

What's New in Omnis Studio 4.3.1

TigerLogic Corporation

May 2008

The software this document describes is furnished under a license agreement. The software may be used or copied only in accordance with the terms of the agreement. Names of persons, corporations, or products used in the tutorials and examples of this manual are fictitious. No part of this publication may be reproduced, transmitted, stored in a retrieval system or translated into any language in any form by any means without the written permission of TigerLogic.

© TigerLogic Corporation, and its licensors 2008. All rights reserved.

Portions © Copyright Microsoft Corporation.

Regular expressions Copyright (c) 1986,1993,1995 University of Toronto.

© 1999-2008 The Apache Software Foundation. All rights reserved.

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

OMNIS® and Omnis Studio® are registered trademarks of TigerLogic Corporation.

Microsoft, MS, MS-DOS, Visual Basic, Windows, Windows 95, Win32, Win32s are registered trademarks, and Windows NT, Visual C++ are trademarks of Microsoft Corporation in the US and other countries.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world.

IBM, DB2, and INFORMIX are registered trademarks of International Business Machines Corporation.

ICU is Copyright © 1995-2003 International Business Machines Corporation and others.

UNIX is a registered trademark in the US and other countries exclusively licensed by X/Open Company Ltd.

Sun, Sun Microsystems, the Sun Logo, Solaris, Java, and Catalyst are trademarks or registered trademarks of Sun Microsystems Inc.

J2SE is Copyright (c) 2003 Sun Microsystems Inc under a licence agreement to be found at: <http://java.sun.com/j2se/1.4.2/docs/relnotes/license.html>

Portions Copyright (c) 1996-2008, The PostgreSQL Global Development Group

Portions Copyright (c) 1994, The Regents of the University of California

MySQL is a registered trademark of MySQL AB in the United States, the European Union and other countries (www.mysql.com).

ORACLE is a registered trademark and SQL*NET is a trademark of Oracle Corporation.

SYBASE, Net-Library, Open Client, DB-Library and CT-Library are registered trademarks of Sybase Inc.

Acrobat is a trademark of Adobe Systems, Inc.

Apple, the Apple logo, AppleTalk, and Macintosh are registered trademarks and MacOS, Power Macintosh and PowerPC are trademarks of Apple Computer, Inc.

HP-UX is a trademark of Hewlett Packard.

OSF/Motif is a trademark of the Open Software Foundation.

CodeWarrior is a trademark of Metrowerks, Inc.

This software is based in part on ChartDirector, copyright Advanced Software Engineering (www.advsofteng.com).

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

This software is based in part of the work of the FreeType Team.

Other products mentioned are trademarks or registered trademarks of their corporations.

Table of Contents

ABOUT THIS MANUAL	6
WHAT'S NEW IN OMNIS STUDIO 4.3.1.....	7
WINDOW CLASSES	8
<i>Window toolbars.....</i>	<i>8</i>
<i>Tab panes.....</i>	<i>8</i>
<i>Group boxes.....</i>	<i>8</i>
<i>Buttons and System focus.....</i>	<i>8</i>
<i>Lists.....</i>	<i>9</i>
<i>Icon Arrays.....</i>	<i>9</i>
<i>Transbutton.....</i>	<i>9</i>
<i>Sheets.....</i>	<i>10</i>
<i>Alpha colors.....</i>	<i>10</i>
WINDOWS VISTA	10
LIBRARY CONVERSION.....	11
IDE TOOLS	11
<i>Omnis VCS.....</i>	<i>11</i>
<i>Component Store.....</i>	<i>11</i>
<i>Method Editor.....</i>	<i>11</i>
<i>Omnispic.....</i>	<i>11</i>
ORACLE.....	11
<i>Universal Oracle DAM for Mac OS X.....</i>	<i>11</i>
POSTGRESQL DAM.....	12
<i>PostgreSQL Client Libraries.....</i>	<i>12</i>
<i>Server-specific Programming.....</i>	<i>13</i>
<i>Session Properties.....</i>	<i>15</i>
<i>Session Methods.....</i>	<i>17</i>
<i>Statement Properties.....</i>	<i>17</i>
<i>Data Type Mapping.....</i>	<i>18</i>
<i>PostgreSQL 8.3 Data Types.....</i>	<i>20</i>
<i>Troubleshooting.....</i>	<i>21</i>
MYSQL.....	21
<i>Session Methods.....</i>	<i>21</i>
WHAT'S NEW IN OMNIS STUDIO 4.3.....	22
WINDOWS VISTA	23
<i>End User Security.....</i>	<i>23</i>
<i>Program and Data directories.....</i>	<i>23</i>
<i>Deployment under Vista.....</i>	<i>24</i>
<i>System Fonts.....</i>	<i>25</i>
<i>Detecting Vista.....</i>	<i>25</i>

<i>Default button preference</i>	25
MAC OS X.....	25
<i>Detecting Leopard</i>	25
NET OBJECTS	26
UNICODE	26
<i>Data File Conversion</i>	26
<i>Future versions of Omnis</i>	27
<i>Handling Char & Binary data under Unicode</i>	27
OMNIS VCS.....	28
<i>Method Inspector</i>	28
<i>Scheduled Builds</i>	28
<i>Superclass names</i>	29
GRAPH2.....	30
WEB CLIENT	30
<i>Safari for Windows</i>	30
<i>Windows Web Client installer</i>	31
<i>Apache plug-ins</i>	31
<i>Linux Web Client</i>	31
<i>Printing on the Web Client</i>	32
<i>Web Data Grid</i>	33
LINUX	36
<i>Printing under Linux</i>	36
<i>Printer management (CUPS)</i>	38
<i>Linux Fonts</i>	38
<i>Gnome & KDE environments</i>	39
<i>File names</i>	39
<i>Save window setup</i>	39
<i>Tooltips, Menus & droplists</i>	40
<i>Events on inactive objects</i>	40
WINDOW PROGRAMMING	40
<i>Radio Button Groups</i>	40
<i>Tab/Page panes</i>	41
<i>Headed lists and Data grids</i>	41
<i>Tree lists</i>	42
<i>Refreshing window instances</i>	42
<i>Losing property values</i>	42
<i>Format Strings</i>	42
<i>Input Masks</i>	43
<i>Calendar Component</i>	43
REPORTS	43
<i>Using MS Sans Serif with Report Fields</i>	43
ODBC	44
<i>ODBC Administration</i>	44
ORACLE8 DAM	48
SQL PROGRAMMING	49

<i>Session Pools</i>	49
FURTHER ENHANCEMENTS	49
<i>Screen resolution</i>	49
<i>Java</i>	50
<i>Method lines</i>	50
<i>Help</i>	50
<i>External Components</i>	51
<i>System key codes</i>	51
<i>Item References</i>	51
<i>Find & Replace</i>	51
<i>Remote Studio</i>	52
FUNCTIONS.....	52
<i>isclear()</i>	52
<i>isleopard()</i>	52
<i>isvista()</i>	52
<i>mouseover()</i>	53
<i>rxpos()</i>	53
<i>sys(6)</i>	53
<i>sys(7)</i>	53
<i>sys(115)</i>	53
<i>sys(202)</i>	53
<i>sys(215)</i>	53
LOTUS NOTES	54
<i>NSF List Attachments command</i>	54

About This Manual

This document describes the new features and enhancements in Omnis Studio 4.3.1, which mainly concerns improvements for Omnis under Mac OS X 10.5 Leopard. This manual also contains the information from the ‘What’s New in Omnis Studio 4.3’ manual, published with Studio 4.3 in November 2007. In particular, developers who have used PostgreSQL in Studio 4.3 should note that the section about PostgreSQL has been updated and republished here under the 4.3.1 section.

The information in this document supplements that provided in the main Omnis Studio 4.1 manual set (published in November 2005), together with the ‘What’s New in Omnis Studio 4.2’ manual issued in August 2006.

Please see the file `Readme.txt` for details of bug fixes and any last minute notes in this release.

If you are new to Omnis Studio

If you are new to Omnis Studio you should start by reading the *Introducing Omnis Studio* manual and then the *Omnis Programming* manual. All the Omnis Studio manuals are available on the product DVD and to download from the Omnis web site.

What's New in Omnis Studio 4.3.1

Omnis Studio 4.3 was released in November 2007 and fully supports Windows Vista and Mac OS X 10.5 (codenamed “Leopard”) for Development and Runtime versions. This latest release, Omnis Studio 4.3.1, is intended specifically to further improve the appearance and behavior of Omnis Studio running under Mac OS X 10.5.

The following enhancements and fixes have been added to Omnis Studio 4.3.1.

- ❑ **Window toolbars, Tab panes and Group boxes**
Window toolbars on Mac OS X are now displayed as part of the window title bar, while Tab panes and Group boxes have the standard Mac OS X appearance
- ❑ **System focus for buttons**
The system focus for Radio buttons, Check boxes and standard Pushbuttons is improved, as well as the appearance of the Transbutton control
- ❑ **List appearance**
The appearance of several list types is improved; alternating colors are displayed in the background of list boxes and headed lists, while icon arrays look better on Mac OS X
- ❑ **Message boxes as sheets**
All built-in and custom messages (OK/Working/Prompt messages) are displayed as sheet windows, plus you can control the transparency of custom sheet windows
- ❑ **Icons**
Omnispic contains some new and improved icons, including new sizes (32x32) for some icons that were not previously supplied
- ❑ **Component Store**
the default appearance of the Component Store has changed; it opens in small icons with text mode, while component groups can be selected from the toolbar
- ❑ **Method Editor**
The lists of methods that appear in the parameter pane of the Method Editor are lengthened to accommodate more method names, making coding easier
- ❑ **Library Conversion**
Omnis Studio 4.3.1 will convert a library created with any previous version of Omnis Studio, including Studio 4.1, 4.2, and all previous revisions of Studio 4.3
- ❑ **Oracle**
there is a new Universal Binary edition of the Mac OS X Oracle DAM
- ❑ **PostgreSQL**
several minor enhancements to the PostgreSQL DAM have been made including support for PostgreSQL 8.3 data types

Window Classes

Window toolbars

When the window style supports it, window toolbars now default to being standard operating system toolbars that are part of the window title bar. All toolbar controls are supported under Mac OS X 10.5 apart from the combo box toolbar controls (which cannot be supported due to focus issues).

You can access the old toolbar styles using a new option in the `$toolbaroptions` property of the window class, as follows:

`kTBOptionOSXOmnisTopToolbar`

On Mac OS X (for window styles that support Mac OS X style top toolbars), this option uses the Omnis style rather than the Mac OS X style for the top toolbar; this option is only applied when opening the window.

There is a new window property called `$toolbarbutton`. If true, the window has a button in the title bar to hide and show the toolbar (Mac OS X only, this only applies when the window has a Mac OS X style top toolbar).

There is also a new method called `$toolbarbuttonclick()` which sends a click to the toolbar button on the window instance (Mac OS X only, this only applies when the window has a Mac OS X style top toolbar).

Tab panes

When using tab pane controls with `kDefaultPanels` as the tab style, the tab panes now have the standard Mac OS X appearance. However, the multi-row property is ignored for `kDefaultPanels` on Mac OS X; this limitation on Mac OS X applies to custom tab panes, as well as built-in windows that use tab panes, such as the Omnis Catalog.

Group boxes

Group boxes now have the correct Mac OS X appearance when using the `kBorderCtrlGroupBox` effect.

Buttons and System focus

In general, the system focus rectangle now draws tighter around many controls, giving it a more standard Mac OS X appearance. Therefore, all window controls that display the system focus, such as standard pushbuttons, Radio buttons, check boxes, popup menus, and list controls, now correctly display the focus using the standard operating system theme.

Note that if you turn on full keyboard access, via the System Preferences or by typing `Ctrl-F7`, Omnis reacts to this property change only after it has been put into the background and then restored to the foreground, and only when new windows are opened (this applies when the library preference `$canfocusbuttons` is `kPlatformdefault`).

Lists

List boxes

Lists with a background theme of `kBGThemeControl` now draw their line background using alternating colors – see the Mac Finder for an example. Lists now also display selected lines in the same way as the Mac Finder.

Headed lists

Headed lists have a new property called `$headerfontsize` which specifies the font size for the text in the header; if this is zero, the header text font size is the same as `$fontsize`. This means that Omnis can use a small font size for the header, like the Mac Finder.

Check boxes

Check box lists now display selected lines in the correct manner for Mac OS X.

Dropdown lists

The `$backgroundtheme` property controls how dropdown lists look and behave on Mac OS X 10.5, as follows:

- If the theme is `kBGThemeControl`, and provided the list has ≤ 1000 lines, the dropdown list looks and behaves as a popup menu (the standard form of dropdown list on OS X).
- If the theme is `kBGThemeWindow`, the dropdown list behaves like a normal dropdown list, and has a standard themed OS X appearance. In the case where the list has 1001 lines or more, and theme `kBGThemeControl`, the look and behavior becomes that of `kBGThemeWindow`.

Combo boxes

When the `$backgroundtheme` property is set to `kBGThemeControl` Combo boxes look and behave as they should on Mac OS X 10.5.

Icon Arrays

Icon Arrays now display selected icons in a similar manner to the Mac Finder. Icon Arrays have two new properties (which apply on all platforms, and both default to zero): `$largetextwidth` is the width in pixels of the text, when displaying large icons; to use the default width, set this property to zero; and `$extraspacing` which is the extra spacing in pixels added to each icon.

Transbutton

Under Mac OS X, the `Transbutton` control now has the appearance and behavior more like a standard Mac OS X toolbar button.

Sheets

When the `$usesheets` window property is `kTrue`, the *Prompt for input* command displays a sheet window. In addition, most OK messages, working messages, and so on, in the Omnis IDE itself appear as sheet windows.

Alpha colors

Window classes and instances have a new property called `$alpha` which allows you to specify the transparency of the window. The value of the property is in the range 0-255, where 0 is completely transparent and 255 is opaque. The property applies to Mac OS X only.

The `$alpha` property contains the alpha channel value which controls the transparency of the entire window, including its title, background, and all the controls on the window. You can therefore use the `$alpha` setting to specify the transparency of a window opened as a sheet.

Windows Vista

In the Windows Vista version of Omnis Studio 4.3 (or above), when *User Account Control* (UAC) is turned on, you get the following message when you start different versions of the Omnis executable, that is, the Development, Runtime, Unicode or Non-Unicode versions:

“Omnis Studio needs to run `studiorg.exe` to install registry settings. User Account Control will ask you to allow `studiorg.exe` to run. Please allow it to run.”

This problem cannot be overcome as there is only one set of registry entries associating file extensions (such as, `lbs`) with the executable; the UAC prompt is related to the executable associated with the file associations changing.

Instead, you can disable this file association checking and updating process by adding the entry `"UpdateFileAssociations=0"` to the General section of `omnis.ini` in the Data Directory under Windows Vista (the `..\AppData\Local\` folder), or the normal Omnis root folder on other Win32 platforms. You will need to create the `omnis.ini` file, if one does not exist, in the following format:

```
[General]
UpdateFileAssociations=0
```

This causes the file associations to be left set to the last executable that was allowed to update the registry, therefore stopping the UAC message on Windows Vista. This has the same effect on other Win32 platforms, but since there is no UAC and therefore no messages, the inclusion of the ini file entry may not be necessary.

See the “What’s New in Studio 4.3” section for more information about the configuration of the Omnis Program and Data Directories in Windows Vista regarding the UAC.

Library Conversion

Omnis Studio 4.3.1 will convert all libraries created with any previous version of Omnis Studio, including version 4.1 and 4.2, as well as all previous releases of Studio 4.3. This means that Studio 4.3.1 will attempt to convert a library created with 4.3.0 and 4.3.0.1.

IDE tools

Omnis VCS

There are two new preferences in the VCS Options (on the Other tab).

- Ignore Unicode / Non-Unicode Directory Structure**
allows you to stop the folders Unicode & non-Unicode being created in the build folder tree. If unchecked, the Unicode & non-Unicode folders are created.
- Only Create Locked and Unlocked Folders When Required**
prevents the Locked/Unlocked folders from being created in the build folder tree unless you have selected 'Use separate locked and unlocked folders' in the build window.

Component Store

On all platforms, the Component Store now has a toolbar for selecting the group of components to be displayed. Under Mac OS X, the default appearance of the Component Store has changed; it now opens in small icons with text mode.

Method Editor

Under all platforms, the lists of methods that appear in the parameter pane of the Method Editor have been lengthened to accommodate more method names.

Omnispic

There is a new copy of the omnispic.df1 icon data file containing some improved icons, including new sizes (typically 32x32) for some icons that were not previously supplied.

Oracle

Universal Oracle DAM for Mac OS X

Omnis Studio 4.3.1 includes a Universal Binary edition of the Oracle object DAM (DAMORA8). On the Intel platform however, please note that the Intel Oracle client library (libclntsh.dylib.10.1) is compatible with Mac OS 10.5 only. To use this DAM with Mac OS 10.4, you still need to install the Mac PPC edition of Instant Client and run Studio in Rosetta mode. For further details, please refer to Omnis technotes TNSQ0010 and TNSQ0011.

PostgreSQL DAM

Note: *Developers who have used PostgreSQL in Studio 4.3 should note that this section, previously contained in the 'What's New in Omnis Studio 4.3' manual, has been updated and republished here.*

There is a new Omnis DAM that supports connections to PostgreSQL database. The following section contains the additional information you need to access a PostgreSQL database, including server-specific programming, trouble-shooting and data type mapping to and from the database. For additional information on changes to the PostgreSQL DAM, refer to the Readme file. For more information about PostgreSQL itself, please see: www.postgresql.org

For general information about logging on to and managing your database using the SQL Browser, please refer to the *Omnis Programming* manual.

PostgreSQL Client Libraries

This section discusses the PostgreSQL client libraries, which must be present on the library search path before the PostgreSQL DAM can be used.

Win32 platforms

For Win32 platforms, the library search path includes the Windows\System32 folder or any location in the PATH environment variable, including the folder containing omnis.exe. The Win32 client library is named **libpq.dll**.

Linux and Mac OS X platforms

The Unix ports of the PostgreSQL DAM look for libpq.so or libpq.dylib under Linux and Mac OS X respectively. (There are no other dependencies).

In most cases the library present on your system will be labelled according to the version you have installed. For example on Linux, libpq.so might be a symbolic link to the target library libpq.so.5.0. A detailed directory listing shows this relationship, e.g.

```
12 2007-01-15 10:20 libpq.so -> libpq.so.5.0
117338 2007-01-15 10:20 libpq.so.5.0
```

Under Linux and Mac OS X therefore, it is essential that the target library and symbolic link to it *both* exist either in the library search path or in the same folder as the Omnis executable.

Mac OS X only

Note that unless your client library is built as a universal binary, use will be restricted to machines with either the Mac-Intel or Mac-PPC architecture. If you are unsure which architecture(s) your library supports, use the **file** terminal command to display the target architecture. E.g. `file libpq.5.0.dylib`

A universal binary library should give the following output:

```
libpq.5.0.dylib: Mach-O universal binary with 2 architectures
libpq.5.0.dylib (for architecture i386): Mach-O dynamically linked
  shared library i386
libpq.5.0.dylib (for architecture ppc): Mach-O dynamically linked
  shared library ppc
```

Whereas a library built for a single architecture would display only one of these lines.

Server-specific Programming

Logging on to PostgreSQL

In addition to the hostname, username and password parameters provided by the `$login()` method, the PostgreSQL DAM provides several session properties which enable additional logon parameters to be set. These should be set before calling `$login()`.

`$database` is used to specify the `dbname` connection parameter.

`$port` is used to specify the port connection parameter.

`$logontimeout` is used to specify the `connect_timeout` parameter.

`$options` is used to specify further optional connection parameters.

`$service` is used to specify a service (filename) to use for additional parameters.

Metadata Functions

`$indexes()`

The `DamInfoRow` for `$indexes()` is defined with a single column containing the SQL text used to define the index.

`$tables()`

The PostgreSQL DAM implements `$tables()` slightly differently. In particular- only the `kStatementServerTable` and `kStatementServerView` parameters are supported. This is because the processes for querying tables are incompatible with those for querying views. (`kStatementServerAll` defaults to `kStatementServerTable`).

The `DamInfoRow` for `$tables()` is defined with three Boolean columns with additional information on the table or view: `HasIndexes`, `HasRules` & `HasTriggers`.

Transaction Support

PostgreSQL supports two transaction isolation levels: Read Committed (the default) and Serializable. Using Read Committed mode, a statement can only see rows that were committed before it began. Using Serializable mode, all statements in the current transaction can only see rows that were committed before the first query or data-modification statement was executed in this transaction.

Transactions can also be instantiated as read-only if required. This enables significant performance improvements for read operations. When a transaction is read-only, the following SQL commands are disallowed: `INSERT`, `UPDATE`, `DELETE`, and `COPY FROM` if the table they would write to is not a temporary table; all `CREATE`, `ALTER`,

and DROP commands; COMMENT, GRANT, REVOKE, TRUNCATE; and EXPLAIN ANALYZE and EXECUTE if the command they would execute is among those listed. Please refer to the PostgreSQL documentation on transactions for further details.

When using manual transaction mode (kSessionTranManual) the transaction isolation level can be switched between Read Committed and Serializable using the \$serializable property. The access mode can be changed using the \$readonly property.

The PostgreSQL DAM treats the kSessionTranAutomatic and kSessionTranServer transaction modes identically. In either of these modes the server automatically begins and commits read/write transactions.

Remote Procedure Calls

Note that PostgreSQL does not support the concept of stored procedures but supports functions instead. This has a few implications as described below.

\$rpcprocedures()

The DamInfoRow returned by \$rpcprocedures is defined with the following columns:

Language	The implementation language or call interface for this function.
IsAgg	kTrue if this is an aggregate function.
SecDef	kTrue if this function is a security definer (i.e. a "setuid" function).
IsStrict	kTrue if this is a "strict" function. Strict functions must be prepared to handle null inputs.
RetSet	kTrue if the function returns a result set (i.e. multiple values of the specified data type).
Volatile	Indicates whether the function's result depends only on its input arguments, or is affected by outside factors. It is i for "immutable" functions, which always deliver the same result for the same inputs. It is s for "stable" functions, whose results (for fixed inputs) do not change within a scan. It is v for "volatile" functions, whose results may change at any time. (Use v also for functions with side-effects, so that calls to them cannot get optimised away.)
Source	Indicates how the function should be invoked. It might be the actual source code of the function for interpreted languages, a link symbol, a file name, or just about anything else, depending on the implementation language/call convention.

\$rpc()

Calling \$rpc() is similar to executing a SQL SELECT statement of the form:

```
SELECT * from proc_name (param1, param2, ... )
```

with the exception that \$rpc() will also set any InputOutput or Output parameters.

Any return value generated by the function will be available via \$rpcreturnvalue although in the case where the function generates a result set, it may be preferable to retrieve the

entire set by calling `$fetch()`. The value returned by `$rprcreturnvalue` is also returned as the first row of this result set.

Session Properties

Property	Description
<code>\$maxvarchar</code>	Defines the maximum size above which- Omnis Character fields will be mapped to TEXT type instead of VARCHAR. The default value for this property is 2000.
<code>\$database</code>	Used to set the additional dbname logon parameter. If not specified, defaults to be the same as the user name.
<code>\$service</code>	Service name to use for additional parameters. It specifies a service name in <code>pg_service.conf</code> that holds additional connection parameters. This allows applications to specify only a service name so connection parameters can be centrally maintained.
<code>\$protocolversion</code>	(Read-only)This property reports the communication protocol version supported by the client library. DAMPGSQL requires version 3.0 or higher in order to work correctly.
<code>\$backendpid</code>	(Read-only) Following logon, this property holds the process ID of the backend server process handling the connection. This may be useful for debugging purposes since the PID is reported in NOTIFY messages.
<code>\$port</code>	Used to set the additional port logon parameter. This property has a default value of 5432.
<code>\$socket</code>	(Read-only) Following logon, this property holds the file descriptor number of the connection socket to the server. A valid descriptor will be greater than or equal to 0; a result of -1 indicates that no server connection is currently open.
<code>\$logontimeout</code>	Maximum wait for a connection, in seconds. Zero implies wait indefinitely. The default timeout is set to 15 seconds. A timeout of less than 2 seconds is not recommended.
<code>\$timezone</code>	Character string representing the time zone to be appended on to bind variables being inserted into TIMETZ and TIMESTAMPTZ columns. The default time zone is “+00” but <code>\$timezone</code> will accept any character string (80 characters max).
<code>\$usetimezone</code>	If set to <code>kTrue</code> , the value contained in <code>\$timezone</code> is appended to outgoing Time and Datetime bind variables. This property also affects the text returned by <code>\$createnames()</code> for Time and DateTime columns. Insertion to TIMETZ and TIMESTAMPTZ columns requires that <code>\$usetimezone</code> is <code>kTrue</code> . Likewise, insertion to TIME and TIMESTAMP columns requires that <code>\$usetimezone</code> is <code>kFalse</code> . For this

Property	Description
	reason, it is best not to mix these column types within a given table.
\$serializable	If set to kTrue, manual transactions will be created using the Serializable isolation level. When set to kFalse (the default), manual transactions will be created using the Read Committed isolation level.
\$readonly	If set to kTrue, manual transactions will be created using read-only access mode. When set to kFalse (the default), transactions will have read/write access.
\$schema	The optional schema name to be prepended to table names. Used by the SQL Browser when performing SELECTs. The default schema name is an empty string.
\$numericprecision	Defines the precision used by \$createnames() when mapping Omnis number (dp) columns to the NUMERIC type. Cannot be set lower than the default value: 15.
\$sequencetoint	If set to kTrue, the Omnis Sequence type is mapped to INTEGER. If set to kFalse (the default), the Sequence type is mapped to SERIAL. Affects \$createnames() and outward bind variables.
\$char38touuid	If set to kTrue, Omnis character types of field length 38 are mapped to the PostgreSQL 8.3 Universally Unique Identifier type (UUID).

Session Methods

Method	Description
\$connectstatus()	Returns a PGSQLDAM- Connection Status constant representing the current state of the connection to the database server, or empty if not connected.
\$transactionstatus()	Returns the current in-transaction status of the server. The status can be kPgSqlTranIdle (currently idle), kPgSqlTranActive (a command is in progress), kPgSqlTranInTrans (idle, in a valid transaction block), or kPgSqlTranINError (idle, in a failed transaction block). kPgSqlTranUnknown is reported if the connection is bad. kPgSqlTranActive is reported only when a query has been sent to the server and not yet completed.
\$parameterstatus()	Looks up a current parameter setting of the server. Supported (string) parameters include server_version, server_encoding, client_encoding, is_superuser, session_authorization, DateStyle, TimeZone, integer_datetimes, and standard_conforming_strings. For a full list, refer to the API documentation for the PqparameterStatus function.
\$reset()	Resets the communication channel to the server. This function will close the connection to the server and attempt to reestablish a new connection to the same server, using all the same parameters previously used. This may be useful for error recovery if a working connection is lost.
\$cancel()	Requests that the server abandon processing of any transactions pending on the session. Successful execution is no guarantee that the request will have any effect, however. If the cancellation is effective, the current command(s) will terminate early and return an error result.
\$addcustomtype()	Creates a custom data type mapping for specified Omnis character subtypes. Intended to allow creation and insertion into PostgreSQL 8.3 enum and xml columns.
\$clearcustomtypes()	Removes previously created custom data type mappings.

Statement Properties

Property	Description
\$sqlstate	(Read only) On error, this property contains the five-character SQLSTATE associated with the \$nativeerrortext. Refer to the PostgreSQL reference manual for a full list of SQLSTATES.

Data Type Mapping

Omnis to PostgreSQL

Omnis Data Type	PostgreSQL Data Type
CHARACTER	
Character/National n (n<=\$maxvarchar)	VARCHAR(n)
Character/National n (n>\$maxvarchar)	TEXT
Character(38)	UUID ^[3]
NUMBER	
Long integer	INTEGER
Short integer	SMALLINT
Number 0..14dp	NUMERIC(15 ^[1] ,0..14)
Short number 0/2dp	NUMERIC(9,0/2)
Number floating dp	REAL
DATE/TIME	
Short date (all subtypes)	DATE
Short time	TIME
Datetime (all subtypes)	TIMESTAMP
OTHER	
Boolean	BOOLEAN
Sequence	SERIAL/INTEGER ^[2]
Picture	BYTEA
List	BYTEA
Row	BYTEA
Object	BYTEA
Binary	BYTEA
Item reference	BYTEA

[1] Numeric precision for Number (dp) columns uses the value of \$numericprecision.

[2] The mapping used for the Omnis Sequence type depends on the value of \$sequencetoint.

[3] This mapping occurs only if \$char38touuid is set to kTrue

PostgreSQL to Omnis

PostgreSQL Data Type	Description	Omnis Data Type
NUMBER		
INT2/SMALLINT	-32768 to +32767	Long integer
INT/INT4/INTEGER	-2147483648 to +2147483647	Long integer
INT8/BIGINT	-2 ⁶³ to +2 ⁶³ -1	Character
SERIAL	1 to 4294967296	Long integer
SERIAL8/BIGSERIAL	1 to 2 ⁶⁴	Character
FLOAT4/FLOAT/REAL	1E-37 to 1E+37	Number floating dp
DOUBLE/FLOAT8	1E-307 to 1E+308	Number floating dp
NUMERIC	Numbers with max precision 1000	Number floating dp ^[1]
MONEY	-21474836.48 to +21474836.47	Number 2dp
DATE/TIME		
DATE	(with/without time zone)	Datetime (#FDT)
TIMESTAMP	(with/without time zone)	Datetime (#FDT)
TIME	(with/without time zone)	Datetime (#FDT)
INTERVAL	Flexible format time interval	Character
CHARACTER		
CHAR	Blank-padded characters with size limit	Character
VARCHAR	Variable length characters with size limit	Character
TEXT	Variable length characters with no size limit	Character
BOOLEAN/BOOL	{'f', 'false', 'n', 'no', '0', 't', 'true', 'y', 'yes', '1'}	Boolean
CIDR, INET, MACADDR	Strings containing address information	Character
UUID	Universally Unique Identifier	Character 36
ENUM	Custom enumerated types	Character 64
XML	Extensible Markup Language content	Character
OTHERS (including, but not limited to)		
BYTEA, BIT, VARBIT, BOX, CIRCLE, POINT, LINE, PATH, POLYGON, LSEG		Binary

^[1] The DAM will map decimal values to the Omnis Number dp data type where the column scale is <=14

PostgreSQL 8.3 Data Types

Support for the following data types is available as of Omnis Studio 4.3.1.

UUID

The PostgreSQL DAM is able to read and write Universally Unique Identifiers. An example of a UUID in standard form might be:

a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11 (36 characters), but the DAM also accepts UUIDs formatted without hyphens and/or encapsulated using curly braces.

Output from UUID columns is always in the standard form.

To allow input binding of UUIDs and to make \$createnames() return UUID types, it is necessary to set \$char38touuid to kTrue. Once set, the Omnis Character 38 data subtype maps to UUID.

Note: there is no facility either in the PostgreSQL client library or in the DAM to create UUID values. This must be implemented in the Omnis application.

ENUM

Enumerated types are created by executing CREATE TYPE statements, e.g.

```
Do statObj.$execdirect(
    "CREATE TYPE mood AS ENUM ('sad', 'ok', 'happy')")
    Returns #F ;;creates the enumerated type
```

To make Omnis map certain character sub types to ENUMs, the \$addcustomtype() method is provided.

The following example maps the Omnis Character 2001 data subtype to the “mood” enumerated type:

```
Do sessObj.$addcustomtype(2001, 'mood') Returns #F
```

Once set, this mapping affects the text generated by \$createnames() as well as input binding.

To clear previously defined enumerated type mappings, the \$clearcustomtypes() method is provided.

XML

The \$addcustomtype() method can also be used to force an Omnis Character subtype to map to the XML data type, e.g.

```
Do sessObj.$addcustomtype(10001, 'xml') Returns #F
```

As above, this mapping affects the text generated by \$createnames() as well as input binding and remains in effect until \$clearcustomtypes() is called.

Troubleshooting

The following points may help in resolving programming issues encountered using PostgreSQL session and statement objects. For additional updated troubleshooting issues, refer to the readme file which accompanies the installation media.

\$rpcparameters()

When calling `$rpcparameters()`, the DAM uses defaults for the column precision and/or scale since this information is not provided by the `pg_proc` system table.

For this reason, the API may report parameter-matching problems when calling certain functions and the list (passed to `$rpcdefine()`) may need to be manually coerced.

Error Messages

The following additional error messages may be returned via the session or statement `$errortext` property:

"Native error text could not be retrieved". No connection currently exists to the server or there is no message corresponding to the current error code.

"Unsupported client protocol version". The protocol version reported by the client API is too low. The DAM cannot use this version and you should upgrade to a newer version of the client library. Use the PostgreSQL access library supplied with Omnis Studio.

"Client or interface function not available". The most likely cause of this error is that the client library (or one of its dependencies) was not found and has not been loaded. Can also occur if the client library being used does not provide a required interface function.

MySQL

Session Methods

The `SessObj.$connectoption()` method can be used to set extra connect options and affect behaviour for a MySQL connection. Its functionality has been enhanced. This function should be called before logging on and may be called multiple times to set multiple options. Available option constants can be found in the Catalog (F9) under `MYSQLDAM-ConnectOptions`. `vArgument` should be passed either as an integer or as text, since `$connectoption()` currently cannot evaluate constants. For further details regarding the options available, refer to the MySQL C API documentation for `mysql_options()`.

What's New in Omnis Studio 4.3

The following is a summary of the most significant features included in Omnis Studio version 4.3. There are many other minor enhancements that have been added to Omnis Studio 4.3 that are also documented in this manual.

- ❑ **Windows Vista and Mac OS X 10.5 “Leopard” support**
Omnis Studio 4.3 fully supports Windows Vista and Mac OS X 10.5 (codenamed “Leopard”) for development and Runtime versions
- ❑ **.NET Objects**
Studio 4.3 allows you to call functionality within the Microsoft .NET Framework or third-party class libraries; .NET support is provided via the .NET Objects external component, which is described in the *Omnis_NetObjects.pdf* manual
- ❑ **PostgreSQL DAM**
There is a new Omnis DAM that supports connections to the PostgreSQL database, an open source relational database system released under a BSD-style license, providing Omnis developers with yet another alternative to other database systems.
- ❑ **Unicode Data File Conversion**
The Unicode version of Studio 4.3 has a full data file converter which converts the Character data in your Omnis data files and rebuilds the indexes. *IMPORTANT: Please see the “Unicode” section in this manual for full details about the data file conversion process in Unicode Omnis Studio 4.3 and comments regarding the next major release of Omnis.*
- ❑ **Omnis VCS enhancements**
The Omnis VCS contains a new tool, called the Method Inspector, that allows you to examine the methods in a class and look at the code in each method. In addition, the VCS supports scheduled builds and allows library names in superclass names.
- ❑ **New Graph types**
Gauge and meter style graphs have been added to the Graph2 component, together with several other enhancements, which are described in the *Omnis_Graph2.pdf* manual.
- ❑ **Web Client support for Windows Safari**
The Web Client plug-in now supports the Windows version of Safari, recently released by Apple. In addition, there is now only one Web Client installer under Windows that installs the Web Client for all supported browsers into one folder.
- ❑ **Linux enhancements**
There has been a number of enhancements made to the Linux version of Omnis Studio mainly to improve printing, as well as improvements in font handling and other user interface elements.

Windows Vista

Omnis Studio 4.3 fully supports Windows Vista for development and Runtime versions. Please read the following sections for details regarding Vista support in Omnis Studio.

End User Security

Windows Vista introduced a system called *User Account Control* (UAC) to improve security against unwanted attacks of one form or another on an end-user's computer. UAC provides a secure mechanism for running end-user accounts with Standard User privileges, while removing the need for Administrator privileges when performing many common tasks, such as installing a printer driver. In Omnis, the measures introduced under UAC affect where files in the Omnis Studio product tree are stored on your computer to allow you to create/modify Omnis libraries or write to other files in your application. This will also affect how you construct installers for deploying your own products under Windows Vista.

Program and Data directories

Due to the introduction of the UAC mechanism, Omnis now uses two locations for installing the Omnis executable, plus any read-only files and other files that require write access during development. The Omnis installer places all files in the correct location during installation, and changes to the product tree are made automatically the first time Omnis is run.

❑ Program Directory

For the Omnis executable and read-only files: "C:\Program Files\RainingData\Omnis Studio xxx". Note that executable files do not include Omnis libraries, as these often need write access.

❑ Data Directory

For writeable files, which are placed in the \AppData\Local\ folder as follows (note the AppData folder is usually hidden so you may need to show it using the Tools>>Folder Options option in the Windows File Manager):

The location of the Data Directory is either:

C:\users\name\AppData\Local\RainingData\Omnis Studio xxx

Or

C:\Windows\System32\config\systemprofile\AppData\Local\RainingData\Omnis Studio xxx (when Omnis is running as a service)

The Omnis installer places all the writeable files into a folder called 'firstruninstall' in the Program directory: these files are then copied to the Data directory the first time Omnis Studio is run. You will need to take this into account when you are building installers for your own products, see below.

Note that Omnis can still run from a single location, provided that the location is not in Program Files, and you have write access to the tree in that location.

Under Vista, sys(115) returns the path of the Data directory, i.e. the path to all writeable files, while sys(215) returns the path of the Program directory; on all other platforms, sys(215) returns the same value as sys(115).

Deployment under Vista

Due to the presence of the Program and Data directories in the Windows Vista installation of Omnis, you need to understand how the Omnis product tree is installed under Vista and therefore structure the installers for your own products a little differently.

Firstruninstall directory

For Windows Vista, the installed tree in Program Files must now have a directory called 'firstruninstall' in the Program directory. This contains the following directories and files, previously present in the Program directory:

Adhoc	Appbuild	Charmap.ini
Convert	Help	Html
Icons	Java	Local
Localclient	Startup	Studio
Welcome	Serial.txt (if present)	

After installation, when Omnis starts up for the first time, it copies the files in firstruninstall to the Data Directory – this happens in quite a short time, so no progress dialog is displayed while it occurs. Any files or directories in firstruninstall that are not present in the Data directory, are copied recursively to the Data directory.

Registry Settings

There is a new program called studiorg.exe. The installer runs this program in order to update the registry with the entries required by Omnis Studio beneath HKEY_LOCAL_MACHINE, such as file associations and event logging settings. Therefore, studiorg.exe must also be present in the Program directory, since Omnis can invoke it at startup if the registry needs updating.

When Omnis starts up on Windows Vista it first checks the registry to see if it is up to date (if for example you move the Studio tree, or run multiple copies of Studio from different locations, the registry needs changing to reflect the new path to omnis.exe). If the registry needs changing, Omnis invokes the program studiorg.exe. If Omnis is not running with administrative privileges (the default), it will display a message explaining that studiorg.exe needs to run, and after the user clicks OK, the Windows Vista UAC dialog will open, asking the user if studiorg.exe can run. Assuming the end user answers Yes, file associations and event logging are set up correctly.

System Fonts

Windows Vista introduces a new system font called ‘Segoe UI’ and uses a default font of Segoe UI 9 point. This can cause conflicts with the current default font on Windows XP which is Tahoma 8 point. The following changes have been made in Studio 4.3 to handle the new default font.

There is a new font name available in the Omnis Window & Report font tables, and the #STYLES system table, called ‘Omnis Windows System’ which represents the different system fonts under the different versions of Windows; therefore, on Windows XP it maps to Tahoma, and on Windows Vista it maps to Segoe UI. If you specify the Omnis Windows System font with a point size of *Zero*, Omnis will use 8 point Tahoma on Windows XP, and 9 point Segoe UI on Windows Vista.

The default Window and Report font tables on Windows now use Omnis Windows System for all new libraries. In addition, the default Omnis styles specified in #STYLES will use the new Omnis Windows System font.

Detecting Vista

There is a new function called *isvista()* which returns true when Omnis is running on Windows Vista; it returns false for all other platforms and earlier versions of Windows. It can be executed in the Web Client.

The following additions have been made to *sys(6)* and *sys(7)* to support Windows Vista although you should tend to use the new *isvista()* function: *sys(6)* returns N on Vista, while *sys(7)* returns 6.0.

Default button preference

The *\$osxflashdefaultbutton* preference has been renamed to *\$flashdefaultbutton* and controls whether or not default OS buttons animate on Windows Vista, as well as on Mac OS X. The current setting of this preference on Mac OS X is maintained.

Mac OS X

Omnis Studio 4.3 fully supports Mac OS X 10.5, codenamed “Leopard” by Apple, for Development and Runtime versions. Mac OS X 10.5 was released in October 2007 and has many extra features and enhancements over previous versions of OS X.

Detecting Leopard

There is a new function called *isleopard()* which returns true when Omnis is running on Mac OS X 10.5 (Leopard); it returns false for all other platforms and earlier versions of Mac OS X. It can be executed in the Web Client.

NET Objects

Studio 4.3 allows you to call functionality within the Microsoft .NET Framework or third-party class libraries. .NET support is provided via the *.NET Objects* external component, which is described in the *Omnis_NetObjects.pdf* manual. Note there are various software requirements for using the new component; these are described in the manual.

There is a new example library showing how the *.NET Objects* component works, which is located under the Examples link in the Welcome window when you first launch Omnis – you can open this window by clicking on the New Users button in the main Omnis toolbar.

Unicode

Data File Conversion

WARNING: You should make a secure backup of your Omnis data files before converting them in the Unicode version of Studio 4.3.

The Unicode version of Studio 4.3 has a full data file converter which converts the data in your Omnis data file and rebuilds the indexes. In the Unicode version of Studio 4.2, the data was not converted, simply the indexes were dropped and rebuilt, and the data file was marked as Unicode.

When this full data file conversion takes place, in Studio 4.3, all data marked as Character is converted. In the case where character data is stored in a binary, for example, text stored in a document file, conversion of this data *does not* take place.

When you access a data file in the Unicode version of Studio 4.3 you are asked to confirm that you want to convert the data. After you select Yes, Studio displays a dialog which offers two types of conversion:

- Quick**
whereby the indexes are dropped and rebuilt, but the data is not converted (that is, the same conversion process as Studio 4.2). This is OK for files containing only 7 bit data: Omnis does not check that the file only contains 7 bit data, so it's your responsibility to know whether or not it is safe to run this conversion process.
- Full**
the new conversion process whereby a full conversion of the Character based data takes place.

Data File commands

The *Open data file* and *Prompt for data file* commands have an existing option called "Convert without user prompts". If this is checked, the new conversion dialog is not displayed. There is a new option for these commands, "Quick Unicode conversion" or "Full Unicode conversion", which allows you to select the level of conversion to take place.

Future versions of Omnis

You should note the following regarding support for Unicode in the next major version of Omnis Studio.

❑ **Unicode only**

The next major release of Omnis Studio will be Unicode-only, that is, there will not be a separate Unicode and non-Unicode version of Omnis

❑ **Unicode version of Omnis Studio for Linux**

There will be a Beta version of Studio 4.3 Unicode for Linux, available after Studio 4.3 is released

Data file conversion

The Unicode version of Omnis Studio 4.3 performs a full conversion of Omnis data files to Unicode, as described above. Bearing in mind that the next major release of Omnis Studio will be for Unicode only, we suggest that you convert your Omnis data files to Unicode using the Unicode version of Omnis Studio 4.3, and report any problems to us as soon as possible, so that any issues can be resolved before the Unicode-only version of Omnis Studio is released.

WARNING: Again, we strongly urge you to make a secure backup of your Omnis data files before opening and/or converting them in the Unicode version of Studio 4.3. You should make a copy of your data and test the Unicode conversion in Studio 4.3 and report any problems to us.

For developers using Unicode Studio 4.3, please check the results of full conversion carefully, and do not discard your backup of the non-Unicode data file until you are satisfied that the data has converted successfully. You could perform some regression tests on your application and data – you should normally do this with a new version of Studio, but when converting to Unicode Omnis, and converting your data files, you need to be especially sensitive to possible data file and indexing issues.

Handling Char & Binary data under Unicode

You cannot concatenate a Character variable to a Binary in the Unicode version of Omnis Studio. The correct method is to use \$readfile to read the file into a Binary variable, and then parse the binary variable. Assigning Character to Binary and vice-versa is likely to cause problems, especially in the Unicode version, and should be avoided.

Omnis VCS

Method Inspector

The Method Inspector is a new tool in the Omnis VCS that allows you to examine the methods in a class, and look at the code in each method. This saves you having to Check Out or Copy Out a class or build the library if you want to check what methods are contained in the class or what the latest code looks like.

To open the Method Inspector, you can select the class and choose the 'Method Inspector' hyperlink option in the VCS Browser, or Right-click on the class and select the option from the context menu.

On the first tab of the Method Inspector window, there is a list of all the methods within the class. If the class has a superclass, the superclass methods are also shown. They are displayed in blue to differentiate them as inherited methods.

If the selected class is a window, report, remote form, menu or toolbar, there is another checkbox displayed at the bottom of the list to allow you view field methods. By default this option is not selected except for menu and toolbar classes.

The other method properties, are also displayed in this tab: description, execute on client and web service method. There is also a context menu option to allow you to view the path to the method within the class – this is of most use if you are looking at field methods as you can see the exact path to the method.

If you double-click a method, the code for the method is displayed on the second tab. Should the method be inherited, there is a context menu option to allow you to view the superclass code.

Scheduled Builds

The Omnis VCS now allows you to build out projects at a specific time and date, available under the 'Scheduled Build' option in the Omnis VCS browser.

To support scheduled builds, your VCS repository requires a minor update. When opening a repository, the Omnis VCS will prompt you if it finds the repository is not in Studio 4.3 format.

Note: If you are updating from Studio 4.1 or an earlier version, there is a 2 stage upgrade. The first will involve running the Update VCS library (UpdateVCS.lbs) to bring the repository into Studio 4.2 format. This library is in the Convert folder of the main Studio folder. The second stage will occur automatically when you attempt to logon to the VCS.

Scheduled Builds are accessible from the 'Scheduled Build' hyperlink. When you select this link, the Build Manager will open containing a new button called 'Schedule...', which opens the 'Schedule Build Process' window.

The Schedule Build window allows you to control the frequency of the builds and set the time that you want the build to run. The following options are available:

- Never
- Daily
- Weekly
- Monthly

The best time to schedule the build is when there is no one else logged on to the VCS. You must ensure that Omnis Studio is running at the time of the build and that you are logged on to the correct VCS repository. If you do not have Studio open at the time of the build schedule, when you next log on during that day, the VCS will prompt you that you have missed the scheduled build and ask if you want to run it then. For example, if you have the build set to run at 7am on a Monday, but do not start Omnis until 8am, Omnis will remind you about the build that you missed.

You can choose to label the build either at the time you schedule the build or when the build runs. Note that if you select 'Label at Build' and there is already a label with that name, the VCS will redefine the label with the current state of the classes. This may or may not be what you require, so caution should be exercised, particularly if you are setting the build to run every day and therefore you may lose the ability to rebuild to a specific point in the project development at a later point in time. If you select 'Label Now', the VCS will prompt you if there is already a label defined and give you the chance to confirm that you want the label redefined.

If you have selected a path to save them into, the build notes will be saved to a file prefixed 'AutomaticBuildNotes_' within that path. By default the path is the BuildFolder within the main Studio folder.

You should use the 'Build Manager' window to define the projects you want to build and how you want them to be built using the normal options that are available. When you click the Finish button these preferences will be saved and the timer started according to the schedule defined.

Superclass names

A new preference has been added to the Omnis VCS to allow classes to retain the library name as part of the superclass name.

Previously the library name was stripped out at checkout or build time. This meant that if you had 'LibraryA' with a class called 'classA' which had a superclass 'LibraryB.ClassSuper' and there was also a 'ClassSuper' in 'LibraryA', Omnis would switch the superclass to 'LibraryA.ClassSuper' whenever the class was checked out or built.

To enable the preference, go the Checkout tab on the VCS Options dialog and check the box "Maintain superclass library name". The preference is unchecked by default to ensure backward compatibility.

Graph2

The Graph2 component contains a number of enhancements which are described in the revised Omnis Graph2 manual (Omnis_Graph2.pdf). The new features include:

- ❑ New gauge and meter type graphs
- ❑ a new property called \$imagesearchpath to specify the folder where Graph2 will search for images (which can be added to data points in charts)

There is a new example library showing how the Graph2 component works, which is located under the Examples link in the Welcome window when you first launch Omnis – you can open this window by clicking on the New Users button in the main Omnis toolbar.

Web Client

Safari for Windows

The Netscape-type Web Client plug-in released with Studio 4.3 supports the Safari browser under Windows XP/Vista. Note that this is included in the single Windows Web Client installer; see below. Under Windows Vista, the Web Client must be installed into a folder to which the user has full permissions.

MIME type

There is a new MIME type for the Safari for Windows browser.

```
type='application/omnis-plugin'
```

So, you may want to add the new Safari MIME type to your HTML files to support Safari on Windows. See below for info about the omniswebclient.js file containing the new type.

Javascript file

There is a new omniswebclient.js file in the HTML folder under the main Omnis folder which supports the new Windows Safari browser. The new sections are:

```
function browserIsSafari()  
{  
    return checkUserAgent("safari");  
}  
...  
if (navigator.platform == "Win32" && browserIsSafari())  
    document.write("<center><embed type='application/omnis-plugin'  
    name=\"rccl\" width=\"")
```

Consequently, the HTML files created for you automatically in Studio 4.3 (when you press Ctrl-T to preview your remote form) will provide support for Safari under Windows. You may need to update your own Web Client based pages to support Safari on Windows.

Windows Web Client installer

There is now only one Web Client installer for Windows XP/Vista that supports all the most popular web browsers, including Internet Explorer, Firefox, Safari, Opera, and all other Netscape compatible browsers. Both the ActiveX plug-in (required for IE) and the Netscape-type plug-in (required for all other browsers) still exist but they are installed into the same folder and the appropriate one is used when a Web Client based web page is displayed. The new Windows installer is called omweb.exe, and installs the Web Client plug-ins in a user defined directory (a default directory is supplied, as below).

The default destination provided in the Web Client installer under Windows NT/XP is:

```
C:\webclient
```

Under Windows Vista, the default destination is:

```
C:\Users\
```

Under Vista, the Omnis Web Client ActiveX is considered to be a low-integrity component, and should therefore be installed in the LocalLow folder which allows write access to such low-integrity components. If the Web Client plug-in is installed elsewhere on Vista, then unless the web site containing your Web Client page is listed in the trusted sites setting of IE, performance is likely to be poor, or the plug-in may not run at all.

The location of the Web client plug-in specified during installation is written to the Windows environment variable MOZ_PLUGIN_PATH to allow the plug-in to work for Firefox. Likewise, a registry key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Mozilla\Omnis Firefox\extensions
```

is added (or overwritten) to make the plug-in work in Safari.

Apache plug-ins

The Apache web server plug-ins mod_omnis.so (for Omnis web client server) and mod_ows.so (for Omnis Web Services server) now support Apache 2.2. The new plug-ins will not work with previous versions of Apache.

Linux Web Client

Under Linux, the Web Client now requires the omnisgnm.so file. This file is updated automatically by the Web Client if required.

Printing on the Web Client

Note this is not a new web component, the following information was omitted from previous versions of the documentation.

The Formpri web component lets you print a report on the client machine. The Printing component can send an Omnis report, which was previously printed to memory on the Omnis Server, to either the Screen, Preview, or Printer on the client machine (note the Printer destination is not supported on Unix clients). This allows you to create all types of report on the client that would not be possible with HTML alone. For example, in a shopping application you could generate a graphically rich order confirmation and print it to the screen or preview; the user could then print the order to their printer connected to the client. Note that images with a high definition may lose their high definition when printed to the memory device on the server and may therefore suffer a loss of quality when printed on the client.

To print a report to the client you first need to create a method on the Omnis Server that prints your report to the kDevMemory printing device. Next you need to assign the resulting binary report data to the Formpri component on the remote form, and assign a destination to the \$action property.

You can direct the report to the Screen, Preview, or Printer device by assigning the appropriate constant to the \$action property; note this is a runtime property only. When this property is assigned it sends the current report data to the specified destination on the client machine. The report data must have been previously assigned to \$reportdata. \$action can be assigned one of the following constants:

Constant	Description
kFPriNone	The report is hidden.
kFPriSendToPreview	The report is shown on screen in Page Preview mode.
kFPriSendToPrinter	The report is printed to the current local or network Printer; the user will be prompted by the Job Setup dialog prior to printing.
kFPriSendToScreen	The report is shown on Screen.

The \$reportdata property takes the binary representation of an Omnis report, which was previously printed to the memory printing device; note this is a runtime property only. The following method assigns the report data in the binary variable 'myRepData' to the Printing component in the current remote form instance:

```
Do $cinst.$objs.RepField.$reportdata.$assign(myRepData)
```

The \$reporttitle property lets you assign a title to the document when printed to printer. The \$zoomon property specifies the initial zoom state for page preview mode. When this property is true, the report is shown at 100%. In addition, \$showtoolbar and \$showstatusbar let you display the toolbar and statusbar in screen or preview mode, while \$showscroll and \$showvscroll let you enable the horizontal and vertical scrollbar.

You can change the text on the Job Setup dialog, the Toolbar buttons, and the Status bar using the \$strjobdlg, \$strbutt, and \$strstatus properties, respectively. For all these properties

you can turn on the Help Tips in the Property Manager and move the mouse over the property name to show the original text. When these properties are empty, the text from the component's resources is used.

The following method sets the current print device to memory, sets the report name, sets the paper size and orientation, prints the data to the binary variable 'myRepData', and assigns the resulting data to the Printing component (called RepField) on the remote form. Note the report is sent to the screen, i.e. the client's browser.

```

; Prepare the memory device
Do $cdevice.$assign(kDevMemory)
Do $root.$prefs.$reportdataname.$assign(myRepData)
; set required paper size (optional)
Do $root.$prefs.$paper.$assign(kPaA4)
Do $root.$prefs.$orientation.$assign(kOrientPortrait)
; print the report
Set report name {myReport}
Print report
; assign report binary to report field
Do $cinst.$objs.RepField.$reportdata.$assign(myRepData)
; print report to printer on client machine
Do $cinst.$objs.RepField.$action.$assign(kFPriSendToScreen)

```

Web Data Grid

Note this is not a new web component, the following information was omitted from previous versions of the documentation.

The Web Data Grid component (Formgrid) provides a tabular display of a list into which the user can enter data, modifying the list contents. You supply the list using an instance variable (or row variable column) specified in the \$dataname property.

When you add a data grid to a remote form from the Component Store, the grid has a fixed vertical column, and a fixed horizontal row, which are never available for data entry at runtime. These are filled with the color kColor3DFace. The remaining cells are cells that can be available for data entry. The fixed row along the top of the grid is populated using the property \$columnnames; this is a comma separated list of column headings. The remainder of the grid, including the fixed column, is populated using the list data. There are properties on the appearance tab of the property manager, \$fixedrow and \$fixedcol, which specify if the fixed row or column is visible. The default behavior used to populate columns not in the fixed row, is that list column N is displayed in grid column N, where column 1 is the left-most column of the grid, including the fixed column if it is visible.

The following properties give you finer control over the appearance of the data grid.

Property	Description
\$autosize	A Boolean, if kTrue the grid resizes according to its contents
\$canresizecolumns	A Boolean, if kTrue the user can resize the grid columns
\$designcols	The number of columns the grid displays in design mode. If \$userdefined is kTrue, this is also the number of columns the grid displays at runtime. If \$userdefined is kFalse, the number of columns displayed at runtime is the number of columns in the list variable specified by \$dataname.
\$booleanstrings	contains the strings to display for False and True Boolean values, False first, separated by a comma, e.g. False,True
\$boolascheck	if kTrue booleans are drawn as check boxes

Column Specific Properties

There is a Column tab in the Property Manager that allows you to set various column-specific properties for web data grids. To select a column, you can either set the \$currentcolumn property, or you can click on the column in the data grid object; the current column is highlighted in red. The column specific properties are:

Property	Description
\$columndatacol	This specifies the list column number (in the list specified by \$dataname) which will be used to populate this column. You can use this to override the default behavior described above.
\$columnenabled	Set this property to kFalse if you want to disable user entry into this column. If the current column is column 1, and column 1 is the fixed column, this property has no affect.
\$columnforecolor	The color of the cells in the column. If the current column is column 1, and column 1 is the fixed column, this property has no affect.
\$columnjst	The justification of text in the column.
\$columnpicklist	The name of a list (which must be an instance variable) used for a pick list. See \$columnstype below for details.
\$columntextcolor	The color of text drawn in the column. If the current column is column 1, and column 1 is the fixed column, this property has no affect.
\$columnstype	The column type, a constant: kDataGridAutoData, kDataGridIcon, kDataGridComboPicker, kDataGridDroplistPicker, see below
\$columnwidth	The width of the column in pixels. Note that you can also

Property	Description
	resize columns by clicking on the column separator on the design window, and dragging.
\$columnheadercolor	The color of the column header.
\$columnheadertextcolor	The text color of the column header.

The following column types (\$columnstype) are available:

- kDataGridAutoData**
 the column is populated directly from the list variable column, and displays data according to its data type.
- kDataGridIcon**
 the column displays an icon. In this case the column data must be numeric, and also the icon pages required must be specified in the \$iconpages property of the remote form class. If the current column is column 1, and column 1 is the fixed column, this column type has no effect.
- kDataGridComboPicker**
 the column displays text data, and has an associated combo box populated using the \$columnpicklist property, which can be used to enter data. If the current column is column 1, and column 1 is the fixed column, this column type has no effect.
- kDataGridDroplistPicker**
 the column data must be numeric. The value of the column is a line number in the \$columnpicklist list. A drop list containing the entries in \$columnpicklist can be used to enter data. If the current column is column 1, and column 1 is the fixed column, this column type has no effect. Note that you can disable the behavior caused by the properties accessible via the column tab in the property manager, by setting the property \$userdefined (on the General tab) to kFalse. Changing the value of \$userdefined does not affect any previously stored column-specific values.

Web Data Grid Events

The data grid generates a number of events in addition to the general events `evAfter` and `evBefore`:

Event	Description
<code>evClick</code>	Sent after the user has clicked in a drop list or combo box which is active in a cell. It has one event parameter, <code>pLineNumber</code> , which is the number of the line in the combo box or drop list on which the user clicked.
<code>evFixedCellClicked</code>	Sent when the user clicks on a fixed cell (a cell which can never be enterable, for example the a column header). This has two event parameters: <code>pHorzCell</code> (the column number of the clicked cell) and <code>pVertCell</code> (the row number of the clicked cell).
<code>evGridCellChanged</code>	Sent when a cell has been changed. This has three event parameters. <code>pData</code> (the data value of the new cell), <code>pHorzCell</code> (the column number of the new cell) and <code>pVertCell</code> (the row number of the new cell).
<code>evGridCellChanging</code>	Sent when the current cell is about to change. This has three event parameters. <code>pData</code> (the new data value), <code>pHorzCell</code> (the column number of the cell) and <code>pVertCell</code> (the row number of the cell).

Note: unless you use client method execution, the grid cannot receive both `evGridCellChanging` and `evGridCellChanged` events.

Linux

Printing under Linux

In the Linux version of Omnis Studio 4.3 the PostScript printing device has been replaced by a PDF printing device, called `PrintPDF`, that creates a PDF file from your Omnis reports. `PrintPDF` can print Unicode characters.

Note the PostScript device is no longer provided for any platform.

The prerequisites for `PrintPDF` are:

- Python**
normally part of the Linux installation
- Reportlab 2.0**
installed as part of the Omnis RPM installation
- PIL**, Python imaging library (version 1.1.4 or higher)
You will need to download and install PIL from <http://www.pythonware.com/products/pil>

PrintPDF creates a folder called `printpdf` in the Omnis tree which it uses for temporary PNG files and Python scripts.

PrintPDF device parameters

The PrintPDF device is selected by default in the Omnis Print destination dialog, rather than the Printer destination or Postscript file (now removed). It has a number of parameters:

Output command (text)

See the automatic output parameter for details.

Automatic output (Boolean)

If checked, the output command parameter must be completed. PrintPDF generates the PDF in a temporary file, and then spools the file using the command, after adding the temporary file pathname to it:

If the command does not include the token `%f`, PrintPDF appends a space, followed by the pathname of the temporary file to the command text.

If the command contains the token `%f`, then PrintPDF replaces all instances of `%f` with the temporary file pathname less the `.pdf` extension. This allows the command, for example, to look like this:

```
../xpdf-3.01p12-linux/pdftops -noshrink -nocenter -level3
%f.pdf;lp -d myprinter %f.ps;rm %f.ps
```

In this example, PrintPDF uses an alternative install of XPDF, to the default on the system - the command runs the `pdftops` filter, spools the PostScript, and then deletes the Postscript file. Note that PrintPDF deletes the temporary PDF file.

The parameters are:

`%f` = the pdf path (without the `.pdf` extension)

`%d` = printer name

`%o` = printer options

If automatic output is not checked, PrintPDF generates the PDF in the pathname identified by the Omnis preference `$prefs.$reportfile`. The pathname must identify a file with the `.pdf` extension.

Disable transparent background objects (Boolean)

The various PDF to PS filters do not all handle transparency correctly.

You can disable transparency with this option. In practice, most reports can use this option checked. `ext` always overlays the background objects. XPDF 3.0.1 makes a much better job of transparency

Keep Python script and PNGs (Boolean)

If checked, various temporary files are not deleted after generating the PDF. This may be useful for debugging and support purposes.

Printer management (CUPS)

Printing in the Linux version of Omnis Studio now uses the Common UNIX Printing System (CUPS) to manage printer destinations, print jobs and their settings. According to the CUPS web site (www.cups.org): “CUPS provides a portable printing layer for UNIX based operating systems, ... and is the standard printing system in Mac OS X and most Linux distributions. CUPS uses the Internet Printing Protocol (IPP) as the basis for managing print jobs and queues and adds network printer browsing and PostScript Printer Description (PPD) based printing options to support real-world printing.”

Print dialogs

Using CUPS means that Omnis now has better print setup, job setup and printer destination dialogs that allow you to manage printer settings.

Python Imaging Library

The Python Imaging Library (PIL) no longer needs to be in the python sub-folder of the main Omnis folder. If you need python-imaging it must be installed with python, typically from the Linux operating system CDs.

Linux Fonts

Omnis Studio 4.3 uses the standard Fontconfig/Xft mechanism to manage fonts, which is supported by most other applications on Linux. This provides good support for scalable fonts, including anti-aliasing and support for Unicode character sets (although a Unicode version of Omnis for Linux is, at present, not available), both in the IDE and runtime environments. Omnis Studio no longer has a font cache.

As a result, the set of fonts available to Studio under Linux has completely changed, as have their names, as follows:

Font	Has become
"-adobe-helvetica-"	"Luxi Sans" (if Luxi fonts are available) or "DejaVu Sans"
"-adobe-courier-"	"Luxi Mono" (if Luxi fonts are available) or "DejaVu Sans Mono"
"-adobe-times-"	"Luxi Serif" (if Luxi fonts are available) or "DejaVu Serif".

All other font names remain unchanged. Font name conversion occurs for both the Unix font tables and font names used in \$fontname property of the standard #STYLES system table. If a font is a TrueType font an icon is displayed against its name in the Change Font table dialog.

Note that the system substitutes a default font where the font name does not match one available, so text will be displayed even if its font does not exist.

Report fonts

The report fonts listed in #UXRFONTS comprise TrueType fonts only (those whose file has a .ttf extension). These are the most straightforward fonts to use with Reportlab, especially for Unicode.

Menu bar height/font

There are two new properties in the setup section of `omnisxi.ini`, which default as follows:

MenuBarHeight=22

MenuBarHeight is the height of the bar in pixels at 75dpi. It is scaled if the resolution is different.

MenuBarPointSize=11

MenuBarPointSize is the point size used for the menu bar font.

When you start up Omnis the defaults are written to `omnisxi.ini`.

Gnome & KDE environments

Omnis Studio 4.3 supports the Gnome & KDE environments; no other environments are supported. There has been a significant number of changes and enhancements made in the Linux version of Omnis Studio to improve the user interface under Gnome and KDE. In particular, native file dialogs, print dialogs and message boxes are now displayed and colors have also been improved.

To support these improvements, a number of files have been added to the root directory of Omnis Studio under Linux. There are two new shared libraries, and a new executable: `omnisgnm.so`, `omniskde.so` and `omkdedlg`. So when you deploy Omnis in the Runtime environment or using the Web Client, these files must be included. For example, the Web Client always uses the Gnome file `omnisgnm.so`, since the supported web browsers on Linux are GTK (Gnome)-based. All these files are supplied in the Studio development and Runtime environments by default.

In addition to the above changes, window ordering has been improved. Therefore, under KDE if you attempt to bring a window to the front that should not be allowed, Omnis will prevent this by restoring the window ordering. As part of this change, window decorations (close box, etc) will appear and disappear according to whether they can be used, i.e. a window that cannot be clicked on does not have a close box. Note that minimize boxes and shade command are always present. Using function keys or the window menu to bring a window to the front now works correctly.

File names

File names are now treated as UTF-8 which means that path names containing accented characters now look correct in the KDE and Gnome file managers. This may cause issues with file names with accents created with previous versions of Omnis under Linux, but this change was essential to make Omnis consistent with other applications and Gnome/KDE.

Save window setup

The Save window setup option now saves and restores window positions correctly, although if you switch between Gnome and KDE you may notice that the main Omnis window is not the same height in both environments.

Tooltips, Menus & droplists

Tooltips and the list part of droplists and popup menus are now displayed correctly.

Events on inactive objects

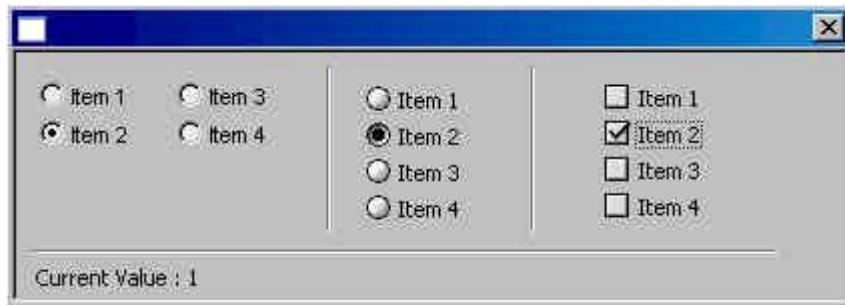
Under the Linux version of Omnis, events do not pass through inactive objects on Linux as they do on Windows and Mac OS X. This is because in X11, mouse messages propagate through the chain of parent windows, rather than to the next window(s) in the Z order.

Window Programming

Radio Button Groups

Radio Button Groups are now available for Window classes, *in addition to the existing* Radio button window field. Radio Button Groups for window classes are similar to the Radio Group field currently available for remote forms (for the Web Client).

The existing Radio button fields are available as separate fields, whereby you have to create a number of individual radio button fields to make a group of buttons. However, the new Radio Button Group field displays a number of radio buttons in a group but is handled as a *single field control*, making it easier to create and layout a group of Radio buttons.



The variable you associate with a Radio Button Group is specified in \$dataname and should be numeric. Clicking a radio button sets the value of the field specified in \$dataname to zero for the first button in the group, one for the second button, two for the third button, and so on.

The Radio Group field has the following properties:

Property	Description
\$text	Comma separated list of values specifying the text of the buttons, e.g. Item 1,Item 2,Item 3,Item 4
\$buttoncnt	The number of buttons in the group
\$columncnt	The number of columns to split the group of buttons, e.g. using values \$buttoncnt=4 and \$columncnt=2 gives you a group of 2x2 buttons
\$horizontal	if kTrue the Radio group will display the controls in a horizontal order; if kFalse the group is displayed vertically
\$iconid	0 causes the control to display a default radio button according the current OS; alternatively this can be set to a multistate icon, e.g. id = 1642 size = k16x16; see above

The Radio Group control supports the evClick event, so when the control is selected or a value altered using the keyboard (using tab and space bar to select a button), an evClick event is generated. The value of the variable specified in \$dataname is updated when a button is selected.

Tab/Page panes

The method \$listobjects() has been added to tab/page panes for window instances. The syntax \$listobjects([iPaneNumber]) returns a list of objects contained within the specified pane, including all foreground and background window objects. If iPaneNumber is omitted, the list contains information about the objects on *all* panes in the Tab/Page pane field. The list has three columns: *object name* (empty for background objects), *ident* of the object, and *pane number*.

If you mark an object as “all panes”, it will be included in the list regardless of the pane number specified.

Headed lists and Data grids

The properties \$columnnames, \$columnwidths, and \$columnsvisible can now be edited in the Property Manager by clicking on the droplist button for the property.

For user defined data grids (on windows and remote forms), the column tab has a new property, \$columnname. This represents another way of accessing the value stored in \$columnnames. There is also a new design-mode only property on the column tab called \$movecolumn. This allows you to re-order the column definitions for the grid: entering a number in this property moves the current column and re-orders the other columns accordingly.

The \$calculation property for headed lists can now be edited via a droplist button, which opens a window. The window has two tabs, one for calculations that are of the form *con(x,sep,y,sep,...)* where *sep* is either kTab or chr(9), and one for other calculations. The

first tab has a grid, allowing you to enter the calculation for each column, the second has an entry field. The Notation Helper and Catalog both work in this window.

Tree lists

Three new constants have been added to the tree list field to replace and enhance the last parameter of the \$movenode() method.

```
$movenode( rItem, rItemMoveAfter, bMoveNode )
```

where

kMoveNodeAfterDst	moves rItem, after rItemMoveAfter - nodes all operate on the same level
kMoveNodeToDstFirstChild	moves rItem into rItemMoveAfter and sets rItem as the first child node
kMoveNodeToDstLastChild	moves rItem into rItemMoveAfter and sets rItem as the last child node

Previously, the third parameter of \$movenode() let you move the specified node to the same level as the current node or as a child but the node was always placed at the end of child list; the new constants let you specify either the top or bottom of the child list. This enhancement should not effect existing code as the old boolean maps to kMoveNodeAfterDst & kMoveNodeToDstLastChild.

Refreshing window instances

There is a new property called \$norefresh for window instances. When set to kTrue screen updates are disabled and the window is not refreshed. You can use this property to improve performance, for example when setting a large number of exceptions for a complex grid. Setting the \$norefresh property to kFalse will enable screen refreshing.

Losing property values

Certain notation properties affect other properties when they are assigned, for example, assigning \$calculated to kFalse clears \$text for the field. Therefore, if the \$calculated property is set within a reversible block, and the state of \$calculated is reversed, the value in the \$text property is not reinstated. Such relationships between properties are not supported by the reversible block mechanism. If you wish to maintain the value of a property that may get cleared during notation execution, you should store the value in your own variable and assign the value to the property at runtime.

Format Strings

On page 119 of the *Omnis Programming* manual incorrectly states that 'A' truncates the value. This should be 'X'. Therefore, in the examples on this page, X should be used in place of A.

Input Masks

You can use various characters as placeholders when specifying an input mask (`$inputmask` property), including the ‘n’ character to represent any alphanumeric character. However, if you want the same characters to appear as literals or as part of a string, you have to prefix the character(s) with a ‘\’ (backslash). Therefore the example, ‘Enter digit #’ provided on page 123 of the Omnis Programming manual should be ‘E\nter digit #’.

Calendar Component

The Calendar component, available for window and remote form classes, stops working after the end of the year 2037 (December 31, 2037). Unfortunately, this is the end of time as far as the APIs we use are concerned! This may be corrected in a future version.

Reports

Using MS Sans Serif with Report Fields

When using MS Sans Serif font for report fields with `$align` set to `kCenterJst`, you may find that text in the field is not correctly centered. All other fonts are working correctly with center justified report fields.

When Omnis prints text on a report, it calculates the width of the text on the screen, and the width for the printer, and uses the larger of the two measurements to place the text in the field. This is usually fine, as there is normally only a small difference between the two widths. However, there is a large difference between the two widths for MS Sans Serif, resulting in the non-centered text.

ODBC

ODBC Administration

The following additional session methods allow the ODBC DAM to be used to add, modify and remove ODBC Data Source Names (DSNs) as well as to retrieve and modify information about ODBC drivers and general ODBC administration attributes.

All methods are implemented by the Session Object and return a Boolean value to indicate successful operation. Errors generated by these methods are returned via the session object's \$nativeerrorcode and \$nativeerrortext properties.

Session Methods

Method	Description
\$getdrivers()	<p>SessObj.\$getdrivers(IResult) retrieves a list of all ODBC drivers installed on the system. IResult is populated with the list of drivers installed and is defined with the following character columns:</p> <p>DriverName The alternate driver name (i.e. the descriptive name)</p> <p>Version The version string reported by the driver</p> <p>CompanyName The Company name embedded within the driver file</p> <p>FileName The physical path and file name of the driver</p> <p>CompanyName is only obtainable for Win32 and will return as empty for other platforms. DriverName is obtainable directly from the driver only for Win32. Other platforms require each driver to be loaded and called in order to obtain the version string. Hence, there will be a commensurate delay when calling this method.</p> <p>Example: Do sessObj.\$getdrivers(iDriverList) Returns #F</p>
\$getdatasources()	<p>SessObj.\$getdatasources(IResult, kDSNMode) retrieves a list of ODBC DSNs of type specified by kDSNMode which should be passed as either kODBCSystemDSN or kODBCUserDSN. \$getdatasources() does not support File DSNs (see below). On return, IResult is defined with two character columns:</p> <p>DSNName The User assigned name for the data source</p> <p>Driver The alternate name of the driver associated with the data source</p> <p>Example: Do sessObj.\$getdatasources(iDSNList, kODBCSystemDSN) Returns #F</p>
\$getinfo()	<p>\$getinfo(IResult, cDSNName, kDSNMode) retrieves the information defined for the specified data source or driver as a list of keyword-value pairs. kDSNMode should be passed as either kODBCSystemDSN, kODBCUserDSN or kODBCDriverInfo.</p>

Method	Description				
	<p>\$getinfo() does not support File DSNs for which standard FileOps methods can be used to read/modify as required.</p> <p>On return IResult is defined with the following character columns:</p> <p>KeyWord The name of the DSN/driver attribute Value The value of the DSN/driver attribute</p> <p>Example: Do sessObj.\$getinfo(iDSNinfo,'myDsn',kODBCUserDSN) Returns #F</p>				
\$setinfo()	<p>\$setinfo(cDSNName, kDSNMode, IData) writes the information contained in IData to the specified Data source or Driver key in the system information. IData should be defined with Keyword and Value columns as returned by \$getinfo().</p> <p>If kDSNMode is kODBCDriverInfo, this has the effect of modifying system information for the specified driver. cDSNName should contain the descriptive name of the ODBC Driver as opposed to the physical file name.</p> <p>If kDSNMode is kODBCSystemDSN or kODBCUserDSN, this has the effect of modifying the specified data source.</p> <p>\$setinfo() does not register a new data source or driver although it will write data to the DSN as though it already exists. To properly create a data source, use the \$configdsn() method instead. To properly register a driver, you should refer to the vendor's installation program.</p> <p>Example: Do sessObj.\$setinfo('myDsn',kODBCUserDSN,iDSNinfo) Returns #F</p>				
\$configdsn()	<p>\$configdsn(kDSNMode, kRequestType, cDriverName, lAttributes) allows the specified datasource to be created, modified or removed. kDSNMode should be either kODBCSystemDSN or kODBCUserDSN. \$configdsn() does not support configuration of File DSNs- for which an alternative strategy is provided. kRequestType should be passed as either kODBCAddDSN, kODBCModifyDSN or kODBCRemoveDSN. cDriverName should correspond with the descriptive name of the driver (i.e. not the physical file name).</p> <p>lAttributes should be defined with two character columns and is used to pass keyword-value pairs to the driver manager sufficient to perform the required action. Usually this involves adding a single line to the list to identify the DSN to be created/modified/removed, e.g.</p> <table data-bbox="625 1433 862 1496"> <tr> <td>KeyWord</td> <td>Value</td> </tr> <tr> <td>DSN</td> <td>dsnname</td> </tr> </table> <p>but can also include other keywords that are allowed by the driver.</p>	KeyWord	Value	DSN	dsnname
KeyWord	Value				
DSN	dsnname				

Method	Description												
	<p>When \$uselogonprompt is set to kODBCPromptNever, this prevents \$configdsn() from opening setup dialogues. The DSN is created/modified silently using values read from the attribute list instead.</p> <p>Example: Do sessObj.\$configdsn(kODBCUserDSN,kODBCAddDSN,'SQL Server',lAttribList) Returns #F</p>												
\$getoption()	<p>\$getoption(kOption, cAttribute) allows the value of an ODBC configuration attribute to be retrieved.</p> <p>kOption should be passed as one of the following constants:</p> <table border="0"> <tr> <td>kODBCTrace</td> <td>Requests the TRACE on/off flag</td> </tr> <tr> <td>kODBCTraceLib</td> <td>Requests the name and path to the ODBC trace library</td> </tr> <tr> <td>kODBCTraceFile</td> <td>Requests the name and path to the ODBC trace log</td> </tr> <tr> <td>kODBCFileDSNDir</td> <td>Requests the default directory containing file DSNs</td> </tr> <tr> <td>kODBCPerfMon</td> <td>Requests the Performance monitoring on/off flag</td> </tr> <tr> <td>kODBCRetryWait</td> <td>Requests the connection pool RetryWait timeout</td> </tr> </table> <p>On return, cAttribute contains the value of the requested option as a character string.</p> <p>Example: Do sessObj.\$getoption(kODBCFileDSNDir,iFileDSNDir) Returns #F</p>	kODBCTrace	Requests the TRACE on/off flag	kODBCTraceLib	Requests the name and path to the ODBC trace library	kODBCTraceFile	Requests the name and path to the ODBC trace log	kODBCFileDSNDir	Requests the default directory containing file DSNs	kODBCPerfMon	Requests the Performance monitoring on/off flag	kODBCRetryWait	Requests the connection pool RetryWait timeout
kODBCTrace	Requests the TRACE on/off flag												
kODBCTraceLib	Requests the name and path to the ODBC trace library												
kODBCTraceFile	Requests the name and path to the ODBC trace log												
kODBCFileDSNDir	Requests the default directory containing file DSNs												
kODBCPerfMon	Requests the Performance monitoring on/off flag												
kODBCRetryWait	Requests the connection pool RetryWait timeout												
\$setoption()	<p>\$setoption(kOption, cAttribute) allows the value of an ODBC configuration attribute to be modified.</p> <p>kOption should be either kODBCTrace, kODBCTraceLib, kODBCTraceFile, kODBCFileDSNDir, kODBCPerfMon or kODBCRetryWait. cAttribute should contain a character string representing the new value for the specified configuration option.</p> <p>Example: Do sessObj.\$setoption(kODBCTraceFile,iTraceFile) Returns #F</p>												

Session Properties

Property	Object	Description
\$uselogonprompt	Session	\$uselogonprompt previously accepted values of kTrue and kFalse to indicate whether the driver should prompt for missing or incomplete logon information. This property is now also used to force the ODBC Administrator library to display a configuration dialogue when connecting to File DSNs. \$uselogonprompt now accepts constant values of: kODBCPromptNever (0) kODBCPromptComplete (1) kODBCPromptAlways (2)

Configuration of File DSNs

Using \$getoption() to retrieve the default directory for file DSNs allows the FileOps component to be used together with other 4GL techniques in configuring File datasources.

To create a File DSN, the user prompts for the new filename (the DSN Name) and uses FileOps to create the new file. One or more KeyWord-value pairs should also be written to the file, e.g. DRIVER=drivername, the minimum requirement for a File DSN. To complete the setup of the new DSN, you should additionally follow the procedure for modifying/testing the File DSN.

To modify or test a File DSN, use the following procedure:

- Set \$usefiledsn to kTrue
- Set \$uselogonprompt to kODBCPromptAlways
- Execute \$logon() with the name of the File DSN as the hostname.

Under Win32, this prompts the ODBC Administrator library to display the driver specific logon dialog which prompts for information necessary to make the connection. If the DAM is successful in logging on, the Administrator library writes the additional information back to the File DSN, hence modifying the data source.

To remove a File DSN, you should use the FileOps component to manually delete the specified filename. An appropriate confirmation dialog should be displayed prior to deletion.

Mac OS X and Linux Considerations

Under Unix, the DAM locates user DSN information (odbc.ini) using the value of the ODBCINI environment variable if it is set. If ODBCINI is not set, the DAM attempts to use the ".odbc.ini" (hidden file) in your user's home directory or failing that, it defaults to "/Library/ODBC/odbc.ini".

The DAM locates system DSN information using the value of the ODBCYSINI environment variable if it is set. If ODBCYSINI is not set, the DAM attempts to locate the system driver information using the value of ODBCINSTINI instead. If ODBCYSINI is

set, this location is also assumed for the location of the driver information file (odbcinst.ini). Note that if set, ODBC_SYSINI should identify a folder only, not a file name. The ODBC Driver manager used on your system must support the necessary API calls in order to perform certain administration functions (editing and modifying DSN information, for example). If the required API calls are missing, these functions will not be available.

Oracle8 DAM

There are new session properties and methods in the Oracle8 DAM (DAMORA8).

Session Properties

Property	Description
\$internalcharmapping	This property; originally added for diagnostic purposes, allows Omnis internal character mapping to be turned off if required. This property has no effect under Mac OS X where mapping to and from the Omnis character set is not performed- since Omnis uses the MacRoman character set internally. Conversion to the Omnis character set occurs before any custom character map '.out' file is applied and after any character map '.in' file is applied. Thus turning internal character mapping off allows you to test that your custom character mapping tables are working as expected. For cross-platform applications, this property should remain enabled, otherwise your Windows and Mac applications may behave differently.
\$useprecision	When set to kTrue, Oracle NUMBER types which would otherwise map to Omnis Number floating dp are mapped to Omnis Number 0..14dp. When the precision of the column exceeds 15 or when the column has no precision/scale, the mapping defaults back to floating dp. Affects data type mapping as well as output from \$columns() and \$results(). Default value is kFalse for backwards compatibility.
\$emptyasnull	When kTrue, empty Omnis strings are inserted as NULL. When kFalse, they are inserted as chr(0). \$emptyasnull defaults to kTrue. (See also \$nullasempty).

Session Methods

Method	Description
\$proxyas()	<p>SessionObj.\$proxyas(cUsername [,cPassword, lRoles]). Allows the supplied user to connect to Oracle through the current connection, which must already be logged-on. The session then acquires the roles and privileges associated with that user. An additional list of roles to be used with the proxy session can also be supplied if required. The list should consist of a single column of type Character. The password should be supplied if the proxy requires authentication by password.</p> <p>\$proxyas() can be called repeatedly with different usernames if required, in which case the current proxy is implicitly terminated before the new proxy is established.</p>
\$sendproxy()	<p>SessionObj.\$sendproxy(). Explicitly tests for and terminates a proxy session if one exists, returning the session roles and privileges back to those of the primary connection.</p>

SQL Programming

Session Pools

Errors encountered when creating session pools are now returned via #ERRCODE and #ERRTEXT.

Further Enhancements

This section includes smaller enhancements to Omnis Studio, as well as small updates and corrections to the main Omnis manual (PDFs).

Screen resolution

The following changes have been made to the design DPI library preference, \$designdpi, which specifies the screen resolution in 'dots per inch' (DPI) of the current library, depending on the platform the library is running under. The preference stores a comma separated list of values in the following order: Windows DPI, Mac OS X DPI, Linux DPI, and the value for the current operating system is used as appropriate. If you assign a new value to this property, the new value does not take effect until you close and re-open the library.

This preference allows for the fact that 96 DPI on Windows is equivalent to 72 on Mac OS X and 75 DPI on Linux; these are the standard screen resolutions (measured in DPI) at which Omnis Studio operates when creating or opening Omnis libraries.

Scaling now occurs using the display DPI for the current platform, and the DPI value in \$designdpi for the platform. Existing libraries may need the value of \$designdpi to be set.

Omnis will adapt the screen resolution of the library according to the current platform, that is, a library with design DPI set to 96, will display correctly to 96 on Windows, 72dpi on Mac OS X, and 75dpi on Linux.

One further effect of this change is that the FontOps methods \$wintextheight() and \$wintextwidth() now scale their result from the current display DPI, to \$designdpi. This then allows the results of these methods to be used correctly in conjunction with notation coordinates (\$left, \$top etc.), since these values all use the design DPI as their units.

Java

Java options

The Java preferences \$javainitialheap, \$javamaxheap and \$javathreadstack have been removed from the Java group of preferences within \$root.\$prefs. In addition the \$javaother property has been renamed to \$javaoptions. If you want to use the original properties, you can add the following parameters to the \$javaoptions property, assuming \$usejavaoptions is set to true.

Original heap size	To specify the original heap size for the JVM add "-Xms <heapsize>" to the \$javaoptions property
Maximum heap size	To specify the maximum heap size for the JVM add "-Xmx <maxheapsize>" to the \$javaoptions property
Stack size	To specify the stack size for the JVM add "-Xss <stacksize>" to the \$javaoptions property

Object parameters

Java reflection requires that parameter objects exist on the main classpath. To ensure that objects or object references can be passed to a Java method, place the path to the jar file in the CLASSPATH variable.

Method lines

Methods can now have up to 1024 lines; the previous limit was 512 lines per method.

Help

IDE Help

Due to changes in Omnis Studio 4.3 to accommodate Windows Vista, the location of the Help folder available in the development version of Omnis has changed. The function of the IDE Help is more-or-less unchanged, and this has no effect in the Runtime version of Omnis.

The IDE help directory, omnis, is now in a directory called idehelp. On Windows Vista, idehelp is in the Program directory, as opposed to the normal help directory for application help, which is in the Data directory (see earlier in this manual for an explanation of the Program and Data directories).

When you open the help window in the Omnis IDE, help is opened in read-only mode. The context menu for a displayed help page no longer allows you to rebuild the indexes for the help file, while the help on help command and the change default browser options have been removed.

Help folder name

The \$helpfoldername specifies the location of the Help files. The property can be set to a full path where the help files are stored which can be on a server location. This allows multiple users to share the same help files and therefore allows for easier maintenance. The help files do not have to be stored in the Omnis/Help folder (as stated on page 577 in the *Omnis Programming* manual).

External Components

There is a new external component callback, FILEgetOmnisProgramFolder, and a new method getOmnisProgramFolder in EXTfile, which returns the path to the Program directory, which is now required as part of the Omnis installation under Windows Vista (see earlier in this manual for an explanation of the Program directory). On all platforms except Vista, this returns the same path as FILEgetOmnisFolder.

System key codes

The System key codes (pSystemKey) for the evKey event are as follows:

0	Letter/Number	26	End
1...12	F1...F12	27	Tab
17	Up Arrow	28	Return
18	Down Arrow	29	Enter
19	Left Arrow	30	Backspace
20	Right Arrow	32	Esc
21	Page Up	34	Delete
22	Page Down	35	Insert
25	Home	53	Context Menu

Item References

Item References can be greater than 255 characters, but such long references cannot be dragged successfully from the Property Manager or the Notation Inspector to the Method Editor.

Find & Replace

Note the Omnis Find & Replace supports the standard notation for regular expressions, such as described at http://en.wikipedia.org/wiki/Regular_expressions.

Remote Studio

The 'Client Method' example on page 546 of the *Omnis Programming* manual should read:

```

; Client Method
Do rsapp.$createobject()
Calculate rsapp.$omnislibrary as "RSTUDIO"
Calculate rsapp.$omnisclass as "rtrStudio"
Calculate rsapp.$omnisserver as "5112"
Calculate rsapp.$serverurl as "http://127.0.0.1"
Calculate rsapp.$serverscript as "/cgi-bin/nph-omniscgi.exe"
Do rsapp.$open()    ; this line was missing
Do rsapp.$executeNoParams("rtnVarList") Returns result
Calculate rcnt as result.$getrowcount()
Calculate ccnt as result.$getcolumncount()
For row from 1 to rcnt step 1
    For col from 1 to ccnt step 1
        ; To confirm with Visual Basic, offsets are zero based
        Calculate parml as result.$getelement(row-1,col-1)
        Send to trace log {[parml.$getchar()]}
    End For
End For

```

Functions

The following are new or updated functions.

isclear()

`isclear(expression|fieldname)`

Returns true if the *expression* or *field* has value NULL, zero (for numeric data types only), or empty. *isclear()* can execute in client methods.

isleopard()

Returns true when running on Mac OS X 10.5 (Leopard); it returns false for all other platforms and earlier versions of Mac OS X. *isleopard()* can execute in client methods.

Note you can use *isleopard()* to reliably detect Leopard without checking `sys(6)` and/or `sys(7)`.

isvista()

Returns true when running on Windows Vista; it returns false for all other platforms and earlier versions of Windows. *isvista()* can execute in client methods.

Note you can use *isvista()* to reliably detect Vista without checking `sys(6)` and/or `sys(7)`.

mouseover()

`kMHorzCell` has been extended, so that `mouseover(kMHorzCell)` now works for data and string grids in the fat client.

rxpos()

`rxpos(rx,string,ignorecase,words,mchlen)`

Returns the position of characters matching the regular expression (regex) *rx* in *string*, or zero for no match; the function sets *mchlen* to length of match; *ignorecase* and *words* are Booleans, that return true for ignore case or only match whole words. *rxpos()* can execute in client methods.

The expression in *rx* can be formatted using the standard notation for regular expressions, such as described at http://en.wikipedia.org/wiki/Regular_expressions.

sys(6)

Returns N on Windows Vista, and X on Mac OS X 10.5 (Leopard).

sys(7)

Returns 6.0 on Windows Vista, and 10.5.0 on Mac OS X 10.5 (Leopard).

sys(115)

Returns the path of the Data directory on Windows Vista. You can use this to get or create the path of a writeable file which is stored in the Data directory under Vista.

sys(202)

Returns the command line parameters passed to Omnis, for Win32 and Linux only.

Therefore, if you execute:

```
omnis /param1=val1 /param2=val2
```

```
sys(202) returns "/param1=val1 /param2=val2"
```

This was omitted from previous versions of the documentation.

sys(215)

Returns the path of the Program directory on Windows Vista. On all other platforms, it returns the same value as `sys(115)`.

Lotus Notes

NSF List Attachments command

The following commands have been added to or updated in the Lotus Notes external component.

The command *NSF List Attachments* has been added which returns a list of attachments for the current note. The returned list has two columns: Name (column1-Character) and ID (Column 2-Numeric). The ID can be used to reference specific attachments, such as in the amended *NSF Unpack File* command, as described below.

The *NSF Unpack File* command now supports an optional ID parameter to specify a particular attachment from a list of attachments returned in the *NSF List Attachments* command.