# Chapter 1—The Omnis Environment

The tutorial introduced you to some parts of the Omnis IDE and you will have used the common tools in Omnis, such as the Studio Browser and the Catalog. This chapter goes into more detail about these tools and others in the Omnis Studio IDE. Some of the tools and development features in Omnis, such as the Omnis VCS and using the SQL Browser, are explained in greater detail in their own respective chapters later in this manual.

The Omnis development environment contains all the tools you need to build enterprise, web, and mobile applications in a cross-platform, multi-developer environment.

When you start Omnis Studio you will see the **Developer Hub** (shown as Hub) in the Studio Browser which provides information and videos to help you get started in Omnis Studio: if the Studio Browser is not visible, press **F2** or click on the **Browser** button on the main Omnis toolbar.

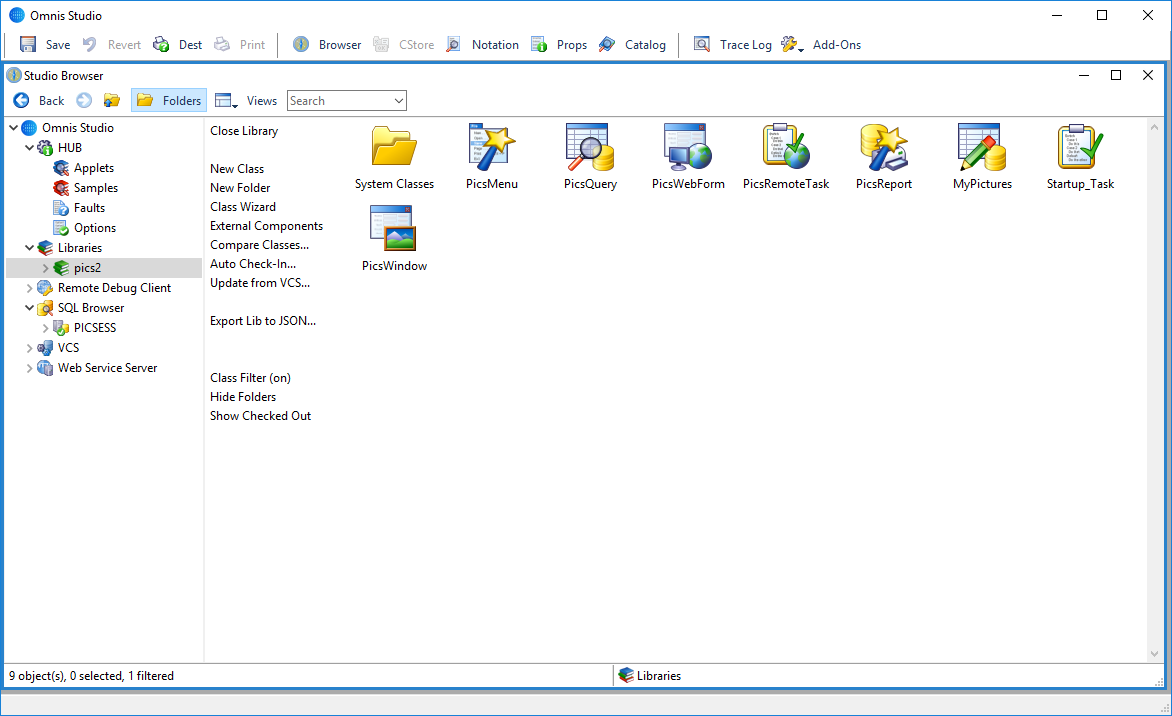


Note: Some editions of Omnis Studio may not provide access to all the development tools or web/mobile development features, such as the Omnis VCS or the DAMs in the SQL Browser.

## Studio Browser

The *Studio Browser* is the main window in Omnis Studio for developing your applications and managing server database sessions. The hierarchical folder list down the left side contains the Hub, a list of all open Libraries, the SQL Browser containing any open database sessions, and the Omnis Version Control System (VCS): note that the SQL Browser, VCS, and Web Service Server are not available in some editions of Omnis Studio.

The toolbar at the top of the Studio Browser window lets you navigate around the Omnis environment and change its view options. For example, the **Folders** button option lets you hide and show the hierarchical folder list. You can right-click on the Browser to change its view options (Large or Small icons, or Details view), and the **Save Window Setup** option lets you save the current settings.

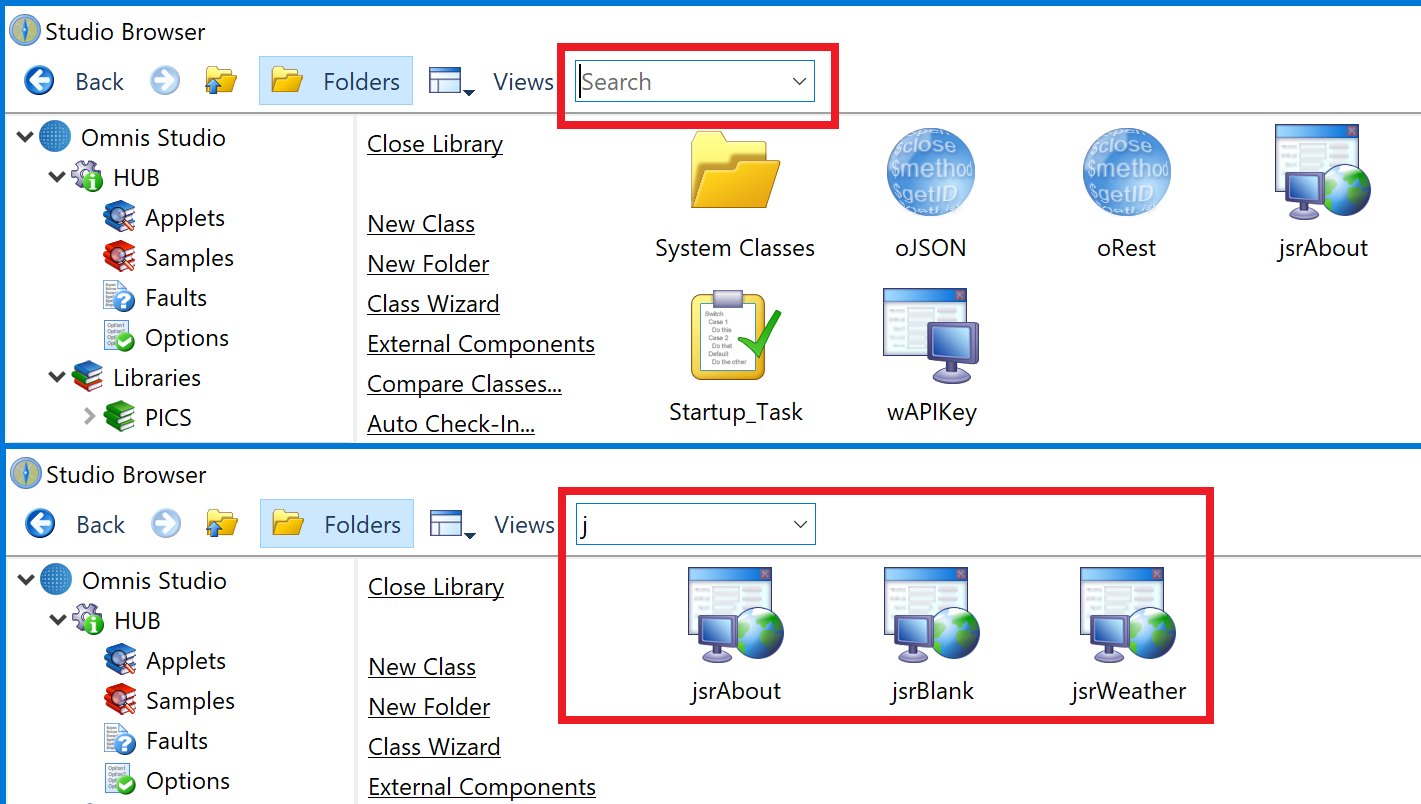
  
The Studio Browser showing the contents of a library in Large Icons view.

The right-hand pane displays a list of objects or files for the current folder, e.g. it displays all the classes in a library when a library is selected in the Folders list on the left. The list of **Options** (they look like hyperlinks) will change depending on the object currently selected in the Folders list or the file browser on the right and will always provide options and shortcuts to perform actions on the current object. For example, when a library is selected on the left, its classes are shown in the list on the right, and the hyperlink options relate to classes, such as creating a **New Class** or **Class Wizard,** or the **Class Filter** allows you to show or hide certain types of classes.

### Search Filter

The Studio Browser has a Search box that allows you to filter the objects displayed in the library or class list allowing you to find objects more easily. The Search filter is available for most of views in the Studio Browser, including Libraries, Classes, SQL sessions, VCS projects, and various parts of the Hub including the Sample apps and Faults.

To search for an item, navigate to the correct view in the Studio Browser, type the first few character of the item(s) you are looking for, and the list will instantly redraw, displaying only those items that *start with* the character(s) you typed. For example, in the class list for a library, you could type “j” to find all the classes starting with the letter j.



In most cases the search string you enter is used to find items that start with those characters, except that in the Fault list, under the Hub option, the search string is used to find items that *contain* the search string. The Search box has a dropdown list that stores the last few searches you typed, which you can select from with the pointer.

#### Copy Class Name

You can copy the name of a selected class to clipboard by pressing Ctrl-N or via the Copy Name option on the Class Browser context menu: for multiple selected classes the names are copied in a list.

### Developer Hub

The Developer **Hub** in the Studio Browser provides useful information for developers, such as the status of the most recent reported and fixed faults, together with information, videos and tips for new Omnis developers. For new users, the Hub is the default option, but you can change this and the contents of the Studio Browser under the **Options** setting.

#### Hub

The Hub option itself contains information and videos for new and existing developers, including ‘How to’ videos about useful functions in Omnis Studio.

#### Applets and Samples

The **Applets** option provides a number of example Omnis applications that show the full capabilities of Omnis Studio for building web and mobile applications. You can open each of the examples in a web browser (when you select an example it is opened in your browser automatically), and you can examine the code in the associated library under the **Libraries** option in the Studio Browser.

The **Samples** option provides a number of sample Omnis libraries demonstrating specific components or programming techniques in Omnis. Once you have opened the sample library, you can examine its classes and underlying code under the **Libraries** option in the Studio Browser.

You can use the Omnis libraries under the Applets and Samples option as templates for your own libraries, or you can reuse individual classes or the Omnis code within the libraries.

#### Faults

The **Faults** option provides information about the latest *Reported* and *Fixed* faults in Omnis Studio – this is real-time information so you can check the most recent faults. If you have reported a fault in Omnis Studio you can check its status here.

#### Options

The **Options** setting allows you to configure the behavior, contents and appearance of the Studio Browser. The Options window is split into different tab sections:

**Browser tab**

1. **Show These Tools**  
   Under the ‘Show These Tools’ option you can specify which tools (nodes) are displayed in the Studio Browser. By default, the SQL Browser and VCS are shown, but the Omnis Datafiles browser is not displayed. If you wish to open and browse Omnis datafiles you will need to enable this option.
2. **Default Browser Node**The ‘Default Browser Node’ option lets you specify whether the Hub or Libraries node is displayed by default when Omnis starts up.
3. **New Libraries**The ‘Use AppBuilder for New Libraries’ option lets you enable or disable the App Builder which will be opened when the ‘New Library’ option is selected in the Studio Browser: see elsewhere in this manual for information about the App Builder. When the App Builder is disabled the New Library option will prompt you to create a new blank library with no data classes, SQL login setup, or remote forms.

**Themes tab**

1. **Appearance Themes**The Appearance option allows you to change the theme used in the Omnis Studio IDE. Together with the Default theme, there are a number of other themes from which you can choose. See the “Color Themes and Appearance” section later in this manual. You can create a custom theme, as well as import or export themes from this window.
2. **Window Frame Theme**(Windows only) The ‘Window Frame Theme’ option allows you to set the color theme or style for the frame edge of windows and forms. The options are Default, Windows 7, 8, or 10.

**Proxy Server tab**

The ‘Use proxy server’ option allows you to access the faults information if use a proxy for outgoing requests. Check the option and enter a hostname and service or port for your proxy server.

### Libraries in the Studio Browser

The **Libraries** option under the Studio Browser lets you create a new library (the main file where you store all the classes in your application) or open an existing library. The **New Library** option will open the App Builder which allows you to create a new library from one of several templates: see the *App Builder* section for more details. The **Open Library** option will open an existing Omnis library file. The **New Lib from JSON** option allows you to import a library from a JSON representation of an Omnis library. See the [Libraries and Classes](02libsandclasses.html) chapter for more information about these options.

#### Library Conversion

When you open a library created in a previous version of Omnis Studio it will be converted and can no longer be opened in the old version of Omnis Studio – *THE LIBRARY CONVERSION PROCESS IS IRREVERSIBLE*. *Therefore in all cases, YOU SHOULD MAKE A SECURE BACKUP of all old Omnis libraries and datafiles BEFORE OPENING THEM in the latest version of Omnis Studio.*

#### Recent Classes

Any classes that you have opened recently are listed in the Recent Classes popup list displayed at the bottom of the Studio Browser (displayed when the Libraries option is selected in the Studio Browser). To open a class, you click on the list and select the class name. The class will be displayed in the appropriate class editor.

You can open the method editor or Interface Manager for a class (that is, a class that can contain methods) from the Recent Classes list in the Studio Browser. To open the method editor for a class, click on the Recent Classes list and hold down the **Shift** key and select the class, or to open the Interface Manager for a class, hold down the **Control/Command** key while selecting the class in the Recent classes list.

#### Library Folders

You can create folders in a library so you can organize the classes within each library. Folders are in fact Omnis classes, but for display only (they are visible during development only). Folders do not perform any function in your library, other than organizing your classes, and they are ignored at runtime.

To hide/show library folders within the Browser, select your library and click on the ‘Library Folders (on/off)’ option, or alternatively, right-click on the white space of the Studio Browser (when any library is selected) and select the ‘Library Folders’ option from the context menu. You can save the state of this option using the Save Window Setup option for the Studio Browser.

#### Hiding and Showing Classes

When the contents of a library is displayed in the Studio Browser, you can hide and show different types of class by pressing the Shift and Control/Command keys plus the appropriate letter key. For example, pressing Shift+Ctrl+W causes only windows to be displayed, or Shift-Ctrl/Cmnd-A shows all classes. The following keyboard shortcuts are available:

|  |  |  |  |
| --- | --- | --- | --- |
| Shift +  Ctrl/Cmnd + | Displays... | Shift +  Ctrl/Cmnd + | Displays... |
| A | all classes | J | tasks only |
| W | windows only | C | code classes only |
| F | files only | T | toolbars only |
| R | reports only | S | schemas only |
| L | searches only | U | query classes only |
| M | menus only | B | tables only |
| O | object classes only | Y | system tables only |
| I | Remote forms only | K | Remote tasks only |

The above key presses activate the Class Filter, but you can set the Class Filter manually. To open the Class Filter, press the Ctrl+Shift+E key or click on the Class Filter (on/off) option in the Studio Browser.

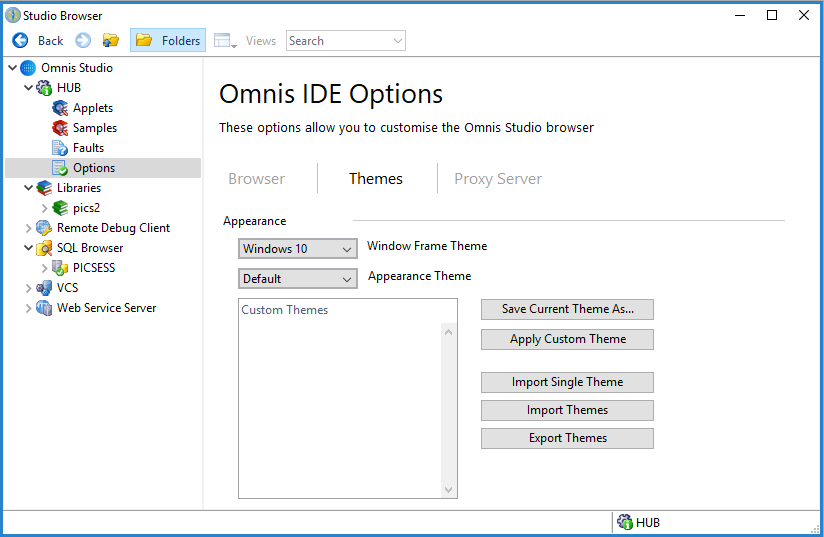
#### Diacritical Character input in the IDE

The Studio IDE will provide the diacritical character popup to allow you to enter diacritical characters wherever text entry is required. For example, you can enter diacritical characters in comments in the Method Editor or into a label on a form. To disable the feature for the IDE, remove or rename the Keyboard folder which is in the local folder.

## Color Themes and Appearance

The colors used throughout the Omnis Studio IDE are stored as a *theme* which can be changed under the **Options>>Themes** setting in the Hub in the Studio Browser: each theme contains a list of color settings for the objects and tools in Omnis. Color themes are available on Windows and macOS only, but not for Linux.

There are a number of themes to choose from in the Hub, listed under the ‘Appearance Theme’ droplist, including the ‘Default’ theme which is intended to match the colors used on different platforms supported in Omnis. You can change individual colors or settings within a theme (under the $appearance property, see below), and in this case, your modified theme will be saved as a ‘Custom’ theme alongside those provided. The themes are stored as configuration files in the ‘Studio’ folder under the main Omnis folder.



### Exporting & Importing Custom Themes

You can create multiple custom themes, and you can export and import themes. There is a list of the custom themes currently installed (located in the folder /studio/themes/custom) underneath the themes droplist.

To create a custom theme, press the "Save Current Theme As" button. Once saved, the name you give the theme will then appear in the list of custom themes.

If you are setting a custom theme, you will need to select it first in the list and then press the "Apply Custom Theme" button, since you need to be able to select a custom theme without applying the theme when exporting.

To export custom themes, select the required themes in the list and press the "Export Themes" button. This allows you to select a folder to copy the themes into.

You can also import either a single theme or a folder of themes. Once imported they are copied to the /studio/themes/custom folder and will appear in the list.

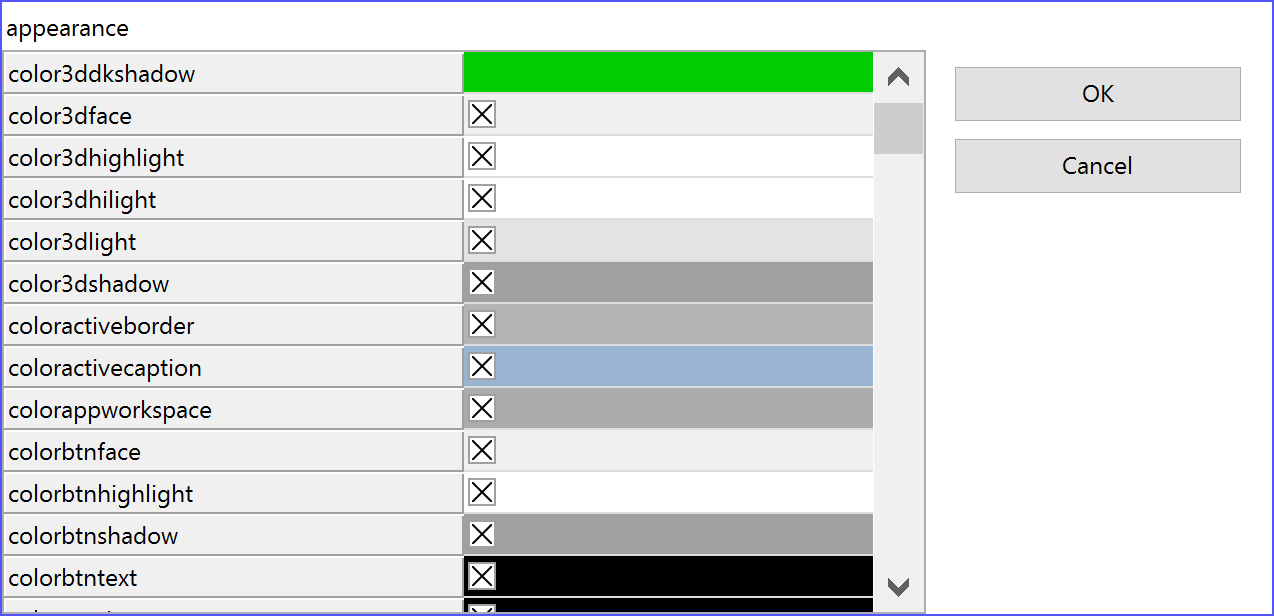
### Window Frame Themes

For Windows only, you can also change the theme used for the outside edge or frame of windows used in the Studio IDE and your own windows, which can be set to match the style of the window frame displayed on various Windows platforms. These are shown under the ‘Window Frame Theme’ droplist.

### Appearance Property

The current theme in the Omnis Studio IDE is stored in an Omnis preference called $appearance, which can be edited in the Property Manager. To edit this property, click on ‘Omnis Studio’ in the Studio Browser, click on Prefs, then select the Appearance tab in the Property Manager and click on the droplist next to the $appearance property.

Note that when editing $appearance in the Property Manager, the default colors may not always draw correctly since the editor grid itself uses the configured colors, and therefore if a color is set to the default setting, there is a check mark icon (an X) to the left of the color.



In addition, there is an Omnis preference called $windowoptions that stores the current Window Frame theme which you can also edit in the Property Manager: this is described at the end of this section.

#### Searching Colors & Themes

There is a search field on the $appearance preference dialog to help you find colors. As you type into the search box, Omnis will highlight any matching lines in the Property Manager and scroll to the first match. Tabbing from the search field sets the focus in the grid to the first match.

### Appearance and Theme Files

The color and appearance settings used in the Omnis Studio IDE, and displayed in $appearance in the Property Manager, are stored in a configuration file called **‘appearance.json’,** which is located in the Studio folder in the main Omnis folder. This file contains the current color theme settings in $appearance which will either be the default theme, or one of the other themes provided, or a custom theme.

The Default and other themes (Blue, Green, etc) are stored in separate JSON files in the ‘studio/themes’ folder. As you change the theme setting under the Options setting in the Hub, the appropriate JSON theme file is written to the appearance.json file in the Studio folder.

When Omnis loads, it will copy the appropriate theme file into /studio/appearance.json. If you alter one of the theme colors using $appearance in the Property Manager, Omnis will recognize that the default or one of the built-in themes has changed and therefore will create a custom theme 'appthemecustom.json' in the themes folder. In this case, when Omnis restarts, it will load the Custom theme and it will add it to the droplist in the Options setting in the Hub.

When a theme has been changed, if you then try to switch to another theme using the Hub, you will receive a Yes/No message asking you to confirm if you wish to overwrite the custom changes.

### Appearance Configuration File Contents

As well as choosing a theme in the Options setting in the Hub and changing colors and settings via $appearance in the Property Manager, you may like to edit the configuration files – we recommend that you do not change the default theme file in the studio/themes folder, but you can make a copy of it and edit that. The color number values in the configuration files are 32 bit integers: if the value is less than zero, it is a representation of a color constant in the form 0x8nnnnnnn where nnnnnnn identifies the constant, otherwise, the value is RGB - 8 hex digits: 00bbggrr e.g. 0x0000ff00 is full green.

There is a file called appearance.desc.en.json in the 'local' folder under the main Omnis folder which contains a description of all the items in the $appearance property and appearance.json file: the descriptions in this file are used as helptips in the Property Manager when viewing the items in the $appearance property.

### Appearance Subgroups

The colors listed in the appearance.json file, and the associated theme files, are grouped into subgroups, to make it easier to locate appearance settings related to specific areas or controls in Omnis, and include the following subgroups:

"checkRadio"

"compareTool"

"dropList"

"edit"

"generalButton"

"groupBox"

"header"

"IDEgeneral"

"IDEmethodEditor"

"IDEmethodSyntax"

"list"

"menu"

"pagePreview"

"pushButton"

"scrollbar"

"system"

"tabPane"

"toolbar"

"tooltip"

"tree"

The subgroups prefixed with IDE refer to colors in the IDE only, such as the Method Editor, so any changes you make to colors in these groups are only visible in the IDE, not the Runtime. All other theme subgroups affect colors in the IDE and the Runtime version of Omnis Studio. When you edit the $appearance preference in the Property Manager you will see the subgroups, so to edit colors in the Method Editor syntax you can open the “IDEmethodSyntax” group.

The following settings are available in the appearance.json file which correspond to the settings under the $appearance property in the Property Manager.

#### Tab Pane

**colortab, colortabhot, colortabselected, colortabdisabled, colortabborder**Colors used for tab panes with $tabstyle set to kDefaultPanes. If these are all set to kColorDefault, then Omnis uses the system theme: otherwise, Omnis imitates the system theme and draws its imitated tab pane using these colors.

**colortabselectedtextmacos** controls the color of the text on a selected tab on macOS only.

**iconidtabpaneleftbutton, iconidtabpanerightbutton**The icon ids of the arrows drawn on tab pane scroll buttons.

#### List

**colorlistlineselectedwin, colorlistlinehotselectedwin, colorlistlinehotwin, colorlistlineunfocusedwin**These colors are used for Windows only. Used for the background of “lines” in different states, e.g. lines in headed list boxes, tree nodes etc. If these are all set to kColorDefault, then Omnis uses the system theme: otherwise, Omnis imitates the system theme and uses these colors.

#### Headed List

**colorheader, colorheaderhot, colorheadpressed, colorheaderseparator, colorheadertextwin**Used in the header of headed list boxes. If these are all set to kColorDefault, then Omnis uses the system theme: otherwise, Omnis imitates the system theme and uses these colors. colorheadertextwin lets you set the color of the header text on Windows only.

#### Pushbutton

**colorpush, colorpushborder, colorpushhot, colorpushhotborder, colorpushpressed, colorpushpressedborder, colorpushpressedtextmacos, colorpushdefault, colorpushdefaultborder, colorpushdefaulttextmacos, colorpushdisabled, colorpushdisabledborder**  
Used for system push buttons and various other controls that use the same theme. If these are all set to kColorDefault then Omnis uses the system theme: otherwise, Omnis imitates the system theme and uses these colors. The …macos properties are used for text on the button in the relevant state, on macOS only.

**colorpushdefaultflash**A Boolean which indicates if the default button is to flash (or pulse).

#### Check box and Radio button

checkradiobordermacos, checkradiounchecked, checkradiochecked, checkradiouncheckedpressed, checkradiocheckedpressed, checkradiomark, checkradiomarkhot, checkradiomarkdisabled  
Used for system checkboxes and radio buttons. If these are all set to kColorDefault Omnis uses the system theme: otherwise, Omnis imitates the system theme and uses these colors. The checkradiobordermacos property is only used on macOS, for an unchecked control: on Windows, the theme uses the mark color as the border color.

#### Tree List

**colortreeiconcollapsed, colortreeiconexpanded, colortreeiconhot**Used for the standard arrows in tree controls. If these are all set to kColorDefault Omnis uses the system theme: otherwise, Omnis imitates the system theme and uses these colors.

#### Borders

**colorctrleditborder, colorctrleditborderinsetmacos, colorctrllistborder, colorgroupboxborder**Used for the borders of controls that use a relevant kBorderCtrl… value. If the color value is kColorDefault, Omnis uses the system theme. Otherwise, Omnis imitates the system theme and draws the border using the specified color. colorctrleditborderinsetmacos is only used on macOS on retina displays - for edit controls, the border is 2 single pixel rectangles, one inside the other and this color is for the inset rectangle.

#### Group Box

**colorctrlgroupboxmacos**   
the background color of a group box on macOS, when the border of the group box is kBorderCtrlGroupBox.

#### Report

**colorreportdesignposnsectiontext**   
the color of the text on a report design mode section, for a positioning section.

#### Page and Print Preview

**colorpreviewpaper**the color of the paper in the page preview window. kColorDefault results in white: otherwise Omnis uses the specified color.

**colorpreviewfoundhighlight**the color of the window that briefly appears to indicate the location of text found when searching a print preview. kColorDefault results in red: otherwise Omnis uses the specified color.

**colorpreviewfoundtextbackground**the color of the background on which found text is drawn, when searching a print preview. kColorDefault results in yellow: otherwise Omnis uses the specified color.

**colorpreviewfoundtext**the color in which found text is drawn, when searching a print preview. kColorDefault results in red: otherwise Omnis uses the specified color.

#### Scrollbar

The colors used for scrollbars are now configurable in the Property Manager under the $appearance preference (and stored in appearance.json). The configured colors apply to all scrollbars except those on system dialogs and third party controls (e.g. OBrowser). If all colors relevant to the platform are default, then the standard system scrollbar is used.

**colorscrollbar**  
The color used to fill the scrollbar.

**colorscrollbararrowwin**  
Windows only. The color of a scrollbar arrow on an enabled scrollbar when the mouse is not over the arrow button, and when the button is not being pressed.

**colorscrollbardisabledarrowwin**  
Windows only. The color of a scrollbar arrow on a disabled scrollbar.

**colorscrollbarhotarrowwin**  
Windows only. The color of a scrollbar arrow on an enabled scrollbar when the mouse is over the arrow button, or when the button is being pressed and the mouse is not over the button.

**colorscrollbarhotbuttonwin**  
Windows only. The color of a scrollbar button on an enabled scrollbar when the mouse is over the button, or when the button is being pressed and the mouse is not over the button.

**colorscrollbarhotthumb**  
The color of the scrollbar thumb when the mouse is over the scrollbar (macOS) or over the thumb (Windows).

**colorscrollbarthumb**  
The color of the scrollbar thumb when the mouse is not over the scrollbar.

**colorscrollbartrackingarrowwin**  
Windows only. The color of a scrollbar arrow on an enabled scrollbar when the button is being pressed and the mouse is over the button.

**colorscrollbartrackingbuttonwin**  
Windows only. The color of a scrollbar button on an enabled scrollbar when the button is being pressed and the mouse is over the button.

**colorscrollbartrackingthumbwin**  
Windows only. The color of the scrollbar thumb when the thumb is being dragged.

**colorscrollbarwarmthumbwin**  
Windows only. The color of the scrollbar thumb when the mouse is over the scrollbar but not the thumb.

#### Menu and Window Menu bar

Checked menu lines on the Windows platform draw the checked indicator based on colorlistlineselectedwin unless colormenu is set to kColorDefault (in which case they draw using the system default). The color of the checkmark (when colormenu is not set to kColorDefault) is either colormenutext or colorgraytext (the latter for disabled menu items).

The hierarchical menu indicator now draws using colormenutext.

The following colors apply to Window Menu bars.

**colormenuwindowmenubarwin**  
The color of a window menu bar on the Windows platform. If set to kColorDefault, the menu bar draws using the system theme.

**colormenuwindowmenubartextwin**  
The color of text drawn on a window menu bar on on the Windows platform. If set to kColorDefault, the menu bar draws using colormenutext.

**colormenupenwin**  
The color of separator lines, and the vertical gutter line, drawn on the menu (rather than the bar) on the Windows platform. If set to kColorDefault, the lines draw using color3dlight.

#### Toolbar and Docking Area

**colortoolbarobjecthighlightwin**Windows only. The color used to draw the rectangle that highlights toolbar items. kColorDefault means Omnis will use the system theme. Otherwise Omnis imitates the system theme using this color.

**dockingareacolor** and **dockingareatextcolor**  
The color used to draw the background area of the main Omnis toolbar docking area, and the text.

#### Client Method

**clientexeccolor, inheritedcolor, nosetpropertycolor, runtimepropertycolor, setpropertycolor, toolobjselectcolor**These replace the notation in $prefs previously used to set these values. If set to kColorDefault, Omnis uses the color to which the property defaulted.

#### Chroma Coding

**bracketcolor, bracketbackcolor, commentcolor, ctrlkeywordcolor, currentblockcolor, keywordcolor, stringcolor, variablecolor**These replace the chroma coding options for the method editor (the option to edit these has been removed from the method editor modify menu). If set to kColorDefault, Omnis uses the color to which the property defaulted.

**commentstyle, ctrlkeywordstyle, currentblockstyle, keywordstyle, stringstyle, variablestyle**These replace the chroma coding options for the method editor. These are integer values representing a style (see the values of the style constants which can be ORed together, e.g. kBold has the value 1).

#### Highlight Color

**colorthemered**  
A color used where Omnis needs to make something stand out. kColorDefault maps to red.

This is used in the following places: #STYLES editor, and the Report class editor when marking the highlighted section Watch panel in method editor, to make a changed variable value Data grid ($userdefined true) to mark the current selected column.

In addition, the drag bitmap when dragging more than one line on macOS shows the count using a circle drawn in highlight color, with text in highlight text color.

#### Property Manager

**colorpropertymanager**The color of the property manager background. If left as kColorDefault, Omnis defaults to 3dface on macOS, and the toolbar background color for Windows.

### Changing and Testing Colors

For testing purposes, you can add two files to the Studio folder, which are copies of appearance.json: they are ‘appearance\_custom.json’ and ‘appearance\_default.json’. These files must have the same syntax as appearance.json. The default file should contain all colors set to kColorDefault (the initial state of appearance.json). The custom file can contain any color scheme you like. You can then use F11 to select the custom scheme, and shift-F11 to select the default scheme, while running Omnis. This behavior can be disabled by setting the config.json entry “allowSwitchAppearance” in the “ide" object to false, or alternatively by not including these additional files.

### Platform Specific Notes

**System dialogs and Menus**

System dialogs (file dialogs etc) do not use the theme colors.

Menus on macOS do not use the theme colors. On Windows, the colorlistlineselectedwin color is used to highlight menu lines.

**JavaScript Client**

The JavaScript client will use the system colors configured on the Omnis App Server running your app. Therefore if you want to use a specific theme for your deployed web and mobile apps, you need to copy your appearance.json file to the Omnis App Server.

**macOS** **and Cocoa**

The term system theme is a little loose for Cocoa. Omnis tries to match the system theme, but unlike Windows, there are not always APIs in the OS to perform the drawing (hence some system theme drawing on Cocoa is actually Omnis imitating the OS theme).

Some of the theme background colors have been rationalized (this does not apply to Carbon) as follows:

kBGThemeWindow, kBGThemeContainer and kBGThemeTabStrip now fill with kColorWindow on both platforms. Previously these used kColor3DFace on Windows.

kBGThemeTabPane now fills with the selected tab background (using either the system theme or colortabselected - the system theme applies with colortabselected is kColorDefault).

### Window Frame Appearance on Windows

In addition to the color management outlined above, you can change the appearance of the frame edge of window classes (on Windows operating systems only), which allows you to comply with the latest style for window frame edges on Windows 8, 8.1 and 10. For most purposes you can accept the default settings for the current Windows platform, but you can change the frame theme if you want.

You can change window frame themes under the **Options** setting under the **Hub** section of the Studio Browser (Windows only). You can select Default, Windows 7, 8, or 10 which allows you to view how your application will appear on different Windows platforms.

In addition there is a new property in the Omnis preferences, $windowoptions, which allows you to edit the appearance of window frames in libraries running on Windows – note this preference is only editable on the Windows platform.

#### Window Frame Configuration files

There is a file called ‘window.json’ in the Studio folder, which stores the values of the $windowoptions preference. The window.json file configures the appearance of the window frame, and also configures the operating systems for which the configured appearance will be used, including the old appearance for Windows 7, and the new configured appearance for later Windows operating systems. There are a number of theme files in the ‘studio/themes’ folder which are copied to window.json as appropriate.

#### Active Caption Colors

**useborderactivecolorfordefaultactivecaption**  
specifies the color of the active caption in Windows (stored in window.json). This is an integer with 3 possible values.

If it is zero, the behavior is as before (the active caption defaults to white if it is set tokColorDefault).

It it is one, and the active caption color is kColorDefault, then the active caption color is the same as the active border color.

If it is two (the default), then the default active caption color depends on the system setting on the accent color settings panel: "show colour on start, taskbar, action centre and title bar" - if the system setting is off, then this is equivalent to useborderactivecolorfordefaultactivecaption equal to zero - if the system setting is on, then this is equivalent to useborderactivecolorfordefaultactivecaption equal to one.

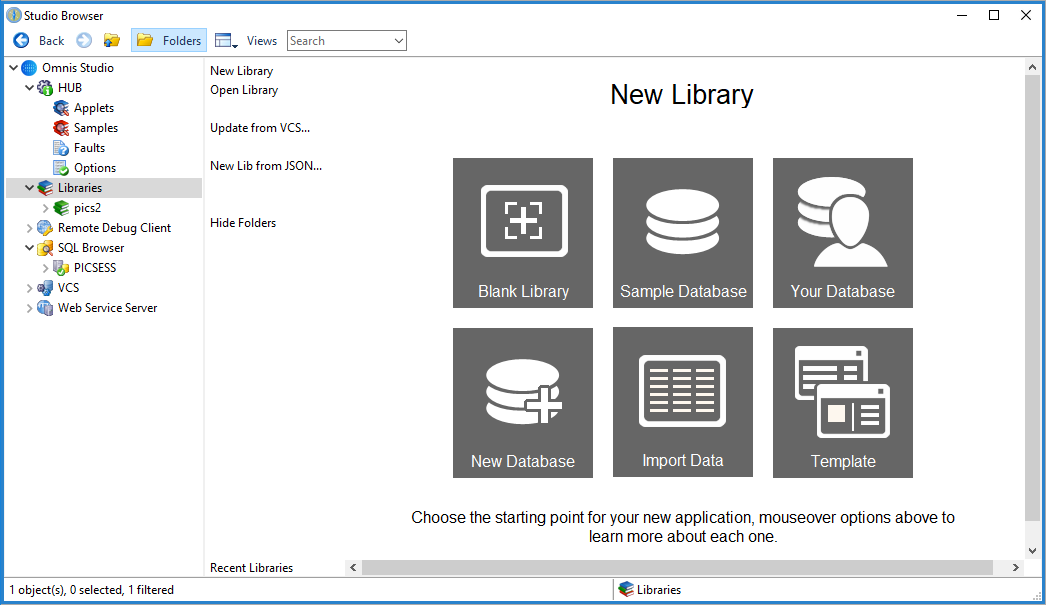
Note that this applies to both small and normal size captions, and only applies when the relevant caption colour is kColorDefault.

## App Builder

The **App Builder** allows you to create an application from a number of different options including a sample database, your own database, or by importing some data, for example, from a spreadsheet. From the initial selection, the App Builder lets you set up the tables (including primary keys), select the layout for your web and mobile forms, then you can select the color themes, branding, and the navigation in your app, and finally Omnis will build the application for you which you can test straight away in a web browser.

### Creating a New Library

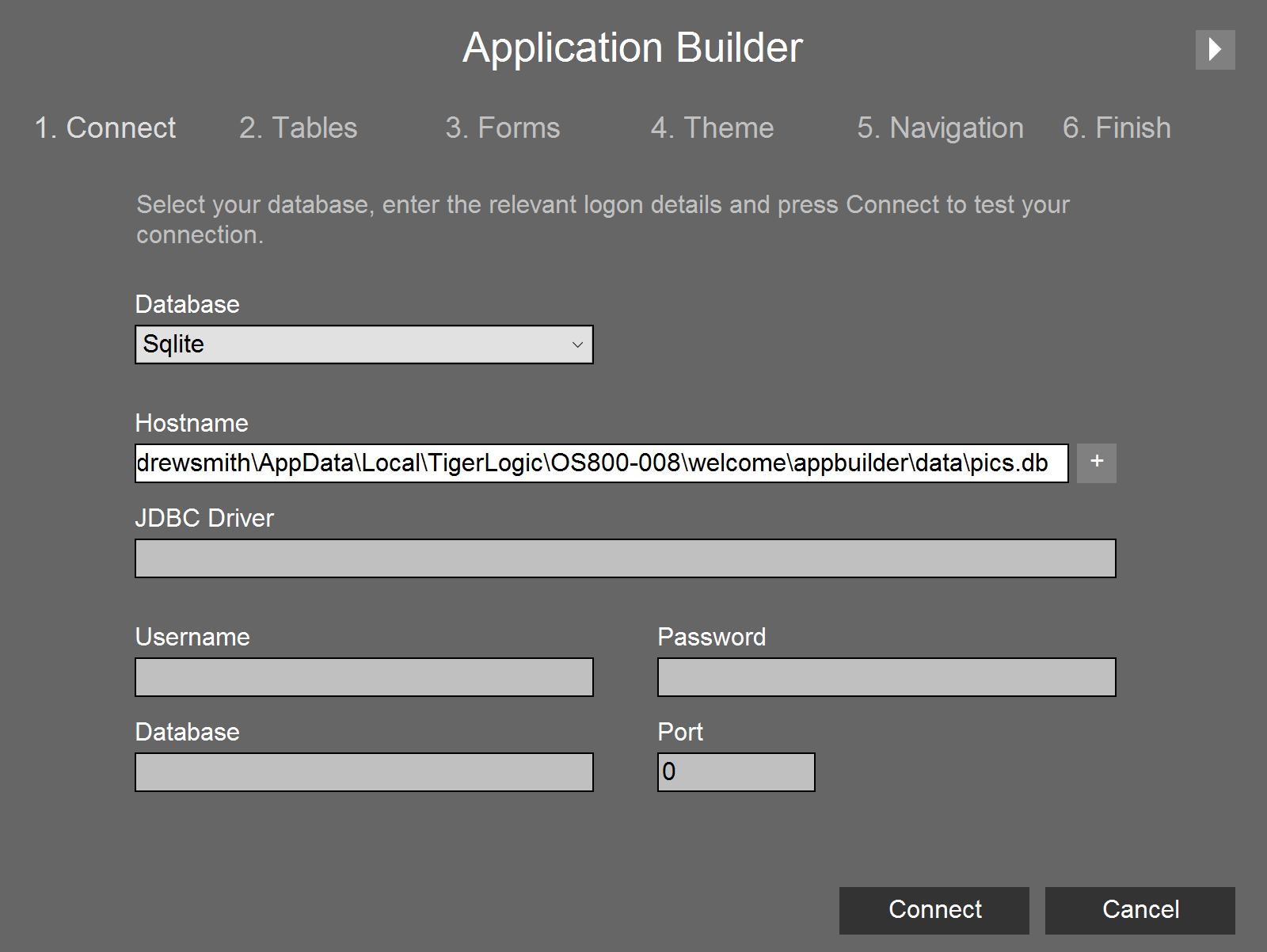
The App Builder is available when you choose the **New Library** option in the Studio Browser, and will allow new users to create apps quickly or evaluate Omnis Studio, and it will allow existing users to prototype apps and web forms quickly based on an existing database or from a template.



When you have created your application you can test it in your web browser. The library you have created is loaded into the Studio Browser under the **Libraries** option. You can examine the classes in the library and start to make modifications or add further classes.

### Creating an app from your Database

The App Builder lets you create a new app (library) based on an existing database. To do this, click on Libraries in the Studio Browser, click on New Library, select the ‘Your Database’ option and step through the process of building a library. For this option you’ll need to be able to access and logon to your database: for most types of database you’ll need the hostname, username and password, and for some other database types you might need additional information.



After logging on to your database and connecting, you then need to select the tables you want to include in your app, select the type and layout for the web forms (JavaScript remote forms), then you can select a color theme and add your company logo, then choose the navigation scheme (either a list or menu), and finally you can test the app in your web browser or open the library in the Studio IDE.

## Omnis Preferences

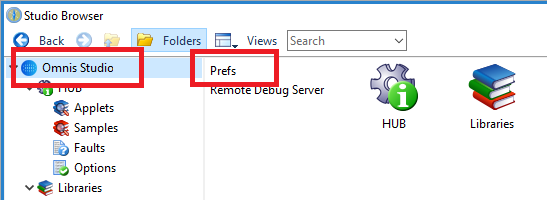
The *Omnis Preferences* or *Options* control the behavior and appearance of the Omnis Studio IDE, rather than individual libraries, and include groups of preferences for: **General** options, **Appearance, Devices, Page Setup,** and **Methods.** You can change the Omnis preferences in the Property Manager. The Omnis preferences are sometimes referred to as the “Omnis Root Preferences”, since they are properties of the Omnis Root object (referenced in the notation as $root.$prefs.<property-name>).

**To view the Omnis preferences**

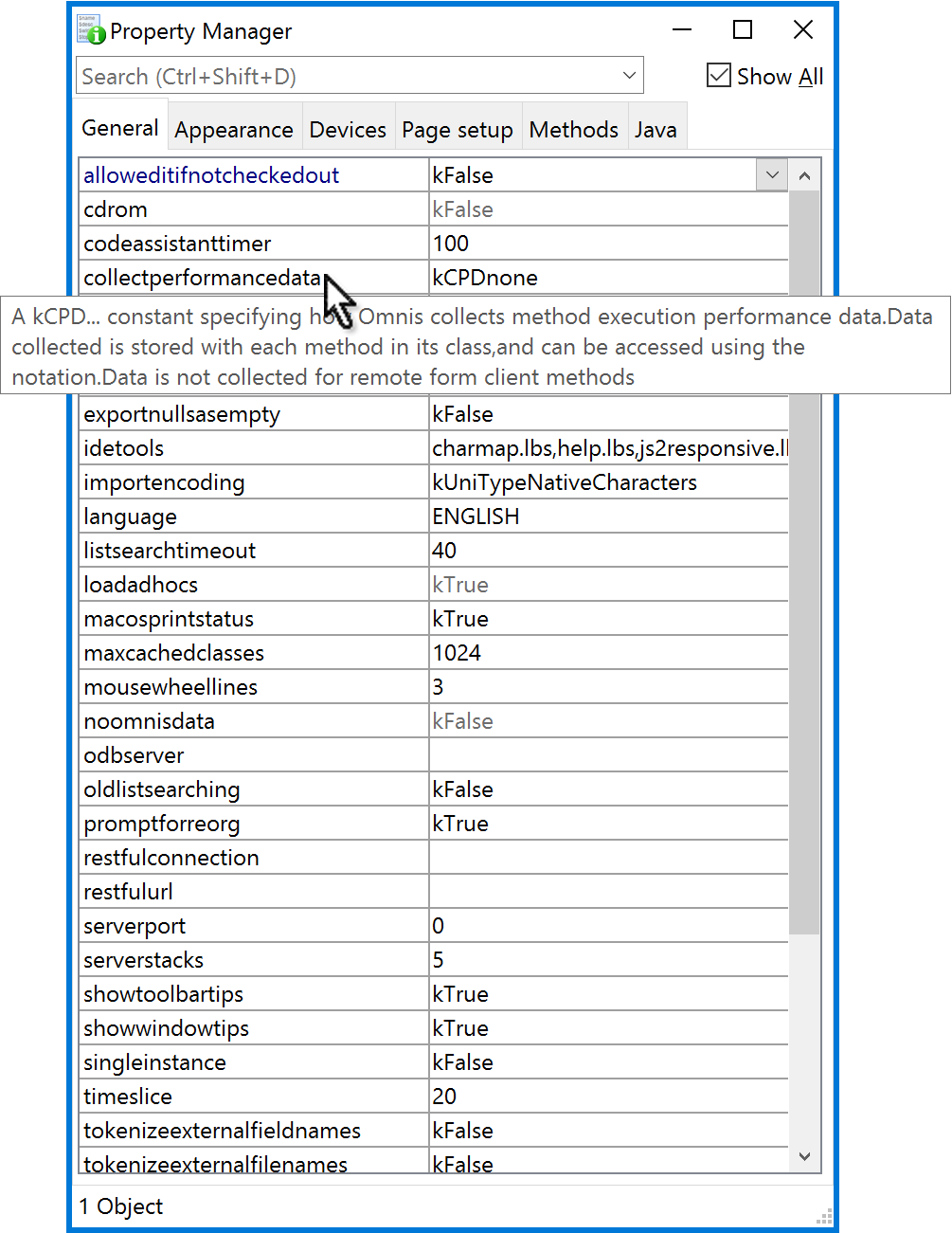
Select the Options/Preferences menu option in the **Tools** menu or toolbar on the main Omnis toolbar

or

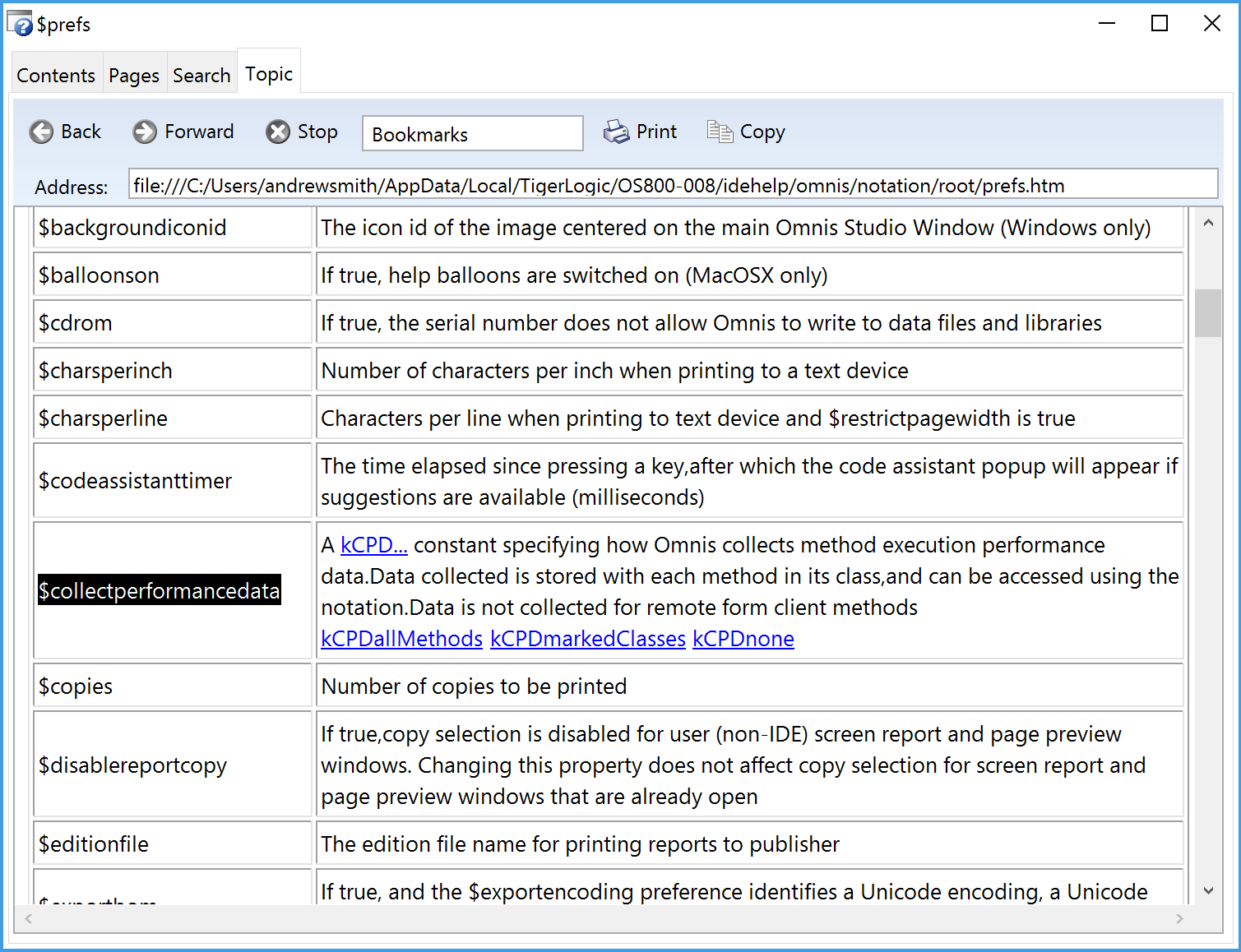
Click on ‘Omnis Studio’ in the Studio Browser and click on the **Prefs** option



The Omnis Preferences are displayed and updated in the Property Manager: to view all of the Omnis preferences, ensure the ‘Show All’ checkbox is checked. You can position your mouse over any property in the Property Manager and view its description in a tooltip. If you are looking for a specific property or preference, you can type a few characters into the Search box to find a property located on any tab.



You can select a preference in the Property Manager and press F1 to open a Help window for that preference (or property).



### Environment Font Size

The Omnis preference $idelistpointsize specifies the font point size used for lists within the main IDE windows in Omnis, including the Studio Browser, Catalog, and Property Manager. If you change this property, the new value is used when you restart Omnis.

## Omnis Configuration

You can control the behavior of the Omnis executable and some elements of the Omnis IDE by editing a configuration file called ‘config.json’, *which is created when Omnis is first launched* and is located in the ‘Studio’ folder under the main Omnis folder. The file is in JSON format and can be edited with any text editor or word processor.

When you start developing your application you may not need to edit this file since it contains all the default settings for running Omnis Studio in development mode. Some options in the config.json file relate specifically to how the Omnis App Server is setup for running web or mobile apps: when you deploy such apps you will need to edit the config.json file in the Studio folder in the Omnis App Server installed tree to configure your server settings, such as the server port number.

### Template Configuration File

The config.json file created in the Studio folder contains only a subset of settings needed initially for development, but there are many more settings you can add to the file to enable or configure further features in the Omnis IDE or the Omnis App Server. There is a template config.json file containing all possible sections and settings located in the ‘templates’ folder in the ‘Studio’ folder: you can copy sections out of this file and add them to your copy of the config.json file in the Studio folder.

### IDE Animation

Various parts of the Omnis IDE are animated, including the main tree list in the Studio Browser, and the Method names tree list in the Method Editor is animated when you open the editor or redraw the list.

The "animateIDEcontrols" option in the “ide” section of config.json controls whether or not animation is enabled in the IDE: it is set to True by default. Set this to false if you don’t want any objects in the IDE to be animated.

### Configuration File Methods

There are some methods in the Omnis Preferences that allow you get and set the contents of the Omnis configuration file. These would allow you, for example, to create your own config.json from code which could be used for deployment of your app.

1. $getconfigjson()   
   Returns config.json as a row (empty if config.json could not be parsed)
2. $setconfigjson(wConfigJson)   
   Sets config.json to the supplied row

These are methods of $root.$prefs, and they appear on the Methods tab of the property manager, but only when used with the Notation Inspector.

You can use them to modify existing items, or add new items. For example:

###### Do $prefs.$getconfigjson() Returns cRow

###### If isnull(cRow.obrowser.$cols.$findname("newitem"))   Do cRow.obrowser.$cols.$add("newitem",kCharacter,kSimplechar,1000000) End If Calculate cRow.obrowser.newitem as "my test2"

###### If isnull(cRow.obrowser.$cols.$findname("newitem2"))   Do cRow.obrowser.$cols.$add("newitem2",kBoolean) End If Calculate cRow.obrowser.newitem2 as kTrue

###### If isnull(cRow.obrowser.$cols.$findname("newitem3"))   Do cRow.obrowser.$cols.$add("newitem3",kInteger,0) End If Calculate cRow.obrowser.newitem3 as 123

Do $prefs.$setconfigjson(cRow)

## Studio Toolbars and Menus

The main toolbar at the top of the Omnis application window provides access to all the development tools in Omnis, such as the Studio Browser, the Catalog, the Property Manager, and so on. The Standard, View, and Tools toolbars contain many of the same options as the File, View, and Tools menus respectively. The Desktop toolbar (hidden by default) lets you switch between design and runtime environments.

You can drag any of the toolbars out of the docking area and place them anywhere in the Omnis application window (referred to as “floating”). You can place your mouse over any button to display its name or tooltip description.

The visibility of the main Omnis Toolbar and Menubar is different for Windows and macOS, as follows.

### Windows

Under Windows, the main Omnis Toolbar is shown but the main Menubar is not, by default. To hide or show any of the toolbars, or install the main Omnis menu bar, under Windows, you can Right-click/Ctrl-click in the toolbar area of the main Omnis development window and select the **Toolbar Options…,** **Toolbars…** or **Menu Bar** option: in addition, you can double-click in the toolbar area to open the **Toolbar Options.** See the *Omnis Menu Bar* section.

### macOS

On macOS, the main Omnis Menubar is shown but the main Toolbar is not, by default: the concept of an application toolbar is not present in macOS but you can open the main Omnis Toolbar manually. To hide or show any of the toolbars on macOS, select the Toolbars… option from the View menu: this allows you to show or hide the Standard, View, or Tools toolbars which you may find useful for development.

### Standard toolbar



The **Standard** toolbar has more-or-less the same options as the File menu. In addition, the File menu lets you create new libraries and open existing libraries.

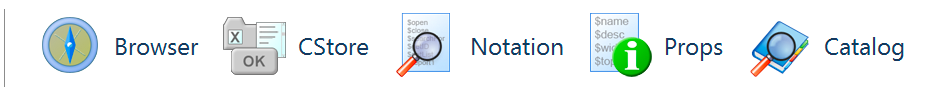
The **Save** option (Ctrl/Cmnd-S) saves the class you are currently working on. If a class design window is not currently selected this option is grayed out.

The **Revert** option rolls back any changes you have made to the class you are currently working on. Note that if you close an Omnis class it will be saved automatically and therefore cannot be reverted.

The **Dest** option opens the **Print Destination** (Shift-Ctrl/Cmnd-P) dialog which lets you set the destination of the current output. Reports are sent to the Screen by default, but you can choose another destination from this dialog.

The **Print** option (Ctrl/Cmnd-P) prints the current class or report to current destination, if applicable. For example, when you are working in the method editor the option prints the currently selected method or set of class methods.

### View toolbar



The **View** toolbar opens the main Studio Browser and other tools available in the Studio development environment, and has the same options as the View menu. Many of these tools are described in greater detail later in this chapter.

The **Browser** option (F2/Cmnd-2) opens the Studio Browser which lets you create and examine libraries and classes. If the Studio Browser is already open and in Single Window Mode this option will bring it to the top. If the Browser allows multiple copies of itself, this option opens the initial Browser displaying the libraries.

The **CStore** option (F3/Cmnd-3) opens the **Component Store** which contains data and GUI objects, as well as external components, to allow you to build windows, web forms, and reports. This option will be grayed out if there is no design window or report on top. (Note you create classes from the Studio Browser itself, so this option will be grayed out when a library is selected in the main Studio Browser.)

The **Notation** option (F4/Cmnd-4) opens the **Notation Inspector** which lets you view the complete Omnis notation tree. If the Notation Inspector is already open and in Single Window Mode this option will bring it to the top. If the Notation Inspector allows multiple copies of itself, this option opens a new instance of the Notation Inspector. When you open the Notation Inspector the Property Manager will also open.

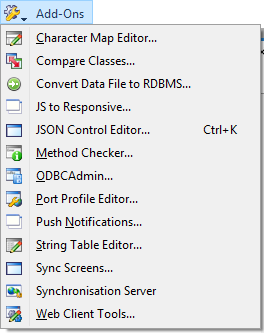
The **Props** option (F6/Cmnd-6) opens the **Property Manager** which lets you view or change the properties of an object: the properties displayed in the Property Manager will depend on the object or window currently selected or on top in the Omnis development environment (the ‘Show All’ option must be checked to see all the properties for the current object). If the Property Manager is already open, this option will bring it to the top.

The **Catalog** option (F9/Cmnd-9) opens the **Catalog** which lists all the field names and variables in your library, together with the functions, constants and event messages. If the Catalog is already open this option will bring it to the top.

In addition, the View menu contains the Inheritance option:

The **Inheritance Tree** option (F5/Cmnd-5) lets you view the inheritance or superclass/subclass hierarchy in the current library. If you select a class in the Browser and open the Inheritance Tree it shows the inheritance for that class.

### Tools toolbar



The **Tools** toolbar lets you open the **Trace Log** which provides a record of the operations and commands you have carried out. If there is an error in Omnis, such as when you start it up, you can look in the trace log for information about possible causes of the error. Omnis code execution and errors are also reported in the trace log: see the Debugging Methods chapter in this manual for further details.

The **Add-Ons** sub-menu option lets you open various additional tools, including:   
  
**Character Map Editor,** for mapping local and server character data  
**Compare Classes** tool, for comparing the methods in classes and libraries   
**Convert Data File to RDBMS,** for converting your Omnis datafiles to SQLite or PostgreSQL  
**JS to Responsive,** migrates $screensize based remote forms to responsive layout  
**JSON Control Editor,** lets you create JSON defined JavaScript components  
**Method Checker,** for checking methods in classes and libraries  
**ODBC Admin** tool, for setting up ODBC access to Omnis datafiles   
**Port Profile editor,** for setting up your ports   
**Push Notifications,** lets you setup notifications for mobile apps  
**String Table Editor,** for providing multi-languages in your application   
**Sync Screens,** for synchronizing layouts in old fixed screensize remote forms   
**Synchronization Server,** opens the Admin tool for the Sync Server   
**Web Client Tools,** for configuring old Web Client based apps

### Desktop toolbar

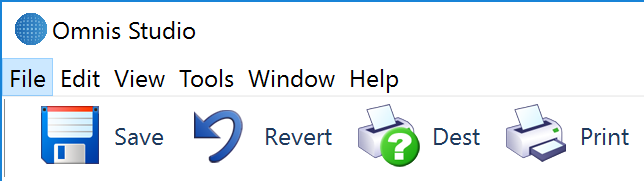


The **Desktop** toolbar (hidden by default) lets you toggle between the design environment and a simulated Runtime environment so you can see your (desktop) application in user mode. You can also change the mode using the **Desktop** option in the **File** menu.

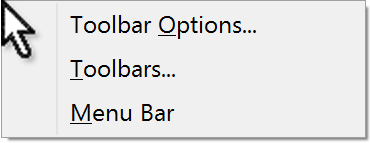
In *Design mode* the standard menus, such as File, Edit, View, and Tools toolbars are visible. In *Runtime mode* all these are hidden and only user menus, data entry windows and toolbars defined in your library are visible. Also in runtime mode there is a cut-down version of the File and Edit menus on the main menu bar. Being able to switch to Runtime mode lets you see exactly how your application will look when the user runs it. In *Combined mode* (the default) all design and user menus, windows, dialogs, and toolbars are visible. When you select the Runtime option, the Desktop toolbar is installed so you can get back to the Combined mode.

### Omnis Menu Bar

The main Omnis Menu Bar gives you access to various File operations, the standard Edit menu functions (Copy, Paste, Etc), as well as the View, Tools, Window, and Help menus. Many of the options in these menus appear on the main toolbars and are described in the previous sections: the other options are reasonably self-explanatory.

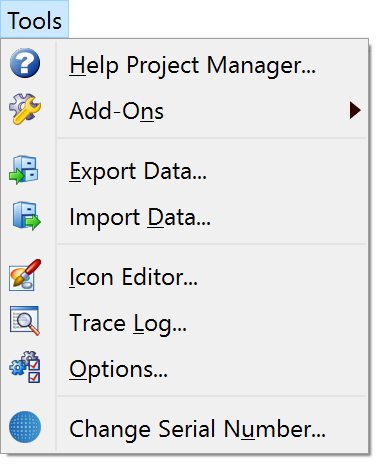


On macOS, the main Omnis Menubar is visible. On Windows, the main Omnis menu bar may not be visible, but you can press the **Alt** key to display it temporarily. To display it permanently, Right-click/Ctrl-click on the main Omnis toolbar and select the **Menu Bar** option.



The other options on the Toolbar context menu let you configure the main toolbars in Omnis Studio. The **Toolbar Options…** option let you install or remove the Standard, View, Tools and Desktop toolbars.

### Tools menu



The **Help Project Manager** lets you build help into your own libraries for the benefit of your own users.

The **Add-Ons** option lets you open various additional tools (described in the Tools toolbar section above)

The **Export Data** option lets you export data from an *Omnis data file* (not a SQL database) using a number of different data formats. While the **Import Data** option lets you import data into a data file from an existing export file or text file from another application.

The **Icon Editor** option opens the Icon Editor which lets you add your own icons to Omnis. However, for JavaScript Client based apps you can used Icon Sets.

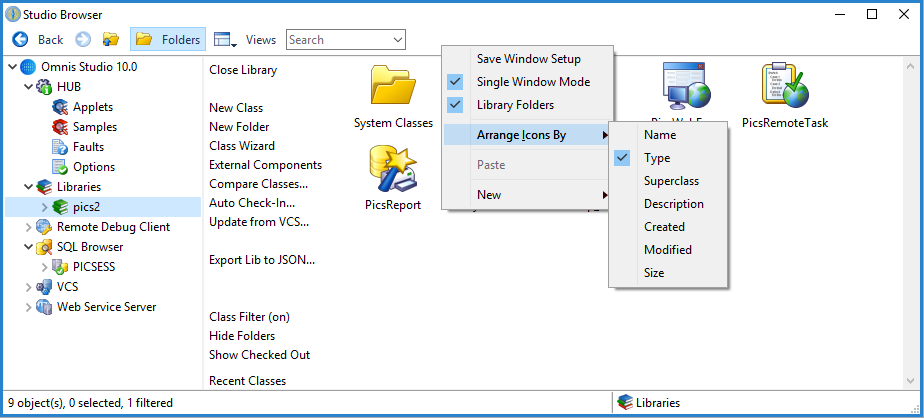
The **Trace Log** option displays the trace log which is a record of the operations and commands you have carried out. See the Debugging Methods chapter.

The **Options** option opens the Property Manager displaying the main Omnis Preferences, including settings for the Appearance of the Omnis environment, the Print devices for Omnis reports, and the main Page Setup.

The **Change Serial Number** option which lets you re-serialize your copy of Omnis, and add serial numbers for Omnis plug-ins such as oXML.

## Context Menus

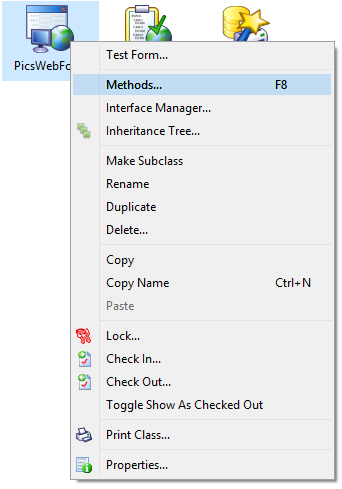
A *context menu* is a useful shortcut when you want to modify an object, or change the Omnis development environment. You can open context menus from almost anywhere in Omnis by **Right-clicking** on an object or window background. On macOS, you can open a context menu by holding down the **Ctrl key** and clicking your pointer on the object. The options in a context menu depend on the object you click on, that is, the *context of the right-click*. For example, if you Right-click on the Studio Browser its View menu will be opened containing options for changing its view and creating new classes. Note the **Save Window Setup** which will save the current setup or view: this useful option is available for many of the Omnis development windows.



##### The context menu on the Studio Browser lets you change the current view, or create a new class or folder when a library is visible in the Browser.

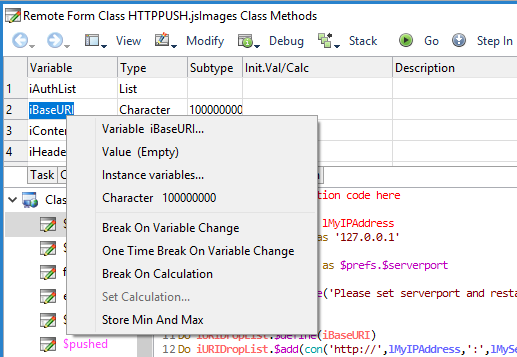
### Class Context Menu

If you right-click or Ctrl-click on a class displayed in the Studio Browser the class context menu will open: the contents of the menu will depend on the type of class, but some opens are available for all classes, e.g. options for the VCS. Classes that can contain methods will show the **Methods** option (F8) which lets you open the Method Editor for the class. UI classes like remote forms or window classes will have an option to **Test** the class instance (or Print for reports).



### Variable Context menus

In the Omnis Method Editor or Catalog, you can right-click on a variable name and get its current value. The Variable context menu shows the variable name, its value and its type and allows you to open the variable window showing its current value.

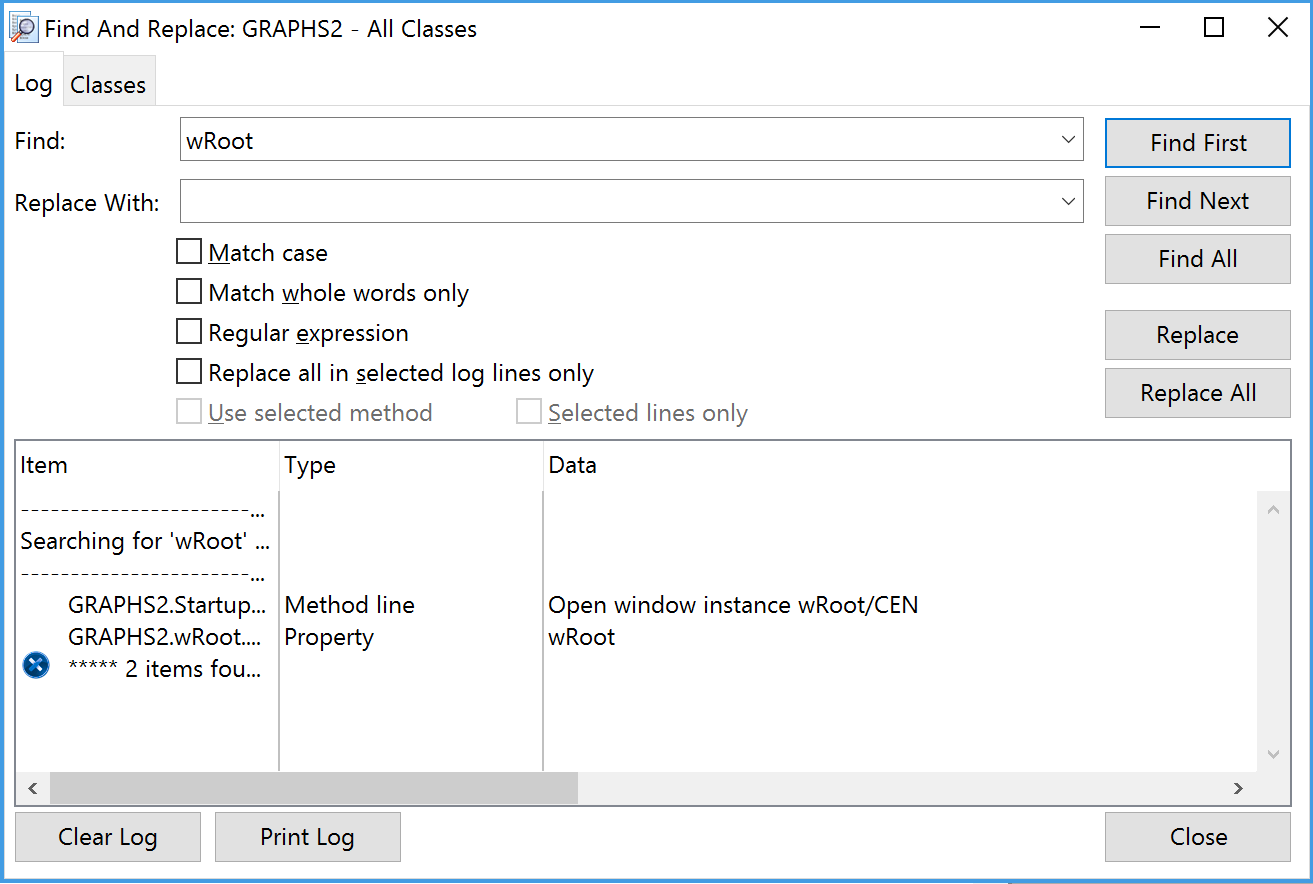


##### You can right-click in many parts of Omnis and open up a menu appropriate to the object or item under the mouse, like a variable displayed in the method editor, as above.

## Find and Replace

The Find & Replace tool lets you search through a class or library, or a number of classes or libraries, to find a particular text string. You can selectively replace occurrences of an item or replace all items.

The **Find And Replace** option under the Edit menu (or Ctrl/Cmnd-F) opens the Find & Replace dialog: note the **Find Next** option (Ctrl/Cmnd-G) lets you find the next occurrence of the current find string.



On the **Classes** tab you can select the libraries and classes in which you want to perform the find and replace. For a single library you can select some or all of the classes in the library. If you select more than one library under the Classes tab, all classes in all selected libraries are searched.

If you click on the Find First button Omnis will jump to the first occurrence of the text string in your selected classes or libraries. For example, if the specified item is found in a method Omnis will open the class containing the method with the found item highlighted. Find Next (Ctrl/Cmnd-G) will jump to the next occurrence of the text string, and so on. The Find All button finds all occurrences of the specified item in all the classes or libraries you selected and lists them in the Find log.

You can interrupt a find and replace operation at any time by pressing Ctrl-Break under Windows, Ctrl-C under Linux, or Cmnd-period under macOS.

### Regular Expressions

Find & Replace supports regular expressions which are sequences of literal characters and metacharacters that let you perform complex text search and modification (Omnis uses the regexp library provided by the University of Toronto). The following metacharacters are supported:

|  |  |
| --- | --- |
| Metacharacters | Action |
| ^ | Beginning of line |
| $ | End of line |
| | | Or |
| [abc] | Match any character enclosed in the brackets. |
| [^abc] | Match any character not enclosed in the brackets. |
| [a-z] | Match the range of characters specified by the hyphen. |
| . | Match any single character. |
| ( ) | Group the regular expression within the parentheses. |
| ? | Match zero or one of the preceding expression. |
| \* | Match zero, one, or many of the preceding expression. |
| + | Match one or many of the preceding expression. |
| \ | Escape the metacharacter that follows so that the literal meaning of the character is used. |
| & | Reference the entire matched text for string substitution. |
| \n | Reference subgroup n within the matched text (n can be 1-9). |

The regexp library *does not* support the following features:

**Word boundaries**   
the \< begin word and \> end word metasequences.

**Back references**   
the \n notation, which lets you reference a previously matched group.

**Repetitions**   
the {min,max} metasequence that specifies a minimum and maximum repetition of a regular expression.

### Find Log sorting & searching

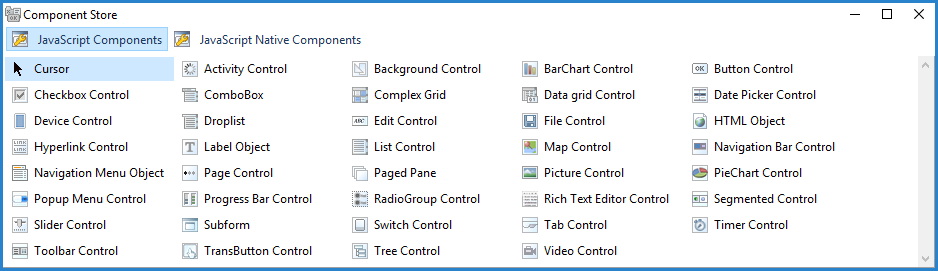
You can sort the Find log list by clicking on the header buttons, plus you can sort the list by either of the first two columns by using the context menu. The context menu also has an item to sort the last column. Keyboard searching of the Find log list searches column one and two, which means you can locate entries of a particular type more easily in the Find log, by clicking on the Type header to sort the list, and then typing the type name to search the list.

### Renaming objects

When you rename certain objects in your library, Omnis will replace all references to the object automatically. For example, if you rename a class variable in the method editor Omnis will replace all references to the variable within the class automatically. However if you try to rename most other types of object, such as renaming a class in the Browser, Omnis will prompt you to find and replace references to the object. If you answer Yes to the prompt, Omnis will open the Find & Replace tool and the Find log which lets you monitor the find and replace, or control whether or not certain references are replaced.

## Component Store

The *Component Store* contains the objects and components you need to build web and mobile forms, reports and toolbars. When you create a new class or modify an existing one, the Component Store will open automatically. For example, when you create or modify a remote form in your web or mobile app, the Component Store will display the JavaScript Components.



You can create an object or component from the Component Store in a number of ways:

1. You can *drag* the appropriate icon from the Component Store and *drop* it onto your remote form, report or toolbar,
2. You can *double-click* on a component icon in the Component Store and the component will be added to your class automatically,
3. You can *select* the component icon you require in the Component Store and *press Return*: this is the same behaviour as double-clicking on a component icon.
4. You can also *select* a component icon in the Component Store, then *click and draw* using the mouse or trackpad on your remote form or report to draw a component the required size and position.

The bitmap for an object dragged from the Component Store has a rectangle with the same dimensions as the control that it will drop, allowing you to position the new component in your remote form or window.

You can change the view of the Component Store using the **View** context menu, by Right-clicking/Ctrl-clicking inside the Component Store and selecting the options you require. For example, you can select the **Text** option to display the name of each component (as shown above). You may find this helps you identify each type of component while you become familiar with Omnis. You can also resize the window to place it along the bottom or down the right side of the Omnis development window.

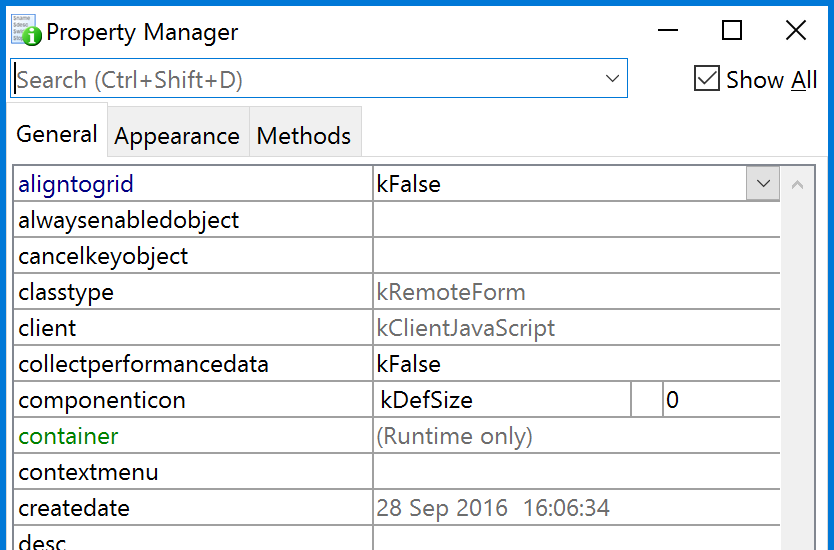
When you have set up the Component Store how you want it, you can save the view options using the **Save Window Setup** option in the context menu, available when you right-click the background of the window.

## Property Manager

The *Property Manager* lets you display or change the properties of the currently selected object in the Studio Browser or on a design window. This could be a library file or a window class selected in the Studio Browser, or a JavaScript component selected in a remote form: in addition, the Property Manager can be used to change the Omnis global preferences. The Property Manager will normally appear automatically when needed, when you click on a window or object that has properties. If it is not visible you can display it either by selecting the **View>>Property Manager** menu item from the main menu bar, or by pressing **F6/Cmnd-6,** or by Right-clicking on an object, such as a component or form background, and selecting the **Properties** option from the context menu.

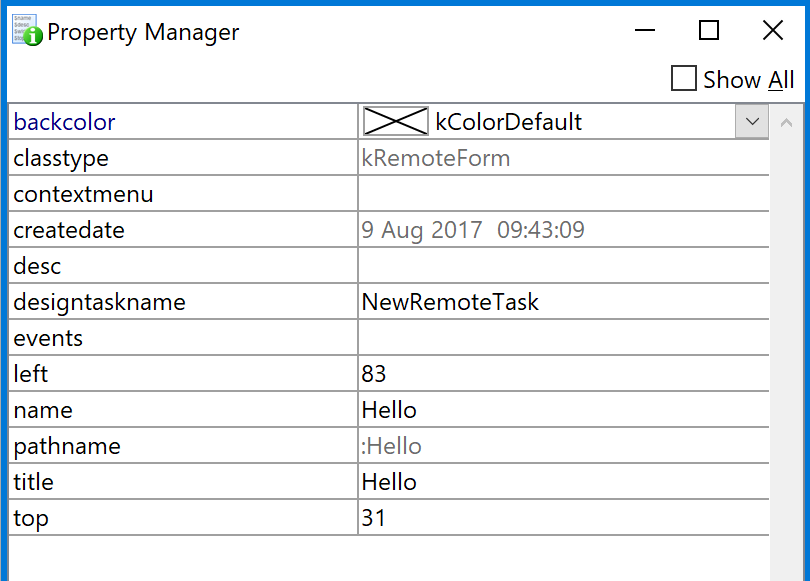
### Property Filter

The Property Manager displays all the properties for an object or a subset of properties. There is a checkbox at the top of the Property Manager window labelled *Show All*, which either shows *all properties* for the current/selected object (and listed under separate tabs), or a single filtered list of “basic” properties (with no tabs) when the Show All option is unchecked: the latter is intended for new users who may only need to access the basic properties.



The value of the *Show All* filter is saved with the window setup, and its initial value is set according to whether you chose “advanced user” or “new user” in the new Welcome dialog: advanced sets the “Show All” or unfiltered mode, while new user sets the filter to basic view.

When the Show All option is *unchecked,* the property list shows a basic subset of properties for the current object (selected library, class or form component), or for the current context in the IDE, such as the Omnis preferences. For example, the following image shows the properties for a **remote form** in “basic” mode:



If you use Find & Replace (on the Edit menu) to locate a property, and double-click on the find and replace log to select the property in the Property Manager, the Property Manager automatically switches to “Show All” mode if the property is not part of the basic set.

#### Modifying the basic set of properties

The basic set of properties is defined in a file called basicproperties.json and stored in the Studio folder under the main Omnis folder. You can modify this file if you want to change the properties shown in the filtered state of the Property Manager. The file is in JSON format, and contains an array of property names which must be lower case, and include the :: prefix if the property name requires one (e.g. some external component properties).

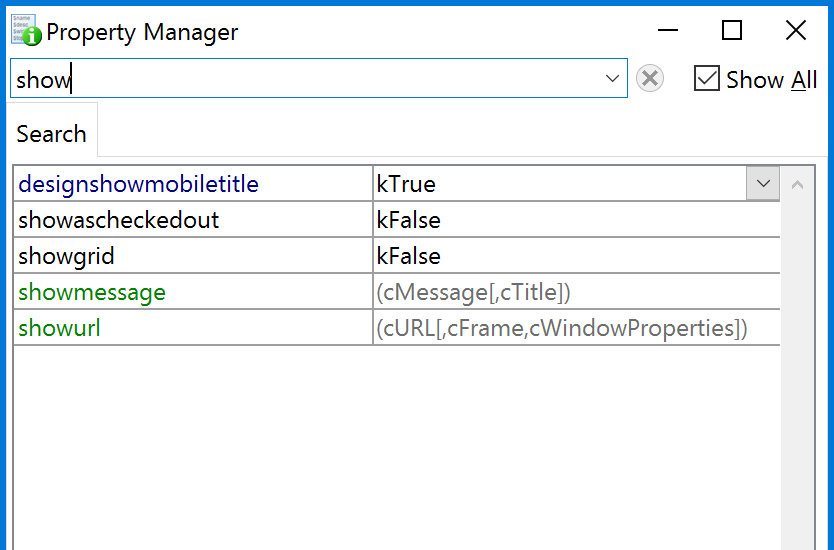
Omnis re-reads this file if it has changed when you uncheck the “all” checkbox in the Property Manager: so checking and unchecking this box forces a re-read. If the file has invalid syntax and cannot be parsed, Omnis writes an error to the trace log, and no basic properties will be displayed.

### Property Search

There is a Search box at the top of the Property Manager window which allows you to search for a property (note the search box is only visible when the “Show All” check box is checked). You can type a word or part of a word into the search box and the property list will update itself as you type.

The search results are property names that *contain the string* you entered, and they are shown in a single tab named ‘Search’. The search results are always sorted by property name, irrespective of the sort list option on the context menu. You can click on a property in the property list and update or set its value.

For example, entering ‘show’ into the property search for a remote form will provide a subset of properties containing the word ‘show’.



You can use the Backspace to clear a search string character by character, or you can click on the X icon to clear the whole string. The shortcut Ctrl/Cmnd+Shift+D moves the focus to the search box: you can press tab to return the focus to the property list. The 20 most recent searches are saved for re-use, which you can view by clicking on the drop arrow in the search box.

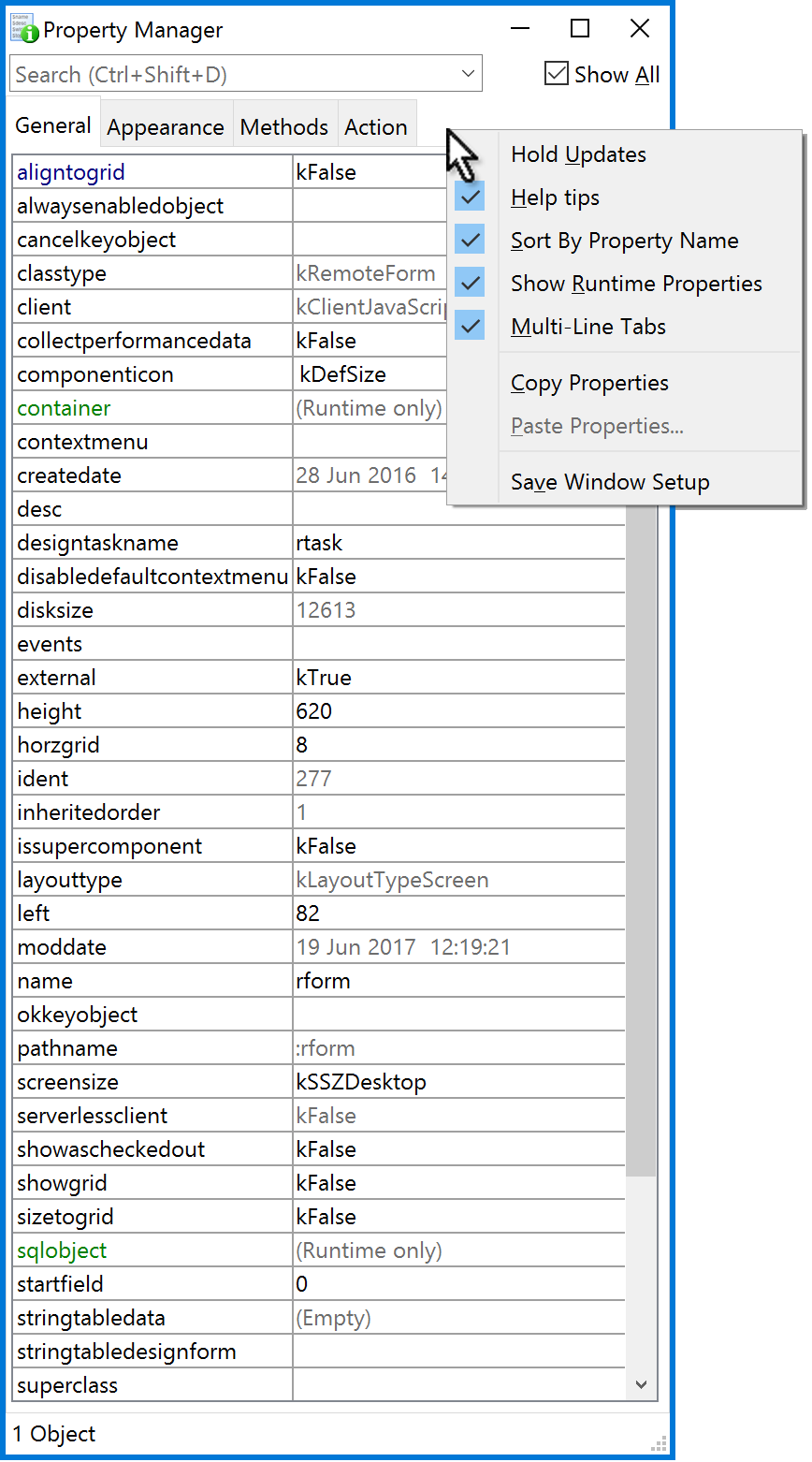
Each keystroke in the Search box performs a search, so there is a delay before a search is saved to the list: the delay defaults to 500ms, but you can change it in the config.json file in the “ide” group: “savePropertySearchDelay”.

If you use Find & Replace (on the Edit menu) to locate a property, and double-click on the find and replace log to select the property in the property manager, the property manager clears the search before selecting the property.

For both the basic mode, and the all mode when search results are being displayed, copy and paste properties are disabled on the context menu.

### Save Window Setup

You can configure the Property Manager using its context menu, which you can open by right-clicking on its background. In addition, you can resize the window, and drag the column separator to resize the columns. When you have set up the Property Manager how you want it, you can save the layout using the **Save Window Setup** option in the context menu.

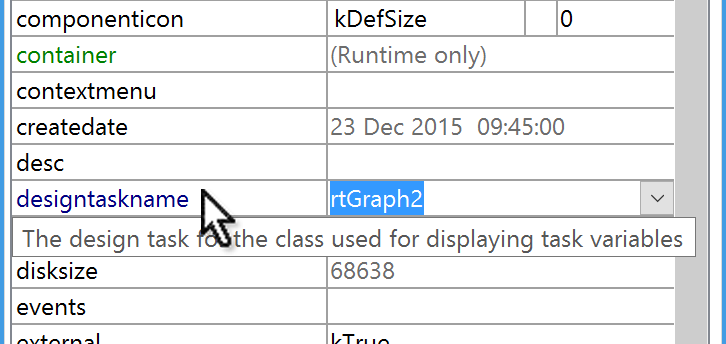


##### The properties of an object are shown in the Property Manager in alphabetical order by default, but you can list them functionally by unchecking the Sort by Property Name option in the Property Manager context menu.

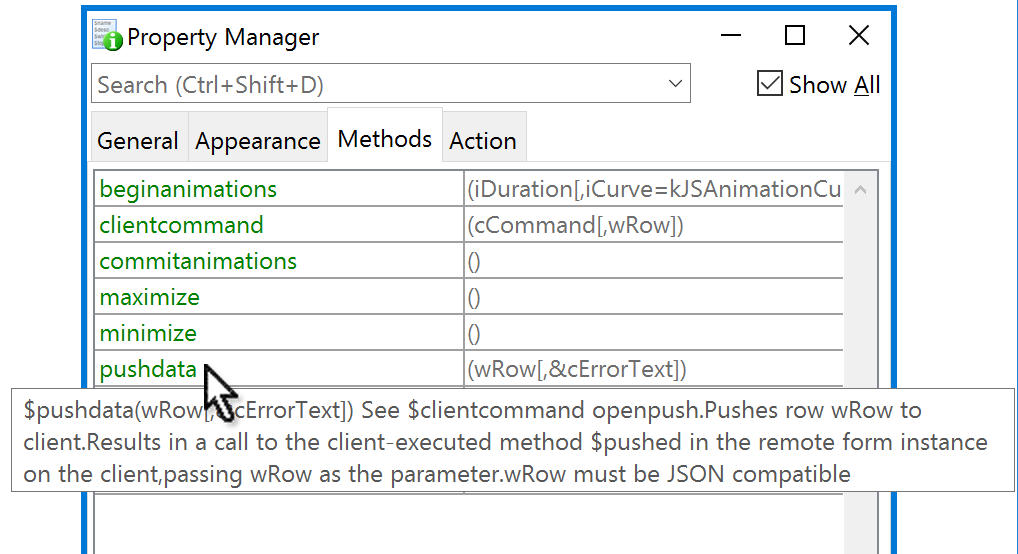
The other options in the Property Manager context menu affect the behavior of the Property Manager. If set, the **Hold Updates** option stops the Property Manager updating its contents automatically when you click on another object. For example, you can click on a window class to show its properties in the Property Manager, select the Hold Updates option, click on a field in the window, and the Property Manager still displays the properties of the window. Most of the time however you’ll want this option turned off so the Property Manager updates itself automatically.

### Property Descriptions

The **Help tips** option in the Property Manager context menu displays short descriptive help messages for each property in the Property Manager: this is particularly useful for showing the parameters of a method. The **Show Runtime Properties** option lets you view properties that are normally available in runtime only, that is, properties of an instance rather than a design class. When runtime properties are visible in the Property Manager the methods for the instance are also shown. You cannot set runtime properties or run methods from the Property Manager itself, but you can drag references to them into your code in the method editor (see below).



With the Help tips option enabled, the Property Manager shows a short description for each property.



##### When methods are visible in the Property Manager, the help tips show a description and the parameters for each method.

Note you can drag a property or a method (and its parameters) from the Property Manager into the Method Editor when you are writing code: see ‘Dragging Methods’ below.

### Copying Properties

The **Copy Properties** option lets you copy all the properties of one object and paste or ‘apply’ them to another object of the same type: this is useful if you want to reproduce or duplicate a particular object, or even apply the properties of a built-in field to an external component. Only those generic properties that are contained in the source and target objects are copied.

### Show Property Context Menu

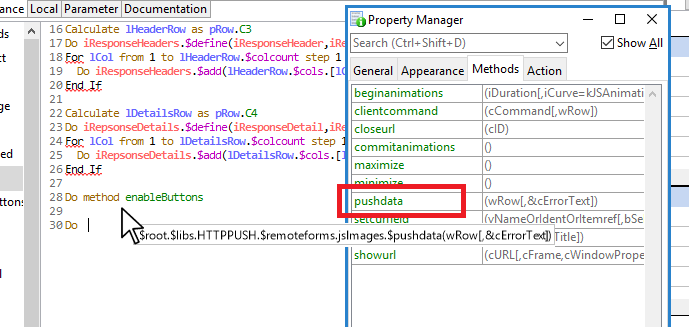
You can specify any built-in property to be displayed in the window or report field context menu by Right-clicking on the property in the Property Manager (when a field is selected) and selecting ‘Show property using editor context menu’. For example, the context menu for window and remote form editors shows ‘Show $order’, which allows you to display field ordering, but you can change it to most other properties. The field $ident is shown in the report editor context menu. This does not work for properties specific to external components, so for such properties the ‘Show property...’ option is grayed out.

Save Window Setup in the Property Manager saves the selected property. A different property can be saved for each class editor.

The property value is displayed for background objects. If a property is not supported by an object, then nothing is displayed for that object.

### Dragging methods from the Property Manager

Apart from displaying the properties and methods of classes and objects, the Property Manager lets you transfer a property or method and its notational path to your Omnis code in the Method Editor. You do this by dragging the property or method out of the Property Manager and dropping it onto your code in the method editor, for example, you could drag a method, such as the $pushdata() remote form method, and drop it onto the calculation field of the *Do* command.



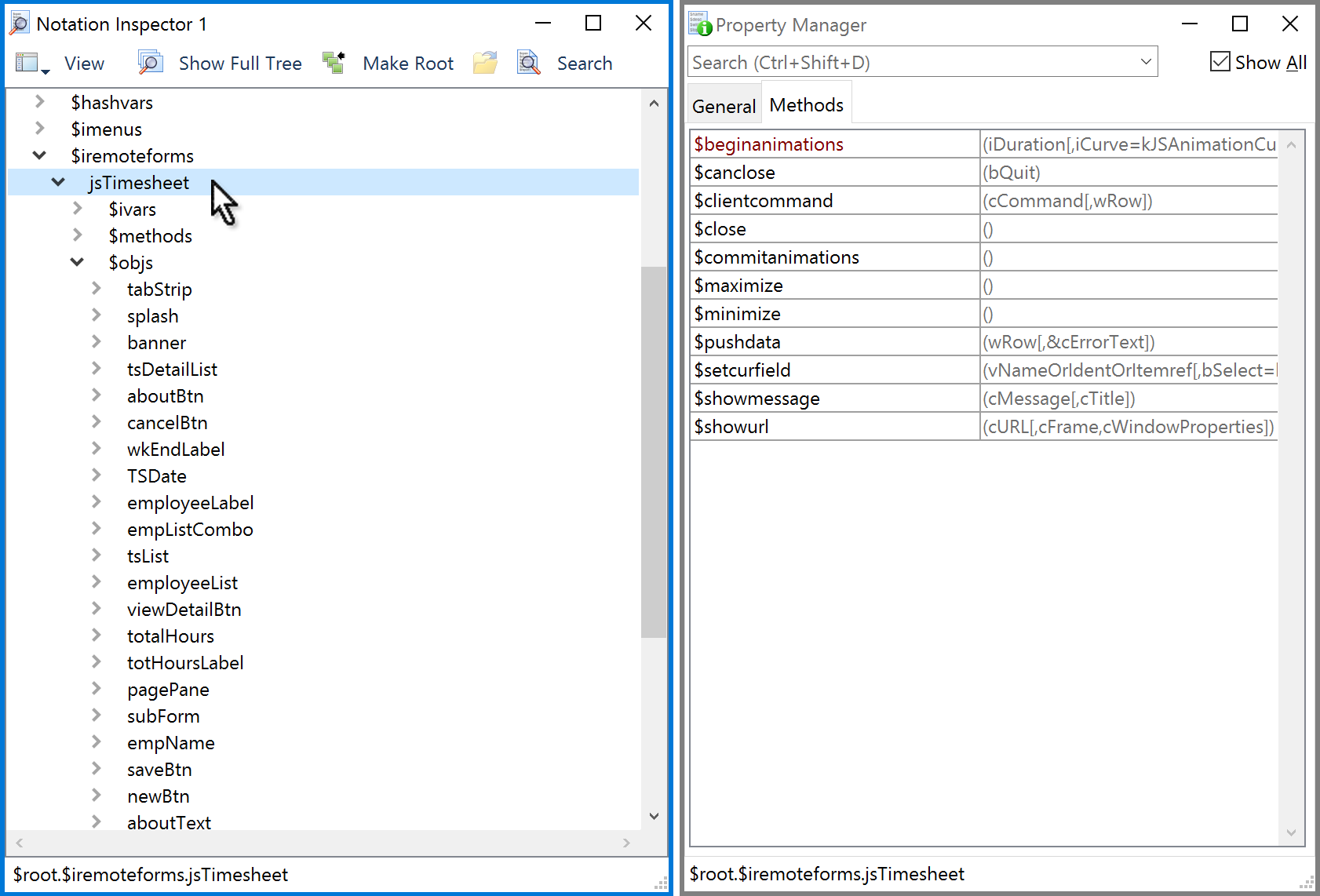
##### When you drag a method out of the Property Manager and drop it onto your code, its full path and parameters are copied as well. You can click in your code and edit the path and replace the parameters with your own variable names.

## Notation Inspector

Omnis structures its objects in an object tree, or hierarchical arrangement of objects and groups that contain other objects. The complete tree contains all the objects in Omnis itself, together with your design libraries, classes, and other objects created at runtime, such as form instances. You can view the complete object tree in the **Notation Inspector,** which you can open from the View menu, or by clicking on the icon in the main Omnis toolbar, or by pressing F3/Cmnd-3.

To facilitate a system of naming or referring to an object in the object tree, and its properties and methods, Omnis uses a system called the *notation*. The notation for an object is really the path to the object within the object tree. The full notation for an object is shown in the status bar of the Notation Inspector. You can use the notation to execute a method or to change the properties of an object, and you can use a notation string anywhere you need to reference a variable or field name.

The *Notation Inspector* therefore lets you view the Omnis object tree from the $root object down. It is particularly useful for viewing the contents of your library and finding the right notation for a particular object or group of objects in your library. For example, you can get the notation for an object on a design or open window using the Notation Search tool on the Notation Inspector toolbar. When you click on an object or group in the Notation Inspector the Property Manager will display its properties and methods: note the ‘Show All’ option has to be checked to view all the properties for the current object.



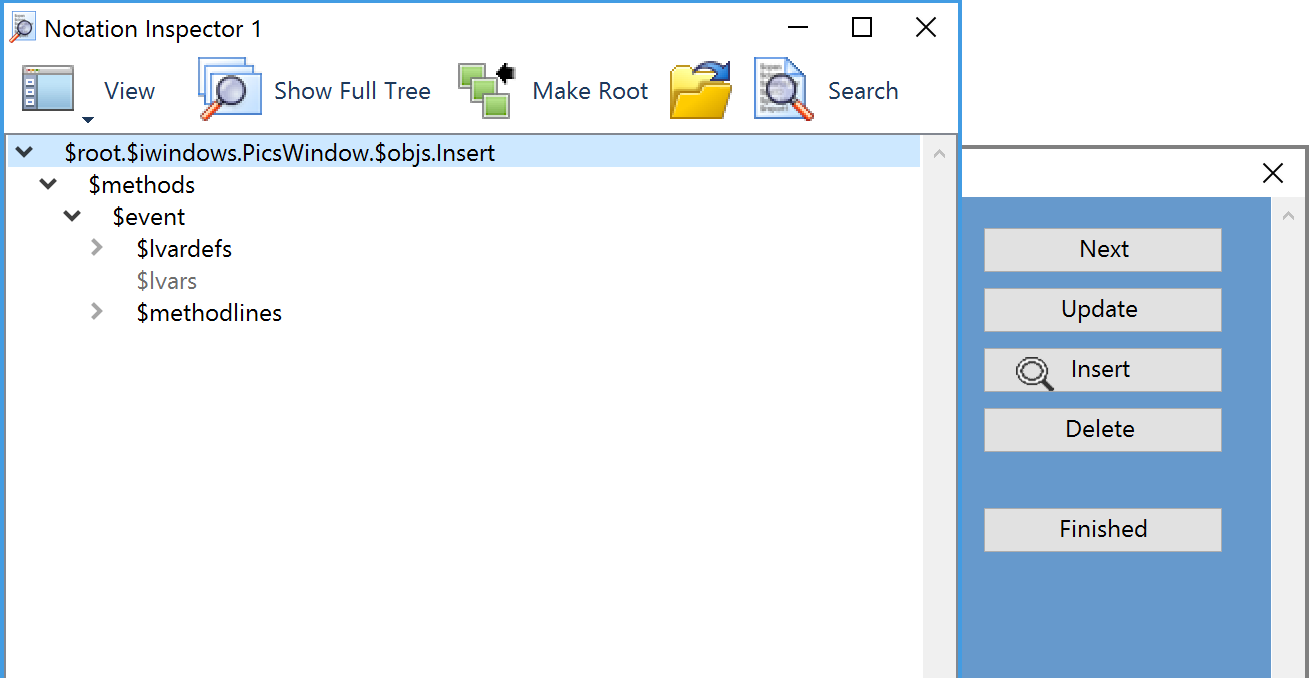
##### The Notation Inspector lets you drill down the hierarchy of objects within the Omnis tree: for example, you can view the object tree for an open remote form in the $iremoteforms group: when you click on an open remote form name, the Property Manager displays the runtime properties and methods of the form.

When the Notation Inspector opens it shows the $root object which contains all the objects in the Omnis object tree including all your open libraries and their contents. It also includes all the objects and groups created at runtime when you run your application. You can expand each branch of the tree to show the contents of an object or group.

All the different types of class in your library are shown in Notation Inspector within their respective object group. For example, all the window classes in your library are contained in the $windows group. Likewise all the toolbar classes are contained in the $toolbars group. Note that the $classes group contains *all* the classes in your library. Also note that a group may be empty if your library does not contain any classes of that type.

### Finding the notation for an object

You can find the notation for a window object in design or runtime mode, also report objects and toolbar objects. To do this, click on the Notation Search button on the Notation Inspector toolbar then click on the object in your open window or toolbar (this works for desktop apps only, not web based forms). The Notation Inspector will refresh itself showing the notation for the object you clicked on. The object becomes the root of the tree, so you can expand the tree to view its contents.



##### You can click the Search button, then click on a button or other object on an open window and view its notation in the Notation Inspector: the tree is redrawn and you can drill down to the view the contents of the object.

### Dragging notation from the Notation Inspector

Having found the object you’re interested in, you can copy its full notation to the clipboard and paste anywhere in your library, or you can drag an item from the Notation Inspector and drop it onto your code in the method editor. For example, you can drag the item into the calculation field of the *Do* command in the method editor.

### Code Assistant

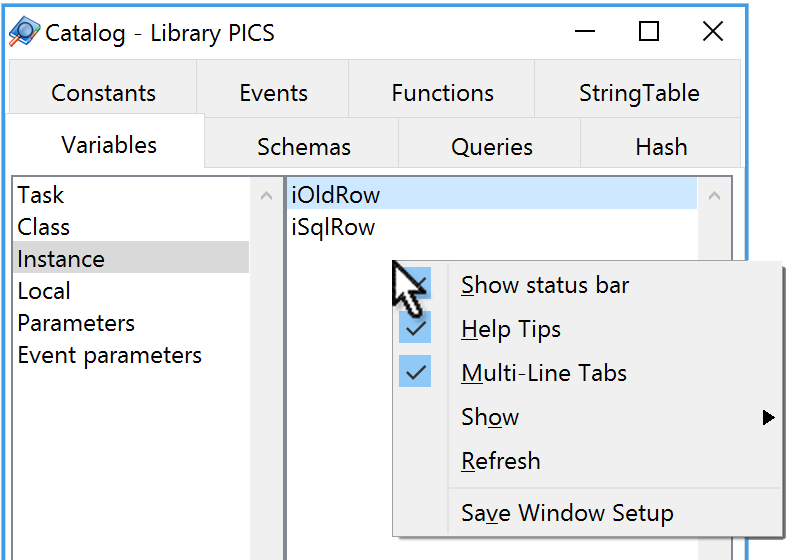
The Code Assistant is built into the Method Editor and assists you in writing Omnis code methods. When you are typing a notation string into the calculation field of the method editor you can pause to allow the Code Assistant to popup a list of possible properties, methods, and variable names that can be used in the current code method.

When the list pops up you can use the up and down arrow keys, and then the Return key to select the property, method, or variable you require, or you can use your mouse.

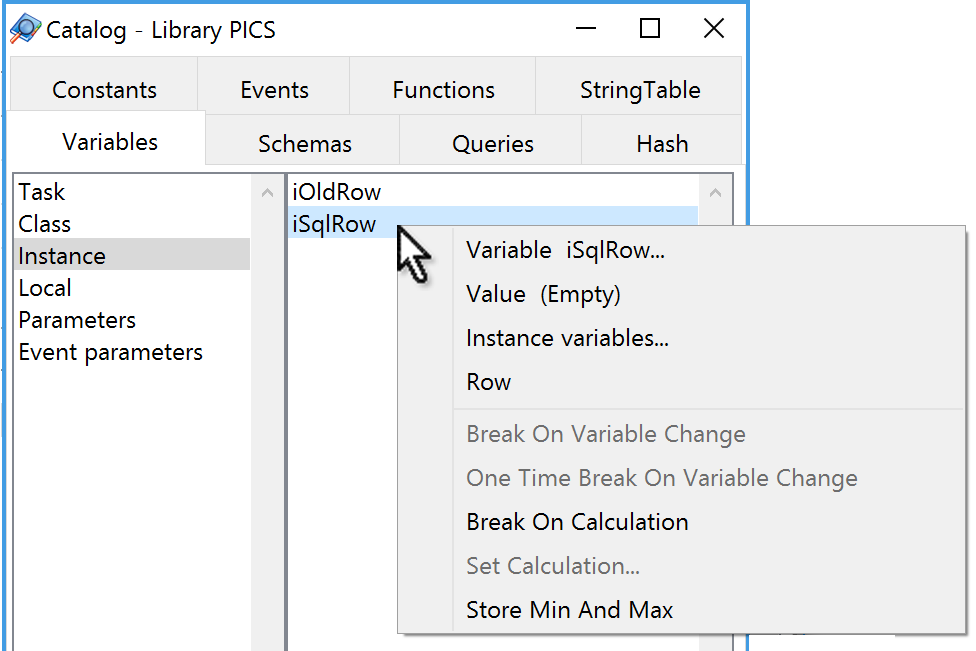
## Catalog

The *Catalog* lists all the variables, schema & query class columns, and string tables in your library, in addition to listing all Omnis functions, constants, event codes, and hash variables.

The Catalog lists all the variables for the current object including task, class, instance, local, and parameter variables, and also event parameters. For example, if you are working in a window class the Catalog will show the class and instance variables for that window class. In addition, when you select a particular method in the method editor, the Catalog will list the local and parameter variables for that method.



The context menu on the Catalog lets you change its appearance and style of tabs.



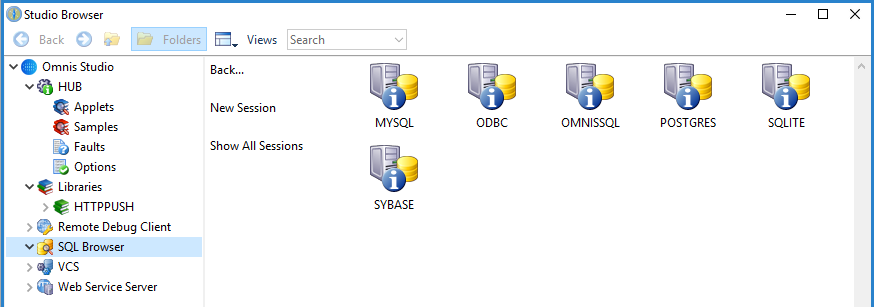
You can right-click on any variable or field name to view its type, current value and other information.

### Dragging items from the Catalog

When you have found an item in the Catalog, a variable, a schema class, an event, and so on, you can enter its name into a calculation field by double-clicking on it (assuming the cursor is in the calculation field) or by dragging it out of the Catalog and dropping the item in the appropriate place into your code in the method editor.

## SQL Browser

The SQL Browser lets you access and interact with a number of remote server databases, including Oracle, MySQL, PostgreSQL, OpenBase and SQLite. Omnis Studio provides a separate *Data Access Module* (DAM), or *Object DAM,* to connect directly to each server database. In addition, the SQL Browser contains an ODBC DAM that allows you to access ODBC-compliant databases, such as MS SQL Server, and many others. The SQL Browser contains a session template for each database server supported in Omnis.



##### The SQL Browser contains a session template for each supported server database including Oracle\*, MySQL, PostgreSQL, Sybase, and SQLite, as well as generic an ODBC connection. \*Note that, in some cases, a session template may not be displayed if the appropriate clientware is not installed on your computer.

The SQL Browser also contains an Omnis SQL DAM that lets you access an Omnis data file using SQL methods rather than the Omnis Data Manipulation Language (DML).

### Creating and Editing a Session

In the SQL Browser, you can create a new session, or duplicate an existing one and modify it adding your own connection parameters. For most purposes, the quickest and easiest way to create a new session is to duplicate an existing template and modify it.

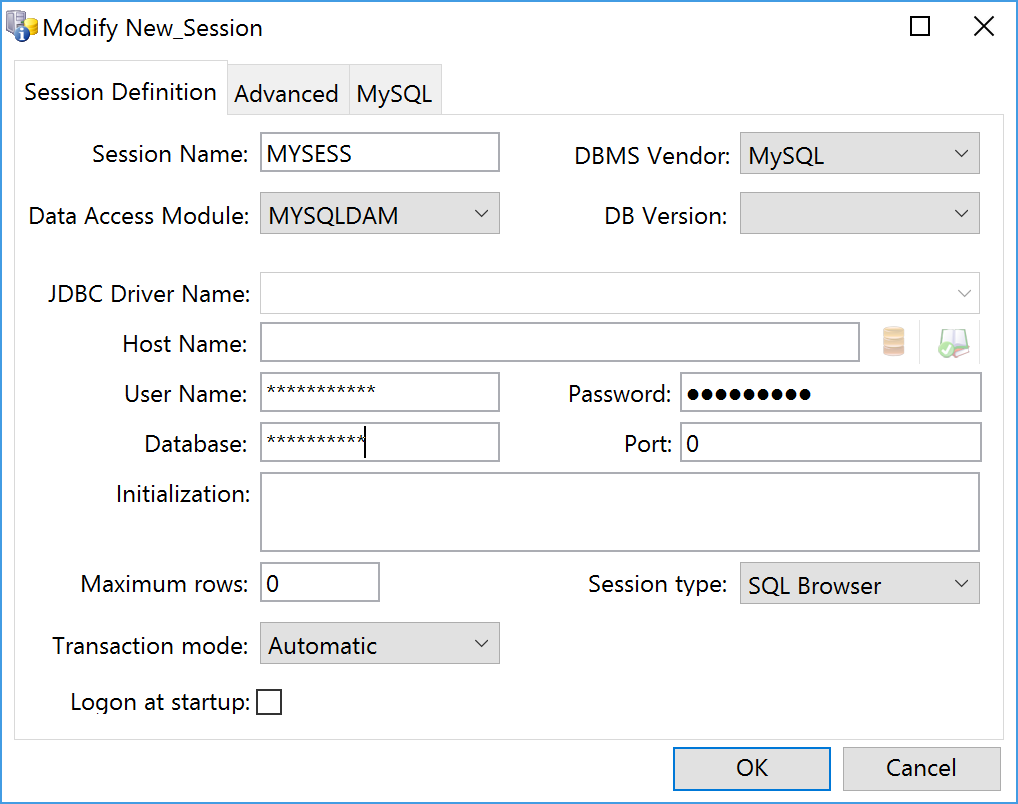
To create a new session to your database

Click on the SQL Browser option in the Studio Browser, then click on the Session Manager option

Select the type of database you want to connect to and click the Duplicate Session option

Rename the session using a suitable name for your database

Double-click the session template and modify the parameters in the Modify Session dialog



When you create or modify a database session you need to select the DBMS vendor, the Data Access Module (DAM) and version, as well as specifying the Username and Password for the session. The connection parameters required will vary depending on your database.

### Opening a session and accessing your data

Having defined or modified your session template, you can open or log onto your session in the SQL Browser.

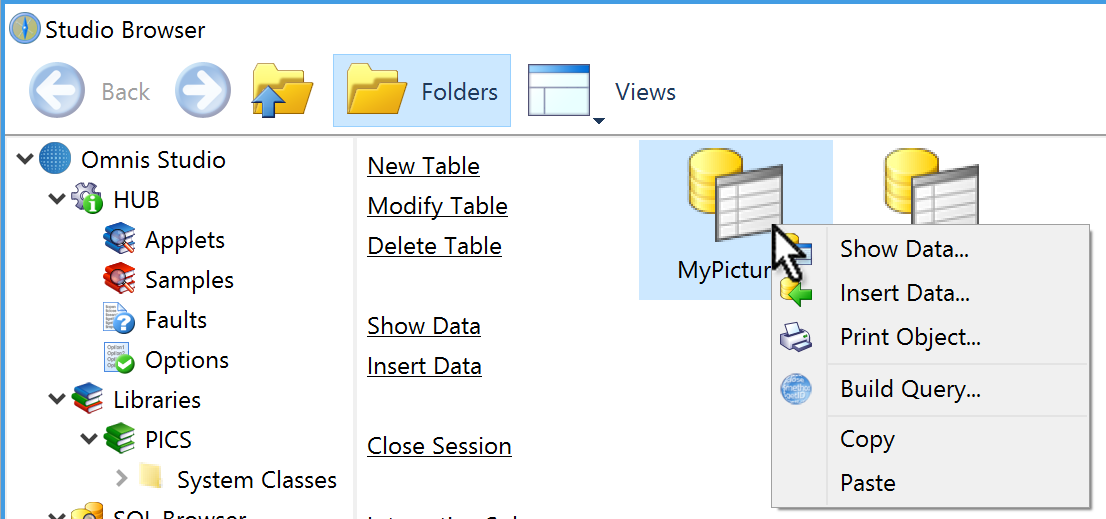
To open a database session

Click on the **SQL Browser** option in the Studio Browser

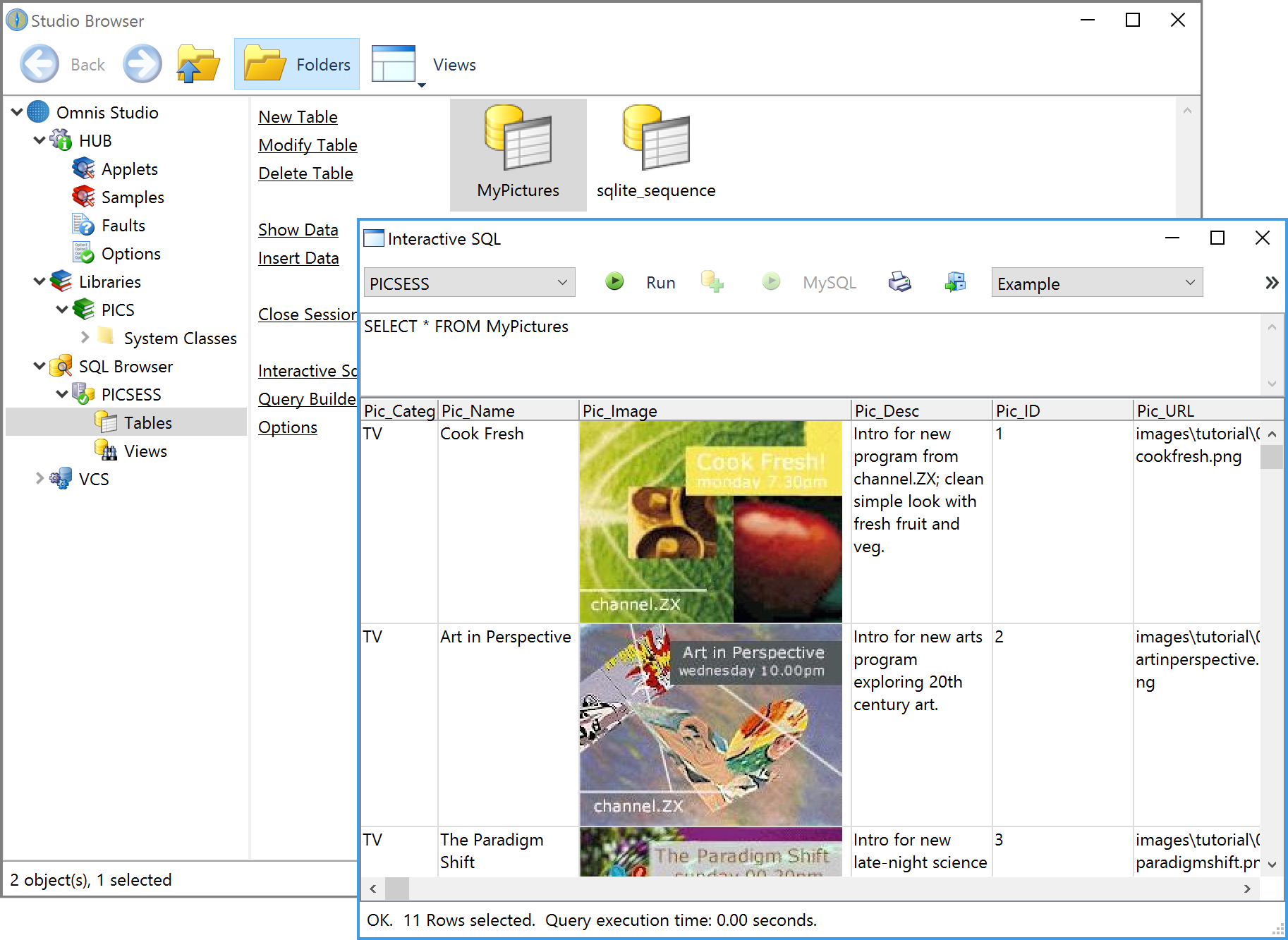
Click on the **Open Session** option and then the session name, e.g. PICSESS

Double-click on the **Tables** object to view the tables on your database

Right-click on a table name (e.g. MyPictures) and select the **Show Data** option



The Show Data option opens the **Interactive SQL** (iSQL) tool which submits a basic SELECT on your table to display the data.



##### If you use the Show Data option on the SQLite database used in the tutorial, you can view the text and images in the database.

Once you have created your database session you can view your data using the Interactive SQL tool, as above, or create windows and forms based on the session using the SQL Form Wizard, or using the App Builder under the New Library option in the Studio Browser.

Note the Tutorial shows how you can build a SQL form to access data via a web browser on a desktop PC or mobile device.

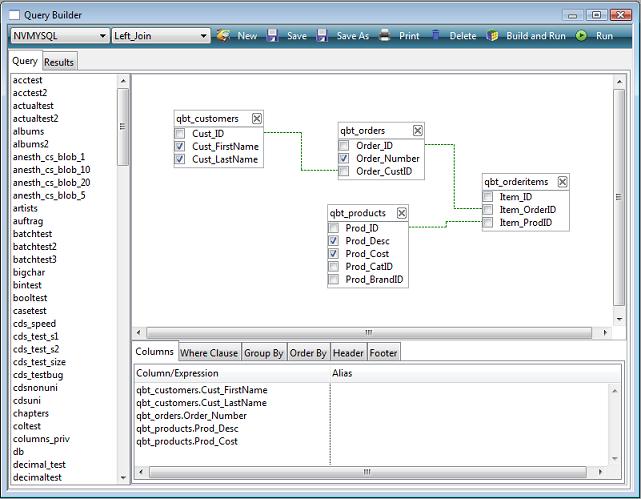
You can resize the font in the iSQL (results pane) using the Ctrl+/Ctrl- shortcut keys.

## SQL Query Builder

The SQL Query Builder (SQB) lets you build, run and store SQL Queries quickly and easily using a graphical interface. The SQB is built into the Omnis Studio IDE and is easily accessed via the SQL Browser. The SQB supports the generation of both simple queries requiring little SQL knowledge and more advanced queries using different join types and clauses. You can save queries for later use and you can create Omnis query classes based on your queries to allow you to take advantage of the queries you build in the SQB in your applications.

### SQL Query Builder window

You can open the SQL Query Builder by clicking on the ‘Query Builder’ option when the SQL Browser option is selected in the main Studio Browser window. The SQB window has three main panes: the table pane on the left showing all available tables, the main design area at the top-right showing the query in graphical format, and the lower tab pane for defining aliases, SQL clauses and expressions.



##### All panes in the Omnis SQB are resizable and can be saved via ‘Save Window Setup’ of the main context menu. The list of available tables may also be refreshed at any time via the table list context menu.

### Creating a Query

The SQL Query Builder allows you to construct most types of query simply using drag and drop and the context menus available as appropriate in each pane of the main SQB window. In order to construct a query, you must be logged on to a SQL session via the SQL Browser.

The first time you open the SQB the query list will be empty allowing you to create a new query. If an existing query is currently displayed pressing either the ‘New’ toolbar button or selecting ‘<new query>’ from the query dropdown list will clear the query from the screen.

#### Selecting a SQL Session

All open sessions are shown in the droplist at the top-left of the SQB window. You can select a session to build your query by dropping down the session list and selecting a session.

#### Adding a Table

Once a SQL session has been selected a list of tables for the current session is shown in the left hand pane. To include a table in a SQL Query simply drag the table into the main design area on the right.

#### Removing a Table

You can remove a table from a query either by clicking on the ‘X’ icon for the table or by opening the context menu for the table and selecting ‘Remove Table’. You have to confirm if you want to remove a table from the current query.

#### Refreshing a Table

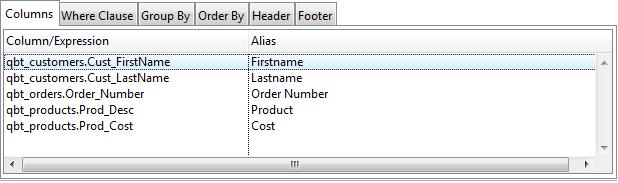
If a table has been modified outside the SQB, you can refresh the table in the SQB by right-clicking on the table and selecting ‘Refresh Table’ from the context menu. This option will add any new columns or remove any deleted columns to reflect the current state of the server table.

#### Selecting Columns

Once a table has been added, you can select columns by selecting their names individually or by right-clicking on the table you can check or uncheck all the columns in the table. Alternatively, you can specify that the SQB checks all columns automatically when a table is added using the ‘Select all Columns on drop’ option within the ‘Options’ dialog (see the Options section).

#### Adding Column Aliases

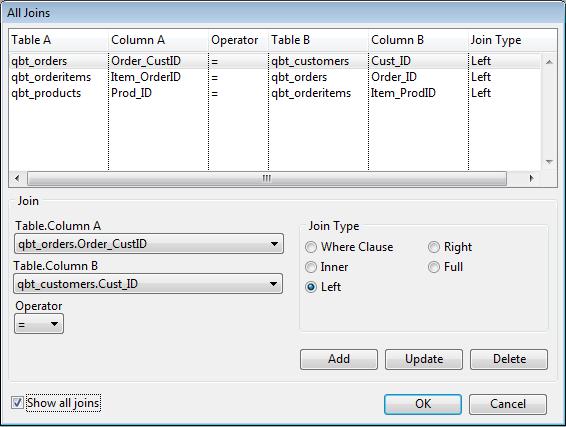
You can add an alias for each selected column in the ‘Columns’ tab of the lower tab pane by clicking in the ‘Alias’ column of the current line.



##### You can reorder columns in this pane by dragging and dropping column names in the list.

#### Creating Joins

To construct your queries, you can create joins between tables either using drag and drop in the main SQB design area or using the Table Joins dialog. To create a join, drag a column name from one table and drop it onto the column name within the table you wish to join with. Alternatively, you can right-click on a table to open the ‘Table Joins’ dialog from the context menu.



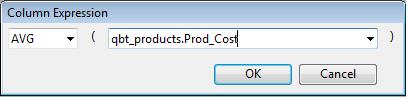
##### The ‘Table Joins’ dialog lets you modify joins, set the operator and type, re-order using drag and drop and switch columns using the context menu.

Joins may also be deleted via the context menu of the joined column in the main design area and Line and style preferences can be set via the ‘Options’ dialog.

You can set the default Join type for queries using 'Join Type' option in the Joins window, which can be opened by right-clicking on a Join and selecting ‘Joins’.

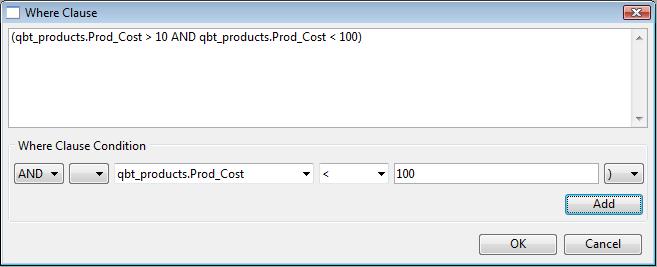
#### Adding Column Expressions

In addition to specifying aliases, the ‘Columns’ tab of the lower tab pane lets you add expressions to a query via the context menu. Selecting ‘Add Expression’ from the context menu opens a dialog where you can add common SQL expressions such as AVG, COUNT, MAX, MIN and SUM.



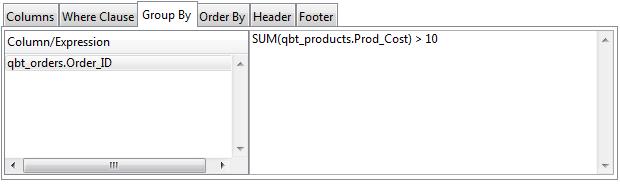
#### Adding a Where Clause

You can add a Where clause condition by dropping a column from a table onto the ‘Where Clause’ pane of the lower tab pane. When you drop a column the ‘Where Clause’ dialog is opened automatically and the column is pre-selected. You can also right-click on the Where Clause pane to Edit the column conditions for the current query.



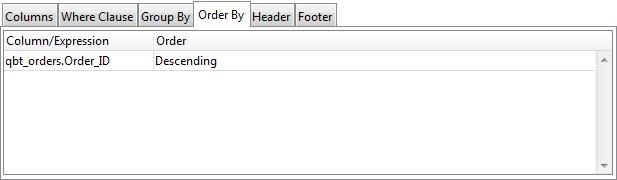
#### Adding a Group by Clause

You can add a Group By clause, to group selected rows together to return a summary of information, by dropping a column name onto the ‘Group By’ pane of the lower tab pane. You can also add a Having Clause to restrict the rows used by the Group By clause either by dropping a column from a table or via the context menu.



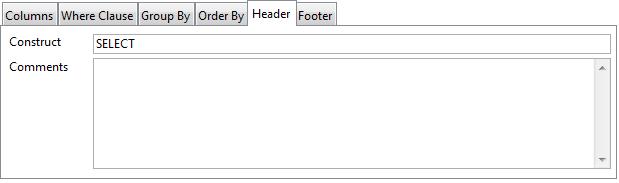
#### Adding an Order by Clause

You can add an Order By clause by dropping columns onto the ‘Order By’ tab of the lower tab pane: when you drop a column Descending order is selected by default but you can change it to Ascending. You can add Expressions to the Order By clause using the context menu by right-clicking on the Order By pane.



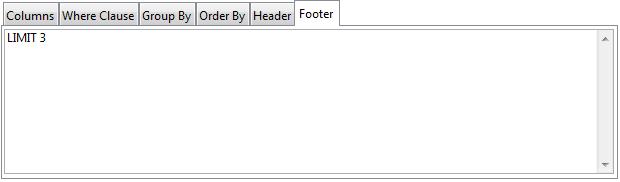
#### Modifying the SELECT Construct

The ‘Header Tab’ in the lower tab pane lets you enter an alternative construct, such as ‘SELECT DISTINCT’ and where supported ‘SELECT TOP 100’. This tab also lets you add comments to precede the generated SQL Statement.



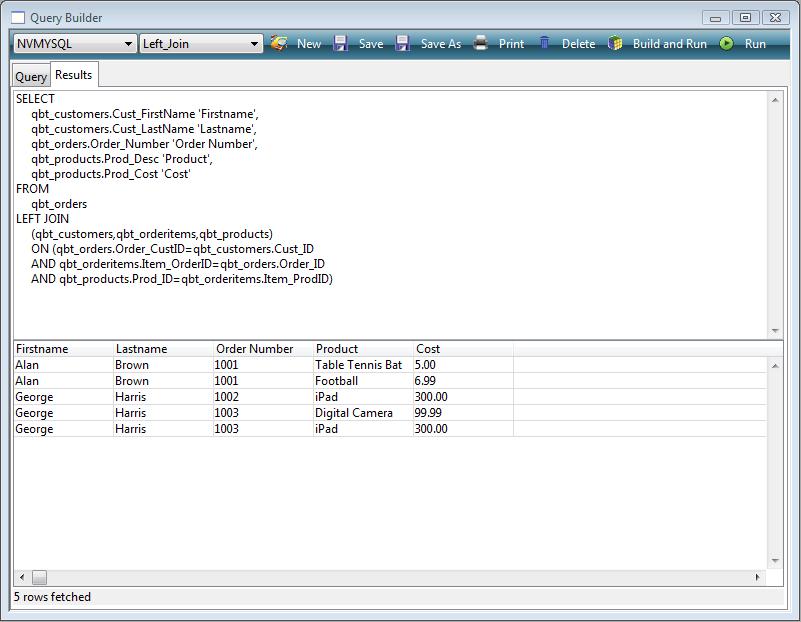
#### Adding Extra Query Text

The ‘Footer Tab’ in the lower tab pane lets you add any additional query text to be appended to the generated SQL Statement.



### Running a Query

You can run the current query from the main toolbar in the SQB window using the ‘Build and Run’ or ‘Run’ option. Both buttons switch the main pane to the ‘Results’ pane showing the SQL script and results generated by the Query. You can make changes to the SQL script generated if required and the query can be executed again using the ‘Run’ button. Note that using the ‘Build and Run’ option will rebuild the statement from the saved query text and therefore overwrite any changes you may have made.



##### Any errors which occur are reported in the status bar. If the full error text is not displayed, you can click on the status bar to open a dialog showing the full error text.

You can resize the font in the Query Builder results pane using the Ctrl+/Ctrl- shortcut keys.

### Saving a Query

You can save a query using the ‘Save’ or ‘Save As’ toolbar option. When using ‘Save’ option for the first time, or the ‘Save As’ option, you can add a description for the query. The description is also available to view/edit via the ‘Query Info’ option of the main context menu. Once saved, a Query is added to the list of Queries available in the dropdown list in the main SQB toolbar.

### Deleting a Query

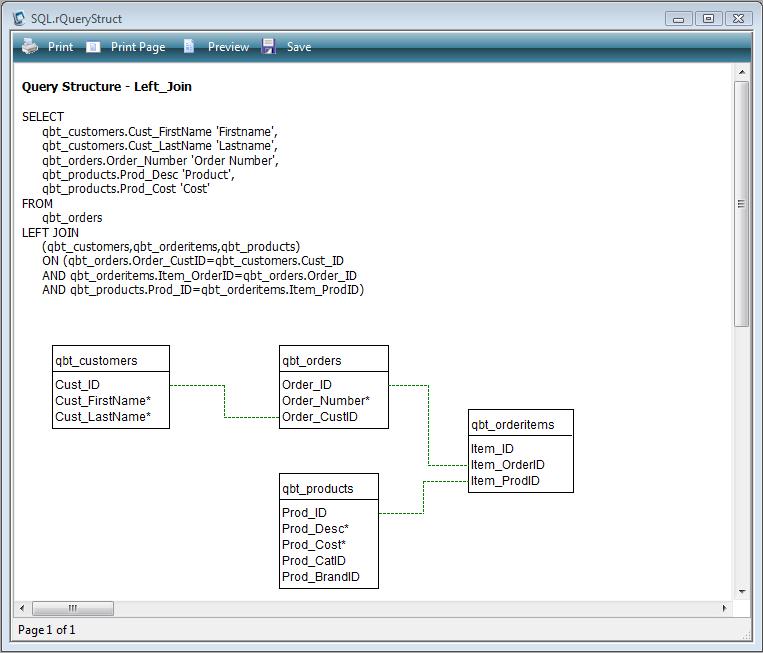
The Delete button in the SQB toolbar button lets you delete the currently selected query (you must confirm the deletion).

### Query Reports

There are two types of report available via the Print button on the main toolbar or the context menu opened by right-clicking on the Query design pane: both these options open the Print Query dialog that allows you to print either the Structure or the Results of the current query. In addition, you can include the generated SQL Script from the last executed SQL query in both reports.

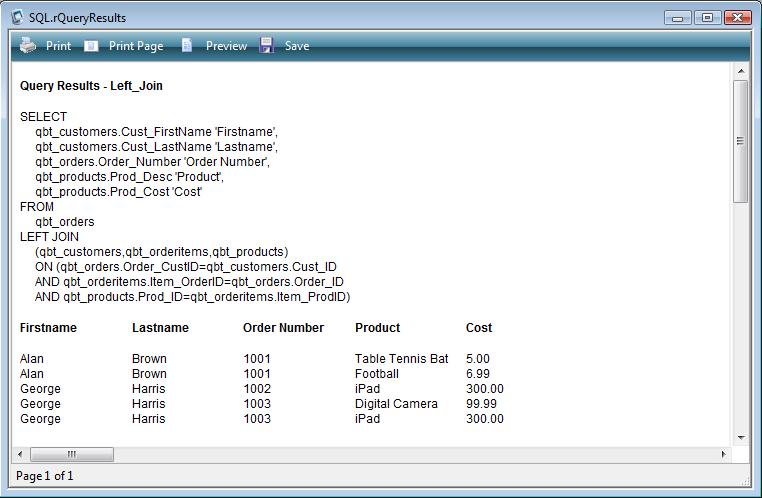
#### Query Structure Report

The ’Query Structure’ report shows the tables and their joins where the selected columns are represented by an astrix ‘\*’.



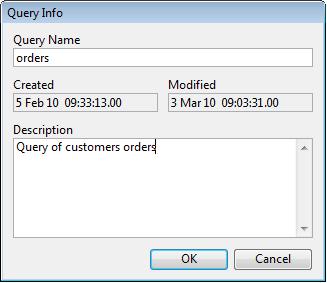
#### Query Results Report

The ‘Query Results’ report shows the results of the last executed SQL query. Column widths reflect those of the results pane and may therefore be adjusted prior to printing.



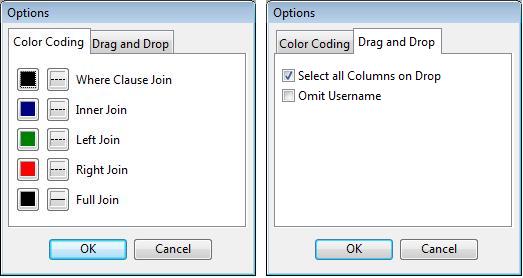
### Query Info

The ‘Query Info’ dialog available from the main query context menu displays information about the current query. You can change the name and description of query.



### Options

You can set various options for the SQL Query Builder in the ‘Options’ dialog which is available from the context menu on the main query window. You can specify the line and color styles for joins, and you can set preferences for dragging and dropping during table creation.



For databases where table names are prefixed by a username, the Omit Username option is particularly useful when running the same query against different databases where the username is different, but the table and column names are the same.

### Creating a Query Class

You can create an Omnis Query class based on the current query. To do this, you can drag the current query from the main query design window on the right and drop it on to an open library in the main Studio Browser. When you create a query class in this way, all the additional query text is added to the class, together with any associated Omnis Schema classes in order to map to the server tables in the query. Note that this feature is restricted by the same limitations as a query class and therefore only supports ‘Where Clause’ joins.

### Creating a Table class

You can create a table class from the current query using the 'Create table class' option which is available on the 'Other' toolbar menu option: the option also gives you the option to create a window class and/or a remote form for viewing the data via the new table class.

### Exporting Data

You can export the results data using the 'Export Data' option available on the 'Other' toolbar menu option.

### Creating a Statement Block

You can copy the generated SQL Script to the clipboard in a ‘Begin Statement’, ’Sta:’, ‘End Statement’ block by right-clicking on the ‘Results’ pane and selecting ‘Create Statement’ from the context menu. This option is also available on the Other menu option.

You can paste the statement into an Omnis method, providing an alternative method of executing your query in an Omnis Studio library where a query class is not appropriate.

### SQL Query Builder App

A runtime or end user version of the SQL Query Builder application is available on request. Please contact your local Omnis sales or support office for details.

## Version Control System

Note that some editions of Omnis Studio do not allow access to the Omnis VCS.

The Omnis *Version Control System* (VCS) lets you manage and revise Omnis libraries and other application components systematically. In a team environment, with several people working on the same application at the same time, you need to ensure that only one person can change a particular component at a time.

Using the Omnis VCS you can control the development of your Omnis applications, or any other project involving many different files such as Internet or Intranet applications. Specifically, the VCS can manage Omnis libraries or their classes, external components, DLLs or Code Fragments, Omnis data files, text or word processor files, Html and web server files, or any other types of file required in your Omnis application.

The Omnis VCS has an easy-to-use environment that allows you to check-in and check-out components, plus it had a useful tool that lets you compare different versions of the same library or different revisions of the same component.

The [Omnis VCS](15vcs.html) is described in detail in a later chapter in this manual.

## Auto Updates

You can perform updates or any other changes to your Omnis desktop application or folder structure upon restarting Omnis by adding a script to the Omnis data folder. You can use the Auto Update feature to update any file in the Omnis Studio tree, including the Omnis executable or program file itself: however, the studiorg.exe file cannot be updated under Windows.

To enable the Auto Update feature, write a batch file under Windows called **update.bat,** or on macOS or Linux create a bash script called **update.sh,** and add it to the Omnis data folder, i.e. the folder containing the Studio, Startup, and Welcome folders.

When Omnis starts up it will execute the update script automcatically at startup, before loading any external components, externals or libraries. If the call to run the script is successful, Omnis then deletes the update.bat/sh file.

When running on Windows, Omnis incorporates a request to run this as part of the existing UAC support implemented via studiorg.exe. In this case, you will get a UAC prompt if the update script needs to run, or if the registry needs updating for some reason, or if both updates and registry updates are required.

The Windows batch file or Unix script must have Execute permissions set in order to run. You can do this in the Properties of the file or via the file system. To do this in your code on macOS or Linux you can use the $setunixpermissions() fileops function:

###### If sys(6)= 'U' ## macOS or Linux Do fileops.$setunixpermissions( scriptPathName,'-rwxr--r--' ) ## set file to execute End if

### Update Feedback

When running the auto update script some feedback that the script is running can be provided in a console window. To enable this, you must place a file (which can be empty) named ‘showconsole.txt’ in the same directory as update.bat. When this file is present, a console window is displayed while update.bat is running.

### Example

The following example shows typical commands that could be used in a batch script: the commands download two new xcomps from a server (xcomp1.dll and xcomp2.dll), and store them in a folder specified by con(sys(115),pathsep(),updates,pathsep(),xcomp):

###### copy /y <studio data folder path>\updates\xcomp\xcomp1.dll <studio program folder path>\xcomp\xcomp1.dll del <studio data folder path>\updates\xcomp\xcomp1.dll

###### copy /y <studio data folder path>\updates\xcomp\xcomp2.dll <studio program folder path>\xcomp\xcomp2.dll del <studio data folder path>\updates\xcomp\xcomp2.dll

When a path has a space or spaces in it, e.g. Program files, the path should be enclosed in quotes:

“C:\Program Files\Omnis Software\OS8.1UPGRADETEST\xcomp\Dummy.dll”

## External Class Editor

$editor and $editordata are properties of all class types except system tables – in previous versions these properties were only available for object classes. The properties allow you to specify your own editor and to access the data for an Omnis class. The definitions of these properties are:

1. **$editor**  
   The name of the add-in tool library used to edit the class
2. **$editordata**  
   Editor data stored with the class. Typically used by the library identified by $editor

By default, these properties are not visible in the Property Manager, therefore to make them visible, edit the show\_editor item under “properties” in the config.json file:

###### "properties": {

###### "show\_editor": true

###### },

In addition, you should note that $editordata will only appear in the Property Manager when used in conjunction with the Notation Inspector.

$editor overrides the default editor for a class. Omnis calls $exectool for the specified add-in library, passing it a single parameter which is an item reference to the class.

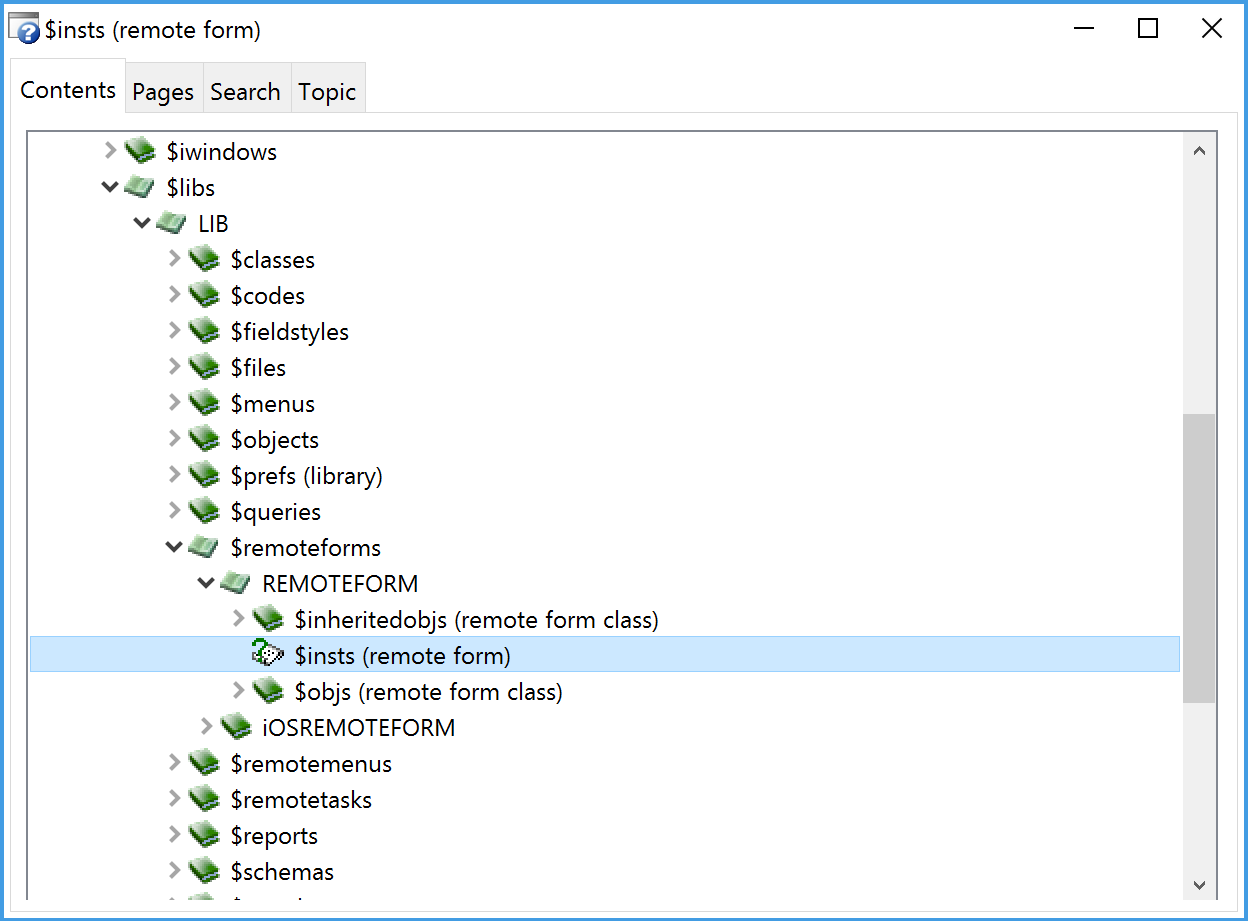
Note that the specified editor is not used when using find and replace – instead, the normal editor for the class opens.

There is a Tech Note [TNID0007](http://developer.omnis.net/technotes/tnid0007.jsp) on the Omnis website that shows how you can create an alternative schema editor.

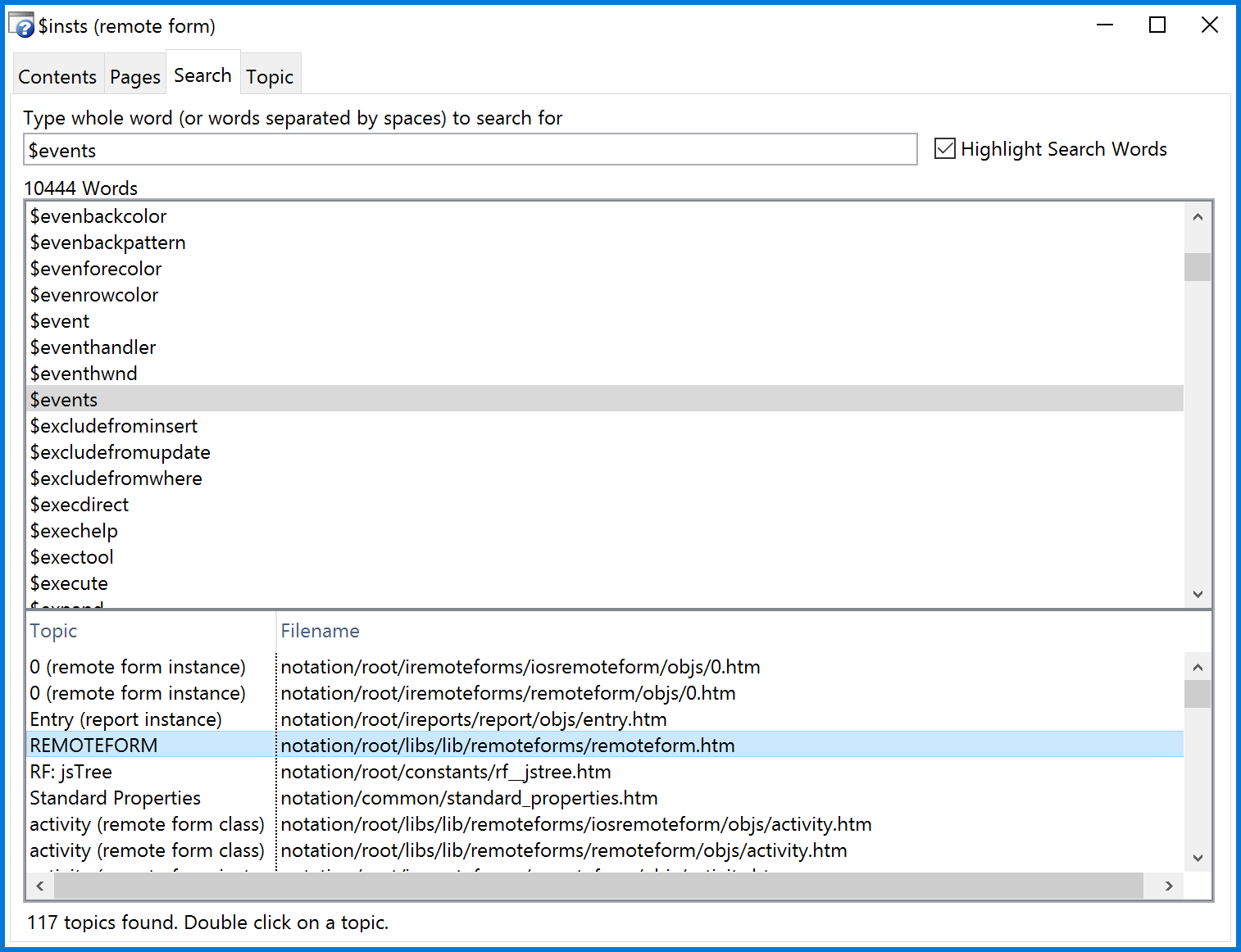
## Omnis Help

In design mode, Omnis provides many different types of help: tooltips and helptips over toolbar controls: help text for the main menus shown in the status bar at the bottom of the Omnis application window: plus when you’re writing a method the Code Assistant will popup automatically to display the appropriate variable name, notation (property or method name), and the correct parameters for the current object or context in your code.

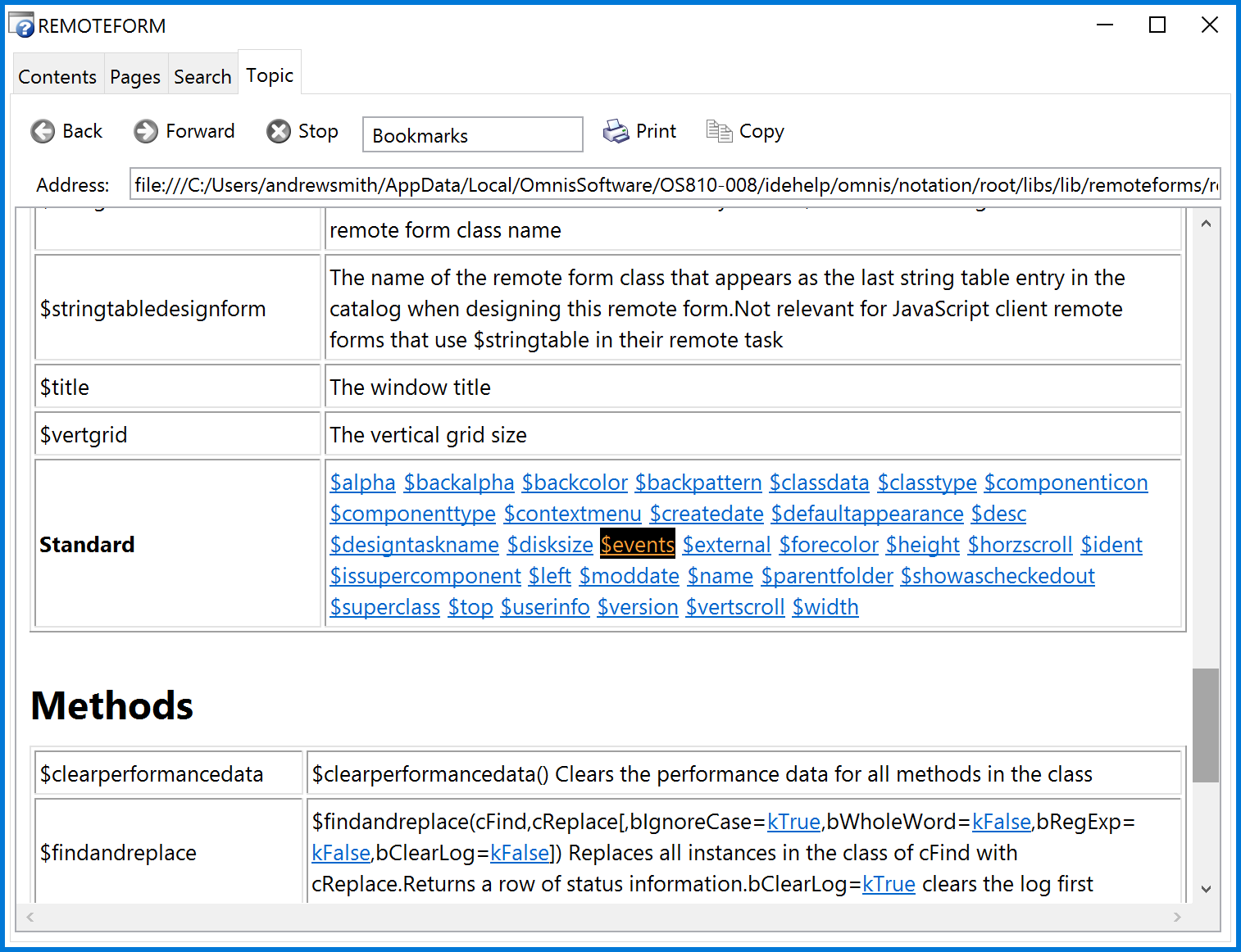
In addition to these types of help, there is a fully context-sensitive **Help system,** available by pressing F1 or via the Help menu. Omnis also provides ‘What’s This?’ help for individual functions and commands while you’re working in the Method Editor: ‘What’s This’ type help is *not* provided for the main tools in the Omnis IDE. (Note that under Windows Vista you need to press the Alt key to show the main Omnis menus, including the Help menu, or you can show it by Right/Ctrl-clicking on the Omnis toolbar and selecting the Menu bar option.)



##### Under the Contents tab you can drill down to the topic or object type you’re looking for: you can double-click on an item in the tree to load the page.



##### Under the Search tab you can type a command, function, or method name and Omnis will provide a list of matching topics: you can double-click an item in the Topic list to load the page under the Topic tab (as shown below).



You can create your own Help system and add it to your own application using the Help Project Manager under the Tools menu. Note that the Omnis Help available in the development version of Omnis is located in the omnis\idehelp folder.

## Code Signed Omnis (macOS)

The Omnis Studio application package on macOS is code signed, which provides increased security for you and your end users. A signed application can be trusted to originate from the developer who signed it, and to not have been altered in any way by any third-party, therefore guaranteeing the authenticity of an application. Signed applications within macOS can automatically be granted permissions to perform actions, such as accessing services from the network and running built-in software such as AppleScript commands.

An application can only be signed if its code portion remains unchanged. For the Omnis application, the code portion is located in the Omnis package, e.g.:

Omnis\ Studio\ 8.1\ x64.app/Contents/MacOS/

### Firstruninstall and Application Support folders

Any files that are updated by Omnis must be stored as user application data located in the user's home directory, that is, in the Application Support folder:

~/Library/Application Support/Omnis/

To do this, when Omnis starts up it will check for the existence of a folder called ‘firstruninstall’ in the macOS folder in the Omnis package. Any items which are contained in this folder are copied by default to a folder in Application Support with the same name as the Omnis package, e.g.:

~/Library/Application Support/Omnis/Omnis Studio 8.1 x64

The copy will not occur if the destination folder already exists, therefore avoiding any files being overwritten.

This provides a mechanism to place all data folders and their contents into the ‘firstruninstall’ folder, e.g. icons, studio, startup. Once copied into Application Support they are only updated in that location and leave the original macOS folder unchanged and its signature valid.

### Updating Components

With the signed version of Omnis Studio, an external or JavaScript component can be added or updated in the user data folder. This allows the signed code part of Omnis to remain unaltered, so it maintains a valid code signature. For example, a standard component can be placed in the following folder:

~/Library/Application Support/Omnis/\Omnis\ Studio\ 8.1\ x64/xcomp

and a JavaScript component here:

~/Library/Application Support/ Omnis/\Omnis\ Studio\ 8.1\ x64/jscomp/

If the required folder does not exist it can be created by the user.

The user data folder is always searched first, so if a component with the same name exists in the code section of the Omnis tree the user version will be loaded in preference.

### Deployment

When deploying your own application, you can update the distributed files in the Omnis package to include your own libraries and components and to edit the name of the application. Those files placed in the firstruninstall folder will be treated as user data and will be copied to the Application Support folder.

By default, user data for each installation of Omnis goes into a subfolder of Application Support called “Omnis” and the name of the Omnis package is used to provide the folder for the individual installation.

So for example an installation here:

/Applications/Omnis Studio 8.1 Beta 3 x64.app

Will have a default user data location of:

~/Library/Application Support/Omnis/Omnis Studio 8.1 Beta 3 x64

To customize the subfolder, edit resource 25599, and to customise the installation folder, edit resource 25600. These resources are located in the Localizable.strings file for the language used, e.g.

/Omnis Studio 8.1 x64.app/Contents/Resources/English.lproj/Localizable.strings

Both entries are empty for default behavior.

"CORE\_RES\_25599" = "";

"CORE\_RES\_25600" = "";

After you update the Omnis package files, the package will need to be re-signed with your own signing identity. You cannot sign a file that has extended Finder information attributes, so these need to be removed before signing. This can be done recursively over the entire package by using the following command:

xattr -r -d com.apple.FinderInfo <package\_path>

For example:

xattr -r -d com.apple.FinderInfo /Applications/My\ Application.app

Signing you own application requires a code signing identity which can be generated by adding a development or production certificate via the Certificate section of the Apple developer member center. The machine where signing is to occur must have the certificate and private key installed. To list all valid code signing identities available on a machine, use the following command from the terminal:

security find-identity -p codesigning -v

Which will, for example, produce the following output with key and identity listed:

###### 1) 44FFBA8B7DFFB1AFFF36FD0613D6E5FC61FF8DFF "Certificate" (CSSMERR\_TP\_NOT\_TRUSTED)

###### 2) B3EF62FF18E0FFB83D3A8FF3672CF80EFF367FFF "Mac Developer: John Doe (24FFEXFF39)"

2 valid identities found

To sign the package use:

codesign -f --deep --verbose -s <identity> <package\_path>

For example:

codesign -f --deep --verbose -s "Mac Developer: John Doe (24FFEXFF39)" /Applications/My\ Application.app

If the command completes with no errors, a similar line to the following should appear in the output:

:signed app bundle with Mach-O thin (x86\_64) [com.myCompany.MyApplication]

The application is now signed and ready for deployment.

Do not subsequently alter the contents of the package as this will invalidate the signature.

You can verify the signature using the following:

codesign --display --verbose=4 <package\_path>

Which will list items such as the signing authority, signing time, etc.

### Patching a signed tree

If you wish to distribute an updated Omnis application (the program file), and replace the application in an existing signed Omnis tree, then this can be achieved by doing the following:

1. Replace the binary in the original signed tree with the new version.
2. Re-sign the Omnis tree with the same signing identity which you used to sign the original tree.
3. Take the patched binary out of the tree for distribution.

Components can be patched without re-signing into the xcomp and jscomp folders of the user data location, e.g.:

###### ~/Library/Application Support/Omnis Studio 8.1 x64

Always ensure the tree has a valid signature by running:

###### codesign --display --verbose=4

## Update Manifest Files (macOS)

On macOS if an Omnis deployment is replacing an existing installation using a simple drag and drop approach, that is, from a disk image to the Applications folder, files which already exist in the current user's Application Support folder will not be updated from the files in the firstruninstall folder of the new disk image.

If there are files which need to be patched, a mechanism needs to be found to remove the existing files from the user data location, e.g. run an update script. Therefore, Omnis now provides the use of *Update Manifest Files* to allow a deployment to specify the files in an existing set of user data which need to be removed so they can be replaced by newer files from the firstruninstall folder.

When Omnis starts it will read an integer deployment version number from a file called “version” in the Omnis application's macOS folder:

###### /Applications/Omnis\ Studio\ 10.0.1.app/Contents/MacOS/version

If this file does not exist then the deployment version number will be set to the Omnis internal build number (as returned by sys(123)).

Omnis will read the version of the user data from a hidden file (.version) in the root of the user's application data folder for the Omnis application.

###### /Users/<username>/Library/Application Support/Omnis/Omnis Studio 10.0.1/.version

If this file does not exist, the user data version is set to zero. If the user data version is lower than the deployment version, Omnis will check for updates that need to be applied.

The updates are specified in a set of files which should be placed in a folder called “manifest” within the Omnis application's macOS folder. Each file should be named for the version which specifies the changes. For example, if the new deployment is version 23071 there should be a file named 23071.

###### /Applications/Omnis\ Studio\ 10.0.1.app/Contents/MacOS/manifest/23071

The manifest file should contain the paths of each file or folder which needs to be updated for that version. Each path should be separated by a new line.

Therefore, if file 23071 contains:

###### studio/v40.lbs

###### startup/vcs.lbs

This will indicate that the Studio Browser and VCS are to be updated (removed from the user data) for version 23071.

There can be a manifest file for interim versions if updating an older version of Omnis. Therefore, if the user version is 23069, the deployment version is 23071 and the manifest folder contains 23070 and 23071 then both files will be used for updating.

If updates are applied, the hidden user data version is updated to the deployment version.

Note that the Omnis application deployment tree (code-side) still needs to be re-signed (and notarised) for each version of a deployment.

It is not recommended that individual files are updated in an existing signed tree (as this will invalidate the signature).

## Omnis data folder

Resources 25599 and 25600 can now be used to specify the Omnis data folder on the Windows platform. You can edit the omnisdat.dll string table with a resource editor and modify 25599 and 25600 to be used to specify the sub-folders of the appdata directory. The Omnis data folder becomes:

<appdata folder>\<resource 25599>\<resource 25600>\

If resource 25599 is not empty, resource 25600 must also not be empty.