the time
Info("Time "+ ...
StartDate..Time)
```

Interval of data

The DateTime type can be used to manage data between 01/01/0001 at 00:00 and 12/31/9999 at 23:59.

2.1.13 The Duration type

The Duration type allows you to easily handle intervals of dates and times. The Duration type can be used to perform calculations (difference, addition, ...).

```
StartTime is Time = "1330"
EndTime is time = "1730"
ConferenceDuration is Duration = ...
EndTime - StartTime
```

Properties available for durations

See the online help to find out the properties that can be used on the Durations.

Supported values

The duration type is used to manage the durations included between plus and minus 2 billion days. The precision is up to the millisecond.

2.1.14 The Variant type

The Variant type is used to:

- store any value of simple type: boolean, numeric, string, date, time, ...
- manage the interactions with the ActiveX objects and the OLE Automation programming
- handle the NULL value in WLanguage

Assigning a Variant type

A Variant variable can be assigned with:

- any literal value
- the content of a variable
- the content of a simple control
- the content of an item

```
nValue is variant = 10
nValue = Edit1
nValue = Customer.Name
```

Variant type and NULL

To specify that a Variant variable contains no value, use the NULL constant.

Note:

- for a variant type, NULL means "Not assigned"

```
vVal is variant
IF vVal = Null THEN ...
// returns True because not assigned
vVal = 0
IF vVal = Null THEN ...
```

```
// returns False because variant
// assigned with an integer whose
// value is 0
vVal = 5
IF vVal = Null THEN ...
// returns False because variant
// assigned
// with an integer whose value is 5
```

- for a numeric type, NULL means "equal to 0".

```
nVal is int
IF vVal = Null THEN ...
// the test returns True because
// nVal=0
nVal = 5
IF vVal = Null THEN ...
// the test returns False because
// nVal=5
```

Type of a variant

The type of a variant is returned by `..Type`. Type is used to find out the type of the value stored in the variable.

Notes:

- **VariantConvert** is used to convert the type of the value stored in a Variant variable.
- **TypeVar** is used to find out the type of a variable (Variant type for a Variant variable).

2.2 Operations available for the dates, times and durations, ...

The following operations are available for the following types of variables: Date, Time, DateTime and Duration:

- addition
- subtraction

- comparison

See the online help for more details.

Note: Several WLanguage functions are used to handle the dates and times (see the functions for managing dates and times in the online help).

2.3 Managing the NULL value

The Null keyword can be used in different ways in WLanguage:

- to specify that a parameter must be ignored in a query
- to specify that a variant variable contains no value

- to compare a value to 0
- in some WLanguage functions, to specify that the parameter must be ignored
- to check whether a dynamic object is allocated or not

2.3.1 Null and the queries

Ignoring the parameters: Null in HExecuteQuery

When running a query with parameters via the *HExecuteQuery* function, all the query's parameters do not necessarily have to be specified. The query conditions that use some parameters that are not specified or whose value is Null will be ignored.

Example: Let's take the "CustomerLastnameFirstname" query whose SQL code is as follows:

```
SELECT * FROM CUSTOMER
WHERE LASTNAME = {Param1}
AND FIRSTNAME = {Param2}
```

• Case #1: Both parameters are specified:

```
HExecuteQuery(...
    CustomerLastNameFirstName, ...
    hQueryDefault, "Smith", "John")
```

will run the query:

```
SELECT * FROM CUSTOMER
WHERE LASTNAME= 'Smith'
AND FIRSTNAME = 'John'
```

• Case #2: Only the last name is specified:

```
HExecuteQuery(...
    CustomerLastNameFirstName, ...
    hQueryDefault, "Smith")
```

will run the query:

```
SELECT * FROM CUSTOMER WHERE LASTNAME='Smith'
```

• Case #3: Only the first name is specified:

```
sName is Variant = Null
// use the mandatory variant
// for the Null variable (not filled)
HExecuteQuery(...
    CustomerLastNameFirstName, ...
    hQueryDefault, sName, "John")
```

Or

```
HExecuteQuery(...
    CustomerLastNameFirstName, ...
    hQueryDefault, Null, "John")
```

will run the query:

```
SELECT * FROM CUSTOMER
WHERE FIRSTNAME='John'
```

Query parameters coming from an edit control: Null if empty

In order for your query to be run even if no value was entered by the user, check "NULL if empty" for the edit controls ("Details" tab of the control description).

When this option is checked, if the control is empty, the value of the parameter passed to the query will correspond to the NULL constant. No error will occur when running the query: the conditions that depend on this parameter will be ignored.

For example, the "Ex1" query corresponds to the following SQL code:

```
SELECT NameOfItems
FROM NameOfFiles
WHERE Item = {Param1}
```

The "Ex1" query is run in the "Btn_OK" button by *HExecuteQuery*. The WLanguage code used is as follows:

```
HExecuteQuery, (...
    Ex1, hQueryDefault, EDT_Edit1)
```

In this code, EDT_Edit1 corresponds to the control in which the user must enter the query parameter.

In this example, the table below describes the use of "NULL if empty":

Value entered in the EDT_Edit1 control	"NULL if empty" for the EditControl1 control	SQL code run
No value	Option checked	SELECT Item-Names FROM NameOfFiles
No value	Option unchecked	SELECT Item-Names FROM NameOfFiles WHERE Item = ''
A value entered	Option checked or unchecked	SELECT Item-Names FROM NameOfFiles WHERE Item='InputValue'

2.3.2 Null and the variants

To specify that a Variant variable contains no value, use the NULL constant.

Notes:

- for a variant type, NULL means "Not assigned"
- for a numeric type, NULL means "equal to 0" (see below)

```
vVal is variant
IF vVal = Null THEN ...
// returns True because the variant
// is not assigned
vVal = 0
IF vVal = Null THEN ...
// returns False because the variant
// is assigned an integer with 0 for
// value
vVal = 5
IF vVal = Null THEN ...
// returns False because the variant
// is assigned an integer with 5 for
// value
```

2.3.3 Null and the numeric values

Used with numeric values, Null is used to compare a value to 0. The equality operators and the comparison operators can be used (= and <>).

Notes:

- for a variant type, NULL means "Not assigned" (see above)
- for a numeric type, NULL means "equal to 0"

```
nVal is int
IF vVal = Null THEN ...
// the test returns True because
// nVal=0
```

```
nVal = 5
IF vVal = Null THEN ...
// the test returns False because
// nVal=5
```

2.3.4 Null and the WLanguage functions

Some WLanguage functions accept Null in parameter to specify that the parameter takes no value.

Some examples:

- **TreeAdd, TreeInsert:** Null is used to avoid displaying the images for the different levels of added elements.
- **TreeListItem:** Null is used to list the child elements from the root of the treeview.
- **TreeModify:** Null is used to avoid modifying the images defined by **TreeAdd**
- **InitWrite:** Null is used to delete a keyword or a section from the INI file

2.3.5 Null and the dynamic objects

For the dynamic objects (class, structure, array, automation object, ...), Null is used to find out whether the object is allocated or not.

For example:

```
// Declare a dynamic automation
// object
MyDynamicObject is object ...
dynamic automation ...
IF MyDynamicObject = Null THEN
// Create a dynamic automation
// object
MyDynamicObject = ...
new object ...
dynamic automation MyServer
END
```

2.4 Constants

The constants are language elements whose value is defined once and for all. This value cannot be modified during the execution of the program.

Syntax: Declaring one or more constants

```
CONSTANT
<Name of constant 1> = <Value 1>
<Name of constant 2> = <Value 2>
END
```

```
CONSTANT
VATRate = 19.6
END
```

The constants must be declared in:

- **the initialization code of the project** to be used in all the processes of the project (code of the project, code of the windows/pages, code of the control, procedures, ...).
- **the code for declaring the global variables of a window/page** to be used in all the processes of the window/page (code of the window/page, code of the controls found in the window/page, local procedures, ...).

- **the opening code of a report** in order to be used in all the processes of the report (code of the report, code of controls found in the report, local procedures, ...).
- **the declaration code of a class** in order to be used in the class.
 - To access this constant from a method of the class, use the syntax "<Constant name>".
 - To access this constant from a code external to the class, use the following syntax "<Class name>:<Constant name>".

Note: A constant is always global:

- to a project.
- to a window or to a page.
- to a report.
- to a class.

Declaring constants with the same name

Several elements (variables or constants) with the same name cannot be declared in the same process.

If a constant and a variable have the same name in the same project:

- the **variable** will be used in the process where this variable is declared and in all the linked processes. For example, the variable is declared in the click code of a button. When a click is performed on this button, this variable is passed in parameter to a local procedure. This variable will be used in this local procedure.
- the **constant** will be used in all the other processes.

If a constant "global" to the project and a constant "global" to a window have the same name:

- the **constant "global" to the window** will be used in all the processes of the window and window controls, as well as in the "local" procedures of the window.
- the **constant "global" to the project** will be used in all the other processes.

Declaring a constant member

A constant member cannot be declared in:

- a structure.
- a composite variable.

2.5 The advanced types

2.5.1 Simple Array

An array is a structured type that is used to group a set of elements of the same type. Each array element can be directly accessed by its subscript.

An array can be resized during the program execution by **Dimension**.

Note: The Array keyword is used to define a "simple" array. Other types of "advanced" arrays are available:

- **Dynamic array:** Array allocated upon request.
- **Fixed array:** Fixed-size array (recommended for the Windows APIs).

Syntax

Declaring a "simple" array

```
<Array Name> is array of <Dimension 1>
[by <Dimension 2>]...[by <Dimension 10>]
<Type of Array Elements>
```

Or

```
<Array name> is array of <Dimension 1>
[,<Dimension 2>]...[,<Dimension 10>]
<Type of Array Elements>
```

```
CustomerArray is array ...
  of 5 by 7 by 3 int
// Equivalent to :
CustomerArray is array of 5,7,3 int
// Resize the array
Dimension(CustomerArray,7,9,4)
```

Referring to a "simple" array

- Referring to an element in a one-dimensional array: <Array name>[Subscript1]
- Referring to an element in a two-dimensional array:


```
<Array name>[Subscript1, Subscript2]
OR
<Array name>[Subscript1][Subscript2]
```
- Referring to an element in an array with N dimensions:


```
<Array name>[Subscript1,...,SubscriptN]
OR
<Array name>[Subscript1]...[SubscriptN]
```

- Passing an array in parameter to a procedure:
<Procedure name>(<Array name>)

Note: An array cannot be handled as a whole. For example, an array cannot be assigned to another array.

```
CustomerArray[2,5,3] = 47
// Equivalent to :
CustomerArray[2][5][3] = 47
```

Passing a "simple" array in parameter to a procedure

An array can be passed in parameter to a procedure. To do so, use the following syntax:

<Procedure name>(<Array name>)

```
SuppArray is array ...
    of 10 by 50 strings
// Call the DisplayArray procedure
DisplayArray(SuppArray)
```

Dimension of a "simple" array

Dimension is used to:

- find out the total number of elements in an array.
- resize an array.

Caution: **Dimension** cannot be used to modify the number of dimensions in an array.

Elements of a "simple" array

An array can be made of classes only if these classes have a constructor without parameter (or with optional parameters).

An array cannot include:

- composite variables.
- arrays.

Declaring variables with the same name

Several variables with the same name cannot be used in the same process (regardless of the type of these variables).

If variables with the same name are declared in different processes of the project, the rule for variable scope applies.

WLanguage functions and simple arrays

Several WLanguage functions allow you to handle the simple arrays. You have the ability to perform sorts, searches, ... See the online help for more details.

Arrays in the classes

When copying instances of classes, all the members of the class are copied into the new instance except for the arrays. Therefore, if the value of an

array member is modified, this value is modified in all the instances.

To get independent arrays in all the instances of classes, a local array must be declared as follows:

```
SystemClass is class
    aDefaultArray is array ...
        local of 1 int
END
```

2.5.2 Dynamic array

A dynamic array is an "advanced" type of array: the dimension of this array are allocated upon request, during the program execution. In most cases, a "simple" array is sufficient.

Reminder: An array is a structured type that is used to group a set of elements of the same type. Each array element can be directly accessed by its subscript.

We advise you to use:

- A **dynamic array** or a "simple" array when the size of the array must be modified during the program execution.
- A fixed array for the Windows APIs.
- An associative array to store elements indexed on any type of information.

Syntax

Declaring a dynamic array

<Array name> **is dynamic array**

```
CustomerArray is array dynamic
```

Allocating a dynamic array

<Name of dynamic array> =

new array dynamic of <Dimension 1>

[by <Dimension 2> ... [by <Dimension 10>]]

<Type of Array Elements>

OR

<Name of dynamic array> =

new array dynamic of <Dimension 1>

[,<Dimension 2> ... [,<Dimension 10>]]

<Type of Array Elements>

```
CustomerArray is array dynamic
CustomerArray = new ...
dynamic array of 4 by 7 int
// Equivalent to :
CustomerArray = new ...
dynamic array of 4, 7 int
```

Referring to a dynamic array

To refer to a dynamic array, this array must be allocated.

- Referring to an element in a one-dimensional array: `<Array name>[Subscript1]`
- Referring to an element in a two-dimensional array:
`<Array name>[Subscript1, Subscript2]`
OR
`<Array name>[Subscript1][Subscript2]`
- Referring to an element in an array with N dimensions:
`<Array name>[Subscript1,...,SubscriptN]`
OR
`<Array name>[Subscript1]...[SubscriptN]`
- Passing an array in parameter to a procedure:
`<Procedure name>(<Array name>)`

Note: An array cannot be handled as a whole. For example, an array cannot be assigned to another array.

```
CustomerArray [2,5,3] = 47
// Equivalent to :
CustomerArray [2] [5] [3] = 47
```

Freeing a dynamic array (optional)

A dynamic array is automatically freed at the end of the lifetime of the variable (when the window is closed for example) or when allocating new dimensions.

To explicitly free a dynamic array, use the following syntax:

Delete <Name of dynamic array>

Passing a dynamic array in parameter to a procedure

A dynamic array can be passed in parameter to a procedure. To do so, use the following syntax:

```
<Procedure name>(<Array name>)
  SuppArray is array...
    dynamic
  SuppArray = new array...
    10 by 50 strings
  // Call the DisplayArray procedure
  DisplayArray(SuppArray)
```

Declaring a dynamic array member

A "dynamic array" member can be declared in:

- a structure,
- a composite variable,
- a class.

The dynamic array must be allocated after the declaration of the structure, composite variable or class.

```
// Declare a structure
Struct is structure
  x1 is int
  x2 is dynamic array
END
// Declare a structure variable
MyStruct is struct
// Allocate the array
x2 = new dynamic array ...
  of 4,7 int
```

Dimension of a dynamic array

Dimension is used to:

- find out the number of elements in a dynamic array.
- resize a dynamic array.

Caution: **Dimension** cannot be used to modify the number of dimensions in a dynamic array.

Elements of a dynamic array

A dynamic array can include classes only if these classes have a constructor without parameter (or with optional parameters).

A dynamic array cannot include:

- composite variables.
- arrays.

Declaring variables with the same name

Several variables with the same name cannot be used in the same process (regardless of the type of these variables).

If variables with the same name are declared in different processes of the project, the rule for variable scope applies.

WLanguage functions and dynamic arrays

Several WLanguage functions can be used to handle the dynamic arrays. You have the ability to perform sorts, searches, ... See the online help for more details.

2.5.3 Fixed array

A fixed array is an "advanced" type of array: the dimensions of this array are defined during the compilation and they cannot be modified.

The dimensions of a fixed array are defined during the compilation, only if the dimensions of this array correspond to:

- an integer.
- a constant that was created beforehand.

Otherwise, a WLanguage error occurs during the compilation of the project.

Reminder: An array is a structured type that is used to group a set of elements of the same type. Each array element can be directly accessed by its subscript.

We advise you to use:

- a **fixed array** to pass an array in parameter to Windows API functions.
- a **dynamic array** or a "simple" array when the array must be resized during the program execution.
- an **associative array** to store elements indexed on any type of information.

Syntax

Declaring a fixed array

```
<Array name> is fixed array of
    <Dimension 1> [by <Dimension 2> ...
    [by <Dimension 10>]]
    <Type of Array Elements>
```

OR

```
<Array name> is fixed array of
<Dimension 1> [, <Dimension 2> ...
[, <Dimension 10>]] <Type of array elements>
```

```
CustomerArray is fixed array...
    of 5 by 7 by 3 int
// Equivalent to :
CustomerArray is fixed array...
    of 5,7,3 int
```

Referring to a fixed array

- Referring to an element in a one-dimensional array: <Array name>[Subscript1]
- Referring to an element in a two-dimensional array:


```
<Array name>[Subscript1, Subscript2]
OR
<Array name>[Subscript1][Subscript2]
```
- Referring to an element in an array with N dimensions:


```
<Array name>[Subscript1, ... , SubscriptN]
OR
<Array name>[Subscript1]...[SubscriptN]
```
- Passing an array in parameter to a procedure:


```
<Procedure name>(<Array name>)
```

Note: An array cannot be handle as a whole. For example, an array cannot be assigned to another array.

```
CustomerArray[2,5,3] = 47
// Equivalent to :
CustomerArray[2][5][3] = 47
```

Passing a fixed array in parameter to a procedure

A fixed array can be passed in parameter to a procedure. To do so, use the following syntax:

```
<Procedure name>(<Array name>)
```

```
SuppArray is fixed array ...
    of 10 by 50 strings
// Call the DisplayArray procedure
DisplayArray(SuppArray)
```

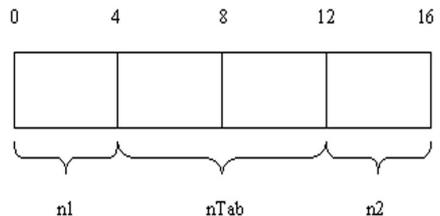
Declaring a fixed array member

A "fixed array" member can be declared in:

- a **class**. This fixed array is directly allocated in the memory zone of this class.
- a **composite variable**. This fixed array is directly allocated in the memory zone of this composite variable.
- a **structure** <Structure name>. This fixed array is directly allocated in the memory zone of each <Structure name> variable.

```
Struct is structure
    n1 is int
    nArray is fixed array of 2 int
    n2 is int
END
MyStructure is Struct
```

Representing the memory zone of "MyStructure":



This memory representation is compatible with the Windows APIs. Therefore, a fixed-size array can be transmitted to a function of the Windows APIs.

Dimension of a fixed array

Dimension is used to find out the number of elements in a fixed array.

Reminder: A fixed array cannot be resized.

Elements of a fixed array

A fixed array can include objects. An object represents the instantiation of a class. If the class has a constructor, this constructor must have no parameter (or optional parameters).

A fixed array cannot include:

- composite variables.
- arrays.

Declaring variables with the same name

Several variables with the same name cannot be used in the same process (regardless of the type of these variables). If variables with the same name are declared in different processes of the project, the rule for variable scope applies.

2.5.4 Associative array

An associative array is an "advanced" type of array; it is used to group a set of elements of the same type. Each array element is indexed on any type of information (and not only on a numeric subscript like in the other types of arrays).

Note: This type of array provides quick access to any element, with some options for strings, but it cannot be used for sorting (especially, no sort on the key).

Syntax

Declaring an associative array

```
<Array name> is array associative
of <Type>
```

```
// Fills an associative array with
// the sizes of files
aaFileSize is array ...
// associative of int
aaFileSize[...
    "File1.txt"] = ...
    fSize("File1.txt")
aaFileSize["File2.txt"] = ...
    fSize("File2.txt")
// Retrieves the size of a file
Trace(aaFileSize["File2.txt";])
```

Note: An advanced syntax is also available. See the online help (keyword: "Associative array") for more details.

2.5.5 Composite variable

A composite variable includes different types of elements (members).

Note: If you are using custom variables, we advise you to declare:

- a **composite variable** if a single variable of this type is used in your project.
- a **structure** type, if several variables of this type are used in different processes of the project.

Syntax

Declaring a composite variable

```
<Variable name> is composed of
    <Members of the composite variable>
END
```

```
ProductRef is composed of
    SCode is int
    PdtCode is fixed string on 10
END
```

Handling a member of a composite variable

```
<Variable name>.<Member name>
```

```
ProductRef.SCode = 7
ProductRef.PdtCode = "Screen"
```

Handling a composite variable

A composite variable can be used without referencing its members.

```
CompositeVar is composed of
    Member1 is fixed string on 8
    Member2 is fixed string on 12
END
Str = CompositeVar
// Str is built by concatenating
// all the members of the composite
// key
// CompositeVar = Str
// Str is transferred to the
// composite variable
```

Type of the members in a composite variable

The members of a composite variable can have the following types:

- Integer (or selected among the advanced integers).
 - Real (or selected among the advanced reals).
 - Currency.
 - Fixed string, ASCIIZ string or Pascal string.
 - Array ("simple" array, dynamic array or fixed array).
 - Automation or Dynamic Automation.
 - Structure.
 - Variant.
 - Class. This class must have a constructor without parameter (or with optional parameters).
- Any other type of data (string, constant, ...) is forbidden.

Declaring an array member in a composite variable

Declaring a "simple array" member:

The array dimensions are fixed when the array is declared. The array dimensions can be modified by **Dimension**.

Declaring a dynamic array member:

The array dimensions must be defined (which means that the array must be allocated) before using the array..

```
// Declare a composite variable
MyCompVar is composed of
  x1 is int
  x2 is dynamic array
END
// Allocate the array
MyCompVar.x2 = new array ...
                dynamic of 4,7 int
// Initialize the array
MyCompVar.x2[1,1] = 25
```

Limits of a composite variable

You cannot:

- directly assign a composite variable to another composite variable: each member must be assigned one by one.
- compare two composite variables.
- pass a composite variable in parameter to a procedure.
- define an inheritance in a composite variable.
- restrict the access to a member of a composite variable: all the members of a composite variable must be public.

Which "advanced" types can be member?

An "advanced" variable can be member of another "advanced" variable. The table below presents the different combinations:

Can be a member of	Composite variable	Structure	Class	Array
Composite variable	No	No	No	No
Structure	Yes	Yes	Yes	Yes
Class	Yes	Yes	Yes	Yes
Array	Yes	Yes	Yes	No

Declaring variables with the same name

You cannot declare:

- two variables with the same name (regardless the type of these variables) in a process.
- two members with the same name in the same composite variable.

If variables with the same name are declared in different processes of the project, the rule for variable scope applies.

2.5.6 Structure

A structure is a custom type of data. A structure groups different types of elements.

Note: If you are using custom variables, we advise you to declare:

- a **structure** type, if several variables of this type are used in different processes of the project.
- a **composite variable** if a single variable of this type is used in your project.

Syntax

Declare a structure

```
<Structure name> is structure
  <Structure members>
```

END

```
ProductRef is structure
  SCode is int
  PdtCode is fixed string on 10
END
```

Declare a structure variable

```
<Variable name> is <Structure>
```

```
ProductRef is structure
  SCode is int
  PdtCode is fixed string on 10
END
Armchair is ProductRef
```

Handling a member of a structure variable

```
<Name of structure variable>:<Name of structure member>
```

```
ProductRef is structure
  SCode is int
  PdtCode is fixed string on 10
END
Armchair is ProductRef
Armchair:SCode = 7
Armchair:PdtCode = "Furniture"
```

Declaring the structures

If a structure named <Structure name> is declared:

- **in the code of the project**, you will be able to declare a <Structure name> variable in the entire project.
- **in the declaration code of the global variables of a window/page** you will be able to declare a <Structure name> variable in this window/page, in the controls of this window/page and in the associated local procedures.
- **in the opening code of a report**, you will be able to declare a <Structure name> variable in this report, in the controls of this reports and in the associated local procedures.
- **in the code for class declaration**, you will be able to declare a <Structure name> variable in this class and in the methods of this class.
- **in a process**, you will be able to declare a <Structure name> variable in this process.

Type of structure members

The members of a structure can have the following types:

- Integer (or selected among the advanced integers).
- Real (or selected among the advanced reals).
- Currency.
- String (or selected among the advanced strings).
- Array ("simple" array, dynamic array or fixed array).
- Automation or Dynamic Automation.
- Object (class instantiation). The class can have a constructor without parameter (or with optional parameters).
- Variant.

Any other type of data (composite variable, constant, ...) is forbidden.

Declaring an array member in a structure

Declaring a "simple array" member:

The array dimensions are fixed when the array is declared. The array dimensions can be modified by **Dimension**.

Declaring a dynamic array member:

The array dimensions must be defined (which

means that the array must be allocated) before using the array.

```
// Declare a structure
Struct is structure
  x1 is int
  x2 is dynamic array
END
// Declare a structure variable
MyStruct is struct
// Allocate the array
MyStruct.x2 = new dynamic array of
4,7 int
// Use the array
MyStruct.x2[1,1] = 25
```

Which "advanced" types can be member?

An "advanced" variable can be member of another "advanced" variable.

The table below presents the different combinations:

Can be a member of	Variable composite	Structure	Class	Array
Variable composite	No	No	No	No
Structure	Yes	Yes	Yes	Yes
Class	Yes	Yes	Yes	Yes
Array	Yes	Yes	Yes	No

Passing a structure in parameter to a procedure

A "structure" variable can be passed in parameter to a procedure.

To do so, use the following syntax:

<Procedure name>(<Name of structure variable>)

```
ProductRef is structure
  SCode is int
  PdtCode is fixed string on 10
END
Armchair is ProductRef
// Call the DisplayProduct procedure
DisplayProduct(Armchair)
```

Limits of a structure

The inheritance of structure is not allowed: a structure cannot inherit from another structure.

The access to a structure member cannot be restricted: all the members of a structure are public.

Declaring variables with the same name

You cannot declare:

- two variables with the same name (regardless the type of these variables) in a process.
- two members with the same name in the same structure.

If variables with the same name are declared in different processes of the project, the rule for variable scope applies.

2.5.7 Dynamic structure

A structure can be allocated dynamically: we talk of dynamic instantiation of the structure.

The dynamic instantiation of structure is used to create a structure at a given time and to free this structure when it is no longer used.

To instantiate a structure, you must:

1. declare a dynamic structure
2. instantiate a structure

Note: the structure is automatically freed when it is no longer used. However, you can force the destruction of the structure.

```
File is structure
  Name is String
  Extension is String
  Directory is String
End
SourceFile is File dynamic
//...
SourceFile = New File
// process on the object
// ...
// free the object
Delete SourceFile
```

Declaring a dynamic structure

`<VariableName> is <StructureName> dynamic`

The important parameters of this syntax are as follows:

- `<VariableName>`: Name identifying the variable of the structure.
- `<StructureName>`: Name identifying the structure, defined when creating the structure.

Instantiating a dynamic structure

```
<VariableName> = ...
  new <StructureName> [<Parameters>])
```

The important parameters of this syntax are as follows:

- `<VariableName>`: Name identifying the variable of the structure.
- `<StructureName>`: Name identifying the structure, defined when creating the structure
- `<Parameters>`: Optional parameters of the constructor.

Freeing a dynamic structure

`Delete <StructureName>`

where

`<StructureName>`: Name identifying the instance of the structure.

2.5.8 Automation object

The **Automation** keyword is used to declare an automation object. This object is allocated during the declaration.

Note: You also have the ability to use a dynamic automation object. It is allocated upon request, during the program execution.

Syntax

Declaring an automation object

`<Name of automation object> is automation object <Name of automation server>`

```
MyAutomationObject ...
  is automation object MyServer
```

Calling a method of automation object

`<Name of automation object>><Method name>(<Parameters>])`

Note: The list of methods that can be used depends on the server. See the documentation about the server for more details.

```
MyAutomationObject>>...
  OpenFile (DocName)
```

Lifespan of an automation object

The automation object is created during its declaration.

The automation object is automatically destroyed at the end of the process containing its declaration.

An automation object declared "global" in the code:

- **for window initialization** will be destroyed at the end of the closing process of the window.
- **for project initialization** will be destroyed at the end of the closing process of the first project window.

Allocating an automation object

When allocating an automation object, the automation server is automatically started.

To allocate an automation object with an existing instance of an automation server, use **GetActiveObject**.

Passing parameters to a method

The methods of automation servers can accept one or more parameters.

When calling a method, the sequence of parameters must be respected (see the documentation about the server for more details).

For some automation servers, some parameters of methods are not valued: only the presence of the parameter is important, not its value.

For example, the "EditReplace" method of Word Basic accepts 10 parameters, the last one (ReplaceAll) being not valued.

In WLanguage, any ordinary value can be passed to non-valued parameters.

The methods of some automation servers can accept optional parameters located anywhere in the list of parameters.

In WLanguage, the optional parameters must necessarily be found after the mandatory parameters.

To give any ordinary value to an optional parameter, assign the * character to the optional parameter. On the contrary, this character will not be required for the optional parameters found after the last mandatory parameter.

Declaring variables with the same name

Several variables with the same name cannot be used in the same process (regardless of the type of these variables).

If variables with the same name are declared in different processes of the project, the rule for variable scope applies.

2.5.9 Dynamic Automation object

A dynamic automation object is an automation object allocated upon request, during the program execution.

Syntax

Declaring a dynamic automation object

<Name of dynamic automation object> is **dynamic automation object**

```
// Declaration
MyDynamicObject...
    is dynamic automation object
```

Allocating a dynamic automation object

<Name of dynamic automation object> = **new dynamic automation object** *<Name of automation server>*

```
// Create a dynamic automation object
MyDynamicObject = new ...
    dynamic automation object ...
    MyServer
```

Call a method of a dynamic automation object

*<Name of dynamic automation object>>>**<Method name>*(*<Parameters>*)

```
MyDynamicObject>>>...
    OpenFile(DocName)
```

Note: The list of methods that can be used depends on the automation server. See the documentation about the server for more details.

Lifespan of a dynamic automation object

The dynamic automation object is created when allocating the object.

The dynamic automation object is automatically destroyed at the end of the process containing its declaration or when a new allocation is performed on the object.

A dynamic automation object declared as "global" in the code:

- **for window initialization** will be destroyed at the end of the closing process of the window.
- **for project initialization** will be destroyed at the end of the closing process of the first project window.

To explicitly free a dynamic automation object, use the following syntax:

```
Delete <Name of dynamic automation object>
```

Allocating a dynamic object

When allocating a dynamic automation object, the automation server is automatically started.

To allocate a dynamic automation object with an existing instance of an automation server, use **GetActiveObject**.

Passing parameters to a method

The methods of automation servers can accept one or more parameters.

When calling a method, the sequence of parameters must be respected (see the documentation for more details).

- With some automation servers, some method parameters are not valued: only the presence of the parameter is important. The value taken by this parameter has no importance.

For example, the "EditReplace" method of Word Basic accepts 10 parameters, the last one (ReplaceAll) being not valued.

In WLanguage, any ordinary value can be passed to non-valued parameters.

- The methods of some automation servers can accept optional parameters located anywhere in the list of parameters.

In WLanguage, the optional parameters must necessarily be found after the mandatory parameters.

To give any ordinary value to an optional parameter, assign the * character to the optional parameter. On the contrary, this character will not be required for the optional parameters found after the last mandatory parameter.

Declaring variables with the same name

Several variables with the same name cannot be used in the same process (regardless of the type of these variables).

If variables with the same name are declared in different processes of the project, the rule for variable scope applies.

2.5.10 Data source

A **Data source** variable is used to describe a temporary data source (query, view, alias, ...). All the operations that can be performed on a view (or on a query) can be performed on a **Data source** variable associated with a view (or with a query).

To describe a temporary data source, you must:

1. Declare a "Data source" variable.
2. Initialize the data source (**HExecuteSQLQuery** or

hCreateView).

3. The data source is automatically freed at the end of the variable life.

Note: When closing the application (or the process where the data source was created), the data source will be automatically deleted.

Syntax

Declaring a data source

<Variable name> is data source

Declaring several data sources

<Name of variable 1>, <Name of variable 2>

are data sources

```
MyDataSource is data source
MyDataSource1, MyDataSource2 ...
are data sources
```

Initializing a "Data source" variable (query or view)

To initialize a "Data source" variable, use:

- **HExecuteSQLQuery** if the variable is associated with a query.
- **HCreateView** if the variable is associated with a view.

```
MyQuery is data source
// MyQuery is associated
// with a query.
// Initialize the variable
HExecuteSQLQuery(MyQuery, ...
    "SELECT NAME FROM CUSTOMER")
MyView is data source
// MyView is associated with a view
// Initialize the variable
HCreateView(MyView, CUSTOMER, ...
    "*", "NAME,CITY", "NAME]='A'", ...
    hViewDefault)
```

Deleting the data source (query or view)

When the same Data Source variable is used several times with different data sources, the first data source must be freed.

To free the memory space occupied by the data source (query or view), use:

- **HCancelDeclaration** if the variable is associated with a query.

- **HDeleteView** if the variable is associated with a view.

```

MyQuery is data source
MyQuery = HExecuteSQLQuery(...
    MyQuery, "SELECT NAME FROM
    CUSTOMER")
// Delete the data source
// associated with the
// MyQuery variable
HCancelDeclaration(MyQuery)
MyQuery = HExecuteSQLQuery(...
    MyQuery, ...
    "SELECT NAME FROM SUPPLIER")
MyView is data source
// MyView is associated with a view
// Initialize the variable
HCreateView(MyView, CUSTOMER, ...
    "*", "NAME,CITY", "NAME]='A'", ...
    hViewDefault)
// Delete the data source
// associated with MyView
HDeleteView(MyView)
HCreateView(MyView, CUSTOMER, ...
    "*", "NAME, FIRTSNAME", ...
    "NAME]='C'", hViewDefault)

```

Handling a query or a view by programming

To handle a query or a view by programming, we recommend that you use a "Data source" variable.

However, you have the ability to give a logical name when initializing the query or the view. In this case, the **Extern** keyword must be used to directly handle the view or query in the code editor. This method can slow down the execution of your processes.

Note: When using a logical name, the query or the view is not automatically deleted: you must use **HCancelDeclaration** and **HDeleteView**.

Declaring variables with the same name

Several variables with the same name cannot be used in the same process (regardless of the type of these variables).

If variables with the same name are declared in different processes of the project, the rule for variable scope applies.

2.5.11 File description

A **File description** variable is used to describe one or more temporary data files. The description of each data file is validated by **HDescribeFile**.

After this validation:

- this data file can be handled like any other data file described in the analysis.

- the "File description" variable is reinitialized and it can be used to describe another temporary data file.

Syntax

Declaring a data file description

<Variable name> is file description

Declaring several file descriptions

<Name of variable 1>, <Name of variable 2> are file descriptions

```

MyFile is file description
MyFile1, MyFile2 are...
file descriptions

```

Describing a "File description" variable

To describe a "File description" variable, use the WLanguage properties specific to the descriptions of data files.

To validate the description of a "File description" variable, use **HDescribeFile**.

```

// Describe the "CUSTOMER" file
MyFile..Name = "CUSTOMER"
MyFile..Type = hFileNormal
MyFile..FicCryptMethod = ...
                    hCryptStandard
// Describe the file items
// Validate the description
// of the "CUSTOMER" file
HDescribeFile(MyFile)

```

Properties specific to the description of data files

The properties specific to the description of data files are detailed in the online help.

How to describe temporary data files?

To describe temporary data files, you must:

1. Declare the "File description", "Item description" and "Link description" variables (if necessary).
2. For each data file:
 - describe the characteristics of the data file via the HFSQL properties.
 - describe the characteristics of the items via the HFSQL properties.
 - validate the description of each item (**HDescribeItem**).
 - validate the description of the data file (**HDescribeFile**).
3. Describe (if necessary) the characteristics of the links via the HFSQL properties.
4. Validate (if necessary) the description of each link (**HDescribeLink**).

Handling the items of a temporary data file

When creating a data file in the data model editor, the names of the data files and items are automatically recognized by the compiler.

When creating a data file via a "File description" variable, the names of the temporary data file and its items are defined by **..Name**. These names are not automatically recognized by the compiler. A compilation error occurs if these names are used to identify the temporary data file or its items.

In order for the name of the temporary data file and its items to be recognized by the compiler, the name of this temporary data file must be declared as a **data source**. This temporary data file can then be handled by its name in the code editor.

```
MyFile is description of file
...
MyFile..Name = "CUSTOMER"
...
HDescribeFile("CUSTOMER")
...
CUSTOMER is data source
...
HReadFirst(CUSTOMER, ...
            CUSTOMER.CUSTNAME)
```

Note: You also have the ability to declare the name of the temporary data file with the **Extern** keyword. However, the execution speed of the process will be slower.

Declaring variables with the same name

Several variables with the same name cannot be used in the same process (regardless of the type of these variables). If variables with the same name are declared in different processes of the project, the rule for variable scope applies.

2.5.12 Link description

A **Link description** variable is used to describe a link between two temporary data files. The description of each link is validated by **HDescribeLink**.

After this validation:

- this link can be handled like any other link described in the analysis.
- the "Link description" variable is re-initialized and it can be used to describe another link.

Syntax

Declaring a link description

`<Variable name> is link description`

Declaring several link descriptions

`<Name of variable 1>, <Name of variable 2>
are link descriptions`

```
MyLink is link description
MyLink1, MyLink2 are ...
link descriptions
```

Describing a "Link description" variable

To describe a "Link description" variable, use the WLanguage properties specific to the link descriptions.

To validate the description of a "Link Description" variable, use **HDescribeLink**.

```
// Describe and validate the City
// and Customer files
...
// Describe the "LIVES" link
MyLink..Name = "LIVES"
MyLink..SourceFile = "CITY"
MyLink..LinkedFile = "CUSTOMER"
MyLink..SourceKey = "CITYNAME"
MyLink..LinkedKey = "CUSTCITY"
// Validate the description of
// the "LIVES" link
HDescribeLink(MyLink)
```

Properties specific to the link description

The properties specific to link descriptions are detailed in the online help.

How to describe temporary data files?

To describe temporary data files, you must:

1. Declare the "File description", "Item description" and "Link description" variables (if necessary).
2. For each data file:
 - describe the characteristics of the data file via the HFSQL properties.
 - describe the characteristics of the items via the HFSQL properties.
 - validate the description of each item (**HDescribeItem**).
 - validate the description of the data file (**HDescribeFile**).
3. Describe (if necessary) the characteristics of the links via the HFSQL properties.
4. Validate (if necessary) the description of each link (**HDescribeLink**).

Handling the links of a temporary data file

When a link is created in the data model editor, the name of the link is automatically recognized by the compiler.

When creating a temporary link with a "Link Description" variable, the name of this link is defined by **..Name**.

This name is not automatically recognized by the compiler. Using this name to define the temporary link triggers a compilation error.

In order for the name of the temporary link to be recognized by the compiler, this link name must be declared by the **Extern** keyword. Then, this temporary link can be handled by its name in the code editor.

```
MyLink is a description of links
...
MyLink..Name = "LIVES"
...
HDescribeLink("LIVE")
...
EXTERN LIVES
...
Cardinality = ...
    LIVE..MaxLinkedCardinality
```

Declaring variables with the same name

Several variables with the same name cannot be used in the same process (regardless of the type of these variables).

If variables with the same name are declared in different processes of the project, the rule for variable scope applies.

2.5.13 Item description

An **Item description** variable is used to describe one or more items of a temporary data file. The description of each item is validated by **HDescribeItem**.

After this validation:

- this item can be handled like any other item described in the analysis.
- the "Item description" variable is re-initialized and it can be used to describe another item of a temporary data file.

Syntax

Declaring an item description

```
<Variable Name> is item description
```

Declaring several item descriptions

```
<Name of variable 1>, <Name of variable 2>
```

are item descriptions

```
MyItem is item description
MyItem1, MyItem2 are
    item descriptions
```

Describing an "Item description" variable

To describe an "Item Description" variable, use the WLanguage properties specific to the item descrip-

tions.

To validate the description of an "Item description" variable, use **HDescribeItem**.

```
// Describe the MyFile file ...
// Describe the "NAME" item
MyItem..Name = "NAME"
MyItem..Type = hItemText
MyItem..Size = 40
MyItem..KeyType = hUniqueKey
// Validate description of
// the "NAME" item
HDescribeItem(MyFile, MyItem)
// Validate the description of
// MyFile ...
```

Properties specific to the item description

The properties specific to the item descriptions are presented in the online help.

How to describe temporary data files?

To describe temporary data files, you must:

1. Declare the "File description", "Item description" and "Link description" variables (if necessary).
2. For each data file:

- describe the characteristics of the data file via the HFSQL properties.
- describe the characteristics of the items via the HFSQL properties.
- validate the description of each item (**HDescribeItem**).
- validate the description of the data file (**HDescribeFile**).

3. Describe (if necessary) the characteristics of the links via the HFSQL properties.

4. Validate (if necessary) the description of each link (**HDescribeLink**).

Handling the items of a temporary data file

When creating a data file in the data model editor, the names of the data files and items are automatically recognized by the compiler.

When creating a data file via a "File description" variable, the names of the temporary data file and its items are defined by **..Name**.

These names are not automatically recognized by the compiler. A compilation error occurs if these names are used to identify the temporary data file or its items.

In order for the name of the temporary data file and its items to be recognized by the compiler, the name of this temporary data file must be declared as a **data source**.

Then, this temporary data file and its items can be handled by their name in the code editor.

```
MyFile ...
    is file description
...
MyFile..Name = "CUSTOMER"
...
HDescribeFile("CUSTOMER")
...
CUSTOMER is data source
...
HReadFirst(CUSTOMER, CUSTOMER.CUST-
NAME)
```

Note: you can also declare the name of the temporary data file with the keyword **Extern**. However, the execution speed of the process will be slower.

Declaring variables with the same name

Several variables with the same name cannot be used in the same process (regardless of the type of these variables).

If variables with the same name are declared in different processes of the project, the rule for variable scope applies.

2.5.14 Font

A **Font** variable is used to create a dynamic font. The properties of this font can be modified in the program.

A dynamic font allows you to select the font used:

- in the drawings (**dFont**).
- in the charts (**grLabelFont**, **grLegendFont**, **grTitleFont** and **grAxisTitleFont**).
- for the prints (**iFont**).
- in the controls of a window (**..Font** property for the window controls).
- in the controls of a report (**..Font** property for the report controls).

Syntax

Declaring a font

```
<Variable name> is font
```

Declaring several fonts

```
<Name of variable 1>, <Name of variable 2>
are fonts
```

```
MyFont is font
MyFont1, MyFont2 are fonts
```

Defining the characteristics of a font

The default font is "Arial", size 12, black, without attribute.

To define the characteristics of a dynamic font, use:

- **FontCreate**.
- the following properties: **..Charset**, **..Angle**, **..Strikeout**, **..Condensed**, **..Color**, **..Extended**, **..Bold**, **..Italic**, **..Large**, **..Name**, **..Underline** and **..Size**.
- **FontSelect** that opens the standard window for font selection.

```
MyFont = ...
    FontCreate("Arial",12, iBold)
MyFont..Name = "Arial"
MyFont..Size = 12
MyFont..Bold = True
```

Declaring variables with the same name

Several variables with the same name cannot be used in the same process (regardless of the type of these variables).

If variables with the same name are declared in different processes of the project, the rule for variable scope applies.

2.5.15 Connection

A **Connection** variable is used to describe one or more connections to external databases (HFSQL Client/Server, Native Access, ...).

The connection can be established by **HOpenConnection**.

The Connection variable can also be used to:

- describe a new connection (equivalent to **HDescribeConnection**)
- create a connection (that will be established by **HOpenConnection**)
- modify an existing connection.

Syntax

Declaring a connection

```
<Variable name> is connection
```

Declaring several connections

```
<Name of variable 1>, <Name of variable 2>
are connections
```

```
MyConnection is Connection
```

Defining the characteristics of a connection

To describe a "Connection" variable, use the WLanguage properties specific to the connections.

```
// Describe the connection
MyCtion..User = "USER"
MyCtion..Password = "PASSWORD"
MyCtion..Server = "MYSERVER"
MyCtion..Database = "Database1"
MyCtion..Provider = ...
                hAccessHFClientServer
MyCtion..Password = "PASSWORD"
```

```
MyCtion..Access = hOReadWrite
myCtion..ExtendedInfo = "Info"
MyCtion..CursorOptions= ...
                        hClientCursor
HOpenConnection (MyConnection)
```

Declaring variables with the same name

Several variables with the same name cannot be used in the same process (regardless of the type of these variables).

If variables with the same name are declared in different processes of the project, the rule for variable scope applies.

2.5.16 Queue

A queue is a structured type that is used to group a set of elements of the same type. The elements are added at the end of the queue and they are retrieved in enqueue order.

Syntax

Declaring and initializing a queue

<Queue name> is Queue of <Type of elements in the queue>

- For example, if the elements are added in the following order: 1, 2, 3, they will be retrieved in the same order 1, 2, 3

```
MyQueue is queue of int
// Enqueue the values 1, 2 and 3
Enqueue(MyQueue, 1)
Enqueue(MyQueue, 2)
Enqueue(MyQueue, 3)
// Display the trace: 1, 2, 3
x is int
WHILE Dequeue(MyQueue, x)
    Trace(x)
END
```

WLanguage properties that can be used with the Queue type

AdditionCompleted	Returns and modifies the addition mode of the elements. This property is used for multithread management
NbPendingThread	Returns the number of pending threads. This property is used for multithread management
Occurrence	Returns the number of occurrences of the queue
Empty	<u>True</u> if the queue is empty, <u>False</u> otherwise

Note: These properties can be used with one of the following syntaxes:

- <Variable name>..<Property name>*
- <Variable name>.<Property name>*

WLanguage functions for managing the queues

The following functions can be used to handle a Queue variable.

Dequeue	Retrieves the element found at the beginning of the queue.
Deserialize	Deserializes a queue.
Enqueue	Adds an element at the end of the queue.
Serialize	Serializes a queue.
Delete all	Deletes all the elements from a queue.

The queues and the multithread

The management of multithread is taken into account when adding and deleting an element (Enqueue and Dequeue functions).

- You also have the ability to use properties during a multithread management but the result is not permanent.

For example:

```
IF MyQueue.Occurrence>0 THEN
// When the thread arrives here, ...
// the queue may be empty
END
```

Browsing the queues:

The FOR EACH syntax can be used to browse the queues. The elements are browsed in enqueue order that is similar to the dequeue order.

The syntax used corresponds to the one used for arrays:

```
FOR EACH [ELEMENT] <Variable> ...
    [, <Counter> [, <Counter>]] ...
OF <Queue>
    ...
END
```

The elements can be modified during the browse. If the queue is modified during a browse, the elements browsed will be the ones found when the browse was initialized:

- the elements enqueued after the initialization of the browse will not be browsed.
- the elements dequeued after the initialization of the browse will still be browsed.

Notes

- In the debugger, the content of the queue is displayed in dequeue order.
- A queue can be used to type a procedure parameter.
- A queue can be copied with the operator =. You also have the ability to copy an instance of class or structure containing a queue.
- A queue can be initialized by a list of elements in enqueue order.

2.5.17 List

A list is a structured type that is used to group a set of elements of the same type. The elements can be added at the end of the list or they can be inserted into the list.

Syntax

Declaring and initializing a list

<List name> is List of <Type of elements in the list>

WLanguage properties that can be used with the List type

The following properties can be used to handle a List variable

Occurrence	Returns the number of occurrences of the list.
Empty	<u>True</u> if the list is empty, <u>False</u> otherwise.

Note:

These properties can be handled by using one of the following syntaxes:

- *<Variable name>.<Property name>*
- *<Variable name>.<Property name>*

WLanguage functions for list management

The following functions can be used to handle a List variable.

Add	Adds an element at the end of the list.
Deserialize	Deserializes a list.
Insert	Inserts an element into a list.
Serialize	Serializes a list.
Deletes	Deleting an element from a the list
DeleteAll	Deletes all the elements from the list.

The lists and the multithread.

The management of multithread is taken into account when adding, inserting and deleting an element.

- You also have the ability to use properties during a multithread management but the result is not permanent.

For example:

```
IF MyList.Occurrence>0 THEN
// When the thread arrives here, ...
// the list may be empty
END
```

Browsing the lists.

The FOR EACH syntax can be used to browse the lists.

- The syntax used corresponds to the one used for arrays:

```
FOR EACH [ELEMENT] <Variable> .....
[, <Counter> [, <Counter>]] ...
OF <List> [<Direction>]
...
END
```

The elements can be modified during the browse. If the list is modified during a browse, the browse is affected by the additions and by the deletions. The functions available during the browse are as follows:

- **Syntax:** *Insert(<List Name>, BeforeCurrentElement, <Value>)*
Caution: the element will be browsed by the next iteration if the loop is descending.
- **Syntax:** *Insert(<List Name>, AfterCurrentElement, <Value>)*
Caution: the element will be browsed by the next iteration if the loop is ascending.
- **Syntax:** *Delete(<List Name>, CurrentElement)*

Notes:

- The content of the list can be displayed in the debugger.
- A list can be used to type a procedure parameter.
- A list can be copied by the list operator.
- A list can be initialized by a list of elements.

2.5.18 Stack

A stack is a structured type that is used to group a set of elements of the same type. The elements are added at the end of the stack and they are retrieved from the most recent one.

Syntax

Declaring and initializing a stack

<Stack name> is stack of <Type of elements in the stack>

- For example, if the elements are added in the following order: 1, 2, 3, they will be retrieved in the following order: 3, 2, 1.:

```
MyStack is stack of int
// Push the values 1, 2 and 3
Push(MyStack, 1)
Push(MyStack, 2)
Push(MyStack, 3)
// Display the trace: 3, 2, 1
x is int
WHILE Pop(MyStack, x)
    Trace(x)
END
```

WLanguage properties that can be used with the Stack type

The following properties can be used to handle a Stack variable.

AdditionCompleted	Returns and modifies the addition mode of the elements. This property is used for multithread management
NbPendingThread	Returns the number of pending threads. This property is used for multithread management
Occurrence	Returns the number of occurrences of the stack.
Empty	<u>True if the stack is empty.</u> <u>False otherwise</u>

Note: These properties can be used with one of the following syntaxes:

- <Variable name>.<Property name>
- <Variable name>.<Property name>

WLanguage functions for managing the stacks.

The following functions can be used to handle a Stack variable.

Pop	Retrieves the element at the beginning of the stack.
Deserialize	Deserializes a stack.

Push	Adds an element into a stack.
Serialize	Serializes a stack.
DeleteAll	Deletes all the elements from a stack.

The stacks and the multithread

The management of multithread is taken into account when adding and deleting an element (Push and Pop functions).

You also have the ability to use properties during a multithread management but the result is not permanent. For example:

```
IF MyStack.Occurrence>0 THEN
// When the thread arrives here, ...
// the stack may be empty
END
```

Browsing the stacks

The FOR EACH syntax can be used to browse the stacks. The elements are browsed in pop order that is the reverse order of the push order.

The syntax used corresponds to the one used for arrays:

```
FOR EACH [ELEMENT] <Variable> ...
    [, <Counter> [, <Counter>]] ...
OF <Stack>
    ...
END
```

The elements can be modified during the browse. If the stack is modified during a browse, the elements browsed will be the ones found when the browse was initialized:

- the elements pushed after the initialization of the browse will not be browsed.
- the elements popped after the initialization of the browse will still be browsed.

Notes:

- In the debugger, the content of the stack is displayed in pop order.
- A stack can be used to type a procedure parameter.
- A stack can be copied by the operator =. You also have the ability to copy a class or structure instance containing a stack.
- A stack can be initialized by a list of elements in push order

2.6 Local variables/global variables

Two types of variables are available:

- **Local variable:** can only be used in the process where it was declared.
- **Global variable:** can be used in all the processes related to process where it was declared.

Important: Two variables must not be declared with the same name (especially a global variable and a local variable).

2.6.1 Global variables

Variables global to a project

The global variables *declared in the initialization process of a project* can be used in all the processes:

- of the project.
- of the windows/pages (processes of the window/page, of its associated controls and local procedures).
- of the reports (processes of the report, its associated controls and local procedures).

Availability of the variables in WebDev Browser code: the global server variables of the project are available in browser code only for the following types: boolean, integer, real, string. **Caution:** the modifications performed on these global variables in browser code are not applied to the server.

Tip: the global variables declared in a server code can be used to transmit information to the browser.

Variables global to a window or to a page

The global variables *declared in the declaration process of the global variables of a window/page* can be used in all the processes:

- of the window/page.
- of the controls of the window/page.
- of the local procedures associated with the window/page.

Limits:

- The global variables of a window cannot be used by its sibling windows.
- The global variables declared in a child window cannot be used in its parent window.
- When the window/page where the variable was declared is closed, this variable cannot be used

anymore.

Availability of the variables in WebDev in Browser code: the Server variables global to a page are available in the Browser codes of the page for the following types: boolean, integer, real, string.

Caution: the modifications performed on these global variables in browser code are not applied to the server.

Tip: the global variables declared in a server code can be used to transmit information to the browser.

Browser variables that are global to a page

The global variables declared *in the "Page Load (onLoad)" code* can be used in all the browser processes:

- of the page.
- of the page controls.
- of the local procedures associated with the page.

Availability of the variables in Server code: the Browser variables global to a page are not available in the Server codes of the page.

Tip: the global variables declared in a browser code can be used to exchange information between different processes run on the browser.

Notes:

- The global browser variables cannot be initialized on the declaration line.
- The global browser variables can be initialized with the value of a server global variable (only for the boolean, integer, real and string types).
- We recommend that you disable the "cache" of your browser when developing the WebDev application. The global variables are translated into JavaScript in ".JS" files. If the "cache" is enabled, the tests of your pages may reload files corresponding to former values of your variables. See the online help for more details.

Variables global to a report

The global variables *declared in the opening process of a report* can be used in all the processes of the report, of the report controls and of the local procedures associated with the report.

Variables global to a set of procedures

The global variables declared in the initialization process of a set of procedures can be used in all the processes:

- of the different procedures found in the set.
- of the current project.

Syntax

Declaring one or more global variables

GLOBAL

<Global variables>

GLOBAL

```
// All the declarations that follow
// are global variables
Subscript is int
CustomerName is string
```

2.6.2 Local variables

The local variables can only be used in the processes where they have been declared. The local variables are unknown outside these processes. These variables cannot be shared among several processes.

By default, a variable is local when it is declared.

Syntax

Declaring one or more local variables

[LOCAL]

<Local variables>

Local

```
// All the declarations that follow
// are local variables
I is int
CustomerFName is string
```

2.7 Rule for variable scope

The rule for variable scope is as follows:

- If a variable "global" to the project and a variable "global" to a window/page have the same name:
 - the **variable "global" to the window/page** will be used in all the other processes of the window/page and its controls, including the "local" procedures of the window/page.
 - the **variable "global" to the project** will be used in all the other processes.
- If a **variable "global"** to the project and a variable "local" to a process have the same name:
 - the **"local" variable** will be used in the process where is was declared.
- the **variable "global"** to the project will be used in all the other processes.

- If a **variable "global" to a window/page** and a variable "local" to a process of this window/page have the same name:
 - the **"local" variable** will be used in the process where is was declared.
 - the **variable "global"** to the window/page will be used in all the other processes of the window/page and its controls (including the "local" procedures of the window/page).
 - **none of these two variables** will be used in the rest of the project.

Exception: The rule for variable scope does not apply to the constants.

For example:

```
Condition1 AND Condition2 OR
Condition3
```

will be evaluated as follows:

```
(Condition1 AND Condition2) OR
Condition3
```

3.3 Arithmetic operators

3.3.1 Overview

The arithmetic operators are:

- "+": Addition (numeric value or string).
- "-": Subtraction (numeric value).
- "*": Multiplication.
- "/": Division.
- "++": Incrementation (numeric value).
- "--": Decrementation (numeric value).
- "+=": Add a value to the variable or control (numeric or text).
- "-=": Subtract a value from the variable or control (numeric).

3.3.2 Calculation rules

The different calculations are performed without loss of precision and without being truncated. The flow checks are performed when the result is assigned to a variable.

3.3.3 Notes

Displaying the result

The result of the calculation cannot be directly displayed by the following operators :

- "++": Incrementation.

- "--": Decrementation.
- "+=": Add a value to the variable or control (numeric or text).
- "-=": Subtract a value from the variable or control (numeric).

Therefore, this example generates an error during the compilation:

```
n is int = 10
Trace(n+=1)
```

To display the result, perform the following modifications:

```
n is int = 10
n += 1
Trace(n)
```

Equivalence

- j ++ is equivalent to j = j + 1
- j -- is equivalent to j = j - 1
- j += 3 is equivalent to j = j + 3
- j -= 3 is equivalent to j = j - 3

We recommend that you use the following syntaxes: "j ++", "j --", "j += " and "j -= ", which are faster than the usual syntaxes.

3.4 Binary operators

The operations on the binary values are performed only:

- with the WLanguage functions: BinaryAND, BinaryOR, BinaryNOT, BinaryXOR
- with specific operators: binary operators, offset to the right or to the left operators, bit access operators.

3.4.1 Binary operators

Binary AND, OR and exclusive OR

The following syntaxes can be used:

- Binary AND: <Value 1> & <Value 2>
- Binary OR: <Value 1> | <Value 2>
- Exclusive binary OR: <Value 1> || <Value 2>

The type of result depends on the type of the operands:

Value 2 Value 1	4-byte integer	8-byte integer	Other
4-byte integer	4-byte integer	8-byte integer	4-byte integer
8-byte integer	8-byte integer	8-byte integer	8-byte integer
Other	4-byte integer	8-byte integer	8-byte integer

Binary not

The syntax is as follows: `~ <Value>`

The type of result depends on the type of the operand:

Operand	Result
4-byte integer	4-byte integer
8-byte integer	8-byte integer
Other	8-byte integer

3.4.2 Shift operators

Offset to the left:

```
<Value 1> bitLeftShift <Value 2>
bitLeftShift(<Value 1>, <Value 2>)
```

Offset to the right:

```
<Value 1> bitRightShift <Value 2>
bitRightShift(<Value 1>, <Value 2>)
```

Notes

- The bits of `<Value 1>` are shifted from `<Value 2>` bits to the right or to the left.
For example:

```
bitLeftShift(4,1) // Returns 8
```

Indeed, 4 in decimal corresponds to 0100 in binary. Shifted from 1 bit to the left, we get 1000 in binary that corresponds to 8 in decimal.

Another example:

```
bitRightShift(4,2) // Returns 1
```

Indeed, 4 in decimal corresponds to 0100 in binary. Shifted from 2 bits to the right, we get 0001 in binary that corresponds to 1 in decimal.

- The bits that exceed the size of `<Value 1>` are ignored. For example:

```
bitLeftShift(4,30) // Returns 0
bitLeftShift(4,4) // Returns 0
```

- If `<Value 2>` is greater than the size of `<Value 1>` (32 for a 4-byte integer and 64 for a 8-byte integer), the result is always equal to 0. For example:

```
bitLeftShift(4,35) // Returns 0
```

- The type of result depends on the type of the operand:

Operand Value 1	Result
4-byte integer	4-byte integer

8-byte integer	8-byte integer
Other	8-byte integer

3.4.3 Operator for direct access

Access to a bit

Syntax: `<Value 1> [<n>]`

This syntax is used to read or modify the value of the `<N>` bit in the `<Value 1>` value.

The counting of the bits starts from 1:

- 1 to 32 for a 4-byte integer,
- 1 to 64 for a 8-byte integer.

If the value of `<n>` is incorrect, the operation returns 0.

Examples:

```
// Positions the fifth and
// seventh bits to 1
n is int
n[5] = 1
n[7] = True
// Checks the value of bits 4, 5, 6
and 7
IF n[4] THEN Trace(4) // not displayed
IF n[5] THEN Trace(5) // displays 5
IF n[6] THEN Trace(6) // not displayed
IF n[7] THEN Trace(7) // displays 7
```

Access to a 1-byte, 2-byte or 4-byte integer

Syntaxes:

```
<Value 1> [ <n>, wInt_1 ]
<Value 1> [ <n>, wInt_2 ]
<Value 1> [ <n>, wInt_4 ]
```

These syntaxes are used to read or modify the value of the 1-byte, 2-byte or 4-byte integer in the `<Value 1>` value.

Possible values for `<n>` depending on the type of `<Value 1>`:

Value 1	wInt_1	wInt_2	wInt_4
4-byte integer	1 to 4	1 to 2	1
8-byte integer	1 to 8	1 to 4	1 to 2

If the value of `<n>` is incorrect, the operation returns 0.

Access to the value of several bits

Syntaxes:

<Value 1> [TO <n>]

<Value 1> [<n> TO]

<Value 1> [<n> TO <o>]

<Value 1> [<n> ON <Number>]

These syntaxes are used to read or modify the value corresponding to the specified bits.

3.5 Comparison operators

3.5.1 Overview

The comparison operators can be divided into several categories:

Equality	Strict equality: = Flexible equality: ~= Very flexible equality: ~~ Starts with:
Comparison	Different from: <> Less than or equal to: <= Greater than or equal to: >= Strictly less than: < Strictly greater than: > Starts with: [=
Interval for comparison	Strictly included between: Value1 < x < Value2 Included between: Value1 < x <= Value2 Value1 <= x < Value2 Included between (including bounds) : Value1 <= x <= Value2 Included between (including bounds) : Value1 TO Value2

The comparison operators can be used with all the types of operands.

The result of a comparison expression is a boolean.

3.5.2 Details

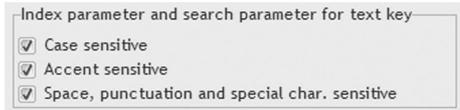
Flexible equality and very flexible equality

The flexible equality (~=) and the very flexible equality (~~) operate on the character strings only (except for the fixed strings). These operators enable you to:

- make no difference between the uppercase characters and the lowercase characters,
- ignore the space characters found before and after the string whose test must be run,
- ignore the lowercase accented characters,
- ignore the space characters and the punctuation characters inside the strings (very flexible equality only)

HFSQL equivalence: To get a behavior equivalent to the very flexible equality when performing a search

on a text key in a HFSQL file, you must configure the following options when describing the item in the analysis:



```
"Dupond" = "DUPOND" // returns
False
"Dupond" ~= "DUPOND" // returns
True
" Dupond" ~= "DUPOND"
// returns True
" Dupond" ~= "Dupond"
// returns True
"Crème brûlée" ~= "Creme brulee"
// returns True
"Comp. S.A.R.L" ~~ "Ent SARL"
// returns True
```

Equality and comparison: Real containing more than 6 different decimal places

The test of equality between two real numbers is run according to the first 6 decimal places. Indeed, the rounding errors caused by the internal coding of the reals require a specific test.

This test must be run by comparing the difference between the two values to test and a reference value. Depending on the type of your application, this value can be equal to 0.00001 or even less.

This rounding management is not specific to WinDev.

It is common to all the programming languages that handle the reals in binary format.

```
Diff, R1, R2 are real
Diff = 0.00001
IF Abs(1-(R1/R2)) < Diff THEN
// R1 and R2 are equal
ELSE
// R1 and R2 are different
END
```

Comparison intervals

The comparison intervals are used to simplify the syntax of complex comparisons. Therefore, the line

```
IF x>5 and x<10 THEN ...
```

can be replaced by

```
IF 5<x<10 THEN...
```

The line :

```
IF x>5 and x<10 THEN ...
```

can be replaced by

```
IF X=5 TO 10 THEN...
```

Example:

```
MyArray is array of 5 strings
I is int
MyArray[1] = "Dupond"
MyArray[2] = "Aida"
MyArray[3] = "Parapoline"
MyArray[4] = "Moulin"
MyArray[5] = "Foolamour"
FOR I = 1 TO 5
  IF "B"<MyArray[I]<="M" THEN
    Trace(MyArray[1])
    // Displays Dupond and Foolamour
  END
END
```

Comparing instances of structures and instances of classes

The dynamic structures are instantiated when they are allocated.

The "=" operator enables you to compare instances of dynamic structures (or instances of dynamic classes).

Example:

```
// Declare the structures
O is ST1
P is ST2
// Declaration and instantiation
// of dynamic structures
P1 is dynamic structure ST1
P1 = O
P2 is dynamic structure ST2
P2 = O
// Comparison
IF P1 = P2 THEN ...
```

The same operation can be performed on the instances of classes.

3.6 Operators on character strings

The character strings can be handled by specific WLanguage functions or by the +, [[and]] operators.

The operators on character strings are as follows:

- "+": To concatenate strings.
- "[[" and "]]" (double opening square brackets and double closing square brackets): Sub-string extraction operator.
- "=": Strict equality
- "~=": Flexible equality (not available in WebDev browser code)
- "[=": Start with

```
Text = "Programming Guide"
Text[[13]] // Returns "G"
Text[[13 to 17]]
// Returns "Program"
Text[[13 to]]
// Returns "Programming"
Text[[to 13]]
// Returns "Programming G"
Text[[13 on 3]] // Returns "Gui"
```

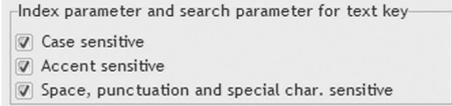
```
Text + " of WinDev"
// Returns "Programming Guide
// of WinDev"
If Text ["Guide" THEN ...
Trace(Text)
```

3.6.1 Flexible equality and very flexible equality

The flexible equality (~=) and the very flexible equality (~~) operate on the character strings only (except for the fixed strings). These operators enable you to:

- make no difference between the uppercase characters and the lowercase characters,
- ignore the space characters found before and after the string whose test must be run,
- ignore the lowercase accented characters,
- ignore the space characters and the punctuation characters inside the strings (very flexible equality only).

HFSQL equivalence: To get a behavior equivalent to the very flexible equality when performing a search on a text key in a HFSQL file, you must configure the following options when describing the item in the analysis:



3.6.2 The [[and]] operator

The [[and]] operator is used to extract and to replace a sub-string.

Some examples:

- `STRINGTOEXTRACT[[<Position>]]`: Extracts a character from a string according to its position.
- `EXTRACTSTRING[[<Position>]] = <New string>`: Replaces the character in the string by the new string.
- `STRINGTOEXTRACT[[<Start> to <End>]]`: Extracts a character string from a string according to its Start and End positions.
- `STRINGTOEXTRACT[[<Start> to]]`: Extracts a character string from the <Start> of the string to the end of the string.
- `STRINGTOEXTRACT[[to <End>]]`: Extracts a character string from the beginning of the string to the <end> position in the string.
- `STRINGTOEXTRACT[[<Start> on <Number>]]`: Extracts a set number of characters based on the start position and the specified number of characters.

Caution: Be careful when you use the [[and]] operators in procedures that handle character strings. You may modify your character strings without being aware of it.

For example, the following procedure can modify part of the string.

```
Procedure P(sString)
  IF sString then sString="5"
  //Call the procedure using the
  //following line: p(sVar[[3 to]])
```

To avoid modifying the initial string, the parameter must be passed by value:

- by using brackets around the parameter in the call to the procedure

- by using the *Local* keyword in the header of the procedure

3.6.3 Operators on the character strings and UNICODE

The available operators are as follows:

- "+": To concatenate strings.
- "[[" and "]]" (double opening square brackets and double closing square brackets: Sub-string extraction operator.

"+" operator

Two UNICODE strings can be concatenated. A UNICODE string and an ANSI string cannot be concatenated.

Note: No compilation error is detected if the concatenation of two ANSI strings is assigned to a UNICODE string (and conversely). However, a WLanguage error will occur at run time.

"[[and]]" operator

All the syntaxes of the [[]] operator are available for UNICODE strings.

If the string passed in parameter is in ANSI format, the result returned by the [[]] operator will be in ANSI format.

If the string passed in parameter is in UNICODE format, the result returned by the [[]] operator will be in UNICODE format.

The position parameters and the length parameters are expressed in number of characters.

Note: No compilation error occurs if the result of the [[]] operator on an ANSI string is assigned to a Unicode string (and conversely). However, a WLanguage error will occur at run time.

3.6.4 Position in a character string

The position of the first character is set to 1 (and not to 0).

Position returns the start position of a given character string inside another character string.

3.6.5 Functions of WLanguage

The character strings can also be handled by the functions:

- *Right.*
- *Left.*
- *Middle.*

3.7 Operator on address

The **&** operator returns the address of a variable in integer format.

```
i is int
z is ASCIIZ string on 50
t is array of 10 reals
s is composed of
  Age is int
  Name is ASCIIZ string on 50
END
addr is int
addr = &i
addr = &z
addr = &z + 2 //address of the 3rd
              //character of the string
addr = &t //Start address of the
          //array (1st element)
addr = &t[5] //address of 5th
            // element
addr = &s //address of the
          //structure, therefore
          //address of s.age
addr = &s.age
addr = &s.name
```

Syntax

```
<Result> = &<Variable name>
```

Notes

- The **&** operator allows you to get the address of all kinds of variables (simple or composite).
- The **&** operator is mostly used to pass addresses to **CallDLL32**.
- The **&** operator does not return the address:
 - of a control
 - of a HFSQL item
- The **&** operator displays an error when the project is compiled then. Indeed, the addresses of these two elements can change at any time.
- If a "Character string" must be passed by address to **CallDLL32**, you must not use a String variable: the address of the string can change at any time. we recommend that you use the ASCIIZ String type.
- If the variable is a local variable, the value returned by the **&** operator must only be used in the process where the variable was declared.
- When the current process is ended, the address will be invalid and it must not be used anymore.

3.8 Indirection operators

The { and } operators allow you to access a control, a variable or a data file item by dynamically building the name of the control, variable or file item.

The { and } operators present several benefits:

- pass a control name or an item name in parameter
- build the name of a control or item by programming

Simple indirection

```
{ "NAME" } = CustName
  // is equivalent to NAME=CustName
{ "NAME" } = { "CU.CUSTNAMEI" }
  // is equivalent to NAME=CU.CUST-
NAME
{ "WINCUST.NAME" } = CustName
  // is equivalent to
  // WINCUST.NAME=CustName
{ "WINCUST"+".NAME" } = CustName
  // is equivalent to
  // WINCUST.NAME=CustName
```

Indirection in a procedure

```
ControlName is string
ControlName = "EDIT1"
// EDIT1 is the name of the control
// Call a procedure used to make
// a control invisible
INVISIBLE(ControlName)
PROCEDURE INVISIBLE(NControl)
{NControl} .. Visible = False
```

Indirection with variable

```
AliasName is string
AliasName = PreviousWin()
// NAME is the name of the control
// CustName is the value to assign
{AliasName+" .NAME" } = CustName
Abbrev is string
ItemName is string
{Abbrev+"+ItemName" } = CustName
{Abbrev+"."+ItemName" } =
  {ControlName}
```

Indirection with a class

```
// Declare a class with
// two members
MyClass is class
    Member1 is string
    Member2 is string
END
// Display the value of a member
GLOBAL PROCEDURE DisplayMember(...
    Number)
// Retrieve the value of the
// selected global member
Value is string = ...
    {"::Member"+Number}
Info(Value)
```

Syntax

Simple indirection

{<Expression>}

<Expression>: is used to identify the control, variable or item to handle. A WLanguage error occurs if this expression corresponds to an empty string ("").

Indirection on a class member

{<Expression>}

<Expression>: is used to identify the member of the class. If the member is:

- global, use the following syntax: {"::Member-Name"}
- not global, use the following syntax: {"Member-Name"}

A WLanguage error occurs if this expression corresponds to an empty string ("").

Indirection by specifying the type of the element (optimizes the execution speed)

{<Expression>, <Type>}

<Expression>: is used to identify the control, variable or item to handle. A WLanguage error occurs if this expression corresponds to an empty string ("").

<Type>: Constant used to specify the type of the element sought:

indControl	Element sought among the controls
indConnection	Element sought among the connections
indReport	Element sought among the reports
indWindow	Element sought among the windows
indFile	Element sought among the data files
indGPW	Element sought among the elements of the user groupware
indLink	Element sought among the links
indPage	Element sought among the pages
indQueryParameter	Element sought among the parameters of queries
IndItem	Element sought among the items
IndVariable	Element sought among the variables

When should I use the indirection?

The indirection can be used for example:

- To access the value of an element (control, variable, item ...)

```
{s_ControlName} = 10
ControlValue = {s_ControlName}
```

- to use a property

```
{s_ControlName}..Height = 10
```

Note: to designate the current object, use the following keywords:

- **Myself:** the current control
- **MyWindow:** the current window
- **MyPage:** the current page
- **MyReport:** the current report
- **MySource:** the current data source
- **MyFile:** the file that activated the current trigger.

3.9 Various operators

The various operators group the following operators:

- "(" and ")": The parentheses
- "[" and "]": The brackets
- ",": The comma
- ".": The period
- ":": The colon
- ";": The semicolon

- "..": The double dot
- "...": The triple dot
- "//": The double bar

3.9.1 The brackets

The brackets are used to:

- group the different elements of an operation while specifying the priority order. For example:

```
If (A-B)*5 THEN ...
```

- Specify the different parameters of a WLanguage function. For example:

```
dCopyBlt (Image1, image2)
```

3.9.2 The square brackets

The square brackets are indexing operators. They are used to identify the indexed objects.

Some examples:

- **Element of an Array HFSQL item.** Example: Customer.Address[1]
- **Element of an array defined in WLanguage.** Example:

```
MyArray is array of 5 strings
MyArray [1] = "Smith"
```

- **Window control (combo box, list box, array, radio button, check box).** Examples:
 - combo: MyCombo[1] = "Boston"
 - check box: checkbox[1] = True

3.9.3 The comma

The comma is used to separate:

- different declarations of variables of the same type performed on the same line. For example:

```
MyExample1, MyExample2 ...
are strings
```

- The different parameters of a WLanguage function. For example:

```
dCopyBlt (Image1, image2)
```

3.9.4 The semicolon

The semicolon is used to separate the different statements written on the same line. For example:

```
I is int;M is string
```

3.9.5 Colon

The colon is used to access the members and methods of the objects.

A double colon is used to access the global members and the global methods of a class.

3.9.6 The dot

The dot is used to access a sub-element. Some examples:

- access to a window/page control: MyWindow.DateEditControl
- access to a global variable of a window/page:

```
MyWindow.gDuration
```

- access to an item of a data file: Customer.Name
- access to a table column: Table.Column1

3.9.7 The double dot

The double dot is used to access a property of a control, window, report, data file, item or variable.

Some examples:

```
Scrollbar..MaxValue = 200
ShippingCost..Title = ...
"Shipping cost window"
MyReport..BottomMargin = 20
MyFont..Name = "Arial"
```

3.9.8 Triple dot

The "..." operator (triple dot) is used to continue a logical line over the next physical line. For example:

```
IF (FamilyCode="LV") AND ...
(FamilyCode="MO") THEN
Process
END
Caption = "Enter the "+ ...
"name of customer."
```

The "..." operator must necessarily be used at the end of a line (and not in the middle of a line).

No character must be found after the "..." operator (apart from comments).

From version 12, the "..." operator is no longer required to continue over the next line:

- the parameters of functions or procedures
- the right operands for the arithmetic and logical operators.

```
// Arithmetic operators
// (+, -, /, *) and logical (AND, OR)
n is int = 5 +
6
```

3.9.9 The double slash

The double slash is used to comment out the text that follows this sign. This text will be ignored when compiling the code and when running the project.

For example:

```
// Search in a string
MyString is string = ...
"WinDev is great"
IF Position(MyString, "W")<>0 THEN
Info("Found") //Result
END
```

4. WLANGUAGE STATEMENTS

4.1 Composite statements

Several types of composite statements are available in WLanguage:

- Conditional statements
 - SWITCH: Run an action or another one according to an expression
 - IF: Select the action to perform according to a condition
- Loop statements
 - LOOP: The statement block is repeated endlessly
 - FOR: The statement block is repeated until a limit value
 - FOR EACH: HFSQL browse (full browse, browse with search, browse with filter)
 - WHILE: The statement block is repeated according to a condition
- Branch statement: GOTO: Branch to a given label

4.1.1 LOOP statement

The statement block is repeated endlessly. The number of iterations in the statement block is not checked, there is no expression to evaluate.

```

LOOP
//read a line of the file
//text
    ALine = fReadLine(FileNum)
    IF ALine = EOT THEN BREAK
    ProcessLine(ALine)
END

```

Syntax

- **Syntax 1: Loop with exit based on an "IF" condition**

```

LOOP
...
IF <Condition> THEN BREAK
...
END

```

- **Syntax 2: Loop with exit based on an "WHILE" condition**

```

LOOP
...

```

```
DO WHILE <Condition>
```

- **Syntax 3: Loop with exit based on the number of iterations**

```
LOOP (<Number of iterations>)
```

```
...
END
```

Code to run

The code to run is found between the **LOOP** and **END** statements.

Exiting from a loop

Several statements are available:

- RETURN: Exit from the loop and exit from the current process (or procedure).
- RESULT: Return a status report to the calling process. Exit from the loop and exit from the current process (or procedure).
- BREAK: Exit the loop and resumes the current process.

Close is used to exit from the loop and to close de current window.

Caution: RETURN and RESULT cannot be used in the same process.

Running the next iteration

To directly run the next iteration **without ending the code of the current iteration**, use the Continue statement:

```

LOOP
...
IF <Condition> THEN CONTINUE
// Go back to the LOOP keyword
...
END

```

Loop without end

During the project compilation, an endless loop (loop without BREAK, RETURN or RESULT statement) is signaled by a warning.

4.1.2 GOTO statement

The **GOTO** statement is used to run a code identified by a given label.

Important: The label must be defined in the same process as the **GOTO** statement.

```

Res = fOpen(FileName, FOWrite
IF Res = -1 THEN GOTO ERROPEN
Res = fWrite(Res, "Process OK")
IF Res = -1 THEN GOTO ERRWRITE
...
RESULT True
ERROPEN :
Info("File "+FileName+...
    " cannot be opened. "+...
    "Check its existence.")
RESULT False

ERRWRITE :
Info("Unable to write"...
    "in + "+FileName)
RESULT False
    
```

Syntax

```

...
GOTO <Label name>
...
<Label name>:
    <Code of the label>
    
```

Code of the label

Once the code of the label has been run, the program runs the code lines that directly follow the label.

Limits of labels

Several labels with the same name cannot be used in the same process (or procedure).

A warning is displayed during the compilation of the project if a label is used by no **GOTO** statement.

Label and FOR and FOR EACH statements

The **GOTO** statement does not allow you to access a FOR or FOR EACH statement block directly.

This is why the following code triggers a compilation error:

```

FOR Subscript = 1 TO 10
    RestartLabel: Res ++
END
GOTO RestartLabel
    
```

4.1.3 FOR statement

The statement block is repeated while a control variable is modified and compared to a limit value (before each beginning of the statement block).

The statement block is run for each one of the values successively taken by the control variable. The initial value is assigned to the control variable during the first entry in the **FOR** statement.

```

FOR Subscript = 1 TO 10
    Array[Subscript] = MyVariable + 10
END
    
```

Syntax

```

FOR <Control variable> = <Initial value> TO
<Final value> [STEP <x>]
...
END
    
```

Code to run

The code to run is found between the **FOR** and **END** statements.

Increment step

The increment step of the **FOR** statement must be constant. A warning is displayed during the project compilation if the increment step is likely to change at each iteration.

Exiting from a FOR loop

Several statements are available:

- **RETURN:** Exit from the **FOR** loop and exits from the current process (or procedure).
- **RESULT:** Return a status report to the calling process. Exit from the **FOR** loop and exit from the current process (or procedure).
- **BREAK:** Exit the **FOR** loop and resume the current process.

Close allows you to exit from the **FOR** loop and to close the current window.

Caution: **RETURN** and **RESULT** cannot be used in the same process.

Running the next iteration

To directly run the next iteration **without ending the code of the current iteration**, use the **Continue** statement:

```

FOR<Control variable> = <Initial
value> TO <Final value> [STEP <x>]
...
IF <Condition> THEN CONTINUE
    // Go back to the FOR keyword
...
END
    
```

In this case, the control variable is automatically incremented.

4.1.4 FOR EACH/FOR ALL statement, file browsing of data

The **FOR EACH** statement is used to perform different types of HFSQL browse:

- full browse,
- full browse based on a specified key,
- browse with filter,
- browse with filter based on a specified key,
- browse with filter on the search key.

Note: Both the **FOR EACH** and **FOR ALL** statements are accepted. The **FOR EACH** statement will be used in this documentation but it can be replaced by **FOR ALL**.

Syntax

Full Browse

```
FOR ALL <File> [<Direction>]
...
END
```

```
FOR EACH Customer
// Add customers into the list
ListAdd(CustomerList,...
Customer.CustomerNum)
END
```

Note: The search key is automatically defined by the HFSQL engine.

Full browse based on a specified key

```
FOR EACH <File> ON <Search item> [<Direction>]
...
END
```

```
FOR EACH Customer ON CustomerNum
// Add customers into the list
ListAdd(CustomerList,...
Customer.CustomerNum)
END
```

Browse with filter

```
FOR EACH <File> WHERE "<1st Condition>
[AND/OR/NOT <2nd Condition>
[AND/OR/NOT...<Nth Condition>]]" [<Direction>]
...
END
```

```
FOR EACH Customer WITH ...
"CustomerCity = 'Boston'"
ListAdd(CustomerList,...
Customer.CustomerNum)
END
City = "Paris"
FOR EACH Customer WITH ...
"CustomerCity = '"+City+...'
" and CustomerAge >= 21"
// Add the customers into the list
ListAdd(CustomerList,...
Customer.CustomerNum)
END
```

Note: The search key is automatically defined by the HFSQL engine.

Browse with filter based on a specified key

```
FOR EACH <File> WHERE "<1st Condition>
[AND/OR/NOT <2nd Condition>
[AND/OR/NOT...<Nth Condition>]]" [<Direction>]
ON <Key Item>
...
END
```

```
FOR EACH Customer WITH ...
"CustomerCity = "+...
"'Boston'" ON CustomerNum
// Add customers into the list
ListAdd(CustomerList,...
Customer.CustomerNum)
END
City = "Paris"
FOR EACH Customer WITH ...
"CustomerCity = "+...
"'"+City+"'" and CustomerAge >=
21"...
ON CustomerNum
// Add customers into the list
ListAdd(CustomerList, ...
Customer.CustomerNum)
END
```

Browse with selection on the search key

1. Comparison filter according to a value

```
FOR EACH <File> WITH ...
<Key Item> [=][<=>] <Value> [<Direction>]
...
END
```

2. Comparison filter according to an interval of values

```
FOR EACH <File> WITH <Key Item> = ...
  <Minimum Value> TO <Maximum Value> [<Direction>]
  ...
END
```

or

```
FOR EACH <File> WITH ...
  <Minimum Value> <= <Key Item> <= ...
  <Max Value> [<Direction>]
  ...
END
```

```
// Comparison according to a
// value
FOR EACH Customer WITH ...
  CustomerName = "Smith"
  // Add customers into the list
  ListAdd(CustomerList, ...
    Customer.CustomerLastName)
END
// Comparison according to a
// value
FOR EACH Order WITH ...
  OrderDate <= "12/31/2003"
  // Add orders into the list
  ListAdd(OrdList, Orders.OrderNum)
END
// Comparison according to a
// list of values
FOR EACH Order WITH OrdersDate = ...
  "01/01/2003" TO "12/31/2003"
  // Add orders into the list
  ListAdd(OrdList, Orders.OrderNum)
END
// Comparison according to an
// interval of values
FOR EACH Order WITH "01/01/2003" ...
  <= OrderDate <= "12/31/2003"
  // Add orders into the list
  ListAdd(OrdList, Orders.OrderNum)
END
```

Browse with "Start with" generic search

```
FOR EACH <File> WHERE <Key Item>
  [= ...
  <Beginning of sought value> [<Direction>]
  ...
END
```

```
FOR EACH Customer WITH ...
  CustomerName [= "From"
  // Add customers into the list
  ListAdd(CustomerList, ...
    Customer.CustomerNum)
END
```

Code to run

The code to run is located between the **FOR EACH** and **END** statements.

Filter (syntaxes 2)

The general syntax of a filter has the following format:

```
"<Item Name> <Operators> <Item Value>"
```

For example:

```
"CustomerName > 'Smith' and ...
  ZipCode = 34101 or CustomerAge >=
  32"
```

The supported operators depend on the type of the items used in the condition:

<\>	Different	Valid for all the types
>	Greater than	Valid for all the types
>=	Greater than or equal to	Valid for all the types
<	Less than	Valid for all the types
<=	Less than or equal to	Valid for all the types
=	Strictly equal to	Valid for all the types
~=	Almost equal to	Valid for the "string" types only
]	Contains	Valid for the "string" types only
=]	Starts with	Valid for the "string" types only

Notes:

- The constant strings must be enclosed in simple quotes. For example: "CustomerName = "+Customer+""
- <Item name> must only contain letters, digits and underscore characters ("_"). If <Item name> contains other characters (quote, ...), the name of the item must be enclosed in double quotes. For example: "e_mail@]fr"

- The comparisons between strings are performed according to the ASCII value of the characters and not according to the lexicographic value ('a' > 'Z').
- The binary memos and the composite keys cannot be part of an <Item Value>.
- If <Item value> contains a simple quote (or a double quote), this simple quote (or double quote) must be preceded by a backslash character.

Exiting from FOR EACH loop

Several statements are available:

- RETURN: Exits from the **FOR EACH** loop and exits from the current process (or procedure).
- RESULT: Return a status report to the calling process. Exit from the **FOR EACH** loop and exit from the current process (or procedure).
- BREAK: Exit the **FOR EACH** loop and resume the current process.

Close is used to exit from the **FOR EACH** loop and to close the current window.

Caution: RETURN and RESULT cannot be used in the same process.

Behavior of the automatic file browse according to the mode used to exit from the loop

The automatic data file browse operation behaves differently whether the operation ends automatically or is triggered by the EXIT, RETURN or Close commands, ...

If the browse ends automatically:

- the position in the data file before the browse operation is restored.
- the possible filter required for the browse is disabled.

If the browse operation is stopped (BREAK, RETURN, RESULT, Close, ...):

- the position in the data file before the browse operation is not restored.
- the possible filter required for the browse is not disabled. It must be disabled manually (**HDeactivateFilter**).

Running the next iteration

To directly run the next iteration **without ending the code of the current iteration**, use the CONTINUE

statement:

```
FOR EACH <File> ON <Key Item>
...
IF <Condition> THEN CONTINUE
// Go back to the FOR EACH keyword
// Go to next record
...
END
```

4.1.5 FOR EACH/FOR ALL statement, parsing strings

The **FOR EACH** statement is used to browse the strings according to different methods:

- browsing sub-strings separated by a separator.
- Browsing the occurrences of a string inside another one

Note: Both **FOR EACH** and **FOR ALL** are accepted. The **FOR EACH** statement will be used in this documentation but it can be replaced by **FOR ALL**.

Syntax

Browse the sub-strings separated by a separator

```
FOR EACH STRING <Sub-string> OF ...
  <Initial String> ...
  [SEPARATED BY <Separator>] [<Direction>]
  ...
END
```

```
// Retrieves the list of libraries
loaded in memory
LibraryList is string
LibraryList = ListDLL()
ALibrary is string
// For each library
FOR EACH STR\xeb NE ALibrary ...
  OF LibraryList SEPARATED BY CR
```

This syntax is used to browse all the sub-strings of the <Initial string> separated by <Separator>. The browse is not performed if <Initial string> is an empty string.

For each iteration, the <Sub-string> variable is assigned with the current sub-string.

The behavior is undefined if the initial string or the separator is modified during the browse.

Note: If <Initial string> ends with the separator, <Sub-string> returns a string empty at the end. Otherwise, <Sub-string> corresponds to the last checked-out element.

Browsing the occurrences of a string inside another string

1. Without option

```
FOR EACH POSITION <Position> OF <Sought>...
  IN <Initial String> [<Direction>]
  ...
END
```

```
// The "C:\Exports.TXT" file contains the list of exported products separated by ";"
// Retrieve each product
ExportedProduct is string
ExportedProduct = fLoadText(...
  "C:\Exports.TXT")
FormerPos is int
CurrentPos is int
// For each product
FOR EACH POSITION CurrentPos ...
  OF ";" IN ExportedProduct
  // Adds the product into ProductList
  ListAdd(ProductList, ...
    ExportedProduct[[FormerPos ...
      + 1 TO CurrentPos - 1]]
  // Store the position
  // FormerPos = CurrentPos
END
```

This syntax will parse all the positions of the <Search> substring in <Initial String>.

For each iteration, the <Position> variable is assigned with the position of the current sub-string.

The behavior is undefined if the initial string or the sought string is modified during the browse.

2. With option

```
FOR EACH POSITION <Position> OF <Sought>...
  IN <Initial String> WITH <Options>
  ...
END
```

This syntax parses all the positions of the <Search> substring in <Initial String>.

For each iteration, the <Position> variable is assigned with the position of the current sub-string.

The behavior is undefined if the initial string or the sought string is modified during the browse.

Code to run

The code to run is located between the **FOR EACH** and **END** statements.

Parsing Unicode strings

<Sub-string>, <Initial string>, <Separator> and <Search> can correspond to:

- ANSI strings.
- UNICODE strings.

However, the ANSI strings and the UNICODE strings cannot be used in the same syntax.

4.1.6 FOR EACH/FOR ALL statement, control browsing

FOR EACH is used to perform different types of browse operations on controls (list boxes, tables, loopers):

- Browsing the control elements.
- Browsing the values of the selected elements.
- Browsing the subscripts of the selected elements.

Note: Both **FOR EACH** and **FOR ALL** are accepted. The **FOR EACH** statement will be used in this documentation but it can be replaced by **FOR ALL**.

Syntax

Browsing the control elements

```
FOR EACH ROW OF <Control>
```

```
...
```

```
END
```

```
// For each user listed in UserList
FOR EACH ROW OF UserList
  // Send an email
  // SendMessage(ColName, ...
  // ColIDmail)
END
```

This syntax is used to browse all the rows found in a list box, table or looper.

For each row browsed:

- <Control name> returns the subscript of the current row.
- <Control name>[<Control name>] returns all the columns separated by TAB characters.
- <Column name> returns the value of the column for the row currently browsed.

The browse operation has no effect on the current selection.

The behavior is undefined if the number of control elements is modified during the browse.

Note: When browsing the table rows, the display of this table is locked. **MultitaskRedraw** is ignored.

Browsing the selected elements in a control

```
FOR EACH SELECTED ROW OF <Control>
```

```
...
END
```

```
// For each user selected in the
// Table control named UserTable
FOR EACH SELECTED ROW OF UserTable
// Send an email
SendMessage(ColName, ColIDmail)
END
```

This syntax is used to browse all the selected rows found in a List Box, Table or Looper control.

For each row browsed:

- <Control name> returns the subscript of the current row.
- <Control name>[<Control name>] returns all the columns separated by TAB characters.
- <Column name> returns the value of the column for the row currently browsed.

The browse operation has no effect on the current selection.

The behavior is undefined if the number of control elements is modified during the browse.

Browsing the subscripts of the selected elements

```
FOR EACH SELECTED ROW <Subscript> ...
```

```
OF <Control>
```

```
...
END
```

This syntax is used to browse all the selected rows found in a List Box, Table or Looper control.

For each row read, the <Subscript> variable returns the subscript of the selected row.

To access a specific column's value, use the <Column>[<Subscript>] syntax

The table's current row is always changing during the browse operation.

The behavior is undefined if the number of control elements is modified during the browse.

Browsing the table rows

When browsing the table rows:

- the display of this table is locked. **MultitaskRedraw** is ignored.
- the selected rows and/or the current row must not be modified (**TableSelectMinus**, **TableSelectPlus**, etc.).
- for a browsing table, in the browse loop, the current record is the record processed by the browse.

4.1.7 FOR EACH/FOR ALL statement, array browsing

The FOR EACH statement is used to perform different types of browse operations on the arrays:

- Browsing the array elements.
- Browsing the values of the array elements.

Note: Both FOR EACH and FOR ALL are accepted. The FOR EACH statement will be used in this documentation but it can be replaced by FOR ALL.

Syntax

Browsing the elements of an array

```
FOR EACH ELEMENT <Variable> OF ...
```

```
<Array> [<Direction>]
```

```
...
```

```
END
```

```
// Browse the elements found in an
// array of reals to calculate
// the sum
// Fill the array
ArrCalc is array of 3 reals
ArrCalc[1] = 12.5
ArrCalc[2] = 10
ArrCalc[3] = 7.5
// Calculate the sum
// AnElement is real
// TotalSum is real
FOR EACH ELEMENT AnElement OF ...
ArrCalc TotalSum += AnElement
END
```

For each iteration, <Variable> directly refers to the current element in the array. If the value of <Variable> is modified, the current element in the array is modified.

When exiting from the loop (standard exit or via the BREAK statement), the value of the last element read is assigned to <Variable> but <Variable> does not directly refer to the array element anymore.

All types of arrays are available: automatic, fixed, dynamic.

The arrays can have several dimensions.

The behavior is undefined if the number of elements is modified in the browse loop.

Browsing the values of an array elements

```
FOR EACH ELEMENT (<Value>) OF
```

```
<Array> [<Direction>]
```

```
...
```

```
END
```

For each iteration, the value of the element browsed is assigned to the <Value> variable. If the value of <Value> is modified, the current element in the array is not modified.

All types of arrays are available: automatic, fixed, dynamic.

The arrays can have several dimensions.

The behavior is undefined if the number of elements is modified in the browse loop.

4.1.8 FOR EACH/FOR ALL statement, browse associative arrays

FOR EACH is used to perform different types of browse operations on the associative arrays:

- Browsing the elements of the associative array.
- Browsing the values of the elements found in the associative array.
- Browse the "key" elements of the associative array.

Note: Both **FOR EACH** and **FOR ALL** are accepted. The **FOR EACH** statement will be used in this documentation but it can be replaced by **FOR ALL**.

Syntax

Browsing the elements of an array

```
FOR EACH ELEMENT <Variable> OF ...
    <Array> [<Direction>]
```

```
...
END
```

```
// Declare an
// associative array of integers
// Array indexed on strings
// and without duplicates
aaIDPerCust is array ...
    associative of int
aaIDPerCust["A"] = 55
// add the ID of customer "A"
aaIDPerCust["B"] = 321
// add the ID of customer "B"
aaIDPerCust["A"] = 56
// modify the ID of customer "A"
// nIdentifier is int
// browse all the identifiers
// 56
// 321
FOR EACH ELEMENT nIdentifier ...
    OF aaIDPerCust ...
    Trace(nIdentifier)
END
```

For each iteration, <Variable> directly refers to the current element in the array. If the value of <Variable> is modified, the current element in the array is modified.

When exiting from the loop (standard exit or via the **BREAK** statement), the value of the last element read is assigned to <Variable> but <Variable> does not directly refer to the array element anymore.

Browsing the elements of an array according to the index key

```
FOR EACH ELEMENT <Variable>, <KeyVariable> OF
    <Array> [<Direction>]
```

```
...
END
```

For each iteration:

- <Variable> directly refers to the current element in the array. If the value of <Variable> is modified, the current element in the array is modified.
- <KeyVariable> contains the value of the element key. This value is read-only and it cannot be modified.

When exiting from the loop (standard exit or via the **BREAK** statement), the value of the last element read is assigned to <Variable> but <Variable> does not directly refer to the array element anymore.

Browsing the <key> elements of the array

```
FOR EACH ELEMENT <Variable> OF <Array> =
    <Key> [<Direction>]
```

```
...
END
```

This syntax browses through all the array elements with the specified <Key> value. For each iteration, <Variable> directly refers to the current element in the array. If the value of <Variable> is modified, the current element in the array is modified.

This syntax is useful when browsing associative arrays with duplicates. In an associative array without duplicates, the number of elements browsed can be 0 or 1. In an associative array with duplicates, the number of elements browsed can be 0 or N.

The array elements are browsed in the order they were added (no direction option).

When exiting from the loop (standard exit or via the **BREAK** statement), the value of the last element read is assigned to <Variable> but <Variable> does not directly refer to the array element anymore.

Browsing the values of the array's key elements according to the index key

```
FOR EACH ELEMENT <Variable> <KeyVariable> OF
<Array> = <Key> [<Direction>]
...
END
```

This syntax browses through all the array elements with the specified <Key> value.

For each iteration :

- <Variable> directly refers to the current element in the array. If the value of <Variable> is modified, the current element in the array is modified.
- <KeyVariable> contains the value of the element key. This value is read-only and it cannot be modified.

This syntax is useful when browsing associative arrays with duplicates. In an associative array without duplicates, the number of elements browsed can be 0 or 1. In an associative array with duplicates, the number of elements browsed can be 0 or N.

The array elements are browsed in the order they were added (no direction option).

When exiting from the loop (standard exit or via the BREAK statement), the value of the last element read is assigned to <Variable> but <Variable> does not directly refer to the array element anymore.

4.1.9 SWITCH statement

The SWITCH conditional statement is used to choose the action that will be run according to the value of an expression.

```
SWITCH Quantity
CASE 1: Comment = "Promotions"
CASE 2: Comment = "Buy"+ "two"+...
      "the second one is free"
OTHER CASE: Comment = ""
END
```

Syntax

```
SWITCH <Comparison Variable>
Case <Expression1> : <Action1>
Case <Expression2> :
    <Action2.1>
    <Action2.2>
Case <Expression3>
    <Action3>
Case <Expression4>, <Expression5> :
<Action4>
Case <Expression6>, <Expression7> :
    <Action5>
Case <Expression8>, <Expression9>
    <Action6>
```

```
Case <Min Expression> TO : <Action7>
Case TO <Max Expression> : <Action8>
Case <Min Expression> TO <Max Expression> :
    <Action9>
Case = <Expression> : <Action10>
Case ~ = <Expression> : <Action11>
Case ~~ <Expression> : <Action12>
Case [= <Expression> : <Action13>
Case > <Expression> : <Action14>
Case >= <Expression> : <Action15>
Case < <Expression> : <Action16>
Case <= <Expression> : <Action17>
Case <Min Expression> <= * <= <Max Expression> :
    <Action18>
Case <Min Expression> < * <= <Max Expression> :
    <Action19>
Case <Min Expression> <= * < <Max Expression> :
    <Action20>
Case <Min Expression> < * < <Max Expression> :
    <Action21>
...
[ CASE ELSE: <Action Case Else>
OR
OTHER CASE:
    <Action Other Case>
OR
OTHER CASE
    <Action Case Else> ]
END
```

OTHER CASE keyword

OTHER CASE must be the last statement used in SWITCH.

BREAK SWITCH statement

The BREAK SWITCH statement allows you to exit from a SWITCH statement and to resume the current process.

4.1.10 IF statement

The conditional IF statement allows you to choose the action to run according to a condition.

```
IF CustomerAge > 60 THEN...
    Elderly += 1
ELSE IF Customer.Age > 18 THEN...
    Adult += 1
ELSE IF Customer.Age > 4 THEN...
    Child += 1
ELSE Baby += 1
END
```

Syntax

• Syntax 1

```
IF <Condition> THEN
    <Action if condition is True>
[ELSE
    <Action if condition is False> ]
END
```

• Syntax 2

```
IF <Condition> THEN
    <Action if condition is True>
[ELSE <Action if condition False> ]
```

• Syntax 3

```
IF <Condition> THEN <Action if condition
True>
[ELSE
    <Action if condition is False> ]
END
```

• Syntax 4

```
IF <Condition> THEN <Action if condition is
True> [ELSE <Action if condition is False>]
```

• Syntax 5

```
IF <Condition 1> THEN
    <Action if condition 1 is True>
[ELSE IF <Condition 2> THEN
    <Action if condition 2 is True>
[ELSE IF <Condition 3> THEN
    <Action if condition 3 is True>
[...]]]
END
```

Condition

<Condition> can take the following format:

- <Value> = <Expression>
- <Value> < <Expression>
- <Value> <= <Expression>
- <Value> > <Expression>
- <Value> >= <Expression>
- <Value> = <Min Expression> TO <Max Expression>
- <Min Expression> <= <Value> <= <Max Expression>
- <Value> IN (<Expression 1>, <Expression 2>, ..., <Expression N>)
Note: in this case, all the expressions are checked.
- <Value> _IN_ (<Expression 1>, <Expression 2>, ..., <Expression N>)

Note: In this case, as soon as an expression is False, none of the following expressions are evaluated.

Composite condition

The **AND** and **OR** keywords are used to perform logical operations and to create composite conditions.

```
IF Customer.City = "Boston" ...
    AND Customer.Title = "Mister"
    THEN BostonMen ++
    // Number of men living
    // in Boston
END
IF Customer.City = "Boston"...
    OR Customer.City = "Miami" THEN
    BostonMiami ++
    // Number of customers living
    // either
    // in Boston, or in Miami
END
```

The conditions made of **AND** and **OR** are entirely evaluated. For example:

A > 10 AND B < 20

Even if the first condition (A > 10) is false, the second condition (B < 20) will be checked.

To optimize the evaluation of composed conditions, use the **_AND_** and **_OR_** keywords. Therefore, if the first condition (A > 10) is False, the second condition (B < 20) will not be evaluated.

ELSE without IF

ELSE cannot be used without the corresponding **IF** statement.

4.1.11 WHILE statement

In a **WHILE** statement, the expression is evaluated at each beginning of the statement block.

The process loops as long as the condition expression is True.

The program will exit from the statement block when the condition is False.

```
List = INIRead("Examples",...
    " ", " ", IniFile)
Keyword = ExtractString(List, Num, CR)
WHILE Keyword <> " "
    nb = nb + 1
    ExplName = INIRead(...
        "Projects installed",...
        Keyword, " ", IniFile)
    Keyword = ExtractString(List, ...
        Nb+1, CR)
END
```

Syntax

```
WHILE <Condition>
    <Action if condition is True>
END
```

Code to run

The code to run is located between the **WHILE** and **END** statements.

Exiting from a WHILE loop

Several statements are available:

- **RETURN**: Exit from the **WHILE** loop and exit from the current process (or procedure).
- **RESULT**: Return a status report to the calling process. Exit from the **WHILE** loop and exit from the current process (or procedure).
- **BREAK**: Exit from the **WHILE** loop and resume the current process.

Close is used to exit from the **WHILE** loop and to close the current window.

Caution: **RETURN** and **RESULT** cannot be used in the same process.

Loop without end

During the compilation of the project, a **WHILE** loop without an obvious end (loop without **BREAK**, **RETURN** or **RESULT** statement) is signaled by a warning.

Running the next iteration

To directly run the next iteration **without ending the code of the current iteration**, use the **Continue** sta-

tement:

```
WHILE <Condition>
    ...
    IF <Condition> THEN CONTINUE
        // Go back to the WHILE keyword
    ...
END
```

Composite condition

The **AND** and **OR** keywords are used to perform logical operations and to create composite conditions. For example:

```
WHILE Price < 100 AND ...
    ProductType = "AA" ...
    NbProduct ++
    // Number of products whose price
    // is less than
    // $100 and whose type is "AA"
END
WHILE Price < 100 OR Price > 500
    NbProduct ++
    // Number of products whose price
    // is included between 100
    // and 500 €
END
```

The conditions made of **AND** and **OR** are entirely evaluated. For example:

A > 10 AND B < 20

Even if the first condition (**A > 10**) is false, the second condition (**B < 20**) will be checked.

To optimize the evaluation of composed conditions, use the **_AND_** and **_OR_** keywords. Therefore, if the first condition (**A > 10**) is False, the second condition (**B < 20**) will not be evaluated.

4.2 Simple statements

The simple statements are as follows:

- **CONTINUE**: Go to the beginning of the next iteration without ending the code of the current iteration
- **RESULT**: Exit from the current process (or procedure) and return a status report
- **RETURN**: Exit from the statement block and exit from the current process (or procedure)
- **BREAK**: Exit from the statement block and resume the current process

4.2.1 CONTINUE statement

The **CONTINUE** statement is used to directly go back to the beginning of the next iteration without ending the code of the current iteration.

CONTINUE can be used in the following types of loops:

- **FOR**.
- **FOR EACH**.
- **LOOP**.
- **WHILE**.

Syntax

FOR statement

```
FOR <Control variable> = <Initial value> TO
<Final value> [STEP <x>]
...
IF <Condition> THEN CONTINUE
...
END
```

If <Condition> is True:

- The code found after the **CONTINUE** statement is not run.
- The loop is run from the beginning of the FOR statement.
- The <Control variable> is incremented.

FOR EACH statement

```
FOR EACH <File> ON <Key Item>
...
IF <Condition> THEN CONTINUE
...
END
```

If <Condition> is True:

- The code found after the **CONTINUE** statement is not run.
- The loop is run from the beginning of the FOR EACH statement.
- The move to the next record is automatically performed.

LOOP statement

```
LOOP
...
IF <Condition> THEN CONTINUE
...
END
```

If <Condition> is True:

- The code found after the **CONTINUE** statement is not run.
- The loop is run from the beginning of the LOOP statement.

WHILE statement

```
WHILE <Condition 1>
...
IF <Condition 2> THEN CONTINUE
...
END
```

If <Condition 2> is True:

- The code found after the **CONTINUE** statement is not run.

- The loop is run from the beginning of the WHILE statement.

4.2.2 RETURN statement

The **RETURN** statement is used to exit from a statement block and to exit from the current process (or procedure).

The **RETURN** statement can be used in:

- a procedure.
- a FOR loop.
- a FOR EACH loop.
- a LOOP loop.
- a WHILE loop.

Syntax

```
Procedure
Procedure <Procedure name> ([[Parameter]])
IF <Condition> THEN RETURN
...
END
```

FOR statement

```
FOR <Control variable> = <Initial value> TO
<Final value> [STEP <x>]
IF <Condition> THEN RETURN
END
```

FOR EACH statement

```
FOR EACH <File> ON <Key Item>
IF <Condition> THEN RETURN
END
```

LOOP statement

```
LOOP
...
IF <Condition> THEN RETURN
...
END
```

WHILE statement

```
WHILE <Condition 1>
...
IF <Condition> THEN RETURN
...
END
```

The following operations are performed if <Condition> is True:

- Exit from the statement block.
- Exit from the current process (or procedure).

Other statements used to exit from a loop or from a procedure

Several statements are available:

- **RESULT**: Return a status report to the calling process. Exit from the loop and exit from the current process (or procedure).
- **BREAK**: Exit from the loop and resume the current process (or procedure).

Close is used to exit from the loop (or from the procedure) and to close the current window.

Caution: **RETURN** and **RESULT** cannot be used in the same process.

4.2.3 RESULT statement

The **RESULT** statement is used to exit from the current process (or procedure) and to return a status report.

The **RESULT** statement can be used in:

- a procedure.
- a FOR loop.
- a FOR EACH loop.
- a LOOP loop.
- a WHILE loop.

Syntax

Procedure

```
Procedure <Procedure name> ([[<Parameter>]])
  IF <Condition> THEN
    RESULT <Value to return>
  ELSE
    RESULT <Value to return>
  END
```

FOR statement

```
FOR <Control variable> = <Initial value> TO
<Final value> [STEP <x>]
  IF <Condition> THEN RESULT <Value>
END
```

FOR EACH statement

```
FOR EACH <File> ON <Key Item>
  ...
  IF <Condition> THEN RESULT <Value>
END
```

LOOP statement

```
LOOP
  ...
  IF <Condition> THEN RESULT <Value to Return>
  ...
END
```

WHILE statement

```
WHILE <Condition 1>
  ...
  IF <Condition> THEN RESULT <Value>
  ...
END
```

The following operations are performed if <Condition> is True:

- Return a status report to the calling process.
- Exit from the statement block.
- Exit from the current process (or procedure).

Other statements used to exit from a loop or from a procedure

Several statements are available:

- **RETURN**: Exit from the loop and exit from the current process (or procedure).
- **BREAK**: Exit from the loop and resume the current process (or procedure).

Close is used to exit from the loop (or procedure) and to close de current window or page.

Caution: **RETURN** and **RESULT** cannot be used in the same process.

4.2.4 BREAK statement

BREAK is used to exit from a statement block and to run the rest of the current process.

The **BREAK** statement can be used in the following types of loops:

- FOR.
- FOR EACH.
- LOOP.
- WHILE.

Note: The **BREAK** statement cannot be used to exit from a procedure. Use the **RETURN** and **RESULT** keywords.

Syntax

FOR statement

```
FOR <Control variable> = <Initial value> TO
<Final value> [STEP <x>]
  ...
  IF <Condition> THEN BREAK
  ...
END
<Rest of the process>
```

FOR EACH statement

```
FOR EACH <File> ON <Key Item>
  ...
  IF <Condition> THEN BREAK
  ...
END
<Rest of the process>
```

LOOP statement

```
LOOP
  ...
IF <Condition> THEN BREAK
  ...
END
<Rest of the process>
```

WHILE statement

```
WHILE <Condition>
  ...
IF <Condition> THEN BREAK
  ...
```

END

<Rest of the process>

The following operations are performed if <Condition> is True:

- Exit from the statement block.
- Runs the rest of the current process.

Other statements used to exit from a loop

Several statements are available:

- RETURN: Exit from the loop and exit from the current process (or procedure).
- RESULT: Return a status report to the calling process. Exit from the loop and exit from the current process (or procedure).

Close is used to exit from the loop and to close de current window.

Caution: RETURN and RESULT cannot be used in the same process.

5. RESERVED WORDS

Several WLanguage words are keywords, used to perform some specific actions:

- External
- MyPage
- Modulo
- MyWindow
- MySource
- MySelf
- MyPopupControl
- MyFile
- STOP
- MyReport
- MyParent

5.1 External

The **EXTERN** keyword is used to:

- include a text file containing WLanguage commands in an application (see use 1).
- declare an external object (see use 2).

Use 1: including a text file in an application/site

Including a text file with **EXTERN** has the same effect than copying a text file in the code editor.

```
// Include the file named
// "WinConst.wl"
// Standard constants of Windows
EXTERN "WinConst.wl"
```

Syntax

EXTERN <File Name>

The file name corresponds to the name of the text file containing WLanguage code that must be included in the application/site. This is a standard text file, created by any text editor.

A full path can be specified ("C:\WDPProject\MyConst.wl" for instance). If the path is not specified, the file will be sought:

- in the directory of the project.
- in the "Personal\Extern" directory of WinDev/WebDev.

Including constants

The inclusion of files is very useful to describe some constants common to several projects or used by the operating system. Use **EXTERN** to include in the code a file containing the common constants.

Several files defining the constants are supplied with WinDev/WebDev:

- **WinConst.wl**: Standard constants of Windows.
- **Limits.wl**: Constants corresponding to the limits of the WinDev/WebDev data types.
- **ListeDefinitionHF.wl**: HFSQL constants used for the log process.
- **Except.wl**: Constants used for managing the exceptions
- **KeyConst.wl**: Standard constants of Windows used for the keyboard keys. These constants can be used by **KeyPressed**, in the "Key Down" or "Key Up" optional processes with the `_EVE.wParam` variable.

Use 2: Declaring an external object

EXTERN is used to declare a variable that will exist only when the application/site is run.

For example, **EXTERN** can be used to handle a window/page found in an external library (loaded by **LoadWDL**).

```
EXTERNMyWindow
LoadWDL("User1.WDL")
Open(MyWindow)
```

Syntax

EXTERN <Name of External Object>

<Name of External Object> corresponds to the name of the external object (file, variable, constant, ...) to declare.

5.2 MyWindow

MyWindow is used to handle the current window. When the window is run, **MyWindow** is replaced by the window (and not by the name of the window). Therefore, **MyWindow** can be used like a window.

Benefit: **MyWindow** enables you to write local code (control, button, ...) or global code (global procedure, class, ...) that is independent of the current window.

```
// Retrieve the title of the current
window
WindowTitle = MyWindow..Title
```

Syntax

MyWindow

Handling the current window

MyWindow is always replaced by the current window. For example:

- Using the **..Title** property

```
MyWindow..Title = ...
    "Enter your data"
CurrentTitle = MyWindow..Title
```

5.3 MyPage

MyPage is used to handle the current page. When the page is run, **MyPage** is replaced by the page (and not by the name of the page). Then, **MyPage** can be used like any page.

Benefit: **MyPage** enables you to write local code (control, button, ...) or global code (global procedure, class, ...) that is independent of the current page.

```
// Retrieve the title of the current
page
PageTitle = MyPage..Title
```

Syntax

MyPage

Handling the current page

MyPage is always replaced by the current page. For example:

- Using the **..Title** property

```
MyPage..Title = ...
    "Enter your data"
CurrentTitle = MyPage..Title
```

- Passing a parameter:

```
CallProcedure(MyWindow)
```

MyWindow can only be used in the processes that handle the current window (in the processes associated with a control, a window, ...). **MyWindow** cannot be used in a report.

Using MyWindow in a procedure

MyWindow can be used in a local procedure or in a global procedure only if the procedure handles the current window. In this case, **MyWindow** refers to the current window.

For a local procedure, **MyWindow** corresponds to the window to which the procedure belongs.

WLanguage functions and current window

To specify the current window in the WLanguage functions that accept a window name in parameter, use:

- an empty string.
- the **MyWindow** keyword directly.

- Passing a parameter:

```
CallProcedure(MyPage)
```

MyPage can only be used in the processes that handle the current page (processes associated with a control, with a page, ...). **MyPage** cannot be used in a report.

Using MyPage in a procedure

MyPage can be used in a (local or global) procedure only if the procedure handles the current page. In this case, **MyPage** refers to the current page.

For a local procedure, **MyPage** corresponds to the page to which the procedure belongs.

WLanguage functions and current page

To specify the current page in the WLanguage functions that accept a page name in parameter, use:

- an empty string.
- **MyPage** directly.

5.4 MySource

The **MySource** keyword is used to handle the current data source (file, view or query). When the project is run, this keyword is automatically replaced by the current data source.

Benefit: **MySource** enables you to:

- make the code independent of the current data source.
- access the value of an item of the current record in the current data source.
- retrieve the name of the data source to perform a HFSQL process.

```
// Retrieve the name of the current
data source for the report
DataSource = MySource..Name
```

```
// Retrieve the name of the customer
currently printed
IF MySource.CustName = "Martin" THEN
    TotalBT..BrushColor = iLightRed
END
```

Syntax

MySource

Limit

MySource can only be used in the processes associated with:

- a report.
- a browsing table.

5.5 Modulo

Returns the remainder of a division.

Note: **Modulo** can be used as a keyword or as a WLanguage function.

```
// Retrieve the remainder of the
division
// DivisionReminder = Modulo(21,4)
// DivisionReminder is equal to 1
// Equivalent: DivisionReminder = 21
Modulo 4
```

Syntax

Modulo function

```
<Division Remainder> = Modulo ...
    (<<Dividend>, <Divisor>)
```

Modulo keyword

```
<Division Remainder> = <Dividend> ...
    Modulo <Divisor>
```

5.6 MySelf

MySelf is used to handle the current control. When the project is run, **MySelf** is replaced by the control (and not by the name of the control). Then, **MySelf** is used like a control.

Benefit: **MySelf** enables you to write local code (control, button, ...) or global code (global procedure, class, ...) that is independent of the current control.

```
-- whenever a combo box is modified
// Call the UpperMask procedure
UpperMask ()
--UpperMask procedure
// Mask: 1st letter uppercase
// (editable combo box)
Procedure UpperMask ()
    Value = MySelf
    IF Size(Value) > 1 THEN
        CursorPos = MySelf..Cursor
        MySelf = ...
```

```
Upper(Value[[1]])+...
Lower(Value [[2 to Length(...
Value]])
MySelf..Cursor = CursorPos
END
```

Syntax

MySelf

Handling the current control

MySelf is always replaced by the current control. For example:

- Retrieving the value of the current control:

```
ControlValue = MySelf
```

- Modifying the value of the current control:

```
MySelf = "Smith"
```

- Using a property (Caption property for example):

```
MySelf..Caption = "Name of custo-
mers"
ControlCaption = MySelf..Caption
```

- Passing a parameter:

```
CallProcedure(MySelf)
```

A WLanguage error occurs if there is no current control.

Using MySelf in a procedure

MySelf can be used in a local procedure or in a global procedure only if the procedure is called in a process associated with a control. In this case, **MySelf** refers to the current control.

5.7 MyPopupControl

MyPopupControl is used to handle the control that opened a popup window. At run time, **MyPopupControl** is replaced by:

- the control that opened the popup window (which means the control from which **OpenPopup** or **OpenPopupPosition** was called)
- the control specified in **OpenPopupPosition** if the display mode was set to *poAccordingToControl*.

In any case, **MyPopupControl** is replaced at run time by the corresponding control (and not by the name of the control). Then, **MyPopupControl** is used like a control.

Benefit: **MyPopupControl** enables you to write local code (control, button, ...) or global code (global procedure, class, ...) that is independent of the current popup window.

Syntax

MyPopupControl

Handling the control that opened the popup window

MyPopupControl is always replaced by the control that opened the popup window. Therefore, it can be handled from the code of the popup window.

Myself is used to make a procedure "generic": the name of the control is not spelled out. This procedure can be called by several controls.

WLanguage functions and current control

To specify the current control in the WLanguage functions that accept a control name in parameter, use:

- an empty string.
- **Myself** directly.

For example:

```
If MyPopupControl..Type = ...
    typButton THEN
    MyPopupControl..Caption = ...
        "Selection in progress"
END
```

Passing a parameter

```
CallProcedure(MyPopupControl)
```

MyPopupControl can only be used in the processes that handle the current popup window (in the processes associated with a control for example). **MyPopupControl** cannot be used in a report or in a window that is not a popup report or a popup window.

Using MyPopupControl in a procedure

MyPopupControl can be used in a local or global procedure only if the procedure is called in a process associated with a control of the popup window or in a process of the popup window. In this case, **MyPopupControl** refers to the control that opened the popup window.

MyPopupControl is used to make a procedure "generic": the name of the control is not spelled out. This procedure can be called by several controls.

WLanguage functions and current window

To specify the control that opened the current popup window in the WLanguage functions that accept a control name in parameter, use **MyPopupControl** directly.

5.8 MyReport

MyReport is used to handle the current report. When running the project, **MyReport** is replaced by the report and not by the name of the report. **MyReport** is then used like a report.

Benefit: **MyReport** enables you to write some local code (control, item, ...) or some global code (global procedure, class, ...) that is independent of the current report.

```
// Retrieve the name of the current
report
MyReport = MyReport..Name
```

Syntax

MyReport

Handling the current report

MyReport is always replaced by the current report.

For example:

- Using a property (**BottomMargin** for example):

```
MyReport..BottomMargin = 15
ResMargin = MyReport..BottomMargin
```

- Passing a parameter:

```
CallProcedure(MyReport)
```

MyReport can only be used in the processes that handle the current report (processes associated with a control, an item, ...). **MyReport** cannot be used in a window.

Using MyReport in a procedure

MyReport can be used in a local or global procedure only if the procedure handles the current report. In this case, **MyReport** refers to the current report.

For a local procedure, **MyReport** corresponds to the report to which the procedure belongs.

5.9 MyFile

In a trigger, **MyFile** is used to identify the data file that releases the trigger. At run time, this keyword is automatically replaced by the current data file.

Benefit: **MyFile** enables you to create some generic triggers, without having to make indirections on the **H.FileName** variable. This allows you to handle the data file, its properties or its items.

5.10 MyParent

MyParent is used to handle:

- the current supercontrol from an element of the supercontrol.

When running the window or the page, **MyParent** is replaced by the supercontrol (and not by the name of the supercontrol). Then, **MyParent** is used like a supercontrol.

Benefit: **MyParent** enables you to make the code of the supercontrol controls (control, button, ...) or the code of the functions associated with the supercontrol independent of the supercontrol name.

- the current report from one of the report controls.

- the current window/page from one of the controls of this window/page.

```
// Code of selection button
MyFile=fSelect(...
  fExtractPath(...
    FileName, fDirectory), ...
  fExtractPtah(FileName, ...
    fFileName+fExtension), ...
  "Select a file", ...
  "All files (*.*)" + TAB + ...
  "**.*", "")
// The value of the supercontrol
// corresponds
// to the selected file
MyParent..Value = FileName
```

Syntax

MyParent

Handling the current supercontrol

MyParent is always replaced by the current supercontrol. For example, passing a parameter:

```
CallProcedure(MyParent)
```

MyParent can only be used in the processes that

handle the current supercontrol (processes associated with a control found in the supercontrol, with the procedures associated with the supercontrol).

Using MyParent in a procedure

MyParent can be used in a procedure associated with the current supercontrol. In this case, **MyParent** refers to the current supercontrol.

5.11 STOP (calling the debugger)

STOP is used to call the WLanguage debugger when running a test in the editor (see the online help for more details). Once this keyword has been called, the current test is run in the debugger.

```
// Window opening
Open(EditWindow)
STOP // Start the debugger
```

Syntax

Start the debugger

STOP

Starting the debugger with condition

STOP IF <Condition>

Start mode of the debugger

To start the debugger during a test, you can use:

- **STOP** in the code editor.

- a breakpoint in the code editor ("Edit .. Breakpoint" or [Ctrl] [B] shortcut, mouse click inside the margin).
- the [Ctrl][Break] key combination when running the test (except in WebDev).

Tracing and debugging a project

To trace a project from the beginning of its execution, on the "Project" pane, in the "Test mode" group, click "Test mode" and select "Trace the project" ([Alt][F9] shortcut).

Executable

STOP has no action in the code of an executable.

6. PROCEDURE AND FUNCTION

6.1 Overview

A **procedure** is used to associate an identifier with a statement block. Then, the procedure can be called in a process.

A **function** is used to define a sub-program that performs several statements before returning a value to the calling program.

Difference between a procedure and a function:

- a **procedure** returns no result.
- a **function** returns a result.

In WLanguage, there is no distinction between the procedures and the functions. The procedures and the functions are managed in the same way. A procedure, like a function, may (or may not) return a result.

The procedures and the functions can have a fixed or variable number of parameters.

The declaration syntax is the same for the procedures and for the functions. See “Declaring a procedure/a function”, page 98 for more details.

The calling syntax is the same for the procedures and for the functions. See “Calling a procedure/function”, page 99 for more details.

You have the ability to create overloaded procedures. See “Overload a WLanguage function”, page 101 for more details.

A procedure (or a function) can be global or local. See the next paragraph for more details.

6.2 Global and local procedure/function

6.2.1 Definition

Two types of procedures are available :

- **Global procedure:** can be used in all the processes of the project. This global procedure is stored in the project.
- **Local procedure:** can be used in all the processes related to the object (window, page or report) where the procedure was declared. This local procedure is stored in the object.

Important: Do not declare two procedures with the same name (especially a global procedure and a local procedure).

Reminder: WLanguage does not differentiate between procedures and functions. The syntaxes for declaring and using the procedures also apply to the functions.

6.2.2 Global procedure

Global procedures and sets of procedures

The global procedures are stored in sets of procedures, associated with the project.

The global procedures **declared in the project initialization process** can be called from any process:

- of the project.

- of the windows/pages (processes of the window/page, of its controls and associated local procedures).
- of the reports (processes of the report, its associated controls and associated local procedures).

Advice: If a procedure is used by only one window/page, it is better to declare this procedure as local to the window/page. The window/page is then “independent”.

Global server and browser procedures

In WebDev, a project corresponding to a dynamic site contains at least two sets of procedures (a “server” one and a “browser” one).

During the project creation:

- the set of server procedures is named “Global procedures of <Project Name>” (“W.DG: file) by default
- the set of default browser procedures is named “Global procedures of <Project Name>_Browser” (“W.DWN” file)

Note: a set of global browser procedures is converted into a JavaScript set of procedures. This set of Javascript procedures is saved in the “<Project Name>_WEB” directory and it is named “<Project Name>.JS”.

Limit: The global server procedures cannot call the global browser procedures (and conversely).

Creating a global procedure

1. From the code editor

To create a global procedure from the code editor:

- Method 1: On the "Code" pane, in the "Procedure" group, expand "New" and select the "New global procedure" option ([Shift] [F4] keyboard shortcut).
- Method 2: Select "Create a procedure .. Create an empty global procedure" in the popup menu.

In WebDev:

- If the current code is a server code, the global procedure is created in the set of server procedures. If the current code is a browser code, the global procedure is created in the set of browser procedures.
- To directly create a global server procedure, on the "Code" pane, in the "Procedure" group, expand "New" and select "New global (server) procedure".
- To directly create a global browser procedure, on the "Code" pane, in the "Procedure" group, expand "New" and select "New global (browser) procedure".

Note: To transform a section of code already entered into a global procedure:

1. Select the corresponding code.
2. Select "Create a procedure .. Create a global procedure containing the selected code" from the popup menu.
3. Specify the name of the procedure. The procedure is created, the corresponding code is replaced by the call to the procedure with the necessary parameters.

2. From the "Project explorer" pane

To create a global procedure from the "Project explorer" pane:

1. Select the "Procedures" folder.
2. Select the requested set of procedures.
3. Display the popup menu of the set of procedures and select "New global procedure".
4. Enter the name of the new global procedure and validate.

Note: you can display the global procedure list in the project explorer by double-clicking the "Procedure" folder ([Ctrl] [F8] keyboard shortcut) in the code editor.

Deleting a global procedure

1. From the code editor

To delete a global procedure from the code editor:

1. Click the procedure bar in the code editor.
2. Select "Delete" from the popup menu.

2. From the "Project explorer" pane

To delete a global procedure from the "Project explorer" pane:

1. Select the "Procedures" folder.
2. Select the requested set of procedures.
3. Display the popup menu of the set of procedures and select "Delete".

Test of a global procedure

1. From the code editor

To run the test of a global procedure:

1. Display the global procedure in the editor.
2. Click the [GO] button.
3. The description window of the test of the procedure is displayed. This window is used to:
 - Enter the different parameters of the procedure.
 - Run the test of the procedure ([Run the procedure test] button).
 - Check the return value.

2. From the "Project explorer" pane

To run the test of a global procedure:

1. Select the "Procedures" folder in the project explorer.
2. Select the name of the global procedure whose test must be run.
3. Select "Run the procedure test" from the popup menu.
4. The description window of the test of the procedure is displayed. This window is used to:
 - Enter the parameters of the procedure.
 - Run the test of the procedure ([Run the procedure test] button).
 - Check the return value.

Public or private global procedure

By default, a global procedure is public: it can be accessed from any piece of code.

In some cases, it may be useful to restrict the access to a global procedure by making it "Private". In this case, the global procedure can only be accessed from another global procedure found in the set of procedures.

To modify the access mode to a global procedure:

1. Select the global procedure in the "Project Explorer" pane.
2. In the popup menu of the procedure, select the new access mode:
 - public.
 - private.

The color of the procedure bar changes according to its access mode:

- bar start is red: private procedure.
- bar start is normal: public procedure.

6.2.3 Local procedure

Procedures local to a window/page

The local procedures *declared in a window/page* can be called from all the processes:

- of the window/page.
- of the controls of the window/page.
- of the local procedures associated with the window/page.

These procedures are stored in the window/page.

Caution:

- The local procedures of a window cannot be used by its sibling windows.
- The local procedures declared in a child window cannot be used in its parent window.

Procedures local to a report

The local procedures *declared in a report* can be called from all the processes:

- of the report.
- of the report controls.
- of the local procedures associated with the report.

These procedures are stored in the report.

Creating a local procedure

1. From the code editor:

To create a procedure local to the current element from the code editor:

- **Method 1:** On the "Code" pane, in the "Procedure" group, expand "New" and select "New local procedure" ([F4]).

In WebDev:

- To create a local server procedure, on the "Code" pane, in the "Procedures" group, expand "New" and select "New local procedure (Server)".

- To create a local browser procedure, on the "Code" pane, in the "Procedures" group, expand "New" and select the "New local procedure (Browser)" option.

- **Method 2:** Select "Create a procedure .. Create an empty local procedure" in the popup menu.

In WebDev:

- To create a local server procedure, select "Create a procedure .. Create a local procedure (Server)".

- To create a local browser procedure, select "Create a procedure .. Create a local procedure (Browser)".

Note: to transform a section of code that is already entered in a local procedure:

1. Select the corresponding code.
2. Select "Create a procedure .. Create a procedure containing the selected code" from the popup menu.
3. Specify the name of the procedure. The procedure is created, the corresponding code is replaced by the call to the procedure with the necessary parameters.

If the selected code is a server code, the created procedure will be a server procedure. If the selected code is a browser code, the created procedure will be a browser procedure.

2. From the project explorer:

To create a local procedure from the "Project explorer" pane:

1. Select the element associated with the local procedure (window, page, report, ...).
2. Expand the options of the element and select "Local procedures".
3. Display the popup menu and select "New local procedure".
4. Enter the name of the new local procedure and validate.

Note: you can display the local procedure list in the project explorer by double-clicking the "Local Procedures" folder of the element ([Ctrl] [F8] keyboard shortcut) in the code editor.

6.3 Set of procedures

6.3.1 Definition

The global procedures are grouped in sets of procedures.

A set of procedures is used to:

- share the global procedures between several developers, for the same project.
- share the global procedures between several projects.
- lock this set only when modifications are made to a global procedure.
- modify one or more global procedures even if the project is currently modified by another user.

The sets of procedures found in a WinDev project or in a WinDev Mobile project can be shared with a WebDev project.

Only the WebDev-compatible code must be used in the shared set of procedures.

6.3.2 Creating a set of procedures

To create a set of procedures:

1. In the "Project explorer" pane, select "Procedures".
2. Select "New set of procedures" from the popup menu (right mouse click).

6.3.3 Importing a set of procedures

A set of procedures can be imported into a project. The procedures found in this imported set can then be used in your project.

To import a set of procedures:

1. In the "Project explorer" pane, "Procedure" option, select the "Import a set of procedures" option from the popup menu.
2. Select the set of procedures to import.
3. Confirm that the selected set must be added to your project.

6.4 Declaring a procedure/a function

6.4.1 Syntax

The method for declaring a procedure is the same for a global procedure and for a local procedure.

Important: Do not declare two procedures with the same name (especially a global procedure and a local procedure).

Declaring a procedure with parameters

```
PROCEDURE <Procedure Name>([<Parameters>])
<Procedure Code>
```

Declaring a procedure without parameter

```
PROCEDURE <Procedure Name>()
<Procedure Code>
```

Declaring a procedure with variable parameters

```
PROCEDURE <Procedure Name>([<Parameters>],*)
<Procedure Code>
```

Note: To make your code more readable, **PROCEDURE** can be replaced by **FUNCTION**.

6.4.2 Exiting from a procedure

To force the exit from a procedure, use the **Return** keyword. In the procedure, the processes that follow the Return keyword will not be run. For exam-

ple:

```
PROCEDURE Calculate(Dividend)...
  IF Dividend = 0 THEN ...
    Error("Division by 0 ...
      "not possible")
    Return
  ELSE
    ...
  END
```

Notes:

- If the procedure returns a result, use the **Result** keyword.
- **Return** and **Result** cannot be used in the same process.

6.4.3 Returning a result

To return the result of a procedure to the calling process, use the **Result** keyword. For example:

```
PROCEDURE Find(File,Key,Value)
HReadSeek (File,Key,Value)
IF HFound=True THEN
  Result True
ELSE
  Result False
END
```

Advice: For procedures that return a result, we recommend that you use the **FUNCTION** keyword instead of **PROCEDURE** to make the code easier to read.

Important: Do not force the exit from a procedure that returns a result with the **Result** keyword. Indeed, the call to the procedure expects a status report that will not be returned if the exit from the procedure is forced.

6.5 Calling a procedure/function

The method for calling a procedure is the same for a global procedure or for a local procedure.

Reminder:

- In WLanguage, there is no distinction between the procedures and the functions. The syntaxes for declaring and using the procedures also apply to the functions.
- For a multi-syntax procedure, the call to the proper syntax is resolved at run time. See “Overload a WLanguage function”, page 101 for more details.

Syntax: calling a procedure

```
[<Returned Value> = ] <Procedure
Name> ([<Parameters>])
```

```
// Call to the Find procedure that
returns a boolean
IF Find(Customer,CustNum,Number)
THEN
    Info("Customer number found")
ELSE
    Info("Customer number" ...
        "not found")
END
```

Notes:

- In order to make your code more readable, the **PROCEDURE** keyword can be replaced by the **FUNCTION** keyword
- To run a procedure of an open window, use **ExecuteProcess**.

6.6 Parameter of a procedure/function

You have the ability to pass parameters to a procedure. This paragraph presents the following topics:

- the type of the parameters.
- passing parameters.
- the local parameters.
- the optional parameters.
- The variable parameters (procedure with a variable number of parameters).

Reminder: WLanguage does not differentiate between procedures and functions. The procedures and the functions are managed in the same way.

6.6.1 Type of the parameters

Default type of the parameters

The description of the parameter type is optional. By default, the type of the variable passed in parameter during the call to the procedure is used in the procedure.

For example:

```
Subscript is int
// Call to MyProc procedure
MyProc(Subscript)
-- Declare procedure MyProc
PROCEDURE MyProc(Counter)
// Counter is an integer
Counter += 1
```

Therefore, the same procedure can be used for several types of variables.

Forcing the type of the parameters

To force the type of the parameters, use the following syntax:

```
PROCEDURE <Procedure Name>(...
    <Parameter 1> is <Type>,...
    <Parameter 2> is <Type> ,...
    <Parameter N> is <Type> )
```

The type of the variable passed in parameter (during the call to the procedure) must be identical to the type described in the declaration of the procedure. Otherwise, an error occurs when compiling the project, the window or the report.

In the following example, the "Subscript" variable is not a real: an error will occur when compiling the project, the window or the report.

```
Subscript is int = 7
// Call to MyProc procedure
MyProc(Subscript)

-- Declaration of MyProc procedure
PROCEDURE MyProc(Subscript is real)
...
```

6.6.2 Passing parameters

During the call to a procedure, the parameters can be:

- passed by address
- passed by value.

Passing parameters by address

By default, when calling a procedure, the parameters are **passed by variable (by address)**. If the parameter is modified in the procedure, the process calling the procedure will retrieve the parameter with **its modified value**.

To pass a parameter by variable to a procedure, use the following syntax:

```
<Procedure Name>(<Name of Variable
passed as parameter>)
```

For example:

```
Subscript is int = 1
// Before the call to the procedure,
Subscript is set to 1
AddOne(Subscript)
// After the call to the procedure,
Subscript is set to 2

-- Declaration of the procedure
PROCEDURE AddOne(Counter)
Counter += 1
```

Passing parameters by value

When calling a procedure, the parameters can be **passed by value**. If the parameter is modified in the procedure, the process calling the procedure will retrieve the parameter with **its unmodified value**.

Solution 1: to pass a parameter by value to a procedure, use the following syntax:

```
<Procedure Name>((<Name of the variable
passed as parameter>))
```

For example:

```
Subscript is int = 1
// Before the call to the procedure,
Subscript is set to 1
AddOne((Subscript))
// After the call to the procedure,
Subscript is still set to 1
-- Declaration of the procedure
PROCEDURE AddOne(Counter)
Counter += 1
```

Solution 2: Using some "local" parameters in the procedure

When declaring a procedure, the variables passed in parameter can become local to this procedure. To do so, the local parameter must be preceded by the LOCAL keyword. For example:

```
PROCEDURE MyProc(LOCAL Subscript,
LOCAL Counter, Number)
```

If this parameter is modified in the procedure, the process calling the procedure will retrieve the parameter with **its unmodified value**.

Passing a control, a window or a report

This element (control, window, page or report) is handled like any "standard" object. To pass an element in parameter to a procedure, use the following syntax:

```
<Procedure Name>(<Element Name>)
```

For example:

```
// Call the ControlVisible procedure
ControlVisible(ControlAddress)

-- Declaration of the procedure
PROCEDURE VisibleControl(ControlAd-
dress)
ControlAddress.Visible = False
```

Notes:

- **Myself** is used to handle the current control.
- **MyWindow** is used to handle the current window.
- **MyPage** enables you to handle the current page
- **MyReport** is used to handle the current report.

6.6.3 Optional parameter

Some of the parameters passed to a procedure can be optional parameters. When declaring the procedure, the optional parameters must be described in last position (to the right), with a default value preceded by "=":

```
PROCEDURE <Procedure Name>(...
    <Mandatory parameters>,...
    <Optional Parameter 1> = <Value>,...
    <Optional Parameter 2> = <Value>,...)
```

For example:

```
-- Declaration of the procedure
PROCEDURE Drawing(Row, Column,...
    BckgrdColor = Black, LineCol = ...
    iLightYellow)
...
-- call the procedure
Drawing(10,15)
```

To keep the default value of an optional parameter, use the "*" character during the call to the procedure. For example:

```
-- Declaration of the procedure
PROCEDURE Drawing(Row, Column,...
    BckgrdColor = Black, ...
    LineCol = iLightYellow)
...
-- Process for calling the procedure
Drawing(10,15, *, iLightGreen)
```

6.7 Overload a WLanguage function

6.7.1 Definition

Overloading a WLanguage function enables you to use a custom function with the same name as the WLanguage function.

For example, instead of using the standard WLanguage function named `Info`, you can use a custom procedure named `Info`. The custom procedure will be run whenever `Info` is called.

6.7.2 How do I proceed?

To overload a WLanguage function:

1. Create a new procedure in your project (local or global procedure). This procedure must have the following characteristics:

- The same name as the WLanguage function to overload.
- The same number of parameters as the WLanguage function to overload. These parameters must have the same type as the parameters of

6.6.4 Procedure with a variable number of parameters

If the procedure uses a variable number of parameters, you must be able to handle the different parameters passed to the function in the code of the procedure. These operations are performed via the **MyParameters** keyword.

Example: Procedure for shifting the controls: The controls passed in parameters are shifted by 10 pixels.

```
// Procedure used to shift the controls
PROCEDURE ShiftControl(*)
FOR I = 1 TO ...
    MyParameters..Occurrence
    MyParameters[I]..X +=10
END
```

The parameters are always indexed from 1 regardless the number of mandatory parameters or the number of optional parameters.

A variable number of parameters can be used with:

- the procedures,
- the class methods,
- the declaration code of global variables of windows, pages or reports.

the WLanguage function.

2. The custom procedure will be used instead of the WLanguage function whenever the name of the function/procedure is used in the project (or in the window/page if the created procedure is a local procedure).

6.7.3 Differentiating between the WLanguage function and the custom function

If a WLanguage function was overloaded and if you want to use the initial function, the name of the function must be prefixed by **WL**. The following syntax must be used:

```
WL.<Function Name>
```

For example, to overload the **Trace** function of WLanguage, you can use the following code:

```
PROCEDURE Trace(Information1)
Message("Start a trace")
WL.Trace(Information1)
```

6.8 Prototype overload

6.8.1 Overview

The procedures and the methods of classes can have several syntaxes.

For example, a procedure can have:

- a syntax that takes a string in parameter.
- a syntax that takes an integer in parameter.

Therefore, several syntaxes can exist for the same procedure or for the same method with different parameters and code. At run time, the engine automatically defines the syntax to call according to the number and to the type of the parameters passed.

This technology is presented under several names and it includes different purposes. The following terms can be used:

- Overload,
- Prototype overload,
- Overload,
- Dynamic dispatch,
- Parametric polymorphism,
- etc.

This feature is available for:

- The global procedures.
- The local procedures.
- The class methods including the Constructors.

In the rest of this document:

- the term of multi-syntax will be used.
- the "Procedure" keyword will be used to identify a global procedure, a local procedure or a method.

6.8.2 How do I proceed?

Adding syntaxes to a procedure

To add a syntax to an existing procedure:

1. In the project explorer, select the procedure.
2. Display the popup menu of the procedure and select "Add a syntax".
3. A new syntax is automatically created in the code editor.

Notes:

- The creation of a procedure with the same name automatically proposes to add a new syntax to the existing procedure.
- If a procedure has several syntaxes, the number of syntaxes is displayed in the project explorer (beside the name of the procedure).

Delete a syntax of a procedure

To delete a syntax:

1. Display the code of the procedure in the code editor.
2. In the syntax bar, select "Delete" from the popup menu.
3. You can:
 - delete the current syntax.
 - delete all the syntaxes (in this case, the procedure is deleted).

6.8.3 Managing the overload at run time

Basic mechanism

The basic mechanism is as follows: dynamic determination of the syntax according to the number and type of parameters

The runtime engine searches for the syntax:

- that has the same number of parameters.
- that has the minimum number of conversions.

If two syntaxes are equivalent, the first one in the order of the code editor is run.

Dynamic dispatch

In the case of a multi-syntax procedure whose parameters expect some class instances, the runtime engine uses the "Dynamic Dispatch" method to define the syntax that must be called.

Let's take a look at the following example:

- a "BaseClass" class
- two classes, "DerivedClass1" and "DerivedClass2", that inherit from "BaseClass".

```
// First syntax
PROCEDURE p(LOCAL p is BaseClass)

// Second syntax
PROCEDURE p(LOCAL p is
DerivedClass1)

// Calls
pBase is Dynamic BaseClass
// Initialization
pBase = New BaseClass
p(pBase)// first syntax

// Initialization
pBase = New DerivedClass1
p(pBase)// second syntax

// Initialization
pBase = New DerivedClass2
p(pBase) // first syntax
```

Virtual methods

To manage virtual methods, several aspects can be taken into account:

- 1st aspect: a syntax of the method of the derived class redefines a syntax of the method of the base class:

```
BaseClass
PROCEDURE meth(s is string)
PROCEDURE meth(n is int)

DerivedClass
PROCEDURE meth(n is int)

// call
oBase is BaseClass
oBase.meth("A")
// calls the 1st syntax
// in the BaseClass class
oBase.meth(1)
// calls the 2nd syntax
// in the BaseClass class
oDerived is DerivedClass
oDerived.meth("A")
// calls the 1st syntax
// in the BaseClass class
oDerived.meth(1)
// calls the 1st syntax
// in the DerivedClass class
```

- 2nd aspect: an additional syntax in the method of the derived class:

```
BaseClass
PROCEDURE meth(p)
PROCEDURE meth(s is string)

DerivedClass
PROCEDURE meth(n is int)

// call
oBase is BaseClass
```

```
oBase.meth("A")
// calls the 2nd syntax
// in the BaseClass class
oBase.met (1)
// calls the 1st syntax
// in the BaseClass class

oDerived is DerivedClass
oDerived.meth("A")
// calls the 2nd syntax
// in the BaseClass class
oDerived.met (1)
// calls the 1st syntax
// in the DerivedClass class
```

- 3rd aspect: special case when the method of the base class and the method of the derived class have a single syntax with different prototypes:

```
BaseClass
PROCEDURE meth(s is string)

DerivedClass
PROCEDURE meth(n is int)
```

The compiler cannot decide whether the method of the derived class is an override of the method of the base class or a new syntax.

To trigger an override, the attribute with the `<override>` extension must be added to the method of the derived class: .

```
PROCEDURE method(...) <override>
```

To trigger an overload, the attribute with the `<overload>` extension must be added to the method of the derived class.

```
PROCEDURE method(...) <overload>
```

7. MANAGING EXCEPTIONS

7.1 Overview

When a programming error occurs in an application or in a site, the security mechanism of WLanguage is enabled. An error message is displayed on the end-user computer and the program execution is ended.

To lessen the consequences of programming errors, WinDev and WebDev propose several solutions:

- the display of a custom error message (not available in WebDev).
- the mechanism of exceptions. This enables you to customize the management of the error.

Caution: these solutions can only be used to manage the programming errors. These solutions cannot be used to manage the runtime errors (such as "unable to write into a read-only file").

7.1.1 Displaying a custom message

To display a custom message when a programming error occurs, all you have to do is enter the text of the error when creating the executable.

This enables you to display, without any programming, a message such as:

"Problem in the XXX application: Write down the text

of the error and contact our technical support at xx.xx.xx.xx.xx".

7.1.2 Exception mechanism

The exception mechanism is used to process the error cases by programming. If an exception process is triggered, no error message is displayed and the exception code is run. This code enables you to perform all the necessary operations:

- to exit from the current application/site "properly" if the error is a fatal error
- to give control back to the user if the error can be fixed.

WinDev/WebDev proposes two types of exception processes:

- **general exceptions:** the exception process is valid for an entire object (project, window, page, report, ...)
- **specific exceptions:** the exception process is valid for a specific section of code.
- **the automated exceptions:** the management of exceptions is implemented from the interface of WinDev, WebDev and WinDev Mobile.

7.2 Mechanism of general exceptions

7.2.1 Overview

A process of general exception is available for all the components of the object with which it is associated. This exception process can be used as long as the object is available.

A process of general exception can be declared in any process.

Some examples:

- if the exception process is declared in the initialization code of the project, it is valid for any error that occurs in the project.
- if the exception process is declared in a window initialization code (global declaration code of a

page or opening code of a report), it is valid for any error that occurs in the window/page, in a window/page control or in an procedure local to the window/page. This exception process is no longer valid when the window/page is closed.

- if the exception process is declared in any process, it is valid for any error that occurs in this process. This exception process will not be valid anymore once the process is completed. Therefore, an exception process declared in the click code of a button will only be valid in this process and in all the procedures called from this process.

7.2.2 Declaration syntaxes

Processing the exception on one line

```
WHEN EXCEPTION <Exception process on 1 line>
<Rest of code>
```

Processing the exception on several lines

```
WHEN EXCEPTION
    <Code for processing the exception>
END
<Rest of code>
```

7.2.3 Declaring several processes of general exceptions

Declaring the processes of general exceptions in the same process

The second exception process will replace the first exception process from the declaration of the second exception process.

Example: In the Example procedure, if an error occurs:

- in <Code1>: no management of exceptions is enabled except if an exception procedure was defined in the calling code (in the current window or in the project)
- in <Code 2>: the active exception process is <Exception Process 1>
- in <Code 3> the active exception process is <Exception Process 2>

```
PROCEDURE Example ()
<Code 1>
WHEN EXCEPTION
    <Exception Process 1>
END
<Code 2>
WHEN EXCEPTION
    <Exception Process 2>
END
<Code 3>
```

Declaring the exception processes in different processes

If exception processes are declared in embedded objects (project and window/page, or window/page and control, for instance), the exception process of the smallest object masks the main exception process during the lifetime of this object.

Example, if the following two exception processes are declared:

- exception process A declared in the initialization code of a project
- exception process B declared in the initialization

code of a window/page.

For all the errors that occur in the window/page, the exception process B will be used and it will hide the exception process A.

Nesting exception processes in different processes: managing the exception via a higher level

If exception processes are declared in embedded objects (project and window/page or window/page and control for instance), the error can be managed in the higher level exception process.

Use the **EXCEPTION** keyword in the exception process. Example:

```
// Initialization code of project
WHEN EXCEPTION
    // stop the application
    EndProgram
END

// Global procedure ControlValue
// returns the value of the control
// returns "" if it does not exist
PROCEDURE Value(sControlName)
WHEN EXCEPTION
    // for the UnknownControl error,
    // "" is returned
    IF ExceptionInfo(errCODE) = ...
        UnknownControl THEN RESULT ""
    // if it is another error,
    // call the project error manager
EXCEPTION
END
```

7.2.4 General notes

An exception process can only be used once

An exception process can only be used once. If a second exception occurs, the exception process will not be run. If an exception process of higher level exists, this exception process will be run.

To re-enable an exception process once it has been used, call **ExceptionEnable**. Caution, this function should be used carefully.

For instance: a general exception process was declared at window level and at project level.

A first programming error occurs in the window/page. This error is processed by the exception process of the window/page.

A second error occurs in the window/page. This error is processed by the exception process of the project.

Quality of WLanguage exception process code

An exception process cannot be declared inside another exception process. We recommend that you check the quality of the code in the exception processes because any error will trigger the display of the standard window/page of the security mechanism of WLanguage.

Quality of the code following an exception process

After the execution of an exception process, the WLanguage code following the process is run. We recommend that you pay attention to the quality of this code (to avoid displaying an error message).

Example: in the following code, the {"Control1"..Value = 5 line triggers an exception process. At the end of the exception process, the same code line is run again: the exception has already been processed and the WLanguage security mechanism is triggered.

To avoid this problem, at the end of the exception process, we advise you to:

- make sure that the problem is corrected and that it will not happen again.
- use RETURN and RESULT to exit from the current process
- use **ReturnToCapture** to give control back to the user.
- use **EndProgram** to close the application.

7.2.5 Special case

A general exception process cannot be declared in the composite statements such as:

- FOR
- LOOP
- WHILE

In this type of statement, use the specific exception process.

7.3 Mechanism of specific exceptions

7.3.1 Overview

A process of specific exception is used to process a risky code (which means a code that could trigger an exception). The exception will be triggered if a programming error occurs in the specified code.

This specific exception process will only be available in the process in which it was declared.

Notes

- A specific exception process can be declared in any process.
- A specific exception process has priority over a general exception process.

7.3.2 Declaration syntaxes

Processing the simple exception

```
WHEN EXCEPTION IN
    <Code that can trigger an exception>
DO
    <Code used to process the exception>
END
<Rest of code>
```

Processing the advanced exception

```
WHEN EXCEPTION IN
    <Code that can trigger an exception>
DO
    <Code used to process the exception>
ELSE
    <Code used if the exception is not triggered>
END
<Rest of code>
```

7.4 Mechanism of automated exceptions

The management of exceptions can be customized by process.

You can choose an automatic exception process or an advanced management of exceptions via to the different WLanguage functions.

Important: The automatic management of errors and exceptions only operates for the process for which it is defined.

7.4.1 Implementation

To implement the automatic management of exceptions:

1. In the code editor, display the process where the exceptions must be managed: initialization code, click code, procedure, ...
2. In the code header, click "When exception by programming".

3. The window that is displayed allows you to specify the type of automatic process to perform:

- Running the error process ("CASE EXCEPTION:" in the code)
- Running a procedure of exception process.

7.4.2 Running the error process ("CASE EXCEPTION:" in the code)

If this option is selected, a code line is added to the current code "CASE EXCEPTION:". The code lines following this statement will be run when an exception occurs in the previous code lines (or in a process called by these code lines).

This option is recommended to group all the exception processes that may occur.

7.4.3 Running a procedure of exception process

If this option is selected, the specified procedure will be automatically run to process the error when an exception occurs in the current process (or in the processes called by the current process). This

option is recommended if a procedure used to process the exceptions is already found in your application. This procedure will be automatically called when an exception occurs. No test to run.

This procedure can return one of the constants found in the table below.

Depending on the constant returned, WinDev or WebDev will run the corresponding process.

OpCancel	Restarts the exception
OpEndProcess	The function that triggered the error returns an error and the current process stops. Equivalent to ReturnToCapture
OpEndProgram	The function that triggered the error returns an error and the program stops. Equivalent to EndProgram
opRelaunch-Program	Ends the application and automatically restarts the application.

7.5 Functions for managing the exceptions

The following functions are used to manage the exceptions:

ExceptionEnable	Re-enables the current exception process if the exception was corrected.
ExceptionDisplay	Displays the standard window of exceptions with the content of the current exception.
ExceptionChangeParameter	Configures the automatic behavior that will be implemented if an exception occurs in the current process.
ExceptionThrow	Artificially triggers the security mechanism of WLanguage by throwing an exception. Then, this exception can be processed by an exception code written for this purpose.
ExceptionInfo	Retrieves information about the current exception.
ExceptionPropagate	Propagates an exception. This function must be run in an exception process and it is used to restart the mechanism of exception process.
ExceptionRestoreParameter	Restores the exception process of a process. This exception process was modified in the current process by a call to ExceptionChangeParameter .

These functions are presented in details in the online help.

8. OBJECT-ORIENTED PROGRAMMING (OOP)

8.1 Overview of OOP (Object Oriented Programming)

The object-oriented programming (called "OOP") is a programming paradigm in which the programs are organized as sets of objects. Each object represents an instance of a given class, all the classes being members of a class hierarchy unified by inheritance relationships.

Therefore, objects are used by the OOP.

- Each object is an instance of a given class.
- The classes are linked between themselves by the inheritance relationships

WLanguage and OOP

WLanguage is an object-oriented language, indeed:

- it supports the objects

- the objects have an associated class
- the classes can inherit from attributes coming from "super classes"

Important: The purpose of this chapter is not to describe the concepts of object-oriented programming but only to describe how to declare the classes, methods and objects in WLanguage. This chapter presents the OOP syntax of WLanguage and the principle for creating an object-oriented program in WLanguage.

We assume that readers are familiar with the concept of classes, objects and inheritances, ...

If you are not familiar with the OOP, we recommend that you read an OOP guide before you continue with this chapter.

8.2 OOP concepts

8.2.1 Class

A class contains the description of a data structure (members) and the processes (methods) used to handle this structure.

Therefore, a class defines a type of data and its behavior.

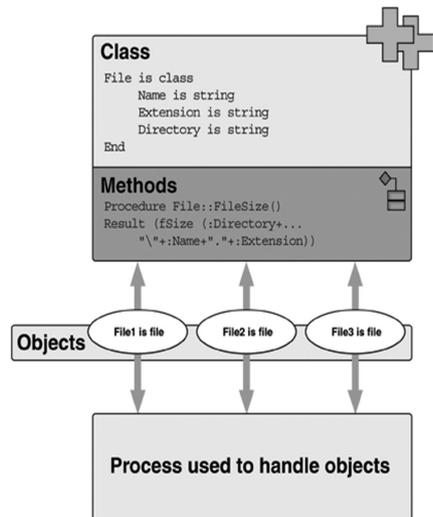
See "Class, members and methods", page 109 for more details.

8.2.2 Object

A class is used to create objects. Each created object owns the members described in its class and it can be handled via the methods of its class. An object is known as "an instance of the class".

A class can be considered as being a model that defines the members and the methods common to several objects.

A member is a parameter of the object. A method is used to act on the object, to modify its members for example.



See "Object", page 113.

8.2.3 Constructor and destructor

The notion of Constructor and Destructor is important because it involves an automatic method call when creating or destroying an object.

The Constructor method associated with a class is automatically called when declaring an object of the class. This ensures that the initialization processes of the object (assigning members for example) will not be forgotten by the developer.

The Destructor method associated with a class is automatically called when deleting the object (exit from the procedure where the object was declared). This allows you to free the resources used by the object without fear of oversight (memory zone for example). It can also be used to update a file related to the object.

See "Constructor and destructor", page 112 for more details.

8.2.4 Inheritance

The inheritance is used to include the characteristics of an existing class (base class) into a new class (derived class). This enables you to create a

new type of data from a known type, in order to add features to it or to modify its behavior. Therefore, the base class will not be modified. A class can inherit from one or more classes; it becomes a subclass of this class.

A derived class inherits the members and methods of its parent classes, in addition to its own members and methods. There is no need to duplicate the members and methods of the parent classes.

See "Class inheritance" on page 114 for more details.

8.2.5 Data encapsulation

The data encapsulation is probably the most important notion in OOP. This technique ensures that the object data will not be wrongly modified by functions (methods) external to the object. It is possible to prevent a user from accessing some or all object members. The members whose access is not allowed are called private members.

They can only be accessed from the methods designed for this purpose in the class.

8.3 Class, members and methods

A class includes:

- data, called **members**
- constants
- procedures, called **methods**. A method is a procedure specifically written to handle the objects found in the class.

To use a class, you must declare one or more objects. All the objects of a given class have the same attributes and behavior, but their members contain different data.

By default, the members of a class are public, they can be accessed by a method of the class as well as by a process of the project or by one of its elements (window, page, control, etc.)

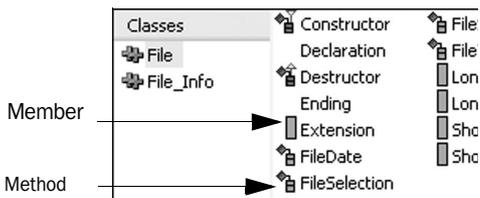
Example of class:

The "File" class includes the following members: Name, Extension, Directory

The "File" class has the following methods:

- FileSelection() to select a file and assign the members of the File object with the selected file

- FileSize() to find out the size of the file of the File object



8.3.1 Declaration of the class

```
<ClassName> is class
    <Declaration of members>
END
```

```
SystemClass is class
    Name is string
END
```

8.3.2 Declaration of members

[<Access>] [CONSTANT] [GLOBAL] <Member Name>
is <Member Type>

```
SystemClass is class
    Name is Private string
    Size is int
END
```

The important parameters of this syntax are as follows:

<Access>: Optional

Used to restrict the access to this class member. 3 levels are available:

- **Private:** access allowed from the code of the class
- **Protected:** access allowed from the code of the class or from the code of a derived class
- **Public (default):** access allowed from any code of the class or project.

CONSTANT: Optional

Modifies the access constraints by allowing an access in read-only.

- **Protected CONSTANT:** The value of the member can be read and modified from the code of the class; it can only be read from the code of a derived class; it cannot be accessed from any other code of the application.
- **Public CONSTANT:** The value of the member can be read and modified from the code of the class or derived class; it can only be read from another code of the application.

GLOBAL: Optional

Defines a global member. This member will exist outside the objects. It can be used without instantiating an object. If several objects are declared, a global member is common to all the objects.

<Name of the member>

Name identifying the member.

<Type of Member>

Type of the member chosen among the available WLanguage types.

8.3.3 Declare the constants

CONSTANT <Constant Name> = ...
<Value of the constant>

or

```
CONSTANT
    <Constant Name> = <Constant Value>
    <Constant Name> = <Constant Value>
END
```

```
CONSTANT K=5
CONSTANT
    K=5
    J=10
END
```

The important parameters of this syntax are as follows:

<Constant Name>

Constant Name. A constant is public.

<Value of the constant>

Value associated with the constant. This value will not change during the execution of the program.

8.3.4 Declaring the methods

Function [<Access>] [Global] [VIRTUAL]
<Class Name>::<Method Name>
([[<Parameter1>, ... [<ParameterN>]])

or

```
Procedure [<Access>] [Global] [VIRTUAL]
    <Class Name>::<Method Name>
    ([[<Parameter1>, ... [<ParameterN>]])
```

```
Global Procedure ...
    SystemClass::SeeObject(obj)
// The private member Size is
// accessible from the code of
// the class
Info("Name: "+Obj:Name + ...
    "Size: "+Obj:Size)
```

```
Global Function ...
    SystemClass::Attempt(a,b)
Result a+b
```

The important parameters of this syntax are as follows:

<Access>: Optional

Used to restrict the access to this method. 3 levels are available:

- **Private:** the method can be called from a code of the class only
- **Protected:** the method can only be called from a code of the class or from a code of the subclass
- **Public (default):** the method can be called from any code of the class or the project.

Global: Optional

Defines a global method. This method will not operate on a specific object: no class object is required to call this method.

This class can also be used to handle the global members.

Virtual: Optional

Defines a virtual method. By default, an overridden method is virtual.

<Class Name>

Name identifying the class.

<Name of the method>

Name identifying the method.

<Parameter 1 Parameter N>

Optional parameters that must be passed to the method.

Scope identified by the color of the bar

A method of a class can be public, private or protected.

The start color of the event bar changes according to the scope of the method:

- bar start is red: private method.
- bar start is orange: protected method.
- bar start is normal: public method.

Deleting a method

A method can be deleted:

- from the "Project explorer" pane ("Delete" from the popup menu)
- from the code editor, via the popup menu of the method bar ("Delete").

8.3.5 Creating and declaring properties

A property is a code element with two processes: a process for retrieving a value and a process for assigning a value.

A property can be used like a variable or like a member (direct retrieval of the value, assignment via the '=' symbol, ...). At run time:

- Any operation that requires to read the property runs the process for retrieving the value. This process must return a value.
- Any operation that requires to read the property runs the process for assigning the value that must process a parameter.

Creating a class property

To create a class property:

1. In the project explorer, display the available classes (expand the "Classes" folder).
2. Select the requested class. Display the popup menu of the class and select "New property".
3. In the window that opens, enter the name of the property and validate.
4. The code editor displays the processes linked to the property:

- Process for retrieving the property. This process contains:
 - the "RESULT" keyword that is used to get the value of the property.
 - the "RETURN =" keyword to return the value of the property.
- Process for assigning the property. This process is used to give a value to the property. This value is passed in parameter. This process must return no result.

Note: A new property can also be created from the popup menu of a member. In this case, the procedures automatically handle the specified member.

See the online help for more details.

Access rights to a property

The property cannot be read if the retrieval process is empty. A compilation error will be displayed in the editor and an error will occur at run time.

The property cannot be written if the assignment process is empty. A compilation error will be displayed in the editor and an error will occur at run time.

The retrieval and assignment processes can be public, private or protected. The access rights of the property correspond to the less restrictive rights of the two processes.

The properties can be global. A property is global to the class when the retrieval and assignment processes are global. If one of the processes is global, all the processes must be global otherwise a compi-

lation error is displayed.

See the online help for more details.

8.4 Constructor and destructor

When creating a class in the code editor, the Constructor and Destructor methods are automatically created by default.

The Constructor method, if defined by the developer, is automatically called when instantiating an object. It is used to perform the initialization steps for the object or related to the object (assigning members, checks, ...)

The Destructor method, if it is defined by the developer, is automatically called when deleting the object (exit from the procedure in which the object was instantiated). It can be used to free resources for example...

The Constructor and Destructor methods cannot contain the following functions: *Event*, *Timer*, *Multi-task*, *DnDEvent*, *DDEvent*, *CallDLL32*, ...

8.4.1 Constructor of the class

Declaration of the constructor

Syntax:

```
PROCEDURE [<Access>] Constructor([<Parameters>])
```

The important parameters of this syntax are as follows:

<Access>

3 access levels are available:

- **PUBLIC** (default): the constructor is accessible outside of the class.
- **PROTECTED**: the constructor can be accessed inside the class and the derived classes.
- **PRIVATE**: the constructor is only accessible inside the class.

<Parameters>

Optional parameters of the constructor.

8.4.2 Constructor of the base classes and of the members

If a base class or a class type member has a constructor, this constructor is automatically called without parameter. You must:

- assign default values to the parameters of the constructor (or to the parameters of the base class or member)
- explicitly call the constructor by passing the parameters.

Calling the constructor method to build a class

Syntax:

```
Constructor <ClassName>(<Parameters>)
```

where

<ClassName>

Name identifying the class

<Parameters>

Parameters of the constructor.

Calling the constructor method to build a member

Syntax:

```
Constructor <Membername>(<Parameters>)
```

where

<Membername>

Name identifying the member of the class

<Parameters>

Parameters of the constructor.

8.4.3 Destructor method

Declaration of the method

```
Procedure <ClassName>::Destructor()
```

The destructor accepts no parameter.

```
SystemClass is class
    Name is string
END
```

8.5 Object

To access a class, the **object** must be declared as being part of the class to handle, this is called **instantiation of an object**.

An instance is an object that belongs to a given class.

To handle an object, you must

1. describe its class,
2. declare the object as being part of the class.

An object can be passed in parameter to a function or to a procedure.

Note: You have the ability to dynamically instantiate an object to a class. See dynamic instantiation for more details.

Object example: "SourceFile" is object of class "File". For this object, you can handle the members named "Name", "Extension"...

8.5.1 Declaring an object

```
<ObjectName> is [object] <ClassName>
([<Parameters>])
SourceFile is File object
SourceFile is File
```

The important parameters of this syntax are as follows:

<ObjectName>

Name identifying the instance of the class.

<Class name>

Name identifying the class, defined when the class is created in the code editor.

<Parameters>

Optional parameters of the constructor.

8.5.2 The members of an object

A member of an object is a data associated with the object. An object necessarily owns all the members described in the class. A member is also called **property of the object**.

The members of an object correspond to all the members defined for the source class of the object.

Reminder: a member is declared in the class.

8.5.3 The methods of an object

The methods of an object are features associated with the object. An object necessarily owns all the methods described in the class. These methods can be called in different ways according to the

location of the call.

1. Call a method belonging to an object other than the current object

```
<Object Name> : <Method Name> ([<Parameters>])
```

The member is sought among the methods of the object class. If the method is not found, it is sought among the methods of the ancestor classes of the object class.

2. Calling a method of the current object

```
: <Method Name> ([<Parameters>])
```

3. Call a method of an ancestor class that was redefined

```
<Object Name> : <Class Name> :: <Method Name>
([<Parameters>])
```

4. Call a method of general class

```
<Class Name>::<Method Name>([<Parameters>])
```

```
SFile is File
FileD is File
str1, str2 are strings
// Method for file selection
SFile:FileSelection()
FileD:FileSelection()
// Call the method for checking the
available space
If SFile:RemainingSpace(...
    DFile:Directory[[1]]) then
    ch1= SFile:Directory+"\"...
        SFile:Name+"."+ ...
        SFile Extension
    str1= DFile:Directory+"\"...
        FileD :Name+"."+ ...
        FileD :Extension
    fCopyFile(SFile,DFile)
ELSE
    Error("Insufficient space")
END
```

8.5.4 Lifespan of an object

The object is created during its declaration. The object is local by default. The object is automatically destroyed at the end of the process containing its declaration:

- An object declared as global in the global declaration code of a window/page will be destroyed at the end of the closing process of the window/page.

- An object declared as global in the initialization code of a project will be destroyed when closing the project.

8.6 Dynamic instantiation of an object

An object can be dynamically associated with a class, we talk of dynamic instantiation of the object.

The dynamic instantiation of an object enables you to create an object at a given time and to free this object when it is no longer used.

To instantiate an object, you must:

1. declare a dynamic object
2. instantiate the object

Note: the object is automatically freed when it is not used anymore. However, you can force the destruction of the object (to provoke the execution of the destructor for example).

```
File is class
  Name is String
  Extension is String
  Directory is String
End
  SourceFile is object
  Dynamic file
//...
//create the object
//SourceFile = New File
//process on the object ...
//...
//free the object
//Delete SourceFile
```

8.7 Class inheritance

The hierarchical organization in class and sub-class has allowed to create the notion of inheritance.

In other words, an object of the sub-class A (derived class) that has the same characteristics as the class B (ancestor class) as well as its own characteristics inherits from all the characteristics of class B without having to duplicate the programs in the object of the sub-class A. The number of code lines is reduced.

8.6.1 Declaring a dynamic object

`<ObjectName> is dynamic <ClassName>`

The parameters of this syntax are as follows :

- `<ObjectName>`: Name identifying the instance of the class.
- `<Class Name>`: Name identifying the class, defined when creating the class in the code editor.

8.6.2 Instantiating a dynamic object

```
<ObjectName> = ...
  new <Class Name> [<Parameters>])
```

The parameters of this syntax are as follows:

- `<ObjectName>`: Name identifying the instance of the class.
- `<Class Name>`: Name identifying the class, defined when creating the class in the code editor.
- `<Parameters>`: Optional parameters of the constructor.

8.6.3 Freeing a dynamic object

`Delete <ObjectName>`

The inheritance is the mechanism by which the class currently described uses the methods and members defined in existing classes.

- The existing class is called **Ancestor Class** or **Base Class**.
- The new class is called **Derived class**. The derived class incorporates the ancestor class and adds new methods and new members to it.

The purpose of the inheritance is to retrieve, for a class, the methods developed for another class, by adding the specific features of the new class.

The objects found in a sub-class can access all the methods and all the members of the ancestor classes; it is as if the methods and the members of the ancestor classes were part of the sub-class.

Characteristics of an inheritance:

- An inheritance can be multiple. In this case, the derived class can derive from several ancestor classes.
- An inheritance can be private or public (by default).

If the inheritance is public, you have the ability to access the methods and the members inherited from the outside of the class.

If the inheritance is private, only the methods of the derived class can access the inherited members and methods.

8.7.1 Syntax

```
<NameDerivedClass> is class
  [PRIVATE, PROTECTED, PUBLIC]
  Object <NameAncestorClass>
  <Name of Derived Class Member>
                                     <Member type>
  ...
END
```

```
File is Class
  Name is String
  Extension is String
  Directory is String
End

FileInfo is class
  a File object
  FSize is int
  FDate is String
  FTime is String
End
```

The important parameters of this syntax are as follows:

- <NameDerivedClass>: Name identifying the derived class currently declared.
- PRIVATE: Optional keyword. Indicates whether the inheritance is private or not. If this keyword is not specified, the inheritance is public.
- <NameAncestorClass>: Name of the ancestor class.
- <Name of Derived Class Member>: Name of the member of the derived class. This member can only be used in an object of the derived class.
- <Member Type>: Type of the member, chosen among the available types.

8.7.2 Redefining the methods

In a derived class, a method of the base class can be redefined by creating in the derived class a method with the same name as the one of the base class.

The redefinition of methods is used to modify the behavior of the method defined in the base class ; the derived class can eventually redefine the method according to its requirements.

The redefined method is a virtual method by default.

Note: overloading a method (using two methods with the same name in the same class) is not supported by WLanguage.

Note: the **Object** keyword is used to access the current object inside a method.



PART 3

**Windows, pages
and controls**

10

DEVELOP 10 TIMES FASTER



1. MANAGING THE WINDOWS

WD WDMobile

1.1 Overview

The windows represent the main interface of a WinDev application. Indeed, windows allow users to view and handle information. This information can be dynamic information (coming from data files or queries) or static information (entered in the window controls directly).

WinDev and WinDev Mobile propose several functions for managing the windows.

This chapter presents:

- the functions for window management.
- the MDI functions.
- the functions for menu management.

1.2 Functions for managing the windows

The following functions are used to manage the windows:

Abandon	Closes the window and possibly runs the code of the "ABORT" button
ChangeSkinTemplate	Changes the skin template of a window.
ChangeSourceWindow	Dynamically changes the window displayed in an Internal Window control
Close	Closes a WinDev window (and returns a value if necessary)
CurrentTitle	Modifies the title of the current window and the title bar
CurrentWin	Identifies the window currently in edit
DelayBeforeClosing	Limits the display time of a window (when the specified time is over, the code of one of the window buttons is run)
DisableAAF	Disables an Automatic Application Feature (AAF) for a control, for a window or for the current application.
DrawingStyle	Allows you to find out and modify the display mode of grayed buttons as well as the display mode of the translucent border for the application windows.
EnumElement	Enumerates the windows of a project
EnumSubElement	Enumerates the groups or the menus of a window.
ExecuteAAF	Runs an Automatic Application Feature (AAF) on a control or on a window.
Iconize	Minimizes a WinDev window
InitWindow	Initializes (or not) the controls to zero and runs the initialization processes of the controls and window
IWListAdd	Adds a new internal window to the list of internal windows browsed by an Internal Window control
IWListCount	Returns the number of internal windows currently found in the list of internal windows browsed by an Internal Window control
IWListDelete	Deletes an internal window found in the list of internal windows browsed by an Internal Window control
IWListDeleteAll	Deletes all the internal windows from the list of internal windows browsed by an Internal Window control
IWListDisplay	Refreshes the display of an "Internal window" control with automatic browse on a data file or query.
IWListNext	Displays the next internal window in the list of internal windows browsed by an Internal Window control
IWListPosition	Displays the internal window corresponding to the specified position or returns the position of the internal window currently displayed

IWListPrevious	Displays the previous internal window in the list of internal windows browsed by an Internal Window control
Maximize	Displays a WinDev window in its maximum size
MultitaskRedraw	Immediately redraws the window controls that must be graphically refreshed
NextTitle	Modifies the title of the next window to open
Open	Opens a modal WinDev window
OpenChild	Opens a non-modal child window
OpenMainMenu	Opens the main menu of the current window
OpenPopup	Opens a popup window
OpenPopupPosition	Opens a popup window at the specified location
OpenSister	Opens a non-modal sibling window
PreviousWin	Identifies the window that was in edit before the window that is currently in edit
RESET	Re-initializes the controls of the current window.
Restores	Displays a WinDev window to its initial size
Use	Opens a WinDev window and closes all the other windows that were opened beforehand
WinActivateDDW	Enables or disables the DDW feature
WinAdaptHeight	Adapts the window height to the content of controls.
WinAdaptSize	Adapts the size of the window to the content of the controls. The window is resized in order for the controls to be displayed in the best possible way (no empty row and no scrollbar)
WinAdaptWidth	Adapts the window width to the content of the controls.
WinAnimationClosing	Modifies the type and duration of the animation used when closing the windows.
WinAnimationNext	Modifies the type and duration of the next window animation
WinAnimationOpening	Modifies the type and duration of the animation used when opening the windows.
WinBackgroundImage	Modifies the background image of a window and specifies the display mode of this image
WinChangeAlias	Modifies the alias of a window
WinCopyForm	Copies the content of a form into the clipboard.
WinDisableEffect	Disables the visual effects of the graphic engine of WinDev.
WindowBitmap	Creates the image of the specified window in a BMP file
WindowCount	Calculates the number of windows currently opened in the current application
WinForceDDW	Forces the DDW feature (Dim Disabled Windows) on a window.
WinGiveSuitableHeight	Returns the height of a window, adapted to the content of controls.
WinGiveSuitableWidth	Returns the width of a window, adapted to the content of the controls.
WinIconBarHeight	Returns or modifies the current height of the "Icon Bar" area in the current MDI parent window
WinInactiveEffect	Enables or disables the DDW feature
WinInHeight	Returns the internal height of a window
WinInitialized	Allows you to find out whether the "end of initialization" code was run for a window.
WinInput	Identifies the window containing the WLanguage code currently run
WinInWidth	Returns the internal width of a window
WinInXPos	Returns the horizontal position of the inside area of a window in relation to the top left corner of the screen
WinInYPos	Returns the vertical position of the inside area of a window in relation to the top left corner of the screen
WinOrientation	Modifies or retrieves the display orientation of the current window.

WinOutHeight	Returns the full height of a window
WinOutWidth	Returns the full width of a window
WinOutXPos	Returns the horizontal position of a window in relation to the top left corner of the screen
WinOutYPos	Returns the vertical position of a window in relation to the top left corner of the screen
WinPasteForm	Pastes the content of a form stored in the clipboard.
WinRateDDW	Allows you to find out and modify the level of gray for the windows to which the DDW feature is applied
WinRedraw	Immediately redraws the window controls that must be graphically refreshed
WinScreenRectangle	Returns the coordinates of the screen that contains a window.
WinSize	Returns or modifies the display mode of a window, modifies the height and/or the width of a window, moves a window and modifies its height and/or width
WinStatus	Identifies or modifies the status of a window
WinUsefulSize	Dynamically changes the useful size of the internal window displayed in an Internal Window control.

See the online help for more details about these functions and for the availability of these functions in the different products (WinDev, WinDev Mobile).

1.3 MDI functions

The following functions are used to manage the MDI windows:

MDIActive	Identifies or enables an MDI child window in the foreground
MDIEnumChild	Returns the alias of the specified MDI child window
MDIMother	Identifies the name of the MDI parent window
MDIOpen	Opens an MDI child window
MDIWindowMenu	Modifies the layout of the MDI child windows

See the online help for more details.

These functions are available in WinDev only.

1.4 Functions for managing the menus

WinDev and WinDev Mobile propose several WLanguage functions specific to the menu options:

ControlPopupOwner	Identifies the control on which the popup menu was opened
EmulateMenu	Emulates the next menu that will be opened and automatically runs the menu option passed in parameter.
EnumMenu	Returns the name of the nth menu option
EnumSubElement	Enumerates the drop-down or popup menus found in a window or control.
grMenu	Enables or disables the popup menu of a chart.
MenuAdd	Adds a popup menu into a menu
MenuAddMenu	Adds a new menu into a window.
MenuAddOption	Adds a new menu option at the end of a menu.
MenuAddPopup	Transforms a menu option of a page in order for this option to open a popup.

MenuAddSeparator	Adds a new separator into a menu.
MenuAddURLOption	Adds a new menu option at the end of a menu found in a page. This menu option runs an URL passed in parameter and displays the corresponding page.
MenuClone	Clones a menu or a menu option as well as the associated code.
MenuDelete	Deletes a menu or a menu option
MenuExist	Indicates whether a menu option exists in a menu.
MenuInsertMenu	Inserts a menu before another menu in a window.
MenuInsertOption	Inserts a new option at a specific position. This menu option runs a procedure passed in parameter.
MenuInsertSeparator	Inserts a separator into a menu.
MenuInvisible	Makes a menu option invisible
MenusMarked	Allows you to find out whether a checkmark ✓ is displayed in front of the menu option
MenuLabel	Identifies or modifies the caption of a menu option
MenuMark	Positions the checkmark ✓ in front of the menu option
MenuSelectMinus	Disables (grays) a menu option
MenuSelectPlus	Enables a menu option
MenuState	Identifies the status of a menu option: enabled, disabled or hidden
MenuUnmark	Removes the checkmark ✓ in front of the menu option
OpenPopupMenu	Opens a popup menu for the current control or window

See the online help for more details.

These functions are available in WinDev only.

2. MANAGING THE PAGES

WB

2.1 Overview

The pages represent the main interface of a WebDev site. Indeed, the pages allow the Web users to view and handle the information.

This information can be:

- dynamic information coming from data files, queries, ... This information changes according to the requests made by the Web users.
- static. The information is fixed and it does not change.

2.2 Functions for managing the pages

The following functions are used to manage the pages:

CancelAWPContext	Deletes from the AWP context a variable added by DeclareAWPContext .
CellCloseDialog	Hides a cell displayed in the page via CellDisplayDialog .
CellDisplayDialog	Displays a cell in a page with a DDW effect (Dim Disabled Windows).
ChangeAction	Allows you to specify the action to perform when the HTML page displayed in the browser is no longer synchronized with the page context on the server
ChangeTarget	Changes by programming the target frame of the current page after the execution of a button click
ConfigureAWPContext	Defines the operating mode of the AWP contexts.
ContextClose	Closes a page context
ContextExist	Allows you to find out whether a page context exists on the server.
ContextOpen	Opens a new page context without returning the information to the browser
CurrentPage	Returns the name of the page containing the WLanguage code currently run.
DeclareAWPContext	Allows you to declare a list of variables whose value will be persistent between the successive displays of AWP pages.
DynamicSiteDisplay	Displays a dynamic site (created by WebDev) in the browser of the Web user from a dynamic or static WebDev page
EnumControl	Enumerates the controls in a page.
EnumSubElement	Enumerates the sub-elements of a page
FramesetDisplay	Displays a frameset in the browser of the Web user
FramesetRefresh	Refreshes a frameset displayed in the browser of the Web user from the context found on the server
FramesetUse	Displays a WebDev frameset in the browser of the Web user and closes all the current page contexts and frameset contexts
IdentifierAWPContext	Returns the identifier of the AWP context.
PageActivateDDW	Enables or disables the DDW effect when displaying a modal page.
PageAddress	Allows you to find out the Internet address of a WebDev page
PageCloseDialog	Closes the current page. This page was opened by PageDisplayDialog . A return value can be returned to the calling page.
PageDisplay	Displays a page in the browser of the Web user
PageDisplayDialog	Displays a page in modal way (establishes a dialog with the user).
PageExist	Checks whether the page is currently displayed in the browser of the Web user
PageInitialization	Resets (or not) the controls found in the current page and runs the initialization processes of the controls

PageParameter	Returns the value of a parameter passed to the current page
PagePosition	Scrolls a page up to position a control in the visible part of the page (top) in the browser
PageRateDDW	Defines and returns the rate of gray used by the DDW.
PageRefresh	Refreshes a page displayed in the browser of the Web user from the context found on the server
PageSubmit	Validates the specified page and starts the execution of a button
PageToASP	Sends the data found in a page displayed in the browser to an ASP server
PageToEmail	Sends the data found in a page displayed in the browser in an email
PageToJSP	Sends the data found in a page displayed in the browser to a JSP server
PageToPHP	Sends the data found in a page displayed in the browser to a PHP server
PageUse	Displays a WebDev page in the browser of the Web user and closes all the current page contexts
PopupAnimate	Displays a popup in a cell of the page.
PopupClose	Hides a popup displayed in the page via <code>PopupDisplay</code> .
PopupDisplay	Displays a popup in a page with a DDW effect (Dim Disabled Windows).
PreviousPage	Returns the name of the previous page
SemiDynamicPageDisplay	Displays a semi-dynamic page in the browser of the Web user from a dynamic or static WebDev page.
SiteAddress	Returns the Internet address for connecting to a dynamic WebDev site found on the same server

See the online help for more details.

See the online help to check the availability of these functions in the different types of code (server code and browser code).

2.3 Functions for managing the menus

WebDev proposes several WLanguage functions specific to the menu options:

ControlPopupOwner	Identifies the control on which the popup menu was opened
EnumMenu	Returns the name of the nth menu option
MenuAddMenu	Adds a new menu into a window.
MenuAddOption	Adds a new menu option at the end of a menu. This menu option runs a procedure passed in parameter.
MenuAddSeparator	Adds a new separator into a menu.
MenuAddURLOption	Adds a new menu option at the end of a menu found in a page. This menu option runs an URL passed in parameter and displays the corresponding page.
MenuDelete	Deletes a menu or a menu option
MenuInsertSeparator	Inserts a separator into a menu.

See the online help for more details.

3. MANAGING THE "BACK" BUTTON IN A PAGE

WB

3.1 Overview

The browser "Back" button allows Web users to go back to the pages that were already visited.

In a WebDev site, each HTML page displayed on the browser is associated with a page context, found on the server. Each action performed in a page displayed by the browser must trigger an identical action on the corresponding page context found on the server.

However, the browser "Back" button is used to perform an action on the browser only: the page displayed in the browser and its context on the server can become "out-of-synch" if the browser "Back" button is used.

3.1.1 Two methods can be used to manage the browser "Back" button

To avoid any out-of-sync problems between the pages displayed on the browser and the corresponding contexts found on the server, WebDev proposes two modes for managing the browser "Back" button:

- **Solution 1:** Preventing from going back to this page with the browser "Back" button.

If the browser "Back" button is used to display the previous page, this action will have no effect.

- **Solution 2:** Managing the synchronization (default solution)

For each action performed in a page from the browser, a synchronization test is automatically run between the HTML page and its context.

Two modes are available for managing the synchronization:

- default synchronization (mode used by default when creating a new page).
- programmed synchronization.

3.1.2 Example of desynchronization

Let's see a site example:

- A browser page contains a file table linked to the ITEM data file and a "Next" link.
- The ITEM data file contains a single item, each record is made of a letter in the alphabet.
- The page is used to display 6 table rows, the "Next" link is used to display the next 6 rows.

When opening the page, the table displays the 6 first records of the data file (from 'A' to 'F'). Let's see the sequence of actions performed by the user :

1. click the "Next" link

Result: the server is positioned on the next 6 records of ITEM and returns their content to the browser. The browser displays the next page of the table with the 6 new contents ('G' to 'L').

2. click the browser "Back" key

Result: the browser displays the page that precedes the first action. The table displayed contains the letters 'A' to 'F'. The server was not contacted, therefore it is still positioned on the records 'G' to 'L'.

3. click "Next"

Result: the server is positioned on the next 6 records of ITEM (from 'M' to 'R'). The browser synchronizes with the server and displays the same elements: the Web user has the feeling that some information is not displayed.

This behavior can have unexpected consequences when modifying a data file record (modification of a record other than the one viewed by the Web user for example).

Reminder: each action performed on the browser must trigger an action on the server: then, the server sends a response to the browser. The click on the browser "Back" button being an action independent of your WebDev site, the second condition may not be performed.

3.2 Preventing from using the "Back" button

If the browser "Back" button is used to display the previous page, this action will have no effect.

3.2.1 Operating mode

Disabling the "Previous page" feature of the browser triggers the insertion of the following Javascript code into the generated HTML page:

```
<SCRIPT LANGUAGE="JavaScript">
  history.forward()
</SCRIPT>
```

When the page is run in a browser, it will not be possible to go back to this page via the browser "Back" button.

Notes:

- Clicking the browser "Back" button can make the page blink.
- This mechanism can fail if the [STOP] button of the browser is clicked before the forward() statement is run by the browser.

3.2.2 Implementation

To disable the browser "Back" button for a specific page:

1. Display the description window of the page. To do so, on the "Page" pane, click  in the "Edit" group.
2. In the "Details" tab, check "Forbid from using the browser "Back" button to go back to this page".
3. Validate.

To disable the browser "Back" button for a frameset:

1. Display the "Details" tab found in the description window of the frameset ("Description" from the popup menu of the frame).
2. Check "Forbid from using the browser "Back" button to go back to this frameset".
3. Validate.

To disable the browser "Back" button for all the project pages:

1. Display the project description (on the "Project" pane, in the "Project" group, click "Description").
2. Click the "Options" tab.
3. Check "Forbid from using the browser "Back" button to go back to this page".
4. Validate. This option will be automatically taken into account for all the new pages of the site.

3.3 Managing the synchronization

3.3.1 Overview

For each action performed in a page, the mechanism for page synchronization automatically checks the synchronization. This check consists in verifying whether the page displayed in the browser corresponds to the page context found on the server.

Two modes are available for managing the synchronization:

1. Default management of synchronization.
2. Management of synchronization by programming, in the synchronization code of the page.

3.3.2 Default synchronization

The default synchronization mechanism is triggered only if the "Use the mechanism for page synchronization" option is selected for the page.

If a desynchronization occurs, a warning message informs the Web user that the requested action has not been performed. The page corresponding to the context found on the server is redisplayed. The site

can continue to operate.

To implement the management of synchronization in a page:

1. Display the description window of the page. To do so, on the "Page" pane, click  in the "Edit" group.
2. In the "Details" tab, check "Use the mechanism for page synchronization".
3. Validate. This page will be automatically included in the history of browser pages: you will be able to go back to this page via the browser "Back" key.

To implement the management of synchronization in all the project pages:

1. Display the project description. To do so, on the "Page" pane, click  in the "Edit" group.
2. In the "Options" tab, check "Use the mechanism for page synchronization".
3. Validate. The project pages will be included automatically in the history of browser pages: the browser "Back" key will allow you to go back to these pages.

Notes:

- This management mode requires no specific WLanguage code.
- The synchronization mechanism can be disabled for the page controls that do not require any synchronization management ("Close" button for

example): to do so, check "Disable the mechanism for page synchronization for this control" in the "Advanced" tab of the control description.

- The warning message can be customized (see the next paragraph).

3.4 Synchronization by programming

To manage the synchronization by programming:

1. Display the description window of the page. To do so, on the "Page" pane, click  in the "Edit" group.
2. In the "Details" tab, check "Use the mechanism for page synchronization".
3. Configure (if necessary) the page controls for which the synchronization must not be managed. For each control that triggers an action on the server, you can specify whether the page synchronization must be managed (default option) or ignored during this action.
To ignore the management of synchronization, check "Disable the mechanism for page synchronization for this control" in the "Advanced" tab of the control description.
4. Enter the code required for custom management of the synchronization in the synchronization code of the page. Use **ChangeAction** in the synchronization code of the page. This function is used to define the action that will be performed in case of page desynchronization.

Notes:

- **ChangeAction** is initialized with "No action" if a WLanguage function used to display or refresh a page is used in the synchronization code of the page.
- To customize the desynchronization message, all you have to do is enter in the synchronization code of the page:
 1. the custom message.
 2. Redisplaying the current page on the server (with **PageRefresh** for example).
- To perform a synchronization from the information found on the computer of the Web user, we recommend that you:
 1. Use a hidden control containing the identifier of the record displayed and selected.
 2. In the synchronization code, find the current record on the browser. This search is performed from its identifier found in the hidden control.
 3. Refresh the page.

4. COMMUNICATING WITH THE USER

4.1 Overview

WinDev, WebDev and WinDev Mobile enable you to communicate with the user of with the Web user via dialog boxes.

You have the ability to use standard dialog boxes (made of images, text and one or more buttons: OK, Cancel, Yes or No).

WinDev also enables you to use advanced dialog boxes that:

- display a specific text in the buttons, for better understanding.
- the user to perform an input.

4.2 The standard dialog boxes

The dialog boxes are standard windows that allow you to communicate with the user. They can be used to notify of an error, to ask for confirmation, etc.

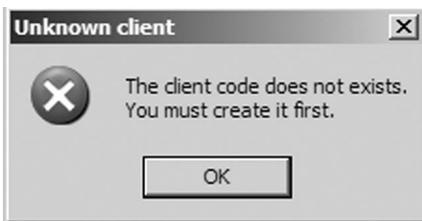
The dialog boxes always contain the same elements:

- An icon used to quickly identify the type of message displayed: Information, Question or Error.
- A title displayed in the title bar.
- One or more buttons allowing the user to choose an answer. The number and the type of these buttons depend on the type of dialog box used.
- A text on one or more lines corresponding to the message intended for the user.

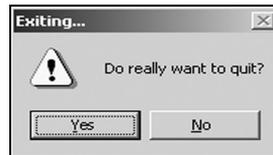
Several types of dialog boxes are available:

- The information or error boxes (WLanguage function named **Warning**, **Info** or **Error**).

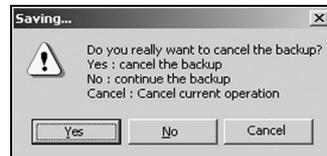
For example:



- The question boxes (WLanguage function named **YesNo**).



- The confirmation boxes (WLanguage function named **Confirm**).



By default, the dialog boxes are closed after the action performed by the user or by the Web user on one of the buttons. If no button is pressed, the application of the site is locked.

To avoid any risk of lock, use **DelayBeforeClosing** and specify the amount of time after which the dialog box is automatically closed.

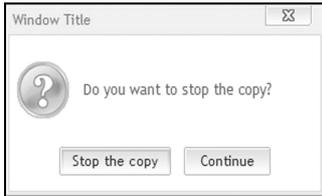
4.3 The advanced dialog boxes

WD

4.3.1 Overview

The advanced dialog boxes are windows allowing you to communicate with the user. These windows are used to manage:

- the directive questioning: the user answers a question via buttons containing the text of the action to perform.



- the immediate input: the user enters the requested value in the dialog box.



4.3.2 Directive questioning

The security of the application data depends on the answers given by the end users to the questions asked in messages. In most cases, these questions are more or less clear. For example, "Do you want to cancel the current deletion?" with "Yes" or "No" buttons.

Whenever a function used to dialog with the user is called, WinDev proposes to:

- select an existing message

- create a new message via a simple window.

The corresponding code is automatically generated when the different message characteristics are entered.

Note: The generated code uses *Dialog*.

4.3.3 Immediate input

When developing an application, you may not want to spend a lot of time creating an edit window with two buttons ("OK" and "Cancel").

The function named *Input* simplifies this task. This function enables you to create a window used to enter data (edit control or check box) and to validate or cancel the input.

4.3.4 Message database

WinDev proposes a list of messages. These messages cannot be deleted.

All the new messages are automatically added to the message database. By default, the message database is found in the "Personal\Messages" directory of WinDev.

To modify this directory:

1. Display the product options: on the "Home" pane, in the "Environment" group, expand "Options" and select "General options of WinDev".
2. In the "Directories" tab, modify the message directory.

To delete a message from the database of messages:

1. Display the code editor.
2. On the "Code" pane, in the "Languages" group, expand the "Translate strings" options and select "Direct interrogation".
3. In the window that is displayed, select the message to delete and click the "-" icon.

4.4 Customizing the dialog boxes

WD

Your applications use a lot of system dialog boxes? Why not customize these windows and give them the "style" of your application?

WinDev enables you to easily include the "WinDev-MessageBox" window in your applications. This window is a system information window (dialog box) fully customizable and that can be used in all your applications.

No code is required. When this window is included in your project, *Info*, *YesNo*, *Confirm* and *Error* automatically display the "WinDevMessageBox" window.

Reminder: By default, the dialog boxes displayed are standard dialog boxes that have the following characteristics:

- gray background.
- icon indicating the type of information displayed.
- standard WinDev buttons.

Note: The use of this window enables you to get multilingual dialog boxes (images, caption of buttons, ...). By default, this window supports French, English, German, Spanish, Italian, Dutch and Portuguese. See "Multilingual dialog boxes" for more details.

4.4.1 Customizing the system information windows

To customize the system information windows:

1. In WinDev, display the project description (on the "Project", in the "Project" group, click "Description").
2. In the "Style" tab, select the "Apply the theme of skin template to the system windows" option and validate. The "WinDevMessageBox" and "WinDevDialogBox" windows are automatically included in your application. The skin template of your project is applied to these windows.

The "WinDevMessageBox" window replaces the standard system information window (displayed by the functions named **Warning**, **Info**, **YesNo**, **OKCancel**, **Confirm** and **Error**).

The "WinDevDialogBox" window replaces the window displayed by the function named **Dialog**.

Tips

- Do not modify the code of the "WinDevMessageBox" and "WinDevDialogBox" windows, nor the code of their controls.

- Don't delete any control.
- If controls are added to the "WinDevMessageBox" and "WinDevDialogBox" windows, no code must be associated with these controls.
- Limit the style modifications applied to these windows and their controls, and test all these modifications.

4.4.2 Stop customizing the system information windows

To cancel the customization of the system information windows:

1. In WinDev, display the project description (on the "Project", in the "Project" group, click "Description").
2. In the "Style" tab, clear the "Apply the theme of skin template to the system windows" option and validate.
3. Delete the "WinDevMessageBox" and "WinDevDialogBox" windows from your project:
 - from the graph of the project
 - from the "Project explorer" pane.
4. If necessary, delete the file corresponding to the "WinDevMessageBox" and "WinDevDialogBox" windows from the project directory.

4.4.3 Multilingual dialog boxes

To display multilingual dialog boxes:

1. Customize the system information windows (see the previous paragraph).
2. Enter the multilingual captions of the various buttons ("Description" option in the popup menu).
3. Select the multilingual images of the various image controls ("Description" in the popup menu).
4. Use multilingual character strings ([Ctrl]+[T]) shortcut in the syntax of the functions named **Info**, **YesNo**, **Confirm**, **Error**, ...

4.5 Automatically close the dialog boxes

WD WDMobile

WinDev enables you to automatically close these dialog boxes to avoid locking the application. By default, the dialog boxes are closed only when the user clicks one of their buttons. If no button is pressed, the application is locked.

To avoid any risk of lock, use **DelayBeforeClosing** and specify the amount of time after which the dialog box is automatically closed.

4.6 Advanced communication with the user

WB

4.6.1 Overview

From now on, with the new Web technologies, the sites can look more like Windows applications and display dialog boxes.

In order for the Web user to easily identify the active page, the system for graying the pages is automatically used. The inactive page is grayed, so the active pages can be easily viewed by the Web user.

4.6.2 The available WLanguage functions

WebDev proposes several functions for communicating with the user:

CellDisplayDialog	Displays a cell in a page with a DDW effect (Dim Disabled Windows). Used to easily simulate a dialog box in browser code.
CellCloseDialog	Hides a cell displayed in the page via CellDisplayDialog .
OKCancel	Displays a message in a standard dialog box that proposes "OK" and "Cancel" and returns the user's choice.
YesNo	Displays a message in a standard dialog box that proposes "Yes" and "No" and returns the user's choice.
PageDisplayDialog	Displays a page in modal mode. Used to establish a dialog with the user. The page is displayed in the foreground and the calling page is displayed in the background, grayed by the DDW mechanism.
PageCloseDialog	Closes the current page. This page was opened by PageDisplayDialog . A return value can be returned to the calling page.

All these functions are used to communicate with the user and to take into account the DDW parameters defined by the following functions:

PageActivateDDW	Enables or disables the DDW effect.
PageRateDDW	Defines and returns the rate of gray used by the DDW.

4.6.3 Managing a dialog via cells

Let's see an example that is used to manage a dialog via a cell found in a page. This solution is recommended if a single page of the site must propose a dialog with the Web user.

To manage a dialog with a cell, you must:

1. Create a cell in the page that must display the dialog. This cell will be used to communicate with the user. This cell must contain:
 - a static control used to display the text of the dialog.
 - a button used to validate the dialog box.
2. The cell (as well as its controls) is invisible.
3. You have the ability to add controls into the cell if necessary (image, ...).
4. The code of the button for closing the dialog found in the cell contains the following code:

```
CellCloseDialog("MyCell")
```

CellCloseDialog is used to hide the cell that was previously displayed.

5. To communicate with the Web user from the page (from a button of the page for instance), use **CellDisplayDialog** and specify the name of the cell that must be used for the dialog.

4.6.4 Managing a dialog with pages

Let's see an example that is used to manage a dialog via a site page. This solution is recommended if several pages of the site must propose a dialog with the Web user. The dialog page can be opened from any page of the site.

To manage a dialog with a page, you must:

1. Create a page (named PAGE_Info for example). This page will be used to communicate with the user. This page must contain:
 - a cell that defines the dialog box.
 - a static control used to display the text. This static control is contained in the cell.
 - a button used to validate the dialog box. This button is contained in the cell.
2. You have the ability to add controls into the cell if necessary (images, ...). We recommend that you center the page in the browser.

3. The code of the button for closing the dialog found in the page contains the following code:

```
PageCloseDialog ()
```

PageCloseDialog is used to return a value to the page that opened the dialog.

4. To call the dialog page from another page, use *PageDisplayDialog* and specify the name of the page that must be opened as well as its parameters if necessary. These parameters can be used to dynamically fill the text displayed in the static for example.

4.6.5 Managing a dialog via YesNo and OKCancel

YesNo and *OKCancel* can also be used to manage a

simple dialog with the Web user. These functions are used to ask a question to the Web user who will answer by either "Yes" or "No", or "OK" or "Cancel".

To manage a dialog with YesNo and OKCancel, you must:

1. Include specific internal pages in your project. These internal pages contain the dialog box:

- WebDevOKCancel for **OKCancel**.
- WebDevYesNo for **YesNo**

These page are included via the page creation wizard. These pages must be saved in your project with the name proposed by default.

2. Use *YesNo* and *OKCancel* in the code of the application.

4.7 Functions for managing the dialog boxes

The following functions are used to manage the messages and the dialog boxes:

Confirm	Displays a message in a window that proposes the "Yes", "No" or "Cancel" answers and returns the user's choice
DelayBeforeClosing	Limits the display duration of an information, question or confirmation dialog box
Dialog	Displays a message box and returns the value of the button that was clicked
Edit	Displays a message allowing the user to enter an information
Error	Displays a custom error message in a system error window
ErrorBuild	Displays a custom error message in a system error window.
ErrorWithTimeout	Displays a custom error message in a system error window for a set amount of time
Info	Displays a custom message in a system information window
InfoBuild	Displays a custom message in a system information window
InfoWithTimeout	Displays an information message inside a system info box for a specified amount of time
Message	Displays (or erases) a message in the status bar of the current window
MessageAddCell	Adds a new cell into the status bar of the current window
MessageDeleteCell	Deletes a cell from the status bar of the current window
OKCancel	Displays a message in a standard dialog box that proposes "OK" and "Cancel" and returns the user's choice
Progress Bar	Displays a progress bar in the status bar of the current window
ShowToolTip	Enables (or not) the display of tooltips
ToastDeleteAll	Deletes all the toasts displayed by ToastDisplay.
ToastDisplay	Displays a "Toast" message.
ToastDisplayPopup	Displays a popup page during a given duration in order to display a "Toast" message.
Warning	Displays a custom message in a system warning window
YesNo	Displays a message in a standard dialog box that proposes "Yes" and "No" and returns the user's choice

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

5. MANAGING DRAG AND DROP

5.1 What is "Drag and Drop"?

The "Drag and Drop" is used to transfer data between several controls via the mouse. This data can be moved between several controls found in the same application or in different applications.

You must distinguish between:

- The source, object containing the data to copy or to cut.
- The target, object into which the data must be copied.

The "Drag and Drop" consists in:

1. Selecting the data in the source with the mouse.
2. Pressing the left mouse button and keeping it down while moving the data to the destination
3. Releasing the mouse button to validate the transfer of data

To copy the data, keep the CTRL key down during the operation.

WinDev supports several types of Drag and Drop operations in your applications:

- **Automatic Drag and Drop** for edit controls, list boxes, listviews, treeviews and treeview tables.
- **Programmed Drag and Drop** for a large number of controls.
- **Programmed Drag and Drop between Windows explorer** and a WinDev application.

WebDev manages the Drag and Drop via HTML 5. You have the ability to implement:

- **Automatic Drag and Drop** for the edit controls, the list boxes, ...
- **Programmed Drag and Drop** for a large number of controls.

5.2 Automatic drag and drop

For some types of controls, WinDev and WebDev propose an automatic management of Drag and Drop.

No specific programming is required. To do so, specify in the control description that "Drag and Drop" must be managed.

You must not:

- call the functions for managing the Drag and Drop by programming,
- enable the codes for managing the Drag and Drop in the processes of the controls

Otherwise, the automatic management will be disabled: the Drag and Drop will need to be managed by programming.

5.2.1 WinDev: The controls affected by the automatic drag and drop

In WinDev, the automatic Drag and Drop is available for the following controls:

- the edit controls,
- the list boxes,
- the listviews,
- the tables,

- the treeviews.

Special cases: Drag and Drop between two tables:

To perform an automatic "Drag and Drop" between two tables:

- The number of columns must be identical. The content of column 1 of the source table will be copied into column 1 of the target table, idem for column 2, ...
- The columns can have different names.
- The type of the columns can be different.
CAUTION: In this case, the data can be altered or lost during the transfer (automatic conversions). We recommend that you use the **programmed Drag and Drop**.

5.2.2 WebDev: The controls affected by the automatic drag and drop

In WebDev, the automatic Drag and Drop is available for the following controls:

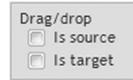
- Upload control,
- Static control,
- Button control,
- Link control,
- Image control,

- Formatted Static control,
- Edit control,
- List Box control,
- Combo Box control.

Note: At this time, only Firefox and Chrome fully support Drag and Drop via HTML 5. If the new versions of the different browsers include the management of Drag and Drop via HTML 5, this feature will be automatically supported by WebDev.

5.2.3 Configuring the default "Drag and Drop"

The default "Drag/Drop" can be configured in the "Details" tab of the description window of the control.



"Is source" is used to define the control as "Source" for "Drag and Drop". The user will be able to select the content of the control and

to move it to another control for example.

"Is target" is used to define the control as "Target" for "Drag and Drop". This control will be able to receive any object "dropped" by the user.

The two options can be selected at the same time.

5.3 Programmed "Drag and Drop"

The programmed "Drag and Drop" enables you to entirely manage the "Drag/Drop" between the different types of controls. Depending on its type, the control can be source or target of "Drag and Drop".

In WinDev, the following controls are supported:

- Source of Drag and Drop:
Only the following controls: edit control, list box, listview, treeview, table, image and bar codes.
- Target of Drag and Drop:
Any type of control except: progress bar, ActiveX, OLE, shape, toolbar, HTML, Web camera and conference control.

In WebDev, the following controls can be source and target of a programmed Drag and Drop:

- Edit control
- Static control
- Button
- List
- Image
- Formatted display control
- Combo box
- Link
- Cell
- Supercontrol
- Control template

5.3.1 Principle

To perform a programmed "Drag and Drop" between several controls of a WinDev application, you must perform the following operations:

1. In the initialization code of the source control:
 - Define that a programmed "Drag and Drop" will be performed (**..DndSource**).
 - Define the procedure run at the beginning of the "Drag" action (**DndEvent** for the **dndBeginDrag** event).
2. In the initialization code of the target control:
 - Define that a programmed "Drag and Drop" will be performed (**..DndTarget**).
 - Define the procedure run when the control is hovered (**DndEvent** for the **dndDragOver** event).
 - Define the procedure run when the "Drop" action is performed on the control (**DnDEvent** for the **dndDrop** event).
3. In each one of the WLanguage procedures called, define the action to run via the "Drag and Drop" functions of WLanguage.

5.3.2 Programming

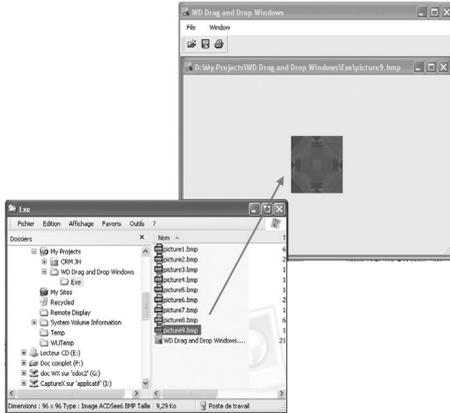
Several examples of programmed "Drag and Drop" are supplied with WinDev:

- WD Puzzle: Programmed Drag and Drop between images.
- Unit example of Drag and Drop (WinDev): programmed Drag and Drop between list boxes, tables and treeviews.
- Educational example of Drag and Drop operation in HTML5 (WebDev): programmed Drag and Drop in a WebDev site.

5.4 "Drag and Drop" from the explorer

The "Drag and Drop" from the explorer consists in selecting one or more files in the Windows explorer and in transferring them into a control in a WinDev window.

For example, the "WD Drag and Drop" example enables you to display the image files "dropped" from the explorer.



5.4.1 Principle

To enable "Drag and Drop" from the explorer to a WinDev application, the following steps must be performed:

1. Enable the ability to perform "Drag and Drop" from explorer (**ExplorerAccept**).
2. To process the action performed by the user, associate a WLanguage procedure with the drop of the file in the window (**Event**).
3. In the procedure called whenever a file is dropped in the window, retrieve the characteristics of the "dropped" files via the function named **ExplorerRetrieve**.

5.4.2 Functions specific to "Drag and Drop" from the explorer

The following functions are used to manage Drag and Drop from the Windows explorer to a window in a WinDev application:

- **ExplorerAccept**: Enables or disables the ability to

perform "Drag and Drop" from Windows explorer to a WinDev window.

- **ExplorerRetrieve**: Retrieves the number of "dropped" files and their name.

5.4.3 Programming

Enabling the management of "Drag and Drop" from the explorer to the window

To enable the ability to perform "Drag and Drop" from the explorer to a window, you must declare in the initialization code of the window:

- The implementation of "Drag and Drop".
- The event used to link a specific WLanguage procedure (GetFiles in our example) to the drop action (Windows event 563). This procedure is a procedure local to the window.

```
ExplorerAccept (True, "")
Event "GetFiles", ...
    "*. ", 563)
```

Processing the "dropped" files

In the procedure called whenever files are "dropped" in the window, you have the ability to get information and to process the relevant files.

In our example, the function named **ExplorerRetrieve** enables you to:

- find out the number of files "dropped" from the explorer,
- retrieve the name and the path of each file and display them in a new window.

```
// Number of files dropped into
// the application by Drag and Drop
nbFiles is int
nbFiles =...
    ExplorerRetrieve (_EVE.wparam)
sFileName is string
// Retrieve all the files
i is int
FOR i = 1 to NbFile
    sFileName = ExplorerRetrieve (...
        _EVE.wparam, i)
    OpenImage (sFileName)
END
```

5.5 Functions for managing Drag and Drop

The following functions are used to manage Drag and Drop:

DnDAccept	Indicates the action accepted by the target of Drag/Drop and manages the mouse cursor during Drag/Drop
DnDCacheData	Indicates the type of data and the data to copy/move during Drag/Drop
DnDCursor	Manages the mouse cursor during Drag and Drop
DnDEvent	Indicates the procedure to run during an event of Drag/Drop
DnDGetData	Retrieves a specific type of data during Drag/Drop
DnDisDataAvailable	Checks whether a specific type of data is available during Drag/Drop
ExplorerAccept	Enables or disables the ability to perform "Drag and Drop" from the Windows explorer to a WinDev window
ExplorerRetrieve	Retrieves the number and the name of "dropped" files

See the online help for more details.

These functions are available in WinDev only.

6. MANAGING THE CONTROLS

6.1 Overview

WinDev, WebDev and WinDev Mobile propose several types of controls that can be easily included in your windows or in your pages.

Several WLanguage functions can be used to handle these controls by programming. There are:

- functions for managing all the types of controls.
- functions for managing a specific type of controls. In this case, the control is associated with a

family of functions easily identifiable by a simple prefix. For example, the functions with the "Tree" prefix manage the "Treeview" controls, the "Table" prefix is used to manage the "Table" controls, ...

This chapter presents all the WLanguage functions that can be used with the various types of controls.

See the online help for more details.

6.2 General functions for handling the controls

The following functions are used to handle the controls. These are general functions that can be used with all the types of controls.

ActiveXEvent	Associates a procedure written in WLanguage with an event of an ActiveX control
AddLink	Adds a link onto a section of text in an edit control
AssistedInputAdd	Adds a row into the list of values available for the assisted input of an edit control.
AssistedInputClose	Closes the list of values proposed by the assisted input for an edit control.
AssistedInputDeleteAll	Clears the list of values proposed for the assisted input of an edit control.
AssistedInputOpen	Opens the list of values proposed by the assisted input for an edit control.
ChangeStyle	Modifies the style of a control dynamically
ConfigureSpellCheck	Allows you to configure the spelling checker of OpenOffice for all the edit controls found in the application
ControlAlias	Identifies, modifies or restores the file link of a control.
ControlClone	Creates a new control from an existing control.
ControlCurrent	Returns the name of the control currently in edit
ControlDelete	Permanently deletes a control or a table column
ControlEnabled	Ungrays a control or a group of controls found in a window.
ControlError	Customizes the error message displayed in the controls when the value cannot be displayed
ControlExist	Checks the existence of a control in a window or in a page
ControlFirst	Returns the name of the first control in edit in the specified window
ControlGrayed	Grays a control or a group of controls. During this operation, an animation can be performed on the controls.
ControlInfoXY	Returns the name of the control located at a given position
ControlInReason	Indicates the origin of the modification in the control currently in edit
ControlInvisible	Makes a control (or a group of controls) invisible in a window.
ControlNext	Identifies the next control in edit
ControlNoSpace	Specifies whether the space characters on the right of the value found in the control are deleted
ControlOver	Identifies the control hovered by the mouse cursor

ControlPopupOwner	Identifies the control on which the popup menu was opened
ControlPrevious	Returns the name of the control that was previously in edit in the current window
ControlTab	Returns the name of the Tab control that contains the specified control
ControlTypeInfo	Returns the icon corresponding to the specified type of control
ControlVisible	Makes a control (or a group of controls) visible in a window. During this operation, an animation can be performed on the controls.
CurrentColumn	Returns the subscript of the current subscripted column in a browsing or memory table
CurrentSubscript	Returns the subscript for the current control
DeleteParameter	Deletes a parameter (or set of parameters) that was saved.
EnumControl	Lists the controls in a window, page, group, tab or supercontrol
HTMLNavigate	Starts a navigation operation in an HTML control
InitParameter	Initializes the management of persistent values in the registry
LoadParameter	Reads a persistent value.
NextSubscript	Returns the number of the next row displayed in the current table.
Occurrence	Returns the number of occurrences of a control in a window/page
OpenPopupMenu	Automatically opens a popup menu for the current control or window
ParentObject	Identifies the "Parent" of a graphic object: control, column, window, page, etc.
Reset	Re-initializes all the controls found in the current window or page
ReturnToCapture	Resumes the input in the control or in the specified window/page
RTFAdd	Adds a character string at the end of the content of an RTF edit control
RTFInsert	Inserts a character string in RTF format into the content of an RTF edit control
RTFLoad	Loads a file in RTF format in an RTF edit control
RTFReplace	Replaces or inserts a character string in an RTF edit control
RTFSearch	Finds a character string in an RTF edit control
RTFSelection	Changes or lists the attributes of the text selected in an RTF edit control
RTFToText	Returns the text found in an edit control in RTF format without the RTF formatting
SaveParameter	Saves a persistent value in the registry
ScreenFirst	Defines the next control that will be in edit in the current window
ScrollBarShow	Displays (or not) a scrollbar in a control (edit control or list box)
SelectionColor	Changes the characteristics of the selection bar.
SetFocus	Gives focus to the specified control
ShowToolTip	Enables or disables the mechanism for displaying the control tooltips
TextHeight	Calculates the height (in pixels) of a text displayed in an edit control, in a static control or in a combo box
TextToHTML	Converts a text into HTML
TextWidth	Calculates the width (in pixels) of a text displayed in an edit control, in a combo box or in a static control
ToolTipColor	Modifies the colors of the tooltips for the current window
ToolTipDelay	Allows you to modify the time-out before the tooltips are displayed as well as the display duration of the tooltips

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile)

7. PROPERTIES OF WINDOWS, PAGES AND CONTROLS

The following properties are used to manage the windows, the pages and their controls:

Address	Allows you to connect the stream with a recipient and to find out the address of the last connection request
AdLoaded	Allows you to find out whether an ad is currently loaded in the Ad control.
Alias	WinDev: Used to find out and modify the alias of a window WebDev: Returns the internal HTML/JavaScript name of a control or page
Anchor	Allows you to find out the current anchor of a control and to modify the characteristics of the anchor applied to a control
AnchorRateBottom	Allows you to find out and modify the anchor rate when a control moves on the vertical axis
AnchorRateHeight	Allows you to find out and modify the anchor rate in height of a control
AnchorRateRight	Allows you to find out and modify the anchor rate when a control moves on the horizontal axis
AnchorRateWidth	Allows you to find out and modify the anchor rate when a control stretches on the vertical axis
Animation	Allows you to find out the current status of the animated image and to start or stop the animation of an image or animated caption
AnimationPeriod	Allows you to find out and modify the rotation period for the Cube and Carousel controls
AutoBrowse	Allows you to find out whether a browse performed in a browsing looper, table, list box or combo box is managed automatically or by programming
AutomaticErase	Allows you to manage the "Automatic erase" mode of an edit control.
AutomaticLink	Allows you to find out the mode for automatic detection of links in the multi-line edit controls and to enable or disable this mode.
AutomaticTooltip	Used to allow (or not) the display of automatic tooltips on the list boxes, the tables and the treeview tables.
BackgroundImage	Allows you to find out and modify the background image of a window, the image associated with a button border, the background image of a progress bar or slider
BackgroundImageState	Allows you to find out and modify the number of drawings found in the image associated with the border of a button
BarVisible	Allows you to find out whether the selection bar is visible on a column found in a table (or in a treeview table) and to make the selection bar visible or invisible on a table column.
BrowsedFile	Allows you to find out and modify the file or query used to display records in the browsing loopers, browsing tables, browsing list boxes or browsing combo boxes
BrowsedItem	Allows you to find out and modify the item used to automatically browse the tables, looper controls, list boxes or combo boxes
BrushColor	Allows you to find out and modify the background color of a control
BrushStyle	Allows you to modify the display style of the background for the cells found in the tables and/or in the treeview tables.
CalculatorButton	Allows you to find out and specify whether a Numeric or Currency edit control proposes a button to display a popup calculator.
CalendarButton	Allows you to find out and specify whether a Date edit control proposes a button used to display a calendar in popup.
Caption	Allows you to find out and modify the caption of a control, the title of a window or the title of a page

Checked	Allows you to find out whether a menu option has a checkmark (✓) or not and to display (or not) this checkmark in front of a menu option
CheckmarkVisible	Allows you to find out whether the checkmark displayed in front of a row of a TreeView control is visible (or not), and to modify the visibility of the checkmark displayed in front of a row of a TreeView control.
CheckSpell	Allows you to find out whether the spelling checker of OpenOffice is proposed (or not) in an edit control or in a table column and to enable it if necessary.
Collapsed	Allows you to define the status (collapsed or expanded) that will be used when new rows are added into the treeview table.
CollapsedImage	Allows you to find out and modify the default image for a collapsed row in a treeview table.
Color	Allows you to find out and modify the color of the text displayed in a control
ColumnWidth	Allows you to find out and modify the width of each column found in a multi-column looper
CompactOption	Allows you to find out whether the options found in a Radio Button or Check Box control are compacted and to compact the options found in a Radio Button or Check Box control.
CompactToolbar	Allows you to find out whether the window toolbars are compacted and to compact (or not) the toolbars.
Cumulated	Allows you to find out whether the value of a row or column found in a pivot table corresponds to a total (total at the end of row or at the end of column).
Cursor	Allows you to find out and modify the position of the mouse cursor in a control
CursorEnd	Allows you to find out and modify the position of the end selection of the mouse cursor
DayBreakHeight	Allows you to find out and modify the height of the breaks between days in a Scheduler control with days in rows and resources in columns.
DayHeight	Allows you to find out and modify the height of the days in a Scheduler control with days in rows and resources in columns.
DayWidth	Allows you to find out and modify the width of the days in a Scheduler control with days in columns and resources in rows.
Description	Allows you to find out and modify the description associated with a page.
Detection	Allows you to find out and modify the mechanism for automatic detection of incoming streams
DirectInputAPT	Allows you to find out and specify whether the user can modify the title of an appointment in a Scheduler control or in an Organizer control.
Display	Allows you to find out and modify the video currently displayed by the Conference control
DisplayCurrency	Allows you to find out and modify the currency used when displaying or entering a value in a "Currency + Euro" edit control or column
DisplayedItem	Allows you to find out and modify the item displayed in a table, looper, list box or combo box
DisplayedItemImage	Allows you to find out and modify the item corresponding to the image displayed in a listview
DisplayedValue	Allows you to find out the current value displayed in a window control or in a page control
DisplayEnabled	Allows you to find out whether the refresh of the display is enabled for a control or window and to enable or disable the refresh of the control or window
DisplayMask	Allows you to find out and modify the display mask.
DisplayOrphan	Allows you to find out whether a row or column found in a pivot table is displayed when it has no parent and to modify the display mode of a row or column found in a pivot table when it has no parent.
DndSource	Allows you to find out and modify the behavior of the source control during "Drag and Drop"

DndTarget	Allows you to find out and modify the behavior of the target control during "Drag and Drop"
DoubleClick	Allows you to find out and modify the name of the button that will be run when a double click is performed on an object
Driver	Allows you to find out the properties of the driver for video capture associated with a Web Camera control
ElementHeight	Allows you to find out and modify the height of the elements in an Organization Chart control.
ElementOrientation	Allows you to find out and modify the orientation of an Organization Chart control.
ElementWidth	Allows you to find out and modify the width of the elements found in an Organization Chart control.
Ellipse	Allows you to find out and modify the management mode of the ellipse in a static control, in a list box or in a Static table column
Emission	Allows you to find out and modify the type of data emitted by the stream
Empty	Allows you to find out whether a table, a loopier, a list box or a combo box is empty
EmptyIfZero	Allows you to find out whether a numeric edit control is empty when its value corresponds to zero and to modify the mode for managing the zero value in a numeric edit control
EndDate	Allows you to find out and modify the end date of a period selected in a Calendar/Organizer/Scheduler control.
EndTotalRange	Allows you to find out and modify the last displayable date or time in a Scheduler control or in a TimeLine control
EndVisibleRange	Allows you to find out and modify the last visible date or time in a Scheduler control or in a TimeLine control
ExpandedImage	Allows you to find out and modify the default image for an expanded row in a treeview table.
FileLink	Allows you to find out and modify the link between a control and a file item
Filter	Allows you to find out and modify the filter used to display records in the browsing loopers, browsing files, browsing list boxes or browsing combo boxes
FilterProcedure	Allows you to find out and modify the procedure that must be called to filter on a row or column header during the calculation of a pivot table.
FocusOnClick	Allows you to find out whether a control takes focus during a click and to modify the effect for taking focus during a click on a control
Font	Allows you to find out and modify the font used in a window control
FontAppointmentContent	Allows you to find out and modify the characteristics of the font used for the content of appointments in the Scheduler and Organizer reports.
FontAppointmentTitle	Allows you to find out and modify the characteristics of the font used for the title of appointments in the Scheduler and Organizer reports.
FontBold	Allows you to find out and modify the "Bold" attribute for the content of a control
FontCharset	Allows you to find out and modify the character set currently used by the font of a control
FontCondensed	Allows you to find out whether the characters of the text displayed in a control are condensed and to condense (or not) the characters of the text displayed in a control
FontExtended	Allows you to find out whether the characters of the text displayed in a control are extended and to extend (or not) the characters of the text displayed in a control
FontItalic	Allows you to find out and modify the "Italic" attribute for the content of a control
FontLarge	Allows you to find out whether the characters of the text displayed in a control are enlarged and to enlarge (or not) the characters of the text displayed in a control
FontName	Allows you to find out and modify the font used in a control
FontSize	Allows you to find out and modify the size of the font used in a control

FontStrikeOut	Allows you to find out and modify the "Strikeout" attribute for the content of a control
FontUnderlined	Allows you to find out and modify the "Underline" attribute for the content of a control
FullName	Allows you to find out the full name of a control, group of controls or window
GranularityDuration	Allows you to find out and modify the size of the grid for resizing the appointments in an Organizer control or in a Scheduler control.
GranularityMovement	Allows you to find out and modify the size of the grid for moving the appointments in an Organizer control or in a Scheduler control.
Grayed	Allows you to find out whether a control or a group of controls is grayed or not and to enable or disable the gray out option for a control or a group of controls.
Group	Allows you to find out whether the control belongs (or not) to a group of controls
HandwrittenInput	Allows you to find out and modify the handwritten input mode in the RTF edit controls.
Height	Allows you to find out and modify the height of a control, window, table row or list row
HelpNumber	Allows you to find out and modify the help number associated with a control
Hint	Allows you to find out and modify the help text displayed in an edit control. This text will disappear as soon as a character is typed.
HorizontalAlignment	Allows you to find out and modify the horizontal alignment of a control
HTLM Format	Allows you to find out whether an edit control accepts the input in HTML format and to modify the input format of an edit control.
HTMLAfter	Allows you to find out and modify the HTML code inserted after the control.
HTMLBefore	Allows you to find out and modify the HTML code inserted before the control.
HTMLEndPage	Allows you to find out and modify the HTML code added at the end of the page.
HTMLHeader	Allows you to find out and modify the HTML code added in the page header.
Identifier	Allows you to find out and modify the name under which the current computer appears to its correspondents (corresponds to the number display)
Image	Allows you to find out and modify the image associated with a control (button, tab, menu option), with a cursor (progress bar, slider) or with the toolbar of a MDI parent window
ImageHeight	Allows you to find out and modify the height of the image section that is displayed in the image control
ImageMode	Allows you to find out and modify the display mode of an image in an image control
ImageState	Allows you to find out and modify the number of drawings found in the image associated with a button or in the image associated with the cursor in a slider
ImageWidth	Allows you to find out and modify the width of the image section that is displayed in the image control
Incoming	Allows you to find out and modify the type of data received by the stream
InitialAnimation	Allows you to find out the initial status of the animation in an image control or in a static control
InitialContent	Allows you to find out the initial content of a memory list box or combo box
InitialHeight	Allows you to find out the initial height of a control or window and to modify the base height used to anchor controls "in height"
InitialState	Allows you to find out the initial status of a control or window
InitialValue	Allows you to find out the initial value of a window control or page control
InitialVisible	Allows you to find out whether a control or a window was visible when it was created
InitialWidth	Allows you to find out the initial width of a window or control and to modify the base width used to anchor controls "in width"
InputEnabled	Allows you to find out and modify the edit options of a control or group of controls.
InputMask	Allows you to find out and modify the input mask
InputMode	Allows you to find out and modify the input mode on a Smartphone.
InputType	Allows you to find out the type of an edit control or table column

InterpretAmper-sand	Allows you to find out and modify the interpretation of the '& character'
Keywords	Allows you to find out and modify the keywords associated with a page.
LeftIndent	Allows you to find out and modify the free space found on the left of the text in the table columns
LineHeight	Allows you to find out and modify the height of the rows in a list box, in a table or in a combo box
ListViewMode	Changes the display mode of the listview (switch from listview to list box and conversely)
LowerValue	Allows you to find out and modify the lower bound of the interval currently selected in a Range Slider control and to modify the lower bound of the interval currently selected in a Range Slider control
MagnifierMode	Allows you to find out and modify the management mode of the magnifier in the title of the columns found in a table or in a treeview table.
Map	Allows you to find out and modify the active plane of a window or to associate a control with another plane
MapMode	Allows you to find out and modify the display mode of the map in a Map control
MaskTitleDate	Allows you to find out and modify the mask used for the title of the day columns in an Organizer control or in a Scheduler control.
MaskTitleTime	Allows you to find out and modify the mask used for the title of the time columns in an Organizer control, in a Scheduler control or in a TimeLine control.
MaxHeight	Allows you to find out and modify the maximum height of a control or window
MaxLeafPerRow	Allows you to find out and modify the maximum number of elements in row for an Organization Chart control.
MaxLineHeight	Allows you to find out and modify the maximum height of the table rows (for the multi-line rows with automatic resizing)
MaxValue	Allows you to find out and modify the upper bound of a control
MaxWidth	Allows you to find out and modify the maximum width of a control or window
MDIBottom	Allows you to find out and modify the distance between the bottom border of the MDI parent window and the bottom border of the MDI area
MDILeft	Allows you to find out and modify the distance between the left border of the MDI parent window and the left border of the MDI area
MDIRight	Allows you to find out and modify the distance between the right border of the MDI parent window and the border border of the MDI area
MDITop	Allows you to find out and modify the distance between the top border of the MDI parent window and the top border of the MDI area
Memory	Allows you to find out whether the specified control is a browsing or memory list box or combo box, looper or table
MemoryCurrency	Allows you to find out and modify the currency used when handing a "Currency + Euro" edit control or column by programming
MemoryFormat	Allows you to find out and modify the format of the value returned for the Date or Time edit controls or columns
Merge	Allows you to find out whether the table cells (or the column titles) are merged and to merge the table cells (or the column titles).
Message	Allows you to find out and modify the help message
MinHeight	Allows you to find out and modify the minimum height of a control or window
Miniature	Allows you to find out and modify the image associated with an element found in a listview
MinValue	Allows you to find out and modify the lower bound of a control
MinWidth	Allows you to find out and modify the minimum width of a control or window
ModificationDurationAPT	Allows you to find out and specify whether the user can modify the duration of an appointment in a Scheduler control or in an Organizer control.

Modified	Allows you to find out whether a control was modified by the user (with the keyboard or with the mouse)
MouseCursor	Allows you to find out and modify the rollover cursor defined for a control or for a window
Moveable	Allows you to find out whether a table column can be moved by the user and to allow/forbid the user to move a table column
MoveableByBackground	Allows you to find out whether a window can be moved by its background and to make the window movable (or not) by its background
MovementAPT	Allows you to find out and indicate whether the user can move a meeting in a Scheduler control or in an Organizer control.
Multiline	Allows you to find out whether an edit control or a table column is multiline
MultimediaReader	Allows you to find out and modify the multimedia reader currently used by a video control to read the multimedia files.
Multiselection	Allows you to find out and modify the selection mode of a table, list box, looper
Name	Allows you to find out the name of a control, group of controls, window or page
NameBrowserFile	Allows you to find out the initial name of the file to upload found on the computer of the Web user.
NameServerFile	Allows you to find out the name of the uploaded file found on the server.
NbDayDisplayed	Allows you to find out and modify the number of days displayed in an Organizer control.
NbLinesPerPage	Allows you to find out and modify the number of rows in a table or in a looper displayed in a page
New	Allows you to find out whether the current row in a table or a in looper was created by TableAdd, TableAddLine, TableInsert and TableInsertLine
Note	Allows you to find out and modify the notes associated with a control, a window or a page
NoteTitle	Allows you to find out and modify the caption used in the program documentation of a control, window or page
Num1stDayOfThe Week	Allows you to find out and modify the 1st day of the week displayed in a Calendar control, in an Organizer control or in an edit control in Date with Calendar format.
NumberColumn	Allows you to find out the number of columns in a multi-column list box or in table and to modify the number of columns found in a multi-column list box
NumberPage	Allows you to find out the number of pages in a "multi-page" image file
ObserverOrientation	Allows you to find out and modify the secondary angle of a carousel control. Used to make the control "turn around".
Occurrence	Allows you to find out the number of elements
OrientationTitle	Allows you to find out whether the titles of the table columns are sloped and to slope (or not) the titles of the table columns (by 45% for example).
PageNumber	Allows you to find out and modify the page number displayed in an image control (for the "multi-page" image files)
Password	Allows you to find out whether an edit control is a "Password" control
PeriodSelection	Allows you to find out and specify whether the user can select a period in a Scheduler control or in an Organizer control.
PopupMenu	Allows you to find out and modify the popup menu associated with a control
Progress Bar	Allows you to find out and modify the Progress Bar control used during the calculation of a pivot table
ProgressBarColor	Allows you to find out the color of the bar (which means the color of the area that moves) in a Progress Bar control, a Progress bar column or a Progress Bar cell
ProgressBarSystem	Allows you to find out and modify the progress bar used as system progress bar (in the taskbar) with Windows 7 (and later).
Pushed	Allows you to find out and modify the status of an on/off button (pressed or not)
Report	Allows you to find out and modify the display status of a control, group of controls or window

Resource	Allows you to identify the visible resources in a Scheduler control and to find out the resource corresponding to the specified subscript in a Scheduler control
ResourceHeight	Allows you to find out and modify the height of the resources in a Scheduler control with the resources in rows and the days in columns.
ResourceWidth	Allows you to find out and modify the width of the resources found in a Scheduler control with resources in columns.
ReturnedValue	Allows you to find out and modify the value returned by a window
RichEdit	Allows you to find out whether a window control is in RTF format (Rich Text Format)
RightClick	Allows you to find out and modify the name of the button that will be run during a right click on an object
RightToLeft	Allows you to find out the text direction currently used in a control or in a column found in a window.
RulerModifiable	Allows you to find out whether the ruler of a TimeLine control can be moved by the user and to allow the user to move (or not) the ruler of a TimeLine control
RulerValue	Allows you to find out and modify the position of a ruler in a TimeLine control.
RulerVisible	Allows you to manage the visibility of a ruler in a TimeLine control.
ScrollValue	Allows you to find out and modify the scroll value of a scrollbar
ScrollWithFinger	Allows you to find out whether the content of a control can be moved with the finger ("scroll with finger") and to allow a control to be handled (or not) with the finger
SecurityHtml	Allows you to find out and modify the status of the security mechanism for the edit controls in HTML format
Selected	Allows you to find out whether a row found in a list box or in a table is selected and to select a row in a list box or in a table
SelectedResource	Returns the name of the resource corresponding to the selection made by the user in a Scheduler control
SelectedText	Allows you to find out and modify the text selected in an edit control, in an editable combo box or in an editable table column.
Size	Allows you to find out and modify the maximum number of characters found in a control of a window or page
Sortable	Allows you to find out whether a column can be chosen by the user as sort criterion for the table and to allow/forbid the user from choosing a column as sort criterion for the table
Sorted	Allows you to find out whether a memory list box or combo box is sorted and to make a memory list box or combo box sortable
StartDate	Allows you to find out and modify the start date of a period selected in a Calendar/Organizer/Scheduler control.
StartTotalRange	Allows you to find out and modify the first displayable date or time in a Scheduler control or in a TimeLine control
StartVisibleRange	Allows you to find out and modify the first visible date or time in a Scheduler control or in a TimeLine control
StatusBar	Allows you to find out whether a window status bar is visible and to make the status bar of a window visible/invisible
StoredItem	Allows you to find out or modify the item stored in a table, looper, list box or combo box
StoredValue	Allows you to find out the value that will be stored when selecting an element in the list box
StoreFilterAAF	Allows you to find out and specify whether the filters set by the user on a Table or Treeview table control are stored between 2 startups of the application.
StoreSortAAF	Allows you to find out and specify whether the sort performed by the user on a Table control or a Treeview Table control is stored between 2 startups of the application.

SubCaption	Allows you to find out and modify the secondary caption of an option found in a radio button.
TABOrder	Allows you to find out the position of a control in the navigation order of a window
TestMode	Allows you to find out whether the ad displayed in the Ad control is a test ad.
TextDirection	Allows you to find out and modify the writing direction used in a window or in a control (useful for managing the languages such as Arabic or Hebrew).
Timeout	Allows you to find out and modify the time-out after which the connection is considered as "having failed"
Title	Allows you to find out and modify the title of a table column, the title of a window or the title of a page
TitleImage	Allows you to find out and modify the image displayed beside the title of a column found in a table or in a treeview table.
ToolTip	Allows you to find out and modify the text displayed in the tooltip associated with a control
ToolTipTitle	Allows you to find out and modify the text displayed in the tooltip associated with the title of a table column
TotalsEnabled	Allows you to find out the operating mode of automatic calculations performed in a table and to forbid (or force) the automatic calculations in a table
TriState	Allows you to find out and modify the management mode for an option of a check box
Type	Allows you to find out the type of an object
TypeFiles	Allows you to find out and modify the filter on the files proposed in the file picker of the Upload control.
Unicode	Allows you to find out whether a control has a Unicode value and to indicate whether a control has a Unicode value
UpperValue	Allows you to find out and modify the upper bound of the interval currently selected in a Range Slider control and to modify the upper bound of the interval currently selected in a Range Slider control
URL	Allows you to find out the address associated with a link, a button or a clickable image
Value	Allows you to find out and modify the value of a control, window or page
Vertical	Allows you to find out whether a control is oriented horizontally or vertically
VerticalAlignment	Allows you to find out and modify the vertical alignment of a control
VerticalOrientation	Allows you to find out and modify the orientation of a listview control
Visible	Allows you to find out whether an element (control, group of controls, window or page) is visible and to make an element (control, group of controls, window or page) visible/invisible
VisibleOutsideWindow	allows you to find out whether a button is visible even if its position (X,Y) places it outside the border of the window (entirely or partially) and modify the visibility of a button positioned "outside the window"
VisualEffect	Allows you to manage a visual effect on a control or on a group of controls found in a window.
Width	Allows you to find out and modify the width of a control, window or table column
WithInput	Allows you to find out whether a Combo Box control is editable (or not) and to modify this status.
WithZoom	Allows you to find out whether the user can perform a zoom in an HTML control found in an Android, iPhone or iPad application.
WorkingHourEnd	Allows you to find out and modify the end time of the working hours used by an Organizer control or by a Scheduler control
WorkingHourStart	Allows you find out and modify the start time of the working hours used by an Organizer control or by a Scheduler control.

X	Allows you to find out and modify the X coordinate of a control, table column or window (position on the X axis)
XAxisMax	Allows you to find out the upper bound on the X axis for a chart control.
XAxisMin	Allows you to find out the lower bound on the X axis for a chart control.
XImage	Allows you to find out and modify the initial X coordinate of the image in an image control
XInitial	Allows you to find out and modify the initial X coordinate of a control or window
Y	Allows you to find out and modify the Y coordinate of a control or window
YAxisMax	Allows you to find out the upper bound on the Y axis for a chart control.
YAxisMin	Allows you to find out the lower bound on the Y axis for a chart control.
YImage	Enables you to find out and modify the initial Y coordinate of the image in an image control
YInitial	Allows you to find out and modify the initial Y coordinate of a control anchored "at the bottom"
ZAxisMax	Allows you to find out the upper bound of altitudes (Z axis) for a Chart control (Surface chart).
ZAxisMin	Allows you to find out the lower bound of altitudes (Z axis) for a Chart control (Surface chart).
Zoom	Allows you to find out and modify the value of the zoom performed in an image control
ZOrder	Allows you to find out and modify the z-order position of a control

See the online help for more details.

See the online help to check the availability of these properties in the different products (WinDev, WebDev, WinDev Mobile).



PART 4

**Standard
functions**

DEVELOP 10 TIMES FASTER



PSoFT

1. HANDLING THE NUMERIC VALUES

1.1 Overview

Several programming functions are used to:

- managing the numeric values,
- performing trigonometric calculations,
- handling the binary values,
- using the matrices,
- performing financial and statistical calculations.

1.2 The matrices

Several WLanguage functions are used to handle the matrices.

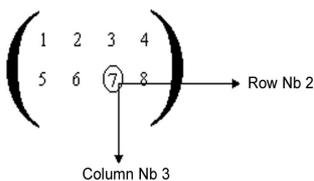
The matrices are mathematical tools used to simplify and solve many problems. The calculations on matrices are used in many fields: economy, physics, etc.

In these fields, the problems can often be formulated as a set of equations. The result of this set of equations can be calculated via the matrices.

1.2.1 Definition

A matrix is an organized set of "n p" numbers, displayed in an array of n rows and p columns.

The numbers found in the matrix are the **matrix elements**. Each element is identified by its row and by its column. A **value** is associated with each element.



In this matrix, $n = 2$ rows and $p = 4$ columns. The value of the element (row 2, column 3) is 7.

The **dimension (or size) of the matrix** corresponds to the number of rows and columns of the matrix.

The size of your matrices can be managed dynamically. Indeed, a matrix is empty when it is created: it contains 0 row and 0 column. The dimension of this matrix is 0 x 0. As soon as an element is initialized in row n and column p, the size of the matrix becomes n x p. For example, once an element is initialized in row 6 and column 12, the size of the matrix becomes 6 x 12.

1.2.2 Handling the matrices

Several WLanguage functions allow you to manage your matrices. These functions start with "MAT". These functions allow you to:

- handle the matrices (creation, copy, inversion, compression, deletion),
- initialize the elements of the matrices,
- read the value of the elements found in the matrices,
- perform various calculations on the matrices (addition, multiplication, transposition, determinant, etc.),
- get various information about the matrices (number of rows, number of columns, etc.).

Example

To perform financial calculations, your program creates several matrices. The matrix elements can be initialized. Several calculations can be performed on these matrices.

1.3 The statistics

Several WLanguage functions are used to perform statistical calculations.

The statistics are calculated from matrices. The values taken into account for calculating the statis-

tics are the values of the elements found in a matrix. The series of values correspond to a row (or a column) of the matrix.

For example, the values of row 1 for this matrix are 3, 2, 4, 5 and 6.

$$\begin{pmatrix} 3 & 2 & 4 & 5 & 6 \\ 8 & 4 & 9 & 1 & 7 \\ 9 & 7 & 12 & 15 & 17 \end{pmatrix}$$

The functions used to perform statistical calcula-

tions start with "Stat". These functions allow you to:

- calculate the variance and the covariance,
- calculate the standard deviation, etc.

1.4 Financial calculations

Several WLanguage functions are used to perform financial calculations. These calculations are performed from matrices. The values taken into account for these calculations are the values of the elements found in a matrix. The series of values correspond to a row (or a column) of the matrix.

The functions used to perform the financial calculations start with "Fin". These functions allow you to:

- calculate the amortization of a good according to different methods,
- calculate the standard deviation, etc.

1.5 Functions for managing the numeric values

1.5.1 Miscellaneous functions

The following functions are used to manage the numeric values:

Abs	Calculates the absolute value of a numeric expression (integer or real)
ArcCos	Calculates the arc cosine of a numeric value (integer or real)
ArcSin	Calculates the arc sine of a numeric value (integer or real)
ArcTan	Calculates the arc tangent of a numeric value (integer or real)
ArcTan2	Returns the arc tangent 2 of the values passed in parameter
Average	Calculates the mean of several elements
Conversion	Converts a value from a unit to another one
Cos	Calculates the cosine of an angle
CoTan	Calculates the cotangent of an angle
DecimalPart	Returns the decimal part of a number
DecimalToSexagesimal	Returns the sexagesimal angle (in base 60) corresponding to a decimal angle
Exp	Calculates the exponential of a numeric value
Factorial	Returns the factorial of an integer number
InitRandom	Initializes the generator of random numbers
IntegerPart	Returns the integer part of a number
IsEven	Identifies an even number
IsOdd	Identifies an odd number
Ln	Calculates the Napierian logarithm of a numeric value
Log	Calculates the logarithm of a numeric value
Max	Returns the greatest of the values passed in parameter
Min	Returns the lowest of the values passed in parameter
NumToString	Returns a character string corresponding to the number passed in parameter
Power	Raises a number to a power
Random	Returns a random number
Root	Calculates the root of a number
RoundDown	Returns the rounded value of a value rounded to the nearest lower integer

Rounded	Calculates the rounded value of a value according to the specified number of decimals
RoundUp	Returns the rounded value of a value rounded to the nearest greater integer
SexagesimalToDecimal	Returns the decimal angle corresponding to a sexagesimal angle
Sin	Calculates the sine of an angle
Sum	Calculates the sum of the array elements.
Tangent	Calculates the tangent of an angle

See the online help for more details about the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

1.5.2 Binary functions

The following functions are used to manage the binary values:

BinaryAND	Returns the result of a logical AND (bit by bit) between two values
BinaryNOT	Returns the result of a logical NOT (bit by bit) performed on a value
BinaryOR	Returns the result of a logical OR (bit by bit) between two values
BinaryXOR	Returns the result of a logical exclusive OR (bit by bit) between two values

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

1.5.3 Matrix functions

The following functions are used to manage the matrices:

MatAdd	Adds two matrices of the same dimension
MatCopy	Creates the copy of a matrix
MatCreate	Creates a matrix
MatDelete	Deletes an existing matrix
MatDeterminant	Calculates the determinant of a square matrix
MatError	Identifies the type of the last error caused by a function for matrix management
MatExist	Checks the existence of a matrix in memory
MatFill	Initializes all the elements found in a matrix of a given size
MatFloatAdd	Adds a value to each matrix element
MatFloatMultiply	Multiplies each matrix element by a value
MatInvert	Reverses a square matrix
MatMultiply	Multiplies two matrices
MatNbColumn	Returns the number of columns found in a matrix
MatNbLine	Returns the number of rows found in a matrix
MatRead	Reads the value of a matrix element
MatReadColumn	Reads the value of all the elements found in a matrix column
MatReadLine	Reads the value of all elements found in a matrix row
MatStack	Compresses the memory footprint occupied by a matrix
MatTranspose	Calculates the transposed matrix
MatWrite	Writes an element into a matrix

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

1.5.4 Financial functions

The financial functions are as follows:

FinCurrentVal	Returns the current value of an investment
FinDecreasingRedemption	Calculates the amortization value of a good over a specific period of time, via the formula for fixed-rate decreasing amortization
FinError	Returns the number of the last error caused by a financial function
FinFutureVal	Returns the future value of an investment (regular fixed payments with fixed interest rate)
FinInterestRate	Calculates the interest rate for a loan over a given period of time with fixed payments
FinLinearRedemption	Calculates the value of linear amortization of a good over a specific period of time
FinNetCurrentVal	Returns the net current value of an investment according to variable financial flows
FinPaymentNb	Returns the number of payments required to pay off a capital according to a given rate
FinPeriodInterest	Calculates for a given period the amount of interests due for a loan paid off by fixed periodic payments with a fixed interest rate
FinRedemption	Creates an amortization matrix for a fixed-rate loan over a defined period of time
FinRepayVal	Returns the amount of each regular payment of an investment with fixed interest rate and fixed payment

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

1.5.5 Statistical functions

The statistical functions are as follows:

StatAverage	Calculates the arithmetic, geometric or harmonic mean for a series
StatAverageDeviation	Calculates the average deviation of values in relation to their arithmetic mean
StatCorrelation	Calculates the coefficient of correlation between two series of values
StatCovariance	Calculates the covariance between two series of values
StatError	Returns the number for the last error caused by a statistical function
StatMax	Calculates the maximum value for a series of values
StatMin	Calculates the minimum value for a series of values
StatStandardDeviation	Calculates the standard deviation for a series of values
StatStandardDeviationP	Calculates the standard deviation for a full series of values
StatSum	Calculates the sum for a series of values
StatVariance	Calculates the variance for a series of values
StatVarianceP	Calculates the variance for a full series of values

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

2. HANDLING THE CHARACTER STRINGS

The WLanguage functions are used to perform several types of operations on the character strings.

You can:

- handle the content of the string,
- perform conversions and encryptions.

2.1 Handling the content of a string

The WLanguage functions allow you to extract part of a character string (*Left*, *Right* or *Middle* for example) as well as perform searches in a string (*Position* or *Replace* for example).

Some features proposed by the WLanguage functions can also be performed via the WLanguage operators on characters strings.

Several WLanguage functions are used to easily convert and encrypt the character strings.

You can for example:

- convert a string into ANSI or OEM format (*AnsiToOem* or *OemToAnsi*),

- convert a string into ANSI or UNICODE format (*AnsiToUnicode* and *UnicodeToAnsi*),
- encrypt or decrypt a character string (*Crypt* and *Uncrypt*),
- convert a character string into phonetics (*Phonetics*).

Note: For the String/Date or String/Time conversions, see “Handling the dates and times”, page 159.

See “Operators on character strings”, page 70 for more details.

2.2 WinDev and the Unicode format

2.2.1 What is the UNICODE format

The Unicode format is an encoding system that assigns a unique number to each character. This encoding is performed on 16 bits. This number can be read regardless of the platform, software or language used.

The Unicode format can support all the character sets of the planet.

2.2.2 WinDev and Unicode

To manage the UNICODE format, WinDev proposes:

- the UNICODE String type
- two conversion functions:

AnsiToUnicode Converts a character string in ANSI format (Windows) into a character string in UNICODE format.

UnicodeToAnsi Converts a character string in Unicode format into a character string in ANSI format (Windows).

- the use of functions for handling character strings (*Right*, *Left*, *Middle*, *Length*, *VarType*, ...)
- Adding a constant used to find out whether a string is an ANSI string or a Unicode string
- the use of operators for handling the character strings
 - the concatenation operator (+)
 - the comparison operator (=)
 - the extraction operator ([[]])
 - the comparison operator (<>)
- the use of functions for handling text files (*fWrite*, *fWriteLine*, *fRead*, *fReadLine*, *fOpen*).

2.3 Handling character strings in Pocket PC

The default format of character strings on PC differs from the default format of character strings on Pocket PC.

In most cases, on the PCs, the Windows applications handle character strings in ANSI format. While on Pocket PCs, Windows applications for Pocket PC handle character strings in UNICODE format by default.

Character string

In most cases, the character strings are handled the same way by the WLanguage functions in WinDev Mobile and in standard WinDev. WinDev

Mobile automatically supports the different formats of character strings in a way that is completely transparent for the developer and for the user.

Text files

In pocket PC, when using text files containing character string in ANSI format, WinDev automatically converts these character strings into the UNICODE format. This conversion is performed even if the opening of this file in ANSI format is explicitly requested. This conversion is completely transparent.

2.4 Functions for managing the character strings

The following functions are used to manage the character strings:

AnsiToOem	Transforms a character string in ANSI format (Windows) into a character string in OEM format (DOS)
AnsiToUnicode	Converts a character string in ANSI format (Windows) into a character string in UNICODE format
ArrayToString	Converts a one-dimensional array or a two-dimensional array into a character string.
Asc	Calculates the ASCII code of the specified character (the ASCII code used corresponds to the ANSI standard used by Windows)
BufferToHexa	Converts a buffer into a displayable hexadecimal string (for example: "4A 5B 00").
BufferToInteger	Extracts an integer found in a binary buffer at a given position
BufferToReal	Extracts a real found in a binary buffer at a given position.
Charact	Returns the character corresponding to the specified ASCII code (the ASCII code used corresponds to the ANSI standard used by Windows)
CharactType	Returns information about the type of a character.
CharactTypeOccurrence	Returns the number of characters matching the information of a given type.
CharactUnicode	Returns the Unicode character corresponding to the specified ASCII code
CommonLength	Returns the number of characters common to two character strings
Complete	Returns a specific character string of a given length.
CompleteDir	Adds (if necessary) the "\" character at the end of a character string
Compress	Compresses a string or a memory block (buffer) in binary format
Contains	Allows you to find out whether a sub-string is included in a string.
Crypt	Encrypts a character string in binary format or in ASCII format
Deserialize	Transforms a string or a buffer into structure, class, array
ExtractLine	Extracts a specific line from a character string
ExtractString	Extracts a sub-string from a character string, according to a given string separator
HexaToBuffer	Converts a string representing bytes in hexadecimal into a binary buffer.
HexaToInt	Returns the numeric value of an hexadecimal string
HTMLToRTF	Transforms a character string or a buffer in HTML format into a text in RTF format.
HTMLToText	Converts a character string or a buffer in HTML format into a text.
IntToHexa	Returns the hexadecimal string of a numeric value
Left	Extracts the left part of a character string (the first characters)

LengthToString	Converts a length expressed in bytes into a character string formatted in kilobytes, megabytes or terabytes
LineToPosition	Returns the position of the first character of a line in a block of text.
Lower	Converts a character string into lowercase characters, according to the localization settings defined in Windows (the accented characters are kept)
MatchRegularExpression	Checks whether a character string matches a specific format and retrieves the different sub-strings that constitute the format
Middle	Extracts a sub-string from a given position in a string
NoAccent	Transforms the accented characters found in a character string into non-accented characters
NoCharacter	Returns a character string without the specified characters found on the left and on the right
NoLeftCharacter	Returns a character string without the specified characters found on the left of the initial string.
NoRightCharacter	Returns a character string without the specified characters found on the right of the initial string
NoSpace	Returns a character string without the space characters found on the right or on the left
NumberInWords	Transforms a number into a character string corresponding to the number "written in words".
NumToString	Transforms a numeric value (integer, real or currency) into a character string according to the specified format
OemToAnsi	Converts a character string in OEM format (DOS) into a character string in ANSI format (Windows)
Phonetic	Detects whether two character strings are phonetically similar (based on French phonetics)
Position	Searches for the position of a character string in another character string
PositionOccurrence	Searches for the Xth position of a character string inside another character string
PositionToLine	Returns the number of the line to which belongs a character identified by its position in a block of text.
RepeatString	Concatenates N repetitions of the same character string
Replace	Replaces all the occurrences of a word found in a string by another word
Reverse	Returns for each character found in a character string the complement to 255 (the character string is reversed)
Right	Extracts the right part of a character string (which means the last characters)
Size	Returns the size of a character string (number of characters)
StringBuild	Builds a character string according to a format and to parameters
StringCompare	Compares two character strings
StringCount	Calculates the number of occurrences of a specific character string in another character string
StringDelete	Deletes all the occurrences of a sub-string inside a string.
StringEndsWith	Checks whether a character string ends with a specific string or with a string found in a list.
StringFormat	Formats a character string according to the selected options
StringInsert	Inserts a string into another character string at a given position
StringRetrieve	Retrieves a character string from an external format (C, etc.)
StringReverse	Reverses the characters of a string.
StringStartsWith	Checks whether a string starts with a specific string or with a string found in a list.
StringToArray	Fills a one- or two-dimensional array with the content of a string.
StringToDate	Transforms a date in character string format into a date in YYYYMMDD format

StringToUTF8	Converts a string in ANSI or UNICODE format into a string in UTF8 format
TextToXML	Converts a character string in ANSI format into a character string in XML format
Truncate	Truncates the right part of a string or buffer
UncompleteDir	Removes (if necessary) the "/" or "\" character found at the end of a string
Uncompress	Decompresses a character string or a memory block (buffer) that was compressed by Compress
Uncrypt	Decrypts a character string that was encrypted by Crypt
UnicodeToAnsi	Converts a character string in Unicode format into a character string in ANSI format (Windows)
Upper	Converts a character string into uppercase characters, according to the localization settings defined in Windows (the accented characters are kept)
URLDecode	Decodes an encoded URL (which means with characters in %xx format)
URLEncode	Encodes a URL with a sub-set of ASCII characters
UTF8ToAnsi	Converts a string in UTF8 format into a string in Ansi format
UTF8ToString	Converts a string in UTF8 format into a string in ANSI or UNICODE format
UTF8ToUnicode	Converts a string in UTF8 format into a Unicode string
Val	Returns the numeric value of a character string
WordOccurrence	Returns the number of words in a string.
WordToArray	Fills an array with the words of a string.
XMLToText	Converts a character string in XML format into a character string in ANSI format

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

3. HANDLING THE DATES AND TIMES

3.1 Overview

Several methods can be used to handle the dates and times in your applications:

- the Date or Time edit controls,
- the Date or Time items found in the data files,
- The Date, Time, DateTime and Duration variables,
- The properties associated with Date, Time, DateTime and Duration variables,
- The WLanguage functions for managing the dates and times.

3.2 Different methods for handling the dates and times

Depending on the type of operation performed on the dates and times, it may be more efficient to use the variables or the WLanguage functions.

The WLanguage functions are recommended:

- to format the dates and times and to perform conversion operations (convert a date into an integer, etc.),
- to get information about the dates and times (validity of the date or time, spell out the day, etc.).

The use of variables is recommended:

- for all the calculations performed on the dates and times (interval, calculation of duration, etc.),
- for extracting part of a date or time (day, month, year).

3.3 Handling dates/times found in each edit control

The values found in the edit controls can be handled by the date or time variables.

To do so, assign the content of the control to the variable.

See the Date type (page 42), the Time type (page 43), the DateTime type (page 43) and the Duration type (page 43) for more details.

3.4 Functions for managing the dates and times

The following functions are used to manage the dates and times:

Age	Returns the age according to the date of birth
CurrentYear	Returns the current year in integer format
StringToDate	Transforms a date (character string) into a date in YYYYMMDD format
StringToDuration	Transforms a duration in character string format into a duration that can be used by a Duration variable
ChronoStart	Starts a stopwatch to find out the duration of a process (in milliseconds) and resets an existing stopwatch
ChronoEnd	Stops a stopwatch and returns the time passed (in milliseconds) since the start of timing (call ChronoStart).
ChronoPause	Pauses a stopwatch in order to measure the time spent on a process.
ChronoReset	Stops and resets an existing stopwatch to zero.
ChronoResume	Restarts a stopwatch that was previously stopped
ChronoValue	Indicates the time passed since the call to ChronoStart
DateDifference	Calculates the number of days between two dates
Today	Returns or modifies the system date

DateTimeDifference	Calculates the difference between two dates and times
DateTimeLocalToUTC	Converts a date and a time expressed in local time (time zone, summer time, winter time) into a date and time expressed in universal time (UTC)
DateTimeByDefault	Define the default value of Date, Time and DateTime variables
SysDateTime	Returns or modifies the system date and time
DateTimeUTCToLocal	Converts a date and a time expressed in universal time (UTC) into a date and time expressed in local time (time zone, summer time, winter time).
DateSys	Returns or modifies the system date
DateValid	Checks the validity of a date (between January 01 0001 and December 31 9999)
DateToString	Formats the specified date into the specified format
DateToInteger	Transforms a date into an integer
DateToDay	Returns the day corresponding to a given date
DateToDayInAlpha	Spells out the day of the week corresponding to the specified date
DateToMonthInAlpha	Spells out the month corresponding to the specified date
DateToWeekNumber	Returns the week number corresponding to the specified date
LastDayOfWeek	Returns the date of the last day of the week for a given date.
LastDayOfMonth	Returns the date of the last day for the specified month
DurationToString	Formats the specified duration into the specified format
IntegerToDate	Transforms an integer into a date in "YYYYMMDD" format
IntegerToTime	Transforms an integer into a time in "HHMMSSCC" format
IntegerToDay	Returns the day of the week corresponding to the specified date
IntegerToDayInAlpha	Spells out the day of the week corresponding to the specified date
IntegerToMonthInAlpha	Spells out the month corresponding to the specified date
IntegerToWeekNumber	Returns the week number in the year corresponding to the date
TimeDifference	Calculates the difference between two times in hundredths of a second
TimeSys	Returns or modifies the system time
TimeValid	Checks the validity of a time
TimeToString	Formats the time passed as parameter according to the "HH:MM:SS:CC" format
TimeToInteger	Transforms a time into an integer
BankHoliday	Indicates whether a day is a bank holiday or not.
BankHolidayAdd	Indicates that a day (or a list of days) is a bank holiday.
BankHolidayDeleteAll	Clears the list of all bank holidays.
Now	Returns or modifies the system time
DayNumberInAlpha	Returns the name of the day of the week according to its number.
MonthNumberInAlpha	Returns the name of the month according to its number.
CurrentMonth	Returns the current month
WeekNumber	Returns the week number in the year corresponding to the date
Easter	Returns the date for Easter for a given year
FirstDayOfWeek	Returns the date of the first day of the week (which means Monday).
FirstDayOfMonth	Returns the first day of the month
WeekToDate	Returns the date of the Monday for the specified week and year

See the online help for more details. See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

4. HANDLING THE CHARTS

WD WM WebDev

4.1 Overview

WinDev, WebDev and WinDev Mobile propose a chart control (that can be used in the window editor, page editor and report editor) as well as functions for creating and handling charts.

Most of the applications and sites handle an important number of numeric data. The charts are used

to efficiently view this type of data.

Several types of charts can be created: Pie, Bar, Line and Scatter charts. The display options are used to produce different visual effects for each type of chart.

4.2 The different types of charts

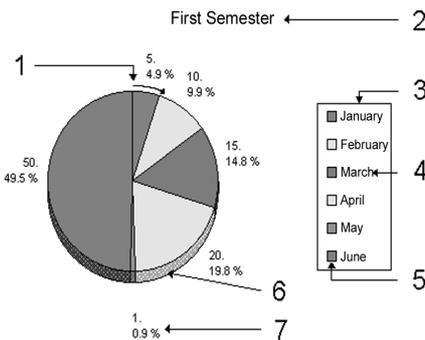
The different types of charts are as follows:

- pie chart,
- column chart,
- line chart,
- scatter chart,
- stock chart,
- 3D chart.

4.2.1 The "Pie" charts

A pie chart represents a circle divided into sections. The size of each section is proportional to the data found in the series.

A pie chart displays a single series of data. It is used to highlight a particularly important element.



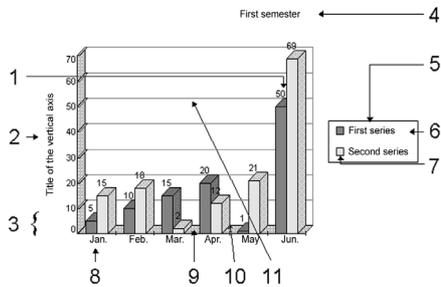
1. Start angle.
2. Title of the chart.
3. Legend.
4. Section Label.
5. Color of section.
6. Raised.
7. Caption of section.

4.2.2 The "Column" charts

The data points are interpreted as "bars" whose height is proportional to the value of the data point.

The column charts are used to represent the data trend over a given time period. They are also used to perform comparisons between elements.

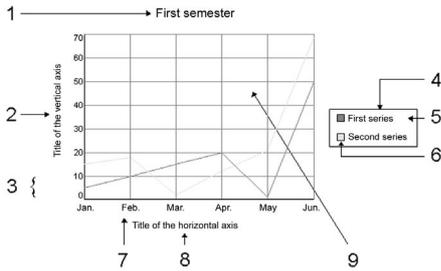
The categories are organized horizontally and the values are organized vertically in order to highlight their variation in time.



1. Caption of data.
2. Title of vertical axis.
3. Mark.
4. Title of the chart.
5. Legend.
6. Series label.
7. Color of series.
8. Caption of category.
9. Spacing.
10. Raised.
11. Gridlines.

4.2.3 The "Line" charts

In the Line charts, the data is interpreted as successive Y coordinates of points. Then, the dots found in each series are linked by a line.



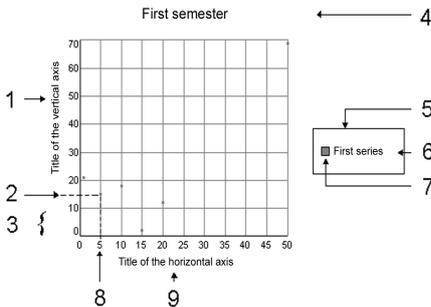
1. Title of the chart. 2. Title of vertical axis. 3. Mark. 4. Legend. 5. Series label. 6. Color of series. 7. Caption of category. 8. Title of horizontal axis. 9. Gridlines.

The line charts are mainly used to represent the data trend.

4.2.4 The "Scatter" charts

A scatter chart (XY) represents the relationship between two numeric values in two series of data. The odd series are used to identify the X coordinates of the points while the even series are used to identify the Y coordinates of the same points. The scatter charts highlight regular intervals and they are also used to group the data.

The "scatter" charts are often used to represent scientific data.

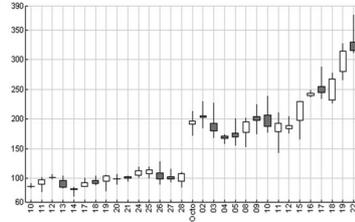


1. Title of vertical axis. 2. Y value. 3. Mark. 4. Title of the chart. 5. Legend. 6. Series label. 7. Color of series. 8. X value. 9. Title of horizontal axis.

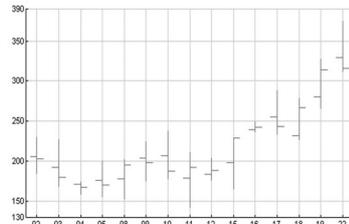
4.2.5 The "Stock" charts

The Stock charts are used to represent the variations in the values of shares. Three types of stock charts are available:

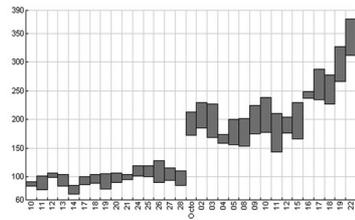
- Candlestick



- Barcharts



- Minmax



4.2.6 3D charts

All types of charts can be displayed in 3D.

4.3 How do I create charts?

4.3.1 Creating the charts in the editors

A chart control is available in the window editor, page editor and report editor. When creating this control, the wizards ask you questions to define the type of chart to create.

The description window of the control enables you to modify these options at any time.

The WLanguage functions can also be used to modify the charts displayed in the Chart controls. To do so, specify the name of the Chart control in the different functions.

4.3.2 Creating the charts by programming

Several WLanguage functions can be used to manage the charts. These functions enable you to create the charts, to modify them, to change the display options, etc.

The following operations are used to create the charts:

1. Create the chart with the function named **grCreate**.

2. Initialize the chart data with the following functions:

- **grAddData** for line and pie charts, ...
- **grScatterAddDataXY** for the scatter charts,
- **grStockAddData** for the stock charts.

3. Choose the display destination of the chart with:

- **grDestinationControl** to display the chart in an image control,
- **grDestinationWnd** to display the chart in an independent WinDev window,
- **grDestinationBmp**, **grDestinationEmf** or **grDestinationWMF** to save the chart directly in files in BMP, EMF or WMF format respectively.

4. Draw the chart with **grDraw**.

Several functions are used to configure and modify the appearance of the chart. You have the ability to display a legend, the title of axes, etc.

4.3.3 Charts and threads

A single chart can be handled in a thread.

Consequences:

- Two threads cannot handle the same chart.
- A chart created in a thread is automatically destroyed at the end of this thread. The chart exists

in the thread that contains the call to **grCreate**.

4.3.4 Default values of a chart

The default values of a chart are as follows:

- WinDev popup menu enabled (can be configured by **grMenu** and **grMenuOption**).
- Automatic marks on the axes (can be configured by **grGraduate**).
- The colors are selected according to a preset order among a set of 13 colors (can be configured by **grSeriesColor**).
- Gradient colors (can be configured by **grGradient**).
- Size of WinDev file (can be configured by **grWndSize**).
- Size of WinDev window (can be configured by **grWndSize**).
- Size of WebDev image (can be configured by **grImageSize**).
- No label for the series (can be configured by **grSeriesLabel**).
- No caption for the elements (can be configured by **grCategoryLabel** and **grLabel**).
- No legend (can be configured by **grLegend**).
- No title for the chart (can be configured by **grTitle**).
- No title for the axes (can be configured by **grAxisTitle**).
- No gridlines (can be configured by **grGridLines**).
- No data (elements can be added by **grAddData**).
- No destination (can be configured by **grDestinationControl**, **grDestinationWnd** and **grDestinationWMF**).
- No display of the chart (the chart can be drawn by **grDraw**).
- No raised effect (can be configured by **grRaised**).
- For the pie charts, the start angle is equal to 0 (can be configured by **grPieStartAngle**).
- For the bar charts and the MinMax charts, the spacing between two categories is equal to 10 pixels (can be configured by **grColumnSpacing**).
- For the scatter charts, the points are linked and their size is equal to 1 (can be configured by **grScatterLinkPoint** and **grScatterPointSize**).

4.4 Functions for managing the charts

The following functions are used to manage the charts:

gr3DSParameter	Retrieves or modifies a parameter for drawing a 3D spatial chart
grAddData	Adds a data into a chart
grAutoRefreshCategoryLabel	Identifies or defines whether the category labels are updated whenever the chart is drawn
grAutoRefreshSeries	Defines whether a series is refreshed whenever the chart is drawn
grAxisTitle	Defines the title for one of the chart axes
grAxisTitleFont	Modifies the font used for the titles of the chart axes
grCategoryLabel	Initializes the label of a data category (or section) in a chart
grClearPict	Erases the drawing of a chart
grColor	Returns and initializes the color of the different chart elements.
grColumnSpacing	Indicates the spacing between each data category in a bar chart
grCreate	Creates a chart of a specific type
grCreateFont	Creates a font for the charts
grDeleteAll	Erases a chart and destroys this chart
grDeleteSeries	Deletes a data series from a chart
grDestinationBMP	Defines a file in BMP format as destination of the chart
grDestinationControl	Defines an image control as destination of the chart
grDestinationEMF	Defines a file in EMF format as destination of the chart
grDestinationWMF	Defines a meta file as destination of a chart
grDestinationWnd	Defines a window as destination of a chart
grDonutHolePercentage	Modifies the percentage corresponding to the radius of the hole in a donut chart.
grDraw	Draws a chart according to the specified parameters
grExist	Checks the existence of a chart in memory
grGradient	Displays the colors in gradient mode or not
grGraduate	Indicates the frequency of the marks on the vertical/horizontal axis of a chart
grGridlines	Displays or hides the gridlines of a chart
grImageSize	Modifies the size of the image containing the chart
grIncreaseData	Adds a value to a data in a chart
grInfoPoint	Returns the screen coordinates of a point or the screen coordinates of a value found in a chart.
grInfoXY	Returns information about the series found at a specific point of the chart
grLabel	Indicates various parameters for the additional legend in a chart
grLabelFont	Modifies the font used for the labels of a chart
grLegend	Indicates the presence and position of the legend in a chart
grLegendFont	Modifies the font used for the legend of a chart
grLineThickness	Changes the value of the line thickness in a line chart
grLoadParameter	Restores the parameters used to draw a chart.
grMask	Defines a display mask for the values of the additional legend
grMenu	Enables or disables the popup menu of a chart
grMenuOption	Modifies the text of an option in the popup menu of a chart
grOrientation	Modifies the orientation of the chart axes
grOrigin	Modifies the start and end marks on the horizontal/vertical axis
grOverlayChart	Allows you to display two different types of charts in the same chart
grParameter	Retrieves or modifies a parameter of a chart.
grPiePullOut	Pulls out a section in a pie chart

grPieStartAngle	Indicates the start angle of the first section in a pie chart
grPrint	Prints a chart
grRaised	Indicates the depth of the raised sections for the 3D charts
grRotation	Allows you to turn a surface chart around one of its axes
grSaveBMP	Saves a chart that was drawn beforehand in BMP format
grSaveEMF	Saves a chart that was drawn beforehand in EMF vectorial format
grSaveParameter	Saves the parameters of a chart as a compressed string
grSaveWMF	Saves a chart that was drawn beforehand in EMF vectorial format
grScatterAddDataXY	Adds a data into a "Scatter" chart
grScatterLinkPoint	Links (or not) the points found in a "Scatter" chart
grScatterPointSize	Defines the size of the points found in a "Scatter" chart
grScatterSeriesColor	Initializes the color of a series for a "Scatter" chart
grScatterSeriesLabel	Initializes the series label in a "Scatter" chart
grSeriesColor	Initializes the color of a series or the color of a section
grSeriesLabel	Initializes the label of a data series in a chart
grSeriesSecondaryAxis	Defines whether a series will be drawn on the secondary axis
grSmoothing	Smooths a chart with the "cubic splines" algorithm
grSourceCategoryLabel	Defines the source of the category labels
grSourceSeries	Defines the source of a series
grStockAddData	Adds a data into a stock chart
grSurfaceAddData	Adds a data into a Surface chart
grSurfaceAltitudeColor	Defines the color used for the altitude (Z axis) of a surface chart
grSurfaceDeleteAltitudeColor	Deletes the altitude colors.
grSurfaceDeleteMarkCaption	Deletes the captions of the marks found in a Surface chart.
grSurfaceGridlines	Displays the gridlines for the different planes of a surface chart
grSurfaceMarkCaption	Allows you to specify the caption of a mark for a Surface chart
grSurfaceMesh	Enables or disables the mesh on a surface chart
grTitle	Modifies the caption and/or the position of the chart title
grTitleFont	Modifies the font used for the title of a chart
grTooltip	Displays and formats the tooltip associated with a chart
grType	Modifies or returns the type of a chart
grWndSize	Defines the size of the window containing the chart

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

5. HANDLING THE MEMORY ZONES

WD WebDev

The following functions are used to manage the memory zones:

MemAdd	Adds an element into a memory zone
MemCount	Returns the number of elements found in a memory zone
MemCreate	Creates a memory zone
MemCurrent	Returns the subscript of the current element
MemDelete	Deletes an element from a memory zone
MemDeleteAll	Clears and deletes a memory zone
MemExist	Checks the existence of a memory zone
MemFirst	Positions on the first element of a memory zone and returns the value of this element
MemFound	Checks whether the sought element was found
MemKeyVal	Returns the value (added or modified by <i>MemAdd</i> or <i>MemModify</i>) of the current element in a memory zone
MemLast	Positions on the last element of a memory zone and returns the value of this element
MemModify	Modifies an element in a memory zone
MemNext	Positions on the next element of a memory zone and returns the value of this element
MemOut	Allows you to find out whether the browse performed on the memory zone is outside the memory zone
MemPrevious	Positions on the previous element of a memory zone and returns the value of this element
MemRetrieve	Retrieves the value of an element in a memory zone
MemSeek	Seeks an element in a memory zone
MemSetPosition	Positions the memory zone on an element
MemSort	Sorts the elements found in a memory zone

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

6. HANDLING THE EXTERNAL FILES

6.1 Overview

Several WLanguage functions are used to manage the external files.

An external file is a file with direct access, also called "text file". The external files can have any type (".TXT", ".INI", etc.). The external files can contain:

- readable characters (in character string format),
- non-readable characters (in binary format).

Caution: Do not confuse external files with HFSQL files. The HFSQL files are data files in a WinDev/

WebDev format and they must be handled by the HFSQL functions.

Three groups of functions are used to handle:

- the content of your external files,
- the files found on your disks (file copy for example),
- the directories of your disks (directory creation for example).

6.2 Handling the content of external files

You have the ability to:

- create, open and close your external files,
- manage the lock for your external files,
- read the content of your external files: read a line or a block of bytes (characters),
- write into your external files: write a line, a character string or a section of memory,
- move inside your external files.

Example

Your program manages a ".INI" file whose size

exceeds 64KB. This ".INI" file is too large to be handled by the INI functions. The functions for managing the external files enable you to handle this file.

To retrieve the content of this file: read the content of the file and assign the information read into a section of memory.

To modify the content of this file: add various information into this file at a given position.

This file must be partially locked in order to be updated: lock the file. This file will only be accessible by the application or the site that locks it.

6.3 Handling files

You have the ability to:

- compress and encrypt the files,
- find out the characteristics of a file,
- find out and modify the name and path of the files,
- compare the content of several files,
- list the files found in a directory by running a pro-

cedure that performs a process on each file.

Example

Your program manages large files containing confidential information. Encrypt and compress these files before transferring them by network.

Perform a search on your files and run a specific procedure on the files found.

6.4 Handling disks and their directories

You have the ability to:

- get information about the accessible directories and disks,

- handle the directories.

6.5 Functions for managing the external files

The following functions are used to manage the external files:

CompleteDir	Adds (if necessary) the "\" character at the end of a character string. This function is useful for building full file names when the format of the path is unknown (if the user enters a file path for example)
fAddBuffer	Adds the content of a buffer at the end of a file
fAddText	Adds the content of a character string at the end of a file
fAttribute	Identifies or modifies the attributes of a file
fAttributeReadOnly	Identifies or modifies the "Read-only" attribute of a file
fChangeSize	Resizes a file
fClose	Closes an external file
fCompare	Compares the content of two files bit by bit
fCompress	Compresses a file
fCopyDir	Copies a directory and its content
fCopyFile	Copies a file
fCopyFileWebFolder	Copies an image file from the data directory of the application (or from one of its sub-directories) to the "_WEB" directory of the application (or to one of its sub-directories).
fCreate	Creates a new external file
fCreateLink	Creates a link on a file at the specified location.
fCrypt	Encrypts a file in binary format or in ASCII format
fCurrentDir	Identifies or modifies the current directory
fCurrentDrive	Returns or modifies the current disk
fDataDir	Returns the full path of the directory for the HFSQL data files
fDataDirCommon	Returns a directory path for the shared data of the current application. This data is shared among all the users of the computer.
fDataDirUser	Returns a directory path for the data of the current application. This data is specific to the current user for the current application.
fDate	Returns or modifies the different dates associated with a file (creation, modification or access)
fDateTime	Returns or modifies the different dates and times associated with a file or a directory (creation, modification or access).
fDelete	Deletes a file accessible from the current computer
fDeleteFileWebFolder	Deletes an image file from the "_WEB" directory of the application (or from one of its sub-directories).
fDetectRemovableStorage	Detects whether a removable storage unit (CD, USB key, USB camera, ...) was added or removed
fDir	Finds a file or a directory
fDirAttrib	Identifies the access rights granted to a user for a specified directory
fDirAttribute	Returns the attributes of a directory
fDirectoryExist	Checks the existence of a directory
fDirSize	Returns the size (in bytes) of a directory.
fDriveInfo	Returns information about a disk
fDriveReady	Allows you to find out whether the floppy drive is available or whether the disk exists
fExeDir	Returns the full path of the runtime directory of the project
fExtractPath	Returns the different elements of a path: drive, directories, name and extension of the file
fExtractResource	Extracts a resource from the application into a physical location of the device

fFileExist	Checks the existence of an external file
fFind	Finds a character string or a buffer in a file opened by fOpen
fGlobalDirCommon	Returns a directory path for the global data of the current application (data shared among several applications), regardless of the current user.
fGlobalDirUser	Returns a directory path for the global data of the current application (data shared between several applications), for the current user.
fGraphicFilter	Returns the list of image formats supported by WinDev, in the format expected by the filter of fSelect
fImageSelect	Opens the image picker of Windows
fIsImage	Allows you to find out whether a file found on disk or a file contained in a buffer corresponds to a recognized image format
fListDirectory	Lists the directories found in a given directory (and in its sub-directories) and returns the number of listed directories
fListDisk	Returns the list of disks installed on the computer.
fListFile	Lists the files found in a directory (and in the sub-directories of the directory) and returns the number of listed files
fLoadBuffer	Loads the content of a text file in a buffer variable
fLoadText	Loads the content of a text file into any text control or text variable (string variable, edit control in a window or page, static control in a report, ...)
fLock	Entirely or partially locks an external file
fLongName	Returns the long name of a file or directory
fLongPath	Returns the full long path of a file or directory
fMakeDir	Creates a directory
fMoveDir	Moves a directory and its content
fMoveFile	Moves a file. You have the ability to rename it
fNameOfFile	Returns the name of an external file that is currently opened.
FolderWeb	Returns the path of the directory containing the images, the Javascript files, the Java applet files and the other files accessible from the browser
fOpen	Opens an external file
fOpenTempFile	Creates and opens a temporary file. A unique name is given by the system to each temporary file
fParentDir	Returns the path of the parent directory for the specified directory.
fRead	Reads the content of an external file and assigns it (or not) to a memory section
fReadLine	Reads a line in an external file
fReadLineRegularExpression	Reads a line in an external file and retrieves in variables the sections of this line according to a regular expression.
fRemoveDir	Deletes a directory from a disk
fRename	Modifies the name of a file
fReportsAndQueriesDir	Returns the full path of the directory for the custom or shared reports and queries
fResourceDir	Returns the path of the read-only resources of the application.
fSaveBuffer	Creates and fills a text file with the content of a string or buffer variable
fSaveText	Creates and fills a text file with the content of a text control or text variable (string variable, edit control in a window, static control in a report, ...)
fSeek	Returns and modifies the current position in an external file
fSelect	Opens the file picker of Windows
fSelectDir	Opens a directory picker
fSep	Returns the separator of directory according to the current platform ('\' or '/')
fSeparator	Returns a file path with normalized separators.
fShortName	Returns the short name of a file or directory
fShortPath	Returns the full short path of a file or directory

fSize	Returns the size of a file (in bytes)
fSizeUncompressed	Returns the size of a file before compression
fStopCompress	Stops the operation for compressing or decompressing a file. The compressed or decompressed file is not created
fTempFile	Returns the name of a unique temporary file
fTempPath	Returns the name of the directory where the system stores the temporary files
fTime	Returns or modifies the different times associated with a file: creation, modification or access
fTrackDirectory	Detects the modifications performed on the content of a directory
fTrackFile	Detects the modification of a file.
fTrackStop	Stops all current tracking of a file or directory
fTrackStopAll	Stops all the current tracks performed on the files and directories
fUncompress	Decompresses a file
fUnencrypt	Decrypts a file that was encrypted by fCrypt
fUnlock	Entirely or partially unlocks an external file
fWebDir	Returns the full path of the directory of Web objects (image, flash, ...)
fWrite	Writes a character string or a memory section into an external file
fWriteLine	Writes a line into an external file
UncompleteDir	Removes the "/" or "\" character found at the end of a string. This function is useful for building full file names when the format of the path is unknown (if the user enters a file path for example)

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

7. SHARED MEMORY ZONES

7.1 Overview

WinDev, WebDev and WinDev Mobile propose to handle the shared memory zones.

The shared memory zones are a communication mechanism between several applications on a given computer. They are used to easily exchange data between a service and the setting application for example.

The functions for managing the shared memory zones use the standard APIs of the different operating systems (Windows or Linux) and they facilitate the interoperability between the WinDev/WebDev/WinDev Mobile applications and the applications developed in other languages.

7.2 How do I proceed?

7.2.1 Creating a shared memory zone

The shared memory zones are created and opened by `fMemOpen`.

The first application that accesses the zone will provoke its creation while the other applications will perform a simple opening.

The application that creates the zone defines its size by supplying a minimum size. The operating system will create a zone whose size is a multiple of the size of the memory pages (4 KB sous in Windows 32 bits for example). The actual size of the memory zone is returned by `fSize`.

```
// Open shared memory zone
ZoneID is int
ZoneID = fMemOpen("MyZone"+ ...
    "Shared", 50, shareGlobal)
IF ZoneID>0 THEN
    ZoneSize is int
    ZoneSize = fSize(ZoneID)
    Info(StringBuild(...
        "The %1 memory zone " +...
        "was opened."+ ...
        "Its size is %2"+...
        " bytes.", ZoneID, ZoneSize))
END
...
// Close the shared memory zone
fClose(ZoneID)
```

7.2.2 Finding out whether a shared memory zone already exists

In some cases, it may be useful to find out whether a shared memory zone already exists.

For example, an application used to configure a service can check whether the service is properly started by doing the following: simply check that the service created a specific shared memory zone. If the zone does not exist, an error message can be displayed by the application.

`fMemExist` allows you to find out whether a shared memory zone was already created by another application.

```
// Check the existence of the
// shared memory zone
IF fMemExist("MySharedZone", ...
    shareGlobal)=False THEN
    Error("Check that the XXX " +...
        service was properly started.")
    RETURN
END
```

7.2.3 Handling the content of a shared memory zone by programming

To handle the content of a shared memory zone, you can:

- use the functions for reading and writing in the external files. This solution is recommended for the simple cases (transmission of a string for example).
- use the automatic serialization of `WLanguage` (Serialize and Deserialize), then use the functions for reading and writing in the external files to transmit the elements. This solution is recommended when transmitting classes whose members correspond to the elements to transmit

7.2.4 WLanguage functions

The following functions for managing the external files can be used with the shared memory zones:

fRead	Reads: <ul style="list-style-type: none"> • a block of bytes (characters) in an external file (ANSI or Unicode), • the content of an external file (ANSI or Unicode) and assigns it to a memory zone.
fReadLine	Reads a line in an external file (in ANSI or UNICODE format).
fSeek	Returns and modifies the current position in an external file.
fWrite	Writes a character string into an external file and in memory.
fWriteLine	Writes a line into a text file (in ANSI or UNICODE format).

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

7.3 Dialog between several applications

Two applications can share data by using a shared memory zone.

Two synchronization mechanisms are available:

- the automatic notification (by using the callback functions)
- the manual synchronization via a semaphore

7.3.1 Automatic notification of modifications

The automatic notification of modifications can be implemented as soon as a shared memory zone is opened.

To do so, each application that accesses the memory zone must give the name of WLanguage procedure as the last parameter of the fMemOpen function. This procedure will be automatically called whenever the content of the shared memory zone is modified.

To wait for the other applications to be done processing the notification (before performing a new modification of the memory zone for example), you must use fMemWait.

```
// Open memory zone
// in application 1
ZoneID is int
ZoneID = fMemOpen...
("SharedZone", 200,...
shareGlobal, "ModifyMemory")
-- -- Procedure used for
callback (application 1)
PROCEDURE ModifyMemory...
(ModifZoneName is string,...
IdModifZone is int)
Info(StringBuild(...
PROCEDURE ModifyMemory...
(ModifZoneName is string,...
```

```
IdModifZone is int)
Info(StringBuild(...
"The %1 zone was"+ ...
"modified.",ModifZoneName))
// Open the memory zone
// in application 2
ZoneID is int
ZoneID = fMemOpen(...
"SharedZone", 200,...
shareGlobal, "ModifyMemory")
// Write into the zone
fWrite(ZoneID,"Hello, "...
"I am the application 2.")
// Wait for the validation
// of "Info" of the procedure ...
// "ModifyMemory" in ...
// application 1
fMemWait(ZoneID)
// Second write operation into the
zone
fWrite(ZoneID, ...
"New information.")
```

7.3.2 Manual synchronization

The manual synchronization can be performed according to the following principle:

- The applications open the shared memory zone and the dialog semaphore.
- The first application "takes" the semaphore (SemaphoreStart).
- The first application writes its data into the memory zone.
- The first application frees the semaphore (SemaphoreEnd).

- The second application "takes" the semaphore then.
- The second application reads the data written by the first application.
- The second application frees the semaphore.

Note: if the two applications want to exchange data, you need to use semaphores to regulate the

exchange.

Advice: The manual synchronization will be used preferably when:

- one of the two applications is not written in WLanguage.
- the mechanism for notification of modifications is not available.

7.4 Naming the shared memory zones

The shared memory zones are identified by their name.

7.4.1 Managing the share mode

The share mode differs according to the versions of the operating systems:

- Linux, Windows 2000 and earlier: there is only one memory zone creation space. The <Share> parameter of the fMemOpen function is ignored.
- Windows XP: the <Share> parameter of the fMemOpen function is effective if the quick user change service is enabled, otherwise it is ignored.
- Windows Vista and later: the <Share> parameter of the fMemOpen function is ignored. The services and the users are located in a different space. To share a memory zone between a service and an application in the session of a user, you must use the shareGlobal constant.
- In Terminal Server: the <Share> parameter of the fMemOpen function is supported. Each session opened on the Terminal Server has a different memory space. Applications running in the same TSE session can share a memory zone created with the shareUser constant. Applications located

in different sessions can share a shared memory zone created with the shareGlobal constant.

7.4.2 Correspondence between the name provided to fMemOpen and the opening in C

The two following code examples present the opening of a memory zone (named "myzone" in WLanguage and in C).

```
// Code in WLanguage
// ZoneID is int
ZoneID = fMemOpen("myzone",1024,...
shareGlobal)
```

```
// Equivalent code in C
char * szZoneName = "myzone";
int nSize;
int nMem;
key_t nKey;
int nAccess = 0666;
// Opening in read/write

nSize = 1024;
nKey = ftok(szZone-
Name+sizeof(char),...
(int) szZoneName[0]);
nMem = shmget(nKey, nSize ,...
nAccess);
```

7.5 Functions for managing the shared memory zones

The following functions are used to manage the memory zones:

fMemExist	Checks the existence of a shared memory zone
fMemOpen	Opens a memory zone shared between several applications
fMemWait	Waits for the end of process about the notifications of modification of a shared memory zone

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

8. PRINTING IN WLANGUAGE

8.1 Overview

Several methods can be used for printing:

- The report editor, used to create "Preset reports".
- The print functions in WLanguage.

The report editor

The report editor is used to create printed reports (in report or label format) containing information coming from data files.

The print functions in WLanguage

For special printing, the report editor may not be able to create the requested reports. In this case, you must use the print functions of WLanguage.

Furthermore, for very simple prints (text or images), you don't even need to use the report editor.

Notes:

- The print functions are not available in external language.
- This chapter describes the principle for printing in WLanguage.

WB Important: Using the print functions of WLanguage requires a specific configuration of your server. See the online help for more details.

8.2 Principle for printing in WLanguage

The steps for printing in WLanguage are as follows:

- **Step 1 (optional):** Configuring the print parameters.
- **Step 2 (optional):** Creating the print fonts.
- **Step 3:** Printing the characters, lines or images and ending the print.
- **Step 4:** Starting the print (`iEndPrinting`).

Tip: To speed up your prints, limit the graphic options (frame box, etc.) that really slow down your prints and that are memory intensive.

8.2.1 Step 1: Configuring the print parameters

This step allows you to choose:

- The print settings of the document (quality, number of copies, etc.).
- The settings for printer configuration (printer driver, orientation, etc.).

This step is optional and it should be performed only if the default parameters (parameter defined in the control panel of Windows) are not suitable for the print.

The different operations possible

The possible operations are as follows:

- 1. Selecting the print settings of the document (`iParameter`).** You can:
 - open the standard window for configuring the print (WinDev and WinDev Mobile),

- configure the print parameters one by one (orientation, paper format, height and width of the page, number of copies, etc.).

2. Choosing the printer used for printing (`iConfigure`).

You can:

- open the standard window for printer selection (WinDev and WinDev Mobile),
- modify the printer used for the current print. This modification can be temporary (for the current program only) or permanent (modification of the default printer in the control panel of Windows).

3. Redefining the print margins (`iMargin`).

The top, bottom, left and right margins can be modified. If this function is not called, the default margins (different depending on the printer) will be used.

Caution: The order of these operations must be respected. If a print is in progress, `iParameter` and `iConfigure` cancel the current print.

Functions used to configure the destination of the print

The following functions are used to configure the print destination and to find out the current configuration:

- `iPreview/iDestination`: Configures the destination of the print.
- `iConfigure`: Configures the printer.
- `iInfoPrinter`: Retrieves the characteristics of the current printer or the characteristics of the default printer.

- **iListPrinter**: Returns the list of printers installed on the computer.
- **iParameter**: Configures the print.

The configuration and the setting performed by **iConfigure** and **iParameter** re-initialize the print module. Therefore, these functions must be the first print functions called. All the print functions called before these two functions will be ignored.

8.2.2 Step 2: Creating print fonts

This step allows you to define and choose the different fonts that will be used in the printed document. You can choose and define:

- **specific print fonts**: these fonts can be used when printing with the WLanguage functions.
- **dynamic fonts** that can be used for display (in the windows, drawings, etc.) and when printing in WLanguage.

This step is optional.

The default font is used if no font is specified for the print. This font has the following characteristics:

- Font: Don't care
- Font #0
- Size: 12 pica points
- Normal style
- Color: Black

Print font

The print fonts are created by **iCreateFont**. This function enables you to choose:

- the typeface (name and family),
- the size (of the characters),
- the attributes (bold, italic, underlined, etc.),
- the color.

The different parameters of the font are defined via several constants.

Each font is associated with a number. To choose the print font, all you have to do is use **iFont**.

Dynamic font

A "Font" variable allows you to create a dynamic font that can be used in all the project objects: windows, pages, reports, prints, charts, drawings, etc.

To define a dynamic font:

1. Create a Font variable.
2. Define the characteristics of the font:
 - by using **FontCreate**,
 - via the font properties: **...Orientation**, **..StrikeOut**, **..Condensed**, **..Color**, **..Extended**, **..Bold**,

..Italic, **..Large**, **..Name**, **..Underline** and **..Size**.

To choose a print font, all you have to do is use **iFont**.

Printer font

The printer manufacturers install fonts on their printers. These fonts are called "printer" fonts.

Unlike the software fonts (such as "TrueType"), these fonts are directly accessible by the printer, they do not have to be downloaded.

In the lists of fonts, the printer fonts are identified by a "printer" icon displayed in front of the name of the font.

Caution: using printer fonts may produce unexpected display effects in the print preview.

If you are using printer fonts, these fonts will be interpreted by the printer only. During the print preview, the display of the fonts is managed by Windows. As it does not support these specific fonts, Windows uses the nearest display font.

8.2.3 Step 3: Printing

This step consists in "sending to the printer" the different elements to print: text, images, drawings, etc.

The print on the requested support (configured by **iPreview**) will be effective at the end of print (**iEndPrinting**).

Printing text

The main WLanguage functions used to print texts are as follows:

- **iTextHeight**: Calculates the height of the font for the text to print (in millimeters).
- **iZoneHeight**: Calculates the necessary height of a fixed-width area in order for the entire text to be printed in this area.
- **iPrint**: Sends the character string passed in parameter to the print "buffer".
- **iPrintBarcode**: Prints a bar code inside a rectangle.
- **iPrintWord**: Sends the character string passed in parameter to the print "buffer".
- **iPrintZone**: Prints a text in a rectangular area.
- **iPrintZoneRTF**: Prints a text in RTF format in a rectangular area.
- **iTextWidth**: Calculates the width of the text to print in millimeters, according to the specified font.
- **iLink**: Prints a link to a URL when printing an HTML page.

- **iFont**: Selects the default font.
- **iDidotFont**: Selects the unit that must be used for the height of fonts (created by **iCreateFont**): DIDOT point or PICA point.
- **iXPos**: Used to manage the horizontal position (x coordinate) of the print cursor in the page.
- **iYPos**: Used to manage the vertical position (y coordinate) of the print cursor in the page.

Notes:

- The texts can be entered in different fonts: the font is selected by **iFont**.
- The texts can be printed at specific positions: **iXPos** and **iYPos**.

Printing images

Several WLanguage functions are used to print images while respecting their size.

- **iImageHeight**: Calculates the height of the image to print (in millimeters).
- **iPrintBarCode**: Prints a bar code inside a rectangle.
- **iPrintImage**: Sends the image file to print to the print "buffer".
- **iImageWidth**: Calculates the width of the image to print (in millimeters).
- **iTransparentMagenta**: Modifies the management of the Magenta color so that it is considered as being transparent or not during the next prints.
- **iXPos**: Used to manage the horizontal position (x coordinate) of the print cursor in the page.
- **iYPos**: Used to manage the vertical position (y coordinate) of the print cursor in the page.

Printing drawings

The following WLanguage functions are used to easily print different shapes:

- **iBorder**: Prints a border at the specified coordinates.
- **iEllipse**: Prints an ellipse inside a rectangle.
- **iPrintBarCode**: Prints a bar code inside a rectangle.
- **iNewLine**: Prints a line (in Windows standard, with a rounded end) at the specified coordinates.
- **iTransparentMagenta**: Modifies the management of the Magenta color so that it is considered as being transparent or not during the next prints.
- **iMMToPica**: Converts the coordinates of a point

(expressed in millimeters) into system coordinates.

- **iXPos**: Used to manage the horizontal position (x coordinate) of the print cursor in the page.
- **iYPos**: Used to manage the vertical position (y coordinate) of the print cursor in the page.
- **iLine**: Prints a line at the specified coordinates.
- **iHLine**: Prints an horizontal line at the specified coordinates.
- **iVLine**: Prints a vertical line at the specified coordinates.

Other functions

Several WLanguage functions are used to optimize the prints performed in WLanguage. You have the ability to manage:

- the page break,
- the print cancelation by the user,
- the size of the page.
- **iWindowCancel**: Configures the display of the window used to cancel the current print (WinDev only).
- **iEndPrinting**: Signals the end of the document to print and actually starts printing the data stored in the printer spooler.
- **iPageHeight**: Calculates the printable height of the page while taking the margins into account.
- **iPageHeightLeft**: Calculates the remaining available height on the current page while taking the top and bottom margins into account.
- **iDocumentCanceled**: Used to find out whether the user has requested to cancel the print of the current document (WinDev only).
- **iDocumentPrinted**: Used to find out whether the user has requested to print the document from the print preview (WinDev only).
- **iPageWidth**: Calculates the width of the printable page while taking the margins into account.
- **iPageNum**: Returns or reinitializes the number of the page currently printed.
- **iReset**: Re-initializes the print parameters stored in the print library.
- **iSkipLine**: Forces a line break.
- **iSkipPage**: Generates a page break.
- **iReportPrintingStatus**: Returns the status of the current print.

8.3 Print functions

The print functions are as follows:

iAddBookmark	Adds a bookmark into the print preview or during the export in PDF.
iBorder	Prints a border at the specified coordinates
iChangeSubReport-Source	Modifies by programming the report associated with a sub-report in a composite report
iCloseReport	Stops printing the current report immediately
iColumnEnd	Forces a column break in a multicolumn report
IColumnNum	Returns the number of the current column in the multi-column reports.
iConfigure	Configures the printer
iConfigureReport	Configures the printer by opening the configuration window
iCreateFont	Creates a new print font
iDestination	Configures the destination of the print
iDidotFont	Selects the unit that will be used for the height of the fonts (created by iCreateFont): DIDOT point or PICA point
iDirImageHTML	Used to select the directory of generated images when printing in HTML format
iDocumentCanceled	Allows you to find out whether the current print was canceled by the user
iDocumentPrinted	Allows you to find out whether the user has requested to print the document from the print preview
iEllipse	Prints an ellipse inside a rectangle
iEndPrinting	Indicates the end of the document to print and actually starts printing the data stored in the printer spooler
iEndReport	Forces a report created in the report editor to stop from being printed
iEscape	Sends an ESCAPE command to a printer
iFont	Selects the default font
iForceComplement	Forces a body complement block to print
iGroupAdd	Adds a report to a group of reports.
iGroupConfigure	Modifies the parameters of a report found in a group of reports.
iGroupPrint	Starts printing a report found in a group of reports
iHLine	Prints an horizontal line at the specified coordinates
iImageHeight	Calculates the height of the image to print (in millimeters)
iImageWidth	Calculates the width of the image to print (in millimeters)
iInfoPrinter	Retrieves the characteristics of the current or default printer
iInitReportQuery	Initializes the query linked to the report (for a query with parameters)
iInitReportQueryConnection	Initializes the query linked to the report (query based on a specific connection)
iInitSubReport	Initializes the parameters of a sub-report of a composite report
iLastFile	Returns the full name of the last file generated during the print (PDF, RTF, XML, ...)
iLine	Prints a line at the specified coordinates
iLink	Prints a link toward a URL when printing an HTML page
iListNestedReports	Returns the list of nested reports currently printed
iListPrinter	Returns the list of printers installed on the current computer
iMargin	Defines the "logical" print margins
iMMToPica	Converts the coordinates of a point (expressed in millimeters) into system coordinates
iNestedHeaderFooter	Prints (or not) the page headers and footers of the nested report
iNewLine	Prints a line (in Windows standard, with a rounded end) at the specified coordinates
iPageEnd	Forces the move to the next page when printing a report

iPageHeight	Calculates the height of the printable page (in millimeters) while taking the margins (top or bottom) into account
iPageHeightLeft	Calculates the available height (in millimeters) left on the current page while taking the margins (top or bottom) into account
iPageNum	Returns or initializes the number of the page currently printed
iPageWidth	Calculates the width (in millimeters) of the printable page while taking the margins (left or right) into account
iParameter	Configures the print
iParameterDuplicate	Configures the automatic creation of the duplicate copy generated during the next print.
iParameterExport	Configures the export options from the print preview (options for sending emails for example).
iParameterPDF	Defines the protection options for the generated PDF file
iParameterPreview	Configures the display of each button found in the print preview
iParameterReport	Configures the printer by opening the configuration window
iParameterWatermark	Adds a watermark into a report (report created in the report editor or printed report).
iParameterXLS	Defines the options for the XLS file generated during the print. The current layout (color and font) can be taken into account or not.
iPicaToMM	Converts the coordinates of a point (expressed in system coordinates) into millimeters
iPreview	Configures the destination of the print
iPrint	Sends the character string passed in parameter to the print "buffer"
iPrintBarCode	Prints a bar code inside a rectangle
iPrintBlock	Forces a report block to print
iPrintBodyComplement	Forces a body complement block to print while specifying the height of the complement block
iPrintDuplicate	Prints a duplicate copy.
iPrintImage	Sends the image file to the print buffer
iPrintPDF	Prints the content of a PDF file from an iPhone/iPad application. When this function is called, a system window is displayed, allowing the user to select the printer to use, the number of copies, ...
iPrintReport	Prints a report created in the report editor
iPrintWord	Sends the character string passed in parameter to the print "buffer"
iPrintZone	Prints a text in a rectangular area
iPrintZoneHTML	Prints a text in HTML format in a rectangular area
iPrintZoneRTF	Prints a text in RTF format in a rectangular area
iReportPrintingStatus	Returns the status of the current print
iReset	Reinitializes the print parameters stored in the print library
iRoundBorder	Prints a round border at the specified coordinates
iSelectFont	Creates a new print font from the standard window for font selection of Windows
iSequencingAdd	Adds a report into a sequencing of reports.
iSequencingPrint	Prints a sequencing of reports.
iSkipLine	Forces a line break.
iSkipPage	Generates a page break
iSubstBlock	Substitutes a report block for another one when printing the report
iTextHeight	Calculates the height of the font for the text to print (in millimeters)
iTextWidth	Calculates the width (in millimeters) of the text to print according to the specified font
iTransparentMagenta	Specifies whether the Magenta color will be considered as being transparent (or not) in the future prints

iVLine	Prints a vertical line at the specified coordinates
iWindowCancel	Configures the display of the window used to cancel the current print
iXPos	Used to manage the horizontal position (X coordinate) of the print cursor in the page
iYPos	Used to manage the vertical position (Y coordinate) of the print cursor in the page
iZoneHeight	Calculates the necessary height of a fixed-width area in order for the entire text to be printed in this area

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

8.4 PDF functions

WinDev allows you to print PDF files with the print functions. These PDF files can be handled by programming via the PDF functions:

PDFIsProtected	Checks whether the PDF file requires a password in order to be read.
PDFMerge	Merges several existing PDF files into a single PDF file
PDFMergeWithPassword	Merges several existing and password-protected PDF files into a single PDF file
PDFNumberOfPages	Returns the total number of pages found in a PDF file.
PDFToText	Extracts the text found in a PDF file.

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9. MANAGING WINDOWS

9.1 Overview

Several WLanguage functions are used to manage Windows.

You have the ability to manage:

- the registry with the registry functions,
- the Windows clipboard,
- the Windows recycle bin,
- the mouse,
- the serial and parallel ports,
- Twain devices,
- USB devices,
- the system,
- the services,
- Windows,
- Windows events,
- Java applications,
- the executables,
- the DDE dialog,
- the agents,
- the network,
- the SNMP protocol,
- the project,
- the scheduler.

9.2 Functions for managing the registry

The following functions are used to manage the registry:

RegistryCopyKey	Copies a registry key with all its sub-keys and values
RegistryCreateKey	Creates a key in the Windows registry
RegistryDeleteKey	Deletes a key from the Windows registry
RegistryDeleteValue	Deletes a value from the Windows registry
RegistryExist	Checks the existence of a key in the Windows registry
RegistryFirstSubKey	Identifies the path of the first sub-key for the specified key in the registry of Windows
RegistryListKey	Lists the sub-keys of a registry key.
RegistryListValue	Returns the name and the type of values of a key found in the registry.
RegistryNextKey	Identifies the key found after the specified key in the Windows registry
RegistryQueryValue	Reads the value of a register in the Windows registry
RegistryRename	Renames a registry key.
RegistrySeek	Finds a character string in the registry
RegistrySetValue	Writes a value into a register of the Windows registry
RegistryValueType	Returns the type of a value found in the registry.

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.3 Functions for managing the clipboard

WD WDMobile

The following functions are used to manage the clipboard:

Clipboard	Retrieves the text or the image found in the Windows clipboard
ClipboardFormat	Identifies the format of the information found in the clipboard
ClipboardHeight	Calculates the height of an image (in bitmap format) found in the clipboard
ClipboardRTF	Retrieves an RTF string from the clipboard.
ClipboardUnicode	Returns the text found in the clipboard in UNICODE format.
ClipboardWidth	Calculates the width of an image (in bitmap format) found in the clipboard
TableToClipboard	Copies the content of a table to the clipboard.
ToClipboard	Writes text information into the Windows clipboard
ToClipboardRTF	Writes a character string in RTF format into the clipboard.
TwainToClipboard	Enables you to either copy the document coming from the Twain device into the clipboard, or to view the document coming from the Twain device.

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.4 Functions for managing the recycle bin

WD

The following functions are used to manage the recycle bin:

RecycleBinClear	Clears the recycle bin.
RecycleBinDelete	Deletes a file from the recycle bin
RecycleBinListFile	Lists the files found in the recycle bin
RecycleBinRestore	Restores a file found in the recycle bin

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.5 Functions for managing the mouse

WD WDMobile

The following functions are used to manage the mouse:

ControlOver	Identifies the control hovered by the mouse cursor.
CursorCapture	Directs all the events associated with the mouse toward a specific window or a control
CursorDisplay	Displays or hides the mouse cursor
CursorPos	Returns and modifies the position of the mouse cursor
Hourglass	Changes the mouse cursor into an hourglass (and conversely)
MouseXPos	Returns the horizontal position (x-coordinate) of the mouse cursor in relation to the control
MouseYPos	Returns the vertical position (Y-coordinate) of the mouse cursor in relation to the control

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.6 Functions for checking the spelling

The following functions are used to manage the spelling checker :

SpellAddDictionary	Adds a word to the dictionary of OpenOffice for the instance of the current application.
SpellAvailable	Indicates whether an OpenOffice dictionary is available for the spelling checker.
SpellCheck	Indicates whether the spelling of a word is correct.
SpellSuggest	Returns the list of possible words found in the dictionary of OpenOffice..

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.7 Speech recognition functions

WD WDMobile

The following functions are used to manage the speech recognition:

SpeechRecognitionAddCommand	Adds a voice command into the current window.
SpeechRecognitionDeleteCommand	Deletes a voice command that was added by AddVoiceCommand
SpeechRecognitionTrigger	Triggers the service for speech recognition on the device.

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.8 Functions for managing the serial and parallel ports

WD WDMobile

The following functions are used to manage the serial ports and the parallel ports:

sClose	Closes the specified serial port or parallel port
sComputeCrc16	Checks a character string before and after transmission between WinDev applications
sComputeCrc32	Checks a character string before and after transmission between WinDev applications
sEndEvent	Disables the detection of an event on a serial port.
sEscape	Starts different functions that directly affect the bits of the serial port or parallel port, independently of the communication protocol
sEvent	Branches an event on a serial port.
sInEntryQueue	Retrieves the number of pending bytes in the input buffer of a specified serial port
sInExitQueue	Retrieves the number of bytes waiting to be transferred into the output buffer of the specified serial or parallel port
sOpen	Opens and initializes the specified serial port (or parallel port)
sParameter	Defines or modifies the configuration parameters of the specified serial port or parallel port
sRead	Reads a character string in the entry buffer of the specified serial port
sWrite	Writes a character string into the output buffer of the specified serial port or parallel port

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.9 Functions for managing Twain devices

WD

The following functions are used to manage the Twain devices:

TwainCurrentSource	Allows you to find out and change the Twain device used by default.
TwainListSource	Returns the list of drivers for the Twain devices connected to the current computer.
TwainScanZone	Allows you to define or re-initialize the area to scan.
TwainSelectSource	Displays the list of Twain devices available for the current computer and allows you to choose the Twain device used by default.
TwainSourceHeight	Returns the height of the area to scan.
TwainSourceWidth	Returns the width of the area to scan.
TwainState	Returns the status of the current source.
TwainToBMP	Allows you to: <ul style="list-style-type: none"> • save the document coming from the Twain device in a Bitmap file (".BMP" extension). • view the document coming from the Twain device in the user interface of the device.
TwainToClipboard	Allows you to: <ul style="list-style-type: none"> • copy the document coming from the Twain device into the clipboard. • view the document coming from the Twain device in the user interface of the device.
TwainToControl	Allows you to: <ul style="list-style-type: none"> • view the document coming from the Twain device in an image control. • view the document coming from the Twain device in the user interface of the device.
TwainToGIF	Allows you to: <ul style="list-style-type: none"> • save the document coming from the Twain device in a GIF file. • view the document coming from the Twain device in the user interface of the device.
TwainToJPEG	Allows you to: <ul style="list-style-type: none"> • save the document coming from the Twain device in a JPEG file. • view the document coming from the Twain device in the user interface of the device.

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.10 USB functions

WD

The following functions are used to manage the USB devices:

USBDetectRemovableStorage	Detects whether a removable storage unit (CD, USB key, USB camera, ...) was added or removed
USBDisconnect	Disconnects or ejects a USB device
USBFind	Finds a USB device according to keywords
USBListDevice	Returns the list of USB devices currently plugged into the current computer
USBProperty	Retrieves the value of a property for a USB device

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.11 MCI functions

The following functions are used to manage the MCI (Media Control Interface):

MCIBegining	Allows you to position at the beginning of a multimedia file
MCIclose	Closes a multimedia file
MCIEnd	Allows you to position at the end of a multimedia file
MCIErr	Returns the number of the last error caused by a MCI function
MCIExecute	Runs a MCI command
MCIHeight	Returns the height of a multimedia file (in pixels)
MCImsgErr	Returns the caption of the last error that occurred on a MCI function
MCIOpen	Opens a multimedia file
MCIPause	Allows you to pause in the execution of a multimedia file
MCIPlay	Plays a multimedia file
MCIPosition	Allows you to find out or modify the position of a multimedia file
MCIPositionFormat	Defines the format of the position for a multimedia file
MCIRetrieve	Returns the last result returned by MciExecute
MCISize	Allows you to find out the size of a multimedia file (expressed in the unit defined by MciPositionFormat)
MCIStatus	Allows you to find out the status of a multimedia file
MCIStopPlaying	Stops the execution of a multimedia file
MCITrack	Returns the number of the current track
MCITrackCount	Returns the number of tracks in the multimedia file
MCIVolume	Modifies or identifies the sound volume for the "MID" or "WAV" files
MCIWidth	Returns the width of a multimedia file (in pixels)
MCIWindowSize	Modifies the size and position of the display window of a multimedia file

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.12 Service functions

The following functions are used to manage the Windows services:

EndService	Ends the execution of the current service.
ServiceContinue	Restarts a paused service.
ServiceControl	Sends a control message to a service.
ServiceExist	Checks whether a service is installed.
ServiceInfo	Returns the characteristics of a service
ServiceInstall	Installs a service according to the information given in the members of the Service variable.
ServiceList	Returns the list of services found on the current computer or on the specified remote computer.
ServiceModify	Modifies the configuration of a service according to the information given in the members of the Service variable.
ServicePause	Pauses a service.
ServiceRefresh	Asks a service to re-read its configuration information.
ServiceRestart	Restarts a service that was stopped beforehand.
ServiceStart	Starts a service.

ServiceStatus	Returns the current status of a service
ServiceStop	Stops a service.
ServiceUninstall	Uninstalls a service.
ServiceWait	Pauses the current service during the specified duration.
ServiceWaitStatus	Waits for a service to be in a specific status.
ServiceWriteEventLog	Writes an event into the log of Windows events.

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.13 System functions

The system functions are as follows:

SysAndroidVersion	Returns information about the Android version used by the application
SysChangeEnvironment	Modifies the environment variables of the operating system found on the current computer
SysChangeScreenResolution	Modifies the resolution of a screen.
SysColor	Identifies or modifies the color of a Windows element
SysColorRes	Returns the resolution of the screen in number of colors
SysDir	Returns the path of a system directory
SysDirStorageCard	Returns the list of storage cards found in the current Pocket PC
SysEnvironment	Returns the environment variables of the operating system
SysErrorMode	Modifies the display mode of errors
SysGetDC	Retrieves the DC (Device Context) of a window, control or screen
SysIconAdd	Adds an icon into the taskbar
SysIconAddImage	Adds an icon that is superimposed over the icon of the application in the taskbar. This icon can be deleted by SysIconDeleteImage .
SysIconDelete	Deletes an icon from the taskbar
SysIconDeleteImage	Deletes the icon that is superimposed over the icon of the application in the taskbar.
SysIconize	Allows you to find out whether a window is minimized
SysIconModify	Modifies the icon file and/or the rollover message of an icon in the taskbar
SysIdentifier	Returns the unique identifier of the current mobile device.
SysIMEI	Returns the IMEI number of a Smartphone
SysInstance	Returns the instance of an application
SysIOSVersion	Returns information about the iOS version used by the application
SysListScreen	Returns the list of screens connected to one or more video cards.
SysListScreenOrientation	Allows you to find out the possible orientations of a Pocket PC screen.
SysListScreenResolution	Returns the list of possible resolutions for a screen.
SysListVideoCard	Returns the name of the video cards connected to the current computer.
SysMetric	Returns the resolution of a specific element
SysNameExe	Returns the name and full path of an application
SysNameMainScreen	Returns the name of the main screen
SysNumberScreen	Returns the number of screens currently connected to the current computer
SysRecentDocAdd	Adds a file into the list of recently opened documents. This list is managed by the shell of Windows.

SysRecentDocList	Returns the list of recently opened documents.
SysReleaseDC	Frees the current DC (Device Context) retrieved by SysGetDC
SysScreenOrientation	Allows you to find out and modify the orientation of the Pocket PC screen
SysScreenRectangle	Returns the coordinates of the display rectangle corresponding to all the screens
SysScreenResolution	Returns the characteristics of the screen resolution
SysSerialNum	Returns the serial number of the current Pocket PC
SysSetFocus	Gives focus to a window or to a control
SysShutdown	Stops the system or closes the current Windows session
SysSpace	Returns the total amount of memory for the current computer
SysStandby	Allows you to find out and modify the amount of idle time before the current Pocket PC switches to standby mode. This function is also used to enable and disable the standby mode for a Pocket PC
SysStatusStorage-Card	Returns the availability status of the storage card on the device.
SysThème	Returns the visual theme currently displayed on the device running Windows Phone 7
SysThumbnailAd- dButton	Adds a button to the thumbnail of the application in the taskbar.
SysThumbnailDele- teAll	Deletes all the buttons associated with the thumbnail of the application from the taskbar.
SysThumbnailDelete- Button	Deletes one of the buttons found in the application thumbnail in the taskbar.
SysThumbnailModify- Button	Modifies one of the buttons found on the thumbnail of the application in the taskbar.
SysValidHandle	Checks the validity of a window handle
SysVersion	Returns information about the PHP version used on the current server
SysWinActive	Returns the handle of the active window or activates a window
SysWindowsVersion	Returns information about the Windows version used on the current computer
SysWinHandle	Returns the handle of a window identified by its title
SysWinShow	Modifies the display status of a window
SysWinTitle	Returns the title of a window
SysXRes	Returns the horizontal resolution of the screen
SysYRes	Returns the vertical resolution of the screen

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.14 Miscellaneous Windows functions

The Windows functions are as follows:

API/CallDLL32	Runs a function found in an external 32-bit DLL
APIParameter	Modifies the options for calling API and CallDLL32
CallInterface	Runs a method of an interface of an object implemented in a DLL external to the WinDev framework. This object can be a C++ object or a COM object.
COMCallMethod	Runs a method of an interface of a COM object instantiated by COMCreateInstance.
COMCreateIns- tance	Instantiates a COM object (Component Object Model).
ComponentLoad	Loads the specified component in memory

COMQueryInterface	Call the QueryInterface method of the specified COM object to get a specific interface of this object so that it can be used with COMCallMethod.
CreateShortcut	Creates a shortcut on the Windows desktop, in the "Start" menu of Windows or in a specific directory
DeleteShortcut	Deletes a shortcut that was created by CreateShortcut
FreeDLL	Frees the 32-bit library (DLL) that was loaded in memory by LoadDLL
HiWord	Returns the two high bytes of an integer
IconInstall	Creates an icon (associated with a program) in a group of Windows programs, with the associated command line
Instance	Returns the "System handle" (HINSTANCE) of the current program
KeyPressed	Checks which key is pressed
LoadDLL	Loads the specified 32-bit library (DLL) in memory
LoWord	Returns the two low bytes of an integer
MakeInteger	Builds an integer from two 2-byte integers
OpenSSLCheck	Checks, via a public key, that the signature of data performed with the OpenSSL library is correct
Ping	Checks whether an address is accessible (similar to the PING network command)
sComputeCrc16	Checks a character string before and after transmission between WinDev applications and WebDev sites
sComputeCrc32	Calculates the Cyclical Redundancy Check (CRC) of a buffer.
SendKey	Simulates keystrokes
StandardOutput	Writes an information into the standard output stream "stdout" (also called "console")
Transfer	Copies a block of bytes or a fixed-length string from a memory address into another one
WindowsVersion	Returns the runtime mode of the program in the current environment

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.15 Windows event

The following functions are used to manage the Windows events:

EndEvent	Cancels the interception of a Windows event on a control or a WinDev window implemented by the Event function
EndTimer	Ends the execution of a timer triggered by Timer
EndTimerSys	Ends the execution of a timer triggered by TimerSys
Event	Intercepts a Windows event on a control, a group of controls or a WinDev window
Handle	Returns the system "Handle" (HWND) of a WinDev control or window
HandleParentNext	Forces the parent of the next window to open. This function allows a WinDev window to be the child of another application, or to be the child of a non-WinDev window.
Multitask	Enables you to define a time-out, to give control back to Windows or to give control back to Windows and to the WLanguage
PostMessage	Sends a Windows message to a control or to a window
SendMessage	Sends a Windows message to a control or to a window
Timer	Periodically and automatically calls a WLanguage procedure. Called in the procedure, Timer is used to identify the timer that started the procedure.
TimerSys	Periodically and automatically calls a WLanguage procedure. Called in the procedure, TimerSys is used to identify the timer that started the procedure.
Wait	Temporarily stops the execution of a program

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.16 Java functions

WebDev WD

The following functions are used to manage the Java applications:

JavaExecute	Starts a Java application or a Java applet by invoking the "Main" method of a given Java class
JavaExecuteFunction	Runs a specific static function of a Java class
JavaLoad	Specifies the location of the classes required to run the Java application or the Java applet

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.17 Miscellaneous WinDev/WebDev functions

The miscellaneous functions are as follows:

AppliControl	Enables (or disables) the remote check for the current multi-user WinDev application
ASPDisplay	Calls an external ASP script and returns the result page in the current browser window
ASPEXecute	Calls an external .asp script and returns the result in a string
AutomationEvent	Branches a procedure to the event of an automation object
AutomationParameter	Configures the management of accesses to the ActiveX objects and to the Automation objects
Beep	Emits a beep
BuildAutomationVariant	Builds an Automation variant parameter
CapsLockVerify	Checks whether the CapsLock key is pressed.
CertificateClientInfo	Returns information about the certificate used by the client computer.
ChangeSkinTemplate	Dynamically changes the skin template associated with a window (specific window, windows of WinDev components, ...)
Compile	Compiles a procedure dynamically
ComponentInfo	Returns information about a component
ComponentLoad	Loads the specified component in memory
ComponentLoadAll	Recursively loads in memory all the components used in the project.
ConfigureAAF	Allows you to specify the behavior of some automatic features of the application (AAF).
ConfigureSpellCheck	Configures the spelling checker of OpenOffice for the edit controls and for the table columns found in the application. You have the ability to specify the language in which the spell check must be performed.
ConnectionCount	Returns the number of instances of the WebDev site currently run on the server
CookieRead	Retrieves the value of a cookie saved on the computer of the Web user
CookieWrite	Writes a cookie onto the computer of the Web user
DataType	Identifies the type of a WLanguage variable
diffApply	Applies a difference buffer to a specific buffer

diffCreate	Compares two buffers and creates a buffer containing the differences between these two buffers
Dimension	Calculates the size of a variable or resizes a dynamic array
DisableAAF	Disables an automatic feature of WinDev on a control or on a window
Disables AAF	Disables an Automatic Application Feature (AAF) for a control, for a window or for the current application
DrawingStyle	Allows you to modify parameters for the display mode of windows
EndAutomatedProcedure	Stops the current or forthcoming execution of an automatic procedure (defined by the properties of the procedure in the code editor)
EndProgram	Ends the execution of the current program
ErrorInfo	Returns information about the last error that occurred in a component function
EvaluateExpression	Evaluates the value of an expression built in a character string.
Execute	Starts the execution of a process by programming
ExecuteAAF	Runs an automatic application feature (AAF) on a specific control
ExecuteAncestor	When the code is overloaded, used to run the process of the template corresponding to the process of the current control.
ExecuteCode	Runs the WLanguage code found in a character string.
ExecuteDelayedProcedure	Runs a delayed procedure (in the WebDev application server).
ExecutePresetAction	Runs a preset action of WinDev
ExecuteProcess	Starts the execution of a process by programming
ExecutionMode	Returns or modifies the behavior of the application (site) when errors are generated
fDataDir	Returns the full path of the directory for the HFSQL data files
FileDisplay	Returns a specific file to the client browser
FolderData	FolderData is kept for backward compatibility
FolderWeb	Returns the path of the directory containing the images, the Javascript files, ...
fWebDir	Returns the physical name of the directory containing the images, the Javascript files and the Java files of the WebDev site
GeneratePassword	Automatically generates a password with specific characteristics (size, characters used, ...).
GetColor	Allows you to get a set of harmonious colors without having to use a table of colors
GetGUID	Calculates a globally unique identifier (named "GUID")
GetIdentifier	Calculates a unique identifier (integer) for a given executable
Hasp	Interrogates a HASP electronic key
HelpFile	Returns or modifies the name of the help file used by the context-sensitive help of a window.
In64bitMode	Indicates whether the code is run in 64-bit mode or not.
InAJAXMode	Identifies whether the code is run from an AJAX process or from a procedure called by AJAXExecute or AJAXExecuteAsynchronous
InAndroidEmulatorMode	Indicates whether the code is run in the Android emulator or not
InAndroidSimulatorMode	Indicates whether the code is run in the Android simulator
InAutomaticTestMode	Enables you to find out whether an automatic test is currently run or not
InAWPMode	Indicates whether the code is run from an AWP page.
InComponentMode	Indicates whether the code is run from a WinDev application or from a component
InDelayedProcedureMode	Indicates whether the code is run from a delayed procedure or from a scheduled task on the WebDev application server
InFactoryMode	Indicates whether the code is run by an action plan of the software factory
IniOSEmulatorMode	Identifies whether the code is running in the iOS emulator (iPhone, iPad) or not.
IniOSMode	Identifies whether the code currently run is in iOS mode (iPhone, iPad) or not.

InIOSSimulatorMode	Identifies whether the code is run in the iOS simulator (iPhone, iPad) or not.
INIRead	Reads the content of an INI file
InitRandom	Initializes the generator of random numbers
INIWrite	Writes a specific value into an INI file
InJavaMode	Indicates whether the code is run in Java
InLinuxMode	Indicates whether the code is run in Linux mode or not
InPHPMode	Indicates whether the code is run in PHP
InPHPSimulatorMode	Indicates whether the code is run in PHP simulator mode
InPocketMode	Indicates whether the code is run from a WinDev Mobile application or from a standard WinDev application
InPocketSimulator-Mode	Indicates whether the code is run in the Pocket simulator
InReportsAndQueries-Mode	Indicates whether the code is run during the test of a report or query in Reports and Queries
InServiceMode	Indicates whether the code is run in a service
InSimulatorMode	Used to find out whether the application is run in "simulator test mode" from WinDev Mobile.
InStoredProcedure-Mode	Indicates whether the code is run on a HFSQL server
InTestMode	Identifies the startup mode of the application or site
InTSEMode	Indicates whether the code is run from a computer in TSE mode or from a remote desktop.
InUMCMode	Indicates whether the code is run from a user macro-code.
InVGAMode	Indicates whether the application is run in VGA mode on a mobile device
InWebMode	Identifies the startup mode of the current code
InWebserviceMode	Allows you to find out whether the execution is in progress in the AWWWS engine.
InWidgetMode	Identifies whether the code is run in Android Widget mode
InWindowsMobile-Mode	Indicates whether the code is run from a Windows Mobile application.
InWindowsMode	Indicates whether the code is run in Windows mode or not
InWindowsPhoneE-mulatorMode	Indicates whether the test is run via the Windows Phone emulator
InWindowsPhone-Mode	Indicates whether the code is run in Windows Phone mode.
InWindowsPhoneSimu-latorMode	Indicates whether the code is run in the Windows Phone simulator
InWindowsStoreAp-pMode	Identifies whether the code is run in Windows Store apps mode
IsNumeric	Allows you to find out whether a variable or a control is a numeric or a string that can be converted into numeric
JSEndEvent	Removes the association between a WLanguage browser function and an event (implemented by JSEvent.
JSEvent	Associates a browser procedure with an event on an object in browser code
JSInfoEvent	Used to handle the JavaScript properties of the browser event that triggered the execution of the code
JSInterruptEvent	Interrupts the process of the current event
JSMethod	Allows you to run a Javascript method on an element found in the current page
JSONExecute	Calls a server URL of the same domain that returns data in JSON format
JSONExecuteExternal	Calls an external server URL that returns data in JSON format
JSProperty	Allows you to handle specific features on the objects found in the current page

LargeFontFactor	Returns the current enlargement ratio of a WinDev window (only if Windows is in large font mode)
LoadProcedure	LoadProcedure is kept for backward compatibility.
LoadWDL	Loads a library of objects in memory (.WDL)
MouseXPos	Returns the horizontal position of the mouse cursor
MouseYPos	Returns the vertical position of the mouse cursor
NumHelp	Returns the number of the help context associated with the specified control
OrderLine	Identifies and retrieves the different elements of the command line passed in parameter to the current program
PHPDisplay	Calls an external PHP script and returns the result page in the current browser window
PHPExecute	Calls an external .php script and returns the result in a string
ProfilerEnd	Stops "collecting data" for the performance profiler
ProfilerStart	Starts "collecting data" for the performance profiler
Random	Returns a random number
RunReportsAndQueries	Starts Reports and Queries
SComputeCrc16	Checks whether a character string has been transmitted properly in case of transmission with risk of information loss
ScriptDisplay	Calls an external script (.php or .asp for example) and returns the result page in the current browser window
ScriptExecute	Calls an external script (.asp or .php for example) and returns the result in a string (an http query is performed, allowing you to use either a POST method or a GET method)
SelectColor	Opens the standard window for color selection
Sound	Plays a sound in WAV format
SSLActive	Allows you to enable or disable the secure SSL mode.
StringDisplay	Displays a character string (or a buffer) in the browser
Trace	Displays the requested information (content of a control for example) in a window opened in parallel of the current window
TraceEnd	Closes the trace window if this window was opened
TraceStart	Opens the trace window
TrialVersion	Used to find out whether the current execution of the application is in "trial version" mode
TypeVar	Identifies the type of an expression, a variable (during a call to a procedure for example) or a control
UnloadProcedure	UnloadProcedure is kept for backward compatibility
UnloadWDL	Frees the library that was loaded in memory by LoadWDL
UploadCopyFile	Saves on the server a file "upload" by a Web user (which means sent by a Web user to the server via an "Upload" edit control)
UploadFileName	Returns the name of a file "upload" by a Web user (which means sent by a Web user to the server via an "Upload" edit control)
VariableReset	Resets the variable to its initial value
VariantConvert	Converts the value stored in a Variant variable
WHelp	Displays a file or a help page in HLP or CHM format
WinDevVersion	Returns the version number of the WD15OWM DLL of WinDev

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.18 Functions for managing the executables

The following functions are used to handle the executables and to get information about the executables found on the current computer:

AutoRunAdd	Allows you to automatically start a WinDev Mobile application when a specific event occurs. This startup is defined from a WinDev Mobile application. Also enables you to automatically start a WinDev application when a specific event occurs. This startup is defined from a WinDev application.
AutoRunDelete	Allows you to cancel the automatic startup of a WinDev or WinDev Mobile application. This automatic startup was defined by the AutoRunAdd function.
ExeGetPID	Returns information about the current process.
ExeInfo	Retrieves the specified information about the version of an executable or DLL
ExeListDLL	Returns the list of libraries (".DLL" files) used by an application currently run
ExeListProcess	Returns the list of applications currently run. For each application found, you can find out its identifier, the identifier of the parent process (the one that started the application), the name and extension of its executable, the name and full path of its executable, its current memory consumption and its maximum memory consumption.
ExePriority	Returns or modifies the priority of an application currently run
ExeRun	Starts the execution of a program (an executable for example) from the current application or site
ExeRunning	Allows you to find out whether an application is already started
ExeTerminate	Terminates the execution of a 32-bit application: all the instances of the executable are "killed"
GetIdentifier	Calculates a unique identifier (integer) for an executable or for a session.
ListDLL	Returns the list of libraries (".DLL" files) used by the current WinDev application, WinDev Mobile application or WebDev site. Only the libraries loaded in memory are listed.
ShellExecute	Directly opens a document in the associated application or site

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.19 Hasp functions

The dongles of the Hasp family (Hasp3, Hasp4, NetHasp, memoHasp and Hasp HL) are used to protect the applications developed with WinDev or WebDev.

The following functions are used to manage the Hasp functions:

Hasp	Interrogates an electronic HASP key (HASP-3, HASP-4, MemoHasp or NetHasp).
HaspHLCrypt	Encrypts a character string by using the algorithms of the specified Hasp key.
HaspHLInfo	Returns information about the electronic Hasp key.
HaspHLLogin	Connects the application with a Hasp key.
HaspHLLogout	Closes the connection of the application to a Hasp key.
HaspHLRead	Reads data (character string or buffer) in the internal memory of the Hasp key.
HaspHLUncrypt	Decrypts a character string by using the algorithms of the Hasp key.
HaspHLWrite	Writes data (character string or buffer) into the internal memory of the Hasp key.

See the online help for more details.

9.20 Functions for DDE management

WebDev WD

The following functions are used to manage the DDE (Dynamic Data Exchange) :

DDEConnect	Establishes a DDE connection between the current program and the recipient according to a given subject
DDEDisconnect	Deletes a connection between the current program and a recipient
DDEError	Returns the status report about the execution of a function used to manage the DDE dialog
DDEEvent	Associates a WLanguage procedure with a DDE event
DDEExecute	Sends a command to run
DDEItem	Identifies the item affected by a DDE event
DDELink	Creates a hot link with a data
DDERecipient	Identifies the recipient of a DDE connection
DDERetrieve	Retrieves a data sent by a program (the recipient of the connection for the specified object)
DDESend	Sends a data to the program connected via DDE
DDEStart	Runs a program from the current application or site
DDEString	Returns the information retrieved by DDERetrieve
DDETopic	Identifies the subject of the conversation associated with a DDE connection
DDEUnLink	Interrupts a link between an item and a data
DDEUpdate	Modifies a linked data
DDEWarmLink	Creates a warm link with a data

See the online help for more details.

9.21 Functions for managing the applications with "live update"

The following functions are used to manage the applications with "Live update":

AppliActivateVersion	Enables one of the application versions available in the history of versions on the reference setup server.
AppliChangeParameter	Changes an information in the mechanism for automatic update of the specified application.
AppliControl	Enables (or disables) the remote check for the current multi-user WinDev application.
AppliDeleteVersion	Deletes a version from the history of versions found on the reference setup server.
AppliInstallUPD	Triggers the automatic update of the current application.
AppliInstallVersion	Installs a specific version of the application.
AppliListVersion	Lists the versions available on the reference setup server. These versions are available in the history of the reference setup.
AppliParameter	Returns information about the mechanism for automatic update of the specified application.
AppliUPDAvailable	Allows you to find out whether an update is available for the specified application.
AppliVersionInfo	Returns information about a version available on the setup server.

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.22 Functions for managing the networks

The following functions are used to manage the networks:

NetworkConnect	Associates a drive with a shared network directory
NetworkDirName	Identifies the directory associated with a network drive
NetworkDisconnect	Disconnects a network drive
NetworkDomain-Name	Returns the domain name associated with the current computer
NetworkUser	Identifies the name of the connected user

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.23 Functions for managing the SNMP protocol

The following functions are used to manage the SNMP protocol:

SNMPCloseSession	Closes an SNMP session.
SNMPGet	Reads one or more values of an agent with which an SNMP session was started.
SNMPGetNext	Reads the value found after the last value retrieved for an SNMP agent.
SNMPGetTable	Reads a table of SNMP values.
SNMPLoadMIB	Loads a MIB file in memory and analyzes it.
SNMPOIDAccess	Returns the authorized access modes of a specific OID.
SNMPOIDDescription	Returns the description of an OID.
SNMPOIDStatus	Returns the status of an OID.
SNMPOIDToString	Converts an OID from its numeric representation to its text representation.
SNMPOIDType	Returns the type of data found in an OID.
SNMPSet	Writes one or more values onto an SNMP agent for which an SNMP session was started.
SNMPStartSession	Starts an SNMP session.
SNMPStringToOID	Converts an OID from its text representation to its numeric representation.
SNMPTrapDisable	Disables the interception of an SNMP trap.
SNMPTrapEnable	Enables the interception of an SNMP "trap" sent by an agent.

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.24 Functions for managing the projects

These functions are used to easily manage the project elements:

ComponentInfo	Returns information about a component.
ComponentList	Returns the list of components (".WDK" file) used by the current application
ComponentLoad	Loads the specified component in memory. This component will be loaded in memory until the end of the application.
ComponentLocate	Specifies the access path to a component found in the project
EnumElement	Allows you to list the project elements (windows, pages, reports, queries, ...)
EnumSubElement	Allows you to list the sub-elements of a project element (windows, reports, queries, ...). This function can be run even if the element is not opened

InComponentMode	Identifies whether the code is run: <ul style="list-style-type: none"> • from a WinDev application, a WebDev site or a PHP page, • from a component
ListDLL	Returns the list of libraries (".DLL" files) used by the current application or installed on the current computer
ProjectInfo	Returns a specific information about the project currently run (project name, associated registry key, ...)
WindowCount	Calculates the number of windows currently opened in the current application

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

9.25 Functions for managing the scheduler

WD

The following functions are used to manage the Windows scheduler:

SchedulerAddTask	Creates a task in Windows scheduler
SchedulerAddTrigger	Creates a schedule for a scheduled task
SchedulerDeleteTask	Deletes a scheduled task
SchedulerDeleteTrigger	Deletes a schedule from a scheduled task
SchedulerModifyTask	Modifies the parameters of a scheduled task
SchedulerModifyTrigger	Modifies the parameters for the schedule of a scheduled task, which means the parameters that will define the execution of a task
SchedulerOpenTaskProperties	Opens the property page of a scheduled task
SchedulerReset	Fills the <code>ScheduledTask</code> and <code>TriggerScheduledTask</code> structures with the default values
SchedulerTaskList	Retrieves the list of scheduled tasks
SchedulerTaskProperties	Reads the properties of a scheduled task and updates the <code>ScheduledTask</code> structure
SchedulerTaskStatus	Retrieves or modifies the status of a task created in the Windows scheduler
SchedulerTriggerProperties	Reads the properties for a schedule of a scheduled task and updates the <code>TriggerScheduledTask</code> structure

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

10. ENABLING AN APPLICATION

10.1 Overview

When distributing an application, it may be useful to propose a full version and a limited version:

- The limited version is supplied by default.

- The full version requires an activation key.

WinDev, WebDev and WinDev Mobile allow you to easily manage this feature.

10.2 How do I proceed?

10.2.1 Principle

An initial key (a serial number for example) is generated upon request by the application on the client computer. This key is unique: it depends on parameters specific to the current computer at the time of the request.

The client can perform an activation request (by Internet for example) toward the application provider by communicating this initial key.

The provider will generate the activation key from the initial key and communicate it to his client.

Then, the client will be able to enter the initial key and the corresponding activation key to activate his application.

10.2.2 Implementation in the client application

The following steps must be performed by the application used by the client:

1. Generation and distribution of the initial key:

- The generation of the initial key is performed by `KeyGenerateInitialKey`.
- The communication of the initial key to the provider of the application can be performed via an

automatic email or via a Web site.

2. Retrieving the activation key and enabling the application:

- Retrieving the activation key can be performed by the application or by the input of the key by the end user.
- The check regarding the conformity between the activation key and the initial key is performed by `KeyCompareKey`. If it is successful, you will be able to activate some features of the application.

10.2.3 Implementation in the application of the provider

The application provider must own an application used to:

- generate the activation key from an initial key (`KeyCalcActivationKey`).
- transmit the activation key to the client application. You have an ability to send an email for example.

These operations can be proposed by a WebDev site or by a Webservice allowing an immediate online activation.

10.3 Functions for managing the activation keys

The following functions are used to manage the activation keys:

<code>KeyCalcActivationKey</code>	Calculates the activation key of the application from the initial key.
<code>KeyCompareKey</code>	Compares an initial key with an activation key
<code>KeyGenerateInitialKey</code>	Generates the initial key (unique license number) for the activation system of the application
<code>KeyGetIdentifier</code>	Retrieves the identifier that was used to generate an initial key

See the online help for more details.

11. HANDLING YOUR XLS FILES

WebDev WD

11.1 Overview

WinDev et WebDev propose two methods for handling the Excel files:

- Method 1: Dynamically handling the XLS and XLSX sheets.
- This method uses different types of variables as well as WLanguage functions. This method is

used to manage the Excel files (XLS or XLSX files) in read and write modes.

Method 2 (for backward compatibility): Using the WLanguage functions to manage the XLS functions.

This method can only be used to manage the XLS files in read-only.

11.2 Method 1: Dynamically handling the XLS and XLSX files

To handle the XLS and XLSX files, the WLanguage proposes:

- different types of variables. These types of variables are used to handle an XLS document, an XLS row or column, an XLS cell.
- several WLanguage functions that handle the different types of data.

To handle the XLS and XLSX files:

1. Declare an `xlsDocument` variable. This type of variable is used to describe and modify an Excel document.
2. Open (if necessary) the XLS file (`xlsOpen`) and associate it with the `xlsDocument` variable.
3. The different elements found in the Excel document can be handled:

- by the WLanguage functions.
- by the properties of the `xlsDocument` type.

Example:

```
IdXLSFile is xlsDocument
MyFile is string
MyFile = fExeDir + ...
"\Follow up1.xls"
IDXLSFile = xlsOpen(...
    MyFile, xlsWrite)
IF ErrorOccurred = False THEN
    Azz is int
    Azz = xlsCurrentWorksheet(IdXls-
File)
Info("Current sheet in"+ ...
```

```
"The Excel file:" + Azz)
// Change the worksheet
IF xlsCurrentWorksheet(...
    xlsFileID,2) THEN
    Info("The current sheet"+ ...
"has been modified.")
// Read a cell for test
Info(xlsData(xlsFileID,11,2))
ELSE
    Error("The selected file"+...
"does not have a second"+...
"worksheet")
END
ELSE
    Error("Caution! the file "+...
"is already opened on another"+...
"computer! ")
END
```

Note

- If the Excel file is opened in read/write mode (`xlsWrite` constant), the file is opened and locked until it is closed. To save the modifications performed, use `xlsSave`.
- The XLS functions do not require "Microsoft Excel" to be installed on the user computers.
- The XLS files and the files specific to Office 2007 (.xlsx files) are supported.

11.3 Method 2: Reading the Excel files (kept for backward compatibility)

WinDev and WebDev propose several functions allowing you to manage your ".XLS" files from your WinDev applications and from your WebDev sites. These functions allow you to retrieve the data entered in Excel.

To handle an XLS document:

1. Use xlsOpen. This function returns the identifier of the XLS file used.
2. Use the XLS functions to retrieve the requested information about the Excel file and its data.

Notes:

- The XLS functions do not require "Microsoft Excel" to be installed on the user computers.

- Only the XLS files are supported. The files specific to Office 2007 (.xlsx files) are not supported.
- Two operating modes are available for the XLS functions:
 - Operating mode compatible with the earlier versions of WinDev and WebDev: in this mode, only the first sheet of the workbook is accessible.
 - New multi-sheet operating mode: in this mode, you can select the workbook sheet where the operations will be performed.

The function used to modify the mode for handling the XLS functions is named xlsOpen.

11.4 WLanguage functions

These functions allow you to:

- handle the ".XLS" files: opening and closing.
- get information about your ".XLS" files: cell data, number of rows, number of columns, column type, column title (see diagram).
- find out the Excel version used.
- identify the errors that occurred during the different operations performed on an ".XLS" file.

	A	B	C	D
1	Name	First name	Address	City
2	Lautier	Guillem	Jordan RD	Chelsea
3	Bompard	Romain	Accacia Avenue	Montpellier
4	Hannotte	Carla	Fleet St	Paris
5	Castaing	Patrick	Bond St	London
6	Bertrand	Arnaud	Western Avenue	Liverpool
7				
8				
9				
10				
11				
12				

Annotations in the image:

- `xlsNbRow` (6 in this example) points to row 6.
- `xlsColumnTitle` ("City" in this example) points to column D.
- `xlsData` ("London" in this example) points to cell C5.
- `xlsColumnType` (string in this example) points to column A.
- `xlsNbColumn` (4 in this example) points to the number of columns.

11.5 XLS functions

The following functions are used to manage the Excel files:

xlsAddWorksheet	Adds or inserts a new worksheet into an Excel document.
xlsClose	Closes an XLS file
xlsColumnName	Retrieves the title of a column found in an XLS file
xlsColumnType	Returns the type of the data entered in a column of an XLS file
xlsCurrentWorksheet	Used to find out and modify the current worksheet in an XLS file
xlsData	Retrieves the data from a cell of the XLS file
xlsDeleteWorksheet	Deletes a worksheet from an Excel document.
xlsMsgError	Returns the caption of the last error caused by an XLS function
xlsNbColumn	Returns the number of columns found in an XLS file
xlsNbRow	Returns the number of rows found in an XLS file
xlsNbWorksheet	Returns the number of worksheets found in an XLS file
xlsOpen	Opens an XLS file
xlsSave	Saves an Excel document.
xlsVersion	Returns the Excel version used

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

Note: *TableToExcel* is used to create an XLS file from the data found in a table.

12. THE ARCHIVES

12.1 Overview

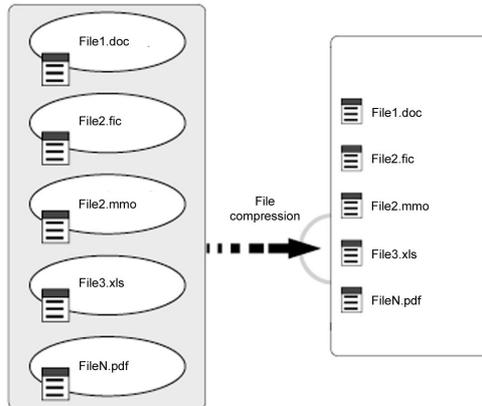
Several WLanguage functions are used to compress and group your files into archives.

An archive is a physical file:

- of ".WDZ" type for a WinDev archive.

- of ".ZIP" type for a WinZip or PKZip archive.

This file groups several automatically compressed files of different types (".pdf", ".fic", ".mmo", etc.).



For example, the archives allow you to:

- save your files on a regular basis while reducing the storage space,
- simplify the transmission of your files over the Internet/Intranet,
- create self-extracting archives,
- etc.

An optimized compression allows your files to take less disk space and to be easily distributed on diffe-

rent media (diskettes, CD, Internet, etc.).

There are two types of archive:

- single-part archive: the archive is made of one file.
- multi-part archive: the archive is made of several files (or sub-archives).

Note: WinDev also supports the standard zip formats (WinZip and PKZip).

12.2 Handling the archives

More than 20 WLanguage functions allow you to manage your archives. These functions allow you to:

- handle the archives (creation, opening, closing),
- handle the files found in the archive (add and compress files (single file or all the files in a directory), extract and uncompress a file, delete files or delete an archive),
- get various information about the archives and about the compressed files (size before and after compression, etc.),
- split and/or merge archive volumes,

- create self-extracting executables.

Example

To save the database of your customer, your program creates a backup archive. The files to save are compressed into the archive.

To store this archive on diskettes: this archive is split into several parts. This enables you to easily transmit this archive!

To simplify the use of the backup files: create a self-extracting archive. All you have to do is run the archive to automatically decompress all the files.

12.3 The single-part and multi-part archives

12.3.1 Overview

Two types of archives are available:

- single-part archive,
- multi-part archive.

Single-part archive

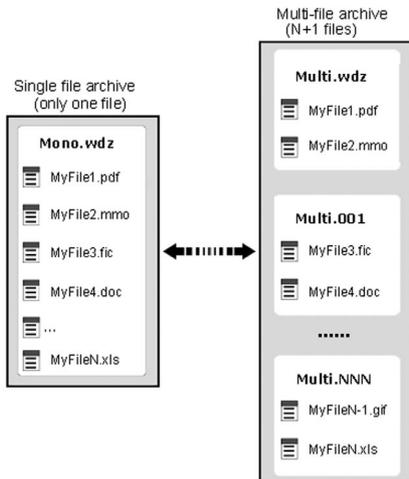
A single-part archive contains a single ".WDZ" or ".ZIP" file: an archive file contains all the compressed files.

The archive occupies the necessary space to contain all the compressed files.

Example: Your program or your site must save several large files. Your program or your site can create an archive containing all the files to save. This archive will increase the available disk space.

Multi-part archive

A multi-file archive contains several files (or sub-archives) of identical size. The size of each sub-archive is defined when creating the multi-file archive.



The first sub-archive is a ".WDZ" or ".ZIP" file. The following sub-archives are files numbered from 1 to N via their extension.

For the archives in WinDev format (WDZ), the sub-archives have the following extensions: ".001", ".002", ..., ".NNN".

For the archives in ZIP format, the sub-archives have the following extensions: ".Z01", ".Z02", ..., ".ZNN".

12.3.2 Principle

To create a multi-part archive:

1. Create an archive (**zipCreate**).
2. Add files to this archive. The added files are automatically compressed (**zipAddFile**, **zipAddDirectory**).
3. Split this archive into several sub-archives (**zipSplit**). The sub-archives have the same size. You define the size of the sub-archives.

You can store the sub-archives on different media (diskettes, CD, etc.).

The sub-archives can be merged (**zipMerge**): the archive becomes a single-part archive.

Note: All these operations can also be performed via **WDZip** (see the online help for more details).

Caution: Some archiving functions can be only used with single-part archives (add files, delete files, create a self-extracting executable, etc.). To perform these operations on a multi-part archive, the sub-archives must be merged (**zipMerge**) to create a single-part archive.

From a multi-part archive, you only have the ability to :

- Merge the different parts (**zipMerge**),
- Get information about the archive and about the files found in the archive (**zipExtractPath**, **zipInfoFile**, **zipListFile**, etc.),
- Extract the files (**zipExtractFile**, **zipExtractAll**).

12.3.3 Examples

- Your WinDev program must store a large amount of data on diskettes. Your program can create an archive containing the requested data. Once created, this archive can be divided into several parts. These volumes have the size of a diskette (1.44 MB). All you have to do is copy these different parts on the diskettes.
- Your WebDev site must propose the download of a large file. For the Web users with a slow connection (phone line for example), there is often an option used to download a file in several parts. Your site can create an archive containing all the requested data. Once created, this archive can be divided into several parts. These volumes can have a suitable size for download. Now, all you have to do is provide a link for each part of the archive to the Web user.

12.4 Archiving functions

The following functions are used to manage the archives:

zipAddDirectory	Adds all the files found in a directory and in its sub-directories into an archive
zipAddFile	Adds a file into an archive and compresses it
zipAddFileList	Adds a list of files into an archive and compresses it
zipChangePath	Modifies the path of a file found in the archive
zipClose	Closes an archive
zipCompressionLevel	Changes the compression level used when creating an archive in ZIP format
zipCreate	Creates a new archive
zipCreateExe	Creates a self-extracting executable from an archive
zipCurrentFile	Returns the name of the file currently processed by the functions for adding and extracting files
zipDeleteAll	Deletes all the files from an archive
zipDeleteFile	Deletes a file from an archive
zipDeleteFileList	Deletes a list of files from an archive
zipExist	Allows you to find out whether an archive exists. An archive exists if it was opened by zipOpen or if it was created by zipCreate.
zipExtractAll	Extracts all the files from an archive and decompresses them
zipExtractFile	Extracts a file from an archive and decompresses it
zipExtractFileList	Extracts and automatically decompresses a list of files found in an archive to a physical location.
zipExtractPath	Returns the initial path of a file found in the archive
zipFileSize	Returns the size of a file in an archive (before and after compression)
zipFindFile	Finds a file in an archive according to a given path
zipInfoFile	Returns various information about a file or about an archive (stored path, size before and after compression, etc.)
zipIsMulti	Returns the type of the archive: single-part archive or multi-part archive
zipListFile	Returns the list of files found in the archive
zipMerge	Merges the different parts of an archive to create a single-part archive
zipMsgError	Returns the message associated with an error number for an archiving process
zipNbFile	Returns the number of files found in an archive
zipNbPart	Returns the number of parts found in an archive
zipNbPartNeeded	Returns the number of parts of a given size required to contain the archive
zipOpen	Opens an existing archive
zipOpenCAB	Opens an existing CAB archive
zipOpenRAR	Opens an existing RAR archive
zipPassword	Defines the password used to add files into the specified archive and to extract files from the specified archive
zipSize	Returns the size of all the files found in an archive (before and after compression)
zipSplit	Splits an archive into several parts

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

13. BURNING A CD OR A DVD

WebDev WD

13.1 Overview

Several WLanguage functions are used to burn the CDs or DVDs. You can create:

- data CDs/DVDs.
- audio CDs.

Note: You can only burn files onto a CD or a DVD. You cannot copy a CD or DVD directly.

In WinDev, the burn functions are available for Windows XP and later.

In WebDev, the burn operation will be performed on the Web server. This server must be running Windows XP or later.

13.2 Burning a CD/DVD

To burn a CD:

1. Specify (if necessary) the burner to use:

BurnerPath	Returns the path of the current burner
BurnerStatus	Returns the status of the current burner
BurnerList	Returns the list of burners installed on the current computer
BurnerProperty	Allows you to find out and modify the properties of the current burner
BurnerSelect	Allows you to select the default burner

2. Open (if necessary) the door of the current burner (**BurnerEject**).

3. Specify (if necessary) the type of the CD/DVD to create:

BurnerMediaInfo	Retrieves the characteristics of the CD/DVD found in the current burner
BurnerMediaType	Allows you to find out or modify the name of the CD/DVD to burn

4. Erase (if necessary) the files already found on the CD/DVD (**BurnerErase**). A WLanguage procedure ("callback") is regularly called during this erase process. This procedure is used to manage the different events that occur.

5. Specify (if necessary) the name that must be given to the CD/DVD to burn (**BurnerVolumeName**).

6. Select the files to burn on the CD/DVD:

BurnerAddFile	Adds a file to the compilation
BurnerAddDirectory	Adds all the files found in a directory to the compilation

7. Retrieve (if necessary) the characteristics of the compilation (**BurnerCompilationInfo**).

8. Burn the selected files on the CD/DVD (**BurnerSave**). A WLanguage procedure ("callback") is regularly called during this burn process. This procedure is used to manage the different events that occur.

9. Cancel (if necessary) the current burn operation (**BurnerCancel**).

13.3 Burn functions

The following functions are used to manage the archives:

BurnerAddDirectory	Adds all the files found in a directory to the compilation
BurnerAddFile	Adds a file to the compilation
BurnerCancel	Cancels the current burn process
BurnerCompilationInfo	Retrieves the characteristics of the current compilation
BurnerEject	Opens or closes the door of the current burner
BurnerErase	Erases the files found on a rewritable CD
BurnerList	Returns the list of burners installed on the current computer
BurnerMediaFound	Allows you to find out whether the burner is empty or not.
BurnerMediaInfo	Retrieves the characteristics of the CD found in the current burner

BurnerMediaType	Allows you to find out and modify the format of the CD to burn
BurnerPath	Returns the path of the current burner
BurnerProperty	Allows you to find out and modify the properties of the current burner
BurnerSave	Burns the files found in the compilation to the CD
BurnerSelect	Allows you to select the default burner
BurnerStatus	Returns the status of the current burner
BurnerVolumeName	Allows you to find out and modify the name of the CD to burn

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

14. FUNCTIONS FOR ACCESSING A POCKET PC

WD

These functions are used to access the Pocket PCs from a standard WinDev application.

ceConnect	Connects the current computer to a Pocket PC
ceConnectionStatus	Allows you to find out the status of the connection between the current computer and a Pocket PC
ceCopyFile	Copies a file found on the current computer to the connected Pocket PC; found on the connected Pocket PC to the current computer; found on the connected Pocket PC to another directory on the Pocket PC
ceCreateShortcut	Creates a shortcut on the Pocket PC connected to the current computer
ceDeleteFile	Deletes a file from the Pocket PC connected to the current computer
ceDeleteShortcut	Deletes a shortcut that was created by ceCreateShortcut
ceDir	Finds a file or a directory on the Pocket PC connected to the current computer
ceDisconnect	Closes the connection between the current computer and the Pocket PC
ceFileDate	Returns or modifies the different dates associated with a file (creation, modification or access)
ceFileExist	Checks the existence of a file
ceFileSize	Returns the size (in bytes) of a file found on the Pocket PC connected to the current computer
ceFileTime	Returns or modifies the different times associated with a file (creation, modification or access)
ceListFile	Lists the files found in a directory (and in its sub-directories) and returns the number of listed files
ceMachineName	Returns the name of the Pocket PC connected to the current computer
ceMakeDir	Creates a directory on the Pocket PC connected to the current computer
ceOEMInfo	Returns the OEM information of the Pocket PC connected to the current computer
cePlatform	Returns the name of the platform for the Pocket PC connected to the current computer
cePowerStatus	Returns information about the main or spare battery of the Pocket PC
ceProcessorType	Returns the type of processor for the Pocket PC connected to the current computer
ceRegistryCreate-Key	Creates a key in the Pocket PC registry
ceRegistryDeleteKey	Deletes a sub-key from the Pocket PC registry
ceRegistryDeleteValue	Deletes a value from the Pocket PC registry
ceRegistryExist	Checks the existence of a key in the Pocket PC registry
ceRegistryFirstSub-Key	Identifies the key found after the specified key in the Pocket PC registry
ceRegistryListValue	Returns the name (and possibly the type) of the values for a key found in the Pocket PC registry
ceRegistryNextKey	Identifies the key found after the specified key in the Pocket PC registry
ceRegistryQueryValue	Reads the value of a register in the Pocket PC registry
ceRegistrySetValue	Writes a value into a register of the Pocket PC registry
ceRegistrySubKey	Identifies the path of the Nth specified sub-key in the Pocket PC registry
ceRemoveDir	Deletes a directory from the Pocket PC connected to the current computer

ceRunExe	Starts the execution of a program (an executable for example) from the current application
ceSysDir	Returns the path of a system directory for the Pocket PC connected to the current computer
ceWindowsVersion	Returns information about the Windows version used on the Pocket PC connected to the current computer
ceWinEnum	Allows you to enumerate the Windows windows currently opened on the Pocket PC
ceWinTitle	Returns the title of the specified Windows window
ceXRes	Returns the horizontal resolution of the screen for the Pocket PC connected to the current computer
ceYRes	Returns the vertical resolution of the screen for the Pocket PC connected to the current computer

See the online help for more details.

15. FUNCTIONS SPECIFIC TO WINDEV MOBILE

15.1 WinDev Mobile and SIM cards

15.1.1 Overview

WinDev Mobile allows you to access the information found in the SIM card of a cell phone via the SIM functions of WLanguage.

The SIM card (Subscriber Identity Module) is the chip found in a cell phone. Required to access the communication network, this smart card identifies the user and stores all the information about the network and about the subscription (phone number, type of contract, ...).

The SIM card also contains a personal directory. The SIM functions of WLanguage are used to manage the information found in this directory.

Note: the personal directory found on the SIM card only contains the name and phone number of the contacts. Only this information can be handled.

15.1.2 Required configuration

To be able to use the SIM functions, the application must be installed:

- on a Pocket PC with phone access (GSM type).
- and/or on a Smartphone.

15.2 Pocket keyboard

15.2.1 Overview

To allow the users of your applications to enter information, the keyboard of the Pocket PC must be used (also called SIP for Software Input Panel).

This keyboard allows you to:

- display a miniature keyboard at the bottom of the screen. The user clicks this keyboard with the stylus to enter information.

15.1.3 Operating mode in GO mode and at run time

In GO mode (simulation on the development computer), a WLanguage error is generated during the call to a function for managing the SIM card.

15.1.4 WLanguage functions

These functions are used to easily manage the information found in the personal directory found on a SIM card:

SIMWrite	Writes or modifies an entry in the directory of the SIM card
SIMRead	Reads an entry in the directory of the SIM card
SIMNbContact	Returns the number of entries found in the directory of the SIM card
SIMDelete	Deletes an entry from the directory of the SIM card

For example:



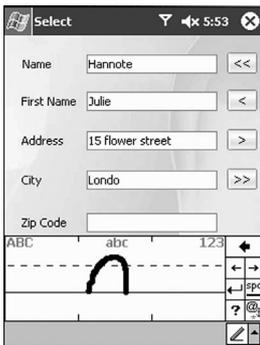
- automatically recognize the different words written on the screen with the stylus (method called "Transcriber").

For example:



- automatically recognize the letters entered in a specific character set (method called "Letter Recognizer").

For example:



- automatically recognize the blocks of words entered in a specific character set (method called "Block Recognizer").

For example:



WinDev Mobile allows you to easily manage this keyboard via the WLanguage functions.

Note: Other types of keyboards may also be available.

15.2.2 WLanguage functions

The functions used to manage the keyboard of a Pocket PC are:

SIPList	Returns the list of keyboard types available on the Pocket PC
SIPMode	Allows you to find out and modify the keyboard currently used
SIPVisible	Allows you to find out whether the current keyboard is enabled and to enable the keyboard



PART 5

**Specific Web
features**

10

DEVELOP 10 TIMES FASTER



PC SOFT

1. UPLOADING FILES

1.1 Overview

The upload consists in saving on the Web server a file accessible from the computer of the Web user. The file found on the computer of the Web user is "uploaded" to the server: it is loaded and saved on the server.

All types of files can be uploaded.

Examples for using the file upload:

- **Address Book site:** when adding a new contact, the user can select a photo for the contact from his computer. In order for the photo to be displayed independently of the Web user, the photo is "uploaded" to the server.
- **Document Management site:** each Web user can make different types of documents available to all: text, ... These documents are selected on the computer of the Web user and "uploaded" to the server in order to be made available to all the Web users.

1.2 Implementing the upload in a WebDev site

1.2.1 Available elements

The following elements are used to manage the file upload:

- **an *UPLOAD edit control*:** In the browser, a "BROWSE" button is automatically added on the right of the Upload edit control. This button allows the Web user to select the file to "upload". A gray button is displayed in the editor. This button cannot be modified (style, caption, ...).

Caution: this edit control is a specific control; the Value properties should not be used on this control. To get the name of the uploaded file, use *UploadFileName*.

- **two server functions of WLanguage:**

UploadCopyFile Saves an "uploaded" file on the server. During this backup, the saved file can be renamed on the server.

UploadFileName Returns the name of an "uploaded" file (initial file name or file name saved on server).

- several WLanguage browser functions are used to handle the Upload control:

UploadCopyCurrentFile Indicates the name of the file currently sent by the upload control.

UploadStart Starts sending the selected files in an upload control.

UploadDelete Deletes a file from the list of files to upload: the file will not be uploaded on the server.

UploadDeleteAll Clears the list of files to upload: no file will be uploaded on the server.

UploadCurrentFileSizeSent Returns the size (in bytes) already sent for the file currently uploaded via an Upload control.

UploadSizeSent Returns the total size (in bytes) of the files already sent by the current upload via an Upload control.

UploadCurrentFileSize Returns the total size (in bytes) of the file currently uploaded via an Upload control.

UploadSize Returns the total size (in bytes) of the file currently uploaded via an Upload control.

1.2.2 Uploading a file in a page

To upload a file in a page:

1. Create an UPLOAD edit control (the type of the control is defined in the "General" tab of the control).

This control allows the Web user to select the file to "upload".

2. Add a button or a link.

This control will be used to upload the file on the server. **This button (or link) must be a "Submit" button (or link)**. When the page is validated via this button, the file is uploaded into a temporary directory of the server. Therefore, this file cannot be viewed by the other Web users

3. In the server click code of this link or button, you can use:

- **UploadFileName** (UploadFileName(<Upload Control>, False) for example) to check that the file was uploaded and to retrieve its name. Indeed, during the upload, the file is renamed with a temporary file name.
- **UploadCopyFile** to copy and rename the file uploaded on the server. This file can be copied into the resource directory (_WEB) or into a directory accessible via an alias.
- **HLinkMemo** to load the image in a memo item of a Hyper File file.

Caution: Directly using the file name on the browser computer (read the value via the **Value** property in server or browser code) in the functions for file management is a BIG mistake. Indeed, this operation operates in development mode (the server and the browser are found on the same computer) but NOT in deployment. In this case, the server and the browser are different computers. The server does not see the browser files. To find out the names of the files to use, call **UploadFileName**.

1.2.3 Displaying the image to upload

To display the image selected in an upload edit control (EDT_UploadControl) in an image control (IMG_ImageControl):

1. In the description window of the "ImageControl" image control ("General" tab), uncheck "Locate the image in the _WEB directory in browser code".

2. In the code of the page, add the optional "OnMouseMove" browser code.

3. Enter the following code lines in this code:

```
IF EDT_UploadControl <> "" THEN
  IF IMG_ImageControl<>
    EDT_UploadControl THEN
    IMG_ImageControl =
EDT_UploadControl
END
END
```

Note: This operation may not operate according to the browser used and to the level of security.

Note: Maximum size of the files to upload:

By default, the size of the files to upload is limited in the WebDev engine.

This size can be limited:

- by the Web server used. See the documentation about the Web server used for more details.
- by the WebDev application server.

The file will not be uploaded if its size exceeds the limit size.

To modify the maximum size of the files to upload in the WebDev application server, the registry must be modified on the server:

1. To modify the size of the uploaded files for all the WebDev 19 applications, select the following key: HKEY_LOCAL_MACHINE\SOFTWARE\PC SOFT\WEBDEV\19.0

2. Add the MAX_UPLOAD string. The value of this string will be the maximum size of the files to upload (in KB).

For a large transfer (several hundreds of MB), we recommend that you use the FTP protocol instead of the upload. You have the ability to perform an FTP transfer from a WebDev page, via a Java applet created with WinDev.

Limits in PHP:

The size of the uploaded files is limited by the "upload_max_filesize" directive in the configuration file of PHP (php.ini file).

The file upload must be allowed by the PHP server. To do so, the "file_uploads" directive must be set to "on" in the configuration file of PHP (php.ini file).

2. DOWNLOADING FILES

2.1 Overview

The download is the operation that consists in saving a file stored on the server onto the computer of the Web user. The file is "downloaded" on the computer of the Web user.

Some examples:

- A site for renting DVDs allows you to download a movie trailer onto your computer.
- Documents can be downloaded into an application for document management.

2.2 Implementing file download in a WebDev site

WebDev proposes several methods to download a file:

- description window of the button or link used to perform the download.
- programming in WLanguage.

2.2.1 Using the description window of controls (button, link,...)

To propose a download:

1. Create a "Link" control, "Button" control, ...
2. Display the description window of this control.
3. Click the "Other actions" button;
4. In the window that is displayed, select "Other actions: enter a link".
5. Specify the full path of the file (directory + name) found on the server. The file must be found in the "_WEB" directory of the site.

At run time, when clicking this control:

- if the file type is recognized, the file will be directly opened in the browser.
- if the file type is not recognized or if it corresponds to an executable, a dialog box allows you to download this file and/or to run it directly.

2.2.2 Programming

To propose file download by programming, use *FileDisplay* in the server click code of the button or link used to download the file.

Example:

```
// Displays in the browser
FileDisplay(CompleteDir(...
    fWebDir() + "NOTES.TXT", ..
    "text/plain")
```

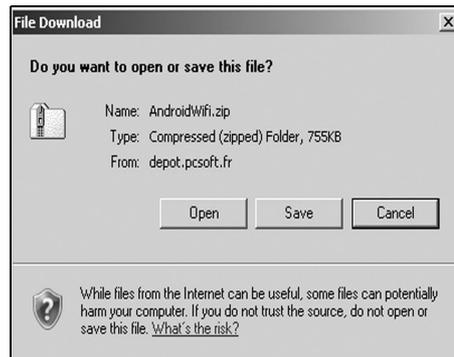
At run time, when clicking this control:

- if the file type is recognized, the file will be

directly opened in the browser.

- if the file type is not recognized or if it corresponds to an executable, a dialog box allows you to download this file and/or to run it directly.

2.2.3 Forcing the file download



To open this dialog box regardless of the type of file to download, enter the following code line:

```
FileDisplay(CompleteDir(...
    fWebDir() + "NOTES.TXT", ...
    "application/unknown")
```

However, the name of the file proposed for download does not correspond to a valid name. To immediately offer the name of the file to download, add this name after the parameters of the *FileDisplay* function:

```
FileDisplay(CompleteDir(...
    fWebDir() + "NOTES.TXT", ...
    "application/unknown", ...
    "Notes.TXT")
```

3. COOKIES

3.1 Overview

A cookie is an easy way to temporarily store an information on the computer of the Web user. This information can be retrieved later.

When a user revisits a site, the site "recognizes" the user via cookies. This enables you to avoid requesting details already provided during a previous visit and to propose custom pages.

Note: A cookie has an expiration date (by default, 30 days after its creation). It is automatically destroyed by the browser of the Web user when its life-time is exceeded.

Example for using cookies

A cookie is used to store on the computer of the Web user personal details such as the user name, the pages visited by the user, the date of the last connection, the backup of options, ...

This information, saved as cookies, will be read during the next connection to the site. Then, the site can propose custom features to the Web user:

- advertising banner related to subjects looked up during the last connection,
- custom home page with the user name and the date of last connection,
- special offers corresponding to the searches performed during the last visit, ...

3.2 What is a cookie made of?

A cookie is a text file stored on the computer of the Web user (in the Internet "cache" of the browser) during a specified duration. The cookie is created by the browser or by the server.

Each cookie contains the following information:

- **Name of the cookie:** used by the site to identify the cookie.
- **Text of the cookie:** information written by the site (page viewed, information supplied by the Web

user, ...).

- **Expiration date,** after which the cookie is not valid anymore.
- **Name of the Internet domain that created the cookie.**

Reminder: the Internet domain is a specific section of the Internet address: `http://machine.domain.com/MyPages/Index.html`.

3.3 WebDev and the management of cookies

3.3.1 Available elements

Two WLanguage functions can be used to manage the cookies in your sites:

- **CookieWrite**
Sends a cookie to the client when the HTML page is displayed in the browser of the Web user.
- **CookieRead**
Retrieves the value of a cookie saved on the computer of the Web user.

These functions can be used in server code and in browser code.

3.3.2 Writing a cookie on the computer of the Web user

To write a cookie on the computer of the Web user:

- **In server code:**
Use **CookieWrite** and specify the name of the cookie, its content and its lifetime.

The cookie will be created on the computer of the Web user when the next page is displayed.

Note: a cookie created in server code can be read by **CookieRead**:

- immediately if **CookieRead** is used in a browser code.

- during the next user connection if **CookieRead** is used in a server code.
- **In browser code:**
Use **CookieWrite** and specify the name of the cookie, its content and its lifetime.
The cookie is immediately created.

3.3.3 Reading a cookie on the computer of the Web user (server code and browser code)

To read a cookie on the computer of the Web user,

use **CookieRead** and specify the name of the cookie.

Notes: During the connection to the Web site, the server automatically reads all the cookies associated with the current domain. **CookieRead** used:

- in server code, reads in memory and retrieves the content of the specified cookie.
- in browser code, directly reads the content of the cookie on the computer of the Web user.

3.4 Checking the management of cookies in a WebDev site

The management of cookies in your site (on the development computer) can be checked:

- from the test page of the WebDev administrator.

- with a simple test via a "GO" of the project from the page editor of WebDev.

4. VALIDITY OF THE SITE PAGES

4.1 Overview

When creating a page, a validity period can be defined for this page.

For example, if a page is related to a specific event (a tradeshow, a promotion), this page must not be displayed anymore from a given date.

To prevent from forgetting it:

- WebDev will inform the developer when the project is opened according to a configurable frequency. A GUI error will be automatically displayed.
- The application server will send an email to a given address with a specific message.

The risks to forget are minimized.

4.2 How do I proceed?

4.2.1 Defining the validity period of pages

To define the validity period of a page:

1. Display the description of the page.
2. In the "Details" tab:
 - Specify the validity period of the page.
 - Click the "Configure the reminders" button. In the window that is displayed, specify:
 - the email address of the recipient of the reminder. If the WebDev application server is configured to find the expired pages, a reminder message will be automatically sent to this address.
 - the text added to the reminder message.
 - the date for sending the reminder (one week and one day before by default). This date will also be used to display a compilation error of GUI. This compilation error will indicate whether the validity period of the page was reached.

3. Validate

4.2.2 Deleting the validity period

To delete the validity period of a page:

1. Display the description of the page.
2. In the "Details" tab, delete the validity period of the page.
3. Validate

4.2.3 Configuring the Windows application server to manage the validity period

In order for the WebDev application server to check the validity period of the pages (version 140025 and later) :

1. Start the WebDev administrator if necessary.
2. In the "Advanced" tab, check "Allow the search for expired pages".
3. The configuration window is automatically displayed.
4. Specify:
 - the start time of the search for the expired pages. At the specified time, the application server will perform a search for the expired pages on all the dynamic WebDev sites managed by the server.
 - the email address of the sender of the alert email. The parameters of the email recipients have been defined in the page description, in the editor.
 - the address and the port of the SMTP server used to send the emails. You can also specify whether the email spooler must be used.
 - the login and password of the user of the SMTP server (for the servers with authentication).
5. Validate.

4.2.4 Configuring the Linux application server to manage the validity period

In order for the WebDev application server to check the validity period of the pages (version 140025 and later):

1. Start the remote WebDev administrator if necessary.
2. In the setting options, display the "Advanced parameters" and click the "Expired pages" tab.
3. Check "Allow the search for expired pages".
4. Specify:
 - the start time of the search for the expired pages. At the specified time, the application server will perform a search for the expired pages on all the

dynamic WebDev sites managed by the server.

- the email address of the sender of the alert email. The parameters of the email recipients have been defined in the page description, in the editor.
 - the address and the port of the SMTP server used to send the emails. You can also specify whether the email spooler must be used.
 - the login and password of the user of the SMTP server (for the servers with authentication).
5. Click the "Apply" button to validate.

5. INCLUDING JAVASCRIPT FILES OR A WEB RESOURCE

5.1 Overview

WebDev enables you to include Javascript files or external resources in a WebDev project. These files can:

- be included at project level or at page level.

- be found in the _WEB directory of the site or be a file external to the site
- be in a specific format (ISO, UTF8, ...).

5.2 How do I proceed?

5.2.1 Including Javascript files (.js)

You have the ability to include one or more Javascript files in a site or in a page (*.js). These files will be used by the generated HTML pages. These files must be found in the "<ProjectName>_WEB" directory of the site.

To include a Javascript file in a page:

1. Display the description of the page. To do so, on the "Page" pane, click  in the "Edit" group.
2. Select the "Advanced" tab.
3. Select the "Javascript" tab.
4. Click the "Add" button and select the path of the Javascript file.
5. Select (if necessary) the requested encoding. If the encoding corresponds to "<Not defined>", the encoding is automatically detected.
6. Validate.

To include a Javascript file in the project:

1. Display the project description (on the "Project" pane, in "Project" group, select "Description").
2. Select the "Advanced" tab.
3. Click the "Additional Javascript files (.js)" button.
4. Click the "Add" button and select the path of the Javascript file.
5. Select (if necessary) the requested encoding. If the encoding corresponds to "<Not defined>", the encoding is automatically detected.
6. Validate.

5.2.2 Including external resources in the site

Including external resources in the site is used to include the interface files (programming interface) proposed by the external services (Web API) on the WEB. This gives you the ability to interface with these external services in browser code.

To include an external resource in a page:

1. Display the description of the page. To do so, on the "Page" pane, click  in the "Edit" group.
2. Select the "Advanced" tab.
3. Select the "Javascript" tab.
4. Click the "Add a Web resource" button and specify the address of the requested Web resource.
5. Select (if necessary) the requested encoding. If the encoding corresponds to "<Not defined>", the encoding is automatically detected.
6. Validate.

To include an external resource in the project:

1. Display the project description (on the "Project" pane, in "Project" group, select "Description").
2. Select the "Advanced" tab.
3. Click the "Additional Javascript files (.js)" button.
4. Click the "Add a Web resource" button and specify the address of the requested Web resource.
5. Select (if necessary) the requested encoding. If the encoding corresponds to "<Not defined>", the encoding is automatically detected.
6. Validate.

5.2.3 Automatic detection of the encoding

If the encoding corresponds to "<Not defined>", the encoding is automatically detected:

- case of a Web reference: the server is queried.

The encoding is supplied by the server.

- case of a file reference: if the file is in UTF8 format, the encoding is automatically switched to UTF8.

5.3 Handling external Javascript objects from WLanguage

The WLanguage in "Browser" mode is used to interface with the Web APIs such as the ones proposed by Google or Yahoo. The interaction with the external components proposed by these sites is simplified.

You have the ability to allocate external Javascript objects in browser code written in WLanguage.

The use of external Javascript objects does not necessarily require the use of the Javascript language: the programming can be done in WLanguage.

To retrieve a task list from a Google calendar:

1. Include the programming interface of Google calendar in the page.

- Display the "Advanced" tab of the page description.
- In the "HTML" tab, add the following code line into the HTML code of the page header: `<script type="text/javascript">google.load("gdata", "1");</script>`.
- This code is supplied in the Google documentation.
- In the "Javascript" tab, click the "Add a Web resource" button. Enter the address used to

include the calendar service: `http://www.google.com/jsapi?key=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx` (the key code corresponds to your personal code).

2. Enter the browser code used to retrieve the task list.

```
MyCalendarService is ...
    dynamic object
MyCalendarService = ...
new object ...
"google.gdata.calendar.CalendarService"
MyTasks is dynamic array
MyTasks = ...
    MyCalendarService:feed:entry
// Browse the array to
// fill the list
FOR i = 1 TO Dimension(MyTasks)
    ListAdd(LIST_Task_Choice,...
        MyTasks[i]:getTitle():getText())
END
```

6. AJAX

6.1 Overview

The AJAX technology is available in WebDev and in the PHP sites developed with WebDev.

What does AJAX mean and what are its benefits?

AJAX (Asynchronous Javascript and XML) is used to refresh the data modified in an HTML page without having to redisplay the entire page. For example, if some elements found in the page displayed (content of the basket, characteristics of a product, list of cities, map, ...) are modified, only these elements will be refreshed. The server does not have to send the entire page onto the computer of the Web user.

This technology presents several benefits:

- the server is less used. Therefore, it can support an important number of simultaneous connections.
- the information that circulates is less bulky.
- the transmission time is shorter.
- the display is immediate and without visual effect

for the Web user.

AJAX can be used at two different levels in a WebDev site:

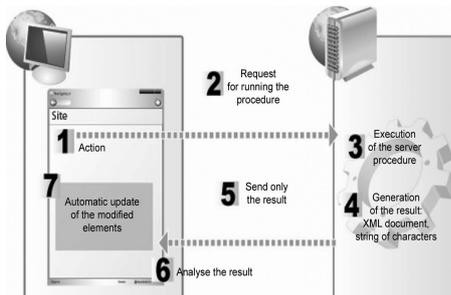
- **Automatic and instant AJAX:** a simple click gives access to the benefits of AJAX. The code remains the same.
- **Programmed AJAX:** use the AJAX functions for complex processes.

Note: Only the recent browsers support the AJAX technology (Internet Explorer 5.5 and later, Firefox 1.0 and later, Netscape 7 and later, Opera 8 and later, Safari 1.2 and later). The function named **AJAXAvailable** enables you to find out whether the current browser supports the AJAX technology. If a process that uses the AJAX technology is run on a browser that does not support this technology, the process is run "without" using the AJAX technology (the entire page is refreshed for example).

6.2 Automatic and immediate AJAX

6.2.1 Overview

The diagram below presents the automatic and immediate use of AJAX in a WebDev site:



For example, a site page is used to find out the different characteristics of a country (capital city, currency, flag, location, ...). The corresponding information is displayed according to the country selected by the Web user.

- 1** Action performed by the Web user. In our example, the Web user selects the country in the combo box named "Select a country".
- 2** Sending the query to the server.
- 3** Running the query: find the characteristics of the selected country.
- 4** Sending the result of the query:
 - without AJAX: the entire page is sent.
 - with AJAX: only the characteristics of the country are sent.
- 5** Displaying the characteristics of the country:
 - without AJAX: the entire page is redisplayed.
 - with AJAX: only the controls containing the characteristics of the country are refreshed.



To use AJAX in this site, the "Whenever Modified" server process of the "Select a country" control is switched to AJAX mode. A simple click is required! The code remains the same.

6.2.2 Processes that can use AJAX automatically

The following processes can be switched to automatic AJAX mode:

- server "Click" process of a button, link, clickable image or pager.
- server "Whenever Modified" process of a list box, combo box, check box or radio button.

To switch to the automatic AJAX mode, click "AJAX" in the bar of the process :



Process that does not use the AJAX technology



Process that uses the AJAX technology

Note: If a process that uses the AJAX technology is run on a browser that does not support this technology, the process is run "as if" it was not using the AJAX technology (the entire page is refreshed for example).

6.2.3 Elements and characteristics that can be automatically used by AJAX

AJAX enables you to modify the characteristics of the following elements (no specific programming is required) :

	Value	Color of the font	Background color	Visibility
Edit control	X (Text entered)	X (Color of the text entered)	X (Background color of the text entered)	X
Formatted display control	X (Text displayed)	X (Color of the text displayed)	X (Background color of the text displayed)	X
Table	X (Content of the rows)	X (Color for the content of the rows)	X (Background color for the even and odd rows)	X
Looper	X (Content of controls)	X (Color for the content of controls)	X (Background color of controls)	X
List	X (Listed elements and selected elements)	X (Color of listed elements)	X	X
Combo box	X (Listed elements and selected elements)	X (Color of listed elements)	X	X

Check Box	X (Selected options)	X (Color of options)	X	X
Radio button	X (Selected option)	X (Color of options)	X	X
Image	X (Image)			X
Clickable image	X (Image)			X
Thumbnail	X (Image)			X
Chart	X (Chart)			X
Static	X	X	X	X
HTML static	X	X	X	X
Button	X (Static)	X (Color of the caption)	X	X
Link	X (Static)	X (Color of the caption)	X	X
Pager *	X (Content)			
Cell control			X	X
Page			X	
TreeView	X	X	X	X

Notes:

- The number of characteristics automatically managed will increase in the forthcoming versions.
- The elements not found in this list cannot be automatically used by AJAX.
- To modify other characteristics, you must use the programmed AJAX.

* The use of Pager controls is specific in AJAX mode. See "Specific feature" for more details.

Automatic AJAX indicator

An automatic progress bar can be displayed in a page while an AJAX process is performed.

Indeed, the AJAX processes do not display the standard progress bar for page load: there is no page load in AJAX

To display an automatic AJAX indicator:

1. Add a control (cell, image, static or HTML static) to the page.
2. Customize this control. This control may contain some text ("Loading..." for instance) and/or an animated image.
3. Display the "Details" tab of the page description ("Description" from the popup menu of the page).
4. In the "Automatic AJAX Progress Bar" area, select the added control and define its position.

6.2.4 Specific features

Specific features of the Pager control

When a table or a looper is **automatically updated** in AJAX mode (when clicking a button that runs AJAX code for instance), the content of the pager associated with this table or with this looper is also automatically updated.

Note: For the clicks performed on the pager to be in AJAX mode, you must:

- check "AJAX Mode" in the description window of the pager control, "General" tab.
- switch the click process of the pager control to automatic AJAX mode (click "AJAX" found in the bar of the process).

Table/Looper control and AJAX mode

When a table or a looper is in AJAX mode (AJAX mode specified in the control description), the table rows and the looper rows are no longer viewed via a pager but via the vertical scrollbar.

See **AJAX tables** for more details.

Note: The scrollbars are not available in Mozilla Firefox version 1.7.8

Special characters

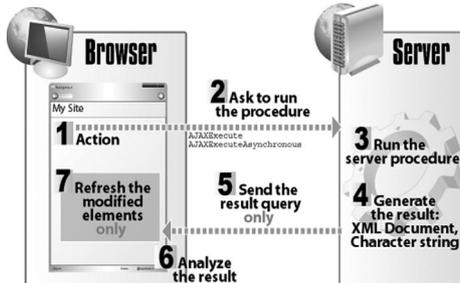
Some special characters are automatically replaced by an empty string ("") in AJAX mode. These characters are the characters whose ASCII code is strictly less than 32, except for the characters 9, 10 and 13 (TAB and CR respectively).

In most cases, these special characters are not used in the strings.

6.3 Programmed AJAX

6.3.1 Overview

The diagram below presents the use of "programmed AJAX" in a WebDev site:



- 1** Run a browser process (`AJAXExecute` or `AJAXExecuteAsynchronous`).
- 2** Request for running a server procedure.
- 3** Running the server procedure.
- 4** Generating the result. The result of the procedure will be contained in a character string or in an XML document.
- 5** Send the result of the procedure (**RESULT**).
- 6** Analyze the result of the procedure.
- 7** Displaying the modified information. Only the necessary controls will be refreshed.

6.3.2 Functions for AJAX management

Several AJAX functions can be used to manage the complex processes:

AJAXAvailable Used to find out whether the AJAX technology is supported by the current browser.

AJAXExecute Runs a server procedure without refreshing the page. This function is a locking function. No process will be run as long as the result of the procedure run is not retrieved.

AJAXExecuteAsynchronous Runs a server procedure without refreshing the page. This function is a non-locking function. The other processes continue to run (no matter whether the result of the procedure run is retrieved or not).

AJAXAsynchronousCallPending Used to find out whether a server procedure called by `AJAXExecuteAsynchronous` is currently run

AJAXCancelAsynchronousCall Cancels the automatic execution of the browser procedure called by `AJAXExecuteAsynchronous`.

These functions enable you to run a server procedure from a browser process.

Caution: Make sure that the information stored on the server is consistent with the information displayed on the computer of the Web users. For example, if data is modified, this data must be modified both on the server and on the page displayed in the browser.

6.3.3 Procedures that can be called by AJAX

To secure the WebDev sites, the server procedures are protected from illegal calls (attempt to re-route a session for example). To run a server procedure from a browser process (**AJAXExecute** or **AJAXExecuteAsynchronous**), you must allow this procedure to be called by AJAX.

To allow a server procedure to be called by AJAX, click "AJAX" in the bar of the procedure:



Procedure that cannot be called by AJAX



Procedure that can be called by AJAX

6.3.4 WLanguage functions useless in AJAX

All the functions available in WebDev server can be used in AJAX. However, the following functions are useless in AJAX. Therefore, these functions are not available in AJAX.

StringDisplay	Displays a character string (or a buffer) in the browser
ContextClose	Closes a page context
ContextOpen	Opens a new page context without returning the information to the browser

FileDisplay	Displays a file in the browser of the Web user
FramesetRefresh	Refreshes a frameset displayed in the browser of the Web user from the context found on the server
FramesetDisplay	Displays a WebDev frameset in the browser of the Web user
FramesetUse	Displays a WebDev frameset in the browser of the Web user and closes all the current page contexts and frameset contexts
InitWindow	Resets (or not) the controls found in the current page and runs the initialization processes of the controls
PageInitialization	Resets (or not) the controls found in the current page and runs the initialization processes of the controls
PageUse	Displays a WebDev page in the browser of the Web user and closes all the current page contexts
ScriptDisplay	Calls an external script (.php, .asp, .mhtml or .mht) and returns the result page in the current browser window
UploadCopyFile	Saves on the server a file "uploaded" by the Web user
UploadFileName	Returns the name of a file "uploaded" by the Web user
Use	Displays a page in the browser of the Web user

This list can evolve. We recommend that you check the documentation for each function. The "AJAX" symbol indicates whether the function is available in AJAX mode or not.

7. VISTA GADGETS

7.1 Overview

Windows Vista enables you to install "Gadgets" on the desktop. These gadgets are HTML pages that can be easily created by WebDev.

WebDev allows you to create Vista gadgets from static sites or from AWP sites.

A Vista gadget can contain three types of pages. These three types of pages must respect specific dimensions:

- **Main page:** This page is displayed in the gadget pane.
The width of this page must be included between 25 and 130 pixels, and its minimum height must be equal to 60 pixels.
- **Configuration page:** This page is optional. This page is used to configure the gadget and it is opened via a special icon.

This page automatically contains 2 buttons: "OK" and "Cancel".

The width of this page must be equal to 278 pixels.

- **Flyout page:** Free pages that can be opened from the main page (via a button, a link, ?).
The maximum size for this type of page is equal to 400 pixels in height and in width.

WebDev enables you to create these three types of pages and to generate the ".gadget" file. This file can be supplied to the users of the gadget so that it can be installed on their computer (gadgets that can be downloaded from a site for example).

Note: The characteristics of the three types of pages are presented in details in "Programming the Vista gadgets", page 226.

7.2 Creating the gadget

7.2.1 The steps

To create a Vista gadget:

1. Open or create a WebDev project.
2. Create a specific configuration for the development of your Vista gadget. To do so, on the "Project" pane, in the "Configuration" group, expand "New configuration" and select "Vista gadget".
3. Create the pages of your Vista gadget. These pages can be static pages or dynamic AWP pages.
4. In the description of the gadget pages ("General" tab), specify the type of page in the gadget: Main page, Configuration page or Flyout page.
5. Program the operating mode of your pages. The "Vista gadget" pages propose specific processes.
6. Generate the Vista gadget. To do so, click "Generate" in the "Home" pane.

Vista gadget in AWP mode:

- In order for the Vista gadgets in AWP mode to operate, you must use a WebDev application server version 120048 or later.
- The server codes will be run in automatic Ajax mode (if the code exists in Ajax mode).
- The AWP site corresponding to the gadget must be deployed on a WebDev application server.

Limitations

The following controls cannot be used in a Vista gadget: TreeView control, Java control, Site Map control, Site Map Path control.

The image control in a looper in generated mode or in database mode (memo) displays no image.

7.2.2 Generating the gadget

To generate a Vista gadget:

1. Display (if necessary) the configuration corresponding to the Vista gadget (double-click the name of the configuration in the project explorer).
2. In the "Home" pane, click "Generate". The wizard starts.
3. Specify the description options of the gadget:
 - Name of the gadget
 - Version of the gadget (in www.xxx.yyyy.zzzz format)
 - Name of the author or company
 - URL and text of the link
 - Logo of the company
 - Copyright
 - Image of the gadget. The rollover image is displayed when the gadget is moved.
 - Description of the gadget.

4. Select the pages that must be included in the gadget. The main page is automatically selected.
5. Select the languages of the gadget.
6. If the Vista gadget uses AWP pages, specify the characteristics of the server (name of the server, virtual directory).

7. Validate. The gadget is created in the "EXE" subdirectory of the project (in the directory corresponding to the project configuration). The extension of the gadget is ".gadget". If the development computer is running Windows Vista, the gadget can be installed directly and its test can be run.

7.3 Programming the Vista gadgets

7.3.1 Overview

WebDev allows you to easily create gadgets for Windows Vista. A gadget is a "mini site" containing specific pages.

To manage these pages and to exploit all the properties of the Vista gadgets, WebDev proposes:

- specific processes associated with the different types of pages of a Vista gadget.
- specific WLanguage functions used to manage some specific features of the Vista gadgets.

7.3.2 The different types of pages

Three types of pages can be used in a gadget:

- Presentation page
- Configuration page
- Flyout page

These three types of pages contain specific processes.

Note: only the browser processes are available if the pages of the Vista gadget are static pages. If the gadget pages are AWP pages, the standard server processes are added to the browser processes.

Presentation page

The presentation page is the page displayed in the gadget pane in Windows Vista. The width of this page must be included between 25 and 130 pixels, and its minimum height must be equal to 60 pixels.

To create a clipped gadget, specify a background image. This image must support the transparent mode (image in GIF or PNG format for example).

This page can be used to display the "Flyout" pages. To do so, **GadgetDisplayFlyout** must be used in the code of the relevant button.

The browser code associated with this page is as follows:

- **Loading (onload) of the page**
Browser code run when the page is displayed in the browser (created by **PageDisplay** or **PageRefresh** for example)

- **Unload (onunload) of the page**
Browser code run when a new page is displayed in the browser.
- **Anchoring the page in the Windows pane**
Browser code run when the Vista gadget is "docked" to the gadget pane of Windows.
- **Exit from the page of Windows pane**
Browser code run when the Vista gadget is "undocked" (exit) from the gadget pane of Windows.
- **When closing the configuration window**
Browser code run when closing the page used to configure the Vista gadget.
- **Opening the configuration window**
Browser code run when opening the page used to configure the Vista gadget.
- **Mouse out**
Browser code run when the mouse cursor is located outside the Vista gadget.

Configuration page

The configuration page is used to configure the Vista gadget. This page is displayed by the configuration button (tool icon) found in the gadget. This button is automatically displayed if a configuration page is found. This button cannot be configured.

The width of this page must be equal to 278 pixels.

The configuration page automatically contains a validation button and a cancelation button.

In the configuration page, a font must be specified in the styles: if the font is "undefined", the default font will be used. In this case, differences in page layout may occur between the edit mode and the runtime mode.

The background color of the configuration page is gray. This color cannot be modified. A background image can be used. If this background image manages the transparency, it is recommended not to write in the transparent areas.

The browser code associated with this page is as follows:

- **Loading (onload) of the page**
Browser code run when the page is displayed in the browser (created by *PageDisplay* or *PageRefresh* for example)
- **Unload (onunload) of the page**
Browser code run when a new page is displayed in the browser.
- **Validating the configuration window**
Browser code run when the validation button is pressed.
- **Canceling the configuration window**
Browser code run when the cancelation button is pressed.

Note: *GadgetLoadParameter* and *GadgetSaveParameter* are used to manage the storage of the parameters.

FlyOut page

The flyout pages are standard pages. These pages are free pages that can be opened from the main

7.3.3 The WLanguage functions

The Gadget functions are as follows:

GadgetCloseFlyout	Closes the popup area of a vista gadget.
GadgetDisplayFlyout	Displays a page of the Vista gadget in a popup area.
GadgetLoadParameter	Loads a persistent value in a Vista gadget. This value was saved by <i>GadgetSaveParameter</i> .
GadgetSaveParameter	Saves a persistent value in a Vista gadget. Then, this value can be read by <i>GadgetLoadParameter</i> .

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

page (via a button, a link, ...).

The maximum size for this type of page is equal to 400 pixels in height and in width. The background color of the flyout pages is white. This color cannot be modified.

In the flyout pages, a font must be specified in the styles: if the font is "undefined", the default font will be used. In this case, differences in page layout may occur between the edit mode and the runtime mode.

The flyout pages contain the following processes:

- **Loading (onload) of the page**
Browser code run when the page is displayed in the browser (created by *PageDisplay* or *PageRefresh* for example)
- **Unload (onunload) of the page**
Browser code run when a new page is displayed in the browser.

These pages can be handle by *GadgetDisplayFlyout* and *GadgetCloseFlyout*.

8. USING OFFLINE SITES

8.1 Overview

WebDev allows you to specify that a site or a set of pages can operate in off-line mode (without an Internet connection).

If the site can operate in offline mode, all the resources (pages, images, .js files, .css files) used by the site or by the pages are automatically cached by the browser on the computer of the Web user.

Therefore, the site can be used without having to reconnect to the Internet.

The browser function named ***BrowsersIsConnected*** is used to find out whether the browser is in online or offline mode and to adapt the site behavior accordingly.

This feature uses the management of manifests available with HTML 5. Some browsers (especially Internet Explorer 8 or earlier browsers) do not support this feature.

Note: The mechanism of site cache is specific to HTML 5:

- Prior to HTML 5, only the visited pages could be cached.
- With HTML 5, the cache mechanism is performed from the first connection to a site. A specific manifest file is used to list all the resources to cache.

8.2 Implementation

8.2.1 Defining the resources to cache

Defining the resources to cache is performed in the development environment. To indicate the pages available in offline mode:

1. Open the WebDev project if necessary.
2. On the "Project" pane, in the "Web" group, select "Cache (offline mode)".
3. In the window that opens, select the pages to cache.

Caution: Only the dynamic AWP pages, the dynamic PHP pages and the static pages can be cached. Close the window.

During the next recompilation of the project, the "<Project_Name>.manifest" file is automatically generated in the language sub-directory found in the _WEB directory of the project. This file will contain the specified pages as well as all their dependencies (images, CSS style sheets, Javascript files, ...).

8.2.2 Configuration of the server

The use of a manifest requires a specific configuration of the server. A specific mime type must be configured: ".manifest" associated with "text/cache-manifest".

To run the test of your site, this configuration must be performed on the development computer.

Notes:

- In this version, the configuration of the mime type must be performed manually. In a forthcoming version, this configuration will be automatic.
- The configuration of the mime type is automatic.
- This mime type is not configured yet for all the hosting companies.

8.2.3 Programming technique

A cached site can be used for example:

- to simulate an iPhone application (creation of a Web application that operates in offline mode only).
- to manage a site that would lose the Internet connection.

Programming tips:

- The principle is to mainly use browser code. Indeed, only this code can be run off-line. However, you have the ability to use a server code via ***AjaxExecute*** (that is used to run a server code in a browser process).
- Only the browser controls are available. Only the list boxes, buttons, edit controls, static controls, images, links, menus, check boxes and radio buttons can be used in the pages in off-line mode.

- The buttons and the links must not have the "Submit" type. However, you have the ability to use the "Display the page XXX" action.
- If the data backup is required in off-line mode, a local database must be used. The HTML 5 standard allows the Internet sites to use a local database managed by the browser. See SQL Database local to a browser for more details.
- If the site must operate both in online mode and in offline mode, you have the ability to use **BrowsersIsConnected** in browser code to find out whether a server code can be run or not. In case of disconnection, a specific procedure or process can be started to prevent the server code from being run.
- Use the optional processes of the page ("Switch to offline mode", "Switch to online mode" and "Whenever the status of the HTML cache changes") to perform the processes required to change status (retrieving the data from the local database for example).

8.2.4 Running the test of the site

To run the test of a cached site:

1. Configure (if necessary) the server of the development computer.
2. Run the test of the site (Go). All the necessary pages are cached.
3. Use the "Work offline" option of the browser.
4. Run the test of your site.

8.3 Access in local mode to a database (SQLite)

WebDev allows a site to create and access a database created by the browser on the computer of the Web user in browser code.

This features gives you the ability to enter data in disconnected mode and to transmit this data to the server as soon as the connection to Internet is restored.

Caution: This feature is available for some browsers only:

- Chrome,
- Safari,
- Opera 11, ...

8.3.1 How do I manage a local database?

1. Use **SQLConnect** in browser code to connect to the local database.

For example:

```
// Connection to a browser database
// named "LocalDatabase"
// The database is created if it
// does not exist
SQLConnect("LocalDatabase",...
"","","","Web SQL database"))
```

2. Use **SQLExec** to perform queries on the local database

Caution: The SQL queries are run in asynchronous mode. Therefore, the syntax of **SQLExec** uses a spe-

cific procedure. This procedure is started at the end of the real execution of the query (regardless of the query result). This browser procedure is used to:

- check the proper execution of the query. The function named **SQLInfo** is automatically run during the call to the procedure. Therefore, all the SQL variables are positioned. If an error occurred, **SQL.Error** will differ from "00000". The error message is returned by the **SQL.MesError** variable.
- browse the result of the query.

If new queries are run in this procedure (to add records for example), you can:

- use the same procedure: the parameter of this procedure is used to find out the query currently run.
- use a different browser procedure to test the result of these new queries.

Notes:

- During the exit from the browser procedure, the values returned by **SQLInfo** are restored. If these values have been modified in the browser procedure, they are overwritten.
- To find out the SQL commands that can be used, see the documentation about the "Web SQL database".

8.3.2 The SQL functions

The following SQL functions are available in browser code:

SQLChangeConnection	Modifies the current connection.
SQLClose	Declares the end of the query execution and frees the memory resources allocated during the execution of the query.
SQLColumn	Returns the characteristics of all the columns (or items) : <ul style="list-style-type: none"> • for a given table. • for a given query.
SQLConnect	Connects the current application to a database that must be interrogated by SQL.
SQLDisconnect	Closes the current connection and frees the memory used by the connection
SQLExec	Names and runs a SQL query.
SQLFetch	Goes to the next line (which means to the next record) of the query result.
SQLGetCol	Retrieves the content of the specified column from the result of the query, for the current row.
SQLInfo	Initializes the different SQL variables with the information relative to the last query run
SQLReqExists	Checks the existence of a query.

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

8.3.3 Saving the data of an offline site locally

WebDev allows you to implement offline sites. If this type of site must save data during the disconnection period, you have the ability to use a local database.

The steps are as follows:

1. Connecting to the database.

The connection to the database is established by **SQLConnect**.

For example:

```
SQLConnect("", "", "", ...
"APTCustomers", "Web SQL database")
```

2. Creating the local database.

The local database must be created before it can be used. To do so, a CREATE TABLE query must be used. This query must be run by **SQLExec**.

For example:

```
sQuery is string
// Code for creating the table
sQuery = [
CREATE TABLE IF NOT EXISTS
"Appointment" ("IDAppointment"
INTEGER PRIMARY KEY , "StartDate-
Time" VARCHAR(16) ,
```

```
"Customer" VARCHAR(100) ,
"Address" VARCHAR(200) ,
"Summary" LONGVARCHAR );
]
// Run the query
SQLExec(sQuery, ...
"QRY_CREATION", _cbQuery)
```

When the "QRY_CREATION" query is completed, the `_cbQuery` procedure is run. In this example, this procedure is used to manage all the queries run and to perform an appropriate process after the query. The creation query must be used once, at the beginning of the day for example. For example, the database can be created only if a specific parameter ("First" for example) is not passed to the pages.

3. Local or remote data access.

In this example, the site can be used in online mode or in offline mode. You must be able to access the local database (in offline mode) or to the remote database (in online mode).

All the accesses (creation, modification, deletion, ...) to the local database must be performed via the **SQLExec** procedure.

The accesses to the remote database can always be performed via the Hxxx functions. However, you must:

- run these functions in a browser code, therefore in AJAX, via **AJAXExecute**.
- make sure that the browser is connected (**BrowserIsConnected**).

For example:

```
sQuery is string
// The browser is connected
IF BrowserIsConnected()...
  = True THEN
  // Saves in the remote database
  //(server)
  AjaxExecute(SaveData, ...
    EDT_ID, EDT_SUMMARY)
END
sQuery = [
UPDATE Appointment
SET Summary='%2'
WHERE AppointmentID=%1
]

// In all cases, saves
// in the local database
SQLExec(StringBuild ...
(sQuery, gnIDAPTBrw, ...
EDT_SUMMARY), 'QRY_BACKUP', ...
_cbQuery)
```

To retrieve records from the remote database (server) into the local database (to initialize it for example), you must:

1. Read the records and store the items as strings, in a server procedure.

```
PROCEDURE GetRec()
FOR EACH MyFile
  sList += [CR] + ...
  MyFile.Item1 +TAB + ...
  MyFile.Item2 +TAB+ ...
  MyFile.Item3
END
RESULT sList
```

2. Retrieve this list in browser code:

```
// Retrieves the list
sList = AJAXExecute(GetRec)
```

3. Browse the string and extract the information with **ExtractString**. Therefore, the records can be easily added with an addition query (**INSERT**).

Similarly, the data can be retrieved from the local database to update the remote database. To do so:

1. Run a query locally to retrieve the records. For example, to retrieve the appointments of the day:

```
// Runs the query that
// selects the appointments
// for the day
sQuery is string
sQuery = StringBuild(...
  "SELECT IDAppointment, Summary ...
  FROM APPOINTMENT WHERE ...
  StartDateTime LIKE '%1%' ORDER ...
  BY StartDateTime ASC;", ...
  Today())
// Starts the query
SQLExec(sQuery, ...
"QRY_SYNCHRONIZEAPT", _cbQuery)
```

2. In the check procedure started by **SQLExec**, all you have to do is run a procedure for updating the remote database via **AjaxExecute**. In our example, the browser procedure named **_SynchronizeDatabase** is started:

```
PROCEDURE _SynchronizeDatabase ...
(sQuery)

// As long as there are appointments
// to synchronize
WHILE SQLFetch(sQuery) = 0
// Synchronizes the appointment
AJAXExecute(_Synchronize ...
ARemoteAppointment, SQLGetCol ...
(sQuery, 1), SQLGetCol...
(sQuery, 2))
END
```

Note: The update of the remote database can be triggered during the reconnection to the server. During the reconnection to the server, the "Switch to online mode" process is run. This process is an optional process of the page. To display it in the code editor, you must:

- Display the code of the page.
- Click [...] in the code bar.
- Select the "Switch to online mode" event.
- Enter the code for updating the remote database.

9. LOCAL STORAGE

9.1 Overview

In a Web site, the local storage is used to store and retrieve values in browser code.

These values can be:

- shared by all the windows and tabs of the same computer browser: they can be restored once the browser is closed then re-opened or in another browser window. In this case, the local storage is persistent.
- specific to a window or to a tab of the browser: in this case, the values will be lost when closing the browser window. On the contrary, these values can be read by the browser windows opened from the window where the values have been stored.

In this case, the local storage is not persistent.

Notes:

- The storage of local parameters is not available by page.
- The storage of local parameters is available for the recent browsers only (supporting the HTML 5 standard).
- Even for the persistent local storage, the storage is not shared by the different browsers (a value stored in Internet Explorer cannot be retrieved by Firefox).

9.2 The WLanguage functions

The following functions are used to perform a local storage:

LocalStorageAdd	Adds a value to the local storage.
LocalStorageAvailable	Indicates whether the local storage is available or not for the current browser.
LocalStorageDelete	Deletes a value from the local storage.
LocalStorageDeleteAll	Deletes all the values from the local storage.
LocalStorageGet	Retrieves a value from the local storage.
LocalStorageOccurrence	Returns the number of values for the local storage.
LocalStorageValueName	Returns the name of a value for the local storage.

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

10. SSL: SECURED TRANSACTIONS

10.1 Overview

By default, when using an Internet application, the data circulating between the computer of the Web user and the Web server is not protected: the data can be accessed by any Web user. The confidentiality of data is not guaranteed.

Several systems can be used to guarantee the confidentiality of data. The most common one is the SSL protocol (Secure Socket Layer).

SSL is a communication protocol used to ensure the authentication, the confidentiality and the integrity of data. This protocol uses a recognized

encryption method: the RSA public key algorithm. A RSA key is the result of operations between prime numbers.

Practically: what changes when a transaction is secured?

- a secured transaction uses a specific address (https instead of http).
- a padlock is automatically displayed in the status bar of the browser, indicating that a secure transaction is in progress.

10.2 Implementing the secured transactions with the SSL protocol

To implement secure transactions (SSL protocol) :

1. On the Web server, create a key, with the IIS key manager for example if your Web server is IIS.
2. Send the certificate request to a company providing key certificates (www.verisign.com for example).
3. The company returns a key certificate (for a fee). This certificate must be installed on the server by the key manager.

4. You can then use secure transactions between the server and the client (see the next paragraph).

Note: "Verisign" provides test certificates, valid for 14 days, allowing you to check whether the HTTPS connection is correct. These test keys enable you to check the operating mode of secure transactions.

The complete procedure to follow in order to obtain an SSL certificate is detailed later in this chapter.

10.3 Obtaining an SSL certificate for IIS2 (2.0 or later)

10.3.1 Step 1: Creating a certificate

To create a certificate:

1. Start the Microsoft Internet Service Manager ("Start .. Run": inetmgr.exe).
2. Select the Web site to secure (for example: "Default Web site").
3. Right-click this site and select "Properties" from the popup menu.

4. Select the "Directory security" tab:

- Click the "Server certificate..." button. A wizard starts.
 - Choose "Create a certificate".
 - In the "Usual site name" pane of the wizard: leave the name of your computer.
 - The wizard creates a text file named "certreq.txt" allowing you to ask for your certificate.
5. Exit the properties of the Web server.

10.3.2 Step 2: Requesting a certificate

The certificate request must be sent to a certificate authority (for example: www.verisign.com).

This example presents the steps to follow to send a certificate request to Verisign. These steps are similar for the other authorities.

1. In your browser, type: "http://www.verisign.com".
2. Click "Free Trial". This option enables you to get a free certificate valid for 15 days.
3. Fill out the short form and click "Submit".
4. Follow the certification wizard step by step:
 - Step 1 to 5: Generate CSR: Click "Continue"
 - Step 2 to 5: Submit CSR: Copy the content of the created file ("certreq.txt") into the relevant input area then click "Continue".
 - Step 3 to 5: Complete Application: The content of the certificate is listed on the screen, enter your personal details (Caution, the accented characters are not accepted, most of the information must be the same as in the certificate. Example: region, company, ...). Enter your email address, the certificate will be sent to this address. Click "Accept".

The certificate will be sent to you within an hour.

10.3.3 Step 3: Certification of the certificate on the server

1. The certificate is found in the email sent by VeriSign.

- Select the part that looks like this.

```

---BEGIN CERTIFICATE---
MIICBzCCA-
bECEFi+J6vFjN+EkkfCKLgI6uUwDQYJKoZIhvc
NAQEEBQAwgaxFjAUBgNVBAoTVDZlcm1TaWduLCB
JbmMxRzBFBgNVBAstPnd3dy52ZXJpc2lnbi5jb20v
cmVwb3NpdG9yeS9UZXRNOQ1BTIEluY29ycC4gQnk
gUmVmLiBMaWFiLiBMVEQuMUYwRAYDVQLEZ1G
b3IgVmVyaVNPZ24gYXV0aG9yaXplZCBOZXNOaW5
nIG9ubHkuIE5vIGFzc3VyYW
    
```

```

5jZXMgKEMpVIMxOTk3MB4XDTAwMDkwMTAwMD
AwMFoXDTAwMDkxNTIzNTk1OVowZDELMAkGA1U
EBhMCRlIxEjAQBgNVBAgTCUxBTkdVRURPQzEUMB
IGA1UEBxQLTU90VFBFTExJRVlxDDAKBgNVBAoUA
1BDUzEPMA0GA1UECxxQGV0VCREVWwQwwCgYDV
QQDFANtcjIwXDTANBgkqhkiG9w0BAQEFAANLADBI
A kEAmqKCTidFBZEwiOQ7WPTBIQzIacQi5nwjNndrW.
U2tjGMElrY0IoduwTgRX+DY42IuptGERQApb5NCx
/4/3iBrQIDAQABMA0GCsqGSib3DQEBAUAAO
EABT7REV9bLiq3Efiug+a/
irIbY17aNIuITsdSVO7j34baFtx
aa2jTE4JoGrNhrWTQZgBl1GOMpuMef/
MuwXH01A==
---END CERTIFICATE---
    
```

Copy the selected part into a new file named "Certif.cer" for example.

2. Start the Internet Service Manager ("Start .. Run": Inetmgr.exe).
3. Select the Web site to secure (for example: "Default Web site").
4. Right-click this site and select the "Properties" option.
5. Select the "Directory security" tab:
 - Click the "Server certificate ..." button, then follow the wizard.
 - Choose "Process the request and install the certificate".
 - Select the created file (Certif.cer).
 - End the wizard.

Important note:

If a test certificate was already installed on this computer:

- Remove the former certificate (select "Delete the certificate" in step 1).
- Change the name of your computer (indeed, two test certificates cannot be installed on the same computer = same name).
- Start again from step 1.

10.4 Inserting secure transactions (SSL) into a WebDev site

In an Internet/Intranet site, all the transactions do not necessarily contain confidential data: all you have to do is secure the transfer of the sensitive data (transferring a credit card number for example).

Caution: the implementation of a secure transaction in a WebDev site assumes that all the steps linked to the SSL protocol have been complied with.

10.4.1 Principle

The secure mode is implemented when opening a new page via a button or a link, with **SSLActive**. As soon as the secure page is opened, all the actions and page opening will be performed in secured mode, regardless of the elements used (link, table, click area).

The secured mode ends when **SSLActive** is used in the code of a link or button used to open a new page.

10.4.2 Implementation

To open a page in secure mode:

1. Create a button used to open a page. This page can be opened either by an action defined in the button description, or by programming in server code. The characteristics of this button are:

- Type: "Send value to server" or "None".
- Action: "Display page XXX" (or "None" if the page is opened by programming).
- Destination:
 - "Current browser" to open the page in the current browser.
 - "New browser" to open the page in a new browser.

2. Enter the following code line in the browser click code of the button:

```
SSLActive (True)
```

3. If necessary (action not defined in the description of the button), use **PageDisplay** in the server click code to display the requested page in secured mode.

4. The page opened by the button will be in secu-

red mode. This secured mode will be used until the non-secure mode is explicitly requested.

10.4.3 Going back to standard mode (non-secured transaction) in the current browser

To go back to standard mode (non-secure transaction) in the current browser:

1. Create a button used to open a page. This page can be opened either by an action defined in the button description, or by programming in server code. The characteristics of this button are:

- Type: "Send value to server" or "None".
- Action: "Display page XXX" (or "None" if the page is opened by programming).
- Destination:
 - "Current browser" to open the page in the current browser.
 - "New browser" to open the page in a new browser.

2. Enter the following code line in the browser click code of the button:

```
SSLActive (False)
```

3. If necessary (action not defined in the description of the button), use **PageDisplay** in the server click code to display the requested page in non-secured mode.

4. The page opened by this button will be opened in non-secured mode. This non-secure mode will be used until the mode is explicitly requested.

Special case: Opening a page in secured mode with **BrowserOpen**:

1. Create a global variable (addrPage) for example in the home page (not secured).

2. Initialize this variable with the secured address of the page:

```
addrPage = PageAddress(...
    <StartPageName>, paSecure)
```

3. In the requested code, use:

```
BrowserOpen (AddrPage)
```

11. JSON

11.1 Overview

JSON (JavaScript Object Notation) is a light format for exchanging data. Based on JavaScript, JSON is a text format independent of any other language.

JSON is based on two structures:

- a set of name/value couples, interpreted in WLanguage by structures.
- a list of organized values, interpreted in WLanguage by arrays.

JSON and WebDev

WebDev is used to:

- get information in JSON format in a WebDev site.
- create WebDev pages that return information in JSON format. You have the ability to create a site that provides JSON services (a site for monitoring packages for example).

11.2 Getting information in JSON format

11.2.1 Operating mode

To get information in JSON format, WebDev allows you to run a page that returns information in JSON format. Two WLanguage functions are available:

- **JSONExecute**: call a server URL from the same domain that returns data in JSON (JavaScript Object Notation) format.
- **JSONExecuteExternal**: call an external server URL that returns data in JSON (JavaScript Object Notation) format. The data is processed in a specific procedure.

11.2.2 Example for using the JSONExecute function

The following code is used to run an AWP page in order to get the list of contacts in JSON format.

The steps are as follows:

1. Declaring a dynamic object. This dynamic object will contain the result in JSON format.

```
MyContacts is dynamic object
```

2. Calling **JSONExecute**:

```
MyContacts = JSONExecute(...
FolderWeb() + ...
"FR/PAGE_Objct.awp?id=12")
```

3. Processing the JSON data: this process is written in browser code.

Let's take a simple example: the JSON data is returned in the following format:

```
{id: 12,
list: [
{last name: "smith", first name:
"john"},
{last name: "dupond", first name:
"marie"},
{last name: "martin", first name:
"laura"}]
}
```

You can for example:

- Retrieve the value of a member of the dynamic object. For example:

```
MyContacts.id
```

- Retrieve the different elements from a list of values. For example:

```
FOR i=1 _TO_ Dimension(...
MyContacts:list)
ListAdd(LIST_List_contacts,...
MyContacts:list[i]:last name+"
"+ ...
MyContacts:list[i]:first name)
END
```

11.2.3 Example for using the `JSONExecuteExternal` function

The principle for using `JSONExecuteExternal` is the same as the principle for using `JSONExecute`.

In this case, the page used to retrieve the JSON data is not found in the same domain. The security rules of Internet impose to use a callback function to process the result in asynchronous way. The function named `JSONExecuteExternal` is used to

indicate the name of the procedure run to process the JSON data.

Example: In this example, an AWP page is called and the data is processed in the procedure named `FunctionResponse`:

```
JSONExecuteExternal(...
"http://MySite/MySite_WEB/US/
PAGE_Object.awp?id=12",...
"JsonCallback",FunctionResponse)
```

11.3 Creating pages that return JSON data

WebDev gives you the ability to create a site that provides JSON services.

The pages used to get the JSON data must return a character string of a specific format. These pages can be AWP pages or PHP pages.

11.3.1 Case of a page called by `JSONExecute`

The following code is used to return an identifier and a list of values:

```
sObject is string = [
{id: 12,
list: [
{last name: "smith", first name:
"john"},
{last name: "dupond", first name:
"marie"},
{last name: "martin", first name:
"laura"}]
}
]
// StringToUTF8 is used to manage
// the accented characters
StringDisplay(StringToUTF8(sOb-
ject))
```

The function named `StringDisplay` is used to return the JSON information. The function named `StringToUTF8` is used to return a string in UTF8 format. This last function is mandatory to get a valid format.

11.3.2 Case of a page called by `JSONExecuteExternal`

In addition to the code used to manage the JSON element to return, the name of the procedure that will be used to process the data must be specified in the data returned. The name of this procedure is indicated in parameter on the URL of the page.

The following example is used to manage the call with `JSONExecute` and `JSONExecuteExternal`. For an external call, you must check the presence of a specific parameter found on the URL. This parameter was supplied to the user of the JSON service.

```
sObject is string = [
{id: 12,
list: [
{last name: "smith", first name:
"john"},
{last name: "dupond", first name:
"marie"},
{last name: "martin", first name:
"laura"}]
}
]

// Manage an external call.
// Check the presence of a name of
// procedure on the URL
IF PageParameter(...
"JsonCallback") <> "" THEN
sObject = PageParameter(...
"JsonCallback") + "(" + ...
sObject + ");"
END
// StringToUTF8 is used to manage
// the accented characters
StringDisplay(StringToUTF8(sOb-
ject))
```

To allow the users of the WebDev sites to use the JSON services, all you have to do is inform these users of the parameters that must be specified in the address of the page to start.

Note: For more details about the syntaxes that can be used in the data returned in JSON format, we recommend that you consult a specific documentation.



PART 6

Communication

10

DEVELOP 10 TIMES FASTER



PC SOFT

1. THE COMMUNICATION

1.1 Communication with WinDev/WebDev

The WLanguage proposes several communication methods.

Ways of communication	Features supplied by WinDev/WebDev	For more details ...
• emails	Sending and receiving emails.	See "Communicate by emails", page 243.
• Lotus Notes	Access to the data handled by Lotus Notes.	See "Accessing Lotus Notes and Outlook", page 252.
• Outlook	Access to the data handled by Outlook.	See "Accessing Lotus Notes and Outlook", page 252.
• phone	Automated use of the phone.	See "WinDev and telephony", page 270.
• fax	Send and receive faxes.	See "Sending faxes", page 277.
• Internet	Retrieve HTML pages via the HTTP functions.	See "Retrieving the HTML pages", page 280.
• data transfer	Send and receive data via the FTP/RPC functions of WinDev.	See "Managing files on Internet", page 281.
• file transfer	Handling files on an FTP server.	See "Communicating with an FTP server", page 284.
• sockets	Communication between applications and sites via the network with the Socket functions.	See "Managing the sockets", page 286.
• Bluetooth and OBEX	Communication between a PC and a Mobile.	See "Managing the Bluetooth keys", page 291.
• multitask management	Running several tasks in parallel in the same application or in the same site.	See "Managing the threads", page 293.
• SOAP server	Create SOAP client and server applications.	See "SOAP", page 301.
• procedures on a server of .Net Web services	Managing the execution of procedure on a server of .Net Web services.	See "XML Webservices", page 305.
• XML Web services	Importing and/or generating .Net and J2EE XML Web services.	See "XML Webservices", page 305.
• XML Files	Managing files in XML format.	See "XML", page 311.
• .NET assemblies	Creating and using .NET assemblies.	See ".NET assemblies", page 315.

1.2 Communication with WinDev Mobile

WinDev Mobile can be used to establish a communication between two Pocket PCs, a Pocket PC and a Smartphone, a Pocket PC and a PC, ...

These "dialogs" are performed via infrared, Wi-Fi, GPRS.

The table below presents the communication modes available for each feature proposed by WinDev Mobile:

	Remote access (RPC on HFSQL)	Email	FTP	HTTP	Telephony	SOAP J2EE .NET	Socket	SMS
ActiveSync	X	X	X	X		X	X	
Network card	X	X	X	X		X	X	
GPRS	X	X	X	X		X	X	
Infrared							X	
Smartphone or phone access (GSM type)					X			X
Wi-Fi	X	X	X	X		X	X	

2. COMMUNICATE BY EMAILS

2.1 Overview

WinDev and WebDev enable you to directly manage the emails from your applications. The emails can be easily sent and received via:

- several WLanguage functions,
- an email structure allowing you to find out and store the characteristics of the email used. You also have the ability to handle the content of the emails without sending them (EmailBuildSource and EmailImportSource).

- the management of advanced types used to easily handle several connections to servers and an important number of messages:

- The Email and EmailAttach types are used to manage the messages and their attachments.
- The EmailPOP3Session, EmailSMTPSession, EmailNotesSession, EmailOutlookSession and EmailIMAPSession types are used to manage the connections with the messaging server.

2.2 Managing the emails

Several methods can be used to manage the emails:

- The POP3/SMTP protocol (most common method): this protocol is an email management protocol recognized by all the service providers. This protocol enables you to directly communicate with the server, available at your ISP.
- The "Simple Mail API (also called SMAPI or Simple MAPI)": this email management protocol is

used by most Microsoft applications, and specifically MS Exchange 4.

- **The Lotus Notes or Outlook messaging software:** these programs allow you to send and receive emails.
- **The IMAP protocol:** this protocol is now available via to the new EmailIMAPSession type.

2.3 Synchronous/Asynchronous mode (WebDev only)

The email functions are locking functions by default. Which means that no other code can be run during their execution. The program will resume only when the current email functions have been run.

WebDev gives you the ability to manage the emails in asynchronous mode. This mode allows your sites to send emails without locking the other processes.

To use the asynchronous mode, you must:

1. Uncheck "Disable the email spooler" in the WebDev administrator ("Configuration" tab).
2. Enable the asynchronous mode when starting the SMTP session (with mailStartSMTPSession or EmailStartSession).
3. All the outgoing emails will be transmitted to a "spooler". The emails are queued up before they are sent.

The execution of the Email functions do not lock the rest of your program anymore. EmailStatus is used find out the status of an email.

Note:

If the WebDev administrator is closed, the email spooler is cleared: the pending emails are not sent and they are removed from the spooler.

If "Disable the email spooler" is checked while some emails are still found in the spooler, these emails will not be lost: the administrator continues to send them but no new email will be accepted by the spooler.

Caution: The asynchronous mode can be only used when starting a session on an SMTP server (**EmailStartSMTPSession** for sending emails). The asynchronous mode is ignored in all the other cases.

2.4 Manage the emails via the POP3/SMTP protocols

2.4.1 Overview of the POP3/SMTP protocols

The POP3 and SMTP protocols are protocols for email management recognized by all the service providers. These protocols allow you to directly communicate with the email server, available at your service provider.

- The POP3 protocol is used to receive emails.
- The SMTP protocol is used to send emails

Note: the reception of emails can also be performed by using the IMAP protocol.

2.4.2 Principle

Sending messages by using the SMTP protocol.

To send messages by using the SMTP protocol, you must:

1. Start a SMTP session:
 - by using an `EmailSMTPSession` variable and ***EmailStartSession***.
 - by using ***EmailStartSMTPSession***.
2. Build the message to send in an Email variable or in the Email structure.
3. Send the message with ***EmailSendMessage***.
4. Close the SMTP session with ***EmailCloseSession***.

Receiving messages by using the POP3 protocol.

To receive some emails by using the POP3 protocol, you must:

1. Start a POP3 session:
 - by using an `EmailPOP3Session` variable and ***EmailStartSession***
 - by using ***EmailStartPOP3Session***.

Example for starting a session by using an `EmailSMTPSession` variable:

```
// Start the session
// for messaging
MySession is EmailPOP3Session
MySession.ServerAddress = .....
    "pop.mycompany.com"
MySession.Name = "user"
MySession.Password = "secret"
EmailStartSession(MySession)
```

2. Read the messages on the messaging server:

- with ***EmailGetAll***,
- with a loop such as:

```
Example that uses an Email variable :
MyMessage is Email
EmailReadFirst(MySession, ...
    MyMessage)
WHILE NOT MyMessage.Out
// Place the process
// of the message read
...
EmailReadNext(MySession, ...
    MyMessage)
END
```

```
Example that uses the Email structure :
EmailReadFirst(MySession)
WHILE NOT Email.Out
// Place the process
// of the message read
...
EmailReadNext(MySession)
END
```

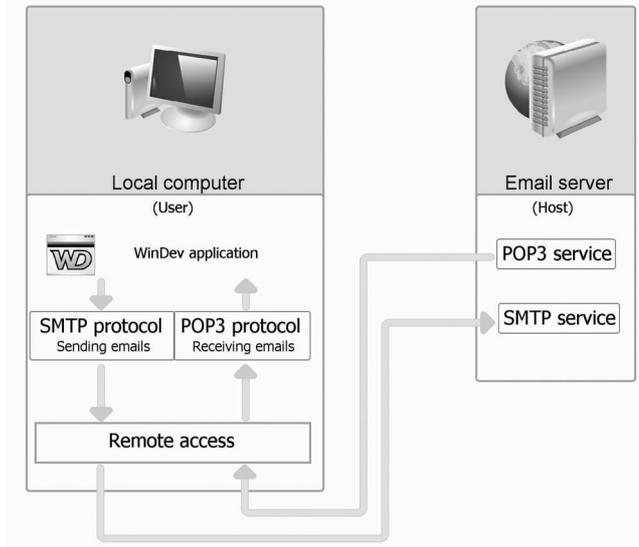
- or with a loop such as:

```
MyMessage is Email
FOR EACH MyMessage OF MySession
// Place the process
// of the message read
...
END
```

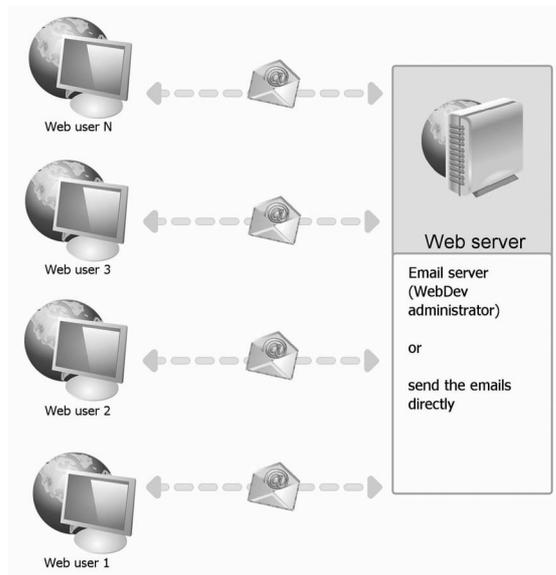
3. Close the POP3 session with ***EmailCloseSession***.

Note: A POP3 session and an SMTP session can be opened at the same time by ***EmailStartSession***.

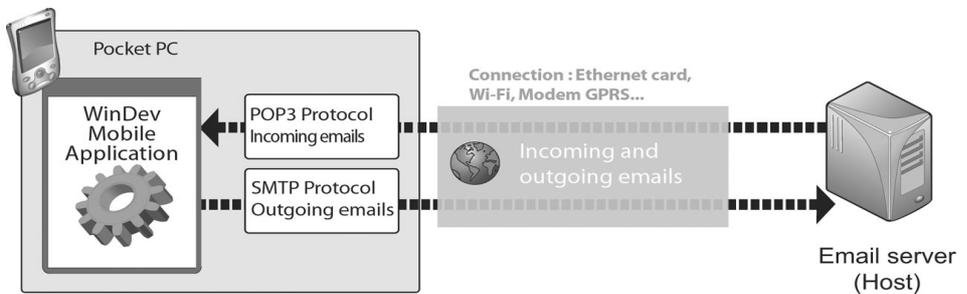
Principle and specific features in WinDev



Principle for a WebDev site:



Principle for a Mobile application:



2.5 Manage the emails with the IMAP protocol

2.5.1 Overview of the IMAP protocol

The IMAP protocol is a standard protocol for email management. Unlike the POP3 protocol, the principle for using the IMAP protocol consists in leaving the messages on the messaging server in order to read them from several clients. The IMAP protocol includes a specific indicator in order to find out whether a message was read. Several messaging servers support both the IMAP protocol and the POP3 protocol.

2.5.2 Using the IMAP protocol

To receive emails by using the IMAP protocol:

1. Open an IAMP session by using an EmailIMAPSession variable and **EmailStartSession**.

Example for opening a session by using an EmailIMAPSession variable:

```
// Start the session ...
// for messaging
MySession is EmailIMAPSession
MySession.ServerAddress = ...
    "pop.mycompany.com"
MySession.Name = "user"
MySession.Password = "secret"
EmailStartSession(MySession)
```

2. Read the messages on the messaging server:

- with **EmailGetAll**,

- with a loop such as:

```
Example that uses an Email variable:
MyMessage is Email
EmailReadFirst(MySession, ...
    MyMessage)
WHILE NOT MyMessage.Out
    // Place the process
    // of the message read...
    ...
EmailReadNext(MySession, ...
    MyMessage)
END
```

```
Example that uses the Email structure:
```

```
EmailReadFirst(MySession)
WHILE NOT Email.Out
    // Place the process
    // of the message read
    ...
EmailReadNext(MySession)
END
```

- or with a loop such as (the "NOT READ" keywords are optional and they allow you to browse only the unread messages or all the messages):

```
MyMessage is Email
FOR EACH MyMessage UNREAD OF ...
    MySession
    // Place the process
    // of the message read
    ...
END
```

3. Close the IMAP session with **EmailCloseSession**.

2.6 Managing emails with "Simple MAPI" (WinDev and WebDev)

Simple MAPI (also called "Simple Mail API") is an API for email management used by the Microsoft applications and specifically by MS Exchange 4.

Simple MAPI simplifies the management of emails received at the hosting company. When an email is read, it is automatically loaded in the local inbox and deleted from the server (at the host).

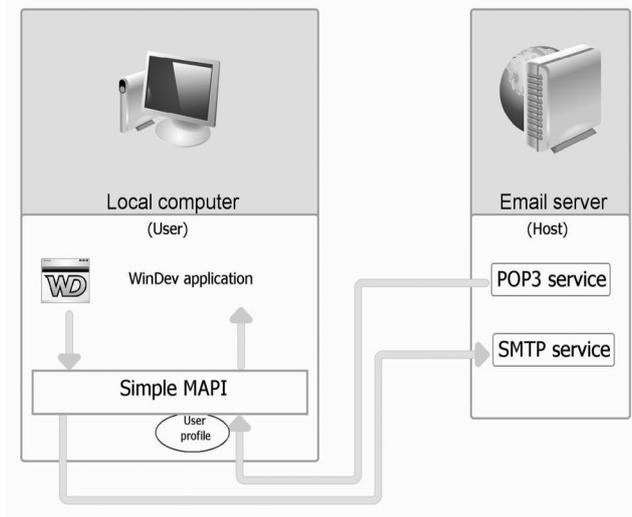
All the characteristics required to manage the emails (POP3 protocol, SMTP protocol, remote access, and so on) are grouped in the "User Profile".

With the email functions of WLanguage, an application or a site can directly handle the emails managed in a application or in a site that uses "Simple MAPI".

2.6.1 Principle

To send or read messages via **Simple MAPI**, you must:

1. Describe a user profile. This user profile must be created in the Microsoft application for email management (MS Exchange for example).
2. From the application or from the site, connect to the application for email management (MS Exchange 4 for example) with **EmailStartSession**.
3. Send and read the messages.
4. Close the session with the application for email management (MS Exchange 4 for example) with **EmailCloseSession**.



2.6.2 Step 1: Creating a user profile

The user profile is used to configure the application for email management (MS Exchange 4 for example).

The following elements are defined in the user profile:

- the SMTP protocol used,
- the POP3 protocol used,
- the different communication services used. To use the "email" functions of WLanguage, the user profile must use the communication service **Microsoft Exchange Server**.

Note: You must create as many profiles on the computer as the number of users or email accounts. The name of the profile will be used to start the email session with **EmailStartSession**.

To create a profile:

1. Open the control panel.
2. Double-click "Email".
3. Click the "Display the profiles" button.
4. In the window named "Choosing a profile", click the "Add" button.
5. Give a name to the profile. This name will be used in the WinDev programs.
6. Select "Add a new email account".
7. Select the "Microsoft Exchange Server" service.

8. Enter the name of the Microsoft Exchange server.

2.6.3 Step 2: Starting an email session

To start an email session, use **EmailStartSession**. This function must be the first "email" function used in your WinDev application (on in your Web-Dev site).

The function named **EmailStartSession** returns the identifier of the session. This identifier will be used by all the WLanguage functions for email management.

Example: The following procedure is used to open a messaging session from a profile. If an error occurs, the Email.Error variable is used to identify the error.

```
Function StartSession(Profile)
  SessionNum is int
  SessionNum = EmailStartSession(...
    Profile)
  IF SessionNum = 0 THEN
    Error("The session was not"+...
      "started. Error: "+...
      Email.Error)
  END
  RESULT SessionNum
```

2.6.4 Step 3: Sending emails

The following functions are used to send emails via **Simple MAPI**:

- **EmailSendMessage**: this function is used to place the message in the out-box of the application for email management (out-box of MS Exchange 4 for example).
- **EmailUpdate**: this function is used to synchronize the email server and the application for email management; the new incoming emails are automatically transferred into the in-box, the email found in the out-box are sent.

Example: The following code is used to send all the emails found in a memory table ("TOSEND" table) via MS Exchange 4. Each table row corresponds to an email.

For each email, the information found in the memory table is transferred into the email structure and the email is sent. Then, the email server is updated.

```

I is int
FOR I = 1 to TableCount("TOSEND")
// The mail is sent
// to a single person
Email.NbRecipient = 1
Email.Recipient[1] = ...
    ExtractString(TOSEND[I], 1)
// Subject and message
Email.Subject = ...
    ExtractString(TOSEND[I], 2)
Email.Message = ...
    ExtractString(TOSEND[I], 3)
// No attached file
Email.NbAttach = 0
// Send the message to
// MS Exchange 4
EmailSendMessage ...
    (SessionNum, False)
END
// Send the messages from
// MS Exchange 4
// to the email server
IF NOT ...
    EmailUpdate(SessionNum) THEN
        Error("Error. Problem"+...
            Email.Error)
END

```

2.6.5 Step 3.1: Reading the emails

Reading the emails via **Simple MAPI** is performed by:

- **EmailUpdate**: this function is used to synchronize the email server and the software for email

management: the new incoming emails are automatically transferred into the in-box, the messages found in the out-box are sent.

- the functions for reading emails (**EmailReadFirst**, **EmailReadNext**, etc.): these functions are used to initialize the email structure of WLanguage with the characteristics of the email currently read (sender, subject, etc.).

Example: The following code is used to read the emails. The incoming emails are stored in a memory table ("Messages"). The SessionNum variable corresponds to the identifier of the session.

In this example, the incoming messages are deleted from the in-box and from the email server by **EmailDeleteMessage**.

```

// Receives the pending messages
// found on the email server
IF NOT ...
    EmailUpdate(SessionNum) THEN
        Error("Error. Problem"+ ...
            Email.Error)
END
// Read the first unread message
IF NOT EmailReadFirst(...
    SessionNum, "NOT READ") THEN
    Error("Error while "+...
        "reading the first message")
END
// Browse the unread messages
// and display them in
// a memory table
TableDeleteAll("Messages")
WHILE NOT Email.Out
    // The reception date,
    // the sender address
    // and the message are
    // assigned in the table
    TableAdd("Messages", ...
        Email.ReceiveDate + TAB + ...
        Email.SenderAddress + ...
        TAB + Email.Message)
// Delete the message
IF NOT EmailDeleteMessage(...
    SessionNum) THEN
    Error("Error: the message"+...
        "was not deleted")
END
// Read the next unread message
IF NOT EmailReadNext(...
    SessionNum, "NOT READ") THEN
    Error("Error while"+...
        "reading the next message")
END
END

```

2.6.6 Step 4: Ending the email session

When the management of the incoming and/or outgoing emails is completed, the session is closed by **EmailCloseSession**. This function must be the last "email" function used.

Example: The following code is a procedure used to

close a messaging session. In this code, the SessionNum variable corresponds to the identifier of the session returned by EmailStartSession.

```
Procedure CloseSession(SessionNum)
// Close the session
EmailCloseSession(SessionNum)
```

2.7 Manage the emails with CEMAPI (WinDev Mobile only)

CEMAPI is an API for email management used by most of the Pocket applications to send and receive emails (Pocket Outlook in most cases).

CEMAPI simplifies the management of the emails received by the hosting company. When an email is read, it is automatically loaded in the local inbox and deleted from the server (at the host).

All the characteristics required to manage the emails (POP3 protocol, SMTP protocol, remote access, etc.) are grouped in the "User Account".

Via the email functions of WLanguage, a WinDev application can directly handle the emails managed in an application that uses "CEMAPI".

See the online help for more details.

2.8 Reading and writing an email

Reading and writing emails can be done:

- via the email structure (kept for backward compatibility)

- via the EmailXXX variables.

See the online help for more details.

2.9 Functions for managing the emails

The following functions are used to manage the emails:

EmailChangeStatus	Changes the status of an email on a messaging server.
EmailSetTimeOut	Changes the value of the "time-out" for connecting to the SMTP and POP3 messaging servers
EmailLoadAttachment	Adds an attached file to an email.
EmailSeekFirstNotes	Seeks one or more emails according to specified criteria, in a local or remote Lotus Notes or Domino database.
EmailBuildSource	Generates the source code of the outgoing email from the variables currently found in the email structure.
EmailCopy	Copies an email found in a directory to another directory of an IMAP server.
EmailSend	Sends a message via the POP3 protocol while entirely controlling the buffer of the email
EmailSendMessage	Sends a message
EmailStatus	Allows you to find out the status of an email sent during an SMTP session opened in asynchronous mode
EmailCloseSession	Closes the POP3 session
EmailImportHTML	Allows you to send an email in HTML format with embedded images
EmailImportSource	Fills the variables of the email structure from the content of the Email.Source variable.
EmailProgressBar	Manages a progress bar for sending and receiving emails
EmailRunApp	Starts the native application for sending emails found on the Android device
EmailReadLast	Reads the last pending message found on the server
EmailReadLastHeader	Reads the header of the last pending message found on the server (POP3 protocol only)

EmailReadMessageHeader	Reads the header of a message identified by its number (POP3 protocol only)
EmailReadPreviousHeader	Reads the header of the previous pending message (POP3 protocol only)
EmailReadFirstHeader	Reads the header of the first message (POP3 protocol only)
EmailReadNextHeader	Reads the header of the next message (POP3 protocol only)
EmailReadMessage	Reads a message identified by its number
EmailReadPrevious	Reads the previous pending message
EmailReadFirst	Reads the first message
EmailReadNext	Reads the next message
EmailGetTimeOut	Reads the value of the "time-out" for connecting to the SMTP and POP3 messaging servers
EmailUpdate	Sends the messages to the Internet email server and receive messages waiting on the Internet server
EmailMsgError	Returns the message corresponding to the identifier of the error
EmailNbMessage	Returns the number of messages currently found on the server
EmailOpenMail	Opens the default messaging software of the Web user on the browser computer
EmailStartSession	Starts a session for email management
EmailStartPOP3Session	Starts a session for reading emails with the POP3 protocol
EmailStartSMTPSession	Starts a session for sending emails with the SMTP protocol
EmailReset	Reinitializes the variables of the email structure
EmailSaveAttachment	Copies the email attachments onto the local computer
EmailDeleteMessage	Deletes a message
EmailMessageLength	Allows you to find out the size of an email before loading it
EmailCheckAddress	Checks the validity of an email address

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

3. ACCESSING LOTUS NOTES AND OUTLOOK

3.1 Access to Lotus Notes

3.1.1 Overview

WinDev and WinDev allow you to easily access the data handled by Lotus Notes (Domino database). You have the ability to retrieve, modify, add or handle your contacts, appointments, tasks, emails, databases, ...

The data managed by Lotus Notes is organized as follows:

- A database per user containing the emails, the tasks and the appointments of this user.
- A database common to all the users containing the contacts and the groups of contacts.
- Several databases containing documents.

To access Lotus Notes, the Lotus Notes client must be installed on the end-user computers.

The access to Lotus Notes is available for Lotus Notes version 6.5 or later.

3.1.2 Method for accessing Lotus Notes

1. Establish the connection with the Domino server: **NotesOpenConnection**.
2. Start session for accessing emails, tasks, appointments, contacts and/or contact groups: **EmailStartNotesSession** (specifying the connection identifier of the local or remote Louts database to use, syntax 2).
3. Start a session for accessing the documents: **NotesOpenDatabase** (by specifying the identifier of the connection and the local or remote Lotus Notes database to use).

4. Change the Lotus Notes database to use: **NotesCloseDatabase** and **NotesOpenDatabase**.

5. Disconnect from the Domino server: **CloseSession** or **EmailCloseSession**.

3.1.3 Quick method if you do not want to access the documents

If you only want to access the emails, tasks, appointments, contacts and/or contact groups, you can establish the connection with the Domino server and start a session for accessing the Lotus database by directly using **EmailStartNotesSession** (syntax 1), without calling **NotesOpenConnection**.

3.1.4 Handling the data

Once the connection was established with the Domino server and once the Lotus Notes database was specified, you can handle:

- the emails via the Email functions (see page 253).
- the tasks via the the Task functions (see page 255).
- the appointments via the Appointment functions (see page 255).
- the contacts via the Contact functions (see page 256).
- the groups of contacts via the Group functions (see page 256).
- the documents via the Notes functions (see page 257).

3.2 Access to Outlook

3.2.1 Overview

WinDev and WinDev allow you to easily access the data handled by Outlook. You have the ability to retrieve, modify, add or handle your contacts, appointments, tasks, emails, ...

3.2.2 Method for accessing Outlook

1. Starting a session used to access the emails, tasks, appointments, contacts and/or groups of contacts: **EmailStartOutlookSession** or **OutlookStartSession**.
2. Closing the session: **CloseSession** or **EmailCloseSession**.

3.2.3 Sending and receiving emails

To send and receive emails, Outlook must be started on the current computer.

To actually send and receive emails:

- **Automatic sending and receiving:** Select the following options in Outlook:
 - "Send the messages immediately during the connection"
 - "Check the new incoming messages every X minutes" and specify the requested number of minutes.
- **Manual sending and receiving:** click the "Send/Receive" button.

3.2.4 Version of Outlook

The access to Outlook is available for Outlook version 97 or later.

Note: The groups of contacts are not supported by Outlook version 97. Therefore, they cannot be accessed in this version of Outlook.

Caution: In Outlook express, you only have the ability to access the emails. In this case, use **EmailS-**

tartSession> to start the session for accessing the emails.

The access to Outlook is compatible with the MS Exchange servers.

3.2.5 Handling the data

Once the connection was established with the Domino server and once the Lotus Notes database was specified, you can handle:

- the emails via the Email functions (see page 253).
- the tasks via the the Task functions (see page 255).
- the appointments via the Appointment functions (see page 255).
- the contacts via the Contact functions (see page 256).
- the groups of contacts via the Group functions (see page 256).
- the documents via the Notes functions (see page 257).

3.3 Lotus Notes and Outlook functions

3.3.1 Email functions for Lotus Notes

EmailSeekFirstNotes	Seeks one or more emails according to specified criteria, in a local or remote Lotus Notes or Domino database.
EmailBuildSource	Generates the source code of the outgoing email from the variables currently found in the email structure. The source code is generated in the Email.Source variable.
EmailSendMessage	Allows you to send a message.
EmailCloseSession	Closes the session.
EmailImportSource	Fills the variables of the email structure from the content of the Email.Source variable.
EmailProgressBar	Manages a progress bar for sending and receiving emails.
EmailReadLast	Reads the last incoming email.
EmailReadLastHeader	Reads the header of the last incoming email.
EmailReadMessageHeader	Reads the header of an incoming email.
EmailReadPreviousHeader	Reads the header of the email found before the current email.
EmailReadFirstHeader	Reads the header of the first incoming email.
EmailReadNextHeader	Reads the incoming email found after the current email.
EmailReadMessage	Reads an incoming email.
EmailReadPrevious	Reads the email found before the current email.
EmailReadFirst	Reads the first message.

EmailReadNext	Reads the next message.
EmailStartNotesSession	Allows you to access the data handled by the Lotus Notes messaging software (emails, contacts, groups of contacts, tasks, appointments).
EmailReset	Reinitializes the variables of the email structure.
EmailDeleteMessage	Deletes a message.
CloseSession	Closes the session.
EmailCheckAddress	Checks the validity of an email address.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

3.3.2 Email functions for Outlook

EmailAddFolder	Adds a folder into the Outlook messaging software
EmailChangeFolder	Modifies the current folder in the Outlook messaging software
EmailSeekFirst	Finds one or more emails according to the criteria specified in the Outlook messaging
EmailBuildSource	Generates the source code of the outgoing email from the variables currently found in the email structure. The source code is generated in the Email.Source variable.
EmailCurrentFolder	Returns the name of the current folder in the Outlook messaging software
EmailSendMessage	Allows you to send a message
EmailCloseSession	Closes the session
EmailImportSource	Fills the variables of the email structure from the content of the Email.Source variable.
EmailImportHTML	Allows you to send an email in HTML format with embedded images
EmailListFolder	Returns the list of folders found in the Outlook messaging software
EmailReadLast	Reads the last incoming email
EmailReadLastHeader	Reads the header of the last incoming email
EmailReadMessageHeader	Reads the header of an incoming email
EmailReadPreviousHeader	Reads the header of the email found before the current email
EmailReadFirstHeader	Reads the header of the first incoming email
EmailReadNextHeader	Reads the header of the email found after the current email
EmailReadMessage	Reads an incoming email
EmailReadPrevious	Reads the email found before the current email
EmailReadFirst	Reads the first incoming email
EmailReadNext	Reads the incoming email found after the current email
EmailNbMessage	Returns the number of incoming messages currently found
EmailStartOutlookSession	Allows you to access the data handled by the Outlook messaging software
EmailSaveAttachment	Copies the email attachments onto the local computer
EmailRemoveFolder	Deletes a folder from the Outlook messaging software
EmailDeleteMessage	Deletes a message
EmailCheckAddress	Checks the validity of an email address.
CloseSession	Closes the session

OutlookListProfile	Lists the available Outlook profiles.
OutlookStartSession	Allows you to access the data handled by the Outlook messaging (emails, contacts, groups of contacts, tasks, appointments and folders).
OutlookDefaultProfile	Retrieves the default profile defined in Outlook

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

3.3.3 Task functions

The following functions are used to manage the tasks found in the task list of a Lotus Notes or Outlook messaging:

TaskAdd	Adds a task into the task list of a Lotus Notes ou Outlook messaging
TaskLast	Reads the last task found in the task list of a Lotus Notes or Outlook messaging
TaskRead	Reads a task previously read in the task list of a Lotus Notes or Outlook messaging
TaskModify	Modifies the current task in the task list of a Lotus Notes or Outlook messaging
TaskPrevious	Reads the task found before the current task in the task list of a Lotus Notes or Outlook messaging
TaskFirst	Reads the first task found in the task list of a Lotus Notes or Outlook messaging
TaskReset	Reinitializes the Task structure.
TaskNext	Reads the task found after the current task in the task list of a Lotus Notes or Outlook messaging
TaskDelete	Deletes the current task from the task list of a Lotus Notes or Outlook messaging

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

3.3.4 Appointment functions

The following functions are used to manage the appointments found in the calendar of a Lotus Notes or Outlook messaging:

AppointmentDisplay	Displays an appointment in the native application for managing the appointments found on the Android device
AppointmentAdd	Adds an appointment into the calendar of a Lotus Notes or Outlook messaging
AppointmentCreate	Displays the window for appointment creation of the native application for managing appointments found on the Android device
AppointmentLast	Reads the last appointment found in the calendar of a Lotus Notes or Outlook messaging
AppointmentList	Lists the appointments found on the Android device and et corresponding to the specified criteria
AppointmentListCalendar	Lists the calendars available on the Android device
AppointmentRead	Reads an appointment found in the calendar of a Lotus Notes or Outlook messaging
AppointmentModify	Modifies the current appointment found in the calendar of a Lotus Notes or Outlook messaging
AppointmentPrevious	Reads the appointment found before the current appointment in the calendar of a Lotus Notes or Outlook messaging
AppointmentFirst	Reads the first appointment found in the calendar of a Lotus Notes or Outlook messaging
AppointmentReset	Reinitializes the Appointment structure

AppointmentNext	Reads the appointment found after the current appointment in the calendar of a Lotus Notes or Outlook messaging
AppointmentDelete	Deletes the current appointment from the calendar of a Lotus Notes or Outlook messaging

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

3.3.5 Contact functions

The following functions are used to manage the contacts found in the address book of a Lotus Notes or Outlook messaging:

ContactDisplay	Opens the form of a contact in the native application for managing the contacts of the device (Android phone).
ContactAdd	Adds a contact into the address book of a Lotus Notes or Outlook messaging
ContactSearch	Searches for a contact in the address book of a Lotus Notes or Outlook messaging
ContactCreate	Displays the window for contact creation of the native application for managing contacts found on the device (Android phone)
ContactLast	Reads the last contact found in the address book of a Lotus Notes or Outlook messaging
ContactEdit	Opens in edit the form of a contact in the native application for managing the contacts found on the device (Android phone)
ContactRead	Reads a contact previously read, found in the address book of a Lotus Notes or Outlook messaging
ContactModify	Modifies the current contact in the address book of a Lotus Notes or Outlook messaging
ContactPrevious	Reads the contact found before the current contact in the address book of a Lotus Notes or Outlook messaging
ContactFirst	Reads the first contact found in the address book of a Lotus Notes or Outlook messaging
ContactReset	Reinitializes the Contact structure
ContactSelect	Displays the list of contacts found on the device (Android phone)
ContactNext	Reads the contact found after the current contact in the address book of a Lotus Notes or Outlook messaging
ContactDelete	Deletes the current contact from the address book of a Lotus Notes or Outlook messaging

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

3.3.6 Group functions

The following functions are used to manage the groups of contacts found in the address book of a Lotus Notes, Domino or Outlook messaging:

GroupAdd	Adds a group of contacts into the address book of a Lotus Notes or Outlook messaging
GroupLast	Reads the last group of contacts found in the address book of a Lotus Notes or Outlook messaging
GroupRead	Reads a group of contacts previously read, found in the address book of a Lotus Notes or Outlook messaging
GroupModify	Modifies the current group of contacts in the address book of a Lotus Notes or Outlook messaging
GroupPrevious	Reads the group of contacts found before the current group in the address book of a Lotus Notes or Outlook messaging

GroupFirst	Reads the first group of contacts found in the address book of a Lotus Notes or Outlook messaging
GroupReset	Reinitializes the Group structure
GroupNext	Reads the group of contacts found after the current group in the address book of a Lotus Notes or Outlook messaging
GroupDelete	Deletes the current group of contacts from the address book of a Lotus Notes or Outlook messaging

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

3.3.7 Notes functions

The following functions are used to manage the Notes documents found in Lotus Notes:

NotesActivateView	Indicates the view that must be handled in Lotus Notes
NotesAddAttachment	Attaches a file to an item of the current document in Lotus Notes
NotesItem	Returns the value of the items for the current document in Lotus Notes
NotesDeactivateView	Disables the view currently handled in Lotus Notes
NotesItemDimension	Returns the dimension of an item for the current document in Lotus Notes
NotesDocumentSeek	Seeks a document in Lotus Notes
NotesDocumentLast	Positions on the last document found in Lotus Notes
NotesDocumentPrevious	Positions on the document found before the current document in Lotus Notes
NotesDocumentFirst	Positions on the first document found in Lotus Notes
NotesDocumentNext	Positions on the document found after the current document in Lotus Notes
NotesDocumentDelete	Deletes the current document from the Lotus Notes database
NotesOut	Allows you to find out whether there is a current document in Lotus Notes
NotesSave	Saves the current document in Lotus Notes
NotesExtractAttachment	Extracts a file attached to an item of the current document in Lotus Notes
NotesCloseDatabase	Closes the local or remote database used (Lotus Notes or Domino)
NotesListItem	Returns the list of items for the current document in Lotus Notes
NotesListAttachment	Returns the list of files attached to an item of the current document in Lotus Notes
NotesListView	Returns the list of views found in the current Lotus Notes database
NotesModifyItem	Modifies the specified item of the current document in Lotus Notes
NotesModifyAttachment	Modifies a file attached to an item of the current document in Lotus Notes
NotesNbAttachment	Returns the number of files attached to an item of the current document in Lotus Notes
NotesOpenDatabase	Gives access to the documents managed by Lotus Notes
NotesOpenConnection	Opens a connection with a local or remote Lotus Notes or Domino database
NotesReset	Creates an empty document in Lotus Notes
NotesDeleteItem	Deletes the specified item from the current document in Lotus Notes
NotesDeleteAttachment	Deletes an attached file from an item of the current document in Lotus Notes
NotesViewSeekLast	Seeks the last document found in the current view in Lotus Notes
NotesViewSeekFirst	Seeks the first document found in the current view in Lotus Notes
NotesViewListColumn	Lists the columns found in the current view in Lotus Notes

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

4. GOOGLE

4.1 Managing the Google contacts

4.1.1 Overview

The Google Contact service is used to manage the contacts on Internet. WinDev and WebDev allow you to create an application for synchronizing the contacts of an existing program with the contacts of Google Contact.

Warning: Before using a feature linked to Google services, we recommend that you check the license for using this service. Some restrictions may apply. The content of licenses may change over time.

PC SOFT is in no case responsible for the way the native access functions are used. Please make sure that you comply with the license of the service provider.

4.1.2 How do I manage the Google contacts?

To manage the Google contacts:

1. Create a Google account if necessary. This account can be created via the following address: <http://code.google.com/intl/en/apis/maps/signup.html>.

Caution: the address of this page may have changed since this page was written.

The Google account is identified by an email address and the associated password.

2. In the code of your application, create a **gglConnection** variable. This variable contains the characteristics of the connection to your Google account.

To create a Google contact:

A Google contact can be created via the Google interface or by programming with the WLanguage functions.

To create a Google contact with the WLanguage functions:

1. Create a **gglContact** variable.
2. Define the characteristics of the contact with the **gglContact** properties
3. Validate the creation of the contact with **GglWrite**

Note:

If a proxy is used to access Internet, the proxy must be configured (**Proxy**) to use the Google functions.

4.1.3 How do I retrieve a Google contact?

To retrieve a Google contact:

1. Declare an array of **gglContact** variables.
2. Use **GglListContact**. This function is used to list the contacts. The contacts found are assigned to the array of **gglContact** variables.

Example:

```
// Retrieves all the contacts
ArrContacts is array of .....
    0 GglContact
ArrContacts = GglListContact(Cnt)
// Browse the contacts
AContact is gglContact
FOR EACH AContact OF arrContacts
    Trace(AContact..Name)
END
```

4.1.4 How do I modify or delete the Google contacts?

Principle:

The principle is straightforward: You must find and position on the contact to modify or delete before performing the requested operation.

To modify a contact:

1. Retrieve the list of contacts.
2. Search for the contact to modify.
3. Modify the characteristics of the contact.
4. Validate the modifications with **GglWrite**.

Example:

```
// Retrieves all the contacts
ArrContacts is array of ...
    0 GglContact
ArrContacts = GglListContact(Cnt)
// Browse the contacts
AContact is gglContact
FOR EACH AContact OF arrContacts
    IF AContact..Name = "MOORE" THEN
        AContact..PostalAddress[1]
            ..Address = "34080 Montpellier"
// Effective update of the
// changes on the server
    GglWrite(Cnt, AContact)
END
END
```

To delete a contact:

1. Retrieve the list of contacts.
2. Find the contact to delete.
3. Delete the contact with **GglDelete**.

Example::

```
// Retrieves all the contacts
ArrContacts is array of ...
    0 GglContact
```

```
ArrContacts = GglListContact(Cnt)
// Browse the contacts
AContact is gglContact
FOR EACH AContact OF arrContacts
    IF AContact..Name = "MOORE" THEN
        // Deletion
        GglDelete(Cnt, AContact)
    END
END
```

4.1.5 Functions for managing the Google contacts

The following functions are used to manage the contacts via the "Google Contacts" service:

GglWrite	Creates or updates a contact or a group of contacts.
GglListContact	Retrieves the list of contacts associated with the specified Google account.
GglListContactByRequest	Retrieves a list of contacts from custom parameters.
GglListContactGroup	Retrieves the list of groups of contacts associated with the specified Google account.
GglRequest	Sends a communication request (HTTP request) to a Google service.
GglDelete	Deletes a contact or a group of contacts.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

4.2 Managing the Google calendars

4.2.1 Overview

The Google Calendar service is an Internet application provided by Google that is used to manage a calendar on Internet.

WinDev and WebDev enable you to create an application for synchronizing the schedules with an existing program for example: using a meeting room, a vehicle, ...

These WLanguage functions also allow you to develop specific interfaces (suited for your business requirements, more user-friendly, ...) and to add specific processes (prints, ...).

Examples of processes that can be implemented natively in WLanguage:

- Retrieving the detailed list of calendars (professional calendars, personal calendars, ...).
- Retrieving the list of appointments from a calendar.
- Performing a search in the appointments of a calendar.
- Adding, modifying and deleting appointments.

Warning: Before using a feature linked to Google services, we recommend that you check the license for using this service. Some restrictions may apply.

The content of licenses may change over time.

PC SOFT is in no case responsible for the way the native access functions are used. Please make sure that you comply with the license of the service provider.

4.2.2 How do I manage a Google calendar?

To manage a Google calendar:

1. Create a Google account if necessary. This account can be created via the following address: <http://code.google.com/intl/en/apis/maps/signup.html>. Caution: the address of this page may have changed since this page was written.

The Google account is identified by an email address and the associated password.

2. In the code of your application, create a **gglConnection** variable. This variable contains the characteristics of the connection to your Google account.

To create a Google calendar:

A Google calendar can be created via the Google interface or by programming with the WLanguage functions.

To create a Google calendar with the WLanguage functions:

1. Create a **gglCalendar** variable.

2. Define the characteristics of the calendar with the *gglCalendar* properties.
3. Define (if necessary) the events linked to the calendar (*gglEvent* variable).
4. Validate the creation of the calendar with *GglWrite*.

Note: if you are using a proxy to access Internet, the proxy must be configured (**Proxy**) in order to use the Google functions.

4.2.3 How do I retrieve a Google calendar and its elements?

1st method: retrieving the list of calendars then their events.

To retrieve a Google calendar from the list of calendars:

1. Declare an array of *gglCalendar* variables (to retrieve several calendars).
2. Use *GglListCalendar*. This function is used to list the available calendars. The calendars found are assigned to the array of *gglCalendar* variables.
3. Use *GglFillCalendar* to retrieve the events. The events can be retrieved from a single calendar or from several calendars. The events to retrieve can be filtered (between two dates for example).

Example:

```

ArrCalendars is array of ...
    0 gglCalendar
ArrCalendars = GglListCalendar(Cnt)
// First calendar
Calendar is gglCalendar
    Calendar = ArrCalendars[1]
// Retrieve the events between
// 01/01/2008 and 01/01/2009
// included
GglFillCalendar(Cnt, Calendar, ...
    "20080101", "20090102")
// Browse the events of a
// calendar
Evt is gglEvent
FOR EACH Evt OF Calendar
    Trace(Evt..Title)
END
    
```

2nd method: retrieving a specific calendar.

To retrieve a specific Google calendar as well as its events:

1. Declare a *gglCalendar* variable.
2. Use *GglGetCalendar*. This function is used to retrieve the Google calendar (and its events) corresponding to the specified title.

Example:

```

// Retrieve the calendar
// named "Work"
Calendar is gglCalendar
    Calendar = GglGetCalendar(Cnt,...
        "Work")
// Browse the events of
// the calendar
IF NOT ErrorOccurred THEN
    Evt is gglEvent
    FOR EACH Evt of gglCalendar
        Trace(Evt..Title)
    END
END
    
```

4.2.4 How do I add, modify or delete events in a Google calendar?

Principle:

The principle for modifying the events is straightforward: the calendar is retrieved locally, the modifications are performed locally and the calendar is updated on the server.

Note: For the shared calendars, we recommend that you regularly update the calendars on the server

To add events to a calendar:

1. Retrieve the requested calendar (and its events if necessary).
2. Declare a *gglEvent* variable.
3. Define the characteristics of the event via the properties of the variable.
4. Use *GglWrite* to update the calendar on the server

Example:

```
// Retrieve the calendar
// named "Work"
Calendar is gglCalendar = ...
  GglGetCalendar(Cnt, "Work")
// Create an event
MyEvent is gglEvent
MyEvent.StartDate = ...
  "20081201085000"
MyEvent.EndDate = ...
  "20081201093000"
MyEvent.Title = "Appointment"
MyEvent.Content = ...
  "Appointment to discuss"+ ...
  "the status report for November"
// Add the event into
// the calendar
Add(Calendar.Event, ...
  MyEvent)
// Update the calendar on the server
GglWrite(Cnt, Calendar) ArrCalen-
dar ...
  is array of 0 gglCalendar
ArrCalendars = GglListCalendar(Cnt)
// Retrieves the events
// of the first calendar
GglFillCalendar(Cnt, ...
  ArrCalendars[1])
// Create an event
MyEvent is gglEvent
MyEvent.StartDate = ...
  "20081201085000"
MyEvent.EndDate = ...
```

```
"20081201093000"
MyEvent.Title = "Appointment"
MyEvent.Content = ...
  "Appointment to discuss the"+...
  "status report for November"
// Add the event into
// the calendar
Add(ArrCalendars[1].Event,...
  MyEvent)
// Update the calendar
// on the server
GglWrite(Cnt, ArrCalendars[1])
```

To delete an event from a calendar:

1. Retrieve the requested calendar and its events.
2. Search for the event to delete.
3. Delete the event.
4. Validate the modifications with **GglWrite**.

Note: Several deletions can be performed before using **GglWrite**.

Example:

```
// Retrieve the calendar
// named "Work"
Calendar is gglCalendar = ...
  GglGetCalendar(Cnt, "Work")
// Delete the second
// calendar event
Delete(Calendar.Event, 2)
// Effective update
// of the changes on the server
GglWrite(Cnt, Calendar)
```

4.2.5 Functions for managing the Google calendars

The following functions are used to manage the calendars via the Google Calendar service:

GglWrite	Creates or updates a calendar.
GglListCalendar	Retrieves the list of Google calendars associated with the specified Google account.
GglGetCalendar	Retrieves a Google calendar and its events according to its title or identifier.
GglFillCalendar	Fills a calendar that was previously retrieved: the events corresponding to the calendar are retrieved (by using criteria if necessary).
GglFillCalendarBy-Request	Fills a calendar that was previously retrieved: the events corresponding to the calendar are retrieved via a custom query.
GglRequest	Sends a communication request (HTTP request) to a Google service.
GglDelete	Deletes a calendar.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

4.3 Using the service for managing the Google Picasa photo albums

4.3.1 Overview

Picasa Albums Web is an application for managing the online images and photos proposed by Google.

Examples of processes that can be performed in WLanguage:

- Retrieve the list of albums as well as their details.
- Retrieve the list of photos found in an album, the tags of a photo, ...
- Sending and retrieving photos, ...
- Retrieve and include the photos found in the Google Picasa accounts.

Warning: Before using a feature linked to a Google service, we recommend that you check the license for using this service. Some restrictions may apply. The content of licenses may change over time.

PC SOFT is in no case responsible for the way the native access functions are used. Please make sure that you comply with the license of the service provider.

4.3.2 How do I proceed?

1. Create a **gglAlbum** variable. This variable contains the characteristics of the album.
2. For each photo that must be included in the album, use a **gglPhoto** variable to describe the characteristics of the photo. You also have the ability to describe the tags and the comments associated with the photo.
3. Add the photo to the album (each photo is an element of the PhotoData array found in the **gglAlbum** variable).
4. Use **GglWrite** to create the photo album on the Google server

Example :

```
MyAlbum is gglAlbum
ATag is gglTag
APhoto is gglPhoto
AComment is gglComment
// Create the album
MyAlbum.Title = "Summer holidays"
MyAlbum.Description = ...
    "Summer vacation in ...
    Germany with the family"
MyAlbum.Timestamp = ...
    "UTC Date and Time"
MyAlbum.CommentingEnabled =
// Photo 1
APhoto.CommentingEnabled = ...
TrueAPhoto.FileName = "Test"
APhoto.Image = CompleteDir(...
```

```
fExeDir()) + CST_ImageDir + ...
    "US.jpg"
APhoto.Legend = ...
    "Tower near the river."
// Add tags
ATag.Text = "tower"
Add(APhoto.Tag, ATag)
ATag.Text = "United States of Ame-
rica"
Add(APhoto.Tag, ATag)
ATag.Text = "By night"
Add(APhoto.Tag, ATag)

// Add comments
AComment.Text = "Great picture !"
Add(APhoto.Comment, AComment)
    AComment.Text = ...
    "Touched up photo !!! ..."
    "Add(APhoto.Comment, AComment)
AComment.Text = ...
    "How did you do it ?"
Add(APhoto.Comment, AComment)
// Add the photo 1 into the album
Add(MyAlbum.PhotoData, ...
    APhoto)
// Photo 2
APhoto = new gglPhoto ..
// Used to reinitialize
APhoto.Image = CompleteDir(...
    fExeDir()) + CST_ImageDir+ ...
    "puzzle.png"
Add(MyAlbum.PhotoData, ...
    APhoto)
// Photo 3
APhoto = new gglPhoto
APhoto.CommentingEnabled = False
APhoto.FileName = ...
    "Tropical fish"
APhoto.Image = CompleteDir( ...
    fExeDir()) + CST_ImageDir + ...
    "tropical.jpg"
Add(MyAlbum.PhotoData, ...
    APhoto)
// Photo 4
APhoto = new gglPhoto
APhoto.FileName = "Drawing"
dStartDrawing(IMG_Map)
dCircle(20,20,50,50,PastelGreen,...
    LightRed)
APhoto.Image = ...
    dSaveImageJPEG(IMG_Map,inMemory)
IMG_Map = ""
Add(MyAlbum.PhotoData, APhoto)
// Create the album
```

```
IF NOT GglWrite(sConnection,...
  MyAlbum) THEN
  Error("Error while creating"+...
    "the album: " + ErrorInfo)
END
```

To retrieve and modify a Picasa album:

1. Retrieve the requested Picasa album. You have the ability to use:

- **GglListAlbum** to find out the list of available albums.
- the function named **GglGetAlbum** to retrieve a specific album. Several options allow you to specify the elements that will be retrieved (thumbnails, tags, covers, ?). The images of the photos will not be retrieved.
- the function named **GglFillAlbum** to retrieve the characteristics of an album (an album listed by **GglListAlbum** for example). Several options allow you to specify the elements that will be retrieved (thumbnails, tags, covers). The images of the photos will not be retrieved.

2. To retrieve the photos from the album, browse the photos found in the album (..PhotoData property of the **gglAlbum** variable) and use **GglFill-**

Photo. Different options allow you to specify the elements that will be retrieved.

3. To add a photo into the album:

- describe the photo via a **gglPhoto** variable.
- add the photo to the album (each photo is an element of the **PhotoData** array found in the **gglAlbum** variable).
- use **GglWrite** on the **gglAlbum** variable to update the album on the Google server.

4. To modify a photo of the album:

- modify the characteristics of the photo in the album (via a **gglPhoto** variable if necessary).
- use **GglWrite** on the **gglPhoto** variable corresponding to the photo to modify to update the album on the Google server.

Limitations

- No image in PNG format can be added from a WinDev Mobile application.
- The Google Picasa functions use the APIs supplied by Google. Some features may operate on the online service and may be temporarily blocked by Google via the APIs.

4.3.3 Functions for managing the Picasa albums

The following functions are used to manage the Google Picasa photo albums:

GglWrite	Creates or updates a Google Picasa album or photo.
GglListAlbum	Retrieves the list of "Google Picasa" albums available for the user.
GglListComment	Retrieves: <ul style="list-style-type: none"> • the list of comments saved for the Google client account. • the list of comments associated with a photo found in a Google Picasa album.
GglListPhoto	Lists the photos found in the Google Picasa albums corresponding to specific criteria.
GglListTag	Retrieves: <ul style="list-style-type: none"> • the list of tags associated with a Google client account. • the list of tags associated with a Google Picasa album • the list of tags associated with a photo found in a Google Picasa album
GglGetAlbum	Retrieves a Google Picasa album.
GglFillAlbum	Retrieves, from a Google Picasa album, the data corresponding to the requested options.
GglFillPhoto	Retrieves, from a Google Picasa album, the data about the photos corresponding to the requested options.
GglDelete	Deletes a Google Picasa album or photo.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

4.4 Managing the Google documents

4.4.1 Overview

Google Docs and Google Spreadsheet are the online word processing and the online spreadsheet proposed by Google.

WinDev and WebDev allow you to manage these documents via several WLanguage functions. You have the ability to:

- Upload documents on the Google server.
- Automatically translate the documents during the upload.
- Delete documents.
- List the documents and perform a search in the documents.
- Retrieve documents.

Warning: Before using a feature linked to Google services, we recommend that you check the license for using this service. Some restrictions may apply. The content of licenses may change over time.

PC SOFT is in no case responsible for the way the native access functions are used. Please make sure that you comply with the license of the service provider.

4.4.2 How do I manage the Google documents?

To manage the Google documents found on a Google server:

1. Create a Google account if necessary. This account can be created via the following address: <http://code.google.com/intl/en/apis/maps/signup.html>.

Caution: the address of this page may have been modified since this page was written.

The Google account is identified by an email address and the associated password.

2. In the code of your application, create a **ggIConnection** variable. This variable contains the characteristics of the connection to your Google account.

3. To manage the documents found on the Google server, use one of the following functions:

GgIUploadDocument	Sends a document to the Google server.
GgIPrintDocument	Prints a document (text or PDF file, images, Word or Excel document, ...) by using the Cloud Print service of Google
GgIListDocument	Retrieves the list of documents available on the Google server for the current user
GgIGetDocument	Downloads a document from the Google Docs service.
GgIDelete	Deletes a Google document from the server

If a proxy is used to access Internet, the proxy must be configured (**Proxy**) to use the Google functions.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

4.5 Using the Google Maps service

4.5.1 Overview

The Google Maps service is an Internet cartography software proposed by Google. It is used to display a map by proposing several viewing features.

Note: The mapping service proposed is Google Static Maps.

Examples of processes that can be performed in WLanguage:

- Retrieving a map, displaying it in a window control or in a report control and sending it by email.
- Defining the map area and the size of the image to retrieve by specifying the latitude, the longi-

tude, the zoom factor, ...

- Drawing an itinerary on the map (point by point): a route can be retrieved by using a GPS device that saves the path.

Warning: Before using a feature linked to Google services, we recommend that you check the license for using this service. Some restrictions may apply. The content of licenses may change over time.

PC SOFT is in no case responsible for the way the native access functions are used. Please make sure that you comply with the license of the service provider.

4.5.2 How do I proceed?

Retrieve a map in your applications or sites

To include a map that uses the Google Maps service in your applications or in your sites:

1. Generate a Google key. This key is supplied by Google. This key can be generated via the following address: <http://code.google.com/intl/en/apis/maps/signup.html>.

Caution: the address of this page may have changed since this page was written.

2. Create an Image control in your WinDev or WebDev project. This control will be used to display the requested map.

3. In the code used to retrieve the map, assign the result of **GglGetMap** to the Image control. This function expects the following parameters:

- the Google key
- the latitude and longitude of the central point of the map
- the zoom performed
- the size of the image to retrieve. The maximum size is set to 640 x 640. This limit is defined by the Google Maps service at the date of publication of this page.
- the type of requested map (road map, satellite, ...).

By default, this map has no border, it is in GIF format and it contains no marker. The language displayed on the map corresponds to the language of the country displayed.

Retrieve a map with advanced setting (markers, ...)

To include an advanced map that uses the Google Maps service in your applications or in your sites:

1. Generate a Google key. This key is supplied by Google. This key can be generated via the following address: <http://code.google.com/intl/en/apis/maps/signup.html>.

2. Create an Image control in your WinDev or WebDev project. This control will be used to display the requested map.

3. In the code used to retrieve the map:

- Create a **gglMapParameter** variable. This variable will allow you to define all the map's settings: presence of a frame, display of a path, format of the received image, markers, ... Specify the requested characteristics only.
- Assign the result of **GglGetMap** to the Image control. In <Advanced Parameters>, specify the name of the **gglMapParameter** variable containing the requested options.

Notes:

- No image is returned if the limitations defined by Google are exceeded (number of points in a path, number of markers, size of the image, ..). In this case, we advise you to modify the parameters of the map by reducing the number of elements to display.
- If a proxy is used to access Internet, the proxy must be configured (**Proxy**) to use the Google functions.

4.5.3 Other services

The Google Maps service can also be used to retrieve the coordinates of an address. This feature can be used via the **WLanguage** function named **GglAddressToCoordinates**.

This function returns the latitude and longitude of the address specified in a **gglCoordinate** variable.

4.5.4 Functions for managing the Google maps

The following functions are used to manage maps via the "Google Maps" service:

GglAddressToCoordinates	Retrieves the coordinates (latitude/longitude) of an address.
GglGetStaticMap	Retrieves the map of a specific location via the Google Maps service.
GglRequest	Sends a communication request (HTTP request) to a Google service.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

5. SALESFORCE

5.1 Use the Salesforce service

5.1.1 Overview

Salesforce is a very popular CRM software. It is commercialized as SaaS (Software as a Service), which means that the application and the data are hosted on server farms by Salesforce.

However, specific requirements are often created by this type of software. The development of additional modules is a strong asset in order to use and to customize this type of software.

The WLanguage Salesforce functions allow your applications to natively interface with Salesforce.

Warning: Before using a feature linked to some Salesforce services, we recommend that you check the user license for the service. Some restrictions may apply. The content of licenses may change over time.

PC SOFT is in no case responsible for the way the native access functions are used. Please make sure that you comply with the license of the service provider.

5.1.2 How do I proceed?

Connecting to Salesforce

To use a WLanguage Salesforce function, you must connect to the Salesforce platform. This connection is established by **SFConnect** and by a **sfConnection** variable.

Using the Salesforce features.

The WLanguage Salesforce functions are used to: read, modify and add records in the files of the Salesforce databases. These files can be preset files or custom files.

Using the Salesforce features.

The WLanguage Salesforce functions are used to:

- read, modify and add records in the files of the Salesforce databases. These files can be preset files or custom files.
- merge records.
- Convert "Lead" into account, contact or opportunity.
- request the execution of a specific process. You can submit records to the approval process or perform an approval action.

5.2 Salesforce functions

The following functions are used to manage the Salesforce databases:

SFAdd	Creates a record in a file of a Salesforce database.
SFForwardQuery	Continues to run a SOQL query (Salesforce Object Query Language) on the records of a Salesforce database
SFConnect	Allows you to authenticate yourself toward a Salesforce platform.
SFConvertLead	Converts the leads. These leads can be converted into account, contact or opportunity.
SFDisconnect	Disconnects the Salesforce platform, the current Salesforce session or the specified Salesforce sessions.
SFExecuteQuery	Runs a SOQL query (Salesforce Object Query Language) on a Salesforce database.
SFExecuteProcess	Requests the execution of a specific process. You can submit records to the approval process or perform an approval action.
SFMerge	Merges the Salesforce records (also called Salesforce objects).
SFRead	Retrieves the Salesforce records from their identifiers.
SFModify	Modifies the records of a file defined in Salesforce.

SFSearch	Performs a search among the records found in a Salesforce database. The search is performed via SOSL (Salesforce Object Search Language).
SFSeekModified	Seeks and retrieves the records (Salesforce objects) modified during the specified period.
SFSeekDeleted	Seeks and retrieves the records (Salesforce objects) deleted during the specified period.
SFUndelete	Restores the Salesforce records that were previously deleted.
SFDelete	Deletes records from a Salesforce database.
SFEmptyRecycleBin	Clears the specified records from the Salesforce recycle bin.

6. RSS STREAM

6.1 Overview

WinDev, WebDev and WinDev Mobile allow you to produce and/or consume a RSS stream (Rapid Simple Syndication).

A RSS stream is used to produce (make available) a data stream in standard format. This data stream

can be read (consumed) by an application.

A reader of RSS stream is used to display one or more RSS streams. WinDev is supplied with an example for using a RSS stream reader: WD RSS Reader.

6.2 How do I proceed?

To read a RSS stream:

1. Declare a ***rssStream*** variable. This variable will contain the information regarding the RSS stream to handle.

2. Load the RSS stream with `rssInitialize`. The ***RSS Stream*** variable is initialized with the specified stream.

```
MyRSS is rssStream
MyRSS = rssInitialize(...
    "http://blogs.webdev.info/
    rss.awp?blog=technicalsupport",...
    fromURL)
```

3. Check whether the channel of the stream is valid and load it. For example:

```
IF MyRSSStream.Channel..Occurrence
    >= 1
    THEN
        MyChannel is rssChannel
        MyChannel = MyRSSStream.Channel[1]
    ...
```

4. The content of the RSS stream can be displayed in a table linked by data binding to the ***rssChannel*** variable containing the entries of the stream (in our example, `MyChannel.Entry`).

Note: We recommend that you use:

- FOR EACH loops to read the entries of a channel associated with a stream.
- manual loops that use the ***..Occurrence*** property (can be used on the ***rssChannel*** type to find out the number of channels and on the `rssEntry` type to find out the number of entries).

Databinding and RSS stream

The Databinding is available for the `rssXXX` variables so that the RSS information can be displayed without programming.

6.3 Functions for managing the RSS streams

The following functions are used to manage the RSS streams:

rssDisplay	Builds a RSS stream and displays the result on the browser of the Web user.
rssBuildString	Builds the RSS stream and returns the result (in XML format) in a character string.
rssInitialize	Loads a RSS stream in memory.
rssSave	Builds the RSS stream and saves the RSS stream in an XML file.

7. LDAP SERVER

7.1 Overview

The LDAP protocol (Lightweight Directory Access Protocol) is increasingly used in the business world. This protocol is used to manage the directories in network. This protocol defines the operations for accessing and searching the data, making incompatible systems compatible.

Depending on the information stored in the LDAP directory, you can for example identify a user when he connects to an application, check the user rights for the application, ... Functions for managing a LDAP server.

The LDAP functions are as follows:

LDAPAddAttribute	Adds an attribute or adds a new value to an existing attribute.
LDAPConnect	Used to connect to an LDAP server
LDAPStartAdd	Initializes the addition of a new object into an LDAP server.
LDAPStartModify	Initializes the modification of an existing object in an LDAP server.
LDAPDisconnect	Used to disconnect from an LDAP server.
LDAPListAttribute	Lists the attributes of an object in an LDAP server.
LDAPListChildren	Lists the children of an object in an LDAP server.
LDAPMode	Modifies the operating mode of an LDAP session
LDAPNbValue	Returns the number of values for an attribute.
LDAPReset	Re-initializes the LDAPSession structure used by LDAPConnect.
LDAPFind	Performs a search in an LDAP server. The search is performed recursively.
LDAPReplaceAttribute	Replaces all the values of an attribute by a specific value.
LDAPReplaceAttributeValue	Replaces a specific value of an attribute by a new value.
LDAPRenameAttribute	Renames an attribute. The value of the attributes is kept during this operation.
LDAPDelete	Deletes an object from the LDAP server. For security reasons, the object will be deleted only if it has no child element.
LDAPDeleteAttribute	Deletes an attribute and all its values.
LDAPDeleteAttributeValue	Deletes a value of a specified attribute.
LDAPValue	Returns the value corresponding to the subscript passed in parameter for the specified attribute.
LDAPValidateAdd	Validates the addition of a new object into an LDAP server.
LDAPValidateModify	Validates the modification of an existing object in an LDAP server.

See the online help for more details.

8. WINDEV AND TELEPHONY

WD WDMobile

8.1 Overview

WinDev allows you to easily manage incoming and outgoing phone calls via the telephony functions. These functions are used to manage a voice box, an answerphone, ... in a WinDev application directly.

Required WinDev configuration

To be able to use the telephony features, you must have:

- a modem.

To save and play recordings, this modem must include voice support.

To identify the caller, the modem must support "Caller ID" (caller ID).

- the TAPI 2.0 protocol. This technology can be used on all the systems; however, you must:
 - install the service pack 4 (or later) if the system used is Windows NT4.
 - upgrade the system if the system used is Windows 95 (can be downloaded from <ftp://ftp.microsoft.com/developr/tapi/tapi2195.zip>). Address valid at the time of publication.

Check the configuration

To check the configuration of your computer, we recommend that you use **WDTelDiagno.exe**, supplied with WinDev. This tool is used to list the availa-

ble TAPI lines and to specify the line that will be used by default by the telephony functions. This allows you to quickly detect the conflicts that may occur between the different TAPI lines.

Note: The TAPI lines differ from the standard phone lines.

Before using the telephony functions, you can define by programming the TAPI device on which the calls must be detected or dialed. Use the functions:

tapiCapability	Allows you to find out the features of a TAPI device
tapiDeviceList	Allows you to find out the list of TAPI 2-compatible devices
tapiDevice	Allows you to select the device that will be used by the TAPI functions.

Required WinDev Mobile configuration

To be able to use the telephony functions, the application must be installed:

- on a Pocket PC with phone access (GSM type).
- and/or on a Smartphone.

8.2 Managing the incoming calls

The management of incoming calls is performed in a specific "thread". When an incoming call is detected, the procedure associated with the thread is run. The management of the call is performed in this procedure.

8.2.1 The different steps

To manage the incoming calls in a WinDev application:

1. Define (if necessary) the TAPI device on which the call detection must be performed.

Use the functions:

tapiCapability	Allows you to find out the features of a TAPI device
tapiDeviceList	Allows you to find out the list of TAPI 2-compatible devices
tapiDevice	Allows you to select the device that will be used by the TAPI functions.

2. Detect the incoming calls with **tapiListen**. This function runs a specific WLanguage procedure. This procedure will be automatically run when an incoming call is detected.

3. In this procedure, you can:

- find out the status of the call via the following constants:

tapiLineBusy	The line is currently busy
tapiLineConnected	The line is connected
tapiLineDialing	Dialing in progress
tapiLineDialTone	The line gets a dial tone
tapiLineDisconnected	The correspondent has hung up
tapiLineProceeding	The call is dialed: looking for the person called
tapiLineRingBack	Ringing in progress
tapiNewCall	New call detected waiting for an answer or for a reject.
tapiCallInformation	The additional information (presentation of the number) is available. In most cases, this information will be available after the first ring.

- manage the call via the following functions:
 - finding out the characteristics of the incoming call:

tapiCallStart	Returns the date and time of the beginning of call
tapiCallsOver	Allows you to find out whether the call is over
tapiCallEnd	Returns the date and time of the end of call
tapiCallDuration	Returns the duration of the call
tapiCallerID	Returns the calling phone number

- perform specific operations:

tapiStop	Forces the playing of a pre-recorded message to stop (tapiPlay)
tapiRecord	Saves the current communication in a .WAV file.
tapiPlay	Plays a sound file (.WAV) for the specified line. You have the ability to play the message found on the answerphone for example.

tapiSendKey Allows you to simulate the use of phone keys.

tapiAnswerCall Answers an incoming call (detected beforehand)

tapiKeyPressed Allows you to find out the history of the keys pressed on the phone keypad.

Caution:

This WLanguage procedure being run in a thread, all the constraints of the threads must be complied with (no window opening, no timer management, no event management, ...). See the online help for more details.

We recommend that you limit the number of processes performed in this procedure. Indeed, during the duration of the call, the detection of the other calls (as well as all the telephony events) is frozen. If long processes must be performed, we advise you to process the call in the main thread of the application (see the example below).

4. To end the session for detecting the incoming calls, use **tapiStopCallDetection**.

8.2.2 Example

This example manages the incoming calls in the main thread.

- **Code for declaring the global variables of the window used for call detection.** The different events used to manage the calls in the main thread of the application are declared in this code.

```
GLOBAL
gnEventID is int
//Event to manage
//the incoming calls in popup

IF Event("DetectedCall","*.*",...
    "PhoneCall")=0 THEN
    Error("Unable to manage"+...
        "the popup for call detection",...
        ErrorInfo())
END
IF Event("EndDetectedCall",...
    "*.*","EndPhoneCall")=0
THEN
    Error("Unable to manage"+...
        "the popup for call detection",...
        ErrorInfo())
END
IF Event(...
```

```

        "CallIdentifierDetected",...
        "*.*", "PhoneCallInfo")=0
    THEN
        Error("Unable to manage"+...
        "the popup for call detection",...
        ErrorInfo())
    END

```

- **Initialization code of the window:** this code is used to start the procedure for call detection.

```

// Service for call detection
IF tapiListen("IncomingCall", ...
    tapiOptionMediaModeFax,...
    "CallDetection") THEN
    // The service for detecting the
    // calls was started
    Message("Detection of" + ...
    "calls "+" enabled")
ELSE
    // The service for detecting the
    // calls was not started
    Error("Unable to " + "start"+...
    "the call detection"+CR+...
    "Error details:"+CR+...
    ErrorInfo(errMessage))
END

```

- **Procedure for call detection:** This procedure is used to detect the incoming calls. For each incoming call, the characteristics of the call are sent to the main thread by **PostMessage**.

Caution: The processes performed in this procedure are called from a thread. The management of the display must be performed from the main thread (this is why PostMessage is used).

To debug this type of process, you must use **Trace**.

```

PROCEDURE CallDetection(...
    nIdService,nIdCall,nStatus)
// Detect the incoming calls
SWITCH nStatus
// Detect a new call:
// Note: We will get
// additional information
// after at least one ring
CASE tapiNewCall:
// Signals the arrival of a new
// call to the main window
// to open a popup
PostMessage(Handle(Win_Call),...
    "PhoneCall",nIdCall,nStatus)
// Information about the call
CASE tapiCallInformation:
// Signals the arrival
// of a new call
// to the main window
// to open a popup
PostMessage(Handle(Win_Call),...
    "PhoneCallInfo",nIdCall,nStatus)
// The line has been hung up
CASE tapiLineDisconnected:
// Signals the arrival of a
// new call to the
// main window to open a popup
PostMessage(Handle(Win_Call),...
    "EndPhoneCall",nCallID,nStatus)
END

```

8.3 Managing the outgoing calls

8.3.1 The different steps

To manage the outgoing calls in a WinDev application:

1. Define (if necessary) the TAPI device on which the calls must be dialed. Use the functions:

tapiCapability	Allows you to find out the features of a TAPI device
tapiDeviceList	Allows you to find out the list of TAPI 2-compatible devices
tapiDevice	Allows you to select the device that will be used by the TAPI functions.

2. Dial the telephone number via a modem (**tapiDial**).

3. This function calls a specific WLanguage procedure used to manage the progress of the call. In this procedure, the following variables enable you to find out the status of the line:

tapiLineBusy	The line is currently busy
tapiLineConnected	The line is connected
tapiLineDialing	Dialing in progress
tapiLineDialTone	The line gets a dial tone
tapiLineDisconnected	The correspondent has hung up
tapiLineProceeding	The call is dialed: looking for the person called
tapiLineRingBack	Ringng in progress

4. Pick up the phone receiver to communicate.
5. During the communication, you have the ability to use the WLanguage functions for handling the call (see below). There is no need to specify the identifier of the call because the call processed will be the current call (which means the outgoing call).
6. At the end of the call, close the line with **tapiHangUp**.
7. Hang up the phone receiver.

8.3.2 Handling an outgoing call

The following operations can be performed on the outgoing calls:

- finding out the characteristics of the outgoing call:

tapiCallStart	Returns the date and time of the beginning of call
tapiCallsOver	Allows you to find out whether the call is over
tapiCallsBusy	Allows you to find out whether the number called is busy
tapiCallEnd	Returns the date and time of the end of call

tapiCallDuring	Returns the duration of the call
tapiNoAnswer	Allows you to find out whether an answer was given to the call
tapiCallerID	Returns the calling phone number
tapiCalledID	Returns the phone number called

- perform specific operations:

tapiStop	Forces the reading of a pre-recorded message to stop (tapiPlay)
tapiRecord	Saves the current communication in a .WAV file.
tapiPlay	Plays a sound file (.WAV) for the specified line.
tapiSendKey	Allows you to simulate the use of phone keys.
tapiKeyPressed	Allows you to find out the history of the keys pressed on the phone keypad.

8.4 Telephony functions

These functions enable you to easily handle all the telephony features of a modem from a WinDev application:

tapiCallStart	Returns the date and time of the beginning of call
tapiCallsWaiting	Allows you to find out whether the call is on hold
tapiCallsBusy	Allows you to find out whether the number called is busy
tapiCallsOver	Allows you to find out whether the call is over
tapiCallEnd	Returns the date and time of the end of call
tapiNoAnswer	Allows you to find out whether an answer was given to the call
tapiStop	Forces the playing of a prerecorded message to stop (tapiPlay)
tapiCapability	Allows you to find out the capacity of the TAPI device used
tapiDial	Dials a phone number for a voice line.
tapiLineDial	Dials a phone number for a voice line and chooses the device to use
tapiListen	Starts a service for call detection
tapiDialerDisplay	Opens the default telephony application (dialer) and displays the specified number. No call is made
tapiDialerCall	Opens the default telephony application (dialer) found on the phone and dials the specified number
tapiCallDuring	Returns the duration of the call
tapiRecord	Saves the current communication in a .WAV file.

tapiError	Indicates whether the last tapixxx function has returned an error into the TAPI module
tapiCompleteTransfer	Transfers a call with ability to retrieve the call
tapiStopCallDetection	Stops the specified call detection.
tapiPlay	Plays a sound file (.WAV) for the specified line. You have the ability to play the message found on the answerphone for example.
tapiDeviceList	Lists the TAPI devices installed on the computer
tapiHold	Puts a call on hold
tapiCallerID	Returns the calling phone number
tapiCalledID	Returns the phone number called
tapiSendKey	Allows you to simulate the use of phone keys.
tapiOrigin	Allows you to find out the origin of a call
tapiDevice	Selects the TAPI device that will be used by default
tapiHangUp	Hangs up a phone line that was opened by tapiDial
tapiAnswerCall	Answers an incoming call (detected beforehand)
tapiUnhold	Picks up a call on hold
tapiKeyPressed	Allows you to find out the key currently pressed.
tapiBlindTransfer	Performs a "blind" transfer

See the online help for more details.

9. MANAGING THE SMSs

WDMobile

9.1 Overview

WinDev Mobile allows you to:

- send SMSs.
- browse the incoming SMSs.
- delete one or more incoming SMSs.

An SMS (Short Message Service) is a text message (up to 160 characters) sent on a cell phone.

Required configuration

To be able to use the SMS functions, the application must be installed:

- on a Pocket PC with phone access (GSM type).
- and/or on a Smartphone.

WinDev for Pocket PC can only be used to send SMSs. The SMSs are received as usual by the device used (Pocket PC, Smartphone, cell phone, ...).

9.2 The SMS structure

9.2.1 Overview

The SMS structure is a preset structure of WLanguage (no declaration is required). This structure is used to create an SMS.

Note: To reset all the variables of the SMS structure to zero, use SMSReset.

9.2.2 The variables of the SMS structure

The structure contains the following members:

Receive-Date	Date and time when the SMS was received Note: On Pocket PC 2002, this member contains the date and time when the SMS was read.
Retry	Boolean (True by default) Indicates whether the message must be sent on a regular basis if no reception.
Subscript	Integer corresponding to the subscript of the incoming SMS.
Message	Character string containing the outgoing message or the incoming message (up to 160 characters).
Number	Character string containing the recipient number or the sender number.

**CountryP
refix** Character string containing the international prefix (33 by default for France).

If the recipient number starts with "0" and if a national prefix is specified, "0" will be replaced by this prefix.

If no national prefix is specified, use a number in international format. For example, 33612345678.

**Number-
Type** Indicates the type of number used :

- smsInternationalNumber (default value): these numbers can be accessed anywhere and are in 06.xx.xx.xx.xx. format.
- smsNationalNumber: short numbers, accessible within the country only.

9.2.3 Reading and deleting the SMSs found on Smartphone

To read and/or delete SMSs on a Smartphone (SMSNbMessage, SMSFirst, SMSNext or SMSDelete), the executable of the WinDev Mobile application and its framework (WinDev Mobile libraries) must be digitally signed. A certificate is required to perform this operation.

Note: The use of **SMSSend** and **SMSReset** requires no specific signature.

9.2.4 Different types of numbers

Two types of numbers can be used to send SMSs:

- The short numbers (also called "National" numbers). These numbers can be accessed from the country only.

- The standard numbers (also called "International" numbers, in 06.xx.xx.xx format). These numbers can be accessed from anywhere.

9.2.5 Operating mode in GO mode and at run time

In GO mode (simulation on the development computer), a WLanguage error is generated when using one of the variables of the SMS structure.

9.3 WLanguage functions

These functions are used to easily send SMSs:

SMSSend	Sends an SMS
SMSNbMessage	Returns the number of SMSs received or the maximum number of SMSs that can be received
SMSFirst	Positions on the first SMS received
SMSReset	Re-initializes all the variables of the SMS structure
SMSNext	Positions on the next SMS received
SMSDelete	Deletes the specified SMS

See the online help for more details.

10. SENDING FAXES

WD WebDev

10.1 Overview

Several WLanguage functions can be used to send faxes via the system fax server available in Windows XP and Windows 2000.

How do I send faxes?

To send faxes from an application or a site, you must:

1. Configure the computer from which the faxes must be sent.
2. Create the application or the site for sending faxes.

3. Configure (if necessary) the options of the fax server by programming.

Notes:

- *iPreview/iDestination* can also be used to send a print to a fax.
- In Windows Vista, the management of faxes is available in Windows Vista Professional Edition and Windows Vista Ultimate Edition.

10.2 Configuring the "fax server"

10.2.1 Configuring the current computer

The following elements are required to send a fax:

- Windows XP or 2000,
- a modem configured on the computer,
- the Windows fax service must be started.

WB Caution: The following operations only apply to the server used to send the faxes (and not to the browser computers).

1. Check the configuration of the modem

Open the control panel of Windows ("Start .. Parameters .. Control panel") and select "Phone and modem options".

2. Install a standard fax server

Windows XP and Windows 2000 are supplied with a fax server.

To install this fax server, select "Add/Remove programs" in the control panel of Windows ("Start .. Parameters .. Control panel").

3. Check whether the standard fax service of Windows is in incoming mode.

10.2.2 Configuring the fax server in Windows 2000

In Windows 2000, the fax server is called "Fax Service". To configure the fax server:

1. Open the control panel of Windows ("Start .. Configures .. Control panel").
2. Double-click "Fax Service". If this option is not

displayed, install the fax server of Windows 2000 (see above).

3. Select the "Advanced options" tab and click the "Open the console for managing the fax service" button.

4. The "Device" option is used to list the modems and faxes installed on the computer.

5. Double-click your modem/fax.

6. In the "General" tab, enable the option for sending faxes in order to send faxes from an application or from a site.

7. Validate.

Note: To create a cover page (.Cov), use the "Cover page" tab.

10.2.3 Configuring the fax server in Windows XP

To configure the fax server:

1. Select "Start .. Fax and Printer".

2. Double-click "Fax". The wizard for fax configuration starts.

3. Select "Tools .. Configure".

4. Validate the default parameters until you reach the "Configuration for receiving and sending faxes" plane.

5. Enable the feature for sending faxes in order to send faxes from an application or from a site.

6. Validate. The fax server is started.

Note: To create a cover page (.Cov), select "Tool .. Custom cover page".

10.3 Application or site for sending faxes

10.3.1 Sending a fax from an application or from a site

To send a fax from an application or from a site, you must:

1. Prepare the fax, that is the file that is going to be sent. A specific first page can also be attached to the fax (called "Cover Page"). This cover page is a ".cov" file and it can be created from the fax server.
2. Use **FaxConnect**. This function is used to connect the application or the site to the fax server installed on the current computer.
3. Send the fax:

- with **FaxSend**. When sending the fax, you have the ability to specify the name of a WLanguage procedure. This procedure will be called whenever the status of the fax is modified by the fax server. The status of the fax is returned by **FaxStatus**.
- with **iPreview/iDestination** and the print functions. This solution enables you to directly print a report created with the report editor. In this case, you can give a name to the fax sent. The status of the fax will be returned by **FaxStatus**.

Note: A single fax can be sent at a time. However, **FaxSend** can be run several times in a row: the faxes will be added to the queue of outgoing faxes. The WLanguage procedure combined to **FaxStatus** is used to identify the fax that is currently processed.

4. Once the faxes have been sent, the application or the site must be disconnected from the fax server with **FaxDisconnect**.

ver with **FaxDisconnect**.

10.3.2 Sending a fax created with the report editor

To send a fax created with the report editor:

You can:

- use **iPreview/iDestination** only (syntax 3). The report printed by **iPrintReport** will be directly sent to the specified fax number. In this case, you cannot follow the progress of the fax.

```
// Sends the "CustInvoice" report
// by fax
iDestination(iFax, "0006050402")
iPrintReport (CustInvoice)
```

- use the fax functions and **iPreview/iDestination** (syntax 4). the function named **iPreview/iDestination** establishes (if necessary) a connection to the fax server and returns the identifier of this connection. This identifier can be used by the Fax functions to follow the progress of the send operation.

```
// Sends the "CustInvoice" report
// by fax
ConnectID is int
ConnectID = FaxConnect ()
ConnectID = iDestination(...
    iFax, "0006050402", ...
    "MyFax", IdConnect)
...
iPrintReport (CustInvoice)
```

10.4 Configuring the fax server by programming

10.4.1 Options of the fax server

The standard fax server of Windows proposes several options that can be directly configured from the interface of the fax server. These different options can be configured by programming via the registry.

You have the ability to configure and find out:

- information regarding the sending of faxes (Windows 2000). The corresponding registry key is HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Fax

Number of attempts	Retries
Number of minutes between two attempts	Retry Delay

Number of days before the unsent faxes are deleted	Dirty Days
Archive the outgoing faxes	ArchiveOutgoingFax
Directory of outgoing faxes	ArchiveDirectory
Print the top header	Branding
Forbid the custom cover pages	ServerCoverPageOnly
Time when the economy rate starts	StartCheapTime
Time when the economy rate ends	StopCheapTime



- information regarding the sender of the fax (Windows 2000):

The corresponding registry key is HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Fax\UserInfo:

Number of the fax sender	FaxNumber
Mailbox Address	Mailbox
Company	Company
Title	Title
Full name	NomComple
Department	Department
Business address	Office
Home Phone	HomePhone
Business Phone	OfficePhone
Displays the status monitor when sending AND receiving faxes	VisualNotification
Status monitor always on the top	AlwaysOnTop

Sound Notification	SoundNotification
Billing Code	BillingCode

10.4.2 Tips

To create a fax server and to manage the redial of faxes, the following options must be specified in the fax manager:

- No retry for sending faxes:
RegistrySetValue("HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Fax","Retries",0)
- Time-out set to 0 mn between two attempts:
RegistrySetValue("HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Fax","Retry Delay",0)
- 0 day for keeping the unsent faxes:
RegistrySetValue("HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Fax","Dirty Days",0)

It is possible to assume that a fax whose status changed from **FaxStatusInit** OR **FaxStatusCall** to **FaxStatusInactive** is a fax on which a send attempt was performed. This fax can be resent thereafter (it will not be resent automatically because "No retry for sending faxes" is set to True).

10.5 Functions for managing the faxes

These functions are used send faxes from an application or from a site:

FaxOutbox	Enumerates the pending faxes or the outgoing faxes
FaxInbox	Enumerates the pending faxes or the incoming faxes
FaxConnect	Establishes a connection to a fax server
FaxDisconnect	Closes the connection to a fax server
FaxSend	Sends a fax
FaxStatus	Returns the status of the specified fax
FaxRestart	Restarts a fax in the spooler
FaxResume	Re-enables a fax that was paused in the spooler
FaxDelete	Deletes a fax from the spooler and cancels it
FaxPause	Pauses a fax in the spooler

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

Note: *iPreview iDestination* can also be used to send a print to a fax.

11. RETRIEVING THE HTML PAGES

11.1 Overview

You have the ability to retrieve pages in HTML format. To retrieve the HTML pages, you must:

1. Run a request on the Web server via **HTTPRequest**.

This request can be a GET or POST request.

This request can be run on a standard URL or on a protected URL.

2. Retrieve the result of the request with **HTTPGetResult**.

These functions allow you to retrieve:

- the full content of an HTML page,
- data files,
- images, ...

11.2 HTTP functions

The following functions are used to manage the HTTP requests:

HTTPAddFile	Adds a file into an HTTP form.
HTTPAddParameter	Adds a parameter into an HTTP form.
HTTPCancelForm	Cancels the declaration of an HTTP form and frees all its resources.
HTTPCertificate	Modifies the client certificate used by default to identify oneself on a server.
HTTPCookieWrite	Adds or modifies a cookie.
HTTPCookieManage	Enables or disables the management of cookies during the calls to HTTPRequest
HTTPCookieRead	Returns the value of a cookie received further to an HTTP request.
HTTPCookieReset	Deletes all the cookies (globally or for a single domain) stored by the calls to HTTPRequest .
HTTPCookieGet	Retrieves the cookies read by a call to HTTPRequest for an HTTP domain.
HTTPCookieReplace	Replaces all the cookies stored by HTTPRequest for a domain.
HTTPCreateForm	Creates an HTTP form.
HTTPDestination	Indicates the backup file for the result of the next HTTP request, run in the same thread.
HTTPGetResult	Retrieves the result or the header of the last HTTP request (run by HTTPRequest)
HTTPSendForm	Sends an HTTP form.
HTTPProgressBar	Manages a progress bar when receiving an HTTP request
HTTPListCertificate	Lists the client certificates found on the computer to authenticate on a server.
HTTPParameter	Allows you to configure the functions that use the HTTP protocol.
HTTPResetForm	Clears an HTTP form that is currently edited.
HTTPRequest	Starts an HTTP request on a server
HTTPTimeOut	Defines the time-out for the WLanguage functions that use the HTTP protocol
InternetConnected	Defines whether an Internet connection (by ADSL or modem) is enabled on the current computer
Proxy	Specifies that the communication functions that use the HTTP protocol (HTTP, DotNet, J2EE and SOAP functions) will go via a proxy to run their requests

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

12. MANAGING FILES ON INTERNET

WD

12.1 Overview

Several methods can be used to upload and download files by Internet:

- The WinDev FTP or RPC. These protocols require the use of a specific FTP/RPC server powered by WinDev
- the standard FTP. This protocol requires the use of a standard FTP server. See Voir "Communicating with an FTP server", page 284. for more details.

Note:

- RPC: Remote Procedure Call
- FTP: File Transfer Protocol

12.1.1 Uploading and downloading files via WinDev FTP or RPC: the rules to follow

To upload and download files, you must comply with the following rules:

1. Connect to a WinDev FTP server with *NetConnect*. This function establishes a connection between WinDev and the server and it provides a

connection identifier.

2. Upload, download files.

3. Close the connection to the server with *NetDisconnect*.

Caution:

The TCP/IP communication protocol must be installed and an IP address must be defined.

A WinDev FTP or RPC server operates in 32-bit mode only.

12.1.2 Other features

It also allows you to:

- Find out the name and IP address of a computer.
- Run a program on a WinDev FTP or RPC server.
- Trigger a Windows element on the WinDev FTP or RPC server.
- Transmit a character string to a WinDev FTP or RPC server.

12.2 Detailed use of WinDev FTP/RPC

12.2.1 Step 1: Establishing a connection to a WinDev RPC/FTP server

To transfer files, a connection must be established with a WinDev RPC or FTP server. The connection is established by *NetConnect*. *The source code for establishing a connection must be found before the first "Net" function. The value returned by NetConnect must be stored because it will be used by the other "RPC" and "FTP" functions.*

The code used to connect to a RPC server is as follows:

```
Function RPCConnection (Address,...
    User, Password)
//Connection to a RPC server
ConnectionNum is int
ConnectionNum = NetConnect(...
Address, RPCServer, ...
User, Password)
Result ConnectionNum
```

12.2.2 Step 2: Transmitting a file to a WinDev FTP server

In the following example, a file is transmitted to the WinDev FTP server (*NetSendFile*). A progress bar is used to follow the progress of the transfer.

```
- - Initialization code of the
- - "RPCClient" window
GLOBAL
Transfer_Completed is boolean
Transfer_Completed = False
Transfer_InProgress is boolean
Transfer_InProgress = False
Event ("ProgressBar_Transfer",...
    "RPCClient", "SendFile")
....
hConnect is int
hConnect = NetConnect(...
    "148.61.125.245", FTPServer,...
    "GUEST", "")
- - Button for sending the transfer
IF Transfer_InProgress = True THEN
...
```

```

Error("A transfer of "+...
      "files is currently "+...
      "in progress")
ELSE
  Transfer_Completed = False
  Transfer_InProgress = True
  IF NetSendFile(hConnect,...
                 "C:\autoexec.bat", ...
                 "C:\autoexec.cli", ...
                 "SendFile", 10) = False
THEN
  Info("The transfer failed")
END
...
END
- - Transfer_Progress procedure:
- - managing the current transfer
Transfer_Progress procedure
  Message("Transfer in progress")
  Gauge(_EVE.wParam, _EVE.lParam)
  IF _EVE.wParam = _EVE.lParam THEN
    Transfer_InProgress = False
    Transfer_Completed = True
    Message("Transfer completed")
    Info("Transfer completed")
  END
END

```

12.2.3 Step 3: Retrieving a file from a WinDev FTP server

The file named *NetGetFile* is used to retrieve a file found on an FTP/RPC server WinDev.

Note: you can easily get the list of directories and files found on a WinDev FTP server. An example is available in the description of *NetDirList*.

```

-- Code for opening the window
// Asks for an available message
// to Windows
GLOBAL
WM_MYMESSAGE is int
lpString is fixed string ...
  on 20 lpString= "ProgBar_Main" ...
    hConnect is int
// Connection
hConnect = ...
  NetConnect ("148.61.125.245",...
             FTPServer, "GUEST", "")
WM_MYMESSAGE = ...
CallDLL32("USER32", ...
"RegisterWindowMessageA" lpString)
// Branch the Progress Bar procedure
// on this message
Event("UPDProgBar", "MAIN",...
      WM_MYMESSAGE)

```

```

-- Code of the button for downloading
-- a file
Hourglass(True)
IF NOT NetGetFile(...
  hConnect,...
  "C:\autoexec.bat", ...
  "C:\autoexec.cli", ...
  WM_MYMESSAGE, 10) THEN
  Error("Error while"+...
        "transferring the files")
END
HourGlass(False)
- - Procedure UPDProgBar()
Procedure UPDProgBar()
  // Display the progress bar
  // If the entire file is
  // transferred,
  // reinitialize the progress bar
  IF _EVE.wParam = _EVE.lParam THEN
    // Transfer completed
    Gauge()
  ELSE
    // Transfer in progress
    Gauge(_EVE.wparam, ...
          _EVE.lParam, ...
          "Transfer in progress")
  END
END

```

12.2.4 Step 4: Closing a connection to a WinDev RPC/FTP server

Once the files have been transferred, you must disconnect from the WinDev RPC or FTP server. The disconnection is performed by *NetDisconnect*. The disconnection code must be found after the last "Net" statement. The "ConnectionNum" variable, required for the disconnection, contains the value returned by *NetConnect*.

The code for disconnecting from an RPC server WinDev is as follows:

```

//Disconnect from a server
//WinDev RPC
//ConnectionNum contains the value
//returned by NetConnect
NetDisconnect(ConnectionNum)

```

12.3 Net functions

The following functions are used to manage the WinDev FTP/RPC communications:

NetIPAddress	Returns the IP (Internet Protocol) address of a computer
NetMACAddress	Returns the MAC address of one of the network cards found on the computer.
NetConnect	Connects to an FTP server (File Transfer Protocol) or to an RPC server (Remote Procedure Call) created by WinDev
NetDisconnect	Disconnects from a WinDev FTP (File Transfer Protocol) or RPC (Remote Procedure Call) server
NetStartServer	Starts an FTP server (File Transfer Protocol) or a RPC server (Remote Procedure Call) created by WinDev
NetSendBuffer	Transmits a character string to an FTP server (File Transfer Protocol) or to a RPC server (Remote Procedure Call) created by WinDev
NetSendFile	Transmits a file to an FTP server (File Transfer Protocol) created by WinDev
NetSendMessage	Triggers a Windows event on the system of an FTP server (File Transfer Protocol) or RPC server (Remote Procedure Call) created by WinDev
NetClientSendMessage	Sends a message from the server to all the connected client computers
NetExecute	Runs a specific program on an FTP server (File Transfer Protocol) or on a RPC server (Remote Procedure Call) created by WinDev
NetCloseRemoteAccess	Closes the line that was previously opened by NetOpenRemoteAccess
NetEndServer	Stops an FTP server (File Transfer Protocol) or a RPC server (Remote Procedure Call) that was created with WinDev and that was previously started by NetStartServer
NetInfoRemoteAccess	Returns information and statistics about the status of a connection
NetIPToMAC	Returns the MAC address corresponding to the given IP address.
NetListRemoteAccess	Returns the list of available remote accesses
NetListIPAddress	Returns the list of IP addresses (Internet Protocol) for a computer.
NetDiskList	Lists the available disks
NetDirList	Lists the directories and the files
NetMACToIP	Returns the IP address corresponding to a MAC address
NetClientMessageBox	Displays a dialog box on each client computer connected to the server
NetRemoteMessageBox	Displays a dialog box on the screen of the FTP server (File Transfer Protocol) or RPC server (Remote Procedure Call) created by WinDev
NetMessageError	Returns the error message corresponding to the error number returned by NetOpenRemoteAccess and NetCloseRemoteAccess
NetMachineName	Returns the name of the local computer
NetServerOption	Manages the rights of the clients on the server (FTP server (File Transfer Protocol) or RPC server (Remote Procedure Call) created by WinDev)
NetOpenRemoteAccess	Establishes a remote connection with a modem
NetGetFile	Retrieves an existing file from an FTP server (File Transfer Protocol) created by WinDev
NetRenameFile	Renames (or moves) a file found on the FTP server (File Transfer Protocol) or RPC server (Remote Procedure Call) created by WinDev
NetEraseFile	Deletes a file from the FTP server (File Transfer Protocol) or from the RPC server (Remote Procedure Call) created by WinDev

See the online help for more details.

13. COMMUNICATING WITH AN FTP SERVER

13.1 Handling files on a RPC server

13.1.1 Overview

The FTP (File Transfer Protocol) is a protocol used to transfer files from a site to another remote site. This protocol is used to exchange files via TCP/IP, Wi-Fi or Internet.

Several thousands of file servers can be accessed by FTP on Internet. These servers propose shareware or freeware to the public.

Several WLanguage functions enable you to manage the files on an FTP server from your applications or from your sites.

13.1.2 FTP

Use convention

- In theory, you cannot connect to an FTP site without an FTP account and a password. Only a user with an account and a password can connect to this site.
- Practically, all the servers found on Internet have an Anonymous account. The password of this account is not implemented but an email address will be requested.

To download the files found on an FTP server, all you have to do is connect as "anonymous user".

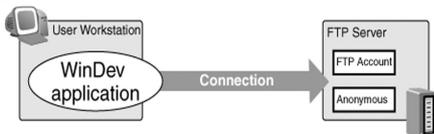
To upload files to an FTP server (to send HTML pages to your Web site for example), a non-anonymous account and a password are required.

13.1.3 Principle

To handle files on an FTP server from an application or from a site, you must follow these rules:

1. Connect to an FTP server (*FTPConnect*).

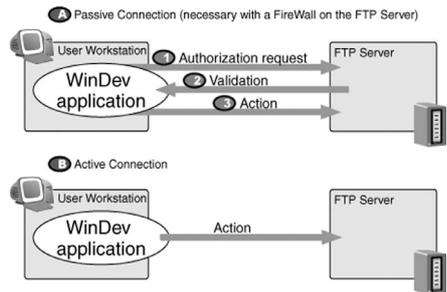
1 Connection



2. Upload and download files (according to the connection mode). For a passive connection, the application or the site must request the authorization from the FTP server before each file operation on the server. It also allows you to:

- find out the characteristics of the files found on the FTP server: attributes, size, ...
- handle the files found on an FTP server: creation, deletion, ...
- list the files of a directory found on the FTP server by running a procedure used to perform a process for each file found.

2 Communication



3. Close the connection to the server (*FTPDisconnect*).

3 Disconnection



13.1.4 Relative path/Absolute path

The notions of relative path and absolute path are very important in an FTP application.

- A path starting with a slash is considered as being an **absolute path**: this is the path from the root of the FTP server (parameter specific to the server).
ex: /pub/user/FIONA

- A path not starting with a slash is considered as being a **relative path**, which means a path given in relation to the current directory. This current directory can be returned or modified by **FTPCurrentDir**.

When connecting to an FTP site, the initial directory (the "home directory" of the user) is not necessarily found at the root of the FTP server. Therefore, we recommend that you use relative paths.

13.1.5 Example

A full example is supplied with WinDev: WD FTP.

This example is used to connect to and to disconnect from an FTP server.

Once the connection is established, all the files found on the server and on the current computer are listed.

You have the ability to:

- transfer the files from the local computer to this FTP server and download the files found on the FTP server to the local computer.
- create, rename and delete the files and/or the directories found on the FTP server.
- find out the characteristics of the files found on the FTP server (size and attributes of the files).

13.2 FTP functions

The following functions are used to manage the FTP (File Transfer Protocol):

FTPAttribute	Identifies the attributes of a file found on an FTP server (File Transfer Protocol)
FTPCommand	Sends a specific FTP command to a server
FTPConnect	Connects the current computer to an FTP server (File Transfer Protocol)
FTPDate	Returns the different dates (creation, modification or access) associated with a file found on an FTP server (File Transfer Protocol)
FTPDisconnect	Disconnects the current computer from the FTP server (File Transfer Protocol)
FTPSend	Transfers a file or directory to an FTP server (File Transfer Protocol)
FTPTime	Returns the times (creation, modification or access) associated with a file on an FTP server (File Transfer Protocol)
FTPListFile	Lists the files found in a directory of an FTP server and returns the number of listed files
FTPName	Returns the name of the last file accessed by an FTP function (File Transfer Protocol)
FTPProxy	Specifies whether the communication functions that use the FTP protocol must go through a proxy to run their requests
FTPGet	Transfers a file or a directory from an FTP server (File Transfer Protocol) to the current computer
FTPRenameFile	Renames or moves a file found on an FTP server (File Transfer Protocol)
FTPMakeDir	Creates a directory on an FTP server (File Transfer Protocol)
FTPCurrentDir	Identifies or modifies the current directory on an FTP server (File Transfer Protocol)
FTPRemoveDir	Deletes a directory and its content (files and sub-directories) from an FTP server (File Transfer Protocol)
FTPDeleteFile	Deletes a file from an FTP server (File Transfer Protocol)
FTPSize	Returns the size (in bytes) of a file found on an FTP server (File Transfer Protocol)

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

14. MANAGING THE SOCKETS

14.1 Overview

Several WLanguage functions are used to perform an advanced management of sockets.

A socket is a communication resource used by the applications to communicate between computers regardless of the network type.

This communication mode can be used, for example, to establish a communication between computers connected by Internet.

With a **Pocket PC**, data can be exchanged between two computers:

- by Wi-Fi.
- by ActiveSync.
- by GPRS (caution: some phone contracts do not allow you to connect two computers).
- by network card.

WinDev, WebDev and WinDev Mobile enable you to manage:

- The standard sockets
- The UDP sockets
- The sockets by infrared
- The SSL sockets

Examples for using the sockets:

- managing a messaging in real-time,

- access to a news server (forum).

14.1.1 Different possibilities

A WinDev application can manage the sockets according to different modes:

- Client WinDev application: the application connects to a standard server and exchanges data via a socket. See “Client WinDev application/Client WebDev site”, page 286 for more details.
- WinDev “Simplified Server” application: the WinDev application is a server that exchanges information via a socket with a single client computer (WinDev socket recommended on the client computer but not mandatory). See “WinDev “Simplified Server” application”, page 287 for more details.
- WinDev “Standard Server” application: the WinDev application is a server that exchanges information via sockets with several client computers. See “Standard socket server”, page 288 for more details.

14.1.2 Example

WinDev is supplied with an example for socket management: **WD Instant messaging**.

14.2 Client WinDev application/Client WebDev site

A client application of a socket server connects to a standard server in order to exchange information via a socket.

Example: A client WinDev application or a client WebDev site can connect to a standard news server on Internet.

14.2.1 Principle of a client application or client site

Step 1: Connecting to the server

To connect to a server socket, all you have to do is use **SocketConnect**. This function is used to establish a request for connecting to the server.

The socket is identified by its port and by an address.



Step 2: Exchanging data

Once two computers have connected their socket, a communication channel is established between these two computers. These two computers can read and write character strings on the socket.

Note: To avoid locking the applications or the sites, the incoming email must be managed by a special thread (see “Managing the threads”, page 293 for more details).

To read and write on the server socket, WinDev the client application or the client WebDev site must use **SocketRead** and **SocketWrite**.



Caution: To perform a read operation, a write operation must have been performed beforehand. For example:

1. The client computer writes onto the socket: it sends a request to the server.
2. The server computer performs a read operation in the socket.
3. If a response to the message is required, the server sends a response to the client computer.

Step 3: Ending the communication

To end the communication, all you have to do is close the socket from the client computer with **SocketClose**.



Note: the communication can also be ended from the server.

14.2.2 Transmission mode of information

The transmission mode of the message defines the mode used to specify the length of the message.

Several methods are available to find out the length of the message during the communication between sockets.

Method 1: WinDev/WebDev mode: By default, the number of characters in the message is specified at the beginning of the message. This transmission mode is recommended when the sockets are used to communicate between two WinDev applications, or between a WebDev site and a WebDev server application.

The message has the following format: "11\r\nHelloWord"

Method 2: Standard mode: The end of the message is signaled by a specific character, defined in advance. This transmission mode is recommended when the sockets are used to communicate between a WinDev application and another application, or between a WebDev site and another site. In this case, a specific character must be included in the message to indicate that the message is over.

The message has the following format: "Hello World<EOF>"

Method 3: Standard mode with buffer: Corresponds to the standard mode optimized for the most common protocols on Internet.

SocketChangeTransmissionMode allows you to modify the transmission mode used.

14.3 WinDev "Simplified Server" application

WD WDMobile

WinDev gives you the ability to create a simplified socket server. This server is used to communicate with a single client computer at a time. This type of application is very useful when two remote applications must communicate between themselves.

Note: WinDev also gives you the ability to create a more elaborated socket server (standard socket server), managing the connection of several client computers. See “Standard socket server”, page 288 for more details.

14.3.1 The simplified server

Step 1: Creating the socket

To create the socket, the server uses **SocketCreate**. A socket is associated with a specific port. To simplify the use of the socket by programming on the

server, specify the name of the socket.

The client computer will connect to this socket in order to exchange data. The connection between the two computers will be actually established during the first exchange of data between the two computers (which means when the server reads information for the first time).

The connection is established during the first successful attempt of **SocketRead** on the server.



Step 2: Exchanging data

When two computers use the same socket, a communication stream is established between these two computers. These two computers can read and write character strings on the socket.

Note: To avoid locking applications, the incoming messages are often managed by a special thread (see “Managing the threads”, page 293 for more details).

To read and write on the socket, the WinDev server application must use **SocketRead** and **SocketWrite**.



Caution: To perform a read operation, a write operation must have been performed beforehand. For example:

1. The client computer writes onto the socket: it sends a request to the server.
2. The server performs a read operation on the socket in a thread. As soon as a message is received, the message is processed by the server.
3. If a response to the message is required, the server identifies the client computer (**SocketClientInfo**) and sends a response to it.

Step 3: Closing the socket

To end the communication, the server can close the socket with **SocketClose**.



Note: the socket can also be closed by the client computer.

14.3.2 Transmission mode of information

The transmission mode of the message defines the mode used to specify the length of the message.

Several methods can be used to define the length of the message during the communication between sockets.

Method 1: WinDev mode: By default, the number of characters in the message is specified at the beginning of the message. This transmission mode is recommended when the sockets are used to communicate between two WinDev applications.

The message has the following format: "11\r\nHelloWord"

Method 2: Standard mode: The end of the message is signaled by a specific character, defined in advance. This transmission mode is recommended when the sockets are used to communicate between a WinDev application and another application. In this case, a specific character must be included in the message to indicate that the message is over.

The message has the following format: "Hello World<EOF>"

Method 3: Standard mode with buffer: Corresponds to the standard mode optimized for the most common Internet protocols.

SocketChangeTransmissionMode allows you to modify the transmission mode used.

14.4 Standard socket server

WD

WinDev gives you the ability to create a standard socket server. This server enables you to manage the connection of several client computers to the same socket. This principle is used to create a news server for example.

14.4.1 The standard socket server

Step 1: Creating the socket

To create the socket, the server uses **SocketCreate**. A socket is associated with a specific port. Several sockets can be created, each socket using a specific port number. A name (used to handle the socket by programming) and a port number are associated with each socket.

Step 2: Waiting for connection on the socket

All the computers wanting to communicate with the server can connect to the socket: these are the client computers. The server manages the different connection requests from the client computers via **SocketWaitForConnection**.

This function is used to find out whether a connection request is performed on a specific socket.

We recommend that you use **SocketWaitForConnection** in a specific thread. Therefore, this function is performed in background task. When a connection request is detected, you can:

- accept the connection (**SocketAccept**): in this case, a specific communication channel is created. To avoid locking the applications, the incoming messages are often managed by a specific thread (see "Managing the threads", page 293 for more details).
- refuse the connection (**SocketDeny**).

Example

The following example is used to create a socket on the server and to manage the connections of the client computers in a thread. If the connection is accepted, the management of this connection is performed by a specific thread.

A thread is run for each connection. Each thread uses the same service function ("ManagementProcedure"). To allow the procedure to be run by several threads simultaneously, the synchronization mode of the threads must be changed: use **Thread-Mode** associated with the **threadCriticalSection** constant in the initialization code of the project. The synchronization between the threads must be done manually (see "Managing the threads", page 293 for more details).

To differentiate the threads, their name corresponds to the name of the communication channel (unique name).

```
IF NOT SocketCreate("Server", ...
    8000) THEN
    Error("creation error " + ...
        ErrorMessage)
ELSE
    ThreadExecute("Thread1", ...
        threadNormal, ...
        WaitProcedure)
END
```

```
PROCEDURE WaitProcedure()
LOOP
IF SocketWaitForConnection(...
    "Server")
THEN
Channel is string
Channel = SocketAccept("Server")
ThreadExecute(Channel, ...
    threadNormal, ...
    ManagementProcedure, channel)
END
END
```

Step 3: Exchanging data

When two computers use the same socket, a communication channel is established between these two computers. These two computers can read and write character strings on the socket.

To read and write on the socket, the WinDev server application must use **SocketRead** and **SocketWrite**.

Caution: To perform a read operation, a write operation must have been performed beforehand. For example:

1. The client computer writes onto the socket: it sends a request to the server.
2. The server reads from the socket in a thread. As soon as a message is received, the message is processed by the server.
3. If a response to the message is required, the server identifies the client computer and sends a response to it.

Step 4: To end the communication, all you have to do is close the socket with **SocketClose**.

14.4.2 Transmission mode of information

The transmission mode of the message defines the mode used to specify the length of the message.

Several methods can be used to define the length of the message during the communication between sockets.

Method 1: WinDev mode: By default, the number of characters in the message is specified at the beginning of the message. This transmission mode is recommended when the sockets are used to communicate between two WinDev applications.

The message has the following format: "11\r\nHelloWord"

Method 2: Standard mode: The end of the message is signaled by a specific character, defined in advance.

This transmission mode is recommended when the sockets are used to communicate between a WinDev application and another application. In this case, a specific character must be included in the message to indicate that the message is over.

The message has the following format: "Hello World<EOF>"

Method 3: standard mode with buffer: Corresponds to the standard mode optimized for the most common Internet protocols.

SocketChangeTransmissionMode allows you to modify the transmission mode used.

14.5 Socket functions

The following functions are used to manage the sockets:

SocketAccept	Accepts the connection of a client computer on the server
SocketWaitForConnection	Checks whether a request for connection was performed by a client computer
SocketChangeTransmissionMode	Changes the transmission mode used on a socket
SocketClientInfo	Allows a server to retrieve information about a socket connected to a client computer
SocketConnect	Allows you to connect to a socket
SocketConnectBluetooth	Connects a client computer to a specific Bluetooth socket.
SocketConnectInfrared	Connects a client computer to a given socket via the infrared port
SocketConnectSSL	Connects a client computer to an SSL server.
SocketCreate	Creates a socket
SocketCreateBluetooth	Creates a Bluetooth socket.
SocketCreateInfrared	Creates a socket that uses the infrared port
SocketCreateSSL	Creates an SSL socket.
SocketCreateUDP	Creates a socket that uses the UDP protocol
SocketWrite	Sends a message between two sockets
SocketExist	Checks the existence of a socket
SocketClose	Closes a socket or a connection to a socket on the server
SocketRead	Reads a message sent by a socket
SocketDeny	Refuses the connection requested by a client computer

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

15. MANAGING THE BLUETOOTH KEYS

15.1 Overview

Create a new type of application! With no contact between the PC and the mobile device, several processes can be run on the PC by using the data found on the mobile device.

Some examples:

- An application on PC (or mobile device) can emulate services on a device that supports the bluetooth standard.
- An application can automatically retrieve photos from a cell phone.
- An application can read the list of contacts found on a cell phone.
- A PC can print the data found on a PDA.

15.2 How do I proceed?

Several groups of functions allow you to communicate via Bluetooth:

- **The Bluetooth functions** (BTxxx): These functions are used to manage the Bluetooth devices. You can for example list the services found on a device, and enable or disable them. You can also authenticate toward a Bluetooth device and get various information about the device.
- The **OBEXxxx functions**: These functions are used to manage the exchanges of files via the OBEX protocol.
- The functions named **SocketConnectBluetooth** and **SocketCreateBluetooth** that allow you to manage the sockets that use the Bluetooth devices.

15.3 Which keys should be used?

The functions allowing you to handle the Bluetooth keys operate with the keys that use a protocol stack issued by Microsoft or Bluesoleil. To find out the stack used, call **BTStack**.

Some WLanguage functions can only be used with some types of stacks. This information is presented in details in each relevant WLanguage function.

The following keys use a Microsoft stack (without installing to driver supplied with the keys):

- D-Link DBT-120,
- Broadcom Bluetooth 2.0 EDR USB,
- EMTEC Dongle Bluetooth 2.0 EDR - EKCOB110,
- HAMA Nano Bluetooth Adapter 2.0 EDR - 10m,
- Conceptoric Dongle Bluetooth 2.0 EDR - CBT200U2A,
- Kensington Bluetooth USB Micro Adapter (Bluetooth 2.0 - USB 2.0),
- (*) Belkin Bluetooth 2.0 Adapter - F8T013FR1 - 10-meter range (USB 2.0),
- (*) Belkin Bluetooth 2.0 Adapter - F8T012FR1 - 100-meter range (USB 2.0),
- TrendNet TBW-104UB - USB Bluetooth 2.0 Adapter,
- (*) TrendNet TBW-102UB - Bluetooth Class 2 Adapter (Bluetooth 1.1), ...

This list is not exhaustive and it will be updated on a regular basis.

(*) Some keys do not directly operate in Windows, a driver must be installed via the following operating mode:

1. If it was installed, uninstall the driver provided with the key via "Add/Remove program" in the Control Panel,
2. Insert the CD provided with the key in the drive, DO NOT start its automatic execution,
3. Connect the key,
4. Ask Windows to find the driver on the CD.

This method enables you to have a key recognized by Windows and therefore to use a Microsoft stack.

15.4 The Bluetooth and OBEX functions

15.4.1 Bluetooth functions

The following functions are used to manage the Bluetooth devices:

BTAcceptConnection	Allows you to find out whether a Bluetooth radio accepts (or not) the connection requests coming from the devices.
BTActivate	Enables or disables the management of Bluetooth on the device
BTEnableService	Enables a service provided by a Bluetooth device.
BTChangeConnectivity	Allows you to configure a Bluetooth radio to accept (or not) the requests for connection coming from the devices.
BTChangeVisibility	Changes the visibility of a Bluetooth radio.
BTConnectDevice	Allows you to be authenticated toward a Bluetooth device.
BTDisconnectDevice	Cancel the authentication beside a Bluetooth device.
BTDisableService	Disables a service on a Bluetooth device.
BTIsVisible	Allows you to find out whether a Bluetooth radio is visible.
BTStatus	Returns the current activation status of Bluetooth on the device or asks to be notified when the activation status changes
BTInfoDevice	Returns specific information about a Bluetooth device.
BTInfoRadio	Returns specific information about a Bluetooth radio connected to the computer.
BTListDevice	Returns the list of accessible Bluetooth devices.
BTListRadio	Returns the list of Bluetooth radios connected to the computer.
BTListService	Returns the list of services provided by a Bluetooth device.
BTOpenDeviceProperties	Opens the window of properties for a Bluetooth device.
BTStack	Returns the Bluetooth stack currently used.
BTSelectDevice	Opens a system window to select a Bluetooth device.

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

15.4.2 OBEX functions

The following functions are used to manage the file exchanges via the OBEX protocol:

OBEXConnect	Allows you to connect to a device that supports the OBEX protocol.
OBEXDisconnect	Allows you to disconnect from a device that supports the OBEX protocol.
OBEXSendFile	Sends a file to a device that supports the OBEX protocol.
OBEXSendVCard	Sends a VCard file (virtual business card) to a device that supports the OBEX protocol.
OBEXListFile	Lists the files shared by a device that uses the OBEX protocol.
OBEXGetFile	Retrieves a file from a device that supports the OBEX protocol.
OBEXGetVCard	Retrieves a VCard from a device that supports the OBEX protocol.

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

16. MANAGING THE THREADS

16.1 Overview

Several WLanguage functions can be used to perform an advanced management of threads.

The threads are used to run a code (or processes) in parallel of the main application. Therefore, several processes can be run in background task without locking the main application.

The threads replace some types of timers.

An efficient thread is a thread that waits for an event such as a user action, an incoming email, an incoming phone call, ...

Examples for using the threads:

- Retrieving emails in background task while a new email is typed.
- Communication application or site: manage the incoming calls, communication with sockets, ...

16.2 Principle for using the threads

An application or a site is currently run. This application or this site is run in a main thread.

This application or this site can start a secondary thread at any time: this thread is run in parallel of the application or site. This thread corresponds to a local or global procedure of the application or site.

This secondary thread will be run in parallel of the main application or site. All the processes that can be run in background task can be performed in this thread: receiving emails, ...

Note: an efficient thread is a thread that waits for a specific event, such as a user action, an incoming call or email, ...

16.2.1 Simple management of threads

A secondary thread is created by *ThreadExecute*.

A secondary thread is automatically stopped when:

- the procedure corresponding to the thread is over,
- the object from which the thread originated is closed.

To force a thread to stop, you also have the ability to use *ThreadStop*.

16.1.1 From simple management to advanced management of threads

Several methods can be used to manage the threads:

- Running threads.
- Managing the semaphores inside threads, which means limiting the simultaneous execution of a code by one or more threads at a given time.
- Managing the signals in order to perform a synchronization between several threads.

16.1.2 Example

WinDev is supplied with an example for thread management: **WD Thread Pool**.

Caution: if a WLanguage function is currently run when the thread is stopped, the thread will be actually stopped after the execution of the function.

WLanguage functions

The following functions are used to manage the threads:

ThreadStop	Stops a secondary "thread".
ThreadExecute	Starts the execution of a secondary "thread". This "thread" is a non-locking thread.
ThreadMode	Changes the management mode of the threads.

16.2.2 Characteristics of the threads

In WLanguage, a secondary thread can be associated with:

- a procedure local to the current window or page,
- a procedure global to the project,
- a method of a class,
- a global method of a class

16.2.3 Access to the existing elements and HFSQL context

When creating a thread, all the declarations, objects, elements, ... are common:

- to the new secondary thread
- to the thread in which the secondary thread was created (the main thread in most cases).

Therefore, these threads can access the variables, procedures, ... All the variables created once a thread is started are accessible in the thread where they have been created.

Similarly, when creating a thread, the HFSQL context is automatically duplicated. Each thread handles a specific HFSQL context. The number of HFSQL contexts is equal to the number of threads currently run. The entire HFSQL context is copied (filter, search condition, ...). The HFSQL context evolves independently in each thread.

This allows you to perform two different browse operations on the same file in two different threads.

Example: A filter is created on Customer file. *ThreadExecute* is called to create the CTX2 thread. The Customer file is filtered in each thread. If the filter is disabled in the main thread, the filter will still be enabled in the CTX2 thread.

Caution: Writing and assigning in a thread: If write operations or assignments are performed in a thread, the other threads currently run do not share this information. Some inconsistencies may occur.

Example:

Code of Thread 1	Code of Thread 2
a=i	b=i
a++	b++
i=a	i=b

These two threads share the variables but they do not manage the access to the common resources. If the thread 1 is run before the thread 2, i will be set to 1 instead of 2.

Note: To share an assignment among several threads, the semaphores must be used.

16.2.4 Limits of the processes performed by the thread

Forbidden processes

Caution: The following processes cannot be run in the threads:

- Opening WinDev windows with the WLanguage functions such as *Open*, *Use*, *Close*, ... A specific management mode must be implemented if windows must be handled in the threads (very rare). See "Managing the opening of a WinDev window in a secondary thread", page 298 for more details.
- Displaying WebDev pages (or page contexts) with the WLanguage functions such as *ContextOpen*, *FramesetDisplay*, *PageDisplay*, *PageUse*, ...
- Managing the events.
- Managing the timers.

Processes of a WinDev application/site WebDev

By default, any WinDev/WebDev process (click code of a button for example), all the procedures as well as the methods of classes can only be run by a single thread at a given time.

To allow several threads to run these processes at the same time, you must:

1. Change the default management mode of threads (*ThreadMode*).
2. Manage the critical sections and the semaphores in the code of the application or site.

16.3 Managing the semaphores in the threads

The semaphores are used to limit the simultaneous execution of a code (procedure, code line, ...) to one or more threads at a given time.

For example: Two specific threads are used in a banking application:

- a thread to credit the accounts,
- a thread to debit the accounts.

At any given time, there can only be a single account credit or a single account debit.

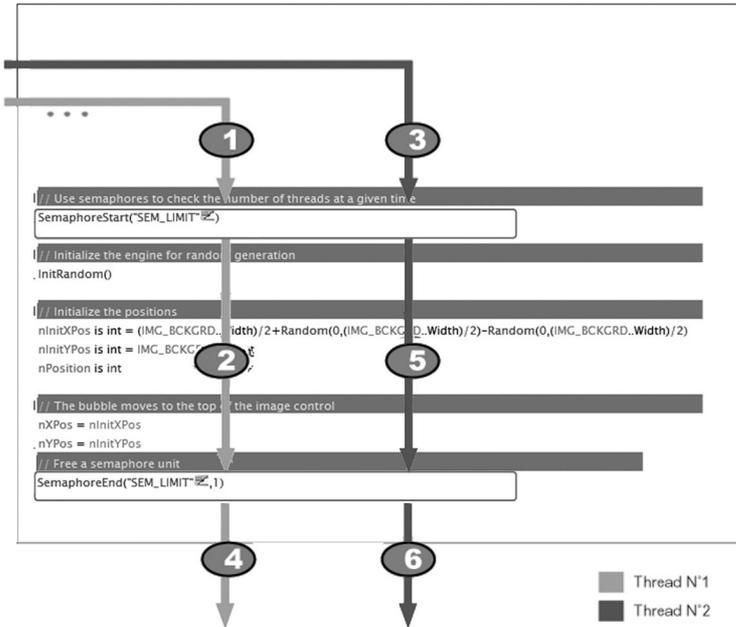
16.3.1 Principle

The semaphore was created by *SemaphoreCreate*.

1. The thread #1 runs *SemaphoreStart*: no thread is currently found in the semaphore.
2. The thread #1 runs the section of code that is protected by the semaphore.
3. While the thread #1 runs the code protected by the semaphore, a thread #2 runs *SemaphoreStart*: the code protected by the semaphore being already run by the thread #1, the thread #2 waits for the semaphore to be unlocked.

4. The thread #1 runs **SemaphoreEnd**: no other thread runs the code of the semaphore.
5. The thread #2 can run the code protected by the semaphore.

6. The thread #2 runs **SemaphoreEnd**: no thread runs the code of the semaphore.



16.3.2 Implementing a semaphore

The different steps for implementing a semaphore are as follows:

1. Create a semaphore with **SemaphoreCreate**. The semaphore is associated with a name.
2. Call **SemaphoreStart** before the section of code to protect.
3. Call **SemaphoreEnd** after the section of code to protect. The code lines found after **SemaphoreEnd** will not be protected.
4. Destroying the semaphore with **SemaphoreDestroy**.

Notes:

- The code sections protected by a semaphore must be as short as possible and they must only affect the "critical" processes.
- A semaphore with the same name can be used to protect several code sections. A single thread can be found at a given time in one of the areas protected by the semaphore.
- When a thread is pending, the resources of the processor are not used.
- The semaphores apply to the main thread and to the secondary threads as well (created by **ThreadExecute**).

You must avoid locking the main thread. Indeed, if the main thread is locked (pending), the application or the site cannot be run anymore.

- **SemaphoreStart** and **SemaphoreEnd** must be used in the same process (in a procedure for example).

The functions for managing the semaphores

The following WLanguage functions are used to manage the semaphores:

SemaphoreCreate	Creates a semaphore
SemaphoreStart	Locks the current thread while waiting for the semaphore to be opened (which means until a free spot becomes "available" in the protected section)
SemaphoreDestroy	Explicitly destroys a semaphore
SemaphoreEnd	Allows one or more threads to exit from the area protected by the semaphore

Example: To perform an assignment shared by several threads, you must encapsulate in a semaphore the assignment of the variables as well as their reading.

16.3.3 A limited semaphore: the critical section

A critical section is a semaphore limited to a single thread on a single code section (process, procedure, ...).

Unlike the semaphores, a critical section can be used once only.

For example, a critical section can be used to protect a procedure for updating the controls in a win-

dow or a page.

The functions for managing the critical sections

The following WLanguage functions are used to manage the critical sections:

CriticalSectionStart Marks the beginning of a critical section: no other thread will be able to run the code as long as the current thread does not exit from the critical section

CriticalSectionEnd Marks the end of a critical section: another thread will be able to run the code

16.4 Managing the mutexes in the threads

The mutexes are used to limit the simultaneous execution of a code (procedure, code line, ...) to one thread at a given time. A mutex can be shared among several applications.

Note: Other systems can also be used to protect a section of code:

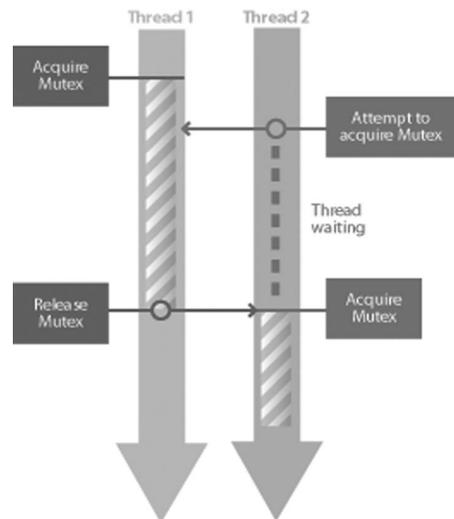
- the semaphores are used to limit the simultaneous execution of a code (procedure, code line, ...) to one or more threads at a given time. A semaphore can be shared among several applications.
- the critical sections are used to limit the simultaneous execution of a code (procedure, code line, ...) to a single thread at a given time in a single application.

16.4.1 Principle

The mutex was created by **MutexCreate**.

1. The thread #1 runs **MutexStart**: no thread is currently found in the mutex.
2. The thread #1 runs the code section protected by the mutex.
3. While the thread #1 runs the code protected by the semaphore, a thread #2 runs **MutexStart**: the code protected by the mutex being already run by the thread #1, the thread #2 waits for the mutex to be unlocked.
4. The thread #1 runs **MutexEnd**: no other thread runs the code of the mutex.
5. The thread #2 can run the code protected by the mutex.

6. The thread #2 runs **MutexEnd**: no other thread runs the code of the mutex.



16.4.2 How do I implement a mutex?

The steps: the different steps for implementing a mutex are as follows:

The different steps for implementing a mutex are as follows:

1. Creating a mutex with **MutexCreate**. The mutex is associated with a name.
2. Calling **MutexStart** before the code section to protect.
3. Calling **MutexEnd** after the code section to protect. The code lines found after **MutexEnd** will no longer be protected.

4. Destroying the mutex with *MutexDestroy*.

Notes:

- The code sections protected by a mutex must be as short as possible and they must only affect the "critical" processes.
- When a thread is pending, the resources of the processor are not used.
- The mutexes apply to the main thread and to the secondary threads (created by *ThreadExecute*). You must avoid locking the main thread. Indeed, if the main thread is locked (pending), the application cannot be run anymore.
- *MutexStart* and *MutexEnd* must be used in the same process (in a procedure for example).
- The mutexes can be shared (or not) among the

different applications run on the computer. All you have to do is specify the share mode of the mutexes during their creation (*MutexCreate*).

16.4.3 The functions for managing the mutexes

MutexCreate	Explicitly creates a mutex
MutexStart	Locks the current thread while waiting for the mutex to be freed
MutexDestroy	Explicitly destroys a mutex.
MutexEnd	Signals that the thread frees the mutex

16.5 Synchronizing the threads via signals

The signals can be used to synchronize the different threads of an application. A thread can wait for the execution of another thread.

Two management modes can be used to manage the signals:

- simple management: synchronization between two threads
- advanced management: synchronization between several threads

16.5.1 Simple management of signals

Two threads are run in parallel (a main thread and a secondary thread for instance). One of the threads waits for a specific action from the second thread before it can be run.

Example: Application or site managing a spelling checker

When the user types the SPACE character in an edit control, the spelling checker is automatically started to check the previous word.

In this case, the spell check is managed in a secondary thread.

Whenever the SPACE key is pressed, the main thread sends a signal to the secondary thread in order to start the spelling checker.

The code is as follows:

- Code of the main thread:

```
IF Right(Edit1, 1) = " " THEN
    ThreadSendSignal(...
                        "ThreadCorrection")
END
```

- Code of the secondary thread (ThreadCorrection):

```
LOOP
    IF ThreadWaitSignal() = True
    THEN StartCorrection()
    END
END
```

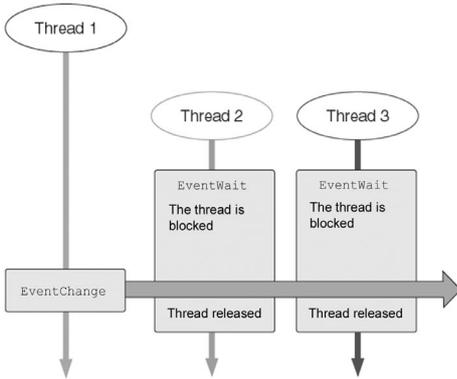
Functions of WLanguage

The following functions are used to perform a simple management of signals:

ThreadWaitSignal	The current "thread" is locked as long as no signal is received from another "thread"
ThreadSendSignal	The current "thread" sends a signal to the specified "thread" to unlock it

16.5.2 Advanced management of signals

An advanced management of signals consists in communicating between several threads (more than 2). Some threads are waiting for a task performed by the main thread. When the main thread performs this task, it sends a signal to all the secondary threads.



Implementation

The steps for implementing the advanced management of signals are as follows:

1. Create a signal (**EventCreate**). By default, this signal is closed.
2. Wait for the signal (**EventWait**).
3. Synchronizing the threads with **EventChange**: the signal is opened. All the pending threads are unlocked and the signal is automatically closed (default behavior).
4. Destroy the signal (**EventDestroy**).

The functions for advanced management of signals

The following functions are used to manage the advanced signals:

EventWait	Locks the current thread while waiting for the specified signal to be opened
EventCreate	Creates a signal
EventDestroy	Explicitly destroys a signal
EventChange	Modifies the status of a signal

16.6 Managing the opening of a WinDev window in a secondary thread

A secondary thread cannot directly open a window with the standard WLanguage functions such as: **Open**, **Use**, ...

Nevertheless, you may want to display a window from a secondary thread. For example, if a thread is used to manage some incoming calls, a window containing the characteristics of the caller can be displayed by this thread when an incoming call is detected.

The solution consists in making the main thread open the window.

16.6.1 Opening a window from a secondary thread

To open a window from a secondary thread:

1. When the secondary thread wants to open a window, it sends a message to the main thread with **PostMessage**.

This message indicates to the main thread the window that must be opened as well as the parameters required to open the window.

2. The main thread manages a specific event, corresponding to the request for window opening. When this event occurs, the WLanguage procedure opens the specified window (with **Open** for exam-

ple) with the specified parameters.

16.6.2 Example

The above-mentioned method is used in the example named WD Instant messaging.

The code for opening the main window is used to:

- implement an event to manage the opening of the window in the main thread.
- trigger the secondary threads.

```
// Implement an event to
// manage the window opening
// for "chat"
gnEventID = Event(...
    "OpenChat","*.*", ...
    "StartChat")
IF gnEventID = 0 THEN
    // Error while implementing
    // the event for managing the
    // "chat" window
    Error("Unable to manage"+...
        "the window for opening"+ ...
        "chats",ErrorInfo())
    // The application cannot
    // operate without this event
EndProgram()
END
```

```
// Listen to the requests for
// connection
// and for presence tests
ThreadExecute("ThreadPresence", ...
    threadNormal, ...
    "ListenPresenceRequest")
ThreadExecute("ThreadRequest", ...
    threadNormal, ...
    "ListenConnectionRequest")
```

- The window is opened in the code of the **ListenRequestStartMessage** procedure for example. This procedure is run in a thread. The code used to open the window from the thread is as follows:

```
// Request the opening of
// the "chat" window
PostMessage(...
    Handle(Main_Window), ...
    "StartChat", CONTACT.CONTACTID, ...
    CONTACT.CONTACTID)
```

The "StartChat" message is sent to the main window (identified by its handle). This message contains the different parameters that must be passed to the window. The message is handled by the event defined in the opening code of the window. When the message occurs, the ChatOpening procedure is automatically run. This procedure is used to open the window.

The code of this procedure is as follows:

```
PROCEDURE OpenChat (nParam, ...
    nContact)
OpenSister (Window_Message, ...
    nContact)
```

16.7 Functions for managing the threads

The following functions are used to manage the threads:

ExecuteMainThread	Triggers the execution of the procedure specified in the main thread of the application.
MutexCreate	Explicitly creates a mutex
MutexStart	Locks the current thread while waiting for the mutex to be freed
MutexDestroy	Explicitly destroys a mutex
MutexEnd	Signals that the thread frees the mutex
CriticalSectionStart	Marks the beginning of a critical section: no other thread will be able to run the code as long as the current thread does not exit from the critical section
CriticalSectionEnd	Marks the end of a critical section: another thread will be able to run the code
SemaphoreCreate	Creates a semaphore
SemaphoreStart	Locks the current thread while waiting for the semaphore to be opened (which means until a free spot becomes "available" in the protected section)
SemaphoreDestroy	Explicitly destroys a semaphore
SemaphoreEnd	Allows one or more threads to exit from the section protected by the semaphore
EventWait	Locks the current thread while waiting for the specified signal to be opened
EventCreate	Creates a signal
EventDestroy	Explicitly destroys a signal
EventChange	Modifies the status of a signal
ThreadStop	Stops a secondary "thread"
ThreadWait	Waits for the end of execution of the specified "thread". A maximum time-out can be specified
ThreadWaitSignal	The current "thread" is locked as long as no signal is received from another "thread"
ThreadCurrent	Returns the name of the thread currently run
ThreadSendSignal	The current "thread" sends a signal to the specified "thread" to unlock it
ThreadState	Returns the current status of a thread
ThreadExecute	Starts the execution of a secondary "thread". This "thread" is a non-locking thread
ThreadMode	Changes the management mode of the threads

ThreadPause	Pauses the current thread during the specified duration
ThreadPersistent	Makes a thread persistent
ThreadPriority	Returns or modifies the priority level of a thread
ThreadResume	Resumes the execution of a thread that was interrupted by <i>ThreadSuspend</i>
ThreadSuspend	Temporarily suspends the execution of the specified "thread". The current processes is not locked

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

17. SOAP

17.1 Overview

"SOAP" (Simple Object Access Protocol) is a communication protocol used to run procedures on a remote server.

This protocol is mainly based on the HTTP and XML protocols. It can also be used with the SMTP protocol.



The data is transmitted between the local computer and the SOAP server as text in XML format (Extensible Markup Language).

Benefits of the SOAP protocol

The main benefit of the SOAP protocol is that it is based on two standard protocols:

- XML for the structure of the messages,
- HTTP for the transport of data.

For this reason, the SOAP protocol is independent from the operating systems and from the programming languages. The SOAP protocol promotes interoperability.

Furthermore, the use of the HTTP protocol (for data transfer) allows the SOAP protocol to go through firewalls without any problem.

17.1.1 WinDev/WebDev and the SOAP protocol

WinDev and WebDev allow you to create a **SOAP client application**. This application uses the SOAP functions of WLanguage to make the SOAP server run procedures and to retrieve their result.

Furthermore, WinDev allows you to create a **SOAP server application**. This application includes procedures that will be started by the client application.

17.1.2 Example

WinDev is supplied with an example that uses the SOAP functions: WD Using SOAP.

17.2 Running procedures on a SOAP server

Several SOAP functions allow you to manage the execution of procedures on a SOAP server from your WinDev applications or from your WebDev sites.

17.2.1 Principle

All the parameters required to run a SOAP procedure are supplied in the documentation of the SOAP server.

To run a procedure on a SOAP server:

1. Initialize the structure of the parameters of the procedure to run (see "structure of a SOAP procedure").

Note (WinDev only): To perform additional checks on the SOAP server, add a procedure header (**SOAPAddHeader**).

2. Run the procedure (**SOAPExecute** or **SOAPRunXML**). The following operations are automatically performed:

- connect the current computer and the SOAP server,

- transmit the parameters of the procedure to the SOAP server,
- check the header if necessary,
- run the procedure,
- send the result or the error of the procedure to the current computer.

3. Check the result of the procedure.

The result of **SOAPRun** or **SOAPRunXML** is used to find out whether the connection was successfully established.

If the connection was not established, **ErrorInfo** returns the error details.

If the connection was established, check the result of **SOAPGetResult**:

- If the result differs from an empty string (""), the procedure was successfully run.
- If the result corresponds to an empty string (""), the procedure was not run and/or an error was returned. To find out the error details, use **SOAPError**.

Note: currently the transfers are not secured (no encryption of the transferred data). The SOAP protocol should not be used to transfer sensitive data.

17.2.2 The SOAP structure

The following structure is used to pass parameters to a procedure run on a SOAP server:

Variable	Type	Details
SOAP.Namespace	Optional character string	"NameSpace" of the parameter
SOAP.Name	Character string	Name of parameter
SOAP.XMLParam	Optional character string	Structures of a parameter expressed in XML format. The other parameters (Value, Name, Type, NameSpace and EncodingStyle) are ignored if this parameter is specified.
SOAP.EncodingStyle	Optional character string	Encoding style ("EncodingStyle") of the parameter
SOAP.Type	Constant	Parameter Type
SOAP.Value	Any type	Value of the parameter

This structure must be used for each parameter.

For example:

```
SOAP.Value[1] = 94010
SOAP.Name[1] = "ZipCode"
```

```
SOAP.Type[1] = SOAPStringType
```

This structure is equivalent to :

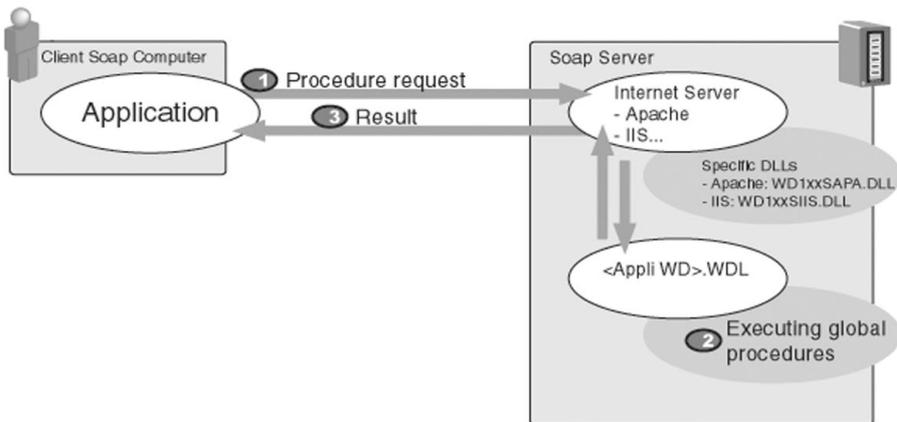
```
SOAP.XMLParam[1] = <ZipCode
xsi:type="xsd:string">94010</ZipCode>
```

17.3 Creating and installing a WinDev SOAP server application

The SOAP Server application is an application containing all the procedures to run. These procedures

are run by the SOAP client application.

17.3.1 Principle



17.3.2 How do I create an application SOAP server

To create a SOAP server application:

1. Create a "Library" type WinDev project.
2. Create the global procedures. Each global procedure can be run by the SOAP client. **Caution:** the server application must have no interface (no window, no report, no trace window or dialog box). This application is also limited by the rights granted to the Web server.
3. Enter (if necessary) the initialization code and the closing code of the project. These two sections of code will be respectively run when loading and unloading the library on the server.
4. Generate your project's library (.WDL). This file contains all the procedures of the project.

Note: In the SOAP client application, to specify the name of the library used (WDL), all you have to do is specify this name in the **NameSpace (SoapRun)**.

Caution: The transfers are not secured at this time (the transferred data is not encrypted). The SOAP protocol should not be used to transfer sensitive data.

To run the test of your SOAP server application on the development computer:

1. Generate the library (.WDL file) of your Server application (on the "Project" pane, in the "Generate" group, click "Generate").
2. Configure the Internet server installed on your computer for your WinDev application (configuration for Apache, IIS5 or Netscape iPlanet 4.1). See the online help for more details.
3. Configure the SOAP administrator (WDSOAPConfig.EXE application in "Data" directory of WinDev). See "Configuring the administrator" for more details.
4. Restart your Web server.
5. Run the test of your SOAP client application.

To install your application on a SOAP server:

1. Generate the library (.WDL file) of your Server application (on the "Project" pane, in the "Generate" group, click "Generate").

Caution: The WinDev DLLs must not be renamed.

2. Create the setup procedure of your application. On the "Project" pane, in the "Generation" group, expand "Setup procedure" and select "Create the setup procedure".

3. The following files must be selected for setup:

- WDSOAPConfig.EXE found in the "Data" directory of the setup directory of WinDev.
- the WinDev DLLs required to run your application. Check whether the following DLLs are selected: WD190IMG.DLL, WD190OBJ.DLL, WD190STD.DLL and WD190VM.DLL.
- the WinDev DLLs specific to the HTTP server used: WD190SAPA.DLL for Apache, WD190SIIS for IIS, etc.

4. Install the application on the server.

5. Configure the Internet server for your WinDev application (configuration for Apache, IIS5 or Netscape iPlanet 4.1). See the online help for more details.

6. Configure the SOAP administrator. See the online help for more details.

7. In order for the SOAP client to be able to contact the server, the server address used in **SoapRun** must have the following format:

```
<IP Address of Server>\ ...
Server.soap
```

Note: If several soap server applications are installed on the same computer, you must necessarily install:

- all the libraries (WDL) in the same directory,
- all the WinDev DLLs used in the same directory.

A single SOAP administrator will be installed and

17.4 SOAP functions

The following functions are used to manage the SOAP procedures:

SOAPAddHeader	Adds a header to the next SOAP procedure to run
SOAPGetResult	Returns the result of the last SOAP procedure that was successfully run
SOAPEncodeResult	Allows you to configure the format of the value returned by the WinDev SOAP server to the SOAP client program
SOAPError	Returns the error of the last SOAP procedure that failed
SOAPRun	Runs a procedure on a SOAP server
SOAPRunXML	Runs a procedure on a SOAP server

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

18. XML WEBSERVICES

WinDev and WebDev allow you to import XML Web services of .Net and J2EE types into your WinDev applications and into your WebDev sites.

Furthermore, WinDev allows you to manage XML Web services using the SOAP technology directly from your WinDev applications.

18.1 Importing XML Webservices

18.1.1 Overview

WinDev, WebDev and WinDev Mobile allow you to directly import Webservices into your applications.

A Webservice is a set of entry points made available to the users in order to perform different processes. For example, a service for remote access provides the processes used to access the data. The exchanges of information with a Webservice are performed in XML format and they use the SOAP (Simple Object Access Protocol) and HTTP protocols.

From the WSDL (Web Services Description Language) description of the service, WinDev, WebDev or WinDev Mobile automatically generates WLanguage types and functions corresponding to the programming interface of the Webservice.

Note: For backward compatibility with earlier versions, you can also request the generation of a set of procedures or of a class whose methods correspond to the operations of the Webservice. However, we recommend you that use the new mechanism for automatically generation of native types.

To use the Webservice, all you have to do is use the functions generated during the import.

18.1.2 Importing a Webservice into a project

To import a Webservice:

1. On the "Project" pane, in the "Project" group, expand "Import" and select the "Import a webservice (SOAP, .Net, J2EE)" option. The wizard for importing a Webservice starts.
2. Select the WSDL description of the service to import. This description can be loaded:
 - from an Internet address,
 - from an XML file found on the current computer.
3. Validate the wizard, the Webservice is automatically added into the project explorer (in the "Webservices" branch) and it is ready to be used.

18.1.3 Updating the description of a Webservice

When a Webservice evolves (corrections, new versions, etc.), its description may also evolve.

To update the type in your project, you must:

1. Select the Webservice in the project explorer.
2. Select "Update" from the popup menu.

Note: the Webservices imported by using the mechanism for backward compatibility cannot be updated.

18.1.4 Properties of a Webservice

Properties of a Webservice modifiable in the editor

To modify the properties of a Webservice in the editor:

1. Select the Webservice in the project explorer.
2. Select "Description" from the popup menu.
3. The window of properties is displayed. The following properties can be modified in this window:
 - The import address of the Webservice: this is the URL used to reach the WSDL file describing the Webservice.
 - The user name and the password used to import the WSDL.

Note: The properties modifiable in the editor and the properties modifiable by programming have no link

Properties of a Webservice modifiable by programming

To address a Webservice by programming, all you have to do is use the name of the Webservice as it is displayed in the project explorer.

Note: you have the ability to perform a drag and drop from the project explorer to the code editor to insert the name of the Webservice.

The following properties can be modified by programming:

Name	Effect
Address	<p>This property is used to replace the call address of the Webservice described in the WSDL by another URL.</p> <p>This property has the following format : "http://server:port/webbservice_path".</p> <p>For example, for a Webservice generated with WinDev and deployed on a WebDev Application Server, it is the URL of the "awws" file.</p> <p>Notes :</p> <ul style="list-style-type: none"> • The modification of this property replaces all the URL described in the WSDL. • If this property receives an empty string, the URLs described in the WSDL will be used again.
User	<p>This character string will be used as user name to perform the authentication during the HTTP requests to the Webservice.</p>
Password	<p>This character string will be used as password to perform the authentication during the HTTP requests to the Webservice.</p>

Note: If a user name and password are specified, the authentication of the HTTP requests will be done by using the "Basic" authentication schema, in which the parameters are in clear in the HTTP request. We advise you to use HTTPS requests if the authentication is required.

18.1.5 Using a Webservice imported into the project

To use the set of procedures of the Webservice, all you have to do is call the functions automatically generated by the import.

The types of variables automatically declared when importing the WSDL offer several properties:

Name	Effect
Exist	<p>True if the type of variable exists in the response of the Webservice. False otherwise.</p>
Occurrence	<p>Number of elements of this type in the response of the Webservice.</p> <p>A Webservice can return arrays of variables.</p> <p>The property named ..Occurrence is used to find out the size of the array and the [] operator is used to access the array elements.</p>
Type	<p>Name of the variable type. This property is used when a Webservice is likely to return responses of different types.</p>
Value	<p>Value of the variable.</p> <p>This property is accessed by default when only the name of the variable is used, for example :</p> <pre>myWebservice.RequestVariable = EDT_Value</pre> <p>is equivalent to :</p> <pre>myWebservice.RequestVariable..Value = EDT_Value</pre>

Advanced operations of the XML stream of the Webservice

In some cases, you may have to handle the XML data stream exchanged with the Webservice. Some Webservices ask for example to add headers into their XML stream to allow the authentication or return meta information in the response headers.

The following functions can be used to respond to these particular requests:

- **SOAPPrepare:** This function builds the request to the Webservice for a given function and parameters but does not send it.
- **SOAPAddHeader:** This function is used to add custom headers into a Webservice call.
- **SOAPGetHeader:** This function is used to read again the information found in the header of the Webservice response.



- **SOAPAddAttribute:** This function is used to declare the additional attributes (not found in the WSDL) on a Webservice variable automatically generated. It is used in advanced programming when the WSDL returned by the Webservice does not exactly correspond to the expected type.

Special case: the Webservice returns a result in a type not recognized by WinDev/WebDev/WinDev Mobile

The types of variables available in WinDev and the ones available in a SOAP Webservice can differ.

The simple types (boolean, integer, etc.) and the complex types (datetime, duration, structures, arrays of simple types and array of structures, nested structures, etc.) used in the Webservice are automatically converted into the WLanguage format (and conversely) when the service is imported into a project. The Array types are also supported.

The more evolved types (classes, advanced types of WLanguage, etc.) are processed as character strings in the WLanguage code. These character strings contain the XML code corresponding to the type of variable returned by the Webservice and to its content.

Therefore, if a Webservice returns a result as a class instance, this result will be processed in the procedure as a character string in XML format. Then, you will have to process this character string (in WLanguage) in order to extract the requested information. See the **XML functions** for more details.

Note: If the Webservice returns a structure, the name of the members of the returned structure is case sensitive.

18.1.6 Distributing a WinDev application that uses a Webservice

To distribute an application that uses a Webservice, all you have to do is include the file describing the Webservice (.wsdl file) in the library of the application.

In order for the application to be able to run the Webservice, the end-user computers must have an access to Internet.

Note: Before distributing an application that uses a Webservice, we recommend that you check the user license and the use rights of this service (case of paying services).

18.2 Generating an XML Web service

18.2.1 Overview

WinDev and WebDev allow you to directly generate Webservices that use the SOAP technology. These Webservices can be used in the WinDev, WebDev, WinDev Mobile projects or in any other language that supports the SOAP protocol.

From the WinDev/WebDev project corresponding to the Webservice, WinDev/WebDev will automatically create:

- A library (.AWWS file). This library will contain all the procedures of your Webservice. These procedures can be used by any application that uses your Webservice.
- A .WSDL file. This file contains the description of the Webservice in WSDL format (Web Services Description Language). This file contains the description of the methods and structures of the Webservice.

To make your Webservice available, these files must be installed on a WebDev Application Server.

Reminder: A Webservice is a set of entry points made available to the users in order to perform different processes. For example, a service for remote access provides the processes used to access the

data. In most cases, the exchanges performed with a Webservice use the XML format and the HTTP protocol

Note: for compatibility purposes, the XML Web services of previous versions have been kept. However, we recommend that you use the new model of Webservice.

The following differences can be noticed between the former XML Web services and the Webservices:

Former XML Web services	Webservices
Deployed on Apache or IIS with an ISAPI module.	Deployed on a WebDev Application Server (compatible with all the Web servers).
Can only be deployed with a setup by physical media on the server directly.	Can be deployed by physical media or remotely (by FTP).
Can only be deployed with a setup by physical media on the server directly.	Allows an important workload.

Limited to the versions of Windows 32 bits.	Operate on all the versions of Windows (32 and 64 bits) as well as in Linux.
---	--

Note: When converting a SOAP server generated by WinDev 14 (or earlier version) to the new Webservice model, you must replace the calls to **SOAPExecute** generated by the import of the Webservice in the WinDev project by the calls to the new convention. See the online help for more details.

18.2.2 How do I make a Webservice available?

To make a Webservice available, you must:

1. Generating a Webservice.
2. Running the test of the Webservice.
3. Deploy the Webservice on a WebDev Application Server.

18.2.3 Generating a Webservice

To generate a Webservice:

1. Create a WinDev project of Webservice type or a Webservice configuration in an existing project.
2. Create the global procedures. Each global procedure can be run by the WinDev Application Server.

Caution: The Webservice must have no GUI (no window, no report, no trace window or dialog box). It is also limited by the rights defined on the Web server ("Internet guest" account usually in Windows).

3. Enter (if necessary) the initialization code and the closing code of the project. These two codes will be respectively run when loading and when unloading the library of the Webservice.
4. On the "Project" pane, in the "Generation" group, click "Generate". The wizard for generating the Webservice starts.
5. Select "Deploy on a WebDev Application Server". For backward compatibility, you also have the ability to choose "Deployment in ISAPI mode". See the documentation supplied with WinDev 14 for more details.
6. The next screen allows you to select the elements that will be included in the library. The library and the current project will have the same name. You can:

- **Add elements**

Any type of file can be added to the list of elements inserted into the library: images, text files, ...

- **Delete elements**

The corresponding files and their dependencies will be ignored in the library.

- **Create the library from an existing library description (*.WDU file)**

When creating a WinDev library, a ".WDU" file with the same name is automatically created. This file contains all the references of the elements included in the library.

7. End the wizard to generate the Webservice.

The deployment wizard is automatically started thereafter.

18.2.4 Deploying a Webservice

In order to be used, a Webservice must be deployed on a WebDev Application Server. This deployment can be performed:

- From the development computer.
- From a management computer (other than the development computer)
- On the WebDev Application Server, by using a setup by physical media

To deploy the Webservice in stand-alone mode on a computer, the setup by physical media can also be generated in a mode containing:

- The Webservice,
- A version limited to 10 connections of the WebDev Application Server,
- The Apache Web server.

Deployment from the development computer

To deploy the Webservice from the development computer:

1. Follow the steps of the wizard for Webservice generation.
2. In the setup wizard, select "Deploy the Webservice on a remote WebDev application server".
3. Enter the connection parameters to the application server: address, user name and password (this information is provided by the Application Server administrator).
4. Specify the deployment parameters of the Webservice:

- The deployment name: by default, it is the name of the project, you can change it to allow for the deployment of several instances of a project on the same server, several projects of the same name or if the project name does not correspond to the requested name for the Webservice.
- The name of the sub-directory that will contain the HFSQL Classic data files of the Webservice: by default, it corresponds to the name of the project but it can be changed in order for two Webservices to share the same data files

- The public address of the Webservice: by default, this address is built with the address of the Application Server
- 5. The installer interrogates the Application Server and displays the list of files to deploy (if it detects an earlier version of the Webservice, it only proposes the files to update). The check boxes found in the first column of the table of files allow you to modify the list of files to deploy if necessary.
- 6. Configure the automatic modification of the Webservice data.
- 7. Specify the parameters for running the Webservice such as:
 - Maximum number of simultaneous connections.
 - Maximum number of simultaneous connections from a given IP address.
 - Maximum duration of a request to the Webservice.
 - You can also specify that a delayed deployment will be performed.
- 8. Specify whether the Application Server must generate statistics when accessing the Webservice and the generation directory of statistics. The statistics can be viewed via **WDStatistics**.
- 9. End the wizard to start the setup. A window displays the progress of the operation and the possible error messages.

Deployment from a management computer

To deploy the Webservice from a management computer other than the development computer, a setup package must be generated by the wizard for setup creation and it must be deployed by WDDeploy.

To perform the deployment:

1. Follow the steps of the wizard for Webservice generation.
2. In the setup wizard, select "Create a remote setup package".
3. You can choose a deployment profile or give this choice to the user of WDDeploy. A deployment profile groups the information about the Application Server to use: server address, authentication information, etc.
4. If you have chosen to use a profile, enter or modify the connection parameter to the Application Server: address, user name and password (this information is provided by the Application Server administrator).

5. Specify the deployment parameters of the Webservice:

- The deployment name: by default, it is the name of the project, you can change it to allow for the deployment of several instances of a project on the same server, several projects of the same name or if the project name does not correspond to the requested name for the Webservice.
 - The name of the sub-directory that will contain the HFSQL Classic data files of the Webservice: by default, it corresponds to the name of the project but it can be changed in order for two Webservices to share the same data files.
 - The public address of the Webservice: by default, this address is built with the address of the Application Server.
6. The wizard displays the files that will be deployed. You have the ability to add or remove files.
 7. Configure the automatic modification of the Webservice data.
 8. Specify the parameters for running the Webservice such as:
 - Maximum number of simultaneous connections.
 - Maximum number of simultaneous connections from a given IP address.
 - Maximum duration of a request to the Webservice.
 - You can also specify that a delayed deployment will be performed.
 9. Specify whether the Application Server must generate statistics when accessing the Webservice and the generation directory of statistics. The statistics can be viewed via **WDStatistics**.
 10. Finally, choose the directory where the setup package of the Webservice will be generated as well as the file name.

To install the Webservice:

1. Start the WDDeploy tool from an administrator machine.
2. Load the deployment package of the Webservice.
3. Enter or modify the parameters of the Application Server and the deployment parameters of the Webservice.
4. Click the "Deploy" button to trigger the deployment of the Webservice. A window displays the progress of the operation and the possible error messages

Deployment by physical media

To generate a setup by physical media of the Webservice:

1. Follow the steps of the wizard for Webservice generation.
2. In the setup wizard, select:
 - "Create a setup of the Webservice via physical media" to be able to install the Webservice on an existing WebDev Application Server.
 - "Create a setup of the Webservice via stand-alone media" to be able to include the WebDev Application Server (the limited, 10-connection version) and the Web server in the setup
3. The wizard displays the files that will be deployed. You have the ability to add or remove files.
4. Configure the automatic modification of the Webservice data.
5. For a stand-alone setup, you have the ability to choose the Web server that will be included in the setup.

6. The wizard displays a pane used to select the installer. The default installer (WBSSetup) is supplied with the sources and it can be customized.

7. You can choose the directories where the files of the Webservice as well as the data files (for a database in HFSQL Classic format) will be deployed. By default, these elements will be associated with an account of the WebDev Application Server.

8. Select the languages proposed to perform the setup. WBSSetup is supplied in French and in English. To perform setups in other languages, all you have to do is translate it via WDMsg (not supplied with the product).

9. Finally, choose the directory where the setup of the Webservice will be generated.

To install the Webservice

1. Copy the directory generated by the setup wizard onto the WebDev Application Server (or any other computer for a stand-alone setup).

2. Starting the setup program:

INSTALL.EXE.

19. XML

19.1 Managing XML documents

19.1.1 Overview

WinDev proposes several functions used to handle the XML code. The XML code used must comply with the XML standard. See a documentation specific to XML for more details.

Reminder: XML is both a standard and a language derived from HTML (Web pages, Internet, ...) that allows you to structure a document containing data.

The XML code is used for example:

- in the information systems (Intranet, ...).
- in the dialogs with the Web services. See “Generating an XML Web service”, page 307 for more details.

19.1.2 Definition

XML is a language containing tags and attributes (called "elements" in this help).

The structure of the XML code corresponds to a tree structure: information is organized in a treelike way.

An XML code (or content) corresponds to:

- the content of an XML file.
- a response from an XML Web service.

The WinDev XML functions allow you to:

- Read, find and analyze the content of an XML file.
- Analyze the answers received from XML Web services.
- Import data (*HImportXML*).

19.1.3 Principle

Two methods can be used to handle an XML document:

1. Using a string variable to store the XML document. Then, the XML document can be handled by the different XML functions.
2. Using an `xmlDocument` variable. This method

allows you to use a XSD file.

19.1.4 Using a string variable

To handle an XML code:

1. Store the XML code in a string variable. This code can come from an XML file or from an XML Web service.

2. Create an XML document (*XMLDocument*). This document is stored in memory and it contains the entire XML code to use.

3. Handle your XML document with the *WLanguage* functions. You have the ability to:

- Browse the XML code (*XMLFirst*, *XMLNext*, *XMLLast*, *XMLPrevious*, *XMLParent*, *XMLChild*, *XMLRoot*).
- Perform searches (*XMLFind*, *XMLCancelSearch*, *XMLExtractString*).
- Retrieve information about the elements (tags or attributes) found in the document. (*XMLElementType*, *XMLElementName*, *XMLParentName*, *XMLData*).
- Modify the XML document (*XMLAddAttribute*, *XMLAddChild*, *XMLInsertDocument*, *XMLInsertElement*, *XMLModify*, *XMLRename*, *XMLDelete*).
- Run an XPath query in an XML document (*XMLExecuteXPath*, *XMLPosition*, *XMLResult*).
- Convert a character string in ANSI format to a character string in XML format and conversely (*TextToXML* or *XMLToText*).

4. Save the modifications (*XMLBuildString* and *fSaveText*).

5. Close the XML document (*XMLClose*).

Note: You also have the ability to create an XML document:

- with the data found in a table (*TableToXML*).
- from a part of an existing XML document (*XMLExtractDocument*).

19.1.5 Using an xmlDocument variable

The `xmlDocument` variables can be declared by indicating a sample document (XML file) or a template document (XSD file). The structure of this document will be read by WinDev/WebDev and it will propose the automatic completion on the names of nodes in the code editor.

To declare an `xmlDocument` variable with a sample document:

1. Add the XML document into the project:
 - via the project's element list.
To display this list, in the "Project" menu, in the "Project" group, click . In the window that is displayed, choose the "Add" button.
 - by Drag and Drop of the file into the "XML description" branch of the "Project explorer" pane.
 - from the "Project explorer" pane (select the "XML description" branch, then "Import an XML description file" from the popup menu).
2. The XML document is displayed in the "XML descriptions" branch of the project explorer. You have the ability to view its structure.

3. Declare the variable as follows:

```
<Variable Name> is ...
    an xmlDocument, description ...
    = <Name of the document>
```

<Document Name> can correspond to the name of the sample document (with or without extension) or to the name of the template document (with or without extension).

Notes:

- This declaration can be obtained automatically by dropping the name of the XML document from the project explorer.
- When using a sample document, you also have the ability to use the following syntax:

```
<Variable Name> is ...
    xmlDocument
<Variable Name> = ...
    xmlOpen(<Document Name>)
```

4. You now have the ability to access the nodes of the variable by their names. These names are automatically proposed by the mechanism for automatic completion of the code editor.

Note: `XMLSave` is used to save the XML document. If an XSD file was used, it will be automatically taken into account.

19.2 Managing the XSD

19.2.1 Overview

WinDev, WinDev Mobile and WebDev propose several functions for handling the XML code. The XML code used must comply with the XML standard. See a documentation specific to XML for more details. See the XML help page for more details.

WinDev, WebDev and WinDev Mobile allow you to import files in XSD format.

An XSD file contains the description of the XML file of the same name. Knowing the structure of an XML document enables you to check the validity of this document. The description language for the content of an XSD document is also in XML format.

An example of XSD file::

```
<?xml version="1.0" ...
    encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
<xsd:element name="person"
<xsd:complexType>
    <xsd:sequence>
        <xsd:element name="last
name" type="xsd:string"/>
```

```
<xsd:element name="firstname"
type="xsd:string"/>
<xsd:element name="dob"
type="xsd:date"/>
    <xsd:element name="email "
type="xsd: string" minOccurs=0
maxOccurs= unbounded />
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

Followed by a valid XML file:

```
<?xml version="1.0" ...
    encoding="UTF-8"?>
<person xmlns:xsi="http:
//www.w3.org/2001/XMLSchema- ...
instance" xsi:noNamespace...
SchemaLocation="person.xsd">
    <lastname>de Latour</lastname>
    <firstname>John</firstname>
    <dob>1967-08-13</dob>
    <email>test@free.com</email>
</person>
</WL code>
```

19.2.2 Importing an XSD file into a project

To import an XSD file into a project:

1. Select "XML Description" from the "Project explorer" pane.
2. Right click and select "Import an XML description file".
3. Select the XSD file of the description to import.
4. Validate. The XSD is automatically added into the project explorer (in the "XML description" branch). It is ready to use.

19.2.3 Using an imported description in the project

To use the description of the XML document, all you have to do is use the variables automatically generated by the import.

1. Declare an XML document in the format of the imported description. For example, if the imported

description is named 'person'

```
cMyDoc is xmlDocument ...
    <description="person">
```

2. Initialize the different variables that are included in the description of the XML document. For example, if the document includes the last name and the first name of a person.

```
cMyDoc.person.lastname="JOHNSON"
cMyDoc.person.firstname="JOHN"
```

Notes:

- Each element is separated by a dot ..
 - A help is proposed when entering the names of the variables.
3. Save the content of the XML document with XMLSave.

```
XMLSave (cMyDoc, ...
    "ListPersons.xml"
```

19.3 Functions for managing the XML documents

The following functions are used to manage XML documents:

TextToXML	Converts a character string in ANSI format into a character string in XML format
TableToXML	Creates an XML file from the data found in a table (browsing table or memory table)
XMLAddAttribute	Adds an attribute into an XML document
XMLAddChild	Adds a child tag into an XML document
XMLCancelSearch	Cancels the search that was started by XMLFind
XMLFindNamespaceBy-Name	Finds an XML namespace from its name in an XML node and in the parents of this node
XMLFindNamespace-ByURI	Finds an XML namespace from its URI (Universal Resource Identifier) in a node and in the parents of this node
XMLBuildString	Retrieves and formats the content of an XML document (created by XMLDocument) and modified by the XML functions
XMLLast	Positions on the last element of the current level in the tree structure or the last tag of the hierarchy
XMLDocument	Creates or defines a new XML document
XMLValidDocument	Validates an XML document from an XSD schema
XMLData	Returns the value of the current element or tag
XMLWrite	Writes the value of an XML element or the value of the attribute of an XML element
XMLOut	Allows you to find out whether the current position is valid or if the tag on which we want to be positioned is outside the file
XMLExecuteXPath	Runs an XPATH query in an XML document
XMLExtractString	Extracts data from an XML code
XMLExtractDocument	Creates a new XML document from an existing XML document
XMLChild	Positions on the first child element of the current element or the first child tag of the current tag
XMLChildExist	Indicates whether the current tag contains child elements corresponding to the sought type (tag, attribute)

XMLInsertDocument	Inserts an XML document into another XML document
XMLInsertElement	Inserts an XML element (tag or attribute) into an XML document
XMLRead	Returns the value of an XML element or the value of the attribute of an XML element.
XMLModify	Modifies the content of the current element found in the XML document
XMLNamespace	Returns the prefix of the namespace used for the current tag in an XML document
XMLNamespaceURI	Returns the URI for defining the namespace used for the current tag in an XML document
XMLValidNode	Validates an XML node, its children and its attributes from its description in the XSD schema linked to the XML document.
XMLElementName	Returns the name of the current element or tag
XMLParentName	Returns the name of the parent element of the current element or the name of the parent tag of the current tag
XMLOpen	Loads an XML document from a file, a URL or a string in an <code>xmlDocument</code> variable
XMLParent	Positions on the parent element of the current element or on the parent tag of the current tag
XMLPath	Returns the current position in the document in XPATH format
XMLPrevious	Positions on the previous element of the current level in the tree structure or on the previous tag
XMLFirst	Positions on the first element of the current level in the tree structure or on the first tag found
XMLRoot	Positions on the root tag of the XML document
XMLFind	Performs a search in an XML document
XMLRename	Renames the current element (tag or attribute) in an XML document
XMLResult	Returns the result of a calculation XPATH query. This query was run by <i>XMLExecutePath</i>
XMLRestorePosition	Restores the context of an XML document (previously saved by <i>XMLSavePosition</i>). The filter used when saving the position can be restored.
XMLSave	Saves an XML document in a file.
XMLSavePosition	Stores the current position in an XML document.
XMLNext	Positions on the next element in the current level of the tree structure or on the next tag of same level
XMLDelete	Deletes the next element in the current level of the tree structure
XMLClose	Closes an XML document that was created by <i>XMLDocument</i>
XMLFound	Allows you to find out whether an element was found during the browse
XMLElementType	Returns the type of the current element or the type of the value for the current tag
XMLToText	Converts a character string in XML format into a character string in ANSI format

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev, WinDev Mobile).

20. .NET ASSEMBLIES

20.1 Overview

20.1.1 Definition

The .NET technology corresponds to the new development platform of Microsoft. This .NET platform greatly simplifies the development of distributed-object applications whose modules communicate via Internet.

This chapter only provides an overview of how to use the .NET technology in WinDev. See a specific documentation for more details.

20.1.2 WinDev and .NET

You have the ability to create .NET programs (or .NET assemblies) from a WinDev application.

These .NET assemblies are directly created from the WinDev classes (with all the benefits from the code editor, the documentation editor, ...).

You also have the ability to use .NET assemblies (not created by WinDev) in a WinDev or WinDev Mobile application, or in a WebDev site:

- in a new application: specify the use of the .NET assemblies in the wizard for project creation.
- in an existing application: select the .NET assemblies to use.

This chapter presents:

- the conditions required for handling a .NET assembly.
- the creation of a .NET assembly.
- the creation of the setup program for a .NET assembly.
- how to use a .NET assembly in an WinDev application.

See the online help (keyword: ".NET assembly") for more details.

20.2 Conditions for using a .NET assembly

To create and use a .NET assembly with WinDev, you must:

- Installing the .NET Framework on the current computer.
- define the .NET security level.
- make the DLLs required to run the .NET assembly accessible.

20.2.1 Installing the .NET framework

To install the .NET framework on the current computer:

- **Method 1:** Using Windows Update
1. Open the control panel of Windows ("Start .. Parameters .. Control Panel") and select "Add/Remove Programs".
 2. Click "Add new programs".
 3. Click "Windows Update".
 4. Find the setup of the .NET Framework.
 5. Install the .NET Framework.

- **Method 2:** Download from the Internet site of Microsoft

1. Consult the Microsoft site.
2. In the "Microsoft Download Center" section, find the setup of the .NET Framework.
3. Install the .NET Framework.

20.2.2 Defining the .NET security level

To define the .NET security level:

1. Open the control panel of Windows ("Start .. Parameters .. Control Panel") and select "Administrative Tools".
2. Select "Microsoft .NET Framework Wizards". The window of the .NET wizard is displayed.
3. Click "Adjust the .NET security".
4. Select "Apply the modifications to this computer" and click "Next".
5. Select "My computer" and configure the security to "Full trust".

Note: To run a .NET program available on the local network, select the "Local Intranet" icon and set the security to "Total trust".

20.2.3 Making the DLLs required to run the .NET assembly accessible

To make the DLLs required to run the .NET assembly accessible:

1. If needed, copy the "WD190NET.DLL" library:
 - into the creation directory of the .NET assembly.
 - into the "Assembly" sub-directory of the setup

directory of Windows.

2. Copy (if necessary) the WinDev libraries:

- into the creation directory of the .NET assembly.
- into the PATH.

Reminder: The PATH list the directories in which the executable (".EXE" files) and the libraries (".DLL" files) are sought first.

20.3 Creating a .NET assembly from WinDev

Before creating a .NET assembly, all the elements required by this assembly must have been developed in a WinDev project.

To create a .NET assembly:

1. Open the WinDev project from which the .NET assembly must be created.
2. Create (if necessary) a project configuration of ".NET assembly DLL" type containing all the necessary elements.
3. On the "Project" pane, in the "Generation" group, click "Generate". The wizard for creating a .NET assembly starts.
4. Select the different WinDev elements (project, windows, classes, ...) that will be included in the .NET assembly.

The "Add" button enables you to add an element to the .NET assembly: images, text files, ...

The "Remove" button enables you to delete an element from the .NET assembly. The corresponding files and their dependencies will not be included in the .NET assembly.

The "WDU" button enables you to display the elements found in a library that was previously created.

Note: If the WinDev project (".WDP" file) is selected:

- the project analysis will be associated with the .NET assembly.
- the initialization process of the project will be run when the .NET assembly is initialized.
- the ending process of the project will be run when the .NET assembly is closed.

5. Select the WinDev classes (".WDC" files) that must be made accessible.

These classes can be handled from the application that will use the .NET assembly.

6. Configure the error message if necessary.

This message will be displayed if an error occurs when using the .NET assembly.

The "Default" button is used to configure a default error message.

7. Specify (if necessary) the information about the .NET assembly: description, version, copyright, ...

The "Advanced" button enables you to define additional properties specific to the .NET assembly.

8. Enter the name of the .NET assembly.

The "Compilation options" and "Advanced properties" buttons enable you to access the advanced options of the .NET assembly. See "Advanced options" for more details.

9. Validate.

Limitations: You cannot:

- rename an existing .NET assembly.
- include the classes of a component in a .NET assembly.

Advanced options

The "Compilation options" button allows you to select a specific C# compiler and to specify additional compilation options.

The "Advanced properties" button is used to define whether the .NET assembly must be accessible by a program that uses the COM technology. In this case, you must:

- select the different classes accessible from this program.
- create a strong-named .NET assembly. The assembly will be authenticated and it will be possible to check its origin.
- specify the ".SNK" file to use. This file contains the encryption keys used for authenticating the .NET assembly.
- specify whether the delayed signature mechanism must be used. This mechanism allows you to not authenticate the .NET assembly during its development.

20.4 Creating a .NET assembly accessible by COM

20.4.1 Overview

Before creating a .NET assembly, all the elements required by this assembly must have been developed in a WinDev project.

20.4.2 Creating a .NET assembly accessible by COM from a WinDev project

To create a .NET assembly accessible by COM, you must:

1. Create a ".snk" file.

The snk file contains the "public key/private key" pair used to digitally sign the assemblies created by a company. In most cases, the same snk file is used to sign all the products of a company.

This file must be kept secret in order to prevent a third party from signing its own assemblies with the same signature.

To generate a ".snk" file, you must use the generator supplied with the .NET SDK (sn.exe file). The following command line must be used to generate a snk file:

```
"sn.exe -k MySNKFile.snk"
```

Un fichier exemple est fourni avec WinDev, dans le répertoire "Programmes\Donnees\Example.snk" du répertoire d'installation de WinDev. This file is used to test the generation of strong-named assemblies.

Note: this file being distributed with WinDev, it does not allow you to uniquely identify the generated assemblies.

2. Modifying the code of the classes found in the WinDev project

- The class must have a constructor without argument (this is the only constructor that can be used by COM),
- The static members will not be accessible by COM,
- The overloaded methods are renamed in COM.

3. In the wizard for generating .NET assemblies, select the following options:

- Check "Accessible by COM",
- Specify the location of the snk file that was previously created.

4. Copying the generated assembly into the directory of shared assemblies of Windows:

- Copy the assembly into the C:\Windows\Assembly directory,
- Copy the wd190net.dll library into the c:\Windows\Assembly directory (otherwise the instantiation of the COM object will fail with a "file not found at the specified location" error),
- Check whether WD190VM.dll is found in the directory of PATH.

To use an assembly in a C++ program

These operations must be repeated whenever loading a GUID of interfaces:

1. Import the assembly: "regasm MyAssembly.DLL /tlb:MyAssembly.tlb".
2. Use the tlb file generated in the C++ project.
3. Recompile the C++ project.

20.5 Creating the setup program of a .NET assembly

The method for creating the setup program of a .NET assembly is identical to the method for creating the setup program of a standard WinDev application.

See the online help for more details (keyword: "Setup of a WinDev application").

20.6 Using .NET assemblies in a WinDev application

To include .NET assemblies in the current application:

1. On the "Project" pane, in the "Project" group, expand "Import" and select "Import a .Net assembly".
2. Select the .NET assemblies that will be included in the current application.

3. Validate.

The method for using a .NET assembly is identical to the method for using a WinDev class.

Reminder: The classes are visible in the "Project explorer" pane.



PART 7

Data files

10

DEVELOP 10 TIMES FASTER



PC SOFT

1. MANAGING DATA FILES

WinDev, WinDev Mobile and WebDev enable you to manage data files in HFSQL format ("WinDev/WebDev" format), in xBase format and in any database accessible by an OLE DB driver.

This chapter presents concepts for managing data files in format recognized by WinDev/WebDev.

To make reading and understanding these concepts easier, we recommend that you only read the

chapters that are relevant to you. You can always go back and read the other chapters when you want to use another feature.

We sometimes refer to programming functions. These functions are presented in details in the online help.

Note: HFSQL is the new name of HyperFileSQL.

1.1 HFSQL data file management

This section explains the mechanism for managing the HFSQL data files:

- creating a data file,
- automatic opening and closing of the data files,
- managing the keys,
- accessing the records.

xBase data file management is not discussed in this chapter, it has its own chapter, "The data files in FoxPro xBase format", page 331.

The long names:

The long names are supported: you have the ability to create data files with long names.

To create data files with long names, long names must be supported by the environment where the program will run (or by the Web server environment where the program will run) and by the disk where the data files will be created.

For example, long names cannot be used with some networks that don't support long names but can be used with Windows XP, 7, ...

1.1.1 Create a data file

To access a data file, this one must exist on disk. Otherwise, it must be created. **The physical data file is created:**

- **by programming.**
- if the "Automatically create the data files if not found when they are opened" options is selected in the "File" tab of the project description. Reminder: To display the project description, on the "Project" pane, in the "Project" group, click "Description".

Note: the physical file is not created by generating the analysis or by the setup procedure.

Two functions can be used to create a data file:

- **HCreate** to create an empty data file (and possi-

bly the index and memo files). If the data file already existed, it is overwritten by an empty data file (it is re-created).

- **HCreationIfNotFound** to create the data file if it does not exist or open the data file if it exists. The data file and possibly the index and "memo" files are created empty.

The data file is created:

- Under the name (called "Name on disk") defined in the analysis (except if **HChangeName** was called before **HCreation**).
- In the directory defined in the analysis, except if **HSubstDir** or **HChangeDir** was called before **HCreation** or **HCreationIfNotFound**.

When the data file is created, it is empty: it only contains a header. This header contains information specific to the data file. It is used by WinDev/WebDev for managing the file.

1.1.2 Opening and closing data files

The data files are automatically opened and closed. This process is described here for information purpose.

WinDev/WebDev manages the list of data files used at a given time by the program.

When a function applies to a data file, it the file is not part of the list of used file, the data file is automatically opened.

An unlimited number of files can be opened at the same time.

If a closed data file is used later in the program, it will be automatically re-opened according to the same process.

This is true for any new data file used.

1.1.3 Managing HFSQL files larger than 2GB

By default, the HFSQL engine allows you to manage large data files (up to 2 gigabytes).

The data files exceeding 2 gigabytes require a specific configuration in the data model editor and in the hardware.

Indeed, to manage the data files exceeding 2 gigabytes, you must use:

- An operating system that supports the files exceeding 4 gigabytes. The following operating systems do not support files exceeding 4 gigabytes: Windows 95, Windows 98, Windows Me.
- A file system that supports the files exceeding 4 gigabytes (NTFS for example in Windows)

WinDev: This configuration is required both on the computer where the data files are installed and the computers that access these files. Therefore, a computer running Windows 98 cannot access a data file exceeding 2 gigabytes installed on Windows 2000.

WebDev: This configuration is required both on the Web server and on the data server.

WinDev Mobile: The memory capacity being limited on a Pocket PC, this option is not supported.

Configuring the data files

To manage the data files exceeding 2 gigabytes:

1. Display the data file description ("Description of data file" from the popup menu).
2. In the "Details" tab, select the "Manage files over 2 GB" option.

Caution: These data files can only be used from a computer running an operating system that supports the files exceeding 4 gigabytes.

Using data files exceeding 2 GB

Dynamic description of data files

During the dynamic description of a data file (via the HFSQL properties and *HDescribeFile*), *..Huge-File* enables you to configure the management mode of the data file size.

Caution: Reindexing data files exceeding 2 GB can be a long operation.

1.1.4 Managing the keys

An item can be defined as unique key, multiple key or not a key when describing a data file.

If the item is described as key (unique or with duplicates), the item can be used as a search criteria for

the data file.

Depending on the item type, the key can be a text key or a numeric key.

General definitions

- The text, date and time items, when defined as keys, are called *text keys*.
- When they are defined as key, the integer (short, long, etc.), real (simple, double, etc.), radio button, combo box and list box items are called *numeric keys*.
- A *composite key* is a key made of text and/or numeric items.
Note: a composite key can be made of the identifier of the data file.

Unique keys and keys with duplicates

Definitions

A key is *unique* when the value of the item is unique for each record in the data file.

A key is called a *duplicate* key when the value of the item can be the same for several records in the data file.

Checking the uniqueness of a key

When *HSetDuplicates* is enabled:

- Automatically check the uniqueness of the key (WinDev only). When a key with duplicates appears after the addition or the modification of a record, a message is automatically displayed, allowing the user to modify the item causing a problem. This automatic management mode can be customized. See the online help for more details.
- Check by programming. All you have to do is check *HErrorDuplicates* after any function that may trigger the creation of duplicates.

```
// Add a record (WinDev)
FileToScreen
HAdd(CUSTOMER)
IF HErrorDuplicates() THEN
Info("A record already exists", ...
    "It is not added")
END
```

```
// Add a record (WebDev)
FileToPage
HAdd(CUSTOMER)
IF HErrorDuplicates() THEN
Info("A record already exists", ...
    "It is not added")
END
```

Note: If the modified or added key is not unique, the functions that may trigger the creation of duplicates are: *HAdd*, *HModify* or *TableSave*.

If **HErrorDuplicates** is not called, the error 10 (duplicate error) will be automatically generated during the next call to a function for file management. This error cannot be ignored: all the data files are closed and the program execution is canceled.

The management of duplicates is enabled by default.

If the management of duplicates is disabled, the check of the key uniqueness is not managed.

If the management of referential integrity is enabled (**HSetIntegrity**), the management of duplicates is automatically enabled.

Identifier: WinDev/WebDev proposes the automatic management of a file identifier (unique key). The identifier management is presented in "Managing an identifier", page 339.

Search on text keys

Definitions

- The *search criterion* is the name of the key (item) used for the search.
- The *search argument* is the value sought for this key.

Sought Value	Options	Current rec	HFound returns	Explanations
Durand		1	True	Durand exists.
Dupuis		1	False	Dupuis does not exist. Positions on the first greater value (Dupont).
Dupon	hGeneric	8	True	Dupon does not exist but the search is generic and a customer named Dupond343 exists among others).
Dupon		8	False	Dupon does not exist.
Martin		4	True	Martin exists.

Note: You have the ability to define specific search criteria when the key is described in the data model editor. You can define whether the item will be:

- Sensitive to the case (uppercase/lowercase characters): For example, PC SOFT will differ from PC Soft.
- Sensitive to the accented characters: For example, "été" will differ from "ete".
- Sensitive to the punctuation (comma, period, ...): For example, "C.E.O." will differ from "CEO".

These different criteria will be automatically taken into account when a search is performed on the key item.

• The argument can be:

- A full value, it is called an *exact-match search*.
- A partial value, it is called a *generic search*.

Example

When a generic search is performed on "Martin" (NAME item), all the records whose Name item starts with "Martin" will correspond to the search. Therefore, the record containing "Martinez" will correspond to the search (**HFound** returns True).

When an exact-match search is performed on "Martin" (NAME item), **HFound** returns True for all the records whose NAME item exactly matches "Martin".

Record Number	Content
8	Dupond
2	Dupont
5	Dupont
1	Durand
3	Durand
6	Durand
4	Martin
7	Martin
9	Martinez

Search on numeric keys

Encoding of numeric keys

The numeric items (int, long int, real, ...) are stored in the data files according to the binary format specific to the language used.

However, in the index file, the numeric keys are encoded differently (to simplify and speed up the searches).

Example: If the encoding of an integer is the one of the language, the negative value is "lexicographically" greater than the positive value (the first bit, the sign bit, is set to "1"). In the index file, the encoding is used to restore the order, which means that

the code of a negative value will be "lexicographically" less than the code of a positive value.

Managing the numeric keys

Important: A search performed on a numeric key is always performed as an exact-match search, which means that the search is performed on the full size of the item.

Example: The zip code can be a text item or a long integer item. Therefore, the key will be a text key or a numeric key.

For a text key, the search on the first two characters will allow a search on the state. This type of search will not be possible for a numeric key.

Tips

- The text keys are easier to use than the numeric keys but they occupy more space in the data file and in the index file.
- The numeric keys of real type should be avoided. The rounding specific to the encoding of the reals in the language may disturb the search.

1.1.5 Managing the composite keys

A composite key is a key item containing several other items. These items can be numeric items or text items.

A composite key is used to simplify the searches performed on several criteria.

The composite keys are binary items. Their value cannot be directly displayed (in a trace or in a control).

Creating a composite key

A composite key is directly created in the data model editor.

To create a composite key in a file description:

1. Display the description of the file items ("Description of items" from the popup menu).
2. Click the icon . A screen allowing you to build the composite key is displayed.
3. The list of items found in the data file is displayed in the table on the left. Double-click the items that must be included in the composite key. These items are displayed in the table on the right.
4. Reorganize (if necessary) the items included in the composite key. **Caution:** the order of items is very important because it defines the sort order. For example, the "Name + State" composite key will be sorted on the name then on the state.

5. Specify the search direction and the search parameters for each key component.

6. Validate. The composite key is displayed in the list of data file items.

Composite key and link

The composite keys can be used in the links between files. When describing the analysis, the composite key found in the linked file does no longer appear as a composite key but as a **binary key**. You will not be able to access the different components of the composite key in the linked file.

Note: The referential integrity is managed on a "composite key" link key.

Value of a composite key

A composite key is stored as a binary string. Its value cannot be displayed (neither in a control nor in the debugger, ...).

Adding a record containing a composite key

When adding or modifying a record containing a composite key, the value of the composite key is automatically defined according to the values of the different key components. No specific programming is required.

Adding a record containing a composite key into a linked file

When a record containing a composite key is added into a linked file, the value of the key must be built. Indeed, in the linked file, the composite key is not considered as being a composite key but as being a binary key. Therefore, a value must be assigned to it.

This value can be assigned:

- directly. For example, a record was added into the Customer file. To add the value of the key into the linked file, all you have to do is perform a direct assignment:
`Link.NameDate = Customer.NameDate`
- via **HBuildKeyValue**. This function is used to build the value of the composite key from its components.

Building the value of a composite key to implement a search or a filter

When a filter or a search is implemented on a composite key, the value of the composite key must be defined (to define the lower bound and the upper bound of the filter for example).

Several solutions can be used to build the value of a composite key:

- Using **HBuildKeyValue**
- Using an array of values

Using HBuildKeyValue

To do so, the relevant file, the name of the key and the values of the components must be specified in **HBuildKeyValue**.

Example: To build the name of the "NAMEDATE" composite key corresponding to "CUSTOMER-NAME+DATE_ENTERED" of the Customer file, use the following code line:

```
HBuildKeyValue (Customer, ...
    NAMEDATE, "MARTIN", "03/11/85")
```

Using an array of values

All you have to do is specify the values of the composite key during the search or the filter.

Example: To create a filter on the "Customer" file, on a composite key made of the name and the date, all you have to do is write:

```
HFilter (Customer, DATENAME, ...
    ("SMITH", "03/11/85"))
```

Properties for managing the composite keys

The following properties are used to manage the composite keys by programming:

Binary	Identifies a binary item (composite key, binary string, binary memo)
CompositeKey	Checks whether the specified item is a composite key
Component	Returns the name of the nth component of a composite key
KeyExpression	Returns the expression of a composite key
NbComponent	Returns the number of components in a composite key

The composite keys can be used to perform:

- exact-match searches,
- generic searches,
- filters.

See the online help for more details.

1.1.6 Accessing the data file items

File record and variable

A record is made of items. The items are defined during the description of the data file.

In the programs, each item is handled by a variable (called file variable) with a defined name and type.

The syntax of this variable includes the logical name of the data file and the name of the item

```
<FileName>.<ItemName>
```

Example: the variable of the "CUSTNAME" item (customer name) of the "CUSTOMER" file is named CUSTOMER.CUSTNAME

Note: The file variables must not be declared. All the record variables are automatically described and declared by WinDev/WebDev.

Case of the composite keys

A composite key can be directly read without going via the items included in it. All you have to do is read the variable like any item.

This operation is not recommended.

For example, NAMEDATE is a composite key made of the NAME and DATE items.

```
Info (INVOICE.NAMEDATE)
// Displays the composite key value
```

Note: Attempting to write into a composite key has no effect. Indeed, the composite keys are rebuilt during each write operation.

Pointed record and record loaded in memory

The notions of pointed record and record loaded into memory are fundamental. It is important to understand them in order to properly use the functions for managing the data files.

The *pointed record* corresponds to the last record read according to the specified key. The notion of pointed record is relative to a key.

The *record loaded* in memory corresponds to the file record whose values are currently loaded in the program variables.

At a given time, there can only be a single record loaded in memory for a data file and a single pointed record for the same data file.

However, the record pointed by the index and the record loaded in memory can be different.

Special case: when the management of contexts is enabled in the windows or in the pages, each context is used to handle different records of the same data file. There will be a single record loaded in memory per context.

Reading a record, automatically or upon request, initializes the different variables of the file with the record loaded in memory.

Writing a record means writing the data file variables into the file when saving the record loaded in memory.

Note: The following table identifies the record on which the functions that position on record operate: the record loaded in memory or the pointed record. The functions not listed in the table use no record.

Functions positioning a record	Record loaded in memory	Pointed record
HAdd	x	
HChangeKey		x
HLast		x
HWrite	x	
HRead	x	
HReadLast	x	x
HReadPrevious	x	x
HReadFirst	x	x
HReadSeek	x	x
HReadSeekLast	x	x
HReadSeekFirst	x	x
HReadNext	x	x
HModify	x	
HPrevious		x
HFirst		x
HSeek		x
HSeekLast		x
HSeekFirst		x
HCross	x	
HRestorePosition		x
HSavePosition		x
HNext		x
HDelete	x	

Reading a record

1. Read functions

HReadSeek, **HReadSeekFirst**, **HReadSeekLast**, **HReadFirst**, **HReadLast** are used to read a record. These functions load a record into memory and they point to a record in the index.

HRead is used to read a record without pointing the record in the index.

HReadNext and **HReadPrevious** are used to read a record if this record was pointed by **HReadSeek**, **HReadFirst** or **HReadLast**.

Important: **HRead** is used to read a record according to its record number, the pointed record is not initialized. Therefore, **HReadNext** and **HReadPrevious** cannot be called immediately after **HRead**.

However, the pointed record can be initialized by **HChangeKey** after **HRead**: **HReadNext** and **HReadPrevious** can be used to browse the data file on the key positioned by **HChangeKey**.

2. Read operation according to a key

HReadSeek, **HReadSeekFirst**, **HReadSeekLast**, **HReadPrevious**, **HReadNext**, **HReadFirst**, **HReadLast** are used to read a record according to the key that was specified in parameter. The same data file can be read according to multiple keys.

Seeking a record

HReadSeek, **HReadSeekFirst**, **HreadSeekNext**, **HReadPrevious**, **HReadNext**, **HSeek**, **HSeekFirst**, **HSeekLast**, **HPrevious**, **HNext** are used to find a record:

- **HReadSeekXX** (or **HSeekXX**) starts the search.
- **HReadNext**, **HReadPrevious** (or **HNext**, **HPrevious**) access the next or previous record.

If the record corresponding to the search is found, **HFound** returns True. Otherwise, it returns False.

When seeking a record according to a key, you must check **HFound**.

Browsing data files

1. Principle

The notions of next record and previous record are relative to the pointed record for a specific key.

To be able to read the next or previous record, the pointed record must necessarily be initialized. Otherwise, the error 19 will be generated.

The following functions initialize the pointed record: **HReadSeek**, **HReadSeekFirst**, **HReadSeekLast**, **HReadFirst**, **HReadLast**, **HSeek**, **HSeekFirst**, **HSeekLast**, **HLast**, **HFirst**, **HChangeKey**.

The data files are accessible via the values of the keys sorted in ascending order. However, a data file

can be read in both directions (ascending or descending).

Caution: The sort directions of the keys as well as the search characteristics (accentuation, case and punctuation) specified in the analysis are taken into account.

The first record of a data file according to a key is the one (or one among several) corresponding to the smallest value of the key. It is reached by **HFirst** or **HReadFirst**.

The last record of a data file according to a key is the one (or one among several) corresponding to

Mechanism for reading files

Let's take a look at an example that illustrates the mechanism for reading a file according to a key.

The "PEOPLE" file includes two key items :

"CODE": customer number.

"FIRSTNAME": first name of customer.

"PEOPLE" contains the following 8 records:

Record number	CODE	FIRSTNAME
1	17	Fabian
2	12	Edgar
3	18	Charlotte
4	05	Louis
5	01	John
6	20	Lara
7	09	Franck
8	14	Mary

In this file, let's see the evolution of the record loaded in memory and the evolution of the pointed record during the program execution:

Functions	Record pointing to CODE	Record pointing to FIRST-NAME	Record in memory
Beginning of program	?	?	?
HReadFirst (PERSON, CODE)	rec.5	?	rec.5
HReadFirst (PERSON, LASTNAME)	rec.5	rec.3	rec.3
HReadNext (PERSON, CODE)	rec.4	rec.3	rec.4
HReadNext (PERSON, CODE)	rec.7	rec.3	rec.7
HReadNext (PERSON, LASTNAME)	rec.7	rec.2	rec.2
HReadNext (PERSON, LASTNAME)	rec.7	rec.1	rec.1
HReadNext (PERSON, LASTNAME)	rec.7	rec.7	rec.7
HReadNext (PERSON, CODE)	rec.2	rec.7	rec.2
Etc.

the highest value of the key. It is reached by **HLast** or **HReadLast**.

N°. Rec	Last name	First Name	Position
1	GONZALEZ	Speedy	
2	MOUSE	Mickey	
3	TALON	Achille	first according to FIRSTNAME
4	LAGAFFE	Gaston	
5	DALTON	Avrell	first according to LASTNAME
6	VAILLANT	Michel	last according to LASTNAME
7	ONEAU	Yoko	last according to FIRSTNAME
8	MALTESE	Corto	

The record found before or after the pointed record are reached by the following functions: **HReadNext**, **HReadPrevious** or **HNext**, **HPrevious**.

Example of ascending browse:

```
HReadFirst (CUSTOMER, Name)
WHILE NOT HOut ()
    PrintLabel ()
    HReadNext (CUSTOMER, Name)
END
```

2. Positioning outside the data file

After the execution of **HReadLast**, **HReadPrevious**, **HReadFirst**, **HReadSeek**, **HReadNext** and **HLast**, **HPrevious**, **HFirst**, **HSeek**, **HNext**, if the pointed record points to the last record loaded in memory, **HOut** returns *True*.

Therefore, when browsing a file, you must check **HOut** after running one of the above-mentioned functions.

3. Changing the search key

The search key can be modified by **HChangeKey**. In this case, the current record is kept even though the search key is modified.

HChangeKey can be used to find a record after being positioned in direct access on a record by **HRead**.

For example, the line:

```
HChangeKey (CUSTOMER, CUSTNAME)
```

can replace the following algorithm:

```
RecNum is int
// Stores record #
// for current Customer
RecNum=HRecNum (CUSTOMER)
// Move relatively to the CUSTNAME
key
HSeek (CUSTOMER, CUSTNAME, ...
    CUSTOMER.CUSTNAME)
WHILE NOT HOut () AND ...
    RecNum<>HRecNum ()
    HNext (CUSTOMER, CUSTNAME)
END
```

4. Browse with duplicates

In case of equality on the value of a key (duplicates, the order of the records is not significant. The first record returned will be one of the records corresponding to the sort argument.

HReadNext or **HReadPrevious** are used to get all the homonyms.

Important: No hypothesis should be made regarding real location of the record. Indeed, on a search value with homonyms, a **HReadNext** loop cannot be used to find out the order in which the homonyms were inserted.

5. Browse according to a filter

You have the ability to define a filter in order to select records. This filter can:

- **browse the data file according to the bounds** and select the records corresponding to the filter. The bounds are defined according to a data file key.
- **browse the data file according to the specified selection condition.** The search key of the data file is returned by **HFilter** according to the specified condition.

The selection of records according to the filter is automatically managed.

The filter (lower and upper bounds or selection condition) is described by **HFilter**.

The filter is automatically enabled. It can be enabled/disabled by programming with **HActivateFilter** and **HDeactivateFilter**.

Filter and composite key

Two methods can be used to create a filter on a composite key:

- Using the advanced filter functions (***HFilterStartsWith***, ***HFilterIncludedBetween***, ***HFilterIdentical***). All you have to do is specify, for each component of the composite key, the value to take into account for the filter.

This method is recommended.

See the documentation about these functions for more details.

- Using ***HFilter*** combined with ***HBuildKeyValue***. This method is presented in the paragraphs below.

Notes

- The filter can be used in external language (WinDev only).
- A single filter can be defined per file at a given time.
- A filter is enabled:
 - on the search key on which it was defined,
 - on the search key returned according to the selection condition.
- If a filter is enabled, ***HFirst***, ***HReadFirst***, ***HReadFirstLock***, ***HReadFirstNoLock*** return:
 - the first record corresponding to the filter,
 - ***HOut*** returns **True** if no record corresponds to the filter.
For a filter on key with defined bounds, the current record is the first record whose key is greater than <LowerBound> (if it exists).
- If a filter is enabled, ***HLast***, ***HReadLast***, ***HReadLastLock***, ***HReadLastNoLock*** return:
 - the last record corresponding to the filter,
 - ***HOut()*** set to **True** if no record corresponds to the filter.
For a filter on key with defined bounds, the current record is the first record whose key is greater than <UpperBound> (if it exists).
- If a filter is enabled, ***HNext***, ***HReadNext***, ***HReadNextLock***, ***HReadNextNoLock*** return:
 - the next record corresponding to the filter,
 - ***HOut*** returns **True** if no other record corresponds to the filter.
For a filter on key with defined bounds, the current record is the first record whose key is greater than <UpperBound> (if it exists).
- If a filter is enabled, ***HPrevious*** and ***HReadPrevious*** return:
 - the previous record corresponding to the filter,

- ***HOut*** returns **True** if no other record corresponds to the filter.

For a filter on key with defined bounds, the current record is the first record whose key is less than <UpperBound> (if it exists).

Addition, deletion and modification of a record

You have the ability to add, modify or delete a record in a file:

- ***HAdd*** adds a record,
- ***HWrite*** writes a record,
- ***HModify*** modifies a record,
- ***HCross*** and ***HDelete*** delete a record.

HAdd, ***HModify***, ***HCross*** and ***HDelete*** use the record loaded in memory.

Before deleting or modifying a record, the record loaded in memory must be initialized by ***HRead***, ***HReadFirst***, ***HReadLast***, ***HReadNext***, ***HReadPrevious*** or ***HReadSeek***, ***HReadSeekFirst***, ***HReadSeekLast***, otherwise an error will be generated (see appendix).

Important: In the browsing tables, to delete a record, you must use ***TableDelete*** (instead of ***HDelete***). To write a record, you must use ***TableSave*** (instead of ***HAdd***, ***HModify***, ...).

Note: ***HCopyRecord*** is used to copy the current record of a data file into the current record of another data file with the same structure. See “Managing aliases”, page 336 for more details.

Assigning the file variables and the window or page controls

A file’s data can be presented in a window or in a page:

- in a form: one record per window or per page
- in a table: one record per table row

Presentation in form

If the controls found in a window or in a page are linked to items, the values of the controls can be assigned to the file variables (or conversely) via a single function:

- ***ScreenToFile/PageToFile*** assigns the value of the controls on the screen to the corresponding file variables
- ***FileToScreen/FileToPage*** assigns the value of the file variables to the corresponding controls on the screen.

The file variables can also be assigned individually. For example:

```
NAME=CUSTOMER.NAME
CUSTOMER.CITY=CITY
```

Important

- Assigning the file variables does not actually modify the file: the modification is only performed when the record is written (by **HAdd**, **HModify**, ...).
- If several controls are linked to the same data file item, **FileToScreen/ScreenToFile** and **ScreenToFile/PageToFile** may behave randomly.

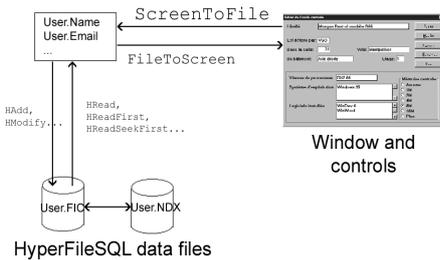
Presentation in table

ScreenToFile/PageToFile and **FileToScreen/FileToPage** must not be used in the browsing tables.

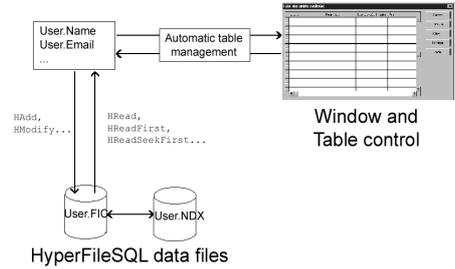
In tables, assigning the table columns with the values of the file items (or conversely) is automatically managed.

The management of files in the tables, called "browsing tables", is specific. It is presented in the online help.

1.1.7 Schemas: the form mode and the table mode



Managing a file in form to form mode



Managing a file in table mode

1.1.8 Moving and positioning in a data file

Three WLanguage functions can be used to position or move in a data file without actually browsing the data file.

HSetPosition Used to position in a data file according to a key or to find out the position of the current record relative to a key in relation to the total number of records

HForward Used to move forward several records at a time from the current position according to a given key

HBackward Used to move backward several records at a time from the current position according to a given key

Example

```
// Position at 50% of CUSTOMER on
// the CUSTNAME key
// from the last one
MaxRec is int
MaxRec = HNbRec()
HLast(CUSTOMER, CUSTNAME)
HBackward(CUSTOMER, CUSTNAME,
MaxRec/2)
IF HOut() THEN
```

```
Error("Outside position")
ELSE
  Info("Position: rec="+...
  hRecNum()+" Customer = "+...
  CUSTOMER.CUSTNAME)
END
```

1.2 The data files in FoxPro xBase format

1.2.1 Overview

A module for Native xBase/FoxPro Access is available with WinDev and WebDev. This Native Access allows you to handle the xBase files from a WLanguage program without using an external driver.

Required configuration

The following files are supplied with WinDev/WebDev: WD180DB.DLL. This file is required by the Native Access to the xBase/FoxPro files in order to work with a HFSQL Classic analysis.

1.2.2 Using the native xBase/FoxPro access

To use the native xBase/Fox Pro access in your WinDev or WebDev applications:

1. Import (if necessary) the description of xBase/FoxPro files into the WinDev or WebDev analysis.
2. Program with the "HFSQL" functions of WLanguage (see "Programming using the HFSQL functions").

The management of the xBase/FoxPro files is similar to the management of the HFSQL files except for the specified limitations (see the online help).

1.2.3 Importing the structure of the files

The import steps

To perform this import in the data model editor:

1. On the "Analysis" pane, expand "Import" and select "Import the descriptions of files/tables". The wizard starts.
 1. Select the type of the database: FoxPro/xBase. A connection to the xBase/FoxPro database is automatically established and associated with the files imported into the analysis. This connection will be used to handle the xBase/FoxPro data file by programming.
 2. Specify whether the data must be kept in current format (option required to use the Native xBase/FoxPro Access).
 3. Select the access mode to the files (Native xBase Access for WinDev) and specify the directory of data files.
 4. Select the tables that will be imported into the WinDev analysis. These tables can be handled by programming with the Hyper File functions of WinDev. These tables will be displayed in blue in the data model editor.

Note: the FoxPro databases (.dbc extensions) are not supported. Each file (.dbf extension) must be individually imported into the analysis.

5. Validate. The tables are imported into the analysis. The xBase sub-type of the imported file can be found in the file description (dBase3+, dBase4, FoxPro/FoxBase, Clipper5, Clipper87).

Notes:

- A Numeric item in FoxPro can be imported as a Currency item into WinDev or WebDev (if the size of the items exceeds the size of a real for example).
- The Logical items in xBase/FoxPro are imported as Text items. Indeed, for a boolean in xBase, you have the ability to specify True and False with the strings 'T' and 'F'.

Taking into account the changes made to the xBase files

To take into account the evolutions of tables imported into the data model editor, on the "Analysis" pane, in the "Analysis" group, expand "Synchronization" and select "Update the analysis from the external databases". A wizard starts, allowing you to:

- analyze the differences for the imported tables.
- analyze the differences for the imported tables.

1.2.4 Important programming points

The management of xBase files is similar to the management of HFSQL files except for the points mentioned in this paragraph.

New features regarding the management of xBase files:

- Management of an automatic identifier
- Ability to use **HIndex**.
- Managing expressions in **HFilter**
- Ability to use **HChangeDir**
- Using the native xBase access with non-latin character sets.

Managing FoxPro files:

- Support of VFP format
- All the index formats are supported (including the indexes in VFP format corresponding to the .CDX extension). The indexes are read and modified.
- Management of an automatic identifier
- Managing expressions in **HFilter**
- Ability to use **HChangeDir**
- Limitation: no creation or reindexing of a FoxPro file.

Dynamic description of an xBase file

An xBase file can be described by programming by the following functions:

HDBCcreation	Ends the dynamic description of the file structure. The file that was just described is created on disk in the path specified in HDBDescribeFile . The data, memo and index files are created on the disk
HDBDescribeFile	Dynamically describes a file in dBase 3 format (most common format). This function is used to specify the name, the abbreviation and the access path of the xBase file.
HDBDescribeIndex	Dynamically describes the different index files that will be created. An index file must be described for each search key. The created index file will be in dBase 3 format.
HDBDescribeField	Dynamically describes each item of the structure of an xBase file described by HDBDescribeFile.

Example:

```
// Describe an xBase file
// that contains the following items
// LASTNAME, string of 20 characters
// FIRSTNAME, string of 20 characters
// AGE, integer on 3 digits
// BALANCE, real on 10 digits and
// 2 decimals
// MARRIED, boolean
// DOB, date
// INFO, text memo
HDBDescribeFile("CUSTOMERDB",...
"CD","C:\FIC\CUSTOMER.DBF")
HDBDescribeField("LASTN,C,20")
HDBDescribeField("FIRSTN,C,20")
HDBDescribeField("AGE,N,3,0")
HDBDescribeField("BALANCE,N,10,2")
HDBDescribeField("MARRIED,L")
HDBDescribeField("DOB,D")
HDBDescribeField("INFO,M")
HDBCcreation()
```

Opening files that were described dynamically

An xBase file is not automatically opened: this file must be opened by **HDBOpen** before its first use unless it was created by **HDBCcreation** (that creates and opens the file).

Opening an index

HDBOpen opens the data file but not the index file(s). Therefore, you must open all the index files required for using the data file with **HDBIndex**.

If the index files are not opened, they will not be updated after a deletion, a modification or an addition.

Links between the xBase or FoxPro files

The links are not automatically managed. They must be managed by programming. Therefore, to access a record in the linked file, you must find the corresponding record according to the link key (**HReadSeek**).

Example: A window displays the customer orders, the "ORDERS" file contains the product number, the "PRODUCT" file contains the product caption and the product number.

For example, the initialization process of the window is:

```
// Read the orders
HReadFirst(ORDERS, Date)
IF HOut() = False THEN
// Find the product
// corresponding
HReadSeek(PRODUCT, ProdNum, ...
ORDERS.ProdNum)
FileToScreen()
END
```

Using the native xBase/FoxPro access with non-Latin character sets

To use the native xBase access with non-Latin character sets, a standard OEM/ANSI conversion must be performed by the native access. To do so, the following string must be specified in the extended information:

OEMTOANSI=WINDOWS;

1.3 The data files found on a mobile device (Pocket PC, iOS, Android)

1.3.1 Handling a HFSQL database

Overview

This HFSQL database format is compatible between WinDev, WinDev Mobile and WebDev. It is a freely distributable Relational DBMS.

However, the available size on a Pocket PC being restricted and the operating system of the Pocket PC being limited, the following features are not supported:

- the transactions.
- the log process.
- the HFSQL logged replication.
- the management of locks for the files and records.
- the management of files in Hyper File 5.5 format.

Handling a HFSQL database from the mobile device

A HFSQL database corresponds to a set of ".FIC", ".NDX" and ".MMO" files.

Each data file can be handled by a Mobile application. These operations are performed via the HFSQL functions.

Note: The sleep mode may have unexpected effects on your data files. We recommend that you close the data files (**HClose**) when the computer might go into sleep mode.

1.4 Data files specific to Windows Mobile

1.4.1 Handling a CEDB database (Pocket PC only)

Overview

The CEDB format is a database format for Pocket PC.

A CEDB database corresponds to a ".CDB" file. A CEDB database can contain several data files (also called "tables").

Two types of CEDB databases are available:

- the standard CEDB databases, that correspond to the databases found by default on the Pocket PC. These databases contain the "Tasks", "Contacts" and "Calendar" data files, ...
- the other CEDB databases (called customizable databases), that correspond to the Access databases (".MDB" file) previously exported from a PC.

When an Access database (".MDB" file) is copied onto a Pocket PC, this database is automatically changed into a CEDB database (".CDB" file).

Handling a Pocket PC database (CEDB)

1. Handling a Pocket PC database (CEDB) from the Pocket PC

A Pocket PC database (called CEDB) corresponds to a ".CDB" file. A Pocket PC database can contain several data files (also called "tables").

This database can be handled by a WinDev Mobile

application. These operations are performed by the *cdbXXX* functions.

2. Handling a Pocket PC database (CEDB) from the PC

If you own WinDev, you also have the ability to create a WinDev application used to directly handle the Pocket PC database. These operations are also performed by the *cdbXXX functions*.

Note: To handle a Pocket PC database from a standard WinDev application, the PC must be connected to the Pocket PC (**ceConnect**).

3. Synchronizing a Pocket PC database (CEDB) with an Access database

An Access database (".MDB" file) is found on the PC. This database is exported to the Pocket PC: ActiveSync automatically transforms it into a Pocket PC database (".CDB" file).

This Pocket PC database can be handled by a WinDev Mobile application.

If you own WinDev, you also have the ability to create a WinDev application used to handle the Pocket PC database.

These operations are performed by the *cdbXXX* functions.

The synchronization between the Pocket PC database and the Access database is performed by ActiveSync.

Notes:

- To handle a Pocket PC database from a standard WinDev application, a connection must be established between the PC and the Pocket PC (**ceConnect**).
- The WinDev application can also handle the Access database via the Native Access.

Handling a standard Pocket PC database

A standard database (containing the data files for managing tasks, contacts, ...) is found on the Pocket PC. This database can be handled by a WinDev application for Pocket PC.

If you own WinDev, you also have the ability to create a WinDev application to directly handle this standard Pocket PC database.

These operations are performed by the **cdbXXX** functions.

The synchronization between the Pocket PC database and the data viewed via Outlook is performed by ActiveSync.

Note: To handle a Pocket PC database from a standard WinDev application, the PC must be connected to the Pocket PC (**ceConnect**).

The format of the ".CDB" files is not compatible with Windows CE 3.0 and Windows CE 4.X. Therefore, the same ".CDB" files cannot be used on Pocket PCs not running the same operating system.

Structure of standard databases

The structure of standard databases is a preset structure of WLanguage (no declaration is required).

This structure is used to:

- create a record,
- modify a record,
- retrieve the content of a record.

To reset all the variables of a structure to zero, use **cdbReset**.

Note: The variables found in the structure of standard databases differ according to the database used: **Contacts**, **Calendar** and **Tasks**. See the online help for more details.

1.4.2 Functions for handling a CEDB database

The following functions are used to manage a CEDB database:

cdbAdd	Adds the record found in memory into a data file
cdbCancelSearch	Cancel the current search criterion
cdbClose	Closes a database (".CDB" file) that was opened by cdbOpen
cdbCol	Returns a characteristic of a column found in the current record (value, type, identifier or name)
cdbDateTimeToInteger	Transforms a DateTime variable into a value compatible with a Date and Time column (unsigned 8-byte integer)
cdbDelete	Deletes the current record or the specified record from the data file
cdbFound	Checks whether the current record corresponds to the current search
cdbIntegerToDateTime	Transforms the value of a Date and Time column (unsigned 8-byte integer) into a Date-Time variable
cdbListFile	Returns the list of data files found in a CEDB database
cdbModify	Modifies the specified record or the record found in memory in the data file
cdbNbCol	Returns the number of columns found in the record in memory
cdbNbRec	Returns the number of records found in a data file
cdbOpen	Opens a database (".CDB" file) on the Pocket PC connected to the current computer
cdbOut	Used to find out whether the record on which we want to be positioned is outside the file
cdbRead	Reads a record in a file according to a given record number
cdbReadFirst	Positions on the first file record and reads this record
cdbReadLast	Positions on the last file record and reads this record
cdbReadNext	Positions on the next file record and reads this record
cdbReadPrevious	Positions on the previous file record and reads this record
cdbReadSeek	Positions on the first file record whose value for a specific column is equal to a sought value
cdbRecNum	Returns the number of the current record in the data file
cdbReset	Resets all the variables found in one of the structures of the standard databases
cdbWriteCol	Modifies the value of a column for a record in memory

2. ADVANCED FEATURES

This chapter presents the features for an advanced management of a database:

- managing aliases,
- managing NULL in HFSQL,
- protection and data encryption,
- checking the referential integrity,
- managing the "memo" files,
- reassigning files,
- full-text search and index,
- managing transactions,
- logging files,
- automatic modification of the files,
- exchanging and sharing data between several programs,
- temporary file,
- retrieving the structure of a file,
- using an ODBC driver on HFSQL Classic.

2.1 Managing aliases

The aliases are used to handle:

- several physical files with the same description in the analysis,
- several logical files described in the analysis that use the same physical file.

2.1.1 Several physical files with identical logical description

For example, you have the ability to handle at the same time:

- the Customer2000.fic that contains the backup of customers for the year 2000.
- the Customer.fic file that contains the references of customers for the current year.

These two files have the same logical description in the analysis: the CUSTOMER file.

By default, the description of CUSTOMER file found in the analysis is linked to the Customer.fic file.

To use the Customer2000.Fic file, you must "copy" the description of CUSTOMER file found in the analysis: all you have to do is create an alias with **HAlias**. The code line is as follows:

```
HAlias(Customer, Customer2000)
```

The two files will be handled as usual by the standard HFSQL functions.

2.1.2 Several logical files linked to a single physical file

You can perform several parallel searches on the same physical file, or a search and a filter in parallel on the file.

Example: The Customer.Fic file is associated with the logical file named Customer in the analysis. To filter the records of Customer.fic file and to browse at the same time all the records of this file, you must:

- "copy" the description of CUSTOMER file into the analysis: all you have to do is create an alias with **HAlias**.
- associate the alias description with the physical file named Customer.fic (**HChangeName**).

Note: This method can be used to manage several HFSQL contexts for the same physical file.

2.1.3 Functions for managing aliases

Several functions are available for handling the aliases:

HAlias	Creates a logical alias of a file or cancels all the existing aliases
HCancelAlias	Cancels an alias that was declared by HAlias

Other functions can intervene in the definition and use of aliases:

HChangeName	Modifies the physical name of a data file
HChangeDir	Modifies the access path to a data file

2.1.4 Create an alias on what?

You have the ability to create and handle aliases on the following elements:

- Data file described in the analysis
- Data file described dynamically
- Query created in the query editor
- HFSQL View (HFSQL Classic format)
- Alias

2.1.5 Characteristics of an alias

• **Physical file associated with an alias:**

By default, the name specified for the alias is assigned to the physical file corresponding to the alias. To associate the alias with a physical file of different name, use *HChangeName* and *HChangeDir*.

• **Password of an alias:**

By default, the password of an alias file is identical to the password of the initial file. This password can be modified by *HPass*. The name of the file used in this function corresponds to the name of the alias.

• **Reindexing:**

The alias files can be reindexed.

• **Automatic modification of data files:**

This operation can be applied to the alias files.

2.1.6 Handling the alias file and its items

• **Code editor:** To avoid the "Unknown identifier" error when using an alias file and its items, a data source must be defined to declare the name of the alias before using *HAlias*.

For example:

```
Orders2000 is data source
HALias (Orders,Order2000)
```

Caution: The *Extern* keyword can be used but it slows down the execution significantly.

• **Window editor or page editor:** the controls and the tables cannot be directly associated with the items found in alias files. The controls must be assigned one by one by programming.

Note: To redefine the links of controls by programming, you have the ability to use either *ControlAlias*, or *..FileLink*.

2.2 Managing NULL in HFSQL

WinDev and WebDev enable you to manage the Null value in the items found in the HFSQL data files and for all the other types of accesses (Native Access, OLE DB, ...).

When calculations are performed on the file records, the records containing a Null item will be ignored. For example, if a query calculates the average grade of students for the quarter, only the grades of the attending students will be taken into account. If a student is not attending, his grade will correspond to the Null value.

2.2.1 How to manage the Null value in one of your items?

To manage the null value in your files, you can use:

- In the data model editor:
 - the "NULL supported" option found in the "Info" tab of the file description. This option is used to specify whether the management of NULL is supported by this file. In this case, the NULL value can be managed for the different file

items.

- the "Default to NULL" option found in the "General" tab of an item description. This option enables you to define the null value as default value for the item.
- In programming, two properties:

Null	Used to: <ul style="list-style-type: none"> • define the NULL value as the default value for a file item during its dynamic description. • associate (or not) the NULL value with a file item.
------	--

NullSupported	Used to: <ul style="list-style-type: none"> • define the management mode of the NULL value for a file during its dynamic description. • find out the management mode of the NULL value for a file
---------------	---

Limitations: The NULL value cannot be used:

- on the array items.
- on the automatic identifiers.
- on the composite keys.

Caution: Writing records by an application that uses a WinDev version earlier than version 75205 may cause the *..Null* property to be inconsistent (will return True or False).

Writing records by an application that uses WinDev version 75205 or later sets the *..Null* property to False, even if the item was set to Null by an application using WinDev.

2.2.2 How can I use the NULL value in my applications?

How do I save a NULL value in an item?

To enter a NULL value in an item:

1. Display the description window of the file ("Description of data file" from the popup menu).
2. In the "Info" tab, check the "NULL supported" box.
3. Validate the window.
4. Generate the analysis.
5. **If no value was entered by the user in a control associated with an item:** to assign the NULL value to this item for the current record, the *..Null* property must be used after **ScreenToFile**.

Example: The "NULL if empty" option is checked for the edit control named EDT_Mark. In this case, you can run the following test before saving the data:

```
ScreenToFile()
IF EDT_Note = NULL THEN
    Lesson.Grade..Null = True
END
```

```
// Add or modify
// the record
HAdd(Lesson)
```

Caution: The "NULL if empty" option of edit controls has no relation with the management of the NULL value in the HFSQL data files.

See the online help (keyword: "Null if empty") for more details.

Note: **If the default value of the item is NULL** (box checked in the editor), **HReset** resets the *..Null* property to True.

Query: Selection condition

In the query editor, you have the ability to take into account or to ignore the records that have a null item. To do so, create a selection condition and choose "Is null" or "Is not null".

In a selection query that performs a calculation, all the records that have a null value (for the calculation item) will be ignored.

Example

The following example is a query used to calculate the average grade of students for the 2012 French class. If one of the grades corresponds to the NULL value, this grade is ignored.

```
SELECT Marks.Course AS Course,
Grades.Date AS Date,
LEFT(Grades.Date,4) AS Year,
AVG(Marks.Mark) AS the_average_Mark
FROM Marks
WHERE Grades.Course = 'French'
AND Marks.Mark IS NOT NULL
AND LEFT(Grades.Date,4) = '2012'
GROUP BY Grades.Course, Grades.Date,
LEFT(Grades.Date,4)
```

2.3 Protecting and encrypting data files

2.3.1 The protection methods

Several methods can be used to preserve the confidentiality and security of data when storing data files. These methods can be applied to the different types of files: data files (.Fic), index file (.NDX), memo file (.MMO). You have the ability to:

- not encrypt the files,
- encrypt the file on 128 bits: only the users who know the password will be able to access them in read and write modes. The password is managed in the program when the file is accessed.

The level of protection for the data file is chosen during the file description, in the data model editor. It cannot be modified by programming.

If a strong security is enabled, the password of the data file will be requested for each automatic modification of data files.

To modify the protection level, you must modify the description of the data file.

To summarize, a data file can be:

- not protected,
- password protected with data encryption,
- password protected with index encryption,
- password protected with memo encryption.

2.3.2 Managing encrypted files

The password associated with a data file must be transmitted by programming just before creating or opening the data file. The developer can:

- ask the password to the user,
- include the password in the program or read it in a configuration file.

We advise you to implement a password containing at least 4 characters. The size of the password is unlimited.

The password can be passed to the data files according to two methods:

- by passing the password in parameter to **HOpen**, **HCreate**, **HCreateIfNotFound**.
- by using **HPass** before opening or creating the file.

When WinDev or WebDev interprets **HCreation** or **HCreationIfNotFound**, the password must be known by **HPass** if it is not specified in the parameters of functions.

```
HPass (CUSTOMER, "Secret code")
```

If the password is incorrect, **HErrorPassword** returns *True*.

Notes

- The data files found in an analysis can have identical or different passwords.
- Each data file is encrypted according to its own password.
- **Caution:** You cannot retrieve a "forgotten" password. If the password is lost, the data file cannot be accessed anymore.

2.4 Managing an identifier

2.4.1 Automatic management

In most cases, each record found in a data file must contain an item having a unique value in the data file. This item, called *identifier*, is used to identify the record in the data file. This is a unique key item.

The record number cannot be used to identify a record. Indeed, this one is not fixed; for example, it can be modified after a reindex operation with compression.

WinDev or WebDev allows you to automatically manage an identifier: the identifier is chosen when

describing the data file, in the data model editor.

In this case, WinDev or WebDev takes care of everything: it automatically creates the item and assigns this item when the record is added into the data file.

Important: Never modify the value of the identifier otherwise you could compromise the uniqueness of records.

You have the ability to manage an identifier by programming, without using the automatic identifier proposed by WinDev or WebDev.

2.4.2 Manual management

The following method is used to manage an identifier manually:

1. When describing the data file, if it does not include a unique key, create an integer item and define it as the unique key.
2. When adding a record, the value of this item is managed by a process: all you have to do is increase the value of the item for each created record.

Example of manual management of an identifier: this simple management can be used in a single-computer application for example. For more advanced examples, see the chapter about the replica-

tion.

```
HReadLast ( INVOICE , INVNUM )
ScreenToFile
INVOICE . INVNUM = INVOICE . INVNUM + 1
HAdd ( INVOICE )
```

2.4.3 Checking the uniqueness of a key

Whether the identifier is managed manually or automatically by WinDev or WebDev, you must check the uniqueness of the key when modifying a record in the data file.

The check for key uniqueness was presented in details in the previous chapter of this section.

2.5 Automatic check of referential integrity

The referential integrity consists in checking that:

- if a record is deleted from the parent (or owner) file, the corresponding records are also deleted from the child (or member) files,
- if a record is added into a child file, a corresponding record exists in the parent file,
- if a record is modified in a parent file, the unique key is not modified,
- ...

The check of referential integrity depends on the nature of the link between the files.

2.5.1 Benefit of the check of referential integrity

For example, when deleting a record from a file, you must check that the records found in the linked file that correspond to the deleted record have been deleted.

Let's take an example for order management containing the following files:

- "ORDERS" containing the references of the order. The identifier is the order number.
- "PRODUCT" containing the references of products. The identifier corresponds to the product number.
- "CUSTOMER" containing the customer details. The identifier is the customer number.
- "ORDERS" is linked to "CUSTOMER" by the custo-

mer number.

- "ORDERS" is linked to "PRODUCT" by the product number.

When a customer is deleted from "CUSTOMER", if the orders corresponding to the customer are not deleted, what may happen?

The customer 1000 is deleted (but not his orders) and he is the last one in the list. When adding a new customer with number 1000, this customer will have orders that do not belong to him but to the deleted customer!

Checking the referential integrity is used to automatically check whether integrity is preserved and to avoid inconsistencies in the database.

To use the feature for automatic check of referential integrity:

- The links must be described in the data model editor or by **HDescribeLink**.
- The type of links must allow the automatic management of referential integrity. The different types of links are described in the following paragraph.
- The management of referential integrity must have been enabled in the project by **HSetIntegrity (True)** in the initialization code of the project.

2.5.2 Definitions

The definitions of the different types of links available are presented in the "Concept" manual.

2.5.3 The different types of links

The different types of links are presented in "Getting Started Guide".

Reminder: Several types of links can be found between the files:

- parallel,
- optional,
- complement,
- shared,
- complex.

2.5.4 Programming the automatic check of referential integrity

The automatic check of referential integrity can be used in WLanguage only. It cannot be used:

- for the files in xBase format,
- in external language.

Enabling the mechanism

By default, the mechanism for automatic check of referential integrity is disabled. It can be enabled:

- for the project description,
- by programming.

Activation at project level

In the project description, you have the ability to define whether the automatic integrity check and/or duplicate check must be enabled.

This choice does not require any additional code line in your project.

Activation by programming

The mechanism for automatic check of referential integrity can be enabled by *HSetIntegrity (True)* or disabled by *HSetIntegrity (False)*.

Caution:

The activation of the mechanism for automatic check of referential integrity implicitly enables the mechanism for duplicate management (*HSetDuplicates*).

You cannot disable the mechanism for duplicate check if the mechanism for integrity check is enabled (*HSetIntegrity (True)*).

Notes:

- The mechanism for integrity check corresponds to searches performed in the linked files during the calls to functions that write into the data files (*HAdd, HModify, HDelete, HCross, TableSave, TableDelete*). This mechanism does not modify the context (positioning) of the base file and linked files.

Summary table of the integrity checks performed by WinDev/WebDev

The table below summarizes, for the functions requiring an integrity check (*HCross, HDelete, HAdd, HWrite and HModify*), the integrity check performed by WinDev/WebDev depending on the link type.

	Comple- ment link (0,1) - (1,1)	Shared link (x,n) - (1,1)	Shared link (1,n) - (0,1)	Shared link (0,n) - (0,1)	Optional link (0,1) - (0,1)	Parallel link (1,1) - (1,1)
HCross	1	1	1	-	-	Dev.
HDelete						
HAdd	2	2	Dev	-	-	Dev
HWrite						
HModify	3 and 4	3 and 4	5	-	-	Dev

Legend :

- 1: when deleting an owner, check there is no linked members
- 2: when adding a member, check the existence of a linked owner
- 3: when modifying a member, check the existence of a linked owner (in the case where the value of the linked key changes)
- 4: when modifying an owner, check there are no members linked to the old owner (in the case where the linked key changes value)
- 5: when modifying a member, check there are no owners linked to the old member (in the case where the linked key changes value).

- The referential integrity is not checked when overwriting a file with **HCreate**. We advise you to use **HCreationIfNotFound** in order not to overwrite any data files.

Checking the integrity

When the mechanism for integrity check is enabled (**HSetIntegrity (True)**), **HErrorIntegrity** must necessarily be checked after each function that writes into the file: **HAdd**, **HModify**, **HWrite**, **HDelete**, **HCross**, **TableSave**, **TableDelete**.

HErrorIntegrity returns *True* if an integrity error occurred. **HErrorInfo** returns the error details:

If **HErrorIntegrity** is not called, the error 15 (Integrity error) will be automatically generated during the next call to a function for file management. This error cannot be ignored: all the files are closed and the program execution is canceled.

Example in single-computer mode (automatic management of locks and integrity management)

```
HSetIntegrity(True)
// Delete the customer whose number
is Num
HReadSeekFirst (CUSTOMER, ...
    CustomerID, Num)
IF HFound() THEN
    IF HDelete(CUSTOMER) THEN
        Info("Customer deleted")
    ELSE
        IF HErrorIntegrity() THEN
            Error("Cannot delete this custo-
            mer", there are still orders")
        END
    ELSE
        Error("Customer not found")
    END
END
```

Example in network

```
HSetIntegrity(True)
HLockFile(CUSTOMER)
IF HErrorLock() THEN
    Info("Unable to lock the file")
ELSE
    HReadSeekFirst (CUSTOMER, ...
        CustomerID,
        Num)
        IF HFound() THEN
            IF HDelete(CUSTOMER) THEN
                Info("Deletion OK")
            ELSE
                Error(HErrorInfo())
            END
        END
    END
END
```

2.6 Managing the "memo" files

2.6.1 Text memo and binary memo

The "memo" files can be managed in WLanguage.

The "memo" files complement the data files. They are used to associate things such as long texts, images and sounds with the records found in the data file.

A "memo" file is described and created at the same time as the data file.

To create a "memo" file:

1. Describe the data file in the data model editor.
2. Create a "text memo" item for the long texts or a "sound, image, binary" item to store images or sounds.
3. Generate the analysis.

In WLanguage, the data file and the "memo" file evolve in parallel:

- **HCreation** and **HCreationIfNotFound** create the empty data file, the index file and the "memo" file (<FileName>.MMO),
- **HAdd** and **HModify** write into the data file and into the "memo" file,
- the read functions read the record in the data file and in the "memo" file.

Caution: **HCross** and **HDelete** have different actions on the "memo" files and on the data files:

- **HCross** delete the record from the data file but not from the memo file. Indeed, **HFree** does not free the crossed records in the "memo" files. We do not advise you to use **HCross** with the "memo" files.
- **HDelete** deletes the records from the data file and from the "memo" file. To delete a record from a "memo" file, you must use **HDelete**.

A function is used to enable the management of "memo" files: **HSetMemo**. This function enables or disables the management of memo items. You have the ability to manage all the memos of a data file or a specific memo item of a data file. The management of memos is enabled by default.

If a process on a file does not use the "memo" file, it may be interesting to disable the management of the "memo" file to speed up the process.

For example, when modifying the "TotalSales" item for all the records found in "Customer" file, all you have to do is disable the management of memos before the process, then re-enable it thereafter.

Note: The memo files can be encrypted and/or compressed.

2.6.2 Image, sound, OLE and other binary memos

Four types of items are available: image, sound, OLE and other binary memo. These types of memos are used to store binary files (images, sounds, OLE files, ...) in the item.

The items can be handled in WLanguage in order to assign them or to retrieve their content.

The following functions are used to manage the binary memos:

- **HLinkMemo** to assign the item.
- **HExtractMemo** to retrieve the content of the item and to save it in a file.
- **HInfoMemo** to get information about the item.

Note: The binary memo items can be handled by the following functions:

- **Sound** if the binary memo item is in WAV format (WinDev only).
- **iPrintImage** if the binary memo item is in IMG format.
- **BitmapInfo** if the binary memo item is in IMG format.

Initialization

A binary memo can contain any type of file. A binary memo can be initialized:

- from another binary memo item (simple assignment)

```
ANIMAL.PHOTO=DOG.PHOTO
```

- from a file via **HLinkMemo**

```
HLinkMemo(ANIMAL, ...
    PHOTO, "DOG1.TIF", hMemoImg)
```

Notes:

- When an OLE file is modified via an OLE control, to assign the OLE file in an OLE binary memo item, you must:
 - save the OLE object with **OLESave**
 - assign the item with **HLinkMemo**.
- When an image control is modified by the drawing functions of WLanguage, to assign the image in an image binary memo item, you must:
 - save the image with **dSaveImage**,
 - assign the item with **HLinkMemo**.
- **ScreenToFile** or **PageToFile** does not automatically assign the content of an image control in the image binary memo item that is associated with it. The item must be assigned by **HLinkMemo**.

- **HLinkMemo** is used to give an information:
 - about the file type,
 - about the memo itself.

WB The sound, binary memo and OLE items exist for compatibility with WinDev.

Retrieval

A binary memo is retrieved by **HExtractMemo**. This function saves the content of the binary memo in a file.

Note: **FileToScreen** or **FileToPage** automatically assigns the content of an image control with the content of the binary memo item that is associated with it, providing that the item content has a recognized image format (BMP, PCX, JPEG, TIFF, ...).

Information about the memo

HInfoMemo is used to retrieve information about the binary memo item:

- type of the memo,
- name, size, date and time of source file,
- free information given when the memo is assigned by **HLinkMemo**.

2.7 Reassigning data files

2.7.1 Benefit

The same file description can correspond to several physical files. The data files have:

- different physical names,
- the same name but they are stored in different directories.

You must be able to modify the name or the directory of the files.

The analysis files are described in a hard-coded directory. In final use, the setup drive of the files can change according to the computer on which the WinDev application is installed or according to the server on which the WebDev site is deployed. You must be able to modify the storage directory of the files.

The name and/or the directory is modified by programming. It must be done before creating or opening the file.

2.7.2 Modifying the storage directory

When the data file is described in the data model editor, a directory for storing the data files and the index files was selected:

- a hard-coded directory,
- the application directory.

The data files can be stored in directories that differ from the ones described in the analysis. All you have to do is modify the name of the directory by programming.

HSubstDir and **HChangeDir** allow you to specify a directory that differs from the one described in the data model editor.

The directory defined in the analysis is the *requested directory*.

The directory of the file on disk is the physical direc-

tory.

For example:

```
// Files described on the C drive
and installed on the drive S
HSubstDir("C:\Dir1",...
         "S:\DataDirectory")
HCreationIfNotFound(NameFile1)
```

2.7.3 Modifying the name of a data file

By default, a data file is created with the name defined in the data model editor.

The data file can be created with another name. All you have to do is modify the name of the data file by programming.

HChangeName is used to specify a name that differs from the one defined in the description of the data file.

The file name that was defined in the analysis is the *logical name*.

The name of the file (or files) on disk is the *name on disk* (this name can be a long name if the operating system supports the long names).

Note: The long name of a file cannot exceed 260 characters (including the access path to the file).

2.7.4 Keeping trace of reassignments

What is the .REP used for?

<MyProject>.REP is a file that contains the list of files used by the application or by the site (identifier, logical name and full path of the physical file).

The GUID of the analysis is the unique identifier of the analysis linked to the project, containing the description of the files. This identifier can be found in the data model editor, in the analysis description ("Options" tab).

The GUID of the file corresponds to the identifier of the logical file. This identifier can be found in the data model editor, in the file description ("Notes" tab).

Analysis GUID : D7A79ED0776C11D486C800902757C224		
Files GUID	Logical Filenames	Physical Filenames
D7A79F06776C11D486C800902757C224	Client	c:\Gestcom\client2000.fic
D7A79F06776C11D486C800902757C224	Client	c:\Gestcom\client2001.fic
D7A79ED4776C11D486C800902757C224	Order	c:\Gestcom\Commands.fic

This file is automatically created in the directory of the application or site and it is filled by the HFSQL engine.

The ".REP" file allows you to easily localize the data files used by the WinDev application or by the WebDev site.

The WinDev application or the WebDev site automatically updates the ".REP" file but it seldom uses the ".REP" file.

This file is used by all the tools that handle the application files or the site files, and mainly by the automatic file update, ...

Example: Updating an application or a site with modification of the analysis.

When updating a WinDev application or a WebDev site, the automatic modification of the data files is automatically run if the database structure is modified.

This procedure uses the ".REP" file to find the physical files used by the application or by the site in order to modify them.

See "Automatic modification of data files", page 356 for more details.

Note: The .REP must not be modified "manually" except in exceptional cases, for example on the advice of a member of PC SOFT technical support.

How is the .REP file managed in WinDev Mobile?

WinDev Mobile enables you to update a list of data files used by the application (equivalent to the .REP file in WinDev).

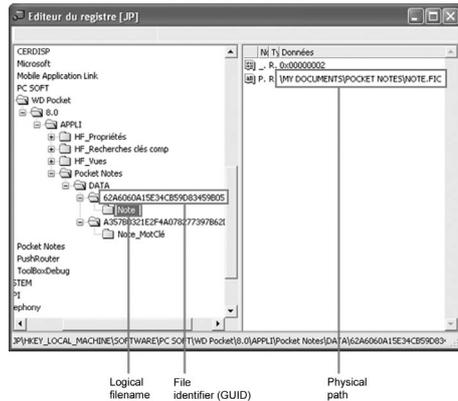
This list is saved in the Pocket PC registry, in the following key:
 "\HKEY_LOCAL_MACHINE\SOFTWARE\PC SOFT\WD Mobile\19.0\<ApplicationName>\DATA".

The following details are stored for each data file used by the current application:

- identifier (file GUID). This identifier can be found in the data model editor, in the file description ("Notes" tab).
- logical name.
- full path of the physical file.

This information is automatically filled by the HFSQL Mobile engine.

For example:



Functions for managing the .REP file

1. Management of .REP

HSetREP Enables or disables the management of .REP file. When the management of the .REP file is enabled, you can specify the name and path of the .REP file. A HFREP.INI file will be created in the Windows directory. The management of ".REP" file is enabled by default.

2. Creating and modifying the .REP file

HCreation Creates a physical data file (and fills the .REP file if the management of .REP is enabled)

HCreationIfNotFound Creates a physical data file if it does not exist (and fills the .REP file if the management of .REP is enabled)

HOpen Opens a physical data file and fills the .REP file if necessary (opening a file that does not belong to the analysis for example)

Note: The information given by *HChangeDir*, *HSubstDir* and *HChangeName* will be taken into account when creating and updating the ".REP" file.

3. Reading the .REP file

HListREP Lists the different files found in the .REP file as well as their physical path

Note: If an application or a site uses several physical files with the same logical name (common case for a multi-company accounting software or for "archive" files), the ".REP" file will contain several lines for the same logical file: each line will reference a physical file.

4. Writing into the file

To write into the .REP file, all you have to do is use the functions for managing the external files:

fOpen Opens the file
fWriteLine Writes a new line into the file
fClose Closes the file

The structure of the added line must be as follows:

```
LOCALIZATION= File Tab Logical file Tab Path of
                GUID name physical
                file
```

The GUID of the file corresponds to the identifier of the logical file. This identifier can be found in the

data model editor, in the file description ("Notes" tab).

Structure of .REP file

The first line contains the information regarding the analysis in which the files have been described. This analysis corresponds to the analysis of the current project.

The structure of this line is as follows:

```
ANALYSISGUID= Analysis GUID
```

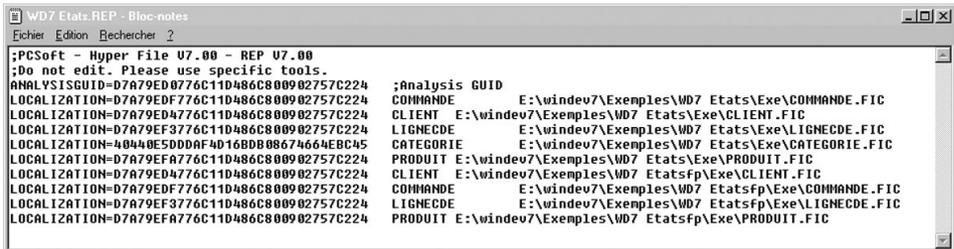
The GUID of the analysis is the unique identifier of the analysis linked to the project, containing the description of the files. This identifier can be found in the data model editor, in the analysis description ("Options" tab).

The following lines describe the different files used by the application or by the site. A single line is created for each physical file used. This line contains the following information:

```
LOCALIZATION= File Tab Logical Tab Path of
                GUID file physical
                name file
```

The GUID of the file corresponds to the identifier of the logical file. This identifier can be found in the data model editor, in the file description ("Notes" tab).

Example of .REP file:



2.8 Full-text search and index

2.8.1 Overview

HFSQL allows you to easily find character strings in the data. This search (called "full-text") is used to find a word or a set of words.

This enables you to index, without programming, the texts found in a HFSQL database. These texts can be found in "Text" items or in "Text memo" items.

An index can index one or more different items. The creation of the index can take the formatted texts into account (RTF, HTML): the tags of these formats will be ignored during the indexing. This enables you to find words stored in RTF or HTML documents.

The results of the full-text indexing are proposed according to a relevance order ("ranking").

Caution: the "Full-text" search is not a simple "contains" search: the punctuation characters are ignored.

2.8.2 How do I perform a "full-text" search?

To perform a "full-text" search, you must:

1. Create a full-text index in the data files affected by this search.
2. Create a query to perform the full-text search.
3. Analyze and display the result of the query.

2.8.3 How do I create a full-text index?

The creation of a full-text index is performed in the data model editor.

To create a full-text index:

1. In the data model editor, display the description of the items found in the relevant file ("Description of items" from the popup menu).
2. Click the icon . The window for defining the full-text index is displayed. Select the items taking part in the composition of the full-text index. Only the "Text" items and the "Text memo" items found in the file description are proposed. Validate.
3. The "Full-text index" item is displayed in the list of items.
4. Select the item and modify its characteristics if necessary:
 - in the list of items, you have the ability to modify the name of the full-text index.
 - on the right side of the window, select the parameter of the full-text index: minimum size of indexed words, management of RTF tags, mana-

gement of HTML tags, management of the case, management of accented characters.

5. Validate the description window of items.

6. Save and generate the analysis. When generating the analysis, the description of the data files is modified and the full-text index is created (file with a ".FTX" extension).

Notes

- To directly create a full-text index from a text item or from a text memo item, select the item in the description window of the items and click the link named "Create a full-text index on the item".
- You have the ability to manage the full-text indexes in the data files described by programming: to do so, use a **FullTextIndex** Description variable to describe the index and **HDescribeFullTextIndex** to validate the creation of the index.
- The composition of an existing full-text index can be modified at any time: to do so, click the "Edit the index" button found in the left section of the item description in the data model editor.
- To create a Full-text Index item, a generation of the analysis and an automatic data modification must necessarily be performed.

2.8.4 How do I perform a full-text search?

The full-text searches are performed via queries: queries created in the query editor or queries created in SQL code. The search condition is entered when creating the query.

Full-text search: Query editor

To create a query performing a full-text search, all you have to do is add a "Full-text index" item to the query. The description window of a full-text search is automatically displayed.

This window is used to specify:

- The search value. This value can be entered directly (a wizard is used to build the sought string) or it can correspond to a parameter. See "Syntax of the search value" for more details.
- the sort options of the result.

When validating this window, the "full-text" item is automatically added into the query elements. A condition was automatically defined: only records whose relevance is greater than 0 will be displayed in the query result.

Full-text search: Query in SQL code

To create a query that performs a full-text search in SQL code, use the following syntax: **MATCH (<Index>) AGAINST <Condition>** where:

<Index> corresponds to the list of items found in the index separated by commas (the order of items is not important).

<Condition> corresponds to the sought string. See "Syntax of the search value" for more details.

Example:

In this example, EDT_Find is an edit control and ConnectedUserID is a variable.

```

MyQuery is string = [
    SELECT * FROM Contacts
    WHERE MATCH(Contacts.LastName,
    ...
    Contacts.FirstName, ...
    Contacts.HTMLComment,
    Contacts.RoughTextComment,...
    Contacts.Comments, ..
    Contacts.Phone, ...
    Contacts.Office,...
    Contacts.Cell, ...
    Contacts.Mail,...
    Contacts.MSN, ...
    Contacts.Internet_site, ...
    Contacts.Country, ...
Contacts.FaxNum, Contacts.City)
    AGAINST (') MyQuery = ...
    MyQuery + EDT_Search + [
    ')
AND Contacts.UserID =
]
MyQuery = MyQuery + ...
    IDConnectedUser + [
    ORDER BY Name DESC
]
HExecutesSQLQuery(QRY_SRCH,...
    hQueryDefault, MyQuery)
FOR EACH QRY_SRCH
    TableAddLine(...
        Table_Contact_by_category,
        QRY_SRCH.idcontact,...
        QRY_SRCH.IDCategory, ...
        QRY_SRCH..IDConnectedUser, ...
        QRY_SRCH.LastName,
        QRY_SRCH.FirstName)
END
CASE ERROR :
Error(HErrorInfo())
    
```

Note: Query with a parameter on a full-text index: how to ignore the parameter?

The "MATCH" of the query must not be found in the query result but in the WHERE. Indeed, if the pertinence must be included in the result, the parameter must be specified to evaluate the result.

In order for a query created with the query editor to have the MATCH statement included in the WHERE statement, the pertinence must not be displayed in the result.

Example with pertinence:

```

SELECT
MATCH(XX, YY, ZZ) AGAINST ...
    ({ParamFullText}) AS ...
    FullTextPertinence
FROM TABLE
WHERE
<Parameters>
AND PertinenceFullText > 0
ORDER BY
PertinenceFullText DESC
    
```

Example with pertinence:

```

SELECT *
FROM TABLE
WHERE <Parameters>
AND MATCH(XXX, YYY, ZZZ)
AGAINST({ParamFullText }) > 0
    
```

Syntax of the search value

The search value can contain the following elements:

Element	Meaning
A single word	The specified word will be sought. The relevance will be increased if the text contains this word. Example: "WinDev" find the word "WinDev"
Two words separated by a space character	Searches for one of the words. Example: "WinDev WebDev" find the texts that contain either "WinDev" or "WebDev".
A word preceded by the "+" sign	The specified word is mandatory. Example: "+WinDev" searches for the texts that necessarily contain "WinDev".
A word preceded by the "-" sign	The specified word must not be found in the text. Example: "-Index" find the texts that do not contain the "Index".
A word preceded by the "~" sign	If the text contains the specified word, the relevance will be reduced.

One or more words enclosed in quotes	The specified words are searched in group and in order. Caution: if "Ignore the words smaller than " differs from 0, the words in-between quotes whose length is shorter than the specified length will not be sought.
A word followed by the "*" sign	The type of the search performed is "Starts with" the specified word.

2.8.5 Analyzing the result of a "full-text" query

The result of a full-text query gives, for each record found in the data file, the relevance of the record in relation to the search value.

This relevance depends on several factors:

- the number of times the sought word is found in the record.
- the number of words in the record and their number of repetitions.
- the ratio between the records that contain the sought words and the records that do not contain the sought words. Indeed, the more the sought word is found in all the records, the less the relevance will be important.
- ...

The result of a "full-text" query can be processed like any other query result: you can for example display the result in a table, sorted according to relevance, ...

2.8.6 Managing the full-text indexes by programming

Several WLanguage functions are used to manage the full-text indexes:

HDescribeFullTextIndex	Describes a full-text index by programming for a data file created by programming.
HListFullTextIndex	Returns the list of full-text indexes of a file (query or view) recognized by the HFSQL engine

Notes:

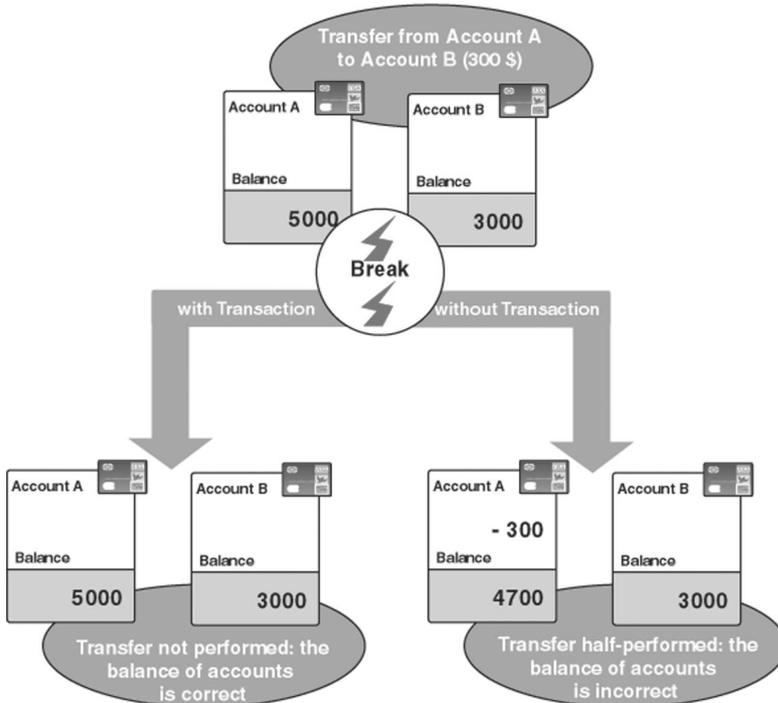
- The creation of a full-text index in a data file created by programming can be done via a **FullText Index Description** variable.
- **HIndex** is used to reindex the full-text indexes.

2.9 Transactions

2.9.1 What is a transaction?

A transaction is a set of indissociable operations: either all the operations in the transaction are performed, or none are performed. Transaction management is the best way to ensure the integrity of a set of indissociable write operations performed on HFSQL files.

For example, in an application or a banking site, a wire operation consists in a debit for one account and a credit for a second account. These two operations must be managed in a single transaction in order to avoid any incoherence (in case of power outage during the operation for example).



2.9.2 Knowing how to use the transactions according to your requirements

Tips for optimizing the management of transactions:

- Performing transactions in applications installed on reliable computers or in sites installed on reliable servers: disk transaction.

In this case, the purpose of transactions is to be able to cancel a set of operations. You have the ability to display windows or pages in the middle of a transaction, to perform different long processes, ...

Each operation performed during the transaction is saved in a transaction file. The record used is loc-

ked in write mode until the transaction is validated or canceled.

- Performing trusted write operations in files (banking sector, accounting, ...): disk transaction.

The transactions are used to insure the security of a set of operations. In this case, you must group all the write operations in your code in order to run them in a transaction. The code run during the transaction must not display any window or page, perform any file browse, etc.

This code must contain only the operations of modifying the files.

Each operation performed during the transaction is saved in a transaction file. The record used is locked in write mode until the transaction is validated or canceled.

2.9.3 Principles

Transaction principle on HFSQL files

Each write operation performed during a transaction is stored in a transaction file. The transaction can be canceled at any time: all the operations performed since the beginning of transaction will be canceled.

Once the write operations included in the transaction are completed, the program can validate the operations of the transaction.

Special cases

- Program error, the transaction is automatically canceled.
- Call to **EndProgram**, the transaction is automatically canceled.
- Canceling a test from the editor, the transaction is automatically canceled.
- Power failure or sudden stop of the application or Web server (via the task manager for example). When the application or the Web server is restarted, the consistency of the database can be restored:
 - by **WDTRANS**,
 - during the first call to **HTransactionStart**,
 - by **HTransactionCancel**.
- See “Managing the special cases”, page 352 for more details.

2.9.4 Handling the transactions by programming

Implementing the management of transactions

1. If your files are password protected, open all the files used during the transaction before the beginning of transaction.

If your files are not password protected, the files used after the call to **HTransactionStart** will automatically belong to the transaction.

2. Start the transaction with **HTransactionStart**. This function can also be used to define the name of the transaction log.

3. Perform your operations. All the write operations performed on the files in transaction are automatically saved in the transaction file. **Caution:** the processes performed are relatively slower (as each operation is saved in a specific file).

4. Cancel (if necessary) the operations performed during the transaction (**HTransactionCancel**).

5. Specify the end of transaction with **HTransactionEnd**. The transaction is validated.

Summary table of WLanguage functions used (for a HFSQL database)

HSetTransaction	Enables (or not) the management of transactions (the management of transactions is enabled by default)
HTransactionStart	Starts the transaction
HTransactionEnd	Validates the transaction
HTransactionCancel	Cancels the current transaction
HTransactionCancel HTransactionStart HTransactionEnd WDTRANS	Cancels a transaction that failed (power outage)
HTransactionInterrupted	Used to find out whether a transaction was interrupted (the transaction was neither validated nor canceled). Case of power outage.
HTransactionFree	If a record found in the specified data file is considered as being in transaction but does not belong to a transaction in progress, it is automatically freed

Handling the records during a transaction: the rules to follow

- **The records modified during the transaction can be read before or after the beginning of transaction:** they will always be taken into account in the transaction logs.

- Managing the transactions does not exclude managing the locks for the records in transaction.

Indeed, the records handled during the transaction are automatically locked in write mode.

In a WinDev application in network, if the user tries to modify a record in transaction, a message will be displayed, asking him to retry the operation.

Therefore, the transaction must be as short as possible to avoid locking the users.

Errors specific to the management of transactions

- **70031: Operation not allowed in transaction**
You are using a function that is not allowed during a transaction. For example, **HTransac-**

tionStart is used in the middle of a transaction.

• **70034: The last transaction failed**

You are trying to use a record that belongs to a failed transaction (power outage, ...). The program is restarted but the transaction is not canceled. In this case, we recommend that you cancel the transaction that failed (see below).

2.9.5 Managing the special cases

Power failure

If a breakdown (power outage, reboot, ...) occurs during a transaction, the data files may be inconsistent: the transaction was neither validated nor canceled. The transaction file is still found on the computer.

In this case, the consistency of the database will be restored:

- during the first call to *HTransactionStart*,
- by *HTransactionCancel*,
- by *WDTrans*.

Caution: Restoring the consistency of the database may take quite a long time.

Note: To find out whether the consistency of the database must be restored, check the result of *HTransactionInterrupted* in the initialization code of the project (for example).

Tip: restoring the consistency of the database

To restore the consistency of the database, the following operations are recommended:

1. Check the result of *HTransactionInterrupted* in the initialization code of the project for example.

2. If the transaction was interrupted, perform one of the following operations to restore the consistency of the database:

- call *HTransactionCancel*,
- call *HTransactionStart/HTransactionEnd*,
- start *WDTRANS*.

Example:

```

If HTransactionInterrupted() THEN
  If Confirm("The transaction performed by " + h.trrsPost + ...
    "was interrupted. Do you want to restore the integrity of the files?") THEN
    // Cancels the interrupted transactions
    IF HTransactionCancel() = False THEN
      Info("Cannot cancel the transaction")
    END
  END
END

```

Other solution: the error 70034 can also be managed in the initialization code of the project via the *WHEN EXCEPTION* keyword. Therefore, when the error 70034 occurs, the consistency of the database will be restored either by *HTransactionCancel*, or by *HTransactionStart/HTransactionEnd*.

Note: After a power outage, we advise you to reindex the data files of the application.

Error while using the update

When the execution of the application or site stops because of a programming error (division by zero for example), the current transaction is automatically canceled.

Deleting the log of transaction

The transaction log is a HFSQL file created and present only during the time of the transaction. **You must not delete this file otherwise the database may not be consistent anymore.**

2.9.6 Advanced management

Transactions: The created files

Two types of HFSQL data files are created when implementing the transactions:

- **The log of operations in transaction:** Temporary file in HFSQL format containing the different operations performed on the application or site files taken into account by the transaction. This file is created by **HTransactionStart**. By default, it is named <Project Name>_\$_TRS_OPERATION. This name can be modified by **HTransactionStart**.
- **The log of values:** Temporary file associated with each data file taken into account by the transaction. This file is named <File Name>_\$_\$_TRSVAL.Fic. For each operation performed in the transaction, this file contains:
 - the content of the record after the operation (during a deletion for example),
 - the content of the record after the operation (during an addition for example).

Identifier of the computer that performs the transaction

By default, the computer is identified by a unique number and by the name of the computer (defined in Windows).

To easily identify the computer that performs the operations in transaction, **HPost** is used to define an identifier specific to the computer. This identifier replaces the name of the computer. This identifier is saved in the log of operations in transaction and it can be consulted with WDTRANS.

By default, the computer performing the transaction corresponds to the Web server. This computer is identified by a unique internal number and by the name of the computer (name defined in Windows).

To easily identify the computer of the Web user who performs the operations in transaction, **HPost** is used to define an identifier specific to the computer of the Web user. To define a unique identifier for each Web user, call **BrowserIPAddress**. This identifier is saved in the log of operations in transaction and it can be consulted with WDTRANS.

2.10 Logs

2.10.1 General points

"Logging" the files found in an analysis is an interesting feature.

What is a log?

The log is a special file in which WinDev or WebDev automatically stores all the operations performed on one or more data files from a given time (file creation, last automatic data modification, last backup performed by WDLLog, ...).

The log contains the history of file use, which means:

- the full backup before it is used or modified by the user or by the Web user,
- the full backup after it is used or modified by the user or by the Web user,
- the author of the operation or modification,
- the date of the operation or modification,
- the nature of the operation performed: addition, modification, deletion, reading.

What is a log used for?

The log can be handled by WDLLog. The following operations can be performed from a log:

- Restore the content of a logged data file if the data file is lost or destroyed.
- Restore the content of a logged data file up to a given date.
- Find the author, the date and time of an operation performed on a specific record.
- Keep a history of file use (to calculate statistics for example).

Examples:

- The last backup was done a month ago. You made a mistake when using the file. The log enables you to retrieve your data, without losing the data from the last month of work!
- A user entered the orders of last week instead of the orders of this week during the entire morning. To avoid losing a large amount of data, we advise you to restore all the analysis files to their previous status (yesterday evening). The operation is straightforward. The person in charge of maintenance will restore the data files that were previously saved by WDLLog. Via the log, WDLLog can rebuild the files, operation by operation, from the last backup up to the chosen time.

The types of logs

files will be automatically created.

The following options allow you to log your data files. Depending on the selected option, different

Option	Action performed	Files automatically created
No log (default option)		None
Write-to-file log	All the addition, modification and deletion operations will be saved in the log. When to choose this option? To find out who modified the file and what modification was performed.	<File name>.JNL.fic
History of accesses to the file	Only the HFSQL commands used to access the file will be stored. When to choose this option? To find out the operations performed on the file. Caution: you cannot find out the value of the record before and after modification.	JournalOpération.fic JournalIdentification.Fic
Write-to-file log + History of accesses	All the addition, modification and deletion operations will be saved in the log. The HFSQL commands used to access the file will be stored as well. When to choose this option? To find out: - who modified the file, - what modification was performed, - what are the operations performed on the file.	<File name>.JNL.fic JournalOpération.fic JournalIdentification.fic

2.10.2 Implementing the log process

Defining the log process for the data files

The log process for the HFSQL data files is implemented in the data model editor.

To implement the log process on a file described in the analysis:

1. Perform a backup of the data files in their current status via WDLLog.
2. In the data model editor, display the file description ("Description of data file" from the popup menu).
3. In the "Various" tab, select the type of log to manage for this file.

4. Depending on the selected option, specify (if necessary) the directory of the different files created by the log process.

Caution: the JournalIdentification and JournalOpération files are always located in the same directory. By default, these files are created in the directory of the application or site. This directory is defined in the analysis options ("Analysis description" from the popup menu of the analysis, "Logs" tab) and it can be modified for each file.

Advice: The log files are used to save the operations performed on a file in order to replay them on a backup in case of problem (damaged disk for example). We recommend that you save the log files in directories (and even disks) different from the ones used for the data files.

Defining the log process for the items

By default, all the items of a logged file are automatically logged. However, you can specify whether some items must be logged or not.

For example, if one of the data files uses a memo item to store an image (information not that important and that does not change very often), you have the ability not log this item.

To avoid performing a log process on an item:

1. Display the description window of the data file ("Description of items" from the popup menu).
2. Select the requested item.
3. In the "Advanced" tab, uncheck "Log the item".

Generate the analysis

Once the log process was defined in the data model editor, the analysis can be generated.

Caution: Before performing this operation, we recommend that you save the data files with WDLLog.

Automatic data modification and log process

When an automatic modification of data files is performed on logged files:

1. The log files are automatically saved.
2. The log files are flushed.

2.10.3 Files created when implementing the log process

When an analysis file was described with a log option, the following files can be created:

JournalOpération.Fic	List of all the operations performed on the logged HFSQL data files used by the application or by the site. An operation corresponds to a HFSQL function.
JournalIdentification.Fic	List of the physical locations of all the logged files found in the application or site.
*JNL.Fic	File created for each logged file. Contains the backups of the records for all the operations performed by the user.

See the online help for more details.

To configure the location of these files and their passwords:

- **JournalOpération and JournalIdentification files:** by default, these files are created in the directory of the application or site. To modify this directory:
 1. Display the analysis description ("Analysis description" from the popup menu).
 2. Display the "Log" tab.
 3. Select the directory of the file and its password if necessary.

Note: This directory can also be modified for each logged file ("Description of data file" option, "Various" tab). In this case, the JournalOperation and JournalIdentification files will be created for each file at the specified location.

- ***JNL file:** By default, this file is created in the directory of the application or site. To modify this directory:

1. Display the file description ("Description of data file" from the popup menu).
2. Display the "Various" tab.
3. Select the directory of the file.

Note: The password of the *JNL.fic file and the password of the data file will be identical.

2.10.4 WDLLog: Tool for log management

WDLLog is used to:

- Save and restore your data files.
- Check the consistency of a log and clear it if necessary.
- Restore a data file from its log.
- Find out who modified a record, and when, ...

This tool can be freely distributed. See the online help for more details.

2.10.5 Handling the logs by programming

The management of logs is automatically performed. However, several WLanguage functions are used to handle the logs:

HChangeLogDir Dynamically modifies the location of the log files corresponding to a HFSQL file (*JNL files and JournalOperation and JournalIdentification files).

HSetLog Used to enable (or not) the management of logs. This management is enabled by default.

HLogInfo Adds comments into the log when saving the logged operation. These comments can be viewed in WDLLog.

HLogRecreate Re-creates an empty log. This function is used for example to reset the log to 0 after a backup or a replication. The content of the existing files is lost.

HLogRestart Restarts the log process on a file. This log process was stopped by *HStopLog*.

HLogStop Stops the log process of a file. The operations performed in the logged file are not saved anymore.

HRegenerateFile Regenerates a file from its log.

The WLanguage properties can also be used to manage the logged files:

LogFile Used to find out whether a data file is a log file or not.

LogMethod Identifies the log mode used for a data file (defined in the data model editor or dynamically).

LogDirectory Used to manage the directory of the log file described in the analysis. You can:

- Find out the directory of the log for a file defined in the data model editor or dynamically.
- Define the directory of the log for a file that was defined dynamically.

2.11 Automatic modification of data files

2.11.1 Principle

The automatic modification of data files is used to update the description of the data files found on the user computers or on the data servers.

Indeed, if the structure of one or more files has evolved on the development computer (addition, deletion or modification of items), these modifications must necessarily be applied to the user computers or to the data servers when updating the

application or the site.

If the application or the site is updated without performing any automatic data modification, the application or the site may:

- no longer operate properly,
- generate programming errors.

Caution: For a network update with automatic modification of data files, the update must necessarily be installed on the user computers (WinDev

2.11.2 When is the automatic data modification required?

The automatic data modification is required in the following cases:

	Development computer	User computers/Data server
Case 1: Modifying the structure of the data files in HFSQL format	The application or the site is using data files in HFSQL format. The structure of the data files was modified (addition or deletion of items, ...).	The application or the site is using data files in HFSQL format. These modifications must necessarily be applied when updating the application or the site. Therefore, the structure of the files will be identical to the one of the development computer.
Case 2: Migrating Hyper File 5.5 data files (WinDev 5.5 or WebDev 1.5) to HFSQL Classic	The application or the site is using data files in HFSQL Classic format.	The application or the site is using data files in Hyper File 5.5 format (WinDev 5.5/WebDev 1.5). The files must necessarily be migrated from Hyper File 5.5 to HFSQL Classic when updating the application or the site. Therefore, the structure of the files will be identical to the one of the development computer.
Case 3: Modifying the structure of data files in Hyper File 5.5 format (WinDev 5.5/WebDev 1.5)	The application or the site is using data files in Hyper File 5.5 format (WinDev 5.5/WebDev 1.5). The structure of the data files was modified (addition or deletion of items, ...).	The application or the site is using data files in Hyper File 5.5 format (WinDev 5.5/WebDev 1.5) These modifications must necessarily be applied when updating the application or the site. Therefore, the structure of the files will be identical to the one of the development computer.

Note: For a network setup, the automatic modification of data files modifies both the data files found on the server and the ones found on the user computers (WinDev only).

2.11.3 Performing the automatic data modification

Development computer

The automatic modification of data files is systematically performed when the analysis is generated on the development computer.

To apply the automatic modification to the data files found on the user computers or on the data servers, the automatic modification must be performed when installing the application update or the site update. The configuration of this automatic modification can be performed when creating the setup program.

Deployment computer

When installing an update with automatic modification of data files, the user or the site manager will be able to configure the automatic modification by clicking the "Advanced" button. This button is found on the first plane of the setup program. A window is displayed, allowing you to:

- Create a report file for the operations performed during the automatic data modification. If a problem occurs, this file can be transmitted to the application or site manager.
By default, this file is named "LOGMODAUTO.TXT" and it is created in the setup directory of the application or site.
- List the data files in HFSQL format onto which the automatic modification will be performed. You will have the ability to add to this list:
 - additional data files,
 - additional directories containing data files.
- Specify additional directories containing data files in Hyper File 5.5 format that are not automatically found during the update.

Notes:

- During the update, the data files automatically found are:
 - the files found in the setup directory of the application or site,
 - the files listed in the ".REP" file.
 See "Reassigning data files", page 344 for more details.
- To prevent the user from accessing the options for configuring the automatic modification, WDINST (the setup editor) allows you to hide the "Advanced" button in the setup program (WinDev only).

Forcing the automatic modification of files

On the development computer and on the deployment computer, an automatic modification of data files can be performed at any time.

Indeed, in some cases, the automatic data modification cannot be performed properly: data files found on a laptop computer not connected to the network at the time of update, faulty update, ... In this case, the automatic modification of data files must be forced in order for the application or the site to operate. WModFic allows you to force the automatic modification of data files.

2.11.4 Notes

Running the automatic modification of data files several times

The automatic modification will have no effect on the application or the site if it is performed on data that is already updated.

Default configuration of the setup program

If the application or the site is associated with an analysis, by default, the setup program proposes to perform the automatic modification of data files. The user or the site manager can access the configuration options of the automatic modification by clicking the "Advanced" button.

Saving the data files

The data files of the application or site are automatically saved before performing the automatic modification. The backup directories are named:

- "Backup of Auto Modif (<DateAutoModif><TimeAutoModif>)" for an automatic modification of data files without migration
- "Files before conversion (<DateAutoModif><TimeAutoModif>)" for a migration of data files from Hyper File 5.5 to HFSQL.

2.12 Creating dynamic (or temporary) files

In WLanguage, you have the ability to describe temporary HFSQL files by programming.

To describe temporary files, you must:

1. Declare the "File description", "Item description" and "Link description" variables (if necessary). These types are presented from page 56.
2. For each file:
 - describe the characteristics of the file via the HFSQL properties,
 - describe the characteristics of the items via the HFSQL properties,
 - validate the description of each item (*HDescribeItem*),
 - validate the description of the file (*HDescribeFile*).
3. Describe (if necessary) the characteristics of the links via the HFSQL properties.
4. Validate (if necessary) the description of each link (*HDescribeLink*).

To describe a temporary file, an HFSQL analysis must be associated with the current project.

Notes:

- A dynamic file can be linked to a window, a page, a report or a control via the WLanguage properties.
- The dynamic files can be modified by the automatic modification of files.
- A dynamic file can be reindexed by WDTool or by *HIndex*.
- The functions used to describe the temporary files cannot be used in external language. They cannot be used to modify a file described in the data model editor of WinDev/WebDev (non-temporary file)
- Specific xBase functions are used to create temporary xBase files.

2.13 Retrieving the structure of the HFSQL files found in an analysis

You have the ability to retrieve the structure of the files found in an analysis. This enables you to create a tool such as the file viewer (WDMAP).

The following functions are used to retrieve the structure of the files (they cannot be used on xBase files).

HCloseAnalysis	Closes an analysis that was opened by <i>HOpenAnalysis</i>
HListAnalysis	Returns the name of the analyses and the drive on which they are installed
HListKey	Returns information about the keys of an analysis
HListFile	Returns the name of the files found in an analysis
HListItem	Returns the structure (items) of a file
HOpenAnalysis	Opens an analysis to access its data files

HRetrieveItem	Returns the content of the specified item for the current record
HToItem	Writes the content of the specified item for the current record

Notes:

- *HOpenAnalysis* accepts in parameter an optional password if a password was defined during the analysis description (WDMAP password). This password protects the access to the analysis by programming.
- To use *HListFile*, *HListItem* and *HListKey*, there is no need to open the analysis with *HOpenAnalysis*.
- To retrieve the structure of the data files found in an analysis, if the analysis is associated with the project, there is no need to open the analysis with *HOpenAnalysis*. The description file of the analysis (WDD) must be installed in the project library (WDL) or in the analysis directory.

2.14 Speeding up processes and optimizing an application or a site

Several management mechanisms are enabled by default when creating a WinDev application or a WebDev site.

To optimize the response time of your applications or sites, we recommend that you "disable" the unnecessary management mechanisms. The following management mechanisms are enabled by default:

- the transactions,
- log,
- the memos,
- the .REP,
- the replication,
- the triggers,
- RPC.

For each one of these management mechanisms, let's see a summary of the feature and how it can be disabled by programming.

Caution: All these features must be disabled in the initialization code of the project.

2.14.1 Management of transactions

The management of transactions is used to keep the integrity and consistency of a database, regardless of the operation performed on the database. See "Transactions", page 350 for more details.

If no file is in transaction, the management of transactions is useless. It can be disabled by *HSetTransaction*.

2.14.2 Managing the log

The log file is a specific file used to store all the write operations performed on a data file since the last backup. Its operating mode is presented in "Logs", page 353.

If no file is logged, *HSetLog* is used to disable the management of logs.

2.14.3 Managing the memos

The management of memos is required when the files contain memo items. Their management is described in "Managing the "memo" files", page 343.

If no file contain memo items, their management can be disabled by *HSetMemo*.

2.14.4 Managing the ".REP"

The ".REP" file contains the physical name and directory of the HFSQL data files found in an application or site. See "Reassigning data files", page 344 for more details.

If your application or site does not required the management of ".REP", it can be disabled by *HSetREP*.

2.14.5 Management of replication

The data replication is used to keep the data files found on different computers updated. Its use is presented in "The data replication", page 386.

If your application or site does not use the replication, its management can be disabled by *HSetReplication*.

2.14.6 The management of triggers

A trigger corresponds to the association between a procedure and a modification action performed on a record found in one or more files. Its use is presented in "HFSQL triggers", page 380.

If no trigger is used in the application or site, the management of triggers can be disabled by *HSetTrigger*.

2.14.7 The management of RPC

The RPC is used to consult a HFSQL database via Internet/Intranet or STW (Switched Telephone Network). The management of RPC is presented in the online help.

If your application or site does not manage the remote access, we advise you to disable its management via *HSetRemoteAccess*.

2.15 ODBC driver on HFSQL

2.15.1 Overview

The ODBC driver on HFSQL allows you to access a HFSQL database from an external database software that supports the accesses by ODBC.

The HFSQL data can be accessed in read and write mode.

Setup

When installing WinDev/WebDev on the development computer, you have the ability to install the ODBC driver on HFSQL.

Furthermore, the setup of the ODBC driver on HFSQL can be included when configuring the setup program of your applications or sites.

2.15.2 Configuration

To use the ODBC driver for HFSQL:

1. Start the administrator of ODBC data (ODBCAD32.EXE) on your computer or on the server. Select for example "Start .. Run" in Windows and type "ODBCAD32.EXE".
2. Select the "User database" tab.

3. Click the "Add" button.
4. Select the "HFSQL" driver.
5. Click the "Done" button.
6. Enter the name of the HFSQL data source. This name will be used to identify the HFSQL database in the external programs.
7. Click the "Details" button.
8. Select the WDD file corresponding to the analysis ("Browse" button).
9. In the list of analyses, select the requested analysis and the directory of the data files ("Browse" button). All the HFSQL data files corresponding to the selected analysis are grouped in this directory.

Caution: a file directory must be selected for each analysis.

10. Validate ("OK" button).

The database can be used in read/write mode from external programs via the ODBC driver on HFSQL .

For more details about the ODBC driver for HFSQL Classic, see the online help.

2.16 ODBC on HFSQL via J++ and JDBC

2.16.1 Overview

The ODBC driver on HFSQL is used to access a HFSQL database from an external database software that supports ODBC accesses. This gives you the ability to use the ODBC driver on HFSQL 7 via J++ and JDBC.

You must use:

- the Microsoft or Sun JDBC driver
- Sun JDK or Visual J++.

Caution: The following limitations apply when using the ODBC drivers on HFSQL via J++ and JDBC:

To prevent the other programs from modifying your HFSQL data, the driver is read-only. The ODBC driver on HFSQL is a level 2 ODBC driver. See a specific documentation for more details.

2.16.2 Setup

The ODBC driver on HFSQL can be installed when WinDev or WebDev is installed on the development computer.

Furthermore, when configuring the setup program of your WinDev applications, you have the ability to include the setup of the ODBC driver on HFSQL.

2.16.3 Configuration

To use the ODBC driver on HFSQL, you must configure the ODBC driver:

1. Start the administrator of ODBC data (ODBCAD32.EXE) on your computer. Select for example "Start .. Run" in Windows and type "ODBCAD32.EXE".
2. Select the "User database" tab.
3. Click the "Add" button.
4. Select the "HFSQL" driver.
5. Click the "Done" button.
6. Enter the name of the HFSQL data source. This name will be used to identify the HFSQL database in the external programs.
7. Click the "Details" button.
8. Click the "Browse" button to select the WDD file corresponding to the analysis.
9. In the list of analyses, select the requested analysis and the directory of the data files ("Browse" button).

All the HFSQL data files corresponding to the selected analysis are grouped in this directory.

Caution: a file directory must be selected for each analysis.

10. Validate ("OK" button).

The database can be used in read-only mode from the external programs via the ODBC driver on HFSQL.

2.16.4 Use

To use the ODBC driver on HFSQL from your Java program, you must:

1. Define the driver used. For example, with the following code line:

```
// Use the JDBC driver of Microsoft
Class.forName("com.ms.jdbc.odbc.JdbcOdbcDriver");
// Use the JDBC driver of Sun
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

2. Define the URL on the system ODBC connection ("hfodbc" for example):

```
String jURL = "jdbc:odbc:hfodbc";
// Connection
Connection Contact = DriverManager.getConnection(jURL, "<user>", "<pass>");
```

3. Interrogate the HFSQL database in SQL. For example:

```
// Creates a query
Statement jQuery = Contact.createStatement();
// Run the query and retrieve
// ...
ResultSet Result = jQuery.executeQuery("SELECT * FROM CUSTOMER");
int jColumn = 5;
int jRow = 3;
```

```
ResultSetMetaData jMetaData= ...
    Resultat.getMetaData();
for (int i=0;i<jLine;i++)
Result.next();
System.out.println("Column Name : "
+ jMetaData.getColumnLabel(jColumn));
System.out.println("Value : " +
Result.getString(jColumn));
Result.close();
jQuery.close();
Contact.close();
```

Note: The ODBC driver on HFSQL cannot be used to access an encrypted HFSQL database.

2.17 OLE DB provider for HFSQL

2.17.1 Overview

The OLE DB provider for HFSQL is used to access a HFSQL database (Classic or Client/Server) from an external software that manages the accesses by OLE DB.

The provider is available in read/write mode. An application written in external language can be used to read and write in HFSQL files.

See a specific documentation about OLE DB for more details.

2.17.2 Setup

The OLE DB provider for HFSQL is supplied as a setup pack available in the "Install\OLEDB" directory of WinDev/WebDev.

By default, the OLE DB provider is installed in the "C:\Program Files\Common files\PC SOFT\19.0\OLEDB" directory.

Note: In order for an application to use an OLE DB provider, you must install the MDAC component (Microsoft Data Access Component) version 2.8 or later. This component can be downloaded from the Microsoft site. It is included in the operating system from Windows 2003 Server and Windows Vista.

2.17.3 Configuration

The application that uses an OLEDB provider must supply a connection string. This string defines the provider to use as well as the connection parameters that must be given to this provider to establish the connection to the database.

This string (called connection string) can be:

- entered in programming (in an application in C# or VB.Net for example).
- built via a wizard (with Crystal Report for exam-

ple).

The format of the connection string is as follows:

```
<Element1>=<Value1>;<Element2>=<Value2>;...;<ElementN>=<ValueN>
```

See the online help for more details.

Using the analysis in a connection to a HFSQL Classic database

For a connection to a HFSQL Classic database:

- If the path of the WDD file is not specified in the "Data Source" parameter of the connection string:
 - Only the files described in this analysis will be taken into account by the connection.
 - The links and the integrity rules described in the analysis are automatically taken into account.
- If the path of the WDD file is specified in the "Data Source" parameter of the connection string:
 - All the files found in the directory specified in the "Initial Catalog" parameter will be taken into account.
 - No integrity constraint is automatically respected between the files.

Notes:

- During an access by the OLE DB provider for HFSQL, the automatic assistance windows of HFSQL are disabled.
- The "Duration" items found in the HFSQL data files are returned as 8-byte integers by the OLE DB provider. The unit is the millisecond.
- The array items are not supported by the OLE DB provider.
- In this version, the SQL commands with parameters are not supported.

Examples of OLE DB connection strings

- Connection to a HFSQL Classic database without specifying the analysis:

```
/Provider=PCSOFT.HFSQL;Initial  
Catalog=c:\My HFSQL Database
```

- Connection to a HFSQL Classic database and specifying the analysis:

```
Provider=PCSOFT.HFSQL;Data  
Source=c:\My HFSQL Database\ ...  
\MyAnalysis.wdd;Initial ...  
Catalog=c:\My HFSQL Database
```

- Connection to a HFSQL Client/Server database:

```
Provider=PCSOFT.HFSQL;Data  
Source=serverdb.  
mycompany.fr:4910;User...  
ID=admin;Password=secret;...  
Initial Catalog=MyDatabase
```

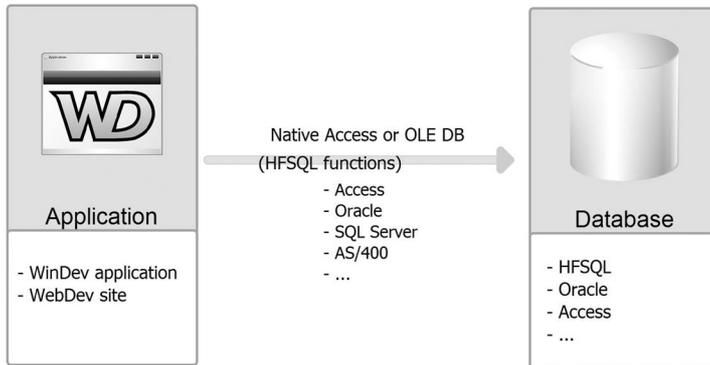
- Connection to a HFQL Client/Server database with password-protected files that use the Russian character set:

```
Provider=PCSOFT.HFSQL;Data  
Source=serverdb.mycompany.fr...  
User ID=user;...  
Initial Catalog=MyRussianDB; ...  
Extended Properties= ...  
"Password=*:secret1; ...  
Password=File2:secret2;...  
Language=KOI8-R"
```

Note: In this example, all the database files are protected by the "secret1" password except for the "File2", that is protected by the "secret2" password.

3. SQL LANGUAGE

3.1 Overview



A program developed with WinDev or WebDev can access a database according to several modes:

- Native Access
- OLE DB

- ODBC via OLE DB

This chapter presents how to access:

- HFSQL files via SQL,
- files of external database by SQL.

3.2 SQL language HFSQL

3.2.1 Overview

Several WLanguage functions can be used to interrogate a HFSQL database via SQL queries (*HExecuteQuery*, *HExecuteSQLQuery*).

The HFSQL files can be interrogated:

- By queries created with the query editor.
See the "Query and report generator" guide for more details.
- By SQL queries directly entered in the code editor (*HExecuteSQLQuery*).
- Via the ODBC driver on HFSQL supplied with WinDev/WebDev, from an application or a site not developed with WinDev/WebDev.

3.2.2 SQL commands that can be used with HFSQL

The following SQL commands can be used on HFSQL files:

- ADD_MONTH
- ALL
- AND
- ANY
- AS
- ASC

- ASCII
- AVG
- BETWEEN
- BIN
- BOTTOM
- CASE
- CBRT
- CEIL - CEILING
- CHAR_LENGTH
- CHARACTER_LENGTH
- COALESCE
- CONCAT
- COUNT
- CREATE TABLE
- DECODE
- DEGREES
- DELETE
- DESC
- DISTINCT
- DIV
- DROP TABLE
- EXISTS
- FROM
- FULL OUTER JOIN
- GROUP BY
- HAVING
- HEX
- IN
- INNER JOIN
- INSERT
- INSTR
- INTO

- IS NULL/IF NULL
- LAST_INSERT_ID
- LEFT OUTER JOIN
- LENGTH
- LIMIT
- LPAD
- MAX
- MIN
- NEW_TIME
- NOT
- OCT
- ON
- ORDER BY
- POSITION
- RANDOM
- RIGHT
- ROUND
- RTRIM
- SET
- SOUNDEX
- SOUNDEX2
- SUBSTR
- SUM
- TOP
- TRIM
- UNION
- UPPER
- VALUES
- LAST_DAY
- LEFT
- LEN
- LIKE
- LOWER
- LTRIM
- MID
- MONTHS_BETWEEN
- NEXT_DAY
- NVL
- OCTET_LENGTH
- OR
- PATINDEX
- RADIAS
- REPLACE
- RIGHT OUTER JOIN
- RPAD
- SELECT
- SOME
- SOUNDEX LIKE
- SOUNDEX2 LIKE
- SUBSTRING
- SYSDATE
- TRANSLATE
- TRUNC
- UPDATE
- UUID
- WHERE

All the other SQL commands not listed here are not supported.

Note: We will not go into details about these commands. We recommend that you read a specialized book or the online help if necessary.

3.2.3 Functions for running queries

The queries created with the query editor will be initialized by *HExecuteQuery*.

The queries in SQL language will be initialized by *HExecuteSQLQuery*. To retrieve the content of an SQL query, a data source is required (or a browsing table, combo box, ... linked to the query).

Important: A "data source" variable must always be declared as GLOBAL to the window, to the page or to the project. Otherwise, it could not be used to fill a table for example because its lifespan would be too short.

If a "data source" variable is declared as a LOCAL variable, a warning will be generated.

Example of query in the query editor (the query is named EditorQuery.WDR) :

```
SrcQuery is data source
IF HExecuteQuery (SrcQuery,...
    EditorQuery) THEN
    Error (HErrorInfo ())
ELSE
    HReadFirst (SrcQuery)
    WHILE NOT HOut ()
        // Browse the result of the query
        HReadNext ()
    END
END
```

Example of query in SQL language:

```
SrcQuery is data source
IF not HExecuteSQLQuery (...
    SrcQuery, "SELECT CustomerName,...
    "+...
    "Address FROM CUSTOMER WHERE "+...
    "CUSTOMER.City LIKE 'PARIS%'" )
THEN
    Error (HErrorInfo ())
ELSE
    HReadFirst (SrcQuery)
    WHILE NOT HOut ()
        // Browse the result of the query
        HReadNext ()
    END
END
```

Specific features of queries for database update (UPDATE, DELETE, INSERT, ...)

As these queries are not intended to return a selection of records, a data source is not required to run them. *HExecuteQuery* and *HExecuteSQLQuery* can be called with a query name instead of a data source.

Example of SQL DELETE query:

```
IF NOT HExecuteSQLQuery(...
    "DeletionQuery",...
    "DELETE FROM CUSTOMER WHERE "+...
    "CUSTOMER.CustomerNum=346") THEN
    Error (HErrorInfo())
ELSE
    Info("Deletion performed")
END
```

Passing parameters to a query

A query on a database can receive parameters (the selection of customers living in state X for example).

To do so, *HExecuteQuery* is used to pass parameters to the query created in the query editor. See the help about the query editor and *HExecuteQuery* for more details.

A query in SQL language (initialized by *HExecuteSQLQuery*) can also accept parameters. To do so, the string that constitute the SQL query must be built by concatenation with the parameters.

For example, to find a customer whose number was entered on the screen:

```
CSQLQry is string
SrcQuery is data source
// EDT_CustomerNum is
// an edit control of the window
cSQLQry="SELECT CustomerName"+ ...
    "FROM CUSTOMER WHERE "+...
    "CUSTOMER.NCustomer=" +...
    EDT_CustomerNum
IF NOT HExecuteSQLQuery(...
    SrcQuery,cSQLQry) THEN
    Error (HErrorInfo())
ELSE
    HReadFirst (SrcQuery)
    Info (SrcQuery.CustomerName)
    //Display the name found
END
```

3.3 SQL language and other databases

3.3.1 Overview

WinDev or WebDev allows you to interrogate an external database:

- via WLanguage (*HExecuteSQLQuery*)
- via a query created in the query editor.

3.3.2 Query created in the query editor

The steps for creating a query on an external database are as follows:

1. Importing the tables from the external database (description of structure) into the project analysis. To perform this operation, on the "Analysis" pane, in the "Creation" group, expand "Import" and select "Import descriptions of files/tables". The wizard allows you to connect to a server or to a database and to select the tables to import.
2. Create the queries with the query editor.

3.3.3 Query created by programming

The steps for creating a query on an external database are as follows:

1. Declare the connection in the data model editor.
2. Two methods are available in programming:

Method 1: Declaring a data source.

Using *HExecuteSQLQuery*.

Method 2:

- Opening the connection with *HOpenConnection*.
- Running the query with *HExecuteSQLQuery*.
- Browsing the query result with the *HReadXXX* functions.
- Closing the connection with *HCloseConnection*.

Declaring the connection

In the data model editor, on the "Analysis" pane, in the "Connection" group, click "New connection".

Programming

The following code is used to run an SQL query on a database accessible via the connection described in the analysis.

Note: All the SQL commands specific to the external database can be used.

```
// Declaring the data source
Customer75 is data source
// Find customer in California
// SalesManagement: Name of the
// connection described
// in the analysis
IF NOT ...
  HExecuteSQLQuery(Customer75,...
    SalesManagement,...
    hQueryBreakable,...
    "SELECT * FROM"+...
```

```
" CUSTOMER WHERE STATE "+...
"LIKE '75%') THEN
  Error(HErrorInfo())
  RETURN
END
// Retrieve the result
FOR EACH Customer75
  CalculateStats()
END
```

4. MANAGING THE FILE LOCKS

4.1 Overview of locks

WD WebDev

4.1.1 Locking files

Why lock the files?

In Windows, several programs running on the same computer or on computers connected via a network can work on the same data file at the same time.

Therefore, the management of files implies the management of concurrent accesses. Specifically, you must prevent several programs from modifying the same record at the same time, by locking the file or the record.

WLanguage allows you to easily manage the lock of a file or the lock of a file record.

Locking files by programming

Several WLanguage functions are used to lock the files or records and to get status report about the lock.

These functions allow you to lock:

- only the record read in write mode,
- the entire file in write mode,
- the entire file in read and write mode.

They also allow you to unlock the locked files or records.

4.1.2 What is a lock?

A lock consists in locking a file (or a file section) to temporarily forbid the access to this file (or to this file section) by another program. A lock can be:

- *in write mode only*, in this case the file (or the file section) can be read by another program while the lock is active;
- *in read and write mode*, in this case the file (or the file section) remains entirely inaccessible to the other programs.

4.1.3 When should you lock and what should you lock?

Some basic rules to manage the locks by programming:

1. Each record that is going to be modified, deleted or rewritten must be read and locked first, by a locking read function (not mandatory).

king read function (not mandatory).

The record is automatically unlocked by the modification, deletion or rewrite operations.

2. When a record is locked by a locking read function and when it is not unlocked by a modification, deletion or rewrite operation, it must be unlocked by an unlocking function.

3. When several records must be unlocked, they can be unlocked in a single operation by closing the file.

4. Only the program that locked the records and the files can unlock them.

5. When a file is locked, there is no need to read the record with the locking read functions. All you have to do is use a "simple" read function.

6. As soon as a file is no longer required by a program, at least for some time, we advise you to close it. Therefore, it remains accessible to other programs.

4.1.4 WLanguage and the locks

WLanguage proposes two management modes of locks. The developer chooses the lock mode according to the final use of the application or site that is developed and according to the requested programming.

The lock modes are as follows:

- **Single-user mode:** Each opened file is automatically locked in read and write mode. Therefore, there is no need to manage the locks and to check the status reports of locks. *The files cannot be shared.*
- **Multiuser mode:** Any opened file can be shared between other programs and other computers. The developer manages the lock of files and records as well as the status reports of locks. He decides which actions to perform in each case.

The Single-user mode is the default lock mode of WinDev.

The Multi-user mode is the default lock mode of WebDev.

4.2 Managing locks

WD WebDev

4.2.1 Example illustrating the need for locks

Several examples of lock processes are presented below. They illustrate the need to lock the files.

- To display a list used to select one or more records, there is no need to lock the displayed records.
However, once the selection is performed, in most cases, the corresponding record must be read to check its existence and it must be locked so that it can be modified if necessary.
- If one or more records are added into a file, the file must be locked in order to calculate the value of an item from another record (customer number for example). In the other cases, there is no need to lock the file.
- During a statistical process used to calculate the weight of one or more articles in relation to all the articles, the file must be locked or modifications must be forbidden before the process in order to make sure that the file content will not be modified during the calculation.
- The management of locks often depends on the situation in which a file is processed. To simplify the management of locks, for a cash register program in a store for example, the same customer cannot be at several cash registers at the same time. Therefore, there is no need to lock the record in the customer file during its process even though it is modified.

4.2.2 Structure of locks

Two lock levels are proposed:

- locking an entire file,
- locking a given record.

Locking an entire file automatically locks all the records found in this file.

Locking an entire file is possible only if no record in this file is locked by another computer or another program.

Unlocking a file automatically unlocks all the records locked by the same program.

Important: The locks remain active as long as an unlocking function (on the file or record) or a write function (unlocks the written record) is not run.

4.2.3 Dead lock (inter locking)

When managing the locks, a situation can occur where all the files are locked and cannot be unlocked. This is called a "Dead lock".

To avoid a dead lock, we recommend that you follow these three steps:

1. Lock all the files used in a process according to their alphabetical order.
2. Perform the requested process.
3. Unlock the files.

Let's look at a specific example. The example below presents a case of "dead lock".

Let's assume that the P1 and P2 programs perform the following process (do not do!):

Program P1	Program P2
LOCK(F1)	LOCK(F2)
LOCK(F2)	LOCK(F1)
< Process 1 >	<Process 2>
UNLOCK(F1)	UNLOCK(F2)
UNLOCK(F2)	UNLOCK(F1)

The programs perform their locks in the following order:

- P1 locks F1.
- P2 locks F2.
- P1 wants to lock F2 but this one is already locked by P2.
- P2 wants to lock F1 but this one is already locked by P1.

Both programs wait for the file to be unlocked: there is no solution if neither of the 2 programs frees one of the files!

To avoid this situation, let's follow our previous tips:

1. Lock the files according to the alphabetical order (F1 then F2).
2. Perform the requested process.
3. Unlock the files.

The P1 and P2 processes become:

Program P1	Program P2
LOCK(F1)	LOCK(F1)
LOCK(F2)	LOCK(F2)
<Process 1>	<Process 2>
UNLOCK(F1)	UNLOCK(F1)
UNLOCK(F2)	UNLOCK(F2)

In this case:

- P1 locks F1.
- P2 cannot lock F1.
- P1 locks F2.
- P2 waits for F1 to be freed.
- The process 1 is run.
- P1 unlocks F1.
- P1 unlocks F2.
- P2 locks F1.
- P2 locks F2. ...

A dead lock is avoided!

4.3 The available lock modes

WD WebDev

Two modes are available for locking the files and records.

This chapter:

- helps you choose the proper mode for your programs,
- explains the method for locking a file or a record,
- explains the operating mode of each mode.

4.3.1 Single-user mode

The Single-user mode is the default lock mode of WinDev.

It is characterized by the automatic lock of each file as soon as it is opened. The file will be unlocked when it is closed.

WinDev 5.5 user: this mode corresponds to *HModeAuto* of WinDev 5.5.

This mode is kept for backward compatibility with WinDev. It must not be used in a WebDev site.

WeDev 1.5 user: this mode corresponds to *HModeAuto* of WebDev 1.5.

When should this mode be used?

The Single-user mode is used to easily develop programs whose files are not shared by several computers or by several programs at the same time.

With the Single-user mode, there is no need to manage the locks by programming, the files are locked as soon as they are opened in read/write. The files cannot be shared.

Important

- The Single-user mode is not suitable for the programs operating in network.
If a program developed in Single-user mode is run several times on the same computer or is run in network, an error message will be displayed, indicating that the Single-user mode is not appropriate for this type of operating mode.
- A program developed in Single-user mode cannot be run several times in Windows.

How are the files locked in Single-user mode?

The Single-user mode is the default lock mode of WinDev.

In Single-user mode, as soon as a file is opened by a function (*HOpen* or any other function, for example *HReadFirst*, *HCreation*, ...), the file is automatically locked for in read/write.

If access to the file is refused (the file is already locked by another program), the program execution is terminated after the display of a message indicating that this mode is not appropriate.

A file that was locked during its opening is automatically unlocked as soon as it is closed:

- by *HClose*, if it is called,
- to be able to open another file when the maximum number of files that can be opened is reached (automatic file closing).

Notes

- If the file is unlocked by *HClose* (or *HUnlockFile*, *HUnlockEntireFile*) or by the automatic closing, in the rest of the program, it will be locked again when it is re-opened by a function.
- In Single-user mode, the file is locked as long as it is opened. Therefore, we advise you to keep a file locked for the smallest amount of time.
We recommend that you close a file as soon as it is not used anymore.

How to share data in Single mode?

In Single-user mode, the files cannot be shared. However, in some analysis, two programs can be run at the same time, one modifying the files, the other one reading the files. In this case, the programs in Single-user mode can access the file at the same time.

If a program (modifying files) developed in Single-user mode is run several times on the same computer or is run in network, an error message will be displayed, signaling that the Single-user mode is

not appropriate for this type of operating mode

Let's assume two programs for example:

- P1 reads the "ORDERS" file,
- P2 modifies the "ORDERS" file.

P1 uses the following algorithm:

```
// Display the list of orders
ResOpen = HOpen(ORDERS, hORead)
IF ResOpen = False THEN
  Info("File not found") ELSE
  HReadFirst(ORDERS, NUM)
  WHILE not HOut()
    Display_order
  HReadNext(ORDERS, NUM)
  END
END
```

P uses the following algorithm:

```
//Delete the orders containing the
PdtCode product
PdtCode = Enter_product_code
HReadSeek(ORDERS, CODE, PdtCode)
WHILE HFound()
  HDelete(ORDERS)
  HReadNext(ORDERS, CODE)
END
```

If P2 is started first then P1 second, then:

- P2 opens and locks the "ORDERS" file
- P1 opens the "ORDERS" file even though it is locked. **HOpen(hORead/hOReadWrote)** opens the file whether it is locked or not.
- P1 can read the file and P2 can modify the file.

Caution: if P2 deletes the record "X" from the "ORDERS" file while P1 reads the record "X" in the "ORDERS" file, P1 will not see that the record "X" is deleted from the file.

To share a file between a program modifying a file and one or more programs reading the file, the program that modifies the file must call a function to open and lock this file, the other programs must call **HOpenNoLock** to access the file even though it is locked.

Note: the functions are presented at the end of this section.

Caution: If the file is opened by **HOpen(hORead/hOReadWrite)**, the file is not locked. Therefore, the file is not protected against potential modifications. This is why **HOpen(hORead/hOReadWrite)** must only be used in the programs that do not modify the file.

Therefore, the following statements must not be run after **HOpen(hORead/hOReadWrite)**:

- **HAdd** - **HWrite**

- **HFree** - **HModify**
- **HCross** - **HDelete**
- **TableSave** - **TableDelete**

Basic rules in Single-user mode

To properly manage the locks of files/records, we advise you to follow these rules:

- The Single-user mode must not be used when the files are shared by several programs.
- The Single-user mode is not intended to develop a program that operates in network.

Summary

In Single-user mode, whenever a file is opened by a function (except for **HOpen(hORead/hOReadWrite)**), it is automatically locked in read/write. The file is automatically unlocked when it is closed. It will be locked again during the call to a function (except for **HOpen(hORead/hOReadWrite)**).

HOpen(hORead/hOReadWrite) opens the file without locking it, no matter whether it is already locked by another program or not.

The Single-user mode is suited for the programs that do not share their files.

Calling the Single-user mode

The Single-user mode is the default lock mode of WinDev.

Note: In a program, you have the ability to switch from the Single-user to the Multi-user mode.

Example in Single-user mode

The following example presents the lock and unlock mechanism used by the Single-user mode. The functions are presented in details at the end of this section.

You will notice that none of these programs includes functions for locking files.

```
// Find a customer according to his
//last name
HReadSeek(CUSTOMER, ...
  LASTNAME, "MARTIN")
IF HFound() THEN
  Info("Customer found")
ELSE
  Info("Unknown customer")
END
```

Note: After the call to **HReadSeek**, the "CUSTOMER" file is opened and locked. It is unlocked at the end of the program execution.

4.3.2 Multi-user mode

The Multi-user mode is the default lock mode of WebDev.

The Multi-user mode is characterized by:

- the locks are managed by the developer in the programs,
- the status reports of locks (to find out whether some file records are already locked by another program) are managed in the programs by the developer who decides on the processes to perform.

WinDev 5.5 user: this mode corresponds to *HModePerso* of WinDev 5.5.

When should this mode be used?

The Multi-user mode must be used when one or more program files must be shared by several programs.

Therefore, the Multi-user mode must be used as soon as a logical update operation performed on several files cannot be interrupted: either all the files are updated or none.

In WinDev, the Multi-user mode proposes an automatic management:

- of lock errors,
- of modification conflicts.

This automatic management can be customized at any time by *HOnError*.

See the online help for more details.

Note: the management of locks used in WinDev 5.5 is available for backward compatibility.

Calling the Multi-user mode

To call the Multi-user mode in a program, all you have to do is run *HMode(hModeMulti)* at the beginning of the program

Two methods are available for implementing the locks:

- **Direct mode:** *hModeDirect* constant (default value): implement the priority locks (on the modification for example).
Therefore, a record on which several read operations are performed will not be modified but the locks are faster.
- **Reservation mode:** *hModeReservation* constant: implement the non-priority locks.
This method enables you to immediately modify a record on which several read operations are performed. This method is slower than the *hModeDirect* method.

Summary

The Multi-user mode allows you to share one or more files between several programs.

The developer manages the file locks by programming. He must also check *HErrorLock* after all the *HFSQL* functions.

The Multi-user mode is suited for the programs that manage files that can be shared. The programs process a logical update operation of several files that cannot be interrupted.

4.3.3 The possible locks in multi-user mode Locking files by programming

Several methods can be used to lock a file or a record:

- locking the file,
- locking record by record.

Lock the file

The entire file must be locked if the two following conditions are fulfilled:

- if several records must be locked at the same time,
- to simplify the programming.

The following functions are used to lock a file:

- *HLockFile*: the file is locked in write mode.
- *HLockFile(hLockRead/hLockWrite)*: the file is locked in read/write.
- *HNoModif*: the file is locked in write mode including for the program that locked it.

The following functions are used to unlock a file:

- *HUnlockFile*: the file was locked by *HLockFile*.
- *HEndNoModif*: the file was locked by *HNoModif*.
- *HLock*: the file is closed and unlocked.

Locking record by record

The records can be locked while they are used without the file being entirely locked.

The following functions are used to lock a record: *HRead*, *HReadSeekFirst*, *HReadSeekLast*, *HReadLast*, *HReadFirst*, *HReadNext*, *HReadPrevious*

After each function, you must check *HErrorLock* to make sure the record was locked.

Caution: at the end of the process, don't forget to unlock the records locked by: *HUnlockFile*, *HClose*, *HEndNoModif*, *HUnlockNumRec*, *HAdd*, *HModify*, *HWrite*, *HCross*, *HDelete*, *TableDelete*, *TableSave*.

Note: Locking a record by using the "HRead..." functions has no effect if the file was previously locked

by the "HLock..." functions, the function is equivalent to "HRead..."

4.4 Assisted management of HFSQL errors

The HFSQL engine enables you to manage different types of errors by programming:

- Duplicate error
- Integrity error
- Password error
- Error of modification conflict and error of modification status conflict
- Lock error
- Error of mandatory input
- Reindexing in progress

At any time, *HOnError* allows you to customize the assisted management of errors or to disable this management. For each type of error, you can:

- manage this error via a specific window or via a procedure,
- Disable the assisted management and manage the error by programming (like in WinDev 5.5).

4.4.1 Principle

To simplify data file management programming, the most common types of errors are automatically managed by the HFSQL engine.

This automatic management helps the user process the error. In most cases, the error is caused by a problem involving the data entered.

This automatic management is proposed by default and it can be customized or disabled.

Note: The assisted management of errors is available for the files in HFSQL Classic format, for the files handled via a native access or for the files handles via an OLE DB provider.

If your files are used via an OLE DB provider, only the following will be managed:

• The duplicate errors

Caution: Some duplicates error may not be recognized as such by the OLE DB access. These errors will not be managed as duplicates errors but they will be considered as being fatal errors.

Example: the WinDev analysis and the description of the external database are not synchronized and the description of the file in the analysis

does not contain all the unique keys defined in the database.

• The errors of mandatory input

Caution: The error of mandatory input occurs only if the OLE DB provider indicates that the item is associated with the "NULL forbidden" property. Otherwise, the error will be processed like a fatal error.

If your files are handled via a native access, only the management of mandatory password is not available.

4.4.2 Standard operating mode

Duplicate error

• Cause of the error

The user adds a record for which the value of a unique key already exists.

For example, the name of the city is defined as unique key of Customer file in the analysis. Adding a city with a name taht is already used triggers a duplicate error.

• Default assisted management (WinDev and WebDev)

If a duplicate error is detected, a window or a page is displayed, asking the user to modify the value of the record that caused the error.

• Processing the error (WinDev Mobile)

Check *HErrorDuplicates* after each call to a HFSQL function that may trigger a duplicate error (*HAdd* or *HModify* for example). The error details are returned by *HErrorInfo*.

Integrity error

• Cause of the error

The user tries to add a record without respecting the integrity constraints defined between the files in the analysis.

For example, the files named CEDEX, STATE and REGION are linked between themselves.

These links comply with the relational integrity constraints. An integrity error occurs when adding a state without creating the corresponding region.

• **Default assisted management (WinDev and WebDev)**

When an integrity error occurs, a window or a page is displayed, allowing the user to cancel the operation or to modify the data entered.

• **Processing the error (WinDev Mobile)**

Check **HErrorIntegrity** after each call to a HFSQL function that may trigger an integrity error (**HAdd** or **HModify** for example). The error details are returned by **HErrorInfo**.

Password error

• **Cause of the error**

The program tries to handle a password-protected file (opening, first read operation, ...). The password was not specified by programming: a password error occurs.

• **Default assisted management (WinDev and WebDev)**

When a password error occurs, a window or a page is displayed, allowing the user to enter the file password.

• **Processing the error (WinDev Mobile)**

Check **HErrorPassword** after each call to a HFSQL function that may trigger a password error (**HOpen** or the first FSQL function that handles the file for example). The error details are returned by **HErrorInfo**.

Error of modification conflict and error of modification status conflict

• **Cause of the error**

When using an application or a site in network, conflicts may occur due to conflicting data entered by different users.

For example:

1. The user X edits the form for the state of "California".
2. The user Y edits the form for the state of "California".
3. The user X renames the state to "California_01".
4. The user Y modifies the name of the state ("California_02") and saves. A modification conflict occurs during this backup.

A modification conflict occurs.

A modification status conflict occurs when the form is deleted by the user X for example.

The different cases are presented in the table below:

	Record read		
Record on disk	Enabled	Crossed	Deleted
Enabled	Modification conflict	Modification conflict	Modification conflict
Crossed	Conflict of modification status	Modification conflict	Modification conflict
Deleted	Conflict of modification status	Conflict of modification status	XXXX

• **Default assisted management (WinDev and WebDev)**

If a conflict occurs when modifying a record, a window or a page is displayed, proposing the different possible values for the record:

- the value read in the file (before modification)
- the value modified by another user,
- the value entered by the current user.

The user can choose the value of the item that will be saved.

If a status conflict occurs when a record is modified, a window is displayed, allowing the user to:

- re-enable the deleted data,
- keep the data in its current status.

The user can choose the value of the item that will be saved.

• **Processing the error (WinDev Mobile)**

For the errors of modification conflict, check **HErrorModification** after each HFSQL function that may trigger this error (**HModify** for example). The error details are returned by **HErrorInfo**.

For the errors of modification status conflict, check **HErrorStatusModification** after each HFSQL function that may trigger this error (**HModify** for example). The error details are returned by **HErrorInfo**.

Lock error

- **Cause of the error**

In a network application or in a site, you have the ability to lock a record or a file (to perform specific operations for example). A lock error occurs when a computer tries to access a locked record.

- **Default assisted management (WinDev or WebDev)**

When a lock error occurs, the management of locks (checking whether the file is locked, processing the lock) is automatically performed by the HFSQL engine.

A window or a page is displayed, indicating that the file or record is locked and allowing the user to retry or cancel the operation. In case of cancellation, the application continues to run properly.

- **Processing the error (WinDev Mobile)**

Check **HErrorLock** after each call to a HFSQL function that may trigger a lock error.

Mandatory input (database accessed via OLE DB)

- **Cause of the error**

Some file items necessarily expect a value. An error of mandatory input occurs if this value was not specified.

- **Default assisted management (WinDev and WebDev)**

If a value is not specified, the management of mandatory input is automatically performed by the HFSQL engine.

A window or a page is displayed, indicating that the item must be filled and allowing the user to enter the value of the item.

- **Processing the error (WinDev Mobile)**

The error number associated with this problem is **70710**: Mandatory input.

These error numbers are returned by **HError**.

Reindexing in progress

- **Cause of the error**

The file used is currently reindexed. The file cannot be used (read, write) during this reindex operation.

- **Default assisted management (WinDev or WebDev)**

A window or a page is displayed, indicating that the file is currently re-indexed. This window or page remains displayed during the entire reindex operation and it cannot be closed. A progress bar indicates the status of the reindex operation.

- **Processing the error (WinDev Mobile)**

The error number associated with this problem is **70720**: reindex in progress.

These error numbers are returned by **HError**.

4.4.3 Customization

The different solutions

WinDev, WebDev and WinDev Mobile enable you to customize the HFSQL error management.

You have the ability to use **custom procedures**

A specific procedure of your application is automatically run as soon as a HFSQL error occurs.

Note: A procedure can be defined for each file and for each type of error.

WinDev and WebDev enable you to use **custom windows or pages**: the standard windows/pages of the HFSQL engine are replaced by windows/pages from your application.

Note: a window/page can be defined for each file and for each type of error.

Implementation

To implement the assisted custom management of HFSQL errors, you must:

1. For each type of error, create the procedure, the window or the page used to customize the error. The same window, page or procedure can manage several types of errors. This window, page or procedure must return a specific constant depending on the process to perform (see the paragraph below).
2. Define the customization of errors with **HOnError**.

Examples of windows or pages that can easily be included (WinDev and WebDev)

The error windows/pages used by default and presented in the previous paragraphs are supplied (as well as their WLanguage code) as examples.

These windows are available in the "Programs\Data\Preset windows\HFSQL - Automatic help window" of the WinDev installation directory.

These pages are available in the "Programs\Data\Preset pages\HFSQL - Automatic help pages" of the WebDev installation directory.

These windows/pages can be included in your projects, customized and passed in parameter to **HOnError**.

Constants used to customize the errors

The procedure, the window or the page used to customize the error must return one of the constants found in the table below.

The corresponding process will be run according to the constant returned.

opRetry	The function that triggered the error is re-run.
OpCancel	The function that triggered the error returns an error and the code continues to run.
OpEndProcess	The function that triggered the error returns an error and the current process stops. Equivalent to the following code line: IF NOT <HFSQL function> THEN ReturnToCapture()
OpEndProgram	The function that triggered the error returns an error and the program stops. Equivalent to the following code line: IF NOT <HFSQL function> THEN EndProgram()
opRelaunchProgram	Ends the application and automatically restarts the application

Note: Customizing the modification errors:

If the window, page or procedure called during a modification conflict returns the **opRetry** constant without doing anything else, the values will be written into the file without triggering a new modification conflict.

4.4.4 Disabling the assisted management (WinDev and WebDev)

WinDev and WebDev allow you to entirely disable the system for automatic management of errors. In this case, the different error cases that may occur must be checked in the application. A WLanguage error occurs and the application stops if this test is not run.

Implementation

To entirely disable the automatic management of errors, all you have to do is use **HOnError** and specify an empty string in window or procedure name.

In this case, a test must be run after each HFSQL function that may trigger an error, with one or more of the following functions:

HErrorLock	Checks whether a lock error occurred.
HErrorDuplicates	Checks whether a duplicate error occurred.
HErrorIntegrity	Checks whether an integrity error occurred.
HErrorPassword	Checks whether an error caused by a wrong password occurred.

The error details are returned by **HErrorInfo**.

The assisted management can be re-enabled by **HOnError**.

Note: For the other errors (modification, mandatory input, reindexing in progress), no test is required. The error numbers associated with these problems are:

- 70700: Modification conflict
- 70710: Mandatory item value
- 70720: Reindex operation in progress on the file

These error numbers are returned by **HError**.

5. THE HFSQL VIEWS

5.1 Overview of HFSQL views

A "view" is a set of records selected according to a selection condition. These records come from a HFSQL file and they are loaded in memory.

A view is a "snapshot" taken at a given time of a section of the database.

A HFSQL view can be compared with a "virtual" HFSQL file stored in memory. This file is not physically stored on disk but it can be used almost like a "real" HFSQL file.

A view can contain records from a HFSQL file selected according to selection criteria.

The records found in a view are "virtual" records. A

virtual record includes some file items or all the file items.

Creating a view is a file read operation.

It can come with a lock operation of the records read if these records must be modified.

You have the ability to cancel all the modifications performed on a view. The modification performed on a view are not immediately applied to the HFSQL file. To update a HFSQL file according to a view, you must explicitly give the order via a WLanguage function.

5.2 Benefits of views

The views bring:

- **More flexibility and power to all the applications or sites that use HFSQL files**

The ability to filter and sort is improved for the views.

A view allows you to perform a file browse with a filter according to any item (and not just the filter key).

- **An additional security level**

The views are used to limit the access to a selected subset of data.

- **A gain of speed**

A view is the result of a query. Therefore, it is a client/server exchange that is well suited for the networks on which it is the number of exchanges (and not the size of exchanges) that slows down the communication.

5.3 Handling views

Once created, the views are handled like the HFSQL files. The view has a name that can be used in most HFSQL functions. You have the ability to:

- browse a view (*HReadFirst*, *HReadNext*, etc.),
- add records into a view (*HAdd*),
- implement filters on the view records (*HFilter*),
- modify records (*HModify*),
- ...

The following functions are used to manage the views:

HCreateView	Creates a view on a HFSQL file
HCreateView_5	Creates a view in WinDev 5.5/ WebDev 1.5 format (compatibility only)

HDeleteView	Destroys a view that was created beforehand
HExecuteView	Runs a view that was created beforehand. Used to update the data of the view with the last modified data in the corresponding HFSQL file
HMergeView	Creates a HFSQL view from two views created beforehand (<i>HCreateView</i>). Several operations can be performed when merging the views.
HSortView	Sorts a view by creating an index on a view item

HViewToFile Saves the modifications performed in a view (by *HModify*, *HDelete* or *HCross*) in the corresponding HFSQL file

FileToMemoryTable Displays the content of a view in a memory table

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev or WinDev Mobile).

5.4 Creating HFSQL views

As mentioned before, a view contains "virtual" records coming from a HFSQL file.

To create a view, several steps are required:

1. Choosing the items included in the view.
2. Choosing the sort item of the view (if necessary).
3. Selecting the records contained in the view.
4. Create the view.
5. Performing union operations if necessary.
6. Displaying the view records if necessary.

The following paragraphs present each one of these steps.

Note: You have the ability to keep the operating mode of views found in WinDev 5.5/WebDev 1.5 (with management of view identifier). To do so, use *HCreateView_55*.

5.4.1 Choosing the items of a view

To create a view, the first step consists in defining the items that must be found in the view. Each view is attached to a HFSQL file: the **view file**.

The view can contain:

- all the items of the file,
- some items of the file.

All the items of the file

In order for the view to contain all the file items, use the following syntax:

```
<Result> = HCreateView(<View Name> <File Name>)
```

- <View Name> is the name of the created view.
- <File Name> is the logical name of the base file of the view.

Example:

```
Extern MyView1
MyFile is string="CUSTOMER"
// View with all the
// records of all the
// file items
HCreateView(MyView1, FileName)
```

Some items of the file

To select some items from the file, use the following syntax:

```
<Result> = HCreateView(<View Name>,...
                    <File Name>, <Item Name>)
```

- <View Name> is the name of the created view.
- <File Name> is the logical name of the base file of the view.
- <Item Name> is the list of items that must be included in the view. The items are separated by a comma in this character string.

Example:

```
Extern MyView1
MyFile is string="CUSTOMER"
ItemView is string= ...
                "LASTNAME, FIRSTNAME, ADDRESS"
// View containing the records for
// the LastName, FirstName and Address
// items
HCreateView(MyView1, FileName, ...
            ItemView)
```

5.4.2 Choosing the initial sort item of the view

When creating a view, you have the ability to specify the view item that will be used for the sort. All the view items can be used for the sort.

Once created, the view can be re-sorted at any time by *HSortView*.

To specify the initial sort, use the following syntax:

```
<Result> = HCreateView(<View Name>, ...
                    <File name>,...
                    <Item Name>, <Sort Item>)
```

- <View Name> is the name of the created view.
- <File Name> is the logical name of the base file of the view.
- <Item Name> is the list of items that must be included in the view.

- <Sort Item> is the name of the sort item. This string contains:
 - The sort direction, with "+" (ascending, default value) or "-" (descending).
 - The name of the sort item or its creation subscript.

Example:

```
Extern MyView1
FileName is string="CUSTOMER"
ViewItems is string
SortItem is view

ItemsView= "LASTNAME,...
    FIRSTNAME, ADDRESS, ZIPCODE"
SortItem = "-LASTNAME"
//OR SortItm="-1"

// View containing the records
// for the LastName, FirstName,
// Address and Zip Code items
// Initially sorted by last name
// reverse order
HCreateView(MyView1, FileName,...
    ViewItems,SortItem)
```

5.4.3 Selecting records from the view

When creating a view, you have the ability to specify a selection condition for the records. The condition is applied to each record before it is included in the view.

To specify a selection condition in a view, use the following syntax:

```
<Result> = HCreateView(<View Name>, ...
    <File Name>, <Item Name>, ...
    <SortItem>,<Condition>)
```

- <ViewName> is the name of the created view.
- <File Name> is the logical name of the base file of the view.
- <Item Name> is the list of items that must be included in the view.
- <Sort Item> is the name of the sort item.
- <Condition> is the selection condition. This condition is found in a character string that uses a syntax in the following format:
"CustomerName > 'Smith' and

(ZipCode=94102 or ZipCode=94103)"

5.4.4 Union operations between several views

When creating views, several selection operations can be performed on the records.

However, these operations may be insufficient in some cases. A view can be created from existing views.

To do so, you have the ability to use the standard union operations: union, exclusive union, intersection, subtraction.

The following syntax must be used:

```
<<Result> =HMergeView(<View Name>,...
    <View Name1>,<View Name2>,...
    Operation, SortAndComparisonItem)
```

- <ViewName> is the name of the created view.
- <View Name1> is the name of the first view.
- <View Name2> is the name of the second view.
- <Operation> is one of the constants used to define the type of operation to perform:
 - **hViewUnion**: union of all the rows found in view A and view B
 - **hViewUnionEx**: union of all the non-common rows found in view A and view B
 - **hViewIntersection**: rows common to A and B
 - **hViewJoin**: rows common to A and B
 - **hViewSubtraction**: rows of A - common rows found in B
- <SortAndComparisonItem> is a string containing the subscript of the comparison item between the two views. This item is also the initial sort item of the created view.

Important: The file of the view generated by an union operation on two other views is the file of the first view passed in parameter to **HCreateView**.

Note: The union operations allow you to perform processes that are faster than some selection conditions. For example, we advise you to use the union instead of OR in the condition of **HCreateView**.

6. HFSQL TRIGGERS

6.1 Overview

6.1.1 Definition

A trigger corresponds to the association between a procedure and a write function on a HFSQL file. The triggers are used to easily run processes when modifying a record.

The procedures called via triggers can be run *before* or *after* the write operation performed on the file.

A **BEFORE** trigger is called *before* running a HFSQL function. It can be used to check the consistency of the record items for example. In this trigger, a HFSQL state variable can be initialized to cancel the execution of the function.

An **AFTER** trigger is called *after* the execution of the HFSQL function (except if the program was interrupted during this function). It can be used to centralize the process of errors for example.

Note: Trigger on the functions for handling tables: The functions for handling tables (*TableAdd*, *TableAddLine*, *TableDelete*, *TableModify*, *TableSave*, ...) implicitly uses the following HFSQL functions: *HAdd*, *HDelete* and *HModify*.

When using one of these functions for handling tables, if a trigger is defined for the corresponding HFSQL function, it is automatically activated.

6.1.2 Benefits of triggers

The triggers allow you to associate processes with all the HFSQL functions for file management without having to worry about the location of these functions in the source code.

The triggers associated with the HFSQL functions used will be run even if it is a window, a page or a code created by a wizard.

The use of triggers allows you to reduce the size of the code, to simplify its reading and to facilitate the future evolutions, by grouping processes.

For more information, the diagram found on the following page presents the detailed operating mode of triggers.

Note: The triggers are also available in HFSQL Client/Server.

6.2 How to create and handle triggers?

6.2.1 Functions for handling the triggers

The following functions are used to manage the triggers:

HActivateTrigger	Re-enables a trigger that was disabled by <i>HDisableTrigger</i>
HDescribeTrigger	Adds or modifies a trigger on a HFSQL file
HDeactivateTrigger	Disables a trigger. This trigger can be re-enabled by <i>HActivateTrigger</i>
HDeleteTrigger	Destroys a trigger. This trigger cannot be used anymore

HSetTrigger Enables or disables the management of triggers

HListTrigger Returns the list of triggers applied to one or more data files

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev or WinDev Mobile).

The functions for handling the triggers on the server are identical and they include "Server" in their name.

6.2.2 Handling triggers

Creating a trigger

A "before" trigger or an "after" trigger is created by *HDescribeTrigger*, in the initialization code of the project.

The trigger can be created on one or more analysis files and for one or more HFSQL functions that handle records.

A specific procedure is run whenever the trigger is run.

To create a trigger, use the following syntax:

```
<Result> = HDescribeTrigger
    (<FileName>, <NameHFFFunction>, ...
     <ProcedureName>, <Type>)
```

- <FileName> corresponds to the logical name of the files affected by a process managed by trigger.
- <NameHFFFunction> corresponds to the name of the WLanguage functions that release the trigger.
- <ProcedureName> corresponds to the associated procedure called by the trigger.
- <Type> is used to define the type of trigger ("before" or "after").

Creating a procedure called by a trigger

A "trigger" procedure accepts no parameter.

However, some HFSQL state variables are positioned before each call:

- **h.filename:** string containing the logical name of the file whose trigger is enabled.

- **h.action:** character initialized to "A" for a "Before", trigger and to "P" for an "after" trigger.
- **h.TriggerFunction:** string containing the name of the HFSQL function that released the trigger, only if the integrity management is disabled.
- **h.ToDo:** During the execution of a before trigger:
 - canceling the execution of the current Hyper File function by assigning "A" to the Hyper File state variable: H.ToDo = "A". In this case, the action is not performed and the function (HAdd, HModify, ...) returns True (no error).
 - canceling the execution of the current Hyper File function by assigning "E" to the Hyper File state variable: H.ToDo = "E". In this case, the action (HAdd, HModify, ...) is not performed and it returns False. The error message is as follows: "The action on XXX file was interrupted by the trigger".

Enable/Disable a trigger

You have the ability to start a process used to enable or disable a check performed by a trigger.

To temporarily disable a specific trigger, use the following syntax:

```
<Result>=HDeactivateTrigger([<ListHFFFiles>
[, <ListHFFFunction>]] [, <Type>])
```

Destroy a trigger

HDeleteTrigger is used to delete one or more triggers via the following syntax:

```
<Result>=HDeleteTrigger([<ListHFFFiles> [,
<ListHFFFunctions>]] [, <Type>])
```

7. THE DATA REPLICATION

7.1 Overview of the replication

7.1.1 Overview

The replication of data is a powerful feature. The replication is the operation allowing the maintenance of remote databases with identical structures. Each one of these databases evolves independently. Via the replication, the operations performed on each one of the databases are applied to all the other databases.

WinDev or WebDev allows you to easily perform these operations. WinDev or WebDev proposes two types of replication:

- **The logged replication** (based on the log process). This type of replication is used to replicate the HFSQL databases between themselves. This type of replication can be implemented by the WLanguage functions or by WDReplic.
- **The universal replication** that is used to replicate any type of database (a HFSQL database with an Oracle database for example). The universal replication can be implemented via the assisted universal replication.

We shall only present the universal replication.

7.1.2 Vocabulary specific to the replication Databases

The replication distinguishes between two types of databases:

- **master database**
It is the reference database. All the updates are performed on this database:
 - modifications performed by the application run on this computer.
 - modifications performed on the remote computers and transmitted by the replication
- **the replica database or subscriber database**
This remote database is identical to the "Master" database. The modifications performed by the remote computer are applied to this database. The replication transmits these modifications to the "master" database.

Type of replication

Two types of replication can be implemented:

- **Unidirectional replication:**
This type of replication consists in performing an update from the "Master" database to the "Subscriber" databases or from a "Subscriber" database to the "Master" database.
- **Bidirectional replication:**
This type of replication consists in performing an update from the "Master" database to the "Subscriber" databases and from the "Subscriber" databases to the "Master" database.

Files created during a replication

The universal replication uses several types of files:

- **.RPM file:** file used to describe a master database as well as the databases that subscribe to it.
- **.RPL file:** file used to describe a subscriber database. A RPL file is created for each subscriber database. This file is found on the subscriber computer.
- **.RPA file:** log file containing the replication information. This file is exchanged between the master database and the subscriber database.
- **.SYN file:** file containing information about the situation on the remote database. This file is used to optimize the size of the synchronization files. This file is found on the master computer and on each subscriber computer.

7.1.3 Note

To implement the universal replication on databases other than HFSQL ones, a **DateTime** item must be created in each file. This item will be updated by the application when modifying or adding a record. If the databases use different time zones, we recommend that you use a universal format (GMT date and time for example).

7.2 Implementing the universal replication

7.2.1 Activation

To enable the universal replication, all you have to do is use *HSetReplication* associated with the *rplReplicationUniversal* constant.

This function is used to disable the standard replication mode (if it was enabled) and to enable the universal replication.

7.2.2 Declaring the master database

This operation must be performed once only on the master database.

To declare the master database, all you have to do is use *HCreateMasterReplica*.

Note: If data can be stored in the current directory, the following code line can be used:

```
HCreateMasterReplica( " " )
```

This code line creates the MasterReplica.RPM file on disk. Then, all you have to do is write the subscribers into this file.

7.2.3 Declaring the subscriber databases

This operation must be performed once only for each subscriber database. This operation must be performed on the master database.

To declare a new subscriber, use *HCreateSubscriberReplica*. This function creates a subscriber (RPL file) with the specified name. This functions also returns a subscriber number.

Note: *HCreateSubscriberReplica* uses specific parameters for the universal replication. See the help about the function for more details.

7.3 Replication between heterogeneous databases

HCreateMoveableReplica creates a log file (.RPA file).

This file is transmitted and run by *HSynchronizeReplica*.

Caution: By default, the master has priority when the synchronisation is performed (*HSynchronizeReplica*): if a replication is performed from the subscriber to the master, the data found in the master database will not be updated. We recommend that you use another constant (*rplMostRecentFirst* for example).

Two specific functions can also be used:

HRplDeclareLink	Used to signal to the replication engine that a link was found between two files. The engine will follow the link to get the list of records that must be replicated in the second file.
------------------------	--

HRplManageFile	Defines the options used for the universal replication of a file.
-----------------------	---

HRplManageItem	Defines the options used for the universal replication of an item.
-----------------------	--

HRplFilterProcedure

Used to specify a specific filter procedure when a given file is replicated.

Notes:

- The exchanges of ".RPA" files are not necessarily symmetrical: it is possible to create and run several logs in one direction without creating or running a single log in the other direction. However, to improve the performances, you should avoid performing exchanges always in the same direction.
- The replication respects the filters applied to the tables or to the files (except for links, to respect the integrity).
- A filter can be implemented on the master side: the subscribers will receive a subset of the data from the master database. In this case, there is no need to implement a filter on the database. If, due to the filter, a record is no longer "visible" on the master database, this record will be considered as being deleted. It will be automatically deleted from the subscriber computer.

7.4 Limitations

The limitations are as follows:

- Each file or table must have at least one unique key.
- You must have the ability to associate a date with a record. If the item used to associate a date with the record is automatically filled by trigger, this trigger must be disabled when running *HSynchronizeReplica* (to avoid filling this item with the replication date).
- The replication does not open the files or the tables. The files or the tables must have been opened by the program before starting the replication.
- During the replication, the memory consumption may be very high. Therefore, only the necessary records should be replicated toward a subscriber computer.
- **Caution: special case:** only part of the linked tables are not replicated. In this case, the replication may redo some modifications as an {addition, deletion} couple. If a non-replicated table is linked with cascading modifications, the linked records may be wrongly deleted.

8. MANAGING FILES IN "BACK OFFICE"

8.1 Overview

Most of the interactive site available on Internet are intended to be run in parallel of traditional applications, usually run on Windows.

For an e-commerce site for example:

- the "Front Office" will be found on the Internet side and it will consist in presenting the products and taking orders.
- the "Back Office" will consist in managing the orders placed: delivery forms, invoices, return

forms, statistics, past due payments, ...

To build the features of the "Back Office", the use of the WinDev IDE is recommended.

WebDev proposes several possibilities to manage the "Back Office":

- the update by email,
- the remote access to HFSQL,
- the replication.

8.2 The update by email

The operation consists in synchronizing two databases, the database of the Back Office application and the database of the Front Office application, via a set of emails and a procedure for reading the emails.

The "Front Office" application sends an email whenever the database is modified.

The "Back Office" application automatically reads the messaging at regular time intervals in order to update the main database.

8.3 The remote access to HFSQL

8.3.1 Definition

The remote access allows you to consult a HFSQL database via Internet/Intranet or via STN (Switched Telephone Network).

Three use modes are available:

- **Mode 1:**
A "Back Office" application interrogates the database found on the Web server (where the WebDev site is located).
- **Mode 2:**
The Web server accesses a database found on the computer of the "Back Office" application.
- **Mode 3:**
A WebDev site found on a server A queries a database found on a server B (where another WebDev site is located).

The WLanguage functions used to manage remote access are: *HOpenAnalysis* and *HConnectRemoteAccess*.

8.3.2 Details of the three use modes

Mode 1: WinDev Back Office application and database on the Web server

The database is found on the Web server. The database is updated by the WebDev site according to the actions performed by the Web users.

The users of the "Back Office" application interrogate and update the database from the WinDev application.

The "Back Office" application establishes the dialog with the database via remote access.

Note: The WinDev "Back Office" application can be accessible on a shared server (see the diagram) or distributed on the computers of the different users. In this case, each user starts a remote access.

Mode 2: Database on the computer of the WinDev Back Office application

The database is found on the computer of the WinDev application. The database is updated by the WebDev site during the actions performed by the Web users via remote access.

The users of the "Back Office" application interrogate and update the database from the WinDev application.

Mode 3: Two WebDev sites on two remote servers for a single database

Depending on the site to which the users connect, the database is updated:

- by the WebDev site located on the same computer.
- from the WebDev site via the remote access (see the online help for more details).

8.4 The data replication

8.4.1 Overview

The replication of data is a powerful feature. The replication is the operation allowing the maintenance of remote databases with identical structures. Each one of these databases evolves independently: different operations are performed on these databases.

Via the replication, the operations performed on each one of the databases are applied to all the other databases.

The data replication can be easily applied to a WebDev site. In this case, the replication allows the following to be kept up to date:

- the database found on the Web server, used by the WebDev site.
- the database found at the company's headquarters and used by a WinDev "Back Office" application for example.

Let's look at a standard use example of data replication:

A company proposes two methods for taking orders:

- from the Web site of the company: the customer enters his order by specifying his personal details and the products ordered.
- via sales people, who use an application for taking orders in Windows. This application is used at the company's headquarters, as a "Back Office" application.

In this example:

- the "Orders" file:
 - found on the Web server contains the orders entered by customers who are using the Internet site.
 - found at the company's headquarters contains the orders entered by the sales people.

The database at the headquarters must be updated with the orders placed on the site.

- the "Product" file:
 - found on the Web server must contain all the references of products.
 - found at the company's headquarters can evolve, taking the new products into account.

Therefore, the database of the WebDev site must be updated with the new products.

Via the replication, the WebDev site found on the Web server and the WinDev application found at the company's headquarters both have their own database.

The database of the WebDev site (or **Replica**) is identical to the "master" database, during its initialization.

Regularly, a "Synchronization" of these databases is used to take into account the modifications performed on each one of the different databases.

8.4.2 Implementing the replication

See "The data replication", page 382 and the online help for more details.

9. ACCESSING EXTERNAL DATABASES

9.1 Overview

There are several execution modes of SQL queries according to the type of access to the database.

- **Access to a HFSQL database** (free distribution with your WinDev applications and your WebDev sites):
No setup constraint.
The SQL and HFSQL functions (HReadxxx, ...) can be used with this type of access.
- **Access via a native access: Native access to Oracle, SQL Server, AS/400, ...**
An additional module is required for each type of native access. Contact the sales department regarding the availability for your database.
For the Oracle or SQL Server accesses, a client layer must be installed on the user computer.
The SQL and HFSQL functions (HReadxxx, ...) can be used with this type of access. This type of access is faster than the accesses by ODBC or via an OLE DB provider.
- **Access via a direct ODBC driver:**
The characteristics of the connection to the database must be defined in the ODBC administrator of Windows. Only the SQL functions can be used

for this type of access. The HFSQL functions (HReadxxx, ...) cannot be used.

- **ODBC access via the OLE DB provider:**
This type of access uses a specific OLE DB provider. This type of access is not recommended because it is slower than an access via an ODBC driver. Indeed, the performance is not as good as with a direct ODBC driver because the access is performed both via the ODBC driver and the OLE DB provider.
The HFSQL (HReadxxx, ...) and SQL functions can be used with this type of access.
The characteristics of the connection to the database must be defined in the ODBC administrator of Windows. The provider as well as MDAC 2.6 (or later) must be installed on the computer.
- **Access via an OLE DB provider:**
This type of access uses an OLE DB provider. The provider as well as MDAC 2.6 (or later) must be installed on the computer.
The SQL and HFSQL functions (HReadxxx, ...) can be used with this type of access.

9.2 Specific features

- **xBase access possible via HFSQL:**
Specific xBase functions can be used in addition to the standard HFSQL functions (*HDBOpen*, ...).
- **ASCII files:**
Use the WLanguage functions specific to the external files (*fOpen*, *fRead*, *fWrite*, ...).
- **INI files:**
Use the WLanguage functions specific to the INI files (*IniRead*, *IniWrite*, ...).
- **Management of the registry:**
Use the WLanguage functions specific to the registry (*RegistryQueryValue*, *RegistrySetValue*, ...).

9.3 Functions for managing the external databases

See the online help for more details about the functions used to manage an external database (con-

necting to the database, running a query, locking records, ...).

10. HFSQL FUNCTIONS

The following functions are used to manage the HFSQL files:

FileToPage	Automatically initializes the controls of a page with the values of the associated items found in the current record (loaded in memory) of the HFSQL file
FileToScreen	Automatically initializes the controls of a window with the values of the associated items found in the current record (loaded in memory) of the HFSQL file
HAccelerateSpeed	Reorganizes the internal structure of the indexes to optimize the speed for accessing the data
HActivateAutoFilter	Enables an automatic filter on the linked files when browsing an XML file
HActivateFilter	Enables the filter that was previously created for the specified file
HActivateTrigger	Re-enables a trigger that was disabled by HDisableTrigger
HAdd	Adds the record found in memory into the data file
HAlias	Creates a logical alias of a file or cancels all the existing aliases
HBackward	Moves backward several records from the current position in the file, according to a specified item
HBuildKeyValue	Builds the value of a composite key to create a filter or to perform a search
HBuildKeyValueANSI	Used to (on a Unicode platform, Pocket PC for example) build the value of a composite key in order to store this composite key in a HFSQL data file
HCancelAlias	Cancels an alias that was declared by HAlias
HCancelDeclaration	Deletes a declaration that was performed by HDeclare , HDeclareExternal or HDescribeFile
HCancelSeek	Cancels the current search criterion
HChangeConnection	Changes the connection to a database used for a file. This change will be taken into account during the next file opening
HChangeDir	Modifies the access path to a data file
HChangeKey	Changes the search key
HChangeLocation	Modifies the search mode of data files
HChangeLogDir	Modifies the access path to a log file (JournalOpération file and JournalIdentification file)
HChangeName	Modifies the physical name of a data file
HChangeRplDir	Modifies the location of the description of the subscriber replica (RPL file). This function must be used on the subscriber computer
HCheckIndex	Checks whether the data found in the index file (.NDX file) properly references the data found in the data file (.FIC)
HCheckStructure	Defines the mode for comparing files.
HClose	Closes a file or all the opened files
HCloseAnalysis	Closes the current analysis
HCloseConnection	Closes a connection to a database
HConnect	Redefines one or more parameters of a connection by native access or by OLE DB access on a specific table or on a set of tables
HConnectRemoteAccess	Opens an analysis in HFSQL format via a remote access
HConvert	Converts a numeric value into a binary string in order to perform a search on a numeric key
HCopyRecord	Copies the content of the current record (loaded in memory) into the current record of another file
HCreateMasterReplica	Creates the description file of a master replica (MasterReplica.RPL file)

HCreateMoveableReplica	Creates a file that can be used to replicate the data from the current database toward a remote database (for sending via email or CD for example)
HCreateSubscriberReplica	Creates the description file of a subscriber replica (ReplicaAbonne.RPL file)
HcreateView	Creates a HFSQL view
HCreation	Creates an empty data file (".FIC" extension) with the index file and the memo file if necessary
HCreationIfNotFound	Creates an empty data file (if the file does not exist) or opens a file (if the file exists)
HCross	Crosses a file record
HDeactivateAutoFilter	Disables an automatic filter on the linked files when browsing an XML file
HDeactivateFilter	Temporarily disables the filter on a file (view or query)
HDeactivateTrigger	Disables a trigger
HDeclare	Declares a file description (found in another analysis) in the current project
HDeclareExternal	Temporarily imports into the current analysis the description of a file from an existing HFSQL file
HDelete	Deletes a record from a file
HDeleteAll	Deletes all the records from a file or from a query.
HDeleteSet	Deletes a set of stored procedures from a HFSQL server.
HDeleteTrigger	Destroys a trigger
HDeleteView	Destroys a view that was created beforehand
HDescribeConnection	Describes a temporary connection
HDescribeFile	Describes a temporary data file
HDescribeFullTextIndex	Describes a full-text index in a data file created by programming.
HDescribeItem	Describes an item found in a temporary data file
HDescribeLink	Describes a temporary link between two files
HDescribeTrigger	Not available in this version
HDuplicateRecord	Duplicates the record read in a data file: the record found in memory is added into the data file (query or view).
HEndNoModif	Unlocks a file that was locked by HNoModif
HError	Returns the number of the last error triggered by the HFSQL engine
HErrorDuplicates	Used to find out whether a duplicate error occurred
HErrorInfo	Returns a detailed information about the last error triggered by the HFSQL engine
HErrorIntegrity	Used to find out whether an integrity error occurred
HErrorLock	Used to find out whether a lock error occurred
HErrorModification	Returns the value of an item during a modification conflict
HErrorPassword	Used to find out whether a password error occurred on this file
HErrorStatusModification	Returns the status of a record during a modification conflict
HExecuteProcedure	Runs a stored procedure or function.
HExecuteQuery	Initializes a query created in the query editor and declares this query to the HFSQL engine
HExecuteSQLQuery	Initializes a query written in SQL language and declares this query to the HFSQL engine
HExecuteView	Runs a view that was created beforehand
HExportXML	Exports the records from a file (HFSQL or OLEDB), view or query into an XML file
HExtractMemo	Extracts the content of a binary memo item into a file
HFileExist	Used to find out whether a file exists
HFilter	Defines and enables a filter on a file, a view or a query
HFilterContains	Defines and enables a "Contains" filter on a data file, a view or a query.

HFilterIdentical	Defines and enables a filter used to search for the exact value of a string item. This filter can be used on a file, a view or a query
HFilterIncluded-Between	Defines and enables an "Included between" filter on a file, view or query. The "Included between" filter is used to select all the records found between two values
HFilterStartsWith	Defines and enables a "Start with" filter on a file, view or query. The "Starts with" filter is used to select all the records that start with a specific set of characters
HFirst	Positions on the first file record (the record is not read)
HFlush	Forces the operating system of the computer where the data files are found to write the data onto the disk
HForward	Moves several records forward from the current position in the file, according to a specified item
HFound	Checks whether the current record corresponds to the current filter or to the current search
HFree	Transforms the crossed records into deleted records
HFreePosition	Deletes a position that was saved by HSavePosition
HFreeQuery	Frees the resources of a query (once HExecuteQuery or HExecuteSQLQuery has been used)
HGetCurrentPosition	Returns the approximate position of the current record in the file
HHistoryModification	Returns the modifications made to one or more items of a given record. The result can be displayed in a list box or in a table to allow the user to view the modifications performed on the specified file
HImportHF55	Imports a Hyper File 5.5 file into a file in HFSQL Classic format
HImportText	Imports a text file into a file in HFSQL Classic format
HImportXML	Imports an XML file into a file in HFSQL Classic format
HIndex	Rebuilds the index of a file
HIndexingInProgress	Indicates that a data file is currently reindexed and returns the percentage of reindexing performed
HInfoAnalysis	Returns information about an analysis (WDD file).
HInfoLog	Returns information about the server logs.
HInfoMemo	Returns the characteristics of binary memos
HInfoReplica	Returns information about the specified replica
HInitSubscriber	Initializes the range of automatic identifiers for the description file of a subscriber replica (".RPL" extension)
HLast	Positions on the last record of a file
HLinkMemo	Allows you to associate a file with a binary memo item
HListAnalysis	Returns the list of analyses available in a given directory
HListConnection	Returns the list of connections defined in the application or in the site (connections defined in the analysis and/or dynamically)
HListCustomFolder	Returns the list of custom-folders (also called groups) defined in the analysis.
HListFile	Returns the list of files found in the current analysis or in a specific analysis recognized by the HFSQL engine
HListFullTextIndex	Returns the list of full-text indexes found in a file (query or view) recognized by the HFSQL engine
HListItem	Returns the list of items found in a file recognized by the HFSQL engine
HListKey	Returns the list of keys found in a file recognized by the HFSQL engine
HListLink	Returns the list of links (Merise type) found in the current analysis or in a specific analysis
HListProvider	Returns the list of OLE DB providers installed on the computer
HListQueryParameter	Returns the list of parameters for a query created in the query editor
HListREP	Returns the list of assignments for the files found in an analysis
HListTrigger	Returns the list of triggers applied to one or more HFSQL data files

HLockFile	Locks a file and restricts the access to this file for all the other applications or sites
HLockRecNum	Locks a record and restricts the access to this record for all the other applications or sites
HLogInfo	Inserts comments when logging an operation
HLogRecreate	Used to re-create an empty log. This function is used for example to reset the log to 0 after a backup or a replication. The content of the existing files is lost
HLogRestart	Restarts the log process on a file. This log process was stopped by HStopLog
HLogStop	Stops the log process of a file. The operations performed in the logged file are not saved anymore
HMergeView	Creates a HFSQL view from two views created beforehand
HMigrateLinkedCompositeKey	Migrates the values of linked composite keys
HMode	Changes the mode and the method used to lock the files
HModify	Modifies the specified record or the record found in memory in the data file
HModifyStructure	Used to update the structure of a HFSQL data file by performing an automatic data modification (also called data synchronization).
HNbRec	Returns the number of records in a file or a HFSQL view
HNext	Positions on the next file record (the record is not read)
HNoModif	Prevents from modifying a file. The records can be accessed in read-only
HOnError	Customizes the management of errors
HOpen	Opens a file
HOpenAnalysis	Opens an analysis in HFSQL format
HOpenConnection	Establishes a connection to a database
HOptimize	Optimizes the access to the indexes of HFSQL files: the indexes are loaded in the system caches. Speeds up the first file browses and the first query executions
HOptimizeQuery	Used to handle the idle periods of a software (period without processes) to optimize the queries. Optimizes the access to the indexes of HFSQL files
HOut	Used to find out whether the record on which we want to be positioned is located outside the file, filter, view or query
HPass	Defines the password used to create or open a file
HPost	Stores a unique computer number in order to use the log operations and the transactions in network
HPrepareQuery	Initializes a query and declares this query to the database server in order to optimize the next executions of this query
HPrepareSQLQuery	Initializes a query written in SQL language and declares this query to the database server in order to optimize the next executions of this query
HPrevious	Positions on the previous file record (the record is read)
HPriority	Used to find out and modify the priority of the calling application.
HPriorityClient	Modifies the priority of a client application.
HRead	Reads a record in a file according to a given record number
HReadFirst	Positions on the first file record, reads the record and updates the HFSQL variables
HReadLast	Positions on the last file record, reads the record and updates the HFSQL variables
HReadNext	Positions on the next file record, reads the record and updates the HFSQL variables
HReadPrevious	Positions on the previous file record, reads the record and updates the HFSQL variables
HReadSeek	Positions on the first file record whose value for a specific item is greater than or equal to a sought value
HReadSeekFirst	Positions on the first file record whose value for a specific item is greater than or equal to a sought value
HReadSeekLast	Positions on the last file record whose value for a specific item is less than or equal to a sought value

HRecNum	Returns the number of the current record in the file or in the HFSQL view
HRecordDate	Returns the date and time of the last write operation performed on a record found in a HFSQL file
HRecordToXML	Retrieves the structure and the value of the current record and exports them into a character string in XML format
HRefreshSet	Creates or updates a set of procedures on a HFSQL server
HRegenerateFile	Regenerates a file from its log
HReset	Initializes one or more variables of the file items with their default values
HRestorePosition	Restores a file context that was saved beforehand
HRetrieveltem	Returns the content of an item for the current record
HRetrieveLog	Writes into a file the server logs performed between two given dates
HRetrieveRecord	Returns the content of the current record
HRplDeclareLink	Declares a (1,1) (0,n) link between two tables
HRplFilterProcedure	Defines the WLanguage procedure that will be called whenever a replication operation is performed on a specific file
HRplManageFile	Defines the options used for the universal replication of a file: <ul style="list-style-type: none"> • the direction for the replication, • the management mode of conflicts.
HRplManagemItem	Specifies the replication options for an item: the item can be replicated or not.
HRplPass	Defines the passwords used to protect the movable replicas for the universal replication
HSavePosition	Stores the current file context
HSecurity	Enables or disables the security mechanism
HSeek	Positions on the first file record whose value for a specific item is greater than or equal to a sought value
HSeekFirst	Positions on the first file record whose value for a specific item is greater than or equal to a sought value
HSeekLast	Positions on the last file record whose value for a specific item is less than or equal to a sought value (the record is not read)
HSetDuplicates	Enables or disables the management of duplicates on a unique key
HSetIntegrity	Enables or disables the management of an integrity constraint on a file link
HSetLog	Enables or disables the management of the log for a logged file
HSetMemo	Enables or disables the management of memo items
HSetPosition	Positions on a record from the approximate position of one of its items (the record is read)
HSetRemoteAccess	Temporarily disables the remote access in order to access HFSQL Classic data files found locally
HSetREP	Enables or disables the management of .REP file
HSetReplication	Temporarily enables or disables the management of replication
HSetServer	Used to find out and modify some server settings.
HSetTransaction	Enables or disables the management of transactions
HSetTrigger	Enables or disables the management of triggers
HSortView	Sorts a view by creating an index on a view item
HStatCalculate	Performs statistical calculations on the keys of a file
HStatDate	Returns the date of the last update for the index statistics
HState	Used to find out the status of a record
HStatNbDuplicates	Returns the number of duplicates for a given item
HStatNbRec	Returns the number of entries for a given item
HStatNbRecRange	Returns an estimate regarding the number of entries for an item within an interval of values

HStatTime	Returns the time of the last update for the index statistics
HSubstDir	Replaces the logical name of the data directory (specified in the analysis) by a physical name
HSynchronizeReplica	Synchronizes a master replica and a subscriber replica
HToFile	Copies a data source (view, query, ...) toward a physical HFSQL file with the same description
HToItem	Assigns the specified value to an item of the current record
HTransactionCancel	If a transaction is in progress , cancels all the operations performed on the files in transaction since the beginning of transaction. If no transaction is in progress , restores the consistency of the database and cancels the transaction that failed (power outage for example)
HTransactionEnd	Validates the current transaction: <ul style="list-style-type: none"> • the modifications performed on the data file since the beginning of transaction (HTransactionStart) are validated • the transaction file is deleted (if this transaction is the last current transaction in case of a network application) • the records locked in read-only by the transaction are unlocked
HTransactionFree	Transforms all the records "in transaction" into "normal" records if these records do not belong to a current transaction. If a record found in the specified data file is considered as being in transaction but does not belong to a transaction in progress, it is automatically freed
HTransactionInterrupted	Used to find out whether a transaction was interrupted (the transaction was neither validated nor canceled). The characteristics of the interrupted transaction are returned by the HFSQL variables
HTransactionStart	Starts a transaction on the HFSQL files and creates the transaction file
HUnlockFile	Unlocks a file that was locked by HLockFile
HUnlockRecNum	Unlocks a record
HVersion	Used to find out whether the file content was modified
HViewToFile	Saves the modifications made to a view in the corresponding file
HWrite	Writes a record into a data file without updating the indexes corresponding to all the keys used in the file
PageToFile	Automatically initializes the memory value of the file items with the value of the page controls
ScreenToFile	Automatically initializes the memory value of the file items with the value of the window controls
WithSpace	Adds or deletes the spaces found on the right of a text item when reading it

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev or WinDev Mobile).

11. HFSQL PROPERTIES

The following properties are associated with the HFSQL files:

Abbreviation	Returns the abbreviation of the file defined in the data model editor during the file description
AccentuationSensitive	Configures how the accented characters will be taken into account in the different searches performed on the item
Access	Used to find out and modify the type of access for a connection
Binary	Identifies a binary item
BrowsedFile	Used to find out and modify the data source used to display the records in the controls based on a data file
CacheSize	Enables you to find out and modify the maximum number of records that can be contained in the cache of the Native MySQL Access
Caption	Returns the caption associated with the item
CaseSensitive	Configures how the case (uppercase/lowercase characters) will be taken into account in the different searches performed on the item
Charset	Enables you to find out the character set used by a HFSQL file
Component	Configures the components of a composite key
CompositeKey	Identifies a composite key item
Compression	Configures the compression for the data of a connection.
Connection	Enables you to find out the connection currently associated with a data file
CtAccess	Manages the type of access to the file during the OLE DB connection to a specific table (while taking into account the modifications performed by HConnect)
CtCacheSize	Enables you to find out and modify the maximum number of records that can be contained in the cache of the Native MySQL Access
CtCursorOptions	Enables you to find out the type of cursor used during a connection to an external database
CtDatabase	Manages the OLE DB data source during the OLE DB connection to a specific table (while taking into account the modifications performed by HConnect)
CtDescribedAccess	Manages the type of access to the file during the OLE DB connection to a specific table (while ignoring the modifications performed by HConnect)
CtDescribedCaption	Configures the caption of the connection to the OLE DB data source
CtDescribedDatabase	Manages the OLE DB data source during the OLE DB connection to a specific table (while ignoring the modifications performed by HConnect)
CtDescribedOLEDBProvider	Manages the name of the OLE DB server (while ignoring the modifications performed by HConnect)
CtDescribedPassword	Defines the password used to open the file during an OLE DB connection (file defined dynamically only)
CtDescribedUserName	Manages the user name during an OLE DB connection (while ignoring into account the modifications performed by HConnect)
CtExtendedInfo	Enables you to find out the additional information inserted into the string for connecting to the database
CtInitialCatalog	Enables you to identify the default database defined for the access
CtOLEDBProvider	Manages the name of the OLE DB provider (while taking into account the modifications performed by HConnect)
CtUserName	Manages the user name during an OLE DB connection (while taking into account the modifications performed by HConnect)

CursorOptions	Enables you to manage the type of cursor used during a connection (HFSQL Client/Server, OLE DB, ...).
Database	Enables you to find out and modify the database associated with a connection
Date	Retrieves and modifies the date in a Date item ("Date and Time" format)
Day	Retrieves or modifies the day or the number of days in a Date item ("Date", "Date and Time" or "Duration" format)
DecimalPart	Enables you to find out and modify the number of decimal digits found in a numeric item
DefaultValue	Configures the default value of an item
DeletionRule	Manages the deletion rule (integrity management) used in a link
DescribedDirectory	Manages the physical directory of HFSQL files (while ignoring the directory changes performed by <i>HChangeDir</i> or <i>HSubstDir</i>)
DescribedName	Manages the logical name of a data file
DescribedPhysicalName	Manages the physical name of the HFSQL files (while ignoring the changes of names performed by <i>HChangeName</i>)
Directory	Manages the physical directory of HFSQL files (while taking into account the directory changes performed by <i>HChangeDir</i> or <i>HSubstDir</i>)
Encryption	Allows you to manage the type of encryption for the communication between the Client computers and the HFSQL server
ExecutionCompleted	Used to find out whether the execution of a query or view (Hyper File Classic or Client/Server) is completed
ExtendedInfo	Returns and modifies the additional information of a data file or item accessed by Native Access
Extension	Enables you to find out and modify the extension of a data file
FicCryptMethod	Configures the encryption mode of the data file
FilterCondition	Returns the selection condition implemented by <i>HFilter</i> on a data file, a HFSQL view or a query
FilteredItem	Enables you to find out the item on which a filter was implemented by <i>HFilter</i> on a data file, a HFSQL view or a query
FilterWithBounds	Used to find out whether bounds have been specified on the filter implemented by <i>HFilter</i> on a data file, a HFSQL view or a query
Format	Configures the format of the data found in the data file (ANSI or Unicode)
GenerationNumber	Defines the generation number of the physical file associated with the specified logical file
GèreHTML	Configures the management mode of the HTML format in a full-text index.
GUIDAnalysis	Returns the GUID of the analysis that was used to create the data file
GUIDFile	Returns the GUID of the file defined in the analysis that was used to create the data file
HugeFile	Configures the maximum size of a data file
InfoThumbnailGeneration	Enables you to find out whether the thumbnail was generated or enables you to find out the generation error
InitialFile	Used to find out the name of the initial file corresponding to an item defined by an alias
InitialItem	Used to find out the name of the source item corresponding to an item defined by an alias
IntegerPart	Enables you to find out and modify the number of digits found in the integer part of a numeric item
Items	Enables you to define and find out the different items included in the full-text index.
KeyExpression	Configures the components of a composite key
KeySize	Enables you to find out and modify the size of the indexed section of a text memo
KeyType	Configures the characteristics of a key item
Language	Defines the parameters of the language used to perform the comparisons and the sort for a Unicode item (while taking into account the modifications performed by the <i>H.LanguageCreationUnicode</i> variable).

LanguageDescribed	Defines the parameters of the language used to perform the comparisons and the sort for a Unicode item (while ignoring the modifications performed by the H.LanguageCreationUnicode variable).
Last name	Manages the name of the different HFSQL objects
LinkedFile	Configures the name of the linked file used in the link
LinkedKey	Returns or modifies the item of the linked file used in the link
LogBackupDirectory	Enables you to find out and modify the backup directory of the log files
LogDirectory	Manages the directory of the log file described in the analysis
LogFile	Enables you to find out whether a data file is a log file
LogMethod	Configures the log mode of a data file
ManageRTF	Configures the management mode of the RTF format in a full-text index.
MaxAutoID	Enables you to find out the maximum automatic identifier on a replicated file
MaxLinkedCardinality	Configures the maximum cardinality for the linked item in a link
MaxSourceCardinality	Configures the maximum cardinality for the source item in a link
MaxThumbnailHeight	Enables you to find out or define the maximum height of the Nth thumbnail
MaxThumbnailWidth	Enables you to find out or define the maximum width of the Nth thumbnail
MaxValue	Retrieves the upper bound of the current filter (defined by HFilter) for a data file, a view or a query
Memo	Identifies a memo item
Millisecond	Retrieves or modifies the milliseconds or the number of milliseconds in a Date item ("Date and Time" or "Duration" format)
MinAutoID	Enables you to find out the minimum automatic identifier on a replicated file
MinimumLength	Enables you to define and find out the minimum length of the words to index in a full-text index.
MinLinkedCardinality	Configures the minimum cardinality for the linked item in a link
MinSourceCardinality	Configures the minimum cardinality for the source item in a link
Minute	Retrieves or modifies the minutes or the number of minutes in a Date item ("Date and Time" or "Duration" format) or in a Time item
MinValue	Retrieves the lower bound of the current filter (defined by HFilter) for a data file, a view or a query
MmoCryptMethod	Configures the encryption mode of the memo file associated with the data file
MmoPackMethod	Configures the compression mode of the MMO file associated with the data file
ModificationRule	Manages the modification rule (integrity management) used in a link
Months	Retrieves or modifies the month in a Date item ("Simple Date" or "Date and Time" format)
Name55	Manages the logical name of a file in Hyper File 5.5 format
NbArrayElement	Configures the number of elements in an array item
NbComponent	Returns the number of components in a composite key
NbDescribedThumbnail	Returns the number of thumbnails described for the item
NbItem	Returns the total number of key items (including the composite keys) found in a record for a file described in the data model editor, defined dynamically, a HFSQL view or a query

NbKeyItem	Returns the total number of key items (including composite keys) found in a record for a file described in the data model editor, defined dynamically, a HFSQL view or a query
NbMemoltem	Returns the total number of memo items found in a record of a file described in the data model editor, a file described dynamically, a HyperFileSQL view or a query
NbThumbnail	Returns the number of thumbnails generated for the item
NdxCryptMethod	Configures the encryption mode of the index file associated with a data file
NewRecord	Enables you to find out whether a record was just entered in the data file.
Null	Enables you to manage the NULL value in the items of the HFSQL files
NullSupported	Allows you to manage the NULL value in a HFSQL file
Numeric	Identifies a numeric item
OperationLogDirectory	Enables you to manage the directory for the operation file of the log associated with a logged file
Password	Enables you to define the password of the user who is using the connection
PasswordProtected	Enables you to find out whether a file is password protected
PhysicalName	Manages the physical name of the data files
Provider	Used to manage the type of a connection (HFSQL Client/Server, OLE DB, ...).
PunctuationSensitive	Configures how the punctuation (comma, dot, etc.) will be taken into account in the different searches performed on the item
RecordSize	Returns the size of a record while ignoring the composite keys
Replication	Enables you to find out the replication mode used for a data file (defined in the data model editor or dynamically). For the files that are dynamically defined, you can specify whether this file is in replication mode or not
ReplicationDirectory	Enables you to manage the directory of the replica (".RPL" or ".RPM" file)
Second	Retrieves or modifies the seconds or the number of seconds in a Date item ("Date and Time" or "Duration" format)
Secured	Configures the security level for file encryption
Server	Enables you to find out and modify the data source of a connection
Size	Configures the size of a text item (character string, text memo, character)
SortType	Configures the characteristics of the sort on an item
Source	Enables you to find out and modify the data source of a connection
SourceFile	Configures the name of the source file used in the link
SourceKey	Returns or modifies the item of the source file used in the link
SQLCode	Returns a string containing the SQL code of a query created in the query editor.
Subscript	Returns the physical subscript of the item in the record
Text	Identifies a text item
TextItemCompleted	Configures the management mode of the text items in a HFSQL file (the text items can be automatically filled with space characters or not)
Thumbnail	Enables you to find out the value of the Nth thumbnail
Time	Retrieves or modifies the hour or the number of hours in a Date item ("Date and Time" or "Duration" format) or in a Time item
Time	Retrieves and modifies the hour in a Date item ("Date and Time" format)
TransactionFile	Enables you to find out whether a data file is a transaction file
Type	Identifies and modifies the type of an item
User	Enables you to find out and modify the user of a connection
VisibleEndUser	Enables you to find out whether an item is visible (or not) to the end users and to define whether an item must be visible (or not) to the end users
WDD55	Path of the WDD file in 5.5/1.5 format used to handle the files in 5.5/1.5 format in a WinDev 18 application or a WebDev 18 site
WDD55Password	Gives the password that must be used to handle a file in Hyper File 5.5 format
Year	Retrieves or modifies the year in a Date item

12. SQL FUNCTIONS

The following SQL functions can be used on the queries written in SQL code (classified by theme):

- **extract characters:**
 - ELT
 - EXTRACTVALUE
 - LEFT
 - RIGHT
 - MID, SUBSTR and SUBSTRING
 - SPLIT_PART
- **deleting characters:**
 - LTRIM
 - RTRIM
 - TRIM
- **substituting characters:**
 - REPLACE
 - REVERSE
 - TRANSLATE
- **merging strings:**
 - CONCAT
 - STRING_AGG
- **string completion:**
 - LPAD
 - REPEAT
 - RPAD
 - SPACE
- **modifying the case of a character string:**
 - LOWER
 - UPPER
 - LCASE
 - UCASE
- **size of a character string:**
 - LEN and LENGTH
 - CHARACTER_LENGTH
 - CHAR_LENGTH
 - OCTET_LENGTH
- **position of a character string:**
 - INSTR
 - FIELD
 - PATINDEX
 - POSITION
- **number of records in a file:**
 - COUNT
- **calculating numeric values:**
 - AVG
 - MAX
 - MIN
 - SUM
- **selecting the first n records or the last n records:**
 - BOTTOM
 - TOP
- **ASCII code:**
 - ASCII
- **Unicode code:**
 - UNICODE
- **phonetic:**
 - SOUNDEX
 - SOUNDEX LIKE
 - SOUNDEX2,
 - SOUNDEX2 LIKE
- **managing the dates**
 - ADD_MONTH
 - LAST_DAY
 - MONTHS_BETWEEN
 - NEW_TIME
 - NEXT_DAY
 - ROUND
 - SYSDATE
 - TRUNC
- **Comparison functions:**
 - COALESCE
 - GREATEST
 - IF NULL
 - IS NULL
 - LEAST
 - NVL
- **Conditional statement:**
 - DECODE
 - CASE
- **"Full-text" search:**
 - MATCH AGAINST
- **Checking:**
 - MD5
 - SHA and SHA1

See a specific documentation about the SQL language for more details.

To find out all the SQL commands (functions, clauses, operators, ...) that can be used in an SQL query managed by HFSQL, see the online help.

Notes:

- These statements can be used:
 - In the SQL code of the queries created in the query editor. These queries will be run by *HExecuteQuery*.
 - in the SQL code of queries run by *HExecuteSQLQuery*.
- Unless stated otherwise, these functions can be used with all types of data sources (Oracle, Access, SQL Server, ...).

12.1 Details of functions

12.1.1 ELT

ELT returns the nth character string found in a list of strings.

Use format:

```
ELT(String number, ...
String1, String2, String3, ...)
```

Example:The following SQL code is used to select the first string of the list:

```
SELECT ELT(1, 'ej', 'Heja', ...
'hej', 'foo')
```

12.1.2 EXTRACTVALUE

EXTRACTVALUE is used to handle the XML strings. This function returns the text (CDATA) of the first text node that is a child of the element corresponding to the XPATH expression. If several correspondences are found, the content of the first child text node of each node corresponding to the XPATH expression is returned as a string delimited by space characters.

Use format:

```
EXTRACTVALUE(XML Fragment, ...
Expression XPATH)
```

<Fragment of XML> must be a valid XML fragment. It must contain a unique root.

Example: The following code is used to count the elements found:

```
SELECT EXTRACTVALUE(' <a><b/> ...
</a>', 'count(/a/b)')
FROM CUSTOMER
WHERE CUUNIKKEY=1
```

12.1.3 LEFT

LEFT extracts the left part (which means the first characters):

- from the content of a file item,
- of a character string.

Use format:

```
LEFT(...
CharacterString or ...
Item, NbCharactersToExtract)
```

Example: The following SQL code is used to list the states of customers:

```
SELECT LEFT(CUSTOMER, ZipCode, 2)
FROM CUSTOMER
```

12.1.4 RIGHT

RIGHT extracts the right part (which means the last characters):

- from the content of a file item,
- of a character string.

Use format:

```
RIGHT(...
CharacterString or ...
Item, NbCharactersToExtract)
```

Example: The following SQL code is used to extract the last 5 characters from the customer name:

```
SELECT RIGHT(CUSTOMER.Name, 5)
FROM CUSTOMER
```

12.1.5 SPLIT_PART

SPLIT_PART divides a character string according to the specified separator and returns the nth part of the string.

Use format:

```
SPLIT_PART(Initial Expression, ...
Delimiter, Number of the Part ...
to Extract)
```

Example: The following SQL code is used to extract the first 3 words corresponding to the address:

```
SELECT SPLIT_PART (ADDRESS,' ', ...
1), SPLIT_PART (ADDRESS,' ',2),...
SPLIT_PART (ADRESSE,' ',3)
FROM CUSTOMER
WHERE CUUNIKKEY=2
```

12.1.6 MID, SUBSTR and SUBSTRING

MID, SUBSTR and SUBSTRING are used to:

- extract a sub-string from the content of an item from a given position,
- extract a sub-string from a string from a given position.

MID

MID can be used only on an Access data source.

Use format:

```
MID(...
CharacterString or ...
Item, StartPosition,...
NbCharactersToExtract)
```

Example: The following SQL code is used to extract the third and fourth characters from the customer name:

```
SELECT MID (CUSTOMER.Name, 3, 2)
FROM CUSTOMER
```

SUBSTR

SUBSTR can only be used on an Oracle data source.

Use format:

```
SUBSTR(...
CharacterString or
Item,StartPosition,...
NbCharactersToExtract)
```

Example: The following SQL code is used to extract the third and fourth characters from the customer name:

```
SELECT SUBSTR(CUSTOMER.Name, 3, 2)
FROM CUSTOMER
```

SUBSTRING

SUBSTRING can only be used on an SQL Server data source.

Use format:

```
SUBSTRING(CharacterString
or Item, StartPosition,...
NbCharactersToExtract)
```

Example: The following SQL code is used to extract the third and fourth characters from the customer name:

```
SELECT SUBSTRING (CUSTOMER.Name, 3,
2)
FROM CUSTOMER
```

12.1.7 LTRIM

LTRIM returns a character string:

- without the space characters on the left,
- without a list of characters.

The characters are deleted from left to right. This deletion is case sensitive (uppercase/lowercase characters). This deletion stops when a character that does not belong to the specified list is found. The deletions of specific characters cannot be performed on an Access or SQL Server data source.

The character string passed in parameter to the function corresponds to:

- the content of an item,
- a character string.

Use format:

```
// Del. spaces found on the left
LTRIM(CharacterString/Item)
// Del. list of characters
LTRIM(...
CharacterString or ...
Item,CharactersToDelete)
```

Example: The name of customers is preceded by the title ("Mr.", "Mrs." or "Ms."). The following SQL code is used to:

- delete the title from each name (letters "M", "r" and "s" as well as the dot character),
- delete the space character found in front of the name (space character found between the title and the name).

```
// Delete the characters
SELECT LTRIM (CUSTOMER.Name, 'Ms. ')
FROM CUSTOMER
// Delete the space character
SELECT LTRIM (CUSTOMER.Name)
FROM CUSTOMER
```

In this example:

If the name of the customer is: The returned string is:

'Ms. DOE'	'DOE'
'Mr. CLARK'	'CLARK'
'Mrs. Davis'	'Davis'

12.1.8 RTRIM

RTRIM returns a character string:

- without the space characters on the right,
- without a list of characters.

The characters are deleted from right to left. This deletion is case sensitive (uppercase/lowercase characters). This deletion stops when a character that does not belong to the specified list is found. The deletions of specific characters cannot be performed on an Access or SQL Server data source.

The character string passed in parameter to the function corresponds to:

- the content of an item,
- a character string.

Use format:

```
// Del. spaces on the left
RTRIM(CharacterString or Item)
// Del. list of characters
RTRIM(CharacterString or ...
Item, CharactersToDelete
```

Example: The following SQL code is used to delete the characters 'E', 'U' and 'R' found on the right of customer names:

```
SELECT RTRIM(CUSTOMER.Name, 'EUR.')
FROM CUSTOMER
```

In this example:

If the name of the customer is: The returned string is:

'DUVALEUR'	'DUVAL'
'DRAFUREUR'	'DRAF'
'Galteur'	'Galteur'
'FOURMALTE'	'FOURMALTE'
'BENUR'	'BEN'

12.1.9 TRIM

TRIM returns a character string:

- without the space characters on the left and on the right.
- without a character string found at the beginning or at the end of the string.
- without a character string found at the beginning of the string.
- without a character string found at the end of the string.

The characters are deleted from right to left. This deletion is case sensitive (lowercase/uppercase characters). This deletion stops when a character that does not belong to the specified string is found.

Use format:

```
// Del. spaces on the left and on
// the right TRIM(
// Initial expression)
// Del. character string
// at the beginning and end of
// the string
TRIM(Initial expression, ...
String to delete)
// Del. spaces on the left and on
// the right TRIM
// (Initial expression)
// Del. character string at
// the beginning and end of
// the string
TRIM(Initial expression, ...
String to delete)
// OR
TRIM(BOTH String to delete
FROM Initial expression)
// Del. string at the beginning of
// the string
TRIM(LEADING String to delete
FROM Initial expression)
// Del. string at the end of the
// string
TRIM(TRAILING String to delete FROM
Initial expression)
```

12.1.10 REPLACE

REPLACE returns a character string:

- by replacing all the occurrences of a word found in character string by another word.
- by replacing all the occurrences of a word found in a string.

The replacement is performed from right to left. This replacement is case sensitive (uppercase/lowercase characters). This replacement stops when a character that does not belong to the specified string is found.

Use format:

```
// Replace all the
// occurrences of a word by
// another one
REPLACE(Initial expression, ...
        String to replace, ...
        New string)
// Delete all the
// occurrences of a word
REPLACE(Initial expression, ...
        String to delete)
```

12.1.11 REVERSE

REVERSE returns a character string in which the order of characters is the reversed order of the initial string.

Use format:

```
REVERSE(Initial String)
```

12.1.12 TRANSLATE

TRANSLATE returns a character string by replacing all the specified characters by other characters. If a character to replace has no corresponding character, this character is deleted.

The replacement is performed from right to left. This replacement is case sensitive (uppercase/lowercase characters).

Use format:

```
// Replace the characters
TRANSLATE(Initial expression, ...
          Characters to replace, ...
          New characters)
```

12.1.13 CONCAT

CONCAT concatenates several strings together.

Use format:

```
CONCAT(String 1, String 2 [, ...
        , String N])
```

CONCAT is not supported by Sybase.

12.1.14 STRING_AGG

STRING_AGG is used to concatenate non-null strings from a list of values.

Use format:

```
STRING_AGG(string, separator)
```

12.1.15 LPAD

LPAD returns a string with a given size. To reach the requested size, the string is completed to the left:

- by space characters.
- by a character or by a given string.

Use format:

```
// Fill with space characters
LPAD(Initial expression, Length)
// Fill with a character
LPAD(Initial expression,
     Character)
// Fill with a string of
// characters
LPAD(Initial expression, ...
     Character string)
```

12.1.16 REPEAT

REPEAT returns a character string containing n times the repetition of the initial string.

- If n is less than or equal to 0, the function returns an empty string.
- If the initial string or n is NULL, the function returns NULL.

Use format:

```
REPEAT(Initial String, n)
```

Example: The following SQL code is used to extract the third and fourth characters from the customer name:

```
SELECT REPEAT(CONTACTNAME,14) FROM
CUSTOMER WHERE CUSTOMERID=10
```

12.1.17 RPAD

RPAD returns a string with a defined size. To reach the requested size, the string is completed on the right:

- by space characters.
- by a character or by a given string.

Use format:

```
// Fill with space characters
RPAD(Initial expression,
Length)

// Fill with a character
RPAD(Initial expression,
Character)
// Fill with a string of
// characters
RPAD(Initial expression, ...
Character string)
```

12.1.18 SPACE

SPACE returns a string containing N space characters.

Use format:

```
SPACE (N)
```

12.1.19 LOWER

LOWER converts to lowercase characters:

- the content of an item,
- a character string.

LOWER cannot be used on an Access data source.

Use format:

```
LOWER(CharacterString or Item)
```

Example: The following SQL code is used to convert the first name of customers into lowercase characters:

```
SELECT LOWER(CUSTOMER.FirstName)
FROM CUSTOMER
```

12.1.20 LCASE

LCASE returns a string with all the characters written in lowercase according to the current character set.

Use format:

```
LCASE(Initial Expression)
```

Example: The following SQL code is used to convert the cities of customers into lowercase characters:

```
SELECT LCASE(City) FROM CUSTOMER
```

12.1.21 UCASE

UCASE returns a string with all the characters written in uppercase according to the current set of characters.

Use format:

```
UCASE(Initial Expression)
```

Example: The following SQL code is used to convert the cities of customers into uppercase characters:

```
SELECT UCASE(City) FROM CUSTOMER
```

12.1.22 UPPER

UPPER is used to convert into uppercase characters:

- the content of an item,
- a character string.

UPPER cannot be used on an Access data source.

Use format:

```
UPPER(CharacterString or Item)
```

Example: The following SQL code is used to convert the cities of customers into uppercase characters:

```
SELECT UPPER(CUSTOMER.City) FROM
CUSTOMER
```

12.1.23 LEN and LENGTH

LEN and **LENGTH** return the size (the number of characters) of an expression.

This size includes all the characters, including the space characters and the binary 0.

LEN

LEN can be used with all types of database, except the Oracle data sources. For the Oracle data sources, use **LENGTH**.

Use format:

```
LEN(CharacterString or Item)
```

Example: The following SQL code is used to get the size of customer names:

```
SELECT LEN(CUSTOMER.LastName) FROM
CUSTOMER
```

LENGTH

LENGTH can only be used on an Oracle data source.

Use format:

```
LENGTH(...
CharacterString or Item)
```

Example: The following SQL code is used to get the size of customer names:

```
SELECT LENGTH(CUSTOMER.LastName)
FROM CUSTOMER
```

12.1.24 INSTR

INSTR returns the position of a character string. This character string can be sought:

- the content of an item,
- a character string.

INSTR can only be used on an Oracle data source that supports the SQL-92 standard.

Use format:

```
INSTR(...
CharacterString or ...
Item, StringToFind, ...
StartPosition, Occurrence)
```

Example: The following SQL code is used to get the position of the first occurrence of the letter "T" in each city name:

```
SELECT INSTR(CUSTOMER.City, 'T', 1,
1) FROM CUSTOMER
```

12.1.25 FIELD

FIELD returns the index of the sought string in the list. If the string is not found, the function returns 0.

Use format:

```
FIELD(String to Find, ...
String1, String2, ...)
```

12.1.26 PATINDEX

PATINDEX returns the position of the first occurrence of a character string corresponding to a specified value (with wildcard characters).

The authorized wildcard characters are:

- '%': represents zero, one or more characters.
- '_': represents a single character.

These generic characters can be combined.

This character string can be sought:

- the content of an item,
- a character string.

PATINDEX cannot be used with a HFSQL 7 or SQL Server data source.

Use format:

```
PATINDEX(ValueSought, ...
CharacterString or Item)
```

Example: The following table indicates the position of the first occurrence found according to the different values sought:

City name	Sought Value		
	'%E%'	'%E_'	'%AR%'
MONTPELLIER	6	10	0
PARIS	0	0	2
TARBES	5	5	2
TOULOUSE	8	0	0
VIENNE	3	0	0

12.1.27 POSITION

POSITION returns the position of a character string in an expression.

Use format:

```
POSITION(String to find, ...
Initial expression)
POSITION(String to find, ...
Initial expression, ...
Start position)
```

12.1.28 COUNT

COUNT returns:

- the number of records found in a file.
- the number of non-null values of an item.
- the number of different values and non-null values of an item

Use format:

```
COUNT(*)
COUNT(Item)
COUNT(DISTINCT Item)
```

Examples:

- The following SQL code is used to find out the number of products found in the Product file:

```
SELECT COUNT(*)
FROM PRODUCT
```

- The following SQL code is used to find out the number of products onto which a VAT rate of 5.5 % is applied:

```
SELECT COUNT (PRODUCT,VATRate)
FROM PRODUCT
WHERE PRODUCT.VATRate = '5.5'
```

12.1.29 AVG

AVG calculates:

- the mean for a set of non-null values.
- the mean of a set of different values, not null.

Use format:

```
AVG (Item)
AVG (DISTINCT Item)
```

Example: The following SQL code is used to get the average wages of employees:

```
SELECT AVG (EMPLOYEE.Salary)
FROM EMPLOYEE
```

12.1.30 MAX

MAX returns the greatest of the values found in an item for all the records selected in the file.

MAX used in a query without grouping must return a single record. If the query contains groupings, a record will be returned for each grouping.

If the data source contains records, the record returned by the query will contain the maximum value.

If the data source contains no record, the value of MAX in the record returned is NULL.

Use format:

```
MAX (Item)
MAX (DISTINCT Item)
```

Example: The following SQL code is used to get the maximum wages of employees:

```
SELECT MAX (EMPLOYEE.Salary)
FROM EMPLOYEE
```

12.1.31 MIN

MIN returns the smallest of the values found in an item for all the records selected in the file.

Use format:

```
MIN (Item)
```

Example: The following SQL code is used to get the minimum wages of employees:

```
SELECT MIN (EMPLOYEE.Salary)
FROM EMPLOYEE
```

12.1.32 SUM

SUM returns:

- the sum of the non-null values found in an item for all the records selected in the file.
- the sum of the different and non-null values found in an item for all the records selected in the file.

Use format:

```
SUM (Item)
SUM (DISTINCT Item)
```

Example: The following SQL code is used to get the total sum of all wages:

```
SELECT SUM (EMPLOYEE.Salary)
FROM EMPLOYEE
```

Note: The item handled by SUM must not correspond to the result of an operation. Therefore, the following syntax generates an error:

```
SELECT (A*B) AS C, SUM(C)
FROM MYFILE
```

This syntax must be replaced by the following syntax:

```
SELECT (A*B) AS C, SUM(A*B)
FROM MYFILE
```

12.1.33 BOTTOM

BOTTOM only returns the n last records of a query result.

BOTTOM can only be used on a HFSQL data source.

Use format:

```
BOTTOM NbrOfLastSelectedRecord
```

Example: The following SQL code is used to list the 10 worst customers:

```
SELECT BOTTOM 10 SUM (ORDERS.TotalIncTax) AS TotalIncTax,
CUSTOMER.CustomerName
FROM CUSTOMER, ORDERS
WHERE CUSTOMER.CustNum
=ORDERS.CustomerNum
GROUP BY CustomerName
ORDER BY TotalIncTax DESC
```

Note: We recommend that you use **BOTTOM** on a sorted query. Otherwise, the records returned by **BOTTOM** will be selected according to their record number.

12.1.34 TOP

TOP returns the first n records of the query result. **TOP** cannot be used on an Oracle data source.

Use format:

```
TOP NbrOfFirstSelectedRecord
```

Example: The following SQL code is used to list the 10 best customers:

```
SELECT TOP 10 SUM(ORDERS.TotalInc-
Tax) AS TotalIncTax,
CUSTOMER.CustomerName
FROM CUSTOMER, ORDERS
WHERE CUSTOMER.CustNum
=ORDERS.CustomerNum
GROUP BY CustomerName
ORDER BY TotalIncTax DESC
```

Note: We recommend that you use **TOP** on a sorted query. Otherwise, the records returned by **TOP** will be selected according to their record number.

12.1.35 ASCII

ASCII returns the ASCII code:

- of a character.
- of the first character in a string.

If the specified character or character string is an empty string (""), **ASCII** returns 0.

Use format:

```
// ASCII code of a character
ASCII(Character)
// ASCII code of the first charac-
ter
// in a string
ASCII(Character string)
```

12.1.36 SOUNDEX, SOUNDEX LIKE

SOUNDEX returns the phonetic representation of a string (based on an English-language algorithm).

Use format:

```
// ASCII code of a character
ASCII(Character)
// ASCII code of the first charac-
ter
// in a string
ASCII(Character string)
```

Note: **SOUNDEX** used on different databases (Hyper File, Oracle, MySQL, ...) may return different results depending on the database used.

SOUNDEX LIKE is not supported by Oracle, MySQL, Progress or Informix. **SOUNDEX** is not supported by Informix.

12.1.37 SOUNDEX2, SOUNDEX2 LIKE

SOUNDEX2 returns the phonetic representation of a character string (based on an algorithm close to French).

Use format:

```
SOUNDEX2(String)
```

SOUNDEX2 and **SOUNDEX2 LIKE** are not supported by Oracle, SQL Server, MySQL, Progress, Informix or DB2.

12.1.38 ADD_MONTHS

ADD_MONTHS is used to add several months to a specified date.

Use format:

```
ADD_MONTHS(Date,Number of months)
```

ADD_MONTHS is not supported by SQL Server, MySQL, Informix, DB2 or Sybase.

12.1.39 LAST_DAY

LAST_DAY is used to find out the date of the last day for the specified month.

Use format:

```
LAST_DAY(Date)
```

LAST_DAY is not supported by Informix, DB2 or Sybase.

12.1.40 MONTHS_BETWEEN

MONTHS_BETWEEN is used to find out the number of months between two specified dates.

Use format:

```
MONTHS_BETWEEN(Date1, Date2)
```

MONTHS_BETWEEN is not supported by MySQL, Informix, DB2 or Sybase.

12.1.41 NEW_TIME

NEW_TIME is used to find out a date after conversion of time zone.

Use format:

```
NEW_TIME(Date, Time Zone 1, Time
Zone 2)
```

NEW_TIME is not supported by SQL Server, MySQL, Progress, Informix, DB2 or Sybase.

12.1.42 NEXT_DAY

NEXT_DAY is used to find out the first day of the week following the specified date or the specified day.

Use format:

```
NEXT_DAY(Date, Day)
```

NEXT_DAY is not supported by SQL Server, MySQL, Progress, Informix, DB2 or Sybase.

12.1.43 ROUND

ROUND is used to round the date to the specified format.

Use format:

```
ROUND(Date, Format)
```

ROUND is not supported by Progress, DB2 and Sybase.

12.1.44 SYSDATE

SYSDATE is used to find out the current date and time.

Use format:

```
DATESYS
```

SYSDATE is not supported by Informix and Sybase.

12.1.45 TRUNC

TRUNC is used to truncate the date to the specified format.

Use format:

```
TRUNC(Date, Format)
```

TRUNC is not supported by SQL Server, MySQL, Progress, DB2 and Sybase.

12.1.46 COALESCE

COALESCE is used to find out the first non-null expression among its arguments.

Use format:

```
COALESCE(Param1, Param2, ...)
```

COALESCE is not supported by Progress or Informix.

12.1.47 GREATEST

GREATEST returns the greatest value of the elements passed in parameter

Use format:

```
GREATEST(Param1, Param2, ...)
```

12.1.48 LEAST

LEAST returns the lowest value of the elements passed in parameter

Use format:

```
LEAST(Param1, Param2, ...)
```

12.1.49 NVL, IF_NULL, IS_NULL

NVL is used to replace the null values of a column by a substitution value. **IS_NULL** and **IF_NULL** are identical.

IS_NULL is used in SQL Server and **IF_NULL** is used with the MySQL or Progress databases.

Use format:

```
NVL(Column name, ...
Substitution value)
```

12.1.50 DECODE

DECODE is used to find out the operating mode of an IF statement .. **THEN** .. **ELSE**.

Use format:

```
DECODE(Column_Name, Compared
value 1, Returned value 1, [Compa-
red value 2, Returned
value 2] [, Default value])
```

DECODE is not supported by SQL Server, MySQL, Progress, Informix, DB2 and Sybase.

12.1.51 CASE

CASE is used to find out the operating mode of a IF statement .. THEN .. ELSE;

Use format:

```
CASE Column_Name WHEN Compared ...
value 1 THEN Returned ...
value 1 [WHEN compared ...
Value 2 THEN Returned ...
Value 2] [ELSE default ...
Returned Value] END
```

Or:

```
CASE WHEN Condition 1 THEN ...
Returned value 1 [WHEN ...
Condition 2 THEN Returned ...
Value 2] [ELSE default ...
Returned Value] END
```

Example:

```
SELECT itmInt, CASE itmInt ...
WHEN 3 THEN 'three' WHEN 4 ...
THEN 'four' ELSE 'other' END
```

Or

```
SELECT itmInt, CASE WHEN ...
itmInt=3 THEN 'three' WHEN ...
itmInt=4 THEN 'four' ELSE ...
'other' END
```

12.1.52 MATCH AGAINST

MATCH AGAINST is used to find out the pertinence of the record during a full-text search.

Use format:

```
MATCH(List of items) ...
AGAINST [ALL] Value
```

Or:

- **List of items** corresponds to the list of index items separated by commas (the order of the items is not important).

- **Value** corresponds to the value sought in the different items. This parameter can correspond to a literal value or to a parameter name. The search value can contain the following elements:

Element	Meaning
A single word	The specified word will be sought. The relevance will be increased if the text contains this word. Example: "WinDev" searches for the word "WinDev".
Two words separated by a space character	Searches for one of the words. Example: "WinDev WebDev" find the texts that contain either "WinDev" or "WebDev".
A word preceded by the "+" sign	The specified word is mandatory. Example: "+WinDev" finds the texts that necessarily contain "WinDev".
A word preceded by the "-" sign	The specified word must not be found in the text. Example: "-Index" find the texts that do not contain the word "Index".
A word preceded by the "~" sign	The specified word must not be found in the text. Example: "~Index" find the texts that do not contain the word "Index".
One or more words enclosed in quotes	The specified words are searched in group and in order. Caution: if "Ignore the words smaller than " differs from 0, the words in-between quotes whose length is shorter than the specified length will not be sought.
A word followed by the "*" sign	The type of the search performed is "Starts with" the specified word.

[ALL] is used to force the replacement of space characters by "+" in the sought value.

Example: In this example, EDT_Search is an edit control and ConnectedUserID is a variable.

```
MyQuery is string =
[SELECT * FROM Contact WHERE
MATCH(Contact.LastName,
Contact.FirstName, Contact.
HTMLComment,
Contact.RoughTextComment,
Contact.Comments,
Contact.Phone, Contact.Business,
Contact.Cell, Contact.Mail, Con-
tact.MSN, Contact.Internet_site,
Contact.Country, Contact.FaxNum,
Contact.City)
AGAINST ('
]
MyQuery = MyQuery +
EDT_Search + [
']
AND Contact.UserID =
]
MyQuery = MyQuery +
IDConnectedUser + [
ORDER BY Name DESC
]
HExecuteSQLQuery(QRY_SRCH,
hQueryDefault, MyQuery)
FOR EACH QRY_SRCH
TableAddLine(...
TABLE_Contact_by_category, ...
QRY_SRCH.IDContact, QRY_SRC.
IDCategory, IDConnectedUser, ...
QRY_SRCH.LastName, QRY_SRCH.First-
Name)
END
CASE ERROR:
Error(HErrorInfo())
```

See the online help for more details about the full-text search.

12.1.53 MD5

MD5 calculates the MD5 check sum of the string passed in parameter. The returned value is an hexadecimal integer of 32 characters that can be used as hash key for example.

Use format:

```
MD5(String)
```

12.1.54 SHA and SHA1

SHA and SHA1 calculate the 160-bit SHA1 check sum of the string passed in parameter according to the RFC 3174 standard (Secure Hash Algorithm). The returned value is an hexadecimal string of 40 characters or NULL if the argument is NULL. This function can be used for hashing the keys

Use format:

```
SHA(String)
```

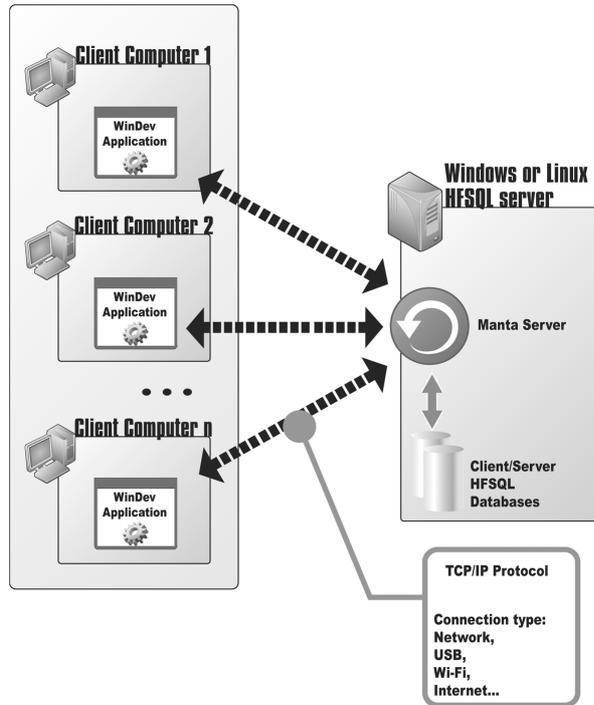
```
SHA1(String)
```

13. HFSQL CLIENT/SERVER

13.1 Overview

WinDev enables you to create applications that access the HFSQL Client/Server databases. A HFSQL Client/Server application runs an application on several computers (called client computers)

and stores the databases and processes on a server. This operating mode makes response times faster and more reliable, and it simplifies the maintenance of the database.



WinDev enables you to:

- create a HFSQL Client/Server application from scratch.
- modify an existing application into a HFSQL Client/Server application.

13.2 Implementing a Client/Server application

The steps for implementing a HFSQL Client/Server application are as follows:

1. Configure the server used.
2. Connecting the client computers to the server.
3. Configure the data files on the server.

4. Using the application.

For more details about HFSQL Client/Server, see the online help.

14. HFSQL CLIENT/SERVER FUNCTIONS

The following functions are used to manage the HFSQL Client/Server engine:

HActivateServerTrigger	Re-enables a server trigger that was disabled by HDeactivateServerTrigger .
HAddGroup	Adds a group of users
HAddLink	Adds an integrity rule between two files on the server
HAddScheduledBackup	Adds a scheduling for full and differential backup on the server defined by the connection
HAddScheduledOptimization	Adds an optimization task of HFSQL Client/Server data files.
HAddTask	Adds a scheduled task on the server defined by the connection.
HAddUser	Adds a user to a database
HBackup	Saves the content of a Hyper File server: all the server databases, one or more databases, one or more files. This backup can be performed while one or more databases are currently used.
HCancelBackup	Cancels a current backup.
HChangePassword	Changes the password of a HFSQL Client/Server data file
HClearWorkingDir	Clears and destroys the temporary directory that was previously created during the execution of HServerWorkingDir
HClusterAddNode	Enables a node in a HFSQL cluster.
HClusterDeleteNode	Disables a node in a HFSQL cluster.
HClusterIgnoreSynchro	Defines a node of the HFSQL cluster as data source to perform the cluster synchronization.
HClusterNodeInfo	Returns the status of each cluster node by interrogating the coordinator
HClusterParameter	Reads and modifies the parameters of a HFSQL cluster
HClusterStart	Starts a HFSQL cluster.
HClusterState	Returns the status of a HFSQL cluster by interrogating its coordinator
HClusterStop	Suspends the execution of a HFSQL cluster
HConnectionQuality	Returns the quality level of the connection: the higher the level is, the faster the connection is
HCopyFile	Copies a HFSQL file
HCreateServerTrigger	Adds or modifies a server trigger on the HFSQL server.
HDeactivateServerTrigger	Disables a Hyper File Client/Server server trigger. This trigger can be re-enabled by HActivateServerTrigger .
HDeleteBackup	Deletes a backup that was performed by HBackup . The backup is physically deleted from the disk.
HDeleteDatabase	Deletes a database found on a HFSQL server
HDeleteDirectory	Deletes a directory found in a HFSQL Client/Server database
HDeleteFile	Deletes the HFSQL files (.fic, .ndx and .mmo files if they exist) from the server
HDeleteGroup	Deletes (from the server) a group of users associated with a connection
HDeleteLink	Deletes an integrity rule between two files on the server
HDeleteParameter	Deletes a parameter that was previously saved by HSaveParameter .
HDeleteQuery	Deletes a query from a Hyper File server.
HDeleteScheduledBackup	Deletes a scheduling for full and/or differential backup from a HFSQL Client/Server server
HDeleteScheduledOptimization	Deletes a scheduled optimization task of HFSQL Client/Server data files
HDeleteServerTrigger	Destroys a server trigger. This server trigger cannot be used anymore.
HDeleteSet	Deletes a set of stored procedures from a Hyper File server.

HDeleteTask	Deletes a scheduled task from a Hyper File Client/Server server.
HDeleteUser	Deletes (from the server) a user associated with a connection
HDescribeServerTrigger	Adds or modifies a server trigger.
HDisconnectClient	Displays a message on the client computers and disconnects the application
HEndNoDatabaseAccess	Re-authorizes the access to one or more databases accessible by a connection
HExecuteProcedure	Runs a stored procedure or function.
HExecuteScheduledBackup	Forces the execution of a scheduled backup.
HInfoBackup	Returns information about one or more backups performed on a Hyper File Client/Server server.
HInfoDatabaseProperty	Used to find out the properties of a database found on a HFSQL server
HInfoDatabaseRights	Allows you to find out the rights granted to a user or to a group on a database
HInfoFile	Returns the characteristics of a file found on a HFSQL server
HInfoFileProperty	Used to find out the properties of a data file found on a HFSQL server
HInfoFileRights	Allows you to find out the rights granted to a user or to a group on a data file
HInfoGroup	Returns information about the specified group of users
HInfoLock	Returns information about the lock performed on a file, on a record or on all the file records
HInfoLog	Returns information about the server logs.
HInfoServer	Returns the specified information about the server
HInfoServerProperty	Used to find out the properties of a HFSQL server
HInfoServerRights	Allows you to find out the rights granted to a user or to a group on a server
HInfoTask	Returns the characteristics of a scheduled task in an advanced hScheduledTask variable.
HInfoUser	Updates the variables for user management with the information about the specified user
HListConnectedUser	Returns the list of users currently connected to one or more files handled by a Client/Server connection
HListDatabase	Lists the Client/Server databases associated with a connection
HListGroup	Returns the list of groups of users defined for a connection
HListParameter	Returns the list of parameters saved from the stored procedures on the server.
HListScheduledBackup	Lists the full and differential backups that have been scheduled on a HFSQL Client/Server server.
HListScheduledOptimization	Lists the scheduled optimization tasks of the HFSQL Client/Server data files for a connection
HListServer	Lists the HFSQL servers installed on a computer
HListServerTrigger	Lists the different triggers available on a connection or on one of the connection files.
HListStoredElement	Returns the list of elements stored on a Hyper File server (sets of procedures, stored procedures or queries).
HListTask	Lists the scheduled tasks found on a server.
HListUser	Returns the list of users defined for a connection
HLoadParameter	Reads a parameter that was saved from a stored procedure by HSaveParameter.
HManageTask	Enables or disables a scheduled task on a Hyper File Client/Server server. This function can also be used to find out the status of a scheduled task.
HModifyDatabaseProperty	Modifies the properties of a database found on a HFSQL server

HModifyDatabaseRights	Modifies the rights granted to a user or to a group for a HFSQL Client/Server database
HModifyFileProperty	Modifies the properties of a HFSQL file found on a server
HModifyFileRights	Modifies the rights granted to a user or to a group on a HFSQL Client/Server data file
HModifyGroup	Modifies the group information according to the elements found in the corresponding variables for group management
HModifyScheduledBackup	Modifies a scheduling for a full backup
HModifyScheduledOptimization	Modifies a scheduled optimization task on the HFSQL server defined by the connection. This task is found in an advanced hScheduledOptimization variable.
HModifyServerProperty	Modifies the properties of a HFSQL server
HModifyServerRights	Modifies the rights granted to a user or to a group on a HFSQL server
HModifyTask	Modifies a scheduled task on the server defined by the connection. This task is found in an advanced hScheduledTask variable.
HModifyUser	Modifies the user information according to the elements found in the corresponding variables for user management
HNoDatabaseAccess	Forbids all the accesses to a database or to all the databases accessible via a connection
HNotifAddCCRecipient	Adds recipients for the notifications sent via the Control Centers (WDBal messaging tool).
HNotifAddEmailRecipient	Adds recipients for the notifications sent by email.
HNotifConfigure	Specifies and configures the server used to send notifications by the HFSQL server
HNotifDeleteCCRecipient	Deletes the recipients of a notification sent via the Control Centers (WDBal messaging tool).
HNotifDeleteEmailRecipient	Deletes the recipients of a notification by email.
HNotifListCCRecipient	Returns the list of recipients for a notification sent via the Control Centers (WDBal messaging tool).
HNotifListEmailRecipient	Returns the list of recipients for a notification by email.
HOnServerCall	Customizes the management of message display on the client computer and the management of disconnection from a client computer
HPriority	Used to find out and modify the priority of the calling application.
HPriorityClient	Modifies the priority of a client application
HReconnect	Establishes a reconnection to the server for all the interrupted connections.
HRefreshQuery	Creates or updates a query on a Hyper File server
HRefreshSet	Creates or updates a set of procedures on a Hyper File server
HResetClient	Initializes the structure for managing the client computers (HClient structure)
HResetGroup	Initializes the variables for group management with the default values
HResetUser	Initializes the variables for user management with the default values
HRestoreBackup	Used to restore a backup performed by HBackup (or via the HyperFileSQL Control Center).
HRetrieveLog	Retrieves in a file the server logs performed between two given dates
HSaveParameter	Saves a persistent value from a stored procedure. This value can be read by HLoadParameter.
HSendMessageToClient	Displays a message on the client computers
HServerStatus	Used to find out the status of a server

HServerWorkingDir	Returns the path of a temporary directory on the server. This directory is automatically created on the server. The stored procedures run on the server will have the rights to write into this directory. This directory will be automatically cleared and destroyed when the client is disconnected.
HSetCache	Used to configure the management of caches in the HFSQL Client/Server engine
HSetServer	Used to modify some parameters of the server.
HSimulateNetwork	Simulates the operating mode of HFSQL Client/Server on an ADSL or 3G network
HStartServer	Used to start a server (uses MantaManager)
HStopServer	Stops a server
HTransactionList	Returns the list of current or interrupted transactions found on the server for the specified connection

See the online help for more details.

See the online help to check the availability of these functions in the different products (WinDev, WebDev or WinDev Mobile).

Index

"Back" key	125
.JNL	353
.NET	315
.Net	
Assembly	315
.REP	344

A

Accessing external databases	387
ADD_MONTHS	406
AJAX	220
Archive	200
ZIP functions	202
Archive file (functions)	202
ASCIIZ string	41
Assembly (.NET)	315
Assign file	344
Assisted management of errors	373
Automatic (code) indent	32
Automatic completion	31
Automatic identifier	339
Automation object (Variable)	54
AVG	405

B

Back (Management)	125
Back Office	385
Binary	
Binary memos	343
Functions	153
BOTTOM	405
Break (Statement)	87
Breakpoint	33
Browse according to filter	328
Buffer (Variable)	42
Buffer type	42
Burn	203
Burner	
Functions	203
Burning	203

C

CD (burn)	203
Character (Variable)	41

Character string	155
Handling	155
Character strings	
Functions	156
Chart	
3D	162
Charts	161
Column	161
Creation	163
Default values	163
Functions	164
Line	161
Pie	161
Presentation	161
Scatter	162
Stock	162
Types	161
Client	
HFSQL Client/Server	410
Client/Server	410
Clipboard (functions)	181
COALESCE	407
Code	
Assisted input	31
Auto completion	31
Automatic indent	32
Breakpoints	33
Check	32
Color	31
Help	31
History	32
Message translation	32
Specific processes	33
Wizard	31
Code check	32
Code history	32
Code wizard	31
Color	
Code	31
Communication	241
Emails	243
Fax	277
FTP server	284
HTML page	280
Server of XML WebServices	305
SOAP	301
Socket	286
Telephony	270
Thread	293
WinDev FTP/RPC	281

XML file	311
Composite key	322, 324
CONCAT	402
Constant	46
Continue (Statement)	85
Control	
Functions	137
Properties	137
Cookies	214
COUNT	404
CTI	270
Currency (Variable)	41

D

Data replication	382, 386
Data source (Variable)	56
Database	387
External	387
Date	
Functions	159
Date (Variable)	42
Date type	42
DateTime type	43
Duration type	43
Time type	43
Dates and times	159
DATESYS	407
DateTime (Variable)	43
DDE	
Functions	193
Dead lock	369
DECODE	407
des actions à refaire	358
Description	
of a file (Variable)	57
Of a link (Variable)	58
Of item (Variable)	59
Dialog (dialog boxes)	128
Dialog box	128
Multilingual	130
Dialog boxes	
Functions	132
Download	213
Download a file	211, 213
Drag and Drop	133
From the explorer	135
Functions	136
Drag and drop	
Automatic	133

Programmed	134
Duplicate	322, 328
Duplicate key	322, 328
Duplicates	373
Duration (Variable)	43
Dynamic array (Variable)	48
Dynamic automation object (Variable)	55
Dynamic structure (Variable)	54

E

Editor menu (Ribbon)	30
Email	243
Functions	250
POP 3 and SMTP	244
Simple MAPI	247
Update HFSQL files	385
Write	250
Encrypting HFSQL files	339
Error	
Assisted management	373
Exception management	104
Excel	199
Functions	199
Exception (Mechanism)	104
Exception mechanism	104
Executable	
Functions	192
Explorer (Drag and Drop)	135
Extern (Keywords)	89
External files	
Functions	168
Handling disks and directories	167
Handling files	167
Handling the content	167
Presentation	167

F

Fax	277
Application for sending faxes	278
Fax server	277
Functions	279
Fax server	277
Programming	278
File	
Zip	200
File alias	336
File browse	327
File variable	325

Triggers	380
Update by email	385
View	377
HFSQL File	
Assign	344
Backup	355
Create	321
Identifier	323, 339
Key	340
Log	353
Memos	343
Open	321
Protection	339
Restore	355
HFSQL file	
Alias	336
Browse	327
Browse according to a filter	328
Close	321
Lock	368
Logical name	321
Long name	321, 344
Optimization	359
Positioning	330
Read operation according to key	326
Remote access (RPC)	385
Size greater than 2 GB	322
Update by email	385
Variable	325
View	377
HFSQL view	377
Programming	378
HTML page	280
Retrieve	280
HTTP (functions)	280

I

Identifier	323
IF (Statement)	83
IF_NULL	407
Importing an XML Webservice	305
Incoming calls	270
INSTR	404
Integer (variable)	40
Integrity check	340
Inter locking	369
IS_NULL	407
Item	325
Key	322

J

J++	361
Java (functions)	188
JDBC	361

K

Key	322
Composite key	324
Key type	322
Read operation according to key	326
Key uniqueness	340
Keyboard	
Associated processes	33
Keywords	
Extern	89
Modulo	91
MyFile	93
MyPage	90
MyParent	93
MyPopupControl	92
MyReport	93
Myself	91
MySource	91
MyWindow	90
Stop	94

L

Language	
SQL	364
LAST_DAY	406
LDAP	269
LEFT	399
LEN	403
LENGTH	403
Link	
Types of links	341
Lock	368
HFSQL file	368
Record	369
Lock Mode	368, 370
Log	353
Programming	356
Log file	353
Logical Name	321
Long name	321, 344
Loop (Statement)	75

LOWER	403
LPAD	402
LTRIM	400

M

MAPI	247
Matrices	151
Functions	153
Handling	151
MAX	405
MCI (functions)	184
MDI	
Functions	121
MDI window (functions)	121
Memo	343
Memo file	343
Memory zone (functions)	166
Menu	
Functions	121, 124
Menu option	
Function	121, 124
Message (functions)	132
MID	400
MIN	405
Modulo (Keywords)	91
MONTHS_BETWEEN	407
Mouse	
Associated processes	33
Functions	181
Moving inside a file	330
Multilingual	32
Dialog box	130
MyFile (keyword)	93
MyPage (Keywords)	90
MyParent (Keywords)	93
MyPopupControl (Keywords)	92
MyReport (Keywords)	93
Myself (Keywords)	91
MySource (Keywords)	91
MyWindow (Keywords)	90

N

Net (functions)	283
Network (functions)	194
NEW_TIME	407
NEXT_DAY	407
NULL	44
NULL in HFSQL	337

Numeric (Variable)	41
Numeric key	324
NVL	407

O

ODBC driver on HFSQL	360, 361, 362
Operator	66
Arithmetic	67
Binary	67
Indirection	72
Logical	66
Miscellaneous	73
On address	72
On character strings	70
operator	
Comparison	69
Optimization	359
Outgoing calls	272

P

Page	
Functions	123
Overview	123
Pane (ribbon menu)	30
Parameters of a procedure	99
Pascal string on	41
Password	339
Variable	339
PATINDEX	404
POP3	244
Popup (window)	
MyPopupControl	92
Popup window	
MyPopupControl	92
Previous (Button)	125
Print	174
Drawings	175
Fonts	175
Functions	177
Images	175
Parameters	174
Principles	174
Text	175
Printing	
Overview	174
Procedure	95
Call	99
Declare	98

Global	95
Local	97
Parameters	99
Sets of procedures	98
Project	
Functions	194
Properties	
HFSQL	394
Property	
DateTime	43
Duration	43
File description	57
Hour	43
Item description	59
Link description	58
Windows and controls	137
Protecting files	339
Protocol	
POP 3	244
SMTP	244
SOAP	301
SSL	233

R

Real (Variable)	41
Record	325
Add	329
Delete	329
Loaded in memory	327
Modify	329
Pointed	325, 327
Read	326
Seek	326
Write	326
Referential integrity	340, 341
Registry (functions)	180
Remote access	
HFSQL file	385
REPLACE	401
Replication file	382, 386
Reserved words	89
Result (Statement)	87
Retrieving the HTML pages	280
Return (Statement)	86
Ribbon (Editor)	30
RIGHT	399
ROUND	407
RPAD	402
RTRIM	401

Rule for variable scope	65
-------------------------	----

S

Search	
Exact match	323
Generic	323
Search argument	323
Search Criterion	323
Search key	327
Secured transaction	232
Securing the transactions (SSL)	233
Seeking records.	326
Semaphore in the threads	294
Send faxes	277
Serial and parallel ports (functions)	182
Server	
HFSQL Client/Server	410
Service (functions)	184
Set of procedures	98
Signals	297
Simple array (Variable)	47
Site	
AJAX	220
Link with WinDev	385
SMTP	244
SNMP (Functions)	194
SOAP	301
Functions	304
SOAP server	301
Running procedures	301
Socket	286
Functions	290
SOUNDEX	406
SOUNDEX2	406
Specific processes	33
Speeding up processes	359
SQL	364
AVG	405
BOTTOM	405
COUNT	404
Functions	398
HFSQL	364
INSTR	404
LEFT	399
LEN	403
LENGTH	403
LOWER	403
LTRIM	400
MAX	405

