What's New in Omnis Studio 5.2

TigerLogic Corporation
February 2012
15-022012-03

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

© TigerLogic Corporation, and its licensors 2012. All rights reserved.
Portions © Copyright Microsoft Corporation.
Regular expressions Copyright (c) 1986,1993,1995 University of Toronto.

© 1999-2012 The Apache Software Foundation. All rights reserved.
This product includes software developed by the Apache Software Foundation (http://www.apache.org/).

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

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

Apple, the Apple logo, Mac OS, Macintosh, iPhone, and iPod touch are registered trademarks and iPad is a trademark of Apple, Inc.

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

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

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

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

Portions Copyright (c) 1996-2008, The PostgreSQL Global Development Group
Portions Copyright (c) 1994, The Regents of the University of California

Oracle, Java, and MySQL are registered trademarks of Oracle Corporation and/or its affiliates

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

Acrobat is a registered trademark of Adobe Systems, Inc.

CodeWarrior is a trademark of Metrowerks, Inc.

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

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

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

Table of Contents

About This Manual

This document describes the new features and enhancements in Omnis Studio 5.2.

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

If you are new to Omnis Studio

If you are new to Omnis Studio you should start by reading the *Introducing Omnis Studio* manual and then the *Omnis Programming* and *Extending Omnis* manuals. All the Omnis Studio manuals are available on the product DVD and to download from the Omnis website (www.tigerlogic.com/omnis).

What's New in Omnis Studio 5.2

Omnis Studio 5.2 enhances your choices for development and deployment and extends your Omnis apps to new platforms, new devices and new markets. With the introduction of a JavaScript based client, Omnis Studio will allow you to create applications that will run on virtually any web-enabled or mobile device via the browser on the device. Add to this support for the SQLite database, Omnis Studio provides a powerful and flexible development environment for creating all types of desktop and mobile applications for many different markets.

- **JavaScript Client**
  the new JavaScript Client will run your Omnis apps in a web browser on a desktop PC, tablet computer, or Smartphone, without the user having to install a plug-in; to enable this functionality there is a new JavaScript enabled Remote Form class type and a new set of JavaScript components available in the Omnis Component Store
- **SQLite**
  there is a new Omnis DAM to support connections to SQLite, a very popular database which is embedded into a whole range of applications on desktop and mobile devices; the new DAM supports SQLite3

New Welcome Window

We have updated the Welcome window and the built-in Tutorial. The Welcome window opens when you first start Omnis Studio 5.2 and contains a set of sample apps that use the new JavaScript Client. The new Welcome window can also be opened by clicking on the **New Users** button on the main Omnis menu bar.

The new sample apps in the Welcome window cover a range of business and home uses and include the following:

- **Contacts** manager for recording information about colleagues or friends
- **Todo** app for handling tasks in the office or home,
- **Holidays** app for managing annual leave requests,
- **Timesheets** app for recording time-based tasks,
- **Guestbook** app for recording feedback from your website visitors,
- **Webshop** app which has a product catalog and shopping cart.

We urge you to examine these apps and in particular try out the new JavaScript forms that they all use and examine the code: you can try the apps in your desktop browser or in the

web browser on your mobile device. (See the *Testing JavaScript Remote forms* section for details of how to test remote forms.)

In addition, we have updated the built-in Tutorial taking advantage of the new JavaScript Client and SQLite. The updated tutorial now uses a SQLite database and shows you how to build a remote form for the new JavaScript Client.

Installation

Please see the Readme txt and Install.txt files accompanying the Omnis Studio 5.2 release for details about installation on different platforms, and for any last minute release notes.

Library Conversion

When you try to open an existing library in Omnis Studio 5.2, Omnis will attempt to convert your library. We therefore strongly recommend that you make a secure backup of all libraries and datafiles before opening them in Omnis Studio 5.2.

Development Serial Numbers

You will require a new serial number to run the Development version of Omnis Studio 5.2.

Omnis App Server Deployment Licensing

There are new Omnis App Server deployment licenses for running Web and/or Mobile apps in the JavaScript Client. Please refer to: www.tigerlogic.com/omnis, or contact your local sales office, for more information about licensing.

Omnis App Server Activation

When you serialize an Omnis App Server, the server will attempt to activate itself: you must be connected to the Internet for activation to occur. There will be a short delay while activation occurs, and if successful your Omnis App Server will be licensed and ready to use. Activation only applies to Omnis App Server Licenses that allow 3 or more users, including the unlimited server license. Without activation, the Omnis App Server will not run.

If activation fails, for example, if you are not connected to the Internet or due to some other communication error, you will be requested to complete activation manually. Instructions on how to do this are provided on screen.

If you attempt to activate your Omnis App Server and your server license is in use on another server, activation will not succeed and the Omnis App Server will not run.

To allow you to switch your license to another server, you can deactivate your current server using the Deactivate option in the File menu in the Omnis App Server. Having deactivated your current server, you can then serialize the new Omnis App Server and activation should succeed.

Contact your local sales office for further information about activation.

JavaScript Client

Omnis Studio 5.2 includes a new JavaScript based client which will allow you to run your Omnis applications in a browser on virtually any platform or any device, including Desktop PCs, Web-enabled TVs, Tablet computers, and Smartphones. Assuming the computer or device has a web browser, which is enabled to execute JavaScript, you can run the new JavaScript Client. Specifically, your end users will be able to run your Omnis apps *without having to install a plug-in* or any other software: you simply point them to a URL and they're up and running!

The JavaScript Client uses scripting compatible with HTML5 and CSS3 to enable the new client interface which is supported in most modern browsers including Google Chrome, Apple Safari, and Microsoft Internet Explorer. If for some reason JavaScript or Client Scripting is disabled in your browser the new JavaScript Client will not work. You can enable JavaScript via the Internet Options in your browser, usually under the Security Settings (for IE) or Content Settings (in Chrome).

Furthermore, you don't need to know anything about JavaScript to create the new JavaScript remote forms or use the new JavaScript components. All the JavaScript needed to render the new JavaScript enabled forms and components is provided for you and is executed automatically in your browser when required. In design mode in Omnis, the components you use to design the new JavaScript remote forms map directly to the JavaScript controls that are displayed in your or your end user's browser.

Web Client Compatibility

Existing remote forms that use the Omnis Web Client plug-in cannot be used in the new JavaScript Client. The new JavaScript remote forms use a completely new set of components and are not compatible with the existing Web Client plug-in.

There is a tool in the Studio Browser that will help you migrate Web Client based remote forms to the new JavaScript based forms, but due to the differences between these form types, the migration process is not complete: see later in this manual in the section 'Web Client Form Migration' for more details about the form migration tool.

Note there is a new property of a remote form called $client that determines which client a remote form uses: see the next section *JavaScript Remote Forms* for more information about setting $client.

Can I still use the Web Client and iOS Client?

The Omnis Web Client plug-in and the Omnis iOS Client are still available in Omnis Studio 5.2 and will be supported in the foreseeable future. The existing client plug-ins provide a native look-and-feel on their respective platforms and will continue to provide a familiar experience for users on those platforms.

The new JavaScript Client provides a browser and device independent platform for your Omnis applications which you may find appropriate for deploying your applications going forward.

Windows Mobile Support

Developers should note that we intend to remove support for the Omnis Windows Mobile client in the next version of Omnis Studio. We recommend that you migrate all Windows Mobile based apps to the new JavaScript Client which is supported on all Windows Phone based devices and many other mobile platforms.

JavaScript Apps Gallery

There is an Apps Gallery web page on the Omnis website containing example JavaScript applications created with Omnis Studio 5.2, and showcasing many of the new JavaScript Components. You can find the Apps Gallery at: www.tigerlogic.com/omnis

JavaScript Remote Forms

Designing a JavaScript based Remote form is virtually the same as designing a remote form in previous versions of Omnis Studio. When you are ready to test a JavaScript form, you can use the "Test Form" option (Ctrl-T) and the Remote form will open in your browser. The remote form and the components you have added to the form are rendered in the browser using JavaScript.

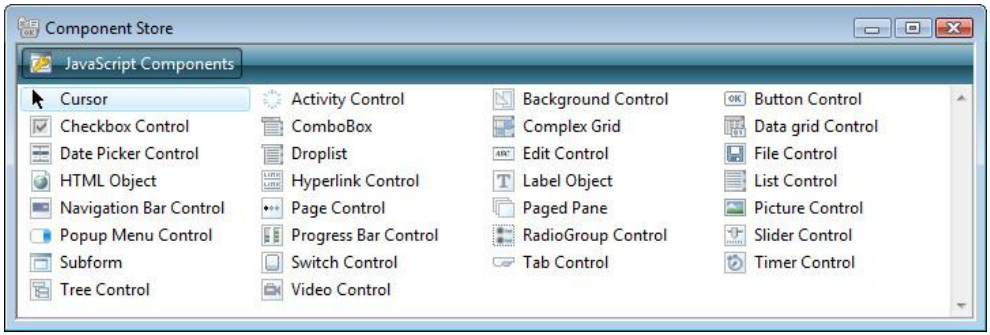
Creating JavaScript Remote forms

The JavaScript Client functionality is enabled by setting the $client property in a new remote form. The options for the $client property include kClientJavaScript for the new JavaScript Client, as well as kClientPlugin or kClientiOS which enable remote forms for the existing Omnis Web Client plug-in or iOS Client, respectively.

To create a new JavaScript remote form

- Click on **New Class,** then **Remote Form** in the Studio Browser and name the new Remote Form

When you use this method to create a new Remote form, the $client property is set to **kClientJavaScript** automatically, and the Component Store contains a new set of components for the JavaScript Client: see the *JavaScript Components* section below for a description of all the new components.



If you want to create a remote form for the Omnis Web Client plug-in or iOS Client, you will need to reset the $client property in the new remote form. When you set the $client property of a remote form, the contents of the Component Store will change to reflect the type of client you have chosen.

Once you have set $client and added some components to the form, the $client property is grayed out and you can no longer change the property in that form. You can only change $client when the remote form is empty.

Remote form name

Like any other class in Omnis, the name of a new Remote form can be anything you like but would normally take account of its function within your application. The class name does not have to conform to any convention other than any conventions you may like to use in your application to identify different class types: so for example, your remote forms could be prefixed with “rf”.

Note that the name of the Remote form class, plus the .htm extension, is used as the name of the HTML file which is created when you test your remote form in design mode. Therefore you should restrict any characters used in the name of your Remote form to only those normally allowed in a web context, or to be sure of removing all possible conflicts, use alphanumeric characters only. For example, a remote form name cannot include the hash symbol (#) or other special symbols since this may cause unexpected results in the browser, or in the case of #, the remote form may not open in test mode at all.

Web Client Form Migration

Omnis Studio 5.2 includes a tool to allow you to migrate remote forms that use the existing Omnis Web Client (plug-in) to the new JavaScript based remote forms. Due to the differences between these form types, the migration process is not complete, that is, not all the existing Web Client controls exist in the JavaScript Client, so you will need to update some of the form objects yourself. The form migration tool creates a copy of your old remote form class and places the new JavaScript form inside a folder within your library.

The **Migrate Forms** option is available in the Studio Browser when a library is selected. When you click on the option, the migration window will open showing all your open libraries and all the Remote forms in each library. The **Options** menu bar option allows you to setup the migration parameters, including the location of the new remote form classes (a

folder in your library called JSFormsFromMigration), the location of the migration log files (default is JSMigrationLogs folder in the main Omnis folder), and the mapping of Web Client controls to the new JavaScript controls.

Control migration mapping

The migration process generates a new remote form with the equivalent JavaScript controls, if they exist. If there is no corresponding control, the migration tool creates a place holder control in order that any methods associated with the original Web Client control are not lost. For example, the Button Area control does not exist for JavaScript remote forms, therefore any Web Client Button Areas are migrated to standard Buttons and the original method from the button area is placed behind the new pushbutton. Heading lists are migrated to Data Grids, while Icon Arrays and Sidebars are migrated to standard List fields.

The **Objects** tab on the Options window allows you to change the control migration mapping, but for most case you should use the suggested migration mapping and then modify the place holder controls manually. In some cases, you may need to significantly change the controls and methods in your new Remote forms in order for them to function correctly in the JavaScript Client.

$enablesenddata property

In older versions of Omnis Studio, the $senddata() method could be used to control when form data was sent to remote forms displayed in the Omnis Web Client; the $enablesenddata remote task property was introduced to enable or disable the $senddata() method. However, the new JavaScript Client handles when the form data is sent to the client automatically, so the $senddata method is not required. If the $enablesenddata property is enabled in any of your remote tasks in your application, the Web Client migration tool will disable it and add a note to the log. If you try to open a form controlled by a remote task with $enablesenddata enabled there will be an error on the client.

Remote Tasks

In order to test or run your remote form in a browser, your library should contain a Remote Task and the $designtaskname property of your remote form should be set to the name of a remote task in the current library. When you create a new remote form from the Studio Browser, and if your library contains a single remote task, the $designtaskname property will be set to the name of that remote task automatically. If your library does not contain a remote task, you will have to create one manually and set the $designtaskname property of your remote form to the name of the new remote task. If your library has multiple remote tasks, and you create a new remote form from the Studio Browser, the $designtaskname property will not be set, so again you will have to assign this manually before you can set the form.

Testing JavaScript Remote forms

You can test a JavaScript remote form using the **Test Form** option, available in the remote form either by right-clicking on the background of the form and selecting Test Form, or by pressing Ctrl-T. The Test Form option will open the remote form in a web browser specified as the default browser on your development computer; the remote form will open

in a new window or separate tab. You can switch back to Omnis and continue to change your remote form, and use Ctrl-T at any time to test your form. You can set breakpoints in your code, so when you test your remote form using the Test Form option or Ctrl-T and a breakpoint is encountered, control will pass back to Omnis and debugger/method editor will open at the breakpoint; this allows you to easily debug your code in the live form.

When you test your remote form using Test Form (Ctrl-T) an HTML page is created for you automatically containing the JavaScript Client and all the required parameters to allow you to open your Omnis app in a web browser. The test HTML file is located in the HTML folder under the main Omnis folder, and can be used or incorporated into the other web pages on your website when you are ready to deploy your application. The name of the test HTML file will be the same as your remote form class name plus the .htm extension. The test HTML is based on a template file which is also located in the HTML folder: see below for more info about the template.

The URL for the test HTML page will be something like the following:

```
http://127.0.0.1:51452/jschtml/<remoteformname>.htm
```

The test URL contains the IP address of your Localhost (127.0.0.1), the port number of your copy of Omnis Studio, a reference to the HTML folder, and the name of the test HTML. The port number during testing will be the port number specified in the $serverport Omnis preference, or if this is not specified (it is empty by default) a port number is selected randomly from the available ports on your computer.

Note if you try to open or navigate to the test URL from your browser history it may not work: in this case such a URL may not have the correct port setting since the port number is assigned dynamically during testing if the $serverport property is empty and therefore may be different from one session to another.

In addition, you cannot open or test your remote form by opening the test HTML in the template folder: again your browser will not have the correct URL to load the test HTML file.

Omnis Studio and your library *must be open and running* to test your remote form.

Testing your remote form on a mobile device

To test your remote form on a mobile device or any other computer, assuming those devices are within the same local network (LAN/WLAN) as your development computer, you can enter the test URL into the web browser on your device but replace the Localhost IP address (127.0.0.1) with the IP address of your development computer. For example, reusing the test URL above and replacing the IP address, the following URL could be used on a mobile device such as a phone or tablet computer:

```
http://194.131.70.184:51452/jschtml/jsMain.htm
```

You can use the ipconfig command to find the IP address of your development computer, via the Command prompt on a PC or the Terminal on a Mac.

Default web browser

When you test your remote form using the Test Form option (Ctrl-T) it is opened in the default web browser on your development computer. If you want to test the form in another

browser on your computer you can copy the test URL and paste it into another browser (note the port number may change from session to session).

If you want to override the default action for the Test Form option, you can specify the name and path of an alrternative browser in the $webbrowser Omnis preference (edit the Omnis preferences via Tools>>Options). If you want to specify which browser the Test Form option will use specifically for displaying JavaScript remote forms, you can specify the name and path of the browser in the $jswebbrowser Omnis preference. If both these preferences are empty, then the default browser on your development computer will be used for testing remote forms.

Template HTML file

The test HTML created when you use the Test Form or Ctrl-T option is based on a template file called 'jsctempl.htm' which is located in the HTML folder under the main Omnis folder. When you press Ctrl-T a copy of the template file is made and the individual parameters for your remote form are written to the test HTML file: this occurs every time you test your form to ensure the test HTML file is up to date and has the correct parameters. Existing users will note that this is a change in behavior: in previous versions of the Omnis Web Client the test HTML file was created the first time you tested a remote form and was not rebuilt each further time you tested the form.

Remote task and form instances

As with the Omnis Web Client in previous versions of Omnis, when the JavaScript Client first connects to the Omnis Server, Omnis creates an instance of the Remote Task Class associated with the Remote form class to which the client is connecting (and specified in the $designtaskname property of the remote form). Once the remote task class has been instantiated, next the Remote form instance is created. The $construct method of first the Remote task and then the Remote form are run, so these methods can include any code you want to run when the form is opened.

Contruct row variable

When a form is opened and the Remote task and Remote form instances are created, Omnis passes a parameter variable of type Row to the $construct() method of the Remote task and then the Remote form (in that order). This parameter variable contains a column for each parameter of the JavaScript Client object instantiated on the client: therefore, it will include columns for OmnisLibrary and OmnisClass (as defined in your HTML file), as well as extra columns containing additional information about the client object.

The construct row variable will contain the following columns and typical values:

Column	Description
OmnisLibrary	<OmnisLibrayName>
OmnisClass	<RemoteFromName>
appName	The navigator.appName of the client, i.e. the browser application name, such as “Microsoft Internet Explorer” or “Netscape”
userAgent	The navigator.userAgent of the client, which usually contains the browser type and version
Flags	Currently indicates if the client supports animation: 1 means the browser does support animation, zero means that it does not
OmnisPlatform	JSU, the JavaScript Client
JSscreenWidth	The screen width of the client, e.g. 1280 for desktop PC
JSscreenHeight	The screen height of the client, e.g. 1024 for desktop PC
JSscreenSize	The initial setting of $screensize for the client, e.g. kSSZDesktop
JStimezoneOffset	offset from UTC in seconds, e.g. “120” for clients on UTC+2
ClientLocale	the Locale language setting of the client, e,g. “en_GB” for clients in the UK
param1, 2, .. 9	Up to 9 pre-defined custom parameters called param1, param2, etc, which you can add to the JavaScript Client object in your HTML page; you can add custom parameters prefixed with “data-“ to send further values to the task or form construct method

The appName and userAgent columns return properties of the client browser and therefore allow you to determine which browser and version the client is using, such as whether it is a desktop or mobile browser.

Using the construct row variable in your code

If you want to use the values in this parameter variable, you can create a parameter variable of type Row in the $construct() method of your remote task or remote form which will receive the parameter variable when the task/form is constructed. To examine the values in the variable, you can set a breakpoint in the $construct() method of your remote task or remote form, open the form (using Ctrl-T), and Omnis will switch to the debugger allowing you to right-click on the variable to examine its value.

For example, many of the sample apps available in the Welcome window use the construct parameter variable to get the screen size of the client device in order to set the size and position of various controls in the initial remote form for the app. The following is the code for the $construct() method of the jsShop remote form in the **Webshop** app, which receives the screen size of the client device, and calls another method to setup the columns for a data grid control on the main jsShop remote form:

```
; $construct method containing a Parameter var called pRow of type
   Row; the form also contains an instance var iScreensize (Char)
Calculate iScreensize as pRow.JSScreensize
Do method setupSizes
Etc.
; code for setupSizes method
Switch iScreensize
Case kSSZjs320x480Portrait
   Do $cinst.$objs.pagePane.$objs.orderGrid.$::columnwidths.$assign(
        "150,50,50,70")
Case kSSZjs320x480Landscape
   Do $cinst.$objs.pagePane.$objs.orderGrid.$::columnwidths.$assign(
        "70,40,40,70")
Case kSSZjs768x1024Portrait
   Do $cinst.$objs.pagePane.$objs.orderGrid.$::columnwidths.$assign(
        "300,75,75,175")
Case kSSZjs768x1024Landscape
   Do $cinst.$objs.pagePane.$objs.orderGrid.$::columnwidths.$assign(
        "250,75,75,175")
Default
   Do $cinst.$objs.pagePane.$objs.orderGrid.$::columnwidths.$assign(
        "100,50,60,75")
End Switch
```

JavaScript Remote Form Properties

JavaScript remote forms have all the standard properties of a remote form as well as a number of properties specific to the new JavaScript Client.

Property	Description
$alpha	The alpha value for the form, a value 0-255, where 0 is completely transparent, 255 is opaque
$backalpha	The alpha component of the background color of the form, a value 0-255, where 0 is completely transparent, 255 is opaque
$designshowmobiletitle	Hides or shows the title bar; if you deploy the form without the window title bar you need to disable the title bar in the client configuration; see below; in addition, if you hide the title, you'll need to add 20 pixels to the $height property of the form
$designtaskname	You need to create a remote task and set this property to the remote task name; this is required to test the form
$events	Lists the events reported in the remote form: you have to enable specific events for the form, otherwise they will not be reported; see the Remote Form Events section below
$client	Specifies the client for the remote form; this is set to kClientJavaScript in new remote forms created from the Studio Browser to enable the JavaScript Client, otherwise it can be kClientPlugin (Web Client) or kClientiOS (iOS Client)
$screensize	Specifies the screen size and layout of the form, a constant: kSSZDesktop ($height and $width of the remote form are used to specify the size of the form on the client) kSSZjs320x480Portrait, kSSZjs320x480Landscape kSSZjs768x1024Portrait, kSSZjs768x1024Landscape Note $height and $width should be set for mobile sizes

Remote task properties

The remote task property $enablesenddata should be set to kFalse for all tasks controlling JavaScript remote forms since the $senddata() method is not implemented for JavaScript remote forms. The Form migration tool will set $enablesenddata to kFalse.

JavaScript Remote Form Methods

JavaScript remote forms have the following methods.

Method	Description
$beginanimations()	$beginanimations(iDuration[,iCurve=kJSAnimationCurveEaseInOut]) after calling this method, assignments to some properties are animated for iDuration milliseconds by $commitanimations()
$clientcommand()	$clientcommand(cCommand,wRow) allows you to execute the command cCommand on the client using the parameters in the row variable wRow; see below for client command examples
$commitanimations()	$commitanimations() animates the relevant property changes that have occurred after the matching call to $beginanimations()
$setcurfield()	$setcurfield(vNameOrIdentOrItemref) sets the current field on the client computer which is useful for data entry forms; when this is set, the focus is placed in the field on the client, and for mobile devices (and depending on the OS) the soft keypad may be initiated; $setcurfield('') removes the focus from the current field
$showmessage()	$showmessage(cMessage [,cTitle]) displays an OK message on the client using the specified cMessage and cTitle; you can create a line break in the message using //
$showurl()	$showurl(cURL[,cFrame,cWindowProperties]) opens the URL in a new window or frame on the client

$redraw and $senddata methods

Redraws are handled automatically for JavaScript forms and controls, so although JavaScript based remote forms and controls contain the $redraw() method, if it is called it does nothing. In addition, $senddata() is not implemented for JavaScript forms due to the modified redraw mechanism for this version. See the next section regarding Redraw.

Remote task methods

Together with the remote form methods, you can use the standard remote task methods in your JavaScript Client based apps, including $openform() and $changeform().

Custom methods

You can add your own methods to JavaScript remote forms and individual components using the Omnis method editor, as in previous versions of Omnis.

Client method execution

Client method execution is not implemented for JavaScript remote forms, therefore you cannot enable a method to Execute on Client in the Method Editor. Events and Methods triggered in the JavaScript Client are executed in your development copy of Omnis Studio, or, for deployed applications, on the Omnis Server. You can execute a number of commands on the client using the $clientcommand() remote form method: see the *Client*

Commands section below for more details. Client method execution will be added to a future version of Omnis Studio.

Redrawing JavaScript forms

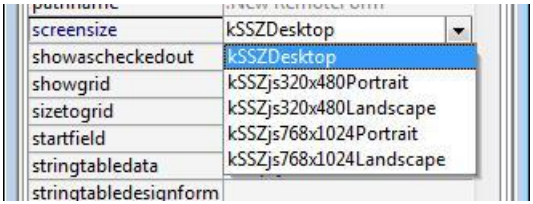
The redraw mechanism for JavaScript based remote forms in Omnis Studio 5.2 is different to remote forms in previous versions of Omnis. Redraws for the new JavaScript controls and remote forms are handled automatically and are optimized to work efficiently on a range of devices including desktop and mobile browsers.

In JavaScript based remote forms, controls are redrawn automatically when their data changes. A control is redrawn when all property changes have been completed as the result of executing the $event method for a control. The $redraw() method is therefore not required for JavaScript forms or controls, and if it is called, it does nothing.

The redraw mechanism for other types of remote forms (Web Client and iOS Client) and desktop windows remains the same as in previous versions. There is a new parameter for the redraw() method for the plug-in based remote forms: see later in this manual.

Screen Size and Orientation

You can design different layouts for the different screen sizes and orientations and the JavaScript Client will chose the most appropriate screen size and orientation for the current client (browser) automatically. The JavaScript remote form property $screensize specifies the screen size and orientation in the remote form and can be set to Desktop, and various Portrait or Landscape options for mobile devices.



When you create the remote form, you place the objects on the form, change the setting of $screensize, and reposition the objects for each screen size and orientation. In other words, each screen size/layout uses the same set of objects (and methods) and the single remote form class stores the position of the fields for each screen size/orientation setting.

The following screen sizes and orientations are available:

- **kSSZDesktop**
  for remote forms running in a web browser on a laptop or desktop based PC or Mac computer; in effect the screen size is unspecified and the $height and $width of the form *in design mode* is used to size the form in the browser
- **kSSZjs320x480Portrait** or **kSSZjs320x480Landscape**
  For mobile devices with screens 320 x 480 px (at 96dpi) in Portrait/ Landscape orientation

- **kSSZjs768x1024Portrait** or **kSSZjs768x1024Landscape**
  For mobile devices with screens 768 x 1024 px (at 96dpi) in Portrait/ Landscape orientation

When opening (constructing) a remote form, the JavaScript Client uses the most appropriate screen size and orientation stored with the form, for the screen size and orientation of the current device. If the user swaps from portrait to landscape, or back again, the JavaScript Client repositions the controls automatically, using the coordinates the new orientation stored in the remote form (assuming you have added a layout for each orientation). If for example, the user switches to landscape orientation, and you haven't added a layout for this orientation to the form, the portrait layout is used by default.

Form width and height

When $screensize is set to one of the mobile sizes/layouts, changing $width and $height for the remote form changes the width and height of the client area of the design window (the area where you can place controls), rather than the size of the design window itself. The design window adds scrollbars to the client area when necessary, therefore you don't need to enable the $horzscroll or $vertscroll for the tablet or mobile screen sizes. In addition, you can control whether the design window shows the mobile title bar, by setting the $designshowmobiletitle property for the form, although you may want to show this for most types of mobile applications.

In most cases, when specifying the width and height for the mobile screen sizes, you can set the $width and $height properties to match the exact coordinates in the current setting of $screensize, allowing for the mobile title bar which is 20 pixels high.

Remote Form Events

JavaScript remote forms report the following events:

<table>
<tr><td>evAnimationsComplete</td><td>The animation has completed
Parameters<table><tr><td>pEventCode</td><td>The event code</td></tr></table></td></tr>
<tr><td>evFormToTop</td><td>The remote form is about to become visible on the client
Parameters<table><tr><td>pEventCode</td><td>The event code</td></tr><tr><td>pScreenSize</td><td>A kSSZ... constant for the current screen size on the client</td></tr></table></td></tr>
<tr><td>evScreenOrientationChanged</td><td>The orientation of the screen displaying the form has switched between portrait and landscape
Parameters<table><tr><td>pEventCode</td><td>The event code</td></tr><tr><td>pScreenSize</td><td>A kSSZ... constant for the current screen size on the client</td></tr></table></td></tr>
<tr><td>evSubFormToTop</td><td>An existing remote form, contained in a subform that has $multipleclasses set to kTrue, is about to become visible on the client
Parameters<table><tr><td>pEventCode</td><td>The event code</td></tr></table></td></tr>
</table>

JavaScript remote forms also report the evOpenContextMenu and evExecuteContextMenu events: see the *Context Menus* section for details.

Enabling form events

If you want to use any remote form events in your code, such as evFormToTop, you have to enable the events in the $events property of the remote form. In design mode, open your remote form, open the Property Manager for the form (right-click the form or press F6), click on the $events property, and check individual events in the popup list.

Form orientation and events

When the orientation of a remote form changes (e.g. when the end user rotates their mobile device), Omnis sends an evScreenOrientationChanged event to the top remote form. This allows the remote form to adjust the coordinates of any dynamically added objects. In addition, evFormToTop also receives the pScreenSize event parameter, allowing other forms to make adjustments if necessary when they come to the top.

Context Menus

You can implement context menus for remote forms or individual form controls by setting the $contextmenu property of the form or control to the name of a Remote menu class. (See the *Extending Omnis* manual for more information about creating Remote menus.) Each line in a remote menu has the $commandid property, so when the user selects a line in the menu this ID can be used to trigger a specific action in your code. When a line in a remote menu is selected, an evExecuteContextMenu event is reported to the form or field with event parameters containing the Command ID of the selected line, and for fields, a reference to the field which was clicked on.

Remote forms and controls have the $disabledefaultcontextmenu property. If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the default menu to be generated). The default menu could be the clipboard menu for the edit control.

Context menu events

Remote forms or controls report the following events in response to a context click.

<table>
<tr><th>Event</th><th>Description</th></tr>
<tr><td>evOpenContextMenu</td><td>Sent to a field or a remote form when a context menu is about to open
Parameters
<table>
<tr><td>pEventCode</td><td>The event code</td></tr>
<tr><td>pContextMenu</td><td>A reference to the remote menu instance that is about to pop up as a context menu</td></tr>
<tr><td>pClickedField</td><td>A reference to the field which was clicked</td></tr>
</table></td></tr>
<tr><td>evExecuteContextMenu</td><td>Sent to a field or remote form when a context menu item is selected
Parameters
<table>
<tr><td>pEventCode</td><td>The event code</td></tr>
<tr><td>pCommandID</td><td>The command ID of the selected remote menu item</td></tr>
<tr><td>pClickedField</td><td>A reference to the field which was clicked</td></tr>
</table></td></tr>
</table>

The remote menu instance itself only exists during the evOpenContextMenu event, therefore it is possible to modify the menu before it is displayed on the client, or you can discard the event to prevent the menu from being displayed.

Remote forms have the property $remotemenu which is the name of the remote menu instance (set only when evOpenContextMenu for the field or form is being processed): for

hierarchical menus, this is the item reference to the remote menu instance of the attached remote menu.

After evOpenContextMenu completes, and the user selects a remote menu item, the client sends an evExecuteContextMenu event to the form or form control that received the evOpenContextMenu, passing the event parameter pCommandID containing the value of $commandid of the selected menu line.

Form and Component Transparency

Remote forms have the $alpha property which sets the transparency of the form (an integer from 0 to 255, with 0 being completely transparent and 255 opaque). In addition, $backalpha lets you control whether or not subforms in the main form use the background color of the subform field or the form itself. The majority of the JavaScript components have the $alpha and $backalpha properties which control the transparency of the foreground and background colors of the component.

In combination with the animation methods, you can use the $alpha of a form or control to make elements in your form appear and disappear. The About windows in the sample apps are displayed by setting the $alpha property of the About subform and using the animation "ease in" effects. See the *Animations* section.

Client Commands

The $clientcommand() remote form method allows you to execute various commands on the client device, including ones that open various type of message boxes; the suitability of certain commands will depend on the current client device (e.g. some of the commands may not be available for certain mobile devices), so these commands should be thoroughly tested, as appropriate, for all devices you wish to support. The $clientcommand() method may be very useful since the JavaScript does not support client-side scripting, i.e. component methods cannot be executed on the client (as in the Omnis Web Client plug-in).

The $clientcommand() method must be executed on the Omnis Server in a remote form instance, and requires various parameters dependent on the command sent to the client. It has the following general syntax:

```
Do $cinst.$clientcommand(cCommand,wRow)
```

where $cinst is the current remote form instance, *cCommand* is the name of the command to be executed on the client, and the *wRow* is a row variable containing any number of parameters to be passed to the client.

The client commands that open a message box allow you to enter the message text in the first parameter. You can create a line break in the message text using //.

Yes/No messages

The “yesnomessage” command opens a Yes/No message box in which Yes is the default button.

```
Do $cinst.$clientcommand("yesnomessage",row-variable)
```

Where *row-variable* is row(message text, title text, name of public form server method called on yes, name of public form server method called on no, name of public form server method called on cancel (leave empty for no cancel button)).

For example, in the **Webshop** sample app a Yes/No message is generated using the $clientcommand method when a user clicks on a product size/type that is not available:

```
Do $cinst.$clientcommand('yesnomessage',row(con('Would you like
   to order >',iProductList.product_size_1,'< instead?'),'Not
   available','$orderYes'))
```

No/Yes messages

The “noyesmessage” command opens a No/Yes message box in which No is the default button.

```
Do $cinst.$clientcommand("noyesmessage",row-variable)
```

Where *row-variable* is row(message text, title text, name of public form server method called on yes, name of public form server method called on no, name of public form server method called on cancel (leave empty for no cancel button)).

Ok/Cancel messages

The “okcancelmessage” command opens an OK/Cancel message box in which OK is the default button.

```
Do $cinst.$clientcommand("okcancelmessage",row-variable)
```

Where *row-variable* is row(message text, title text, name of public form server method called on ok, name of public form server method called on cancel (leave empty for no cancel button)).

Generic message boxes

The “javamessage” command shows a message box on the client. The message box can be various styles (error, warning, success, prompt, message, query) and can have up to three buttons and accompanying text.

```
Do $cinst.$clientcommand("javamessage",row-variable)
```

Where *row-variable* is row(style(error, warning, success, prompt, message, query), text, title text, openatmouse(bool), butt1text:servermethodname, butt2text:servermethodname, butt3text:servermethodname).

Playing sounds

The system bell

The “soundbell” command plays the default system sound on the client.

```
Do $cinst.$clientcommand("soundbell",row-variable)
```

In this case, row() is empty.

Play a sound file

The “playsound” command plays a sound on the client.

```
Do $cinst.$clientcommand("playsound",row-variable)
```

Where *row-variable* is row(name of sound file from html sounds folder).

Date format

The “setcustomformat” command allows you to set the date format used on the client when $customformat is empty (defaults to D m y).

```
Do $cinst.$clientcommand("setcustomformat",row-variable)
```

Where *row-variable* is row(date format). See the *Date and Time Formatting* section for more details about setting the data format on the client.

Client preferences

The following commands allow you to save and load end-user data on the client, such as user preferences. You could use these commands to store and load usernames and/or passwords to allow the end user to log onto your application.

Saving preferences

The “savepreference” command saves a value (as a character string) as a named preference on the client.

```
Do $cinst.$clientcommand("savepreference",row-variable)
```

Where *row-variable* is row(preference name, preference value).

Loading preferences

The “loadpreference” commands loads a named preference value from the client preferences into an instance variable.

```
Do $cinst.$clientcommand("loadpreference",row-variable)
```

Where *row-variable* is row(preference name, instance variable name (e.g. a quoted string containing the name of the variable)).

Animations

JavaScript remote forms have the methods $beginanimations() and $commitanimations() which allow you to control animations for some controls.

- $beginanimations(iDuration[,iCurve=kJSAnimationCurveEaseInOut])
  after calling this, assignments to some control properties are animated for iDuration (milliseconds) by executing $commitanimations()
- iCurve values are:
  kJSAnimationCurveEaseInOut (the default), kJSAnimationCurveEaseIn, kJSAnimationCurveEaseOut, kJSAnimationCurveEase and kJSAnimationCurveLinear

The animated properties are: left, top, width, height, alpha, backcolor, backalpha, textcolor, fontsize, bordercolor, linestyle, buttoncolor (the latter is for pushbutton only).

If you set the same property for an object more than once, the first property change is animated, and then the last property change is animated when the first completes. Property

changes between the first and last are ignored. The evAnimationsComplete event (for remote forms) is generated after the last property change(s) have completed. This allows you to reverse the effect of an animation (which is the equivalent to the autoreverse/repeat options available on iOS).

JavaScript animations do not work in Internet Explorer – they downgrade to immediately assigning the property value(s). As a result, evAnimationsComplete will never get sent in IE.

The sample apps in the Welcome window contain an **About** form which is loaded into a subform on the main sample app form. Animations are used to display the About subform, which initially has the $alpha value of zero (fully transparent) and is increased to 255 during the animation, as follows:

```
; method behind About button
On evClick
   Switch iScreensize
      ; set aboutSubForm size for different devices
      Etc.
   End Switch
   Do $cinst.$objs.aboutSubForm.$classname.$assign("jsAbout")
   Do $cinst.$objs.aboutSubForm.$visible.$assign(kTrue)
   Do $cinst.$beginanimations(500,kJSAnimationCurveEaseIn)
   Do $cinst.$objs.aboutSubForm.$alpha.$assign(255)
   Do $cinst.$commitanimations()
```

Gradients

To create a gradient for the background of a JavaScript remote form, you can select a gradient fill pattern for $backpattern and control the start and end colors for the gradient by setting $forecolor and $backcolor.

String Localization

The Omnis Server looks up the locale in the data from omnisloc.df1, and if it finds the locale, it will use the strings for that localization data file entry (which can default to coming from ICU). The locale match is initially based on the full locale, and if that fails, the language code only (first 2 characters of locale). Note the locale data is loaded at startup, so if you change omnisloc.df1, you need to restart.

The Omnis Server also sends the strings for Yes/No and OK/Cancel messages to the JavaScript Client: these must come from the localization data file, since ICU does not provide string localization for these strings; if there is no matching locale in omnisloc.df1, the strings default to English.

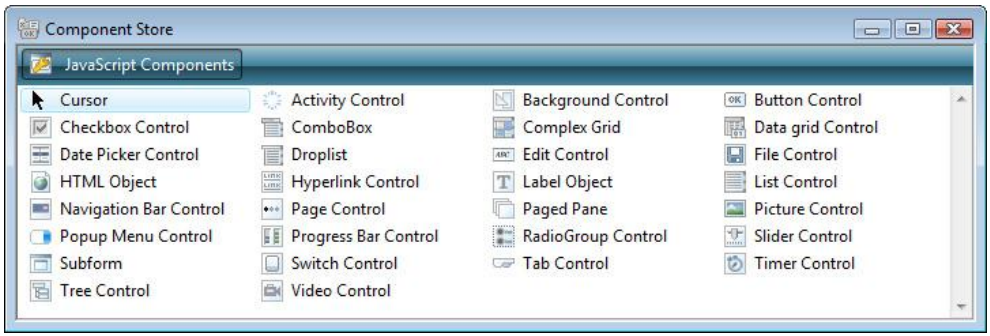
In addition, there is an option, ‘Use locale for defaulted items’, which when turned off, means the defaulted items come from the default system locale. This functionality only applies this option to the current language - other languages will always use the locale for defaulted items.

Time Zones

The JavaScript Client exchanges dates and times between server and client using UTC time (which is essentially the same as GMT but UTC is used globally as the standard time for web servers). You should therefore store all dates and times in UTC and use the time zone offset of the client to either determine or set the local time of the client. The $construct row variable parameter for the remote task/form has a column JStimezoneOffset, which is the timezone offset in minutes for the client relative to UTC time. For example, if the client's local time zone is UTC+2 (or GMT+2), JStimezoneOffset will be 120. See the 'Contruct row variable' section for more information about the $construct row variable for tasks/forms.

JavaScript Components

The Component Store contains a new set of components for the JavaScript Client which you can drag and drop onto a JavaScript remote form. Assuming you have set the $client remote form property to kClientJavaScript, the Component Store will display the new components under the "JavaScript Components" tab. In terms of functionality, many of the new JavaScript components are very similar to the equivalent components in previous versions of Omnis Studio, while there are a number of new components, such as the Date Picker and Navigation Bar, which are well suited to applications for mobile devices with touch capabilities.



The following components are available for JavaScript remote forms:

Component	Description
Activity Control	Animated image to display during Omnis Server activity
Background Control	Object you can set to Rect, Line, Triangle, or Image
Button Control	Standard pushbutton which reacts to clicks
Checkbox Control	Check box for on/off values
ComboBox	Field combining entry box and droplist
Complex Grid	Grid which can display all types of data and formatting
Data grid Control	Simple grid for text and numerical data display
Date Picker Control	Data picker with touch selection
Droplist	List that drops down when clicked
Edit Control	Standard edit field for data entry or display
File Control	Allows end users to upload or download files
HTML Object	Object to display HTML content
Hyperlink Control	List containing hyperlink style options
Label Object	Basic label object
List Control	Standard list field for displaying list variable data

Component	Description
Navigation Bar Control	Navigation with touch selection
Page Control	Allows selection of page pane using touch
Paged Pane	Can contain fields on multiple panes
Picture Control	Standard field for displaying images
Popup Menu Control	A menu that pops up when clicked
Progress Bar Control	Shows progress of server process or calculation
RadioGroup Control	Displays a group of radio buttons for exclusive selection
Slider Control	Slider component for setting values
Subform	Allows you to insert another remote form as a subform
Switch Control	Allows on/off selection; you specify an icon for on/off state
Tab Control	Multiple tabs to control selection of page pane
Timer Control	Timer object triggers an event at a specified interval
Tree Control	List for displaying hierarchical data or list of options
Video Control	Plays a YouTube or Flowplayer video

JavaScript Component Gallery

There is a gallery web page on the Omnis website showcasing many of the new JavaScript Components at: www.tigerlogic.com/omnis

JavaScript Component Properties

The new JavaScript components have properties, like all other Omnis components, which you can view and set in the Property Manager (F6). You can write methods to change the properties of the new JavaScript components.

See the Reference section in this manual for a description of all component properties and methods.

Setting properties using the Field List

When designing a remote form, you often need to click on the background of the form to set its properties in the Property Manager. This may be difficult if your form is completely filled with components and no form background is available to click on, as is often the case for mobile forms. To select the form in this case, you can use the Field List (right-click anywhere on the form, open the Field List and check the form name to open the Property Manager for the form), or if you click on any individual component, then shift-click it to deselect it, the focus will be returned to the form and its properties will be shown in the Property Manager.

$dataname for JavaScript controls

As with the Web Client and iOS Client, the variable specified in the $dataname property of a JavaScript component must be an instance variable, or in some cases a column in an instance row variable.

Naming JavaScript controls

When you create a component in your remote form, a name is generated automatically and assigned to the $name property of the component. This is usually in the format <remoteformname>_<component-type>_<number>, such as ‘rftest_edit_1001’ for an edit control on a remote form called rftest. However, you can enter your own name for a component which may better describe the object within the context of your form; for example, an edit field to allow the end user to enter their first name could be named Firstname.

The name you assign to an object does not have to conform to any convention other than any conventions you may like to use in your forms or the application to identify different objects. The name of a component (specifically the value in $name) can be used in the Omnis notation and throughout your library to refer to the object. You should try to use alphanumeric characters only for object names to avoid any possible conflicts in your code. For example, an object name should not include the dollar symbol ($) since this would cause a conflict when you reference the object using the Omnis notation.

Control size and float properties

All the JavaScript components have the $screensizefloat property which controls their automatic resizing and floating properties when contained inside the wrapper which is provided to allow you to create standalone custom apps for iOS and Android. See the *Creating Standalone Mobile Apps* section for more details.

Date and Time Formatting

You can set the formatting for Date and Time type data for some of the JavaScript components including Edit controls, Combo boxes, Data grids, Droplists, Hyperlink lists and standard Lists. These components have the properties:

- **$jscustomformat**
  a date-time format string using the characters described below. If $jsdisplayformat is kFormatCustom, and the data is of type 'Date Time', this property is used to format the data. If empty, it defaults to the format set using $clientcommand 'setcustomformat'
- **$jsdisplayformat**
  the format used to display 'Date Time' data, a kJSFormat... constant as follows:

kJSFormatNone	No format
kJSFormatTime	Default time format for client locale
kJSFormatShortDate	Default short date format for client locale
kJSFormatShortDateTime	Default short date and time format for client locale
kJSFormatMediumDate	Default medium date format for client locale
kJSFormatMediumDateTime	Default medium date and time format for client locale
kJSFormatLongDate	Default long date format for client locale
kJSFormatLongDateTime	Default long date and time format for client locale
kJSFormatFullDate	Default full date format for client locale
kJSFormatFullDateTime	Default fulll date and time format for client locale
kJSFormatCustom	Use the custom format in $jscustomformat

Date formatting characters

The date formatting characters for $jscustomformat are D, V, w, E, n, M, m, y, Y, A, H, h. Some additional characters are supported for Date/Time formatting for the JavaScript Client components only, as follows:

j day with no leading zero (6)

P month with no leading zero (6)

K hour with no leading zero (0..23)

k hour with leading zero (1..12)

a am/pm

O timezone offset (+01:00)

There is a new entry on the Constants tab in the Catalog (F9) called "Date codes (JavaScript Client only)" that lists the formatting characters.

Date formatting and Locale

When the client connects, the server sends it the date formats, day names and month names for the client locale (the server reads these from ICU). If you assign $ctask.$stringtablelocale in $construct of your remote task, the server sends the client the formats and so on for the assigned $stringtablelocale locale.

Autoscrolling

You can enable automatic scrolling for Edit controls, Lists, Tree lists, Hyperlink controls, Pictures and Html controls by enabling the $autoscroll property. If this property is kTrue for the control, and the client is not a mobile device, the client automatically displays scrollbar(s) when not all of the content in a field is visible.

Setting $autoscroll to kTrue changes $horzscroll and $vertscroll to kFalse, and makes $horzscroll and $vertscroll unassignable. By default, $autoscroll is enabled for Edit

controls, Lists and Tree lists, while for Hyperlink controls, Pictures and HTML controls $autoscroll is set to kFalse.

Rounded Borders

All the JavaScript components can have rounded borders by specifying the corner radius in pixels in the $borderradius property. To set all the corners of the object to the same radius you can enter a single value, or to specify the radius for different corners you can use the syntax "n-n-n-n" which follows the same rules as CSS 3 rounded border syntax. The order for the radius parameters is topleft, topright, bottomright, bottomleft. If bottomleft is omitted the topright value is used, if bottomright is omitted the topleft value is used, if topright is omitted the topleft value is used.

Component Events

Most of the JavaScript components report events, including evClick and evDoubleClick, which you can handle in an event method ($event) inserted behind the component. For example, when the end user clicks a button, an evClick is generated which you can trap in the $event method for the button; this method could display a message, initiate another method or determine some other action depending on the code in the event method.

See the Reference section in this manual for a description of the events for each component.

Enabling Events

Many of the components have their events enabled by default, but for some you may need to enable specific events in the $events property for the component.

To enable an event

- Select the component and open the Property Manager (press F6)
- Click on the $events property in the Property Manager to drop down the list of events for that component (the property will show “No Events” when no events are selected)
- Check the events you wish to enable for this component

If you double-click a component in design mode, the Method Editor will open displaying the method for that individual component. For components with events, the $event method will be shown. For example, if you double-click on a button, the Method Editor will open displaying the $event method containing the code *On evClick;* you can add more code after this line to be run when the end user clicks the button.

Component Icons

Some of the JavaScript components can display icons. The icon is specified in the $iconid property of the component. For example, buttons can display an icon alongside a single line of text, and in this case the icon is specified in the $iconid property of the button; the icon is selected from one of the Omnis icon datafiles or the #ICONS system table in your library (the latter is the preferred method since all the icons are contained in your library).

Web Client and iOS remote forms use the $iconpages property to specify which icons/icon pages are sent to the client. However, JavaScript remote forms do not have this property and therefore component icons are handled in a different way.

When you open/test a JavaScript remote form in development mode, all the icons needed for the components in the form (such as buttons) are copied automatically to the html_icons folder in your development tree, under the following sub-folders:

- html/_icons/lib/<LIBNAME>/ if the icon was created inside the library #ICONS, or
- html/_icons/datafile/omnispic/ if the icon comes from Omnispic

The icons in the _icons folder need to be added to your web server in the same relative location when you deploy your web/mobile application. There is a new JS Icon Export tool in the Add-ons>>Web Client Tools menu option, which you can use to export the icons for deployment in a web server tree. Alternatively, you can copy all the icons from the _icons folder in your development tree that were generated automatically when testing forms, if you are sure that they are all required.

Activity Control

The Activity Control provides an animated image to show some activity on the client, for example, during a long list calculation or search operation on the Omnis Server. It has the following custom properties:

Property	Description
$activitystyle	The style of the indicator, one of the following constants: kJSActivityBar: a moving bar kJSActivityBlock: a moving block kJSActivitySmallSpinner: a small rotating spinner kJSActivityCustomLink: the image in $customlink is used
$customlink	The path of an animated GIF which can be displayed when $activitystyle is set to kJSActivityCustomLink

The following code assigns a custom link to the activity control:

```
Calculate iCustomLink as
   'http://www.mywebsite.com/images/animated1.gif'
Do $cinst.$objs.Activity.$customlink.$assign(iCustomLink)
```

Background Control

The Background Control allows you to draw various shapes in your remote forms. It can be assigned one of a number of shapes including: Ellipse/Circle, Rectangle/Square, Rounded rectangle/Square, Triangle, Horizontal Line, Vertical Line, and Image (the image source is specified in a URL in $imagepath). You assign the shape to the object by setting the $::shape property to one of the kJSBack... constants.

You can assign a solid color or gradient fill to a background component by setting its $backpattern, $forecolor and $backcolor. You can also assign the stroke (border) thickness and color by setting $strokewidth and $bordercolor.

Animations and Changing Attributes

The Background Control has the $animation and $attr properties which allow various animations or effects to be assigned to the object, such as fading the object in or out, or for $attr various attributes of the object to be changed. The $animation and $attr properties must be assigned at runtime and accept a string containing various parameters depending on the function or attribute.

Apart from the scale attribute, the browser must support the Raphael JavaScript library to allow animations and attribute changes (more details about the parameters you can use are available from http://raphaeljs.com).

The $animation property follows the general format:

```
function( newvalue, time(milliseconds), ease(optional),
   complete_context(optional) )
```

The functions available in $animation are:

- **scale**
  increases and decreases the size of the object

```
scale( newvalue, time(milliseconds), ease(optional),
   complete_context(optional)  )
```

- **alpha**
  changes the transparency of the object

```
alpha( newvalue, time(milliseconds), ease(optional) ,
   complete_context(optional)  )
```

- **rotate**
  rotates the object

```
rotate( newvalue, time(milliseconds), ease(optional) ,
   complete_context(optional)  )
```

For example:

```
Calculate $cinst.$objs.backgroundobject.$animation as
   "alpha(0,500,<>,fade_complete)"
```

fades the object to alpha value 0 over 500 milliseconds using a ‘slow, faster, slow’ easing method (see below) and when complete calls evAnimComplete with a parameter "complete_context".

You can use the complete event to chain the next animation, therefore to pulse an object you could use:

```
Calculate $cinst.$objs.backgroundobject.$animation as
   "alpha(0,500,<>,fade_off)"
```

You could use the event handling method:

```
On evAnimComplete
   If ( pAnimContext="fade_off" )
      Calculate $cwind.$objs.backgroundobject.$animation as
         "alpha(0,500,<>,fade_on)"
   Else if ( pAnimContext="fade_on" )
      Calculate $cwind.$objs.backgroundobject.$animation as
         "alpha(0,500,<>,fade_off)"
   End if
```

Ease transition effects

The following ease transition effects or “eases” are supported for animations:

=	linear and default if not specified
>	fast then slowing
<	slow then faster
<>	slow, faster, slow
bounce	object bounces
elastic	object stretches
backIn	object backs in
backOut	object backs out

Changing Background Attributes

The $attr property allows you to change various attributes of the object, such as their transparency. For example, you can assign an alpha gradient to an object using the following method:

```
; note must be assigned at runtime
Do $cinst.$objs.backgroundobject.$attr.$assign('attr(gradient, 0-
   #FFFFFF:10-#FFFFFF)')
Do $cinst.$objs.backgroundobject.$attr.$assign('attr(opacity,0.0)')
```

Button Control

The Button Control is a basic pushbutton that the end user can click to confirm or initiate a process, such as an OK or Cancel button. The button can display an icon, specified in the $iconid property, and/or a single line of text specified in $text. The Button has the following custom properties:

Property	Description
$buttonbackiconid	The icon id of background image for the button. To use the default system button, set $buttonbackiconid to zero and $buttoncolor to kColorDefault
$buttoncolor	The color of the button. To use the default system button, set $buttonbackiconid to zero and $buttoncolor to kColorDefault
$textbeforeicon	If true, and the control has both text and an icon, and the text is displayed to the left of the icon
$::vertical	If true, the text and icon are arranged vertically

When a Button is clicked an evClick event is triggered which you can handle in the event handling method behind the button.

```
On evClick
   Do something…
```

For example, all the sample apps in the Welcome window have an About window which is loaded into a subform and displayed using an animation; see the Animations section for the About button code. The **Close** button on the About windows simply closes the About form by sending a message to the main remote form to run a method; it has the following code:

```
On evClick
   Do $cwind.$closeAbout()
```

In this case, $cwind is a reference to the main parent form which contains a method called $closeAbout which contains code to fade out the About form and reset various buttons on the main form.

Checkbox Control

The Checkbox control can represent On / Off or Yes / No values and is typically used to allow the end user to turn on or off an option. The variable you specify in the $dataname property of a Checkbox should be a Number or Boolean variable. The $text property specifies the text for the Checkbox.

When a Checkbox is clicked an evClick event is triggered with the current value reported in the pNewVal parameter.

For example, you could use a series of check boxes and a radio button group to filter a list of people based on their gender and age group. The group of controls on the remote form could look like this:

The $dataname for each of the check boxes is iAgeRange1, iAgeRange2, and iAgeRange3 respectively. These are all Boolean variables defined in the remote form, and the $dataname of the Radio button group is iFilter, which is defined as a Short integer. Each separate check box and the Radio group has a simple event method, which is:

```
On evClick
   Do method filter
```

which will call the 'filter' class method when any of these objects is clicked. The filter method filters the contents of the list called iList based on the selection of the check boxes and radio buttons, and has the following code:

```
Do iList.$unfilter(0)
If not(iAgeRange1)
   Do iList.$filter(not(iAge<=20))
End If
If not(iAgeRange2)
   Do iList.$filter(not(iAge>=21&iAge<=40))
End If
If not(iAgeRange3)
   Do iList.$filter(not(iAge>40))
End If
Switch iFilter
   Case 1
      Do iList.$filter(iGender='F')
   Case 2
      Do iList.$filter(iGender='M')
End Switch
```

Note the 'Smart list' capability has to be enabled on the iList variable to allow the built-in filtering using the $filter method. You can enable this using the code:

```
Do iList.$smartlist.$assign(kTrue)
```

ComboBox

The Combobox control is a combination of a data field and a dropdown list from which the end user can make a selection or enter their own value into the field. The variable for the data field part is specified in the $dataname property. You can specify a default list of options in the $defaulttext property, which is a comma-separated list of options, or build the list dynamically (with $::listname, see below). When $defaulttext is specified, $defaultline specifies the list line which is selected when the form is opened (set to 1 by default). The $::listheight specifies the height of the droplist.

Alternatively, you can assign the name of a list variable to the $::listname property to assign the contents of the list to the droplist part of the combo box; $listcolumn specifies which column of the list variable is used to populate the droplist.

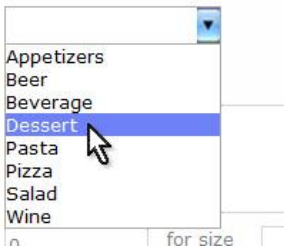
When the list in a ComboBox is clicked an evClick is generated with the selected list line reported in the pLineNumber parameter.

Content Tips

The Combo box control has the $::contenttip property which is a text string which is displayed in the edit field part of the combo box when it is empty to help the user understand what content should be entered into the field. For example, for a Last name field you could enter 'Enter your last name' into $::contenttip to prompt the end user for their last name.

Example

The maintenance screen in the **Webshop** sample app allows the user to enter new products or delete existing ones: specifically, the data in the Webshop app contains food and drink items, but it could be any type of products. When the user enters a new product, they can select the product type from a Combo control; this allows the user to select from a list of given product types or enter a new one.



The $dataname of the combo control is set to iDataRow.product_group, and the $::listname is iGroupList. The evAfter event is enabled in the $events property of the control. In the $construct method of the form, the iDataRow row variable is defined from the T_Products table class, as follows:

```
; $construct of jsMaintenance form
; sets up form sizes, etc, then…
Do iDataRow.$definefromsqlclass($tables.T_Products)
Do $cinst.$objs.$sendall($ref.$construct())
```

The last line of code triggers all the field specific $construct methods which in this case includes the Combo box control; the code defines the iGroupList from the product_group column in the T_Products table class, performs a select on the data, and fetches all the data back into the iGroupList variable.

```
Do iGroupList.$definefromsqlclass(
        $tables.T_Products,'product_group')
Do iGroupList.$selectdistinct()
Do iGroupList.$fetch(kFetchAll)
```

When the user selects an item in the list or enters a new item into the entry part of the combo box, an evAfter event is triggered and the $event method behind the combo control is called, as follows:

```
On evAfter
   Do method newItem
   Do $cinst.$setcurfield('product_name')  ;; puts the focus in the
   product_name field
```

The newItem method is placed behind the Combo box control itself and contains the following code:

```
Do iGroupList.$search(
     $ref.product_group=iDataRow.product_group,kTrue,kFalse,
     kFalse,kFalse)  ;; test iGroupList for group/type entered
If iGroupList.$line=0      ;; if not found in the group list
   Do iGroupList.$add().product_group.$assign(
     iDataRow.product_group)  ;; add a new group to the list
End If
```

Complex Grid

A Complex Grid can display multiple rows and columns of data taken from a list variable specified in the $dataname property of the control. You can use a $construct method behind the grid control to build the list. To create a complex grid you place other controls in the row and header sections of the grid control, including standard entry fields, droplists, buttons, and checkboxes. The $dataname of each field in the grid can correspond to columns in your list variable supplying the data to the grid. You can place event methods behind the embedded controls (rather than the grid field itself) to react to user input and clicks. For example, you can have a button in each row of the grid which when clicked triggers a method that performs an action based on the row clicked.

The JavaScript Complex Grid is much like its Standard Field counterpart: see Chapter 6 in the *Omnis Programming* manual for information about Complex grids, and see the Reference section in this manual for a complete list of properties for the JavaScript Complex Grid.

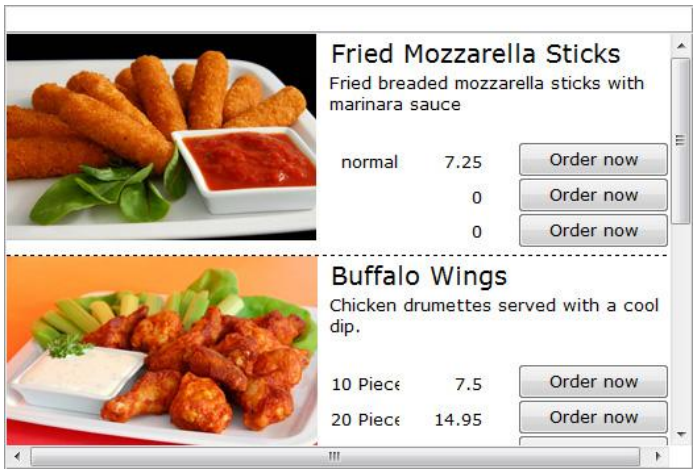
The **Webshop** sample app, available in the Welcome window (open via the New Users button), uses a complex grid in the main product remote form to display a list of products. Individual fields for the picture, name, description, price/size of the product are added to the first line of the complex grid; when the form is opened on the client and the data is

loaded into the grid, these fields and are repeated *for each row* in the data list (one row per product).

The $dataname of the complex grid is set to iProductList which is built from a table class T_Products which is linked to a schema class sProducts. A $construct method is placed behind the complex grid that builds the list needed for the complex grid data.

```
; $construct of complex grid control in jsShop form
Do iProductList.$definefromsqlclass($tables.T_Products)
Calculate whereClause as con('WHERE product_group =
   ',kSq,'Appetizers',kSq)
Do $cfield.$build(whereClause)      ;; calls $build
; $build method also behind complex grid control in jsShop form
Do iProductList.$select(pWhereClause,' ORDER BY product_isfood
   desc')
Do iProductList.$fetch(kFetchAll)
```

When the form is opened, the $construct method is run and the product list is built from database, while the data itself is displayed in the various fields embedded in the complex grid with each product shown on a separate line in the complex grid.



There are 3 order buttons placed in the row of the complex grid; they are repeated for each product in the list and allow the end user to order different sizes of product, such as a small, medium, or large drink or pizza. Each of the buttons has a simple method behind it that

passes a number to the process_order class method; the first button sends value 1, the second button value 2, and the third button value 3.

```
; 'Order now' button method
On evClick
   Do method process_order (1)
```

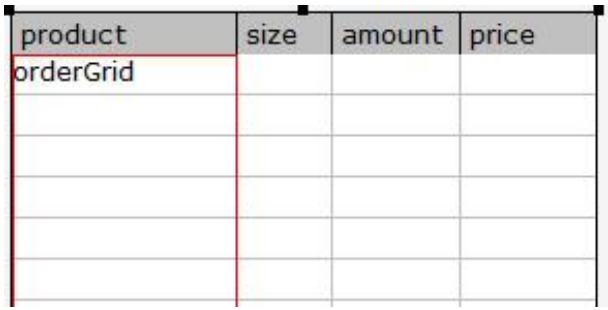
See the *Data Grid* section for the process_order method which updates the iOrderList and the Orders data grid accordingly.

Data Grid Control

Data grids can display numeric and character data in a grid like structure, much like a table or spreadsheet format. The data contents for the data grid is supplied from the list variable specified in the $dataname property. The $currentcolumn property is the current grid column for which properties are being displayed in the Property Manager. See the Reference section in this manual for a list of all properties for the Data Grid control.

The height of the rows in a data grid adjusts to fit the data, and you can control the contents of the grid headers using the column header properties. The end user can enter data into the cells of the grid if $enterable is enabled, and the grid can grow to accommodate more data if $extendable is enabled.

The **Webshop** sample app, available in the Welcome window (open via the New Users button), uses a data grid to display a list of products that have been ordered in the main product form.



The $dataname of the data grid is set to iOrderList which is defined from a table class T_qOrders which is linked to a query class qOrders. When the product form is opened, the $construct method behind the data grid defines the list from the table class.

```
; $construct behind the data grid
Do iOrderList.$definefromsqlclass($tables.T_qOrders)
```

When the end user clicks the 'Order Now' button in the product window, the data for the selected product and size/type is passed to the process_order method (as value 1, 2, or 3), which inserts the data into the list (after a check to see if the user has already ordered the same product) and the list is redrawn. The process_order method is as follows:

```
; process_order class method in the jsShop form
; contains pButtonNumber parameter (Short Int) to receive the value
   of the product button clicked
If iProductList.product_price_[pButtonNumber]>0     ;; price must be
   greater zero
   Do iOrderList.$search(
      $ref.order_product_id=iProductList.product_id
      &$ref.order_size=iProductList.product_size_[pButtonNumber],
      kTrue,kFalse,kFalse,kFalse)
   If iOrderList.$line   ;; found one so increment existing order
      Calculate iOrderList.order_amount as iOrderList.order_amount+1
   Else     ;; new one so add to iOrderList
      Do iOrderList.$add(
         #NULL,#D,iProductList.product_id,iProductList.product_name,
         iProductList.product_size_[pButtonNumber],
         1,0,iProductList.product_price_[pButtonNumber])
      Do iOrderList.$line.$assign(iOrderList.$linecount())
   End If
   Calculate iOrderList.total_price as
      iProductList.product_price_[pButtonNumber]
      *iOrderList.order_amount
   Do $cinst.$objs.checkOutBtn.$enabled.$assign(
      iOrderList.$linecount()>0)
   Do $cinst.$objs.orderGrid.$redraw()
Else
   Do $cinst.$clientcommand('yesnomessage',row(con('Would you like
   to order >',iProductList.product_size_1,'< instead?'),'Not
   available','$orderYes'))
End If
```

Date Picker Control

The Date Picker control allows the end user to select a date and/or time by clicking on the up or down arrows for each part of the date or time. You can assign a Date/Time instance variable to the $dataname property to load the date/time selected by the user, or you can assign a two column instance row variable to contain the date/time and time zone offset of the client in the respective columns (see below of info on the time zone offset).

The $datestyle property specifies the style or date/time content of the date picker control, which can be a combination of date & time, date only, time only, or a calendar view, as specified by a constant:

kJSDatePickerStyleDate	a date display only
kJSDatePickerStyleDateTime	a date and time is displayed
kJSDatePickerStyleTime	a time display only
kJSDatePickerStyleCalendar	a calendar is displayed

The color of the Date Picker is specified with $datefacecolor while $datefacealpha sets the transparency (value 0-255).

When the date is changed on the client an evDateChange event is generated, or when in calendar view an evDateDClick is sent when a date cell is double-clicked.

Calendar style picker

The Date Picker control can be switched to display a calendar style date picker, by setting $datestyle to kJSDatePickerStyleCalendar. There are a number of properties that apply to the control when the date style is set to calendar: see the Reference section for a list of properties for the Calendar style picker.

Picker style on mobile devices

When $datestyle is set to kJSDatePickerStyleCalendar desktop browsers will display the calendar as expected. However on mobile devices, even when $datestyle is set to kJSDatePickerStyleCalendar, a calendar will be replaced with a date picker (same as kJSDatePickerStyleDate), since a picker style date selector is the preferred style on mobile devices. This can be overridden by setting the property $datestyleusepickeronmobile to kFalse.

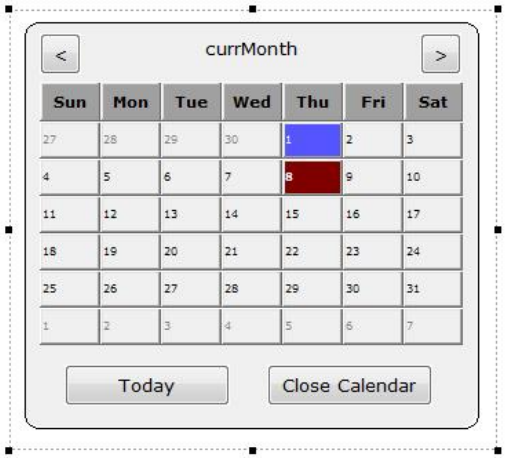
In the **Holidays** sample app uses the Date picker control set to kJSDatePickerStyleCalendar to allow users to select the dates for the holiday applications. The jsUserForm in the Holidays app has two buttons to allow the end user to select a “From” date or “To” date to specify the Begin and End dates for their holiday request. The “From” button has the following code:

```
On evClick
   Calculate iUsingCalendar as kTrue
   Calculate iSelectFrom as kTrue  ;; this is the From button
   Do method openCalendar
```

The openCalendar class method moves the calendarPane into view on the main form and has the following code:

```
Do method enableFields (kFalse) ;; enables the calendar pane &
   disables the name pane
Do $cinst.$objs.calendarPane.$top.$assign(90)
Do $cinst.$objs.calendarPane.$left.$assign(150)
Do $cinst.$objs.calendarPane.$visible.$assign(kTrue)
```

If you examine the Holidays app to look at the Date picker note that it is on a page pane field called calendarPane located on the jsUserForm. The $left property for the calendarPane is set to 990, to hide it from view, therefore you'll need to select it using the Field List (right-click the remote form, select Field List and check the calendarPane) and set its $left property to 200 in the Property Manager in order to see it.



The $dataname of the Date picker control itself is set to iCalendarDate, an instance variable of type Date Time and subtype 'D m y'; when the end user selects a date this variable is set to the selected date automatically. The Date picker has a simple event method to detect when the end user double-clicks on a date cell; it has the following code:

```
On evDateDClick
   Do method closeCalendar
```

The closeCalendar class method passes the date from iCalendarDate into either the iFromDate or iToDate variable defined in the form; it has the following code:

```
If iSelectFrom  ;; if the From button
   Calculate iFromDate as iCalendarDate
Else    ;; it must be the To button
   Calculate iToDate as iCalendarDate
End If
Do $cinst.$objs.calendarPane.$left.$assign(1250)
Do method enableFields (kTrue)
```

The final two lines of the method move the calendarPane off to the right and enables the fields on the Name pane.

Time zone offset

You can return the time zone offset of a date value when it is passed back from the client by using a two column row variable in $dataname; the first column should be defined as a Date Time variable and the second of type Number. If the server passes a new value to the client then only the first column is significant and should specify the new date to be sent to the client.

When evDateChange signals that there has been a change on the client then the updated date is passed back to the server in the first column of the row variable as a UTC/GMT date value and the time zone offset of that value in the client's current time zone is passed back in the second column. The time zone offset is the number of minutes from UTC/GMT, e.g. GMT+2 the time zone offset is 120. This can be used to calculate the date in the time zone of the client.

If time zone offset information is not required, $dataname can be specified as a Date Time instance variable only.

Droplist Control

The Droplist control displays a dropdown list from which the end user can make a selection; the contents of the list can be supplied from a default list or a list variable which can be built dynamically.

You can specify a default list of options in the $defaulttext property, which is a comma-separated list of options; $defaultline is the default line (set to 1) which is selected when the form is opened (only when $defaulttext is used). Alternatively, you can assign the name of a list instance variable to $dataname to populate the list; $listcolumn specifies which column of the list variable is used to display the list. The $::listheight specifies the height of the droplist.

When a line in the droplist is selected an evClick is generated with the selected list line reported in pLineNumber.

The jsUserForm in the **Holidays** sample app uses a droplist to allow users to select an employee to view their holiday leave requests. The $dataname of the empList Droplist control is set to iEmployeeList which is built via the $construct method when the form is opened.

```
; buildEmpList class method in the jsUserForm
Do iEmployeeList.$definefromsqlclass('sEmployee') ;; the schema
Do iEmployeeList.$sessionobject.$assign(iSQLObjRef)
Do iEmployeeList.$select()
Do iEmployeeList.$fetch(kFetchAll)

Do iEmployeeList.$cols.$add(iEmpFullName)
Calculate lTotal as iEmployeeList.$linecount
For lNum from 1 to iEmployeeList.$linecount step 1
   Calculate iEmpFullName as con(
      iEmployeeList.[lNum].FirstName,kSp,
      iEmployeeList.[lNum].LastName)
   Do iEmployeeList.[lNum].$assigncols(,,,,,iEmpFullName)
End For
Do iEmployeeList.$line.$assign(1)
Calculate iName as iEmployeeList.iEmpFullName
```

The droplist control contains a $event method which is triggered when the user selects a line in the list; the code in the event method redraws the holiday list for the selected employee:

```
On evClick
   Do method buildHolidayList
   Calculate iName as iEmployeeList.[pLineNumber].iEmpFullName
   Do $cinst.$objs.pagePane.$objs.holidayList.$redraw()
   Do $cinst.$objs.pagePane.$objs.empName.$redraw()
```

The buildHolidayList class method builds the list of holiday requests for the selected employee and redraws the form.

Edit Control

The JavaScript Edit control is a standard single line edit field which you can use to display data or allow the end user to enter data into. The Edit field can handle all types of character data stored in the instance variable specified in $dataname. When kTrue (the default), $issingleline makes the edit control only allow data entry on a single line, otherwise when kFalse the field can be extended downward to allow data entry on multiple lines. When kTrue, the $ispassword property ensures a place-holder character is displayed when the end user enters something in the field (only applies when $issingleline is kTrue).

The Edit Control reports the evBefore and evAfter events, so you can detect when the focus is about to enter or leave a field in the $event method.

Content Tips

The Edit control has the $::contenttip property which is a text string which is displayed in the edit field when it is empty and before the end user has entered any text. Using content tips may help the user understand what content should be entered into fields in the forms in

your application. For example, for a Last name field you could enter ‘Enter your last name’ into $::contenttip to prompt the end user for their last name. As soon as the end user starts to type something into the field the content tip will disappear.

File Control

The File Control allows end users to upload or download files. The control itself is invisible and to enable the upload or download functions, you need to assign an action, at runtime, to the $action property. There are two actions, the constants kJSFileActionUpload and kJSFileActionDownload, corresponding to the upload or download functionality. See the *Reference* section for a full list of properties for the File control.

When in upload mode the File control opens a standard Upload dialog to allow the end user to select a local file to upload. There are various properties to allow you to change the text and error messages on the Upload dialog.

For downloads, the File control can provide a standard hyperlink pointing to the file to be downloaded, or you can assign a row variable to the $dataname of the control containing the binary data of the file to download. The download function is supported for mobile devices, and if the browser can interpret the contents of the file it is shown in a new browser window or tab.

Using Timer and File controls

Note that if a Timer control is present on the page, timer events will not occur while a file upload dialog is open, or while file download is in progress.

Mobile limitations or issues

Support for the Upload or Download function on mobile devices depends very much on the device itself and the mobile operating system. Therefore if you intend to include an upload or download function in your app, you should test your app thoroughly on all devices you wish to support. With this in mind, we are aware of the following limitations or issues regarding different mobile operating systems.

Upload does not work on iOS because the input element to select a file does not become enabled since iOS does not support it. Download works via the hyperlink because mobile Safari has been implemented to support hyperlinks. However, the other download mechanism does not work: on iOS for example, the download actually transfers the data to the client, but then the browser does not carry out any action with the data, so the downloaded file is lost.

Uploading files

To enable the upload function you must define the File component’s $dataname as a two-column row variable: Column1 should be of type List, and column2 should be of type Character. The value assigned to column2 should be the *name* of a binary task variable which will receive the binary data when the upload is complete. Column1 will receive MIME header list information when the upload is complete.

To trigger the upload dialog, you.need to assign kJSFileActionUpload to the $action property. This action opens a file upload dialog which has two formats: one for up to date

browsers, and the other for browsers which do not support XMLHttpRequest2, i.e. Internet Explorer and Opera. When a file has uploaded an evFileUploaded event is generated. When the dialog closes the control generates evFileUploadDialogClosed.

The $maxfileuploadsize specifies the maximum size in bytes of a file to be uploaded to the server. Zero means no limit. Some clients (IE and Opera) cannot enforce this limit. See example below for details on how to upload a file.

Downloading files

In its simplest form, the File control can provide a hyperlink to allow the user to download a file located on the internet. In this case, you would set the $visible property of the control set to kTrue, set $hyperlinkurl to the URL of the file, and $hyperlinktext to the text for the hyperlink. The URL does not have to be a file download URL, it can be any file on the internet and will open in a new browser window or tab. The File control will display the download as a standard hyperlink on the remote form.

You can also use the File control to download a file from a binary file or one stored in your database. To enable this functionality, you need to assign kJSFileActionDownload to the $action property. In this case, $dataname of the control must be a row variable, with Column1 as the file name, Column2 the media type (e.g. text/plain;charset=utf8), and Column3 the name of a remote task variable which should be a binary containing the file data or character containing a file path.

Example

A File control is used in the jsContacts form in the **Contacts** sample app to allow end users to upload a photo of their contacts. The File control is placed somwehrere on the form, its $dataname is set to iFileRow, an instance row variable, and the evFileUploaded and evFileUploadDialogClosed events are enabled in the $events property of the control. The iFileRow variable is setup in the $construct method of the form, as follows:

```
Do iFileRow.$define(MimeList,"Binary Variable")
Calculate iFileRow.C2 as "tData"
```

MimeList is a local list variable, and tData is a task variable of type Binary – it's important to note that the second column of iFileRow is of type Character, and is set to the ***Name*** of the binary task variable. Elsewhere on the form is a picture control to display the photo, its $dataname is iPicture, and a button to allow the end user to select an image file and initiate the upload process. The code for the button is:

```
On evClick
   Do $cinst.$objs.fileControl.$action.$assign(kJSFileActionUpload)
```

This assigns the kJSFileActionUpload action to the fileControl object which opens the Browse/Upload dialog. The File control has an $event method which detects when a file has been uploaded and the upload dialog has been closed:

```
On evFileUploaded
Calculate iPicture as tData
; transfer the pic data to iPicture
On evFileUploadDialogClosed
Do method setPicBtnTitle (kHavePic) ;; changes the button text
```

When an image is selected by the end user and the file is uploaded the image data is loaded into the tData task variable, as defined in the second column of the iFileRow row variable, which is then tranfered to the iPicture variable assigned to the picture field on the form. The Save button saves the contact record including the image file uploaded by the user.

HTML Object

The HTML Object lets you display an HTML page or fragment on the remote form. The $html property contains the HTML content for the control. This must start with an element declaration such as <div...> or <style>. The $ctrlname is the name of the control.

The following method constructs some HTML and assigns it to the HTML control called HTML.

```
; lHTML is a local var of Character type
Begin statement
;  ===== Styles =====
Sta: <style>
Sta: p {color: #00F; margin: 0 2em; padding: 1em; background-color:
   #fff; border: #DDD solid 2px;}
Sta: h1, h2 { color: #666; margin-left: 0.5em;}
Sta: img {margin: 5px 0 5px 50px;}
Sta: </style>

;  ===== Content =====
Sta: <div>
Sta: <h1>HTML Control</h1>
Sta: <p>
Sta: You can display HTML in the HTML control.<br />
Sta: This is the Second line of content.<br /><br />
Sta: <i>And you can do italic etc!</i>
Sta: </p>
Sta: <h2>Pictures</h2>
Sta: <p>You can embed pictures.</p>
Sta: <img src="/images/cat.jpg" width="200" height="176" alt="cat"
   />
Sta: </div>
End statement
Get statement lHTML
```

```
; the html form object is called HTML
Calculate $cinst.$objs.HTML.$html as lHTML
```

You can embed any of the standard HTML tags into the $html property, including links, styles, and tables. The following text could be added to an HTML control which is placed next to a check box control to allow end users to popup a window containing competition rules that are stored in a static html page:

```
<p>I have read and agree to the <a href="rules.html"
   target="_blank">competition rules</a></p>
```

Events

The HTML Control reports the evBefore and evAfter events, as well as evClick and evDoubleClick.

Attribute placeholders

You can style the text displayed in the HTML object using the font and style properties you specify in the control rather than having to add specific css styling to the html. To inherit the effects and font attributes that you select in the Property Manager in design mode you can add placeholders in the $html property in the format %<letter>, such as %f and %t, as follows:

%b	back color and alpha
%f	font
%z	font size
%s	font style
%j	font align
%t	text color
%e	effect
%p	position (coordinates of client)
%v	vert scroll
%h	horz scroll

For example, you can set $html to <div %e> and the text in the control will take on the effect from $effect (or $linestyle or $bordercolor, as these are effect properties).

HTML on subforms

If the HTML object appears on a subform, you will need to add the following style parameter to the tag containing your Html to allow the text to wrap correctly inside the object:

```
<div style="white-space:normal">
```

Hyperlink Control

The Hyperlink Control allows you to present a list of options to the end user, where each option in the list is displayed as a web-style hyperlink; each link option corresponds to a line in the underlying list variable used to populate the list. If the whole list does not fit inside the control, the list will “pop up” when the end user passes the mouse over the control. The Hyperlink component has the following properties:

Property	Description
$isvertical	if kTrue the list pops up vertically
$dataname	A list variable; the first column of the list is used to create the list of options
$shouldunderline	if kTrue the links are underlined
$selectedtextcolor	The color of the text when the mouse is over it

The options in the Hyperlink list are based on the contents of a list variable (which requires a different format to the existing web client Hyplinks component). The first column in the list specified in $dataname is the text displayed for each option listed in the control. When the end user clicks on the list an evClick is triggered with the line number reported in pLineNumber. You can use the value of pLineNumber in your event method to trigger an action. You can create an empty line by putting ”-“ (hyphen) in the first column.

There is an example library showing how you can build a dymamic hierarchical list using the Hyperlink control in the new JavaScript Component gallery on the Omnis website at: www.tigerlogic.com/omnis

Label Object

The Label object lets you add standard labels to the fields on your remote form. The text for the label is entered into the $text property. The font and alignment of the label is setup under the Text tab in the Property Manager.

List Control

The standard List control allows you to display a single column list on your form allowing end users to select a particular line. For lists with many columns you may prefer to use a data grid, or for a more compact single-column list you can use a droplist. If you want to include an icon next to the text in each list line you can use a Tree list by including your data and icon references at the top-level of the list data without using any child data.

The contents of a standard List is taken from the instance variable (of List type) specified in the $dataname of the control. In this case, the first column of the list contents is used to populate the list control. You can build the list via the $construct method of the remote form; the list can be built from your database or from a number of static values, as follows:

```
Do iListvar.$define(iCol)     ;; defines the list
Do iListvar.$add("Value1")    ;; add the values
Do iListvar.$add("Value2")
Do iListvar.$add("Value3")
Do iListvar.$line.$assign(1)  ;; selects the first line
```

You can control the color for selected lines by setting $selectedlinecolor; use kColorDefault for the default selected line color for the current OS. The $evenrowcolor property sets the background color of even numbered rows displayed in the list; kColorDefault means use the same color as the odd numbered rows ($backcolor).

If $ischecklist is kTrue each list line has a checkbox which allows the end user to select or de-select the line.

When the user selects a line the evClick event is reported with the pLineNumber parameter reporting the selected line number. You can use the value of pLineNumber in your event method behind the list to trigger an action.

Navigation Bar Control

The Navigation Bar Control provides a standard navigation bar which end users can use to navigate to different parts of your application. The navigation bar has a main title in the middle of the control, and it can have a left and/or right button which responds to user clicks. The Nav bar can be connected to a paged pane via the $linkedobject property to allow you to display different fields in your form. Actions for the nav bar can be stacked up using the $push property: see example below.

Property	Description
$initiallefticonid	if this is not zero, and $initiallefttext is empty, the first navigation bar item has a button on the left hand side, displaying this icon
$initiallefttext	if this is not empty, the first navigation bar item has a button on the left hand side, displaying this text
$initialrighticonid	the icon for the initial navigation bar button on the right
$initialrighttext	the text for the initial navigation bar button on the right
$initialtitle	is the initial title displayed on the navigation bar
$lefthidden	if true, the left hand (back) button is hidden for the current navigation bar stack item
$linkedobject	the name a paged pane on the current remote form, used in conjunction with the $push property
$push	At runtime, allows you to assign a 2-5 column row to the paged pane referenced in $linkedobject: col1 is the page number of the paged pane, col2 is the title for pushed item, col3 is the text for right button (pass empty for no right button), col4 is the or icon id for the image for right button, and col5 can be non-zero to hide the left button
$righticonid	if this is not zero, and $righttext is empty, the current navigation bar item has a button on the right hand side, displaying this icon
$righttext	if this is not empty, the current navigation bar item has a button on the right hand side, displaying this text
$::title	the title for the current navigation bar stack item

The Nav Bar reports evClickInitialLeftButton when the initial left button has been clicked, and evClickRightButton when the right button has been clicked.

Example Navigation bar

The following Navigation Bar has a main title and a button on the right which is used to display some information on the second pane of a paged pane.

The Nav Bar is placed across the top of the form and its various properties under the General and Appearance tabs in the Property Manager are set, as follows:

$events	set to receive evClickRightButton events
$linkedobject	set to pPane, the name of the paged pane
$push	can only be assigned at runtime; see below
$initialtitle	set to “Main Page”
$initialrighticonid	set to 1794, the id of an icon in Omnispic
$initialrighttext	set to “Info”

The $event() method for the Nav Bar ($name = oNav) traps a user click on the right button, and has the following event code:

```
On evClickRightButton
   Do $cinst.$doPush(2,'Info','','','')
```

The $doPush method is a class method in the remote form and has the following parameters, variables, and code.

```
; PaneNo (Short Int)
; Title, RightBtnText, RightBtnIcon, NoLeftBtn (all Character)
; lRow is local var of type Row
Do lRow.$define(
        'New class','Title','text for right btn',
        'icon for right btn','no left btn')
Do lRow.$assigncols(
        PaneNo,Title,RightBtnText,RightBtnIcon,NoLeftBtn)
Do $cinst.$objs.oNav.$push.$assign(lRow)
```

The effect assigning to the $push property is to change the pane number in the paged pane specified in the $linkedobject property which, in this case, displays some information for the end user on the second pane.

Page Control

The Page Control links to a Paged pane on a remote form and allows the end user to change the current page in the linked paged pane by swiping over the page control or clicking for non-touch screens. The paged pane linked to the Page control is specified in the $linkedobject property. In this case, when the page control is clicked the linked paged pane control will select the next available pane automatically.

The Page control also gives the end user a visual clue as to the current selected pane in the linked page pane object, since the highlighted dot in the control changes to reflect the current page in the linked paged pane.

Property	Description
$::currentpage	the current page number
$linkedobject	the name of a paged pane object on the current remote form that links to the iPage control
$::pagecount	the number of pages

When the page indicator changes in the Page control an evPageChanged is triggered containing the number of the new page in pValue.

Paged Pane

The Paged Pane provides a very convenient method to show a number fields or controls on separate panes, or to break down an entry form into more manageable parts whereby each pane contains a small number of fields. The $pagecount property specifies the number of panes, and $currentpage specifies the current pane. In design mode, you have to set $currentpage to the number of the pane you wish to add fields to, or you can right-click the background of the paged pane and select the number of the pane you want to edit. You can set $effect to select different border effects for the control (a kJSborder... constant).

You can link a paged pane to a Navigation Bar, Page Control, or Tab Bar control so when the nav bar, page or tab changes the current pane of the paged pane changes accordingly. To link a paged pane to one of these controls, set the $linkedobject property of the Nav bar, page control or Tab bar to the name of the paged pane.

By setting $scrolltochangepage to kTrue the pages are laid out horizontally, and the end user can change the current page by scrolling horizontally (for touch devices the end user can change panes by tapping on the current one); in this case, an evUserChangedPage is triggered with the new page number reported in pPageNumber.

Using $dataname

The Page Pane control has a $dataname property which you can use to set the value of $currentpage. When the form is opened or redrawn the numeric value of $dataname is used to set the current page. If $dataname is empty or returns an invalid page number, the control uses the page number in $currentpage.

Page panes in Complex grids

You can use a page pane in a complex grid. You can set the value of $currentpage by assigning a column in the complex grid list to $dataname of the page pane. Therefore each row in the complex grid could display a different page in the page pane control.

In this case, controls within the page pane control will also get their data from the complex grid control, if their $dataname refers to a column in the list used to build the complex grid.

Picture Control

The Picture control allows you to display an image in your form: you can display an image file in a folder, an image from a database, or an icon, depending on the combination of settings of $dataname, $mediatype and $iconid. If $mediatype is empty (and $iconid is zero), the $dataname of the Picture control is a URL to the image to be displayed, relative to your html page containing the JavaScript Client. If $mediatype is specified, then $dataname is the name of a binary instance variable containing the image data: in this case, $mediatype can be set to one of the standard image types, e.g. image/png, image/jpeg or image/gif. Alternatively, the picture control can display an icon in an icon data file (Omnispic) or #ICONS by setting $iconid to a numeric icon ID, overriding the $dataname and $mediatype properties.

$picturealign is a kPAL... constant which, together with $horzscroll and $vertscroll, identifies where the picture will be positioned in the control. If true, $noscale ensures the images displayed in the control are not scaled.

In the **Webshop** sample app, the product images are shown in a Picture control embedded in the Complex grid control on the main jsShop remote form. In this case, $mediatype is set to JPG and the $dataname of the control is iProductList.product_picture which holds the image data for each product.

Popup Menu Control

The Popup menu is a menu that pops up when the user clicks on the header of the control, or, when $hotwhenmouseover is true, the menu will pop up when the end user's mouse hovers over the control. The contents of the popup menu can be a remote menu class specified in $::menuname, or taken from a list variable specified in $::listname; when specifying one of these properties, the other property must be empty. See the *Reference* section for a complete list of properties of the popup menu control.

When using a list variable to populate the popup menu, the data in the column specified in $coltext is used for the menu options. The $colenabled property is the column name for the menu line enabled state, and $colcommandid is the column name holding the menu line command id.

You can add an icon to the popup menu by setting $iconid to the ID of an icon in an icon file or #ICONS. You can place the icon before or after the menu title by setting $textbeforeicon.

The menu will normally popup when the user clicks on the control, but you can make the menu popup when the end user's mouse passes over the control (make it "hot") by setting

$hotwhenmouseover to kTrue. In addition, if you set $hottitleclicks to kTrue, clicking on the control will generate an evClick event.

You can control the position of the popup by setting $menupos to one of the constants: kJSPopMenuPosBottom, kJSPopMenuPosRight, or kJSPopMenuPosTop.

When the menu is clicked the evClick event is triggered with the selected line reported in pLinenumber. You can use the following code to trap the line number:

```
On evClick
   If pLineNumber>0   ;; a line was selected
      ;; Do something
   End If
```

You can use an HTML <select> tag for the user interface by setting $usehtmlselect to kTrue.

Progress Bar Control

The Progress Bar control lets you display a progress bar in your remote form. The value of the progress bar is reported in the $value property. The range for the progress bar is from zero to the value in $max. The Progress Bar control has the following properties:

Property	Description
$max	The maximum value of the progress bar (a positive integer or zero)
$value	The current value of the progress bar. Must be between 0 and $max inclusive
$sendcarryon	Triggers an evCarryOn event (must be executed at runtime)
$progresscolor	The color of the bar representing completed progress. Only applies when the standard HTML5 progress control is not available

The progress control does not store a value (like the fat/web client control). Rather, $sendcarryon provides a mechanism to send an evCarryOn event to the progress control. To generate an evCarryOn event, assign kTrue to $sendcarryon. The event processing code for evCarryOn can assign $sendcarryon to kTrue again, to generate the next evCarryOn.

The following example assumes the progress control has been added to a remote form and a button is used to initiate some process and send a carryon event to the progress itself. The initial values for $::value and $::max of the progress control are 0 and 100 respectively. The following code could be behind a button:

```
On evClick
   ;; initiate some process and
   Calculate iCancelled as kFalse
   Calculate iValue as 1
   Do $cinst.$objs.ProgressBar.$sendcarryon.$assign(kTrue)
```

The evCarryOn event is sent to the progress bar which has the following event method:

```
On evCarryOn
   If not(iCancelled)
      Calculate iValue as iValue+1
      Do $cinst.$objs.ProgressBar.$::value.$assign(iValue)
      If iValue<100
         Do $cinst.$objs.ProgressBar.$sendcarryon.$assign(kTrue)
      End If
   End If
```

RadioGroup Control

Radio Groups present a number of mutually exclusive buttons that can be either on or off: selecting one of the radio buttons deselects all other buttons in that group. The variable you assign to a radio group should be numeric. Its value is within the range $minvalue and $maxvalue inclusive and directly corresponds to which button in the group is selected, that is, the first button selects the first value in the range, the second button the second value, and so on. The labels for the buttons are assigned in $text which is a comma-separated list. The Radio group has the following properties:

Property	Description
$dataname	a numeric variable
$horizontal	If true, the radio column order is horizontal
$columncount	The number of columns shown for the radio group
$minvalue	The minimum value for the radio group
$maxvalue	The maximum value for the radio group
$text	Comma-separated list of labels assigned to the buttons

The evClick event is reported with pNewValue containing the value selected.

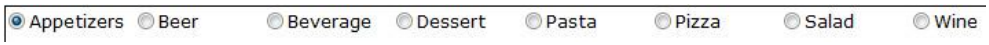
The **Webshop** app uses a Radiogroup control to allow the end user to select a group or category of products to be shown in the main product list. (To examine this control and its properties and methods, open the webshop library and the jsShop remote form.) The $minvalue and $maxvalue of the radiogroup are set to 0 and 8, respectively (although the groups are generated dynamically in the form), and the numeric variable iRadioGroup with an initial value of 1 is assigned to $dataname. When the form is opened, the $construct() method behind the radiogroup calls a $build method.

```
; radiogroup control is called 'filter' containing
; $construct method which is run when the form opens
Do iGroupList.$definefromsqlclass($tables.T_qGroups)
Do $cfield.$build()
```

The $build method generates a list of product groups from the database, which is then concatenated into a single comma-separated list and assigned to the $text property of the radiogroup.

```
; $build method behind the radiogroup
Do iGroupList.$selectdistinct()
Do iGroupList.$fetch(kFetchAll)
For iGroupList.$line from 1 to iGroupList.$linecount() step 1     ;;
   loop through the list
   Calculate text as con(
        text,mid($prefs.$separators,3,1),iGroupList.product_group)
   ;; uses the localized separator (possibly semicolon)
End For
Calculate text as mid(text,2,len(text))
Do $cfield.$minvalue.$assign(1)
Do $cfield.$maxvalue.$assign(iGroupList.$linecount())
; $maxvalue of radiogroup is set to the number of groups in list
Do $cfield.$text.$assign(text)
```

When the form is opened the main product list is built using a method behind the product list itself (also called $build) and the list initially contains the Appetizers only. A group of radio buttons is created, each item representing a different group or category of food or drink; the initial value of the radiogroup is set to 1 selecting the first item in the group.



When the end user clicks on the radiogroup, to select another product type, the click is detected in the $event method in the radiogroup control, the number of the radio button clicked is passed in pNewVal, and the product list is rebuilt based on the selected product group; note a Where clause is created based on the selected group and sent to the $build method behind the main productList control.

```
; $event method for Radiogroup
On evClick
   Calculate whereClause as con(
      'WHERE product_group =
      ',kSq,iGroupList.[pNewVal].product_group,kSq)
   Do $cinst.$objs.productList.$build(whereClause)
```

Slider Control

The Slider control provides a graphical thumb component that the user can drag to control the numeric setting of another component in your form, such as a volume control. The current value of the slider is reported in the property $val according to where the slider is positioned. You can specify the range for the slider in the $min and $max properties.

Property	Description
$min	The minimum value for the slider
$max	The maximum value for the slider
$val	The current value of the slider
$step	The size of each step the slider takes between min and max
$vertical	If true, the slider is a vertical slider
$sliderhorziconid	The id of the icon to use for a horizontal slider handle
$sliderverticonid	The id of the icon to use for a vertical slider handle
$horzmargin	The horizontal drawing margin
$vertmargin	The vertical drawing margin

The Slider reports three events: evStartSlider (when the control is starting to track), evEndSlider (when the control has finished tracking), and evNewValue (when the value has changed). You can detect these events in the $event() method for the component. These events all pass the current value of the Slider in the pSliderValue parameter. As the user drags the Slider thumb the evNewValue event is triggered and pSliderValue is sent to the $event() method for the Slider.

To use the values of the slider in your remote form you can trap the slider events in the $event method of the slider control and transfer the current values to instance variables in your form, as follows:

```
On evStartSlider
   Calculate iStartValue as pSliderValue
On evEndSlider
   Calculate iEndValue as pSliderValue
On evNewValue
   Calculate iNewValue as pSliderValue
```

Subform

The Subform component is a standard Omnis component, but is available in the ‘JavaScript Components’ group in the Component Store. It allows you to place another remote form inside the main remote form. In this manner, using a single “main” form and a number of other remote forms loaded at runtime into a subform control, would allow you to create a powerful and interactive web application with many subforms or layers. Using a subform to display multiple forms is rather like using an iframe in a standard HTML page.

When you have placed the subform on your remote form, you specify the initial remote form to appear in the subform in its $classname property. Alternatively, you can switch subforms at runtime by assigning a new remote form to $classname to switch the current form displayed in the subform control, as follows:

```
; oSub is the name of the Subform control on the main Remote form
Calculate $cinst.$objs.oSub.$classname as NewFormName
```

The **Holidays** sample app uses a subform to display either the User or Admin form. When the main jsHolidays remote form is loaded, its $construct() method calls a class method to setup the initial subform to be shown which, in this case, is the User form. In addition, there are buttons on the main Holidays form to allow the end user to switch forms; for example, the code behind the User button is:

```
; $event method for User button on jsHolidays
On evClick
   Do method setSubForm (kUserForm)
   Calculate iAdminBtnState as kTrue
   Calculate iUserBtnState as kFalse
   Do $cinst.$objs.adminBtn.$enabled.$assign(kTrue)
   Do $cobj.$enabled.$assign(kFalse)
```

The setSubForm method has the following code to switch forms:

```
If pOption=kAdminForm   ;; test if Admin or User
   Do $cinst.$objs.subForm.$classname.$assign("jsAdminForm")
Else
   Do $cinst.$objs.subForm.$classname.$assign("jsUserForm")
End If
```

$dataname and subforms

If the remote form class inside the subform field has only one field you can override its dataname using the $dataname property for the subform field. For example, your subform class may contain a single headed list field that takes its data from a particular list variable. However you can change the list assigned to the headed list by setting the subform field's $dataname property to the name of another list. You could do this in the $construct() method of the subform field.

$contruct and subforms

Opening a remote form containing a subform field or any number of subform objects creates an instance of each form, which belong to the same task as the parent remote form instance. Omnis calls the $construct() methods of all the subform classes first in tabbing order, then the $construct() method of the parent form instance. The reverse happens on closing the parent form, with the subforms being destructed after the parent form instance.

You can send parameters to the subform's $construct() method by including a list of parameters in the $parameters property when you create or modify the subform field.

Subform caching

The $multipleclasses property tells Omnis to keep a set of remote form instances open for use in the subform object, rather than constructing a new instance each time the class is changed. When you assign a new remote form name to $classname at runtime, the new remote form is downloaded to the client and displayed in the client's browser. If the $multipleclasses property is enabled, the previous remote form is cached and hidden on the client, otherwise the remote form instance is destroyed. If any previous remote forms have been cached in this way using $multipleclasses, you can switch back to them instantaneously, otherwise they have to be reloaded each time you assign to $classname of the subform object.

Referencing Subform instances

A subform control has the $subinst property (object) which is the instance contained within the subform control. You can use this property to get a reference to the instance in a subform object and therefore change properties within the instance. If the subform property $multipleclasses is set to kTrue, you must use $subinst(cClassName) to get a reference for the appropriate instance. For example, where a subform control has a single subform class the following code will return a reference to the form instance in the subform:

```
Set reference item to $cinst.$objs.subfrm.$subinst
Do item.$setcolor(kRed)
```

or when you may have assigned multiple classes to the subform control ($multipleclasses is set to kTrue):

```
Set reference item to $cinst.$objs.subfrm.$subinst("classname")
Do item.$setcolor(kRed)
```

Note that the item returned will be null if the instance does not exist.

Switch Control

The Switch control is like a check box insofar as it represents an On / Off value (1 or zero). The variable you specify in the $dataname property should be a Number or Boolean variable. The $switchon and $switchoff properties let you specify the icon IDs for the images to be used when the switch is either on or off.

There is an example library showing the Switch control with a range of different ON/ OFF images in the new JavaScript Component gallery on the Omnis website at: www.tigerlogic.com/omnis

Tab Control

The Tab Control allows the end user to select a tab which can correspond to a specific option in your application. There are many properties under the Appearance tab in the Property Manager to control the general appearance of the Tab Control and the tabs themselves. In particular, you can specify the position of the text and an icon for each tab by setting $tablayout. You can specify the properties for the current selected tab, set in $selectedtab in design mode, under the Tab tab in the Property Manager. See the *Reference* section for a full list of properties for the Tab control.

The Tab Control can be linked to a paged pane (by setting $linkedobject) so when different tabs are clicked, the pane in the linked Page Pane control is changed accordingly; in this case, assigning a new value to the $currenttab tab property at runtime will also change the current pane in the linked Paged Pane control.

The Tab Control can also be linked to a remote menu; clicking on a tab will trigger the corresponding line in the menu. To implement this, the $trackmenus property must be kTrue. The $tabmenu property is the remote menu for the selected tab (the tab with number $selectedtab). If assigned at runtime, the menu instance must already be present on the client (via a $tabmenu or $contextmenu property in the class data when the form was loaded).

In design mode, you can move or re-order the tabs in the control by entering a number into the $movetab property; in effect, the number of the selected tab will become the number you entered, and the other tabs are shuffled along. This is useful if you have setup multiple tabs and need to move a tab easily without having to redefine each tab again.

When the end user clicks on a tab the evTabSelected event is triggered with the new tab number reported in pTabNumber.

The **Contacts** sample app users a Tab control in the main jsContacts remote form to allow the end user to switch from viewing a list of contacts to a form showing details of individual contacts; in this case, the Tab control is linked to a page pane which displays the contact list or contact details view. The $linkedobject property of the contactTabStrip control is set to ‘pagePane’ (the name of the page pane) and the $tabtext for each tab is defined as ‘Contacts’ and ‘Details’ respectively. In addition, the evTabSelected event is enabled in the $events property of the Tab control. The code for the Tab control $event method is:

```
; $event method for Tab control
On evTabSelected
   If pTabNumber=2
      Do method loadRecord (iContactList.$line)
      Do $cinst.$objs.saveBtn.$enabled.$assign(kTrue)
   Else If pTabNumber=1
      Calculate iNewContact as kFalse
      Do $cinst.$objs.saveBtn.$enabled.$assign(kFalse)
   End If
```

If the Details tab is clicked, the second tab, the second pane in the page pane is displayed and the details for the currently selected contact are loaded using the loadRecord class method.

Timer Control

The Timer is an invisible component that triggers an evTimer event after a specified time while $running is set to kTrue. You can specify a $timervalue, which is interpreted as an interval in seconds or milliseconds according to $useseconds, which should be set to kTrue for seconds or kFalse for milliseconds (the default).

In the **Webshop** sample app it would be possible to use the Timer control to rebuild the Orders list periodically; in this case, you could use the Timer object to run a method at a given interval to rebuild the Orders list in the jsShopOwner remote form. In reality, the Orders list is rebuilt every time a new order is placed, but using the Timer object could be used to regulate the rebuilding of the Orders list.

To implement a Timer in the Webshop app, you would need to enabled the evTimer event in the $events property of the Timer control. The $useseconds property is set to kTrue and $timervalue is set to 2 (to give an interval of 2 seconds). The $construct() method of the Orders list (a data grid) includes a line of code to start the timer:

```
; $construct method for Orders list/data grid
Do iOrderList.$definefromsqlclass($tables.T_qOrders)
Do $cinst.$objs.timer.$running.$assign(kTrue)     ;; possible using
   a timer object
Do $cfield.$build()
```

The $build() method behind the Orders list builds the list when the form is opened, but when used with the Timer object it can be used to rebuild the list as well. The $event method for the Timer object is run every 2 seconds, and has the following code:

```
On evTimer
   Do $cinst.$objs.dataGrid.$build()
```

Tree Control

A Tree list provides a graphical way of displaying a list of items in a hierarchical format. Each node can have a check box or its own icon. Tree lists have the following properties:

Property	Description
$checkbox	If true, and $multipleselect is also true, the tree control has check boxes that can be used to select nodes
$datamode	Controls how the list content is used to structure the tree list, a constant: kJSTreeFlatList, kJSTreeFlatListWithTags kJSTreeFlatListOld, kJSTreeFlatListOldWithTags
$dataname	A list variable to specify content and structure of tree list
$iconurlprefix	All icons used in the tree (as a result of $showicons being true) must come from a single icon directory; the default is _icons/omnispic/
$showicons	If true, the tree control shows node icons from location in $iconurlprefix
$showlines	If true, the tree control displays dotted lines connecting nodes
$twostate	If true, and the tree control has checkboxes (see $checkbox), selection of each node is independent

Tree list format

The JavaScript Tree list component requires a similar list format to those in the web/fat clients, as follows. The $datamode property controls how the list content is used to structure the tree list with four possible list formats (constants):

- **kJSTreeFlatList**
  The first N columns represent a node in a tree of depth N.The last 5 columns are node properties:

iconid,ident(int),expanded(bool),textcolor(zero means $textcolor),tooltip

- **kJSTreeFlatListWithTags**
  The first N columns represent a node in a tree of depth N.The last 7 columns are node properties:

iconid,ident(int),expanded(bool),textcolor(zero means $textcolor), tooltip, tag(char), enterable(bool)

- **kJSTreeFlatListOld**
  A list with the same structure as the plugin client tree kTreeDataFlatList
- **kJSTreeFlatListOldWithTags**
  A list compatible with the plugin client kTreeDataFlatListWithTags

Tree Events

When the user selects a node the evClick event is reported, while a double click reports an evDoubleClick. In both cases, the id and tag of the selected node is reported in the pNodeIdent and pNodeTag parameters. Note you cannot get click or double click events for nodes which are enterable, as the click puts them into edit mode.

For enterable nodes, the evRenamed event is reported if the end user has renamed the node. evRenamed has node ident, node tag, old name and new name as event parameters.

Video Control

The Video control allows you to play a video within your remote form. The video can be hosted on YouTube or you can play a video through a Flowplayer client.

The Video control allows you to play a video within your remote form. You can play videos hosted on YouTube, or you can play video files directly accessed by a URL, using the browser's HTML5 video playing capabilities.

YouTube

If you set the $youtube property to kTrue, the $dataname for the video control should be a list containing YouTube ids: the data in the first column of the first row in the list is used to reference the video. Note the YouTube id is not the full URL on youtube.com, but just the id on the end of the URL, e.g. 'Ff-qlTlSkc0' (which in this case will play a video from the EurOmnis 2010 developer conference).

HTML5

If you set $youtube to kFalse, the $dataname should be a 2-columned list. The first column should contain URLs to the video files, and the second column should state the media type. For example:

```
Do iList.$define('VideoURL','VideoType')
Do iList.$add('videos/myVideo.mp4','video/mp4')
Do iList.$add('videos/myVideo.ogv','video/ogg')
Do iList.$add('videos/myVideo.webm','video/webm')
```

Not all browsers are able to play all video file types, so you should provide the video in multiple formats and populate the list with the URL to each file and type. When the client connects, the browser will play the first video file it is able to play from the list.

Flowplayer

Not all browsers support HTML5 video so it is possible to play your video using flowplayer which uses Flash. To do this, you should point the $flowplayerurl property to a flowplayer object, which you can download from flowplayer.org. You then need to set $flowplayerline to the line in your list of video files defined in $dataname which contains the video file that can be played in flowplayer.

Property	Description
$dataname	If $youtube=kTrue: list containing YouTube ids; the id of the video in the first column of the first row is played If $youtube=kFalse: 2 columned list containing URLs to video files in Column 1 and media types in column 2
$showcontrols	If true, the control displays video controls such as the play and pause buttons
$flowplayerline	If not zero, the line number in the $dataname list of the video content to be used for the fallback Flowplayer object
$flowplayerurl	The URL of the Flowplayer to be used as a fallback when HTML5 video is not available
$youtube	If true, the control will play a movie from youtube.com; row 1 of column 1 of the $dataname list is the YouTube video id If false, the control will use HTML5 video (or Flash if HTML5 is not supported) to play video files from URLs

Deploying your Web or Mobile app

Deploying an Omnis app that uses the new JavaScript remote forms is essentially the same as deploying a Web Client based application, except that the end user does not have to install a plug-in and there are no other component files. Once setup, you only need to provide your customers or end-users with a URL to run your Omnis app on a desktop PC or mobile device.

The essential steps in deploying your JavScript based apps are:

- edit your HTML pages containing the JavaScript Client
- install and configure the **Omnis Server,** and add your Omnis library to the Server tree
- install and configure your **Web Server** including the web server plug-in or Java Servlet supplied by TigerLogic
- copy your HTML files to your web server

For further details about setting up the Omnis Server and your web server, refer to Chapter 1 in the *Extending Omnis* manual (available to download from the Omnis website www.tigerlogic.com/omnis).

Editing your HTML Pages

Omnis creates an HTML page when you test your JavaScript remote form (using Ctrl-T) that contains details of the JavaScript Client object, your remote form class, the location of your web server, and so on. You can use this file for deploying your web application, but you will probably want to edit this file, or copy the relevant lines of code containing the JavaScript Client object to your own HTML pages.

The test HTML file has the same name as your remote form plus the .htm extension and is located in the **html** folder under the main Omnis folder. For example, under Windows Vista/Windows 7 the HTML template is located in your AppData\Local folder, such as:

```
C:\Users\<username>\AppData\Local\TigerLogic\omnis\html
```

You can edit the test HTML in a standard web page design tool or a text editor, such as Notepad under Windows.

CSS styles and JavaScript folders

Note that the Omnis **html** folder contains a number of other folders, including **css** and **scripts**, which contain CSS style sheets and JavaScript files which are required to run the Omnis JavaScript Client: *these folders and their contents must be copied* to the same location relative to the HTML page containing your Omnis app. The html folder also contains the icons folder containing all the icons generated during testing, some of which may be required for your application: see the *Component Icons* section for more details.

Fav icon

The icon used for the test HTML page (displayed in the top-left of the browser tab) is the image file called 'favicon.ico' located in the html/images folder. The image file in the development version of Omnis is the blue Omnis logo, but for deployment you can replace this image with your own icon file.

The JavaScript Client Object

The JavaScript Client object is embedded in a in the HTML template with the id="omnisobject1" as follows:

```
<div id="omnisobject1" style="position:absolute;top:0px;left:0px"
   WebServerURL="_PS_" OmnisServerAndPort=""
   OmnisLibrary="myLibrary" OmnisClass="rfRemoteFormName" param1=""
   param2="">
```

The position of the "omnisobject1" within the HTML page is where the Omnis JavaScript Client remote form will be embedded in the browser. By default, the JavaScript Client is displayed in the top left corner of the client browser, but you can reposition the JavaScript Client object to appear anywhere in your HTML page using alternative style parameters. For example, replacing the style parameter in the div tag on the JavaScript Client object with style="width:900px; margin:auto" will center the remote form in the browser (width should be set to the width of your remote form).

You can add another Omnis object to the same HTML page, but it must have a unique id, such as "omnisobject2", and you can set its own server, library, and form parameters.

For example, the HTML page containing the Omnis quiz (available for the Studio 5.2 release) has the following tag containing the JavaScript Client object and its parameters:

```
<div id="omnisobject1" style="position:absolute;top:0px;left:0px"
   WebServerURL="http://194.131.70.208/cgi-bin/js/omnisapi.dll"
   OmnisServerAndPort="5775" OmnisLibrary="STUDIO52QUIZ"
   OmnisClass="jsQuiz" data-screensize="window" param1=""
   param2=""></div>
```

WebServerURL

The **WebServerURL** property identifies the HTTP URL of either the Omnis development environment/runtime or the Web Server plug-in (the latter case applies when using the Omnis web server plugin to run the Omnis application server along-side a Web Server).

When testing with the development version of Omnis, the WebServerURL is set to "_PS_" which will be replaced with the address of the computer from which this HTML page is being served, that is, your development computer. For example, if the test HTML page is at http://127.0.0.1:5000/jschtml/test.htm, then _PS_ will be replaced with http://127.0.0.1:5000. For testing, OmnisServerAndPort will be empty.

For deployment, when using a Web Server, the WebServerURL parameter should be set to the location of the Omnis web server plug-in, such as http://www.myhost.com/scripts/omnisapi.dll, .which handles all the communication between the server and the client.

OmnisServerAndPort

For development and testing **OmnisServerAndPort** can be empty. For deployment, OmnisServerAndPort tells the Omnis web server plug-in how to connect to the Omnis Application Server. If the Omnis Server is on the same machine as the web server, then OmnisServerAndPort can be the port number of the Omnis Server, e.g. it could be "5000" if the Omnis Server is at port 5000 on the same machine as the web server. If the Omnis Server is on a different server from the web server, then the OmnisServerAndPort parameter must be "IP-Address:Port-number" of the Omnis Server, e.g. it could be "111.222.000.111:5000" if the Omnis Server is at port 5000 on a machine with IP address 111.222.000.111.

OmnisLibrary and OmnisClass

The **OmnisLibrary** parameter identifies the library containing your remote form, and **OmnisClass** is the remote form class displayed by the omnisform object.

Additional and Custom Parameters

You can specify up to nine additional parameters (named param1, param2, ..) which can be passed into the row variable parameter of the $construct method of the remote task assigned to the remote form.

In addition to the pre-defined parameters, you can include your own custom parameters if you wish to pass extra values to the task or form $construct method. Custom parameters should be prefixed with "data-" added to the parameter name. The parameter will be added to the construct row variable which you can interrogate in your task or form $construct method.

As a special case, you can add a parameter to the JavaScript object to force a remote form to resize when the end-user resizes a browser window; this applies only when the remote form is opened in a desktop browser and may provide a convenient method to test all the possible screen sizes defined in your remote form (under the $screensize property). To do this, add a parameter called data-screensize="window" to the JavaScript Client object in your HTML file. When you test your remote form in a desktop browser and resize the

browser window, your remote form will switch form size and orientation automatically (assuming you have added alternative layouts to $screensize). Note that the data-screensize parameter only has one possible setting at present.

Omnis Server licensing

When you are ready to deploy your Omnis app you will need to purchase an Omnis Server license to run your app in a web or mobile environment. There are new Omnis Server deployment licenses for running Web and/or Mobile apps in the JavaScript Client. Please refer to www.tigerlogic.com/omnis, or contact your local sales office for details about these new Omnis Server deployment licenses.

Setting up the Omnis Server

Setting up the Omnis Server is the same as in previous versions since you will need to run your Omnis application (library) on the Omnis Server. You will also need to setup a standard web server to host the HTML page or pages containing the embedded JavaScript Client code which references your Omnis app. You will also need to install the Omnis web server plug-in into your web server.

You don't need to provide your end users with the Web Client plug-in installers, or setup the Component Manager since the Web Client plug-in is not required and there are no components to download or update.

Creating Standalone Mobile Apps

In addition to using the new JavaScript Client in the browser on any computer, tablet or mobile device, you can create standalone apps for iOS and Android based devices that have your JavaScript remote form embedded. To do this, there are two custom apps or "wrappers" for the JavaScript Client, one for iOS, the other for Android. The custom apps create a thin layer around a simple Web Viewer which can load your JavaScript remote form. This capability is similar to using the option in iOS Safari to add a URL to the home screen, insofar as using the custom app allows you to create a clickable app on the Home screen of your end users' mobile device. Using the custom app also properly supports multi-tasking for your Omnis app, whereas Home screen links created with Safari always load the URL from scratch, even if it is already open, which may not be appropriate for your Omnis app.

During development, you can open a JavaScript remote form in the custom app using a new option Test Form Mobile (Ctrl-M), assuming the custom app is setup and enabled. This new option occurs beneath Test Form (Ctrl-T) in the remote form context menu.

Test form mobile is only displayed in the relevant menus when both:

1. The custom app is enabled for test form (see the menu of the Android app, and the system settings for the iOS app)
2. The custom app is connected to the Omnis Application Server, using the test form parameters.

The $designshowmobiletitle property has been added to JavaScript remote forms. This affects whether the title of the custom app is visible when Test form mobile (Ctrl-M) is used. For deployment, config.xml allows you to configure whether or not the title is displayed in the custom app.

Please note, further information about using the Custom apps for deployment is available in a Tech note on the Omnis website at: www.tigerlogic.com/omnis

Configuring the Custom app

The custom apps can be configured to run a single JavaScript remote form as the entry point to the app, which, for deployment, can be configured in a config.xml file. The config.xml file contains the URL for the page containing your JavaScript remote form and depending on the platform, may contain a number of other parameters, as described below.

iOS

The iOS config.xml file allows you to hide or show the title. No other parameters are available.

```
<?xml version="1.0" encoding="UTF-8"?>
<OmnisiOSConfig>
   <URL title="1">
      http://194.131.70.199:5000/jschtml/RemoteFormName.htm
   </URL>
</OmnisiOSConfig>
```

Android

Due to the large number of devices available on Android, including smartphones, tablets and desktop devices, the Android custom app has some additional parameters which allow you to control the scaling of forms on devices which are not the standard Omnis sizes (which are 320x480 and 768x1024).

```
<?xml version="1.0" encoding="UTF-8"?>
<OmnisAndroidConfig>
   <URL maintainaspectratio="1" scale="0" floatcontrols="1"
   canhorzscroll="0" canvertscroll="0" standardmenu="1" title="1">
      http://194.131.70.199:5000/jschtml/RemoteFormName.htm
   </URL>
</OmnisAndroidConfig>
```

The extra parameters can be set in the config.xml file for the Android custom app, and are as follows:

- **canhorzscroll and canvertscroll**
  set these to "1" if you want to allow horizontal or vertical scrolling of the form respectively, or "0" if not.
- **scale**
  If you set this to "1" (true), the client scales the form to fit the available screen space. The scaling factor is the screen width or height divided by the width or height of the

closest matching $screensize. For these purposes, the actual screen size excludes the operating system areas such as the status bar.

- **maintainaspectratio**
  If you set this to "1", scaling maintains the aspect ratio of the form. When turned on, and depending on canhorzscroll and canvertscroll, it may reduce the scaling factor in one direction, to make the form fit, and center the form vertically or horizontally as required.
- **floatcontrols**
  This property is only significant when scale is "0" (false). In this case, the client uses the new $screensizefloat property of each JavaScript Client control on the form. When applying the screen size, the client uses $screensizefloat to float the edges of controls using the same rules as $edgefloat (note that the component values are not supported, just the edge-related values). If the form is wider or taller than the screen, floating only occurs if the relevant canhorzscroll or canvertscroll parameter is false. The amount by which the controls float is the difference between the actual screen width or height and the designed width or height of the form for the closest matching $screensize. The value of $screensizefloat is stored for each setting of $screensize in the remote form.

You can also change these parameters by pressing the menu button on the Android device, and using the menu options to change them. The app remembers the last setting made via the menu, so the config.xml lets you set the initial values in the custom app.

Custom app source files

The projects and source code for the iOS and Android custom apps are provided in the 'client' folder in the Omnis development tree, as follows:

- iOS custom app
  client\jsclient\wrappers\iOS\OmnisJS_iOS.zip
- Android custom app
  client\jsclient\wrappers\Android\OmnisJS_Android.zip

The ZIP files contain template config.xml files, together with the project and source files so you can customize the wrapper apps if required. If you do not wish to install Eclipse to build the Android wrapper, the Android zip file contains OmnisAndroidClient.apk, which you can install on your device using adb from the Android SDK. The Android client app is built for a minimum Android version of 2.1.

Troubleshooting

Remote tasks

When I try to test my remote form using Ctrl-T, it does not open and the error "To use a remote form class, you must set the design task of the remote form class to a remote task" is displayed.
You need to create a remote task in your library and set the $designtaskname property of the remote form class to the name of the remote task you created. A remote form needs a remote task instance to run (for testing or deployment), so will not open without a remote task and $designtaskname being set.

Remote forms

When I open my blank remote form in design mode I don't see any JavaScript components in the Component Store.
You need to set the $client property of the remote form to kClientJavaScript to use the new JavaScript components. When you use the New Class>>Remote Form option in the Studio Browser, the client property should be set to kClientJavaScript for you automatically. (Note you cannot switch an existing Web Client based remote form to the JavaScript Client, but there is a migration tool available in the Studio Browser to help you move to the new client.)

Events

I have placed an event method behind my button (or any other event-driven object) but when I click it whole testing the form nothing happens.
You must specify which events are to be triggered by the object or remote form in the $events property of the object or form. So for a button, the evClick event must be checked in the $events property of the button. Events for some objects are checked by default but you may like to check the status of $events and make sure the events you need are enabled.

SQLite

Omnis Studio 5.2 contains a new Omnis DAM to support connections to SQLite, a very popular database which is embedded into a whole range of applications on desktop and mobile devices. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private.

SQLite implements a self-contained, server-less, zero-configuration, transactional SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to a disk file which can contain multiple tables, indices, triggers, and views. For more information about SQLite and to download it, please go to the website: www.sqlite.org (portions of this text are taken from the SQLite website).

The SQLite DAM uses the SQLite v3.0 API which uses a different file format to the v2.8 API. Existing v2.8 datafiles must be converted for use with v3.0. See http://www.sqlite.org/version3.html) for more details about SQLite3.

This section contains the additional information you need to access a SQLite database, including server-specific programming, trouble-shooting and data type mapping to and from the database. For additional information on changes to the SQLite DAM, refer to the readme file which accompanies your installation media.

Server-specific Programming

Logging on to SQLite

To connect using the SQLite DAM, create an object variable of subtype “SQLITESESS”.

You connect to a SQLite data file using the $logon() method. The hostname parameter should be the full path to the data file.

SQLite does not require a username or password, but you can specify a session name that will appear in the SQL Browser and in the Notation Inspector under *$sessions*.

SQLite expects a DOS-style pathname under Windows and an absolute POSIX-style path under Mac OS X and Linux. For example:

```
Do mySession.$logon('C:\mydata\mydatafile.db','','','session1')
   Returns #F ;; on Windows

Do mySession.$logon('/Users/MyUser/mydatafile.db','','','session1')
   Returns #F ;; on Mac / Linux
```

Additionally, you can force SQLite to create the specified data file if it does not exist. To do this, set the $opencreate session property to kTrue before logging on.

To open a read-only connection, set the $readonly session property to kTrue before logging on. (It is not possible to create a data file if the connection is read-only).

If the hostname is ":memory:", then a private, temporary in-memory database is created for the connection. This in-memory database will be deleted when the database connection is

closed. Filenames beginning with ":" should be considered reserved for future SQLite extensions and avoided to remove ambiguity. For in-memory databases, $version will be set to ":memory:" following $logon().

If the hostname is an empty string, then a private, temporary on-disk database will be created. This private database will be automatically deleted as soon as the database connection is closed. For temporary databases, $version will be set to ":temporary:" following $logon().

For standard data file connections, $version is read directly from the file header information and reflects the file format version that the data file supports.

Transaction Support

SQLite supports both automatic and manual SQL transactions.

To invoke manual transaction mode, the $transactionmode session property should be set to kSessionTranManual.

In this mode you must commence each transaction by calling the $begin() session method and terminating each transaction either by calling $commit(), $rollback(), by switching back to kSessionTranAutomatic or by logging off.

The SQL text that is submitted each time $begin() is called may be augmented using the $transactiontype session property as shown below. The different transaction types affect the way in which SQLite acquires row locks on tables:

$transactiontype	**Resulting SQL text**	**Meaning**
kSQLiteTranDeferred	BEGIN	No locks are acquired on the database until the database is first accessed
kSQLiteTranExclusive	BEGIN EXCLUSIVE	EXCLUSIVE locks are acquired on all databases as soon as the BEGIN command is executed
kSQLiteTranImmediate	BEGIN IMMEDIATE	RESERVED locks are acquired on all databases as soon as the BEGIN command is executed

The $commit() and $rollback() methods, invoke the COMMIT and ROLLBACK commands respectively.

$commitmode and $rollbackmode are set to kSessionCommitClose and kSessionRollbackClose respectively for the SQLite DAM. Statement objects are closed upon $commit() / $rollback(). Any pending result set is discarded and the statement is returned to its prepared state ready for re-execution if desired.

Incremental BLOB I/O

SQLite supports incremental Input/Output to BLOB columns in database tables. This means that you effectively bind a placeholder for the BLOB at bind time, then write the data to it later. Similarly, you can open a handle to a BLOB which already exists in the database and read/modify its contents without the need to perform a SELECT statement.

To create a placeholder for a BLOB, you should bind the binary variable inside the SQL statement as normal, but set its contents to #NULL. On execution, this creates a *zero-blob* of size *$blobsize*- bytes padded with zeros.

The session object provides several methods for accessing and modifying BLOBs:

- **$blobopen()**
  Opens a handle to a BLOB column, identified by its database name, table name, column name and row number, optionally as read-only
- **$blobclose()**
  Closes a BLOB handle; you have to close any BLOB handles opened during the session, but any handles left open when the session ends are closed automatically
- **$blobcloseall()**
  Closes all BLOB handles
- **$blobbytes()**
  Returns the size in bytes that was allocated to a BLOB column when it was created
- **$blobhandles()**
  Returns a list of all open BLOB handles including the corresponding database, table name, column name and row number
- **$blobreopen()**
  Moves a BLOB handle to a new row within the same table
- **$blobwrite()**
  Writes binary data to a BLOB column
- **$blobread()**
  Reads binary data from a BLOB column

See the Session Methods section for more details and syntax for these methods.

Session Properties

Property	Description
$blobsize	The default value for $blobsize is set at 32KB for the SQLite DAM since this property is used routinely when creating empty BLOB columns for use with incremental input/output methods.
$opencreate	If kTrue, the data file specified at $logon() will be created if it does not exist. If kFalse (the default), an error will be generated if the data file is not found.
$readonly	If kTrue, the connection will be opened in read-only mode. All attempts to write to the data file will fail with an error. If kFalse (the default), read and write operations are permitted.
$transactiontype	Specifies the locking behavior for manual transactions. This is one of the following constants: kSQLiteTranImmediate, kSQLiteTranExclusive or kSQLiteTranDeferred (the default).

Session Methods

Method	Description
$blobbytes()	$blobbytes(iBlobHandle) returns the size in bytes that was allocated to a BLOB column when it was created
$blobclose()	$blobclose(iBlobHandle) closes a BLOB handle. You should close any BLOB handles opened during the session. Any handles left open when the session ends are closed automatically however. Always returns kTrue
$blobcloseall()	$blobcloseall() closes all BLOB handles. This method always returns kTrue
$blobhandles()	$blobhandles(lHandleList) returns a list of all BLOB handles including their corresponding database, table names, column names and row numbers. Aborted/invalid handles are shown with a row number set to zero. Returns kTrue on success, otherwise kFalse
$blobopen()	$blobopen(cDatabase, cTable, cColumn, iRow [,bReadOnly]) opens a handle to a BLOB column, identified by its database name, table name, column name and row number, optionally as read-only. Returns a BLOB handle on success or zero on failure. The SQLite DAM numbers BLOB handles incrementally starting from 1001
$blobread()	$blobread(iBlobHandle, xBinary [,iSize ,iOffset]) reads binary data from a BLOB column into the supplied binary variable. If iSize is omitted, the value of $blobsize is assumed
$blobreopen()	$blobreopen(iBlobHandle, iRow) moves a BLOB handle to a new row within the same table. If iRow exceeds the number of rows in the table, this invalidates the handle. Only the row number can be modified. To change the database, table name or column name, a new handle should be opened
$blobwrite()	$blobwrite() writes binary data to a BLOB column
$lastrowid()	$lastrowid() returns the rowid of the most recent successful INSERT into the database from the current connection
$rowsmodified()	$rowsmodified() returns the total number of database table rows that have been affected by INSERT, UPDATE and DELETE operations since the connection was opened (includes all statement objects)

Data Type Mapping

Omnis to SQLite

The SQLite DAM creates custom data types in order to preserve information about Omnis subtypes, notably: DATE(n) as well as PICTURE, LIST, ROW, OBJECT and OBJECTREF. There are also single mappings for Omnis Character and National data, implying that CHAR(n) and NCHAR(n) can store up to the maximum field length supported by Omnis (10000000 characters). This is contrary to other relational databases which impose a fixed size on such columns.

Although this greatly improves compatibility between Omnis and SQLite, if portability of the data file is of concern, then it may be preferable to avoid using $createnames() / $coltext() in favor of manual statements that use standard SQL types, e.g. VARCHAR(n), DATE, TEXT and BLOB.

Omnis Data Type	SQLite Data Type
CHARACTER	
Character n	CHAR(n)
National n	NCHAR(n)
NUMBER	
Long integer	INTEGER
Short integer	TINYINT
Number 0..14dp	NUMERIC(15, 0..14)
Short number 0/2dp	NUMERIC(9, 0/2)
Number floating dp	FLOAT
DATE/TIME	
Short date 1900..1999 Short date 1980..2079 Short date 2000..2099	DATE(1900) DATE(1980) DATE(2000)
Short time	TIME
Datetime (all subtypes)	TIMESTAMP
OTHER	
Boolean	BIT
Sequence	INTEGER PRIMARY KEY (auto increments when inserted as NULL)
Picture	PICTURE
List	LIST
Row	ROW
Object	OBJECT
Object reference	OBJECTREF
Binary / other	BINARY

SQLite to Omnis

The SQLite DAM recognises several additional SQL data types in order to maximise compatibility with externally generated data files as well as those generated by Omnis.

SQLite Data Type	Omnis Data Type
NUMBER	
TINYINT	Short integer
INT, SMALLINT, INTEGER	Long integer
SEQUENCE, INT AUTO INCREMENT	Sequence
BIGINT	Character*
FLOAT, REAL, DOUBLE	Number floating dp
NUMERIC(p,s), DEC(p,s), DECIMAL (p,s)	Short number s dp (p <=9, s=0 or 2) Number s dp (p <= 15) Number floating dp (p > 15)
DATE/TIME	
DATE(1900) DATE(1980) DATE(2000)	Short date 1900..1999 Short date 1980..2079 Short date 2000..2099
DATE, TIMESTAMP, TIME	Date Time (#FDT)
CHARACTER	
CHAR, VARCHAR, TEXT, CLOB,	Character
NCHAR, NVARCHAR, NATIONAL	National
OTHERS	
BOOLEAN, BOOL, BIT	Boolean
PICTURE	Picture
LIST	List
ROW	Row
OBJECT	Object
OBJECTREF	Object reference
BINARY / other	Binary

*Omnis cannot currently handle 64-bit integers, hence the mapping to Character data.

Troubleshooting

The following points may help in resolving programming issues encountered using SQLite session and statement objects.

For additional updated troubleshooting issues, refer to the readme file which accompanies the installation media.

For a detailed explanation of the SQL syntax supported by SQLite, please refer the SQLite website: www.sqlite.org

- SQLite does not currently support dynamic creation of SQL stored procedures or functions. The associated methods; $rpcprocedures(), $rpcparameters() & $rpc() therefore return kFalse.
- The SQLite API handles the transfer of binary data automatically. The $blobsize, $lobchunksize and $lobthreshold properties are therefore ignored.
- For performance reasons, journaling mode is set to PERSIST for the SQLite DAM. For optimum performance, especially on Linux it may be desirable to turn off journaling, ("PRAGMA journal_mode = OFF"). **Note:** in this mode however it will not be possible to rollback manual transactions.
- You may experience slow performance during certain INSERT operations. Each INSERT and UPDATE operation is normally committed to the disk drive so as to preserve integrity of the data in the event of a crash or power failure. Executing "PRAGMA synchronous=OFF" tells SQLite not to wait for data to reach the disk surface between writes which results in much faster performance. This risks data loss or corruption in the event of a crash however. Alternatively, you can use manual transaction mode (kSessionTranManual) to commit several INSERT operations at once.

Miscellaneous Enhancements

SQL Programming

Table and Column names

There is a new property of a table instance and session variable called $quotedidentifier which determines whether or not table and column names are contained in quotes. If set to kTrue, table and column name identifiers returned from the $createnames(), $insertnames(), $updatenames(), $selectnames() and $wherenames() methods will be quoted “thus”, facilitating case-sensitive names and names containing spaces. The new property affects table instance methods as well as session object methods (ST/*O/131).

PostgreSQL

There is a new session method in the PostgreSQL DAM (ST/*P/039):

- $escapebinary()
  Returns a text-escaped representation of the supplied binary variable, suitable for use in an SQL statement as a quoted string literal. The returned string does not include the quotes.

ODBC

There is a new session property for the ODBC DAM:

- $infoaserror
  If kTrue (the default), execution results that report SQL_SUCCESS_WITH_INFO are reported as errors. If kFalse, the DAM treats this the same as SQL_SUCCESS and ignores the accompanying message.

Omnis Web and iOS Client

Redraw method

There is a new parameter bExcludeSubForms in the $redraw method for the Omnis Web Client (plug-in) and iOS Client. When set to kTrue (default is kFalse), bExcludeSubForms excludes child controls which are subforms from bSetcontents (note that they cannot be excluded from bRefresh due to the way operating system redraws work).

See earlier in this manual for information about redraw for the new JavaScript Client.

Font Table

The column order of the fonts displayed in the Fonts Table dialog has changed. The first three columns now contain the window font tables. Subsequent columns show the fonts for the JavaScript Client, then the iOS client, and the Windows Mobile client.

Omnis no longer tries to load fonts from a Windows Mobile device when opening the Font Table dialog.

Java

Java Objects

To use Java Objects in Omnis, you must install the latest version of Java SE which is now available from Oracle who acquired Sun Microsystems and the Java technology.

Omnis Java Objects requires Java SE version 5, 6, or above: version 1.4.1 is no longer supported.

Java 7 and environment variables

Developers should be aware of the following when using Java 7 with Java Objects or the Web Services component.

As of Java 7, there is an additional dependency on *MSVCR100.dll*. This is provided as part of the Java installation, but it is not added to any of the default dll search paths.

Therefore, you should add the path to the folder containing MSVCR100.dll to your OMNISJVM environment variable (multiple entries in environment variables should be separated by a semi-colon). This file is installed in the "bin" folder of your Java installation.

After doing this, your OMNISJVM environment variable would look something like:

```
C:\Program Files\Java\jdk1.7.0_01\jre\bin\client\jvm.dll;C:\Program
   Files\Java\jdk1.7.0_01\bin
```

See the documentation for Java Objects (Chapter 6 *Extending Omnis*) and the Web Services PDF for further information about setting the Omnis Environment Variables for using Java Objects and Web Services which depends on Java.

Unicode

Unicode Conversion

Two new sys() functions have been added to assist OEM conversion when using the uniconv() function.

sys(218) modifies OEM conversion to map CR to CR and LF to LF.

sys(219) reverts to the original mapping for the OEM code page.

Using asc()

In the Unicode version of Omnis (Studio 5.2 is Unicode only), the function asc() will return the Unicode value, not the ascii value. Therefore both the following lines of code will return the same result as 8364.

```
Calculate longInt as asc('€',1)
Calculate longInt as unicode('€',1)
```

Localization

String Tables

If you are using String tables to localize your application, the library cannot be called "STRINGTABLE".

HTTP Commands

HTTPPage

You can use a secure connection with the HTTPPage command. If you prefix the URL parameter with https://, the command attempts to use a secure connection. There may be an error on the client if your server is secure and you use http:// or no prefix in the URL parameter.

Omnis Reference

Remote forms

The following sections describe the properties, methods, and children (objects) associated with the new JavaScript Remote forms and components. The current notation for Remote Form instances is still valid for JavaScript Remote Forms with the addition of the properties and methods of the new components for JavaScript based forms.

For those remote form properties, methods, and events listed under **Standard** in the following section, see the Omnis Help (F1) or the *Omnis Notation* manual.

JavaScript Remote Form Instance

Properties

Property	Description
$remotemenu	The current remote menu instance. This is only set when evOpenContextMenu for the field or form is being processed
Standard	$classtype $isprivate $name

Methods

Method	Description
$beginanimations	$beginanimations(iDuration [, iCurve=kJSAnimationCurveEaseInOut]) After calling this, assignments to some properties are animated for iDuration milliseconds by $commitanimations() (JavaScript Client)
$clientcommand	$clientcommand(cCommand, wRow) Executes the command cCommand on the client machine using the parameters in the row variable wRow; possible client commands (cCommand) are: *yesnomessage* Opens a yes-no message box row(message text, title text, yes-servermethod, no-servermethod, cancel-servermethod (or leave empty for no cancel button)) *noyesmessage* Opens a no-yes message box row(message text, title text, yes-servermethod, no-servermethod, cancel-servermethod (or leave empty for no cancel button)) *okcancelmessage* Opens an ok-cancel message box row(message text, title text, ok-servermethod, cancel-servermethod (or leave empty for no cancel button)) *soundbell* Sounds the bell (for desktop browsers, since this is unlikely to work on mobile devices with the JavaScript Client) row(no parameters) *savepreference* Saves a value (character string) as a named preference on the client row(preference name, preference value) *loadpreference* Loads a named preference value (character string) from the client preferences into an instance variable row(preference name, instance variable name (i. e. a quoted string containing the name of the variable)) *javamessage* Shows a message box on the client with up to 3 buttons row(style(error, warning, success, prompt, message, query), text, title text, openatmouse(boolean), butt1text:servermethodname, butt2text:servermethodname, butt3text:servermethodname) *playsound* Plays a sound on the client (for desktop browsers, since is unlikely to work on mobile devices) (first supported file in row() is played, file 1 is also default if there is no HTML5 audio support)

Method	Description
	row(sound file 1 in html sounds folder[, sound file 2, ...]) *setcustomformat* Sets the default custom date format used when $customformat is empty (defaults to D m y) row(date format)
$commitanimations	$commitanimations() animates the relevant property changes that have occurred after the matching call to $beginanimations() (iOS and JavaScript Clients only)
$setcurfield	$setcurfield(vName|Ident|Itemref) sets the current field on the client machine. You can specify the field by name, ident, or item reference. For the iOS client, $setcurfield('') removes the focus from the current field
$showmessage	$showmessage(cMessage[, cTitle]) displays an OK message on the client machine using the specified cMessage and cTitle
$showurl	$showurl(cURL[, cFrame, cWindowProperties]) opens the URL in a new window or frame on the client machine
Standard	$canclose $close

Events

Event	Description
evAnimationsComplete	The animations (started when $commitanimations was last called) have completed **Parameters** pEventCode: The event code
evFormToTop	The remote form is about to become visible on the client **Parameters** pEventCode: The event code pScreenSize: A kSSZ...constant for the current screen size on the client
evScreenOrientationChanged	The orientation of the screen displaying the form has switched between portrait and landscape **Parameters** pEventCode: The event code pScreenSize: A kSSZ...constant for the current screen size on the client
evSubFormToTop	An existing remote form, contained in a subform that has $multipleclasses set to kTrue, is about to become visible on the client **Parameters** pEventCode: The event code
Standard	evExecuteContextMenu evOpenContextMenu

Children

$bobjs	$ivars	$methods	$objs

JavaScript Remote form instance objects ($objs)

This is the group containing all the JavaScript components in the remote form. Note that all the components belong in this group ($objs) including Background and Label.

Methods

Standard	$appendlist $count $findident $findname $first $insertlist $makelist $next $sendall

Children

These are the individual JavaScript controls.

Activity	Background	Button	Checkbox	Combobox
Complexgrid	Datagrid	Datepicker	Droplist	Edit
File	Html	Hyperlink	Label	List
Navigationbar	Pagecontrol	Pagedpane	Picture	Popupmenu
Progress	Radiogroup	Slider	Subform	Switch
Tabcontrol	Timer	Tree	Video	

Activity (JavaScript Remote form instance)

Properties

Property	Description
$activitystyle	The style of the indicator kJSActivityBar kJSActivityBlock kJSActivityCustomLink kJSActivitySmallSpinner
$customlink	The address of the spinner GIF/image
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
Standard	$alpha $componentctrl $componentlib $contextmenu $disablesystemfocus $edgefloat $enabled $events $fieldstyle $height $ident $left $name $objtype $order $screensizefloat $tooltip $top $userinfo $visible $width

Events

Standard	evExecuteContextMenu evOpenContextMenu

Background (JavaScript Remote form instance)

Properties

Property	Description
$::shape	The shape of the background object, a constant: kJSBackEllipse kJSBackHorzline kJSBackImage kJSBackRect kJSBackRoundRect kJSBackTriangle kJSBackVertline
$animation	A string defining an animation, e.g. rotate(90, 4000, elastic, rotate_done). Only supported by browsers which support Raphael
$attr	A string defining an attribute change, e.g. attr(scale, . 5) - reduces object by 50%. Apart from scale, only supported by browsers which support Raphael. See http://raphaeljs.com for more info.
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$imagepath	The path to an image if shape is an image
$strokewidth	The width of the stroke line
Standard	$alpha $backalpha $backcolor $backpattern $bordercolor $componentctrl $componentlib $contextmenu $disablesystemfocus $edgefloat $enabled $events $fieldstyle $forecolor $height $ident $left $name $objtype $order $screensizefloat $tooltip $top $userinfo $visible $width

Events

Event	Description
evAnimComplete	Sent to the control when an animation is complete if the animation had a complete context **Parameters** pEventCode — The event code pAnimContext — The animation context string - as supplied when the animation was set
Standard	evExecuteContextMenu evOpenContextMenu

Button (JavaScript Remote form instance)

Properties

Property	Description
$::vertical	If true, the text and icon are arranged vertically
$buttonbackiconid	The icon id of the button background image. To use the default system button set $buttonbackiconid to zero and $buttoncolor to kColorDefault
$buttonborderradius	The radius in pixels of the button corners
$buttoncolor	The color of the button. To use the default system button set $buttonbackiconid to zero and $buttoncolor to kColorDefault
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$iconid	The numeric icon identifier used to reference the icon in the icon file or #ICONS
$text	The text or calculation stored with the object
$textbeforeicon	If true, and the control has both text and an icon, the text is drawn before the icon
Standard	$align $alpha $bordercolor $componentctrl $componentlib $contextmenu $disablesystemfocus $edgefloat $enabled $events $fieldstyle $font $fontsize $fontstyle $height $ident $left $name $objtype $order $screensizefloat $textcolor $tooltip $top $userinfo $visible $width

Events

Standard	evClick evExecuteContextMenu evOpenContextMenu

Checkbox (JavaScript Remote form instance)

Properties

Property	Description
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$iconid	The numeric icon identifier used to reference the icon in the icon file or #ICONS
$text	The text or calculation stored with the object
Standard	$alpha $backalpha $backcolor $bordercolor $componentctrl $componentlib $contextmenu $dataname $disablesystemfocus $edgefloat $enabled $events $fieldstyle $font $fontsize $fontstyle $height $ident $left $name $objtype $order $screensizefloat $textcolor $tooltip $top $userinfo $visible $width

Events

Standard	evClick evExecuteContextMenu evOpenContextMenu

Combobox (JavaScript Remote form instance)

Properties

Property	Description
$::contenttip	Text which is displayed in the field when it is empty, to help the user understand what content should be entered
$::listheight	Maximum number of lines displayed in the dropped list
$::listname	The name of the list to display combo choices
$defaultline	This will be the line selected when default text is used to initialise the list
$defaulttext	The default text used to initialise the list (a comma-separated list of items). Ignored if a list is assigned to $dataname
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$listcolumn	The column of the list variable that is used to populate the drop list
$selectedlinecolor	The color used to display selected lines. Use kColorDefault for the default color defined in omnis. css
$selectedlinetextcolor	The text color used for selected lines. Use kColorDefault for the default color defined in omnis. css
Standard	$alpha $backalpha $backcolor $bordercolor $componentctrl $componentlib $contextmenu $dataname $disablesystemfocus $edgefloat $effect $enabled $events $fieldstyle $font $fontsize $fontstyle $height $ident $jscustomformat $jsdisplayformat $left $linestyle $name $objtype $order $screensizefloat $textcolor $tooltip $top $userinfo $visible $width

Events

Standard	evAfter evBefore evClick evExecuteContextMenu evOpenContextMenu

Complex grid (JavaScript Remote form instance)

Properties

Property	Description
$headerborder	The border style for the grid header kJSborderEmbossed kJSborderInset kJSborderNone kJSborderPlain kJSborderSingleEmbossed kJSborderSingleInset
$headerfillcolor	The fill color for the grid or headed list box header
$headerheight	The height of the grid header
$headerlinestyle	The line style for the grid header
$horzheaderborder	The border style for the grid horizontal header kJSborderEmbossed kJSborderInset kJSborderNone kJSborderPlain kJSborderSingleEmbossed kJSborderSingleInset
$horzheaderfillcolor	The fill color for the grid horizontal header
$horzheaderheight	The height of the grid horizontal header
$horzheaderlinestyle	The line style for the grid horizontal header
$rowborder	The border style for the grid row section kJSborderEmbossed kJSborderInset kJSborderNone kJSborderPlain kJSborderSingleEmbossed kJSborderSingleInset
$rowdividerlinestyle	The line style for the grid row divider
$rowheight	The height for the grid row
$showheader	If true, the grid has a header
$showhorzheader	If true, the grid has a scrollable horizontal header
Standard	$alpha $backalpha $backcolor $bordercolor $contextmenu $dataname $disablesystemfocus $edgefloat $effect $enabled $fieldstyle $height $ident $left $linestyle $name $objtype $order $screensizefloat $tooltip $top $userinfo $visible $width

Events

<table>
<tr><th>Event</th><th>Description</th></tr>
<tr><td>evExtend</td><td>Sent to grid field when the list is to be extended
Parameters<table><tr><td>pEventCode</td><td>The event code</td></tr><tr><td>pRow</td><td>A reference to the list row</td></tr><tr><td>pLineNumber</td><td>The line number in the list</td></tr></table></td></tr>
<tr><td>evRowChange</td><td>Sent to complex grid field when a field on a different row becomes the target field
Parameters<table><tr><td>pEventCode</td><td>The event code</td></tr><tr><td>pRow</td><td>A reference to the list row</td></tr><tr><td>pLineNumber</td><td>The line number in the list</td></tr></table></td></tr>
<tr><td>Standard</td><td>evAfter evBefore evCanDrop evClick evClipChangedData evDisabled evDoubleClick evDrag evDragFinished evDrop evEnabled evHidden evHScrolled evKey evMouseDouble evMouseDown evMouseEnter evMouseLeave evMouseUp evOpenContextMenu evRMouseDouble evRMouseDown evRMouseUp evSent evShiftTab evShown evTab evVScrolled evWillDrop</td></tr>
</table>

Data grid (JavaScript Remote form instance)

Properties

Property	Description
$::boldheader	If true, the grid header has a bold font
$::candragdisplayorder	If true, the user can drag and drop a column in the header, to change the display order; cannot be changed at runtime
$::columnwidths	A comma separated list of column widths
$::displayorder	Comma separated list of column numbers, indicating the order in which columns are displayed by the data grid. Initially set to 1, 2, ..., $designcols. Reset to the initial value whenever you change the number of columns
$::headerheight	The height of the grid header (when $hasheader is kTrue). If zero, the header height is $rowheight
$::rowheight	The height of a grid row. Zero means the height will be calculated automatically
$autoedit	If true, and the cell is editable, it will automatically go into edit mode when selected
$canresizecolumns	If true, the user can use mouse to resize the columns
$cansortcolumns	If true, the user can click in the column heading to sort a column (see also $columncansort)
$columnbackalpha	The alpha value (0-255) used with $columnbackcolor, when $columnbackcolor is not kColorDefault
$columnbackcolor	The background color of the column. kColorDefault means use the grid background
$columncanresize	If true, and $canresizecolumns is kTrue, the user can resize the column using the mouse
$columncansort	If true, and $cansortcolumns is kTrue, the user can click in the column header to sort the column
$columndatacol	The column number from which to map data
$columnenabled	If true, the column is enabled
$columnfontstyle	The text style of the column
$columnjst	The alignment of the column: kCenterJst kLeftJst kRightJst
$columnminwidth	The minimum width of the column (only relevant when the column can be resized
$columnmode	Specifies how data is handled for the column, a constant: kJSDataGridModeAuto kJSDataGridModeDropList
$columnname	The name of the column

Property	Description
$columnnames	The names of the columns
$columnpicklist	The name of the picklist for the column
$columnpopuptext	If true, and the column uses the editor for character data, use a popup text editor
$columntextcolor	The text color of the column
$columnwidth	The width of the column in pixels
$currentcolumn	The current design column; you need to set this to edit properties for each column
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$enterable	If true, the grid is enterable
$evenrowcolor	The background color of even numbered displayed rows. kColorDefault means use the same color as the odd numbered rows ($backcolor)
$extendable	If true, the grid automatically extends to allow the user to enter more lines
$filterareaheight	The height of the filter area (when $hasfilterarea is kTrue). If zero, the height is calculated automatically
$filtercol	The grid column number to which the filter will apply
$filterlabel	The label for the filter entry field
$filtervalue	The name of an instance variable that contains the value used for filtering
$gridlinescolor	The color of the divider lines in the grid ($gridlinesvisible must be kTrue to display the divider lines)
$gridlinesdotted	If true, the divider lines in the grid are dotted ($gridlinesvisible must be kTrue to display the divider lines)
$gridlinesvisible	If true, the grid divider lines are visible
$hasfilterarea	If true, the grid has a filter area, which can be opened by clicking on a button in the header
$hasheader	If true, the grid has a header. Note that you can only set $hasfilterarea to kTrue if the grid has a header
$hcell	The current grid column
$headertext	The text displayed in the grid header when $hasheader is kTrue

Property	Description
$hideselection	If true, the grid does not highlight selected lines
$movecolumn	Allows you to move a column to the specified position in the grid during design mode
$multipleselect	If true, the field allows the user to select more than one line
$selectedlinecolor	The color used to display selected lines. Use kColorDefault for the default color defined in omnis. css
$selectedlinetextcolor	The text color used for selected lines. Use kColorDefault for the default color defined in omnis. css
$sortascending	If true, the sort indicator displayed when assigning $sortcol will indicate an ascending sort
$sortcol	The grid column number of a sortable column that is to display the sort indicator
$userdefined	If true, the datagrid is developer defined and not automatic
$vcell	The current grid row
Standard	$alpha $backalpha $backcolor $bordercolor $componentctrl $componentlib $contextmenu $dataname $disablesystemfocus $edgefloat $effect $enabled $events $fieldstyle $font $fontsize $fontstyle $height $ident $jscustomformat $jsdisplayformat $left $linestyle $name $objtype $order $screensizefloat $textcolor $tooltip $top $userinfo $visible $width

Events

<table>
<tr><th>Event</th><th>Description</th></tr>
<tr><td>evCellChanged</td><td>Sent to a field after the grid cell has been changed
Parameters<table><tr><td>pEventCode</td><td>The event code</td></tr><tr><td>pHorzCell</td><td>The selected column</td></tr><tr><td>pVertCell</td><td>The selected row</td></tr></table></td></tr>
<tr><td>evColumnsResized</td><td>Sent to the data grid after the user has resized a column
Parameters<table><tr><td>pEventCode</td><td>The event code</td></tr><tr><td>pColumnWidths</td><td>A comma separated list of the new column widths</td></tr></table></td></tr>
<tr><td>evHeadedListDisplay OrderChanged</td><td>Sent to a headed list box when its $displayorder property has been changed when the user drags and drops a column heading
Parameters<table><tr><td>pEventCode</td><td>The event code</td></tr><tr><td>pDisplayOrder</td><td>The order of the columns (a comma separated list of column numbers)</td></tr></table></td></tr>
<tr><td>evHeaderClick</td><td>Sent to a headed list box after the user clicks on a button in the header
Parameters<table><tr><td>pEventCode</td><td>The event code</td></tr><tr><td>pHorzCell</td><td>The selected column</td></tr></table></td></tr>
<tr><td>Standard</td><td>evClick evDoubleClick evExecuteContextMenu evOpenContextMenu</td></tr>
</table>

Date picker (JavaScript Remote form instance)

Properties

Property	Description
$allowchange	If true, the user can change the current date
$currdaycolor	The color used for the current day
$currdaymode	The drawing style for the current day kJSborderDefault kJSborderEmbossed kJSborderInset kJSborderNone kJSborderPlain kJSborderSingleEmbossed kJSborderSingleInset
$currdaytextcolor	The color of the text for the current day
$datebackedgewidth	The width in pixels of the background edge
$datefacealpha	The alpha value (0-255) used with $datefacecolor
$datefacecolor	The color of the picker list face
$datestyle	The date style of the picker, a constant: kJSDatePickerStyleCalendar kJSDatePickerStyleDate kJSDatePickerStyleDateTime kJSDatePickerStyleTime
$datestyleusepicker onmobile	If true, use kJSDatePickerStyleDate instead of kJSDatePickerStyleCalendar on mobile devices
$daycolor	The color of the days in the current month
$dayfont	The font used for the days
$dayfontsize	The font size used for the days
$daymode	The drawing style for the days in the current month kJSborderDefault kJSborderEmbossed kJSborderInset kJSborderNone kJSborderPlain kJSborderSingleEmbossed kJSborderSingleInset
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$firstday	The first day the calendar will show: kFriday kMonday kSaturday kSunday kThursday kTuesday kWednesday
$headingbold	If true, the heading is drawn in bold
$headingcolor	The color used for the heading
$headingfont	The font used for the heading section
$headingfontsize	The font size used for the heading section
$headingmode	The drawing style for the heading

Property	Description
	kJSborderDefault kJSborderEmbossed kJSborderInset kJSborderNone kJSborderPlain kJSborderSingleEmbossed kJSborderSingleInset
$headingtextcolor	The color of the text in the heading
$monthtextcolor	The color of the text of the days in the current month
$otherdaycolor	The color of the days not in this month
$otherdaymode	The drawing style for the previous and next months days kJSborderDefault kJSborderEmbossed kJSborderInset kJSborderNone kJSborderPlain kJSborderSingleEmbossed kJSborderSingleInset
$othertextcolor	The color of the text of the days not in this month
$shortname	If true, the days are drawn using a short name
$showheading	If true, the days of the week are shown
$showmonthnav	If true, the month navigator is shown
$todaybold	If true, today's date is drawn in bold
$todayscolor	The color of today
$todaystextcolor	The color of today's date
Standard	$alpha $backalpha $backcolor $bordercolor $componentctrl $componentlib $contextmenu $dataname $disablesystemfocus $edgefloat $effect $enabled $events $fieldstyle $font $fontsize $fontstyle $height $ident $left $linestyle $name $objtype $order $screensizefloat $textcolor $tooltip $top $userinfo $visible $width

Calendar style picker

The Date Picker control can be switched to display a calendar style date picker, by setting $datestyle to kJSDatePickerStyleCalendar. The following are properties of the Calendar style picker.

Property	Description
$allowchange	If true, the user can change the current date
$currdaycolor	The color used for the current day
$currdaymode	The style for the current day
$currdaytextcolor	The color of the text for the current day
$daycolor	The color of the days in the current month
$dayfont	The font used for the days
$dayfontsize	The font size used for the days

Property	Description
$daymode	The style for the days in the current month
$firstday	The first day the calendar will show
$headingbold	If true, the heading is drawn in bold
$headingcolor	The color of the heading
$headingfont	The font of the heading section
$headingfontsize	The font size of the heading section
$headingmode	The style for the heading
$headingtextcolor	The color of the text in the heading
$monthtextcolor	The color of the text of the days in the current month
$otherdaycolor	The color of the days not in this month
$otherdaymode	The style for the previous and next months days
$othertextcolor	The color of the text of the days not in this month
$shortname	If true, the days are shown using a short name
$showheading	If true, the days of the week are shown
$showmonthnav	If true the current month and year are shown with left and right arrows to change the month
$todaybold	If true, today's date is shown in bold
$todayscolor	The color used for today
$todaystextcolor	The color of today's date

Events

Event	Description
evDateChange	Sent when the current date is changed **Parameters** pEventCode: The event code
evDateDClick	Sent when the user double clicks on a date **Parameters** pEventCode: The event code
Standard	evExecuteContextMenu evOpenContextMenu

Droplist (JavaScript Remote form instance)

Properties

Property	Description
$::listheight	The maximum number of lines displayed in the dropped list
$defaultline	This will be the line selected when default text is used to initialise the list
$defaulttext	The default text used to initialise the list (a comma-separated list of items). Ignored if a list is assigned to $dataname
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$listcolumn	The column of the list variable that is used to populate the drop list
$selectedlinecolor	The color used to display selected lines. Use kColorDefault for the default color defined in omnis. css
$selectedlinetextcolor	The text color used for selected lines. Use kColorDefault for the default color defined in omnis. css
Standard	$alpha $backalpha $backcolor $bordercolor $componentctrl $componentlib $contextmenu $dataname $disablesystemfocus $edgefloat $effect $enabled $events $fieldstyle $font $fontsize $fontstyle $height $ident $jscustomformat $jsdisplayformat $left $linestyle $name $objtype $order $screensizefloat $textcolor $tooltip $top $userinfo $visible $width

Events

Standard	evClick evExecuteContextMenu evOpenContextMenu

Edit (JavaScript Remote form instance)

Properties

Property	Description
$::contenttip	Text which is displayed in the field when it is empty, to help the user understand what content should be entered
$borderradius	Radius for rounded border corners. 1 to 4 pixel values separated by -, in order topleft, topright, bottomright, bottomleft. If bottomleft is omitted the topright value is used, if bottomright is omitted the topleft value is used, if topright is omitted the topleft value is used
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$ispassword	If true, the edit control displays a place-holder for each character, so that the actual value is not visible. Only applies when $issingleline is kTrue
$issingleline	If true, the edit control only allows entry of a single line
Standard	$align $alpha $autoscroll $backalpha $backcolor $bordercolor $componentctrl $componentlib $contextmenu $dataname $disablesystemfocus $edgefloat $effect $enabled $events $fieldstyle $font $fontsize $fontstyle $height $horzscroll $ident $jscustomformat $jsdisplayformat $left $linestyle $name $objtype $order $screensizefloat $textcolor $tooltip $top $userinfo $vertscroll $visible $width

Events

Standard	evAfter evBefore evExecuteContextMenu evOpenContextMenu

File (JavaScript Remote form instance)

Properties

Property	Description
$action	Action for the control, either: kJSFileActionUpload or kJSFileActionDownload
$borderradius	Radius for rounded border corners. 1 to 4 pixel values separated by -, in order topleft, topright, bottomright, bottomleft. If bottomleft is omitted the topright value is used, if bottomright is omitted the topleft value is used, if topright is omitted the topleft value is used
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$downloaderrortext	The error text displayed by the client when it detects a download error. If empty, defaults to: Download error
$hyperlinktext	The text for the hyperlink
$hyperlinkurl	The URL for the hyperlink (this can be for a file download, but it can also be some other URL)
$maxfileuploadsize	The maximum size in bytes of a file to be uploaded to the server. Zero means no limit. Some clients cannot enforce limit
$maxfileuploadsize errortext	The error text displayed by the client when $maxfileuploadsize has been exceeded. If empty, defaults to: File size is larger than maximum allowed upload size
$stopuploadbuttontext	The label of the file upload dialog button to cancel an in-progress upload. If empty, defaults to: Cancel upload
$uploadbuttontext	The label of the file upload dialog button to start the upload. If empty, defaults to: Upload
$uploaderrortext	The error text displayed by the client when an upload error occurs. If empty, defaults to: Upload error
$uploadprogresstext	The progress text displayed during a file upload (when the browser can provide upload progress information). Must contain two %n placeholders. If empty, defaults to: Uploaded %1 of %2 bytes
$uploadstoppedtext	The text displayed by the client when an upload has been cancelled. If empty, defaults to: Upload stopped
$uploadtitle	The title of the file upload dialog. If empty, defaults to: Upload file

Property	Description
Standard	$align $alpha $backalpha $backcolor $bordercolor $componentctrl $componentlib $contextmenu $dataname $disablesystemfocus $edgefloat $effect $enabled $events $fieldstyle $font $fontsize $height $ident $left $linestyle $name $objtype $order $screensizefloat $tooltip $top $userinfo $visible $width

Events

Event	Description
evFileUploadDialogClosed	Sent to the file control when the user closes the file upload dialog **Parameters** pEventCode: The event code
evFileUploaded	Sent to the file control when a file has been successfully uploaded to the server **Parameters** pEventCode: The event code
Standard	evExecuteContextMenu evOpenContextMenu

Html (JavaScript Remote form instance)

Properties

Property	Description
$borderradius	Radius for rounded border corners. 1 to 4 pixel values separated by -, in order topleft, topright, bottomright, bottomleft. If bottomleft is omitted the topright value is used, if bottomright is omitted the topleft value is used, if topright is omitted the topleft value is used
$ctrlname	The name of the control
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$html	The HTML content of the control. This must start with an element declaration such as <div...>. In design mode, you can use %b, %f, %z, %s, %j, %t, %e, %p, %v and %h to insert styles or attributes for the other properties. The design window shows the results
$keepproperties	If false, the control only restores position properties when assigning $html. If true, the control also restores all properties associated with the design-mode % place-holders listed for $html. Default value is true
Standard	$align $alpha $autoscroll $backalpha $backcolor $bordercolor $componentctrl $componentlib $contextmenu $dataname $disablesystemfocus $edgefloat $effect $events $fieldstyle $font $fontsize $fontstyle $height $horzscroll $ident $left $linestyle $name $objtype $order $screensizefloat $textcolor $tooltip $top $userinfo $vertscroll $visible $width

Events

Standard	evAfter evBefore evClick evDoubleClick evExecuteContextMenu evOpenContextMenu

Hyperlink (JavaScript Remote form instance)

Properties

Property	Description
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$isvertical	kTrue if the popup is vertical
$selectedtextcolor	The color of the text when the mouse is over it.
$shouldunderline	kTrue if the links should be underlined.
$text	The text or calculation stored with the object
Standard	$align $alpha $autoscroll $backalpha $backcolor $bordercolor $componentctrl $componentlib $contextmenu $dataname $disablesystemfocus $edgefloat $effect $enabled $events $fieldstyle $font $fontsize $fontstyle $height $horzscroll $ident $jscustomformat $jsdisplayformat $left $linestyle $name $objtype $order $screensizefloat $textcolor $tooltip $top $userinfo $vertscroll $visible $width

Events

Standard	evClick evDoubleClick evExecuteContextMenu evOpenContextMenu

Label (JavaScript Remote form instance)

Properties

Property	Description
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$text	The text or calculation stored with the object
Standard	$align $alpha $backalpha $backcolor $componentctrl $componentlib $contextmenu $disablesystemfocus $edgefloat $events $fieldstyle $font $fontsize $fontstyle $height $ident $left $name $objtype $order $screensizefloat $textcolor $tooltip $top $userinfo $visible $width

Events

Standard	evExecuteContextMenu evOpenContextMenu

List (JavaScript Remote form instance)

Properties

Property	Description
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$evenrowcolor	The background color of even numbered displayed rows. kColorDefault means use the same color as the odd numbered rows ($backcolor)
$iconid	The numeric icon identifier used to reference the icon in the icon file or #ICONS
$ischecklist	If true, each list line has a checkbox which shows if the line is selected, and which also allows the user to select or de-select the line if the control is enabled
$selectedlinecolor	The color used to display selected lines. Use kColorDefault for the default color defined in omnis. css
$selectedlinetextcolor	The text color used for selected lines. Use kColorDefault for the default color defined in omnis. css
Standard	$alpha $autoscroll $backalpha $backcolor $bordercolor $componentctrl $componentlib $contextmenu $dataname $disablesystemfocus $edgefloat $effect $enabled $events $fieldstyle $font $fontsize $fontstyle $height $horzscroll $ident $jscustomformat $jsdisplayformat $left $linestyle $name $objtype $order $screensizefloat $textcolor $tooltip $top $userinfo $vertscroll $visible $width

Events

Standard	evClick evDoubleClick evExecuteContextMenu evOpenContextMenu

Navigation bar (JavaScript Remote form instance)

Properties

Property	Description
$::title	The title for the current navigation bar stack item
$backgroundiconid	If non-zero, the background of the control is filled using this icon after filling with $backcolor and $backalpha
$buttonalpha	The alpha (0-255) to be applied to navigation bar buttons when the client supports alpha
$buttonbordercolor	The border color of the navigation bar buttons
$buttoncolor	The fill color of the navigation bar buttons
$buttonfontsize	The font size for the button text
$buttonheight	The height of buttons on the navigation bar. Zero means calculate the height of each button automatically based on $buttonfontsize and $buttoniconsize, with a minimum value of 26
$buttonhotcolor	The fill color of a navigation bar button when the mouse is over it on a non-touch device
$buttoniconsize	When a button has an icon, it is allocated a rectangle of this width and height
$buttonpressedcolor	The fill color of a navigation bar button when it is being pressed on a non-touch device
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$initiallefticonid	If this is not zero, the first navigation bar item has a button on the left hand side, displaying this icon
$initiallefttext	If this is not empty, the first navigation bar item has a button on the left hand side, displaying this text
$initialrighticonid	If non-zero, the first navigation bar item has a button on the right hand side, displaying this icon
$initialrighttext	If not empty, the first navigation bar item has a button on the right hand side, displaying this text
$initialtitle	The initial title displayed on the navigation bar
$lefthidden	If true, the left hand (back) button is hidden for the current navigation bar stack item

Property	Description
$linkedobject	The name of a paged pane object on the current remote form, used in conjunction with the $push() method. Note that each page of the paged pane can only occur once on the navigation bar stack
$push	Assign a 2-5 column row to $push, to push a new stack item. C1:page number to display in $linkedobject. C2:title for pushed item. C3:Text for right button. C4:Icon URL for right button. C5:Non-zero to hide the left button
$righticonid	The icon id of the icon on the right hand navigation bar button for the current navigation bar item
$righttext	The text on the right hand navigation bar button for the current navigation bar item
Standard	$alpha $backalpha $backcolor $componentctrl $componentlib $contextmenu $disablesystemfocus $edgefloat $events $fieldstyle $font $fontsize $fontstyle $height $ident $left $name $objtype $order $screensizefloat $textcolor $tooltip $top $userinfo $visible $width

Events

Event	Description
evClickInitialLeftButton	Notification of a click on the initial left button **Parameters** pEventCode: The event code
evClickRightButton	Notification of a click on the right button **Parameters** pEventCode: The event code
Standard	evExecuteContextMenu evOpenContextMenu

Page control (JavaScript Remote form instance)

Properties

Property	Description
$::currentpage	The current page number
$::pagecount	The number of pages
$currentpage indicatorcolor	The color of the page indicator for the current page
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$linkedobject	The name of a paged pane object on the current remote form. When the paged pane exists, the page control reflects the current state of the paged pane:its property settings are taken from the paged pane and it does not generate events
$pageindicatorcolor	The color of the page indicators for all pages except the current page
Standard	$alpha $backalpha $backcolor $componentctrl $componentlib $contextmenu $disablesystemfocus $edgefloat $enabled $events $fieldstyle $height $ident $left $name $objtype $order $screensizefloat $tooltip $top $userinfo $visible $width

Events

Event	Description
evPageChanged	Sent to the page control when the page has been changed **Parameters** pEventCode — The event code pValue — The new page
Standard	evExecuteContextMenu evOpenContextMenu

Paged pane (JavaScript Remote form instance)

Properties

Property	Description
$borderradius	Radius for rounded border corners. 1 to 4 pixel values separated by -, in order topleft, topright, bottomright, bottomleft. If bottomleft is omitted the topright value is used, if bottomright is omitted the topleft value is used, if topright is omitted the topleft value is used
$currentpage	The current page. For the window paged pane object, assigning a different value to $currentpage at runtime generates an evTabSelected event
$pagecount	The page count for the control
$scrolltochangepage	If true, the pages are laid out horizontally, and the user can change the current page by scrolling horizontally. When the user changes the current page by scrolling, the paged pane control generates an evUserChangedPage event
Standard	$alpha $backalpha $backcolor $bordercolor $contextmenu $dataname $disablesystemfocus $edgefloat $effect $enabled $events $height $ident $left $linestyle $name $objtype $order $screensizefloat $tooltip $top $userinfo $visible $width

Events

Event	Description
evUserChangedPage	The current page of a paged pane object has been changed by the user, either by scrolling the paged pane (when $scrolltochangepage is kTrue), or by using the linked page control object **Parameters** pEventCode — The event code pPageNumber — The number of the new page selected by the user
Standard	evExecuteContextMenu evOpenContextMenu

Picture (JavaScript Remote form instance)

Properties

Property	Description
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$iconid	The icon ID to reference the icon in an icon file or #ICONS
$keepaspectratio	If true, and $noscale is false, the aspect ratio of the picture is maintained when it is scaled
$mediatype	If you're not using an icon ($iconid is zero or empty), $mediatype specifies the content of the $dataname variable for the picture control: if $mediatype is empty the content is a URL, or if non-empty $mediatype means the content is binary image data of the specified type, e.g. image/png or image/jpg
$noscale	If true, the picture field does not scale the picture
$picturealign	A kPAL... constant which, together with $horzscroll and $vertscroll (and $autoscroll for the JavaScript Client), identifies where the picture will be positioned in the field kPALbottomCenter kPALbottomLeft kPALbottomRight kPALcenter kPALcenterLeft kPALcenterRight kPALtopCenter kPALtopLeft kPALtopRight
$tile	If true, image is tiled
Standard	$alpha $autoscroll $backalpha $backcolor $bordercolor $componentctrl $componentlib $contextmenu $dataname $disablesystemfocus $edgefloat $effect $enabled $events $fieldstyle $height $horzscroll $ident $left $linestyle $name $objtype $order $screensizefloat $tooltip $top $userinfo $vertscroll $visible $width

Events

Standard	evAfter evBefore evClick evExecuteContextMenu evOpenContextMenu

Popup menu (JavaScript Remote form instance)

Properties

Property	Description
$::listname	The name of the list variable used to populate the control. In order to use this, $::menuname must be empty
$::menuname	The name of the remote menu class used to populate the control. Leave this empty in order to use the list data from $::listname
$backiconid	The background icon ID for horizontal tiling of the background
$colcascade	The column name for the menu line cascading menu list data when using $::listname (optional)
$colcommandid	The column name for the menu line command id when using $::listname (optional)
$colenabled	The column name for the menu line enabled state when using $::listname (optional)
$coltext	The column name for the menu line text when using $::listname
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$disbackcolor	The disabled background color
$distextcolor	The disabled text color
$hotbackcolor	The background color for when the control is hot; also used for the current menu line
$hotbackiconid	The background icon ID for when the control is hot (menu is open); also used for the current menu line
$hottextcolor	The text color for when the control is hot; also used for current menu line
$hottitleclicks	If true, and $hotwhenmouseover is true, clicking on the the actual control will generate an evClick event
$hotwhenmouseover	If true, the menu is shown when the mouse is over the control
$iconid	The numeric icon identifier used to reference the icon in the icon file or #ICONS
$menupos	Specifies the popup position of the menu in relation to the control, a constant: kJSPopMenuPosBottom kJSPopMenuPosRight kJSPopMenuPosTop
$text	The text or calculation stored with the object

Property	Description
$textbeforeicon	If true, text comes before the icon ($iconid)
$usehtmlselect	If true, use the HTML <select> tag for the user interface
Standard	$align $alpha $backalpha $backcolor $bordercolor $componentctrl $componentlib $contextmenu $dataname $disablesystemfocus $edgefloat $effect $enabled $events $fieldstyle $font $fontsize $fontstyle $height $ident $left $linestyle $name $objtype $order $screensizefloat $textcolor $tooltip $top $userinfo $visible $width

Events

Standard	evClick evExecuteContextMenu evOpenContextMenu

Progress (JavaScript Remote form instance)

Properties

Property	Description
$::max	The maximum value of the progress bar (a positive integer or zero)
$::value	The current value of the progress bar. Must be between 0 and $max inclusive.
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$progresscolor	The color of the bar representing completed progress. Only applies when the standard HTML5 progress control is not available
$sendcarryon	The progress control does not store a value for this property. Rather, $sendcarryon provides a mechanism to send an evCarryOn event to the progress control. To generate an evCarryOn event, assign kTrue to $sendcarryon
Standard	$alpha $backalpha $backcolor $bordercolor $componentctrl $componentlib $contextmenu $disablesystemfocus $edgefloat $events $fieldstyle $height $ident $left $name $objtype $order $screensizefloat $tooltip $top $userinfo $visible $width

Events

Event	Description
evCarryOn	Sent to the progress control after assigning kTrue to $sendcarryon; the event processing code for evCarryOn can assign $sendcarryon to kTrue again, to generate the next evCarryOn **Parameters** pEventCode: The event code
Standard	evExecuteContextMenu evOpenContextMenu

Radio group (JavaScript Remote form instance)

Properties

Property	Description
$::horizontal	If true, the radio column order is horizontal
$columncount	The number of columns shown for the radio group
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$iconid	The numeric icon identifier used to reference the icon in the icon file or #ICONS
$maxvalue	The maximum value for the radio group
$minvalue	The minimum value for the radio group
$text	The text or calculation stored with the object
Standard	$alpha $backalpha $backcolor $bordercolor $componentctrl $componentlib $contextmenu $dataname $disablesystemfocus $edgefloat $enabled $events $fieldstyle $font $fontsize $fontstyle $height $ident $left $name $objtype $order $screensizefloat $textcolor $tooltip $top $userinfo $visible $width

Events

Standard	evClick evExecuteContextMenu evOpenContextMenu

Slider (JavaScript Remote form instance)

Properties

Property	Description
$::max	The maximum value for the slider
$::min	The minimum value for the slider
$::vertical	If true, the slider is a vertical slider
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$horzmargin	The horizontal drawing margin
$sliderhorziconid	The id of the icon to use for a horizontal slider handle
$sliderverticonid	The id of the icon to use for a vertical slider handle
$step	The size of each step the slider takes between min and max
$val	The current value of the slider
$vertmargin	The vertical drawing margin
Standard	$alpha $backalpha $backcolor $bordercolor $componentctrl $componentlib $contextmenu $disablesystemfocus $edgefloat $effect $enabled $events $fieldstyle $height $ident $left $linestyle $name $objtype $order $screensizefloat $tooltip $top $userinfo $visible $width

Events

<table>
<tr><th>Event</th><th>Description</th></tr>
<tr><td>evEndSlider</td><td>Sent to the slider control when the control is finished tracking
Parameters<table><tr><td>pEventCode</td><td>The event code</td></tr><tr><td>pSliderValue</td><td>The new value of the slider</td></tr></table></td></tr>
<tr><td>evNewValue</td><td>Sent to the slider control when the controls value has changed
Parameters<table><tr><td>pEventCode</td><td>The event code</td></tr><tr><td>pSliderValue</td><td>The new value of the slider</td></tr></table></td></tr>
<tr><td>evStartSlider</td><td>Sent to the slider control when the control is starting to track
Parameters<table><tr><td>pEventCode</td><td>The event code</td></tr><tr><td>pSliderValue</td><td>The new value of the slider</td></tr></table></td></tr>
<tr><td>Standard</td><td>evExecuteContextMenu evOpenContextMenu</td></tr>
</table>

Subform (JavaScript Remote form instance)

Properties

Property	Description
$classname	The class name for the subwindow
$multipleclasses	If true, multiple classes can be open at runtime
$parameters	The constructor parameters for the subwindow
$subinst	The instance contained by the object. If the subform property $multipleclasses is set to kTrue, you must use $subinst(cClassName) to get the appropriate instance
Standard	$alpha $backalpha $backcolor $bordercolor $dataname $disablesystemfocus $edgefloat $effect $enabled $height $ident $left $linestyle $name $objtype $order $screensizefloat $tooltip $top $userinfo $visible $width

Switch (JavaScript Remote form instance)

Properties

Property	Description
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$switchoff	The icon of the image to be used when the switch is off
$switchon	The icon of the image to be used when the switch is on
Standard	$alpha $backalpha $backcolor $bordercolor $componentctrl $componentlib $contextmenu $dataname $disablesystemfocus $edgefloat $enabled $events $fieldstyle $font $fontsize $fontstyle $height $ident $left $name $objtype $order $screensizefloat $textcolor $tooltip $top $userinfo $visible $width

Events

Standard	evClick evExecuteContextMenu evOpenContextMenu

Tab control (JavaScript Remote form instance)

Properties

Note that some of these properties refer to the selected tab ($selectedtab).

Property	Description
$::tabiconid	The icon id for the tab with number $selectedtab
$::tabtooltip	The tooltip for the tab with number $selectedtab
$baseedgealpha	The alpha value (0-255) used with $baseedgecolor
$baseedgecolor	The color of the base edge line. kColorDefault means use black
$baseedgewidth	The width of the line drawn along the base of the tabs. If this is zero, and $linkedobject is present, the tab control updates the paged pane border so that there is the appearance of a gap below the current tab
$currenttab	The current tab. For window objects, assigning a different value to $currenttab at runtime generates an evTabSelected event
$currenttabbackalpha	The alpha value (0-255) used with $currenttabbackcolor
$currenttabbackcolor	The color of the current tab background when $tabbackstyle is kJSTabsBackStyleColor. kColorDefault means use $tabbackcolor
$currenttabbackiconid	The icon id of the current tab background image, used when $tabbackstyle is kJSTabsBackStyleImage
$currenttabextra	The number of extra pixels added to the height (for horizontal tabs), or the width (for vertical tabs), of the current tab
$currenttabtextcolor	The text color for the current tab when the current tab is enabled. kColorDefault means use $textcolor
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$disabledtabtextcolor	The text color for disabled tabs. kColorDefault means use gray
$extratabspacing	Extra spacing added to the calculated width or height of a tab, when $fixedtabsize is kFalse
$fixedtabsize	If true, all icons have a fixed width for horizontal tabs, or a fixed height for vertical tabs. The fixed width or height is the minimum of $maxfixedtabsize and the control width or height divided by $tabcount
$hotcurrent	If true, the current tab displays as hot when it is enabled and the mouse is over it

Property	Description
$hottabbackalpha	The alpha value (0-255) used with $hottabbackcolor
$hottabbackcolor	The color of the tab background when $tabbackstyle is kJSTabsBackStyleColor and the tab is hot (a hot tab is an enabled tab with the mouse over it; current tab hot behavior depends on $hotcurrent). kColorDefault means use $tabbackcolor
$hottabbackiconid	The icon id of the hot tab background image, used when $tabbackstyle is kJSTabsBackStyleImage
$hottabtextcolor	The text color for the hot tab (a hot tab is an enabled non-current tab with the mouse over it). kColorDefault means use $textcolor
$linkedobject	The name of a paged pane object on the current remote form linked to the paged pane – in this case, changing $currenttab also changes $currentpage of the paged pane
$maxfixedtabsize	The maximum fixed width or height of tabs when $fixedtabsize is kTrue
$movetab	Entering a number in this property moves the selected tab so that its tab number is the entered number (design mode)
$selectedtab	The number of the selected tab (the tab which tab-specific properties affect)
$side	How the control is drawn, a constant: kJSTabsSideBottom kJSTabsSideLeft kJSTabsSideRight kJSTabsSideTop
$tabbackalpha	The alpha value (0-255) used with $tabbackcolor
$tabbackcolor	The color of the tab background when $tabbackstyle is kJSTabsBackStyleColor and the tab is neither hot nor current. kColorDefault means use gray
$tabbackiconid	The icon id of the tab background image, used when $tabbackstyle is kJSTabsBackStyleImage
$tabbackstyle	The style of the tab background, a constant: kJSTabsBackStyleColor kJSTabsBackStyleDefault kJSTabsBackStyleImage
$tabborderalpha	The alpha value (0-255) used with $tabbordercolor
$tabbordercolor	The color of tab borders. kColorDefault means use black
$tabborderradius	If greater than zero, the radius of rounded corners for the tab border
$tabborderwidth	The width in pixels of the tab border

Property	Description
$tabcount	The number of tabs
$tabenabled	If true, the tab with number $selectedtab is enabled
$tabiconsize	When a tab has icons, all icons are allocated a rectangle of this width and height
$tablayout	The layout of each tab, a constant: kJSTabsLayoutIconBottom kJSTabsLayoutIconLeft kJSTabsLayoutIconOnly kJSTabsLayoutIconRight kJSTabsLayoutIconTop kJSTabsLayoutTextOnly
$tabmenu	The remote menu for the tab with number $selectedtab. Only used when $trackmenus is true. If assigned at runtime, the menu must already be present on the client (via a $tabmenu or $contextmenu property in the class data when the form was loaded)
$tabsjst	How the tabs are justified within the control. Ignored if scrolling is required kJSTabsJstCenter kJSTabsJstLeftOrTop kJSTabsJstRightOrBottom
$tabspacing	The space in pixels between tabs
$tabtext	The text for the tab with number $selectedtab. You can include a line break by inserting //
$tabvisible	If true, the tab with number $selectedtab is visible
$trackmenus	If true, the tab control displays $tabmenu when the mouse is over the tab. The menu interacts with the tab control using the context menu events. $trackmenus cannot be true for the left and right values of $side
Standard	$align $alpha $backalpha $backcolor $componentctrl $componentlib $contextmenu $disablesystemfocus $edgefloat $enabled $events $fieldstyle $font $fontsize $fontstyle $height $ident $left $name $objtype $order $screensizefloat $textcolor $tooltip $top $userinfo $visible $width

Events

Event	Description
evTabSelected	Send to a tab control when a new tab has been selected **Parameters** pEventCode — The event code pTabNumber — The selected tab
Standard	evExecuteContextMenu evOpenContextMenu

Timer (JavaScript Remote form instance)

Properties

Property	Description
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$running	True if the timer is running
$timervalue	The time between timer events in seconds or milliseconds
$useseconds	True if the timer value is in seconds, otherwise timer is in milliseconds
Standard	$alpha $componentctrl $componentlib $contextmenu $disablesystemfocus $edgefloat $enabled $events $fieldstyle $height $ident $left $name $objtype $order $screensizefloat $tooltip $top $userinfo $visible $width

Events

Event	Description
evTimer	Sent to the control when the timer has expired. **Parameters** pEventCode — The event code
Standard	evExecuteContextMenu evOpenContextMenu

Tree (JavaScript Remote form instance)

Properties

Property	Description
$checkbox	If true, and $multipleselect is also true, the tree control has checkboxes that can be used to select nodes
$datamode	The mode for the list data specified via the data name; a constant: kJSTreeFlatList kJSTreeFlatListOld kJSTreeFlatListOldWithTags kJSTreeFlatListWithTags
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$iconurlprefix	All icons used in the tree (as a result of $showicons being true) must come from a single icon directory. The default is _icons/datafile/omnispic/
$multipleselect	If true, the field allows the user to select more than one line
$selectedlinecolor	The color used to display selected lines. Use kColorDefault for the default color defined in omnis.css
$selectedlinetextcolor	The text color used for selected lines. Use kColorDefault for the default color defined in omnis.css
$showicons	If true, the tree control shows node icons
$showlines	If true, the tree control displays dotted lines connecting nodes
$twostate	If true, and the tree control has checkboxes (see $checkbox), selection of each node is independent
Standard	$alpha $autoscroll $backalpha $backcolor $bordercolor $componentctrl $componentlib $contextmenu $dataname $disablesystemfocus $edgefloat $effect $enabled $events $fieldstyle $font $fontsize $fontstyle $height $horzscroll $ident $left $linestyle $name $objtype $order $screensizefloat $textcolor $tooltip $top $userinfo $vertscroll $visible $width

Events

<table>
<tr><th>Event</th><th>Description</th></tr>
<tr><td>evRenamed</td><td>The user has entered a new name for the node
Parameters
<table>
<tr><td>pEventCode</td><td>The event code</td></tr>
<tr><td>pNodeIdent</td><td>The id of the node</td></tr>
<tr><td>pNodeTag</td><td>The tag of the node</td></tr>
<tr><td>pOldName</td><td>The old node name</td></tr>
<tr><td>pNewName</td><td>The new node name</td></tr>
</table></td></tr>
<tr><td>Standard</td><td>evAfter evBefore evClick evDoubleClick evExecuteContextMenu evOpenContextMenu</td></tr>
</table>

Video (JavaScript Remote form instance)

Properties

Property	Description
$disabledefault contextmenu	If true, the default context menu for the object will not be generated in response to a context click ($clib.$disabledefaultcontextmenu and $cobj.$disabledefaultcontextmenu must both be false for the menu to be generated)
$flowplayerline	If not zero, the line number in the $dataname list of the video content to be used for the fallback Flowplayer object
$flowplayerurl	The URL of the Flowplayer to be used as a fallback when HTML5 video is not available
$showcontrols	If true, the control displays video controls such as play and pause
$YouTube	If true, the control will play a movie from YouTube. In which case, column 1 of row 1 of the dataname list is the YouTube video id
Standard	$alpha $backalpha $backcolor $bordercolor $componentctrl $componentlib $contextmenu $dataname $disablesystemfocus $edgefloat $effect $enabled $events $fieldstyle $height $ident $left $linestyle $name $objtype $order $screensizefloat $tooltip $top $userinfo $visible $width

Events

Standard	evAfter evBefore evExecuteContextMenu evOpenContextMenu