What’s New in Omnis Studio 11.2

Omnis Software
July 2025
75-072025-01

Table of Contents

About This Manual

This document describes the new features and enhancements in **Omnis Studio 11.2 Revision 40173** release.

See the **Readme.txt** file for details of bug fixes in this release and any release notes for Omnis Studio 11.2 Revision 40173.

See the **Install.txt** file to find out System Requirements for running the Development, Runtime, and Server versions of Omnis Studio 11.2.

If you are upgrading from Studio 10.x or before, you may like to read the 'What's New in Omnis Studio 11.1' (Whatsnew111.pdf, published June 2024) for more information about features added in Studio 11.0 and 11.1.

NOTE: *Where a new feature or an enhancement relates to an Enhancement Request or Customer reported fault, the fault reference (e.g. ST/../…) and revision number is included to enable you to track your own ERs and reported faults.*

Software Support, Compatibility and Conversion Issues

The following section contains issues regarding software support, compatibility and conversion in **Omnis Studio 11.2 Revision 40173.**

Serial Numbers and Licensing

You will require a new serial number to run Omnis Studio 11.2, unless you are using Studio Now. If you are on a support program such as ODPP or RMA, you may receive an upgrade serial number. Contact your local sales office to buy a license or obtain an upgrade serial number under your current support program; go to the Contacts page on the Omnis website: www.omnis.net

Library and Datafile Conversion

Converting 10.x Libraries

All Omnis Studio 10.X or earlier libraries need to be converted to run in Omnis Studio 11.X. ONCE A STUDIO 10.0, 10.1 or 10.2 LIBRARY HAS BEEN OPENED WITH OMNIS STUDIO 11.X IT CANNOT BE OPENED WITH STUDIO 10.x – THE CONVERSION PROCESS IS IRREVERSIBLE.

Converting 8.x or earlier Libraries

ALL VERSIONS OF OMNIS STUDIO 11.X WILL CONVERT EXISTING VERSION 8.1.X, 8.0.X, 6.1.X, 6.0.X AND 5.X LIBRARIES – THE CONVERSION PROCESS IS IRREVERSIBLE.

DISCLAIMER: *OMNIS SOFTWARE LTD. DISCLAIMS ANY RESPONSIBILITY FOR, OR LIABILITY RELATED TO, SOFTWARE OBTAINED THROUGH ANY CHANNEL. IN NO EVENT WILL OMNIS SOFTWARE BE LIABLE FOR ANY INDIRECT, PUNITIVE, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES HOWEVER THEY MAY ARISE AND EVEN IF WE HAVE BEEN PREVIOUSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.*

macOS Support

Omnis Studio 11.2 is certified to run on **macOS 15 (Sequoia)** (released in Sept 2024), as well as macOS 14 (Sonoma) and macOS 13 (Ventura). Support for Monterey, Big Sur and Catalina is no longer provided.

When you start Omnis Studio, it will check the version of macOS and will not run if it is older than macOS 13 (Ventura). In this case, Omnis Studio will be marked with a disabled icon.

Windows 32-bit Support

Omnis Studio 11.2 revision 38721 or above is no longer certified to run on **Windows 32-bit** architecture. Therefore, installers for development on Windows 32-bit are no longer provided.

Enter Data Mode

A reported issue with the *Enter Data* command and the window Close box has been resolved in Omnis Studio 11.2 revision 39959. See the Enter Data Mode section below for more details. (Revision 39959, ST/FU/917)

VCS API

The way you access the VCS API has changed in Omnis Studio 11.2 revision 39628. See the VCS API section below for more details. (Revision 39610, ST/VC/837)

NULL values sent to the JS Client

In previous revisions of Omnis Studio, the Omnis server converted NULL values of several data types to Empty or 0 when sending variables to the JavaScript Client. In Omnis Studio 11.2 revision 39291 or above, all NULL values are retained when sent to the client. (Revision 39291, ST/JS/3713)

macOS Tree Restructure

On macOS, the application tree in Omnis Studio 11.2 has been restructured to adhere to the Apple best practice guidelines. See the notes under the ‘macOS Tree Restructure’ section and in the Appendix in this guide for more information. (Revision 38518, ST/IN/281)

Subform Client Commands

The subform set/dialog/palette parameters string now expects a comma as the separator regardless of $prefs.$language. (Revision 38303, ST/JS/3528)

A fault in the subform set/dialog/palette client commands (e.g. $clientcommand('subformdialogshow',…) meant that comma parameter separators were failing when the language in $prefs.$language was set to anything other than English.

This has been fixed and using a comma as a parameter separator in the subform set/dialog/palette client commands now works as expected. Therefore, if you have created a workaround for this issue, your code may not work now and if this is the case it will need to be amended.

What's New in Omnis Studio 11.2

Omnis Studio 11.2 provides new and enhanced **JavaScript Components** and Remote forms, increasing the richness and usability of your web and mobile apps. New **Bookmarks** for methods have been added to make coding in Omnis quicker and easier. And **Support for AI** has been enabled via our unique HTTP Worker object to enable you to add a wealth of dynamic and intelligent features to your applications.

The following is a summary of the enhancements in **Omnis Studio 11.2 Revision 40173:**

❑ JavaScript Components

Subformset Panels: a new JavaScript control to display a number of subforms as vertical, expandible panels, allowing you to create more compact UIs.
Field Border Icons: new properties to add icons to the left or right borders of JS entry fields, adding to the richness of the UI for your web and mobile apps; left icons are also available for Droplists and Combo boxes.
Scroll Shadows: you can now apply a shadow to various lists, grids, and containers to indicate that they have extra scrollable content.
Component Icons: there's a new set of 'feather' icons that you can use in your apps, and the SVG Themer tool allows you to theme a batch of SVG icons, saving you time.
Native List: a new property to add extra height to the end of a Native list, plus buttons can be added to group headings, and button accessories can now take an icon.
Toolbar Control: a new property and event have been added to Toolbars allowing you to add a "back" function to the side menu of a toolbar allowing you to add navigation functions to your web and mobile apps.
Plus there have been enhancements to Data Grids, the Chart control, the Html Link control, Date pickers, the Nav Menu object, Tree lists, and more.

❑ JavaScript Remote Forms

Overriding the Browser History: you can now interact with the web browser's history stack using new form methods, to add navigation events inside your web and mobile apps, overriding the default Back function in the browser.
Customizing Keyboard Shortcuts: you can now intercept keyboard presses in a remote form or on individual controls and modify them with your own event handling methods.
Interacting with the Clipboard: new methods allow you to read the last item of data on the end user's clipboard, or copy some data to the end user's clipboard.
Subform Palettes: new options have been added to hide the arrow on subform palette dialogs, if required, plus you can show a cutout around the target to highlight it.
PDF Printing: new functions in the PDF Device allow you to embed files and data into PDF files.

❑ Debugging Methods

Method Bookmarks: you can now add Bookmarks to methods or to individual method lines which allow you to mark significant places in your code and easily locate them via the new Bookmarks manager window.

❑ Omnis Environment

macOS Tree Restructure: on macOS the application tree has been restructured to adhere to best practice guidelines from Apple, which will improve verification speeds when Omnis first starts up.

❑ Window Components

Border Icons: there is a new event for Entry fields to detect when Border icons have been clicked (also for Token Entry fields, Combo boxes & Drop lists).
OBrowser: there is a new property in OBrowser to toggle the scroll zoom feature, and you can set cefSwitches to the Omnis configuration file.

❑ Report Programming

Enterable Report Fields: there is a new property for report fields allowing end users to enter and save text in the entry fields in a PDF report.

❑ OW3 Worker Objects

Using AI in Omnis: you can now access AI features via the provider's API using the HTTP Worker, including support for Google's Gemini, OpenAI, and Anthropic, including a new sample app in the Hub to demonstrate how you can use AI in Omnis.

❑ Omnis VCS

VCS API: the way you access the VCS API has changed, using the $getapiobject() method, allowing a simpler syntax for VCS API method calls.

❑ Functions

There's a raft of new functions including:
bool() converts a value to a Boolean.
coalesce() and *coalesceempty()* return the first non-NULL or non-Empty parameters from a list of values.
FileOps.$copy() and *$move()* allow you to copy or move files and folders.
FileOps.$joinpath() to join a number of path segments.
idletime() returns the number of milliseconds since the mouse or keyboard was last used.
Omnis PDF Device.$embeddata() and *$embedfile()* allow you to embed files and data into PDF files.
rcedit.$get… functions return information about files (Windows only).

JavaScript Components

Subformset Panels

The Subformset Panel is a new JavaScript control that allows you to display a number of subforms as vertical, expandible panels. (Revision 38468, ST/JS/3467)

The **Subformset Panel** control allows a set of subforms to be displayed as a group of expandible and collapsible panels within its border. Each panel consists of a title bar, with optional expand/collapse and close buttons, and an associated subform arranged vertically within the control.

The end user can expand and collapse each panel to view or hide the content of its subform by clicking on the minimize icon (drop arrow) or by double-clicking the title bar, and panels can be removed by clicking on the close icon. Individual panels can be dragged by the user to change their order on the screen.

Creating a Subformset Panel

To create a Subformset Panel and setup its panels, set the $panelcount property under the Panel tab in the Property Manager to the number of panels you want to include in the control. To setup each panel, set the $currentpanel property to each panel in turn and for each panel enter the name of a subform class (remote form) in the $panelclassname property.

To ensure all the panels stretch to fit the width of the control, you can set $parentwidth to kTrue; this overrides widths set for individual panels in $panelwidth.

You need to set $minbutton to kTrue to add a button to expand and collapse the content of each panel. If you want the end user to be able to remove subforms, you can add a close button to each panel by setting $closebutton to kTrue.

General Properties

The Subformset Panel Control has some properties that are generic to all the panels, including:

Property	Description
$closebuttoniconid	if selected, the default close button icon will be replaced with the selected icon
$font	the font of the title text of each panel title bar
$fontsize	the font size of the title text of each panel title bar. The size of the button icons in the title bar will be proportional to the size of the title text
$minbuttoniconid	if selected, the default minimize / restore button icon will be replaced with the selected icon
$restoreiconid	if selected, and the $minbuttoniconid property is also selected, the minimize icon will be replaced by the restore icon when a panel is collapsed
$textcolor	the text color for the title text of all the panels. This value can be overridden for individual panels by means of the $paneltextcolor property in the Panels tab
$titlebackcolor	the default background color for the title text of all the panels. This value can be overridden for individual panels by means of the $paneltitlecolor property in the Panels tab
$titlebarheight	overrides the default titlebar height of 24 pixels (0 in the Property Manager)
$tooltipsactive	if true, any tooltips which have been defined for individual panels will be displayed when the panel titlebar is hovered

The following properties in the Action tab in the Property Manager determine the behavior of the panels:

Property	Description
$closebutton	if true a close button is shown on the panel title. If the user clicks the close button the panel and subform are removed.
$esctoclose	if true the panels in the set can be closed by pressing the Escape key.
$minbutton	if true a minimize button is shown on the panel title. The user can flip the expanded / collapsed status of the panel by clicking on the button. Alternatively, the user can double-click on the titlebar to achieve the same functionality.
$minbuttonistitle	if true the minimize button icon is removed and the whole title bar becomes the minimize button.
$openmax	if true $panelheight is ignored and open panels expand to the full height of the control. The title bars of any collapsed panels will not show until the open panel has been collapsed.
$openmin	if true the subform panels in the set are initially opened in the minimized state.
$parentwidth	if true the individual $panelwidth for each panel is overridden, and its width is set to fill the width of the control.

Property	Description
$preventdrag	if true the user will not be able to drag the panels to re-order them.
$scrollable	if true allows subforms to scroll. Only affects non-responsive subforms as responsive subforms are scrollable by default.
$singleopen	if true the panels are initially displayed with the first panel only expanded ($openmin must be set to false).

Panel Properties

The 'Panel' tab in the Property Manager contains properties that apply to the panel specified in $currentpanel. First set $panelcount to specify the number of panels in the control.

Property	Description
$currentpanel	panel specific properties are assigned to the current panel; set this in design mode to assign panel properties
$movepanel	allows you to move a panel in design mode; the current panel moves to the position specified by the number entered and the panels after this are bumped down
$panelclassname	the classname of the subform (remote form)
$panelcount	the number of panels in the control
$panelheight	the height of the subform panel when it is in the expanded state
$panelid	an optional unique ID character string used to identify the panel
$panelsubformparams	an optional comma-separated list of parameters that can be passed to a subform
$paneltitle	the title text of the panel
$paneltitletooltip	an optional character string displayed when the title bar is hovered. The $tooltipsactive property must be set to true
$panelwidth	the width of the subform panel if $parentwidth is set to false

Colors and Icons

The colors of the panel title text and panel title background are determined by the general **$textcolor** and **$titlebackcolor** properties. To assign colors to individual panels, the colors can be overridden by the specific color properties **$paneltextcolor** and **$paneltitlecolor** in the 'Panel' tab.

The default minimize and close button icons can be overridden by selecting an icon in the $minbuttoniconid and $closebuttoniconid properties.

Passing Parameters

The Subformset Panel Control has the facility to pass a set of parameters to its subforms. The $panelsubformparams property is a comma-separated list of parameters that can be passed to each subform.

Events

The Subformset Panel control reports the following events. The **evPanelOpened** event reports the panel that has been expanded from the collapsed state, while the **evPanelCollapsed** event reports the panel that has been collapsed from the open state. The **evPanelRemoved** event reports the panel that has been removed from the set.

For each of these events the pPanelID parameter returns the unique ID of the panel, and the pPanelPosition parameter returns the position of the panel in the set.

Methods

The Subformset Panel control has the following methods:

- **$addpanel**(cPanelID, cPaneltitle, cPanelclassname, cPanelsubformparams, iPaneltextcolor, iPaneltitlecolor)
  adds a new panel at the end of the subform set.
- **$removepanel**(cPanelID)
  removes the panel denoted by cPanelID.
- **$openpanel**(cPanelID)
  opens the panel denoted by cPanelID.
- **$collapsepanel**(cPanelID)
  collapses the panel denoted by cPanelID.

The methods that require cPanelID use the IDs assigned to each panel in $panelid.

In addition, the Subformset Panel control has the $showpanel() method which can be used when the control is enabled as a Side Panel ($sidepanel is kTrue).

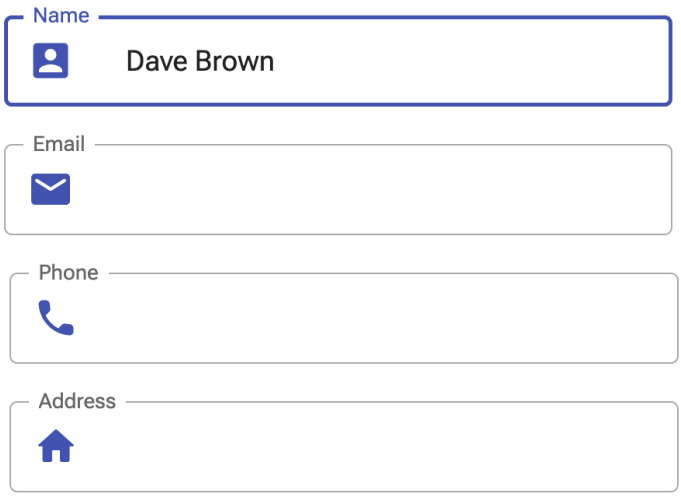
Field Border Icons

The $bordericonstyle property, the $setbordericonstyle() method, and the evBorderIconClicked event have been added to JS Entry Fields to allow you to add icons to the left or right borders of the fields, as well as the left side of Combo boxes and Drop lists. (Revision 39977, ST/JS/3766)

Aria accessibility labels have been added to Field Border Icons, as specified in the $bordericonstyle property, plus the $setbordericonstyle() method has a new optional parameter cAriaLabel to specify the aria label for the icons. (Revision 40101, ST/JS/3776)

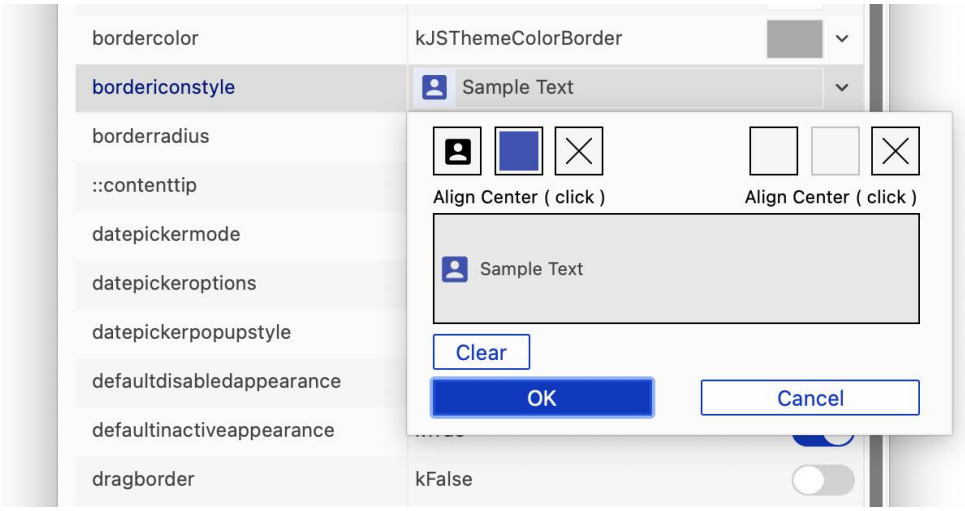
The pEvAfterFired parameter has been added to evBorderIconClicked. It is kTrue if clicking the border icon triggered evAfter for the field. (Revision 40117, ST/JS/3778)

The **$bordericonstyle** property, the **$setbordericonstyle()** method, and the **evBorderIconClicked** event have been added to JavaScript Entry Fields to allow you to add icons to the left or right borders of fields. This enhancement also applies to Droplists and Combo Boxes, where icons can be added to the left side of the field only, due to the drop arrow icon on the right. Border icons are available in existing revisions for Window class Entry Fields and the implementation for JS Entry fields is largely the same.



Field border icons can be added to Entry fields using the $bordericonstyle property, which only applies when the field border style in $effect is set to kJSborderPlain. The icon size is larger for Entry fields in the JavaScript client, that is, 20x20 pixels compared to 16x16 in Window classes.

The single property $bordericonstyle stores the settings for the left and right icons, which you can specify in the Property Manager.



When setting $bordericonstyle you can also set the Aria accessibility label for the right and left icons.

The **$setbordericonstyle()** method allows you to set the left or right icon for a JS Entry Field as defined in the $bordericonstyle property, and has the following syntax:

- **$setbordericonstyle()**
  $setbordericonstyle(bLeftIcon [,cIcnIDname, iIcnTintColor, iBackTintColor, iAlign, cAriaLabel]) sets $bordericonstyle of the left or right icon using the following parameters:
  bLeftIcon set to kTrue for the left icon, kFalse for the right icon
  cIcnIDname is the name or ID of the icon, e.g. the name of an SVG icon or a path using the iconurl() function
  iIcnTintColor is the color of the icon, e.g. primary theme color
  iBackTintColor is the background color, e.g. the default is the background color of the field
  iAlign is the *vertical* alignment of the icon within the field, a constant, either kJsVertCenter (the default), kJsVertBottom, or kJsVertTop
  cAriaLabel is the aria accessibility label used if the icon is clickable

You can enable the **evBorderIconClicked** event (in $events for the control) to make the border icons active or clickable. If evBorderIconClicked is enabled, border icons can be clicked and tabbed to, otherwise they are inactive and for display only. The event has the parameter pLeftBorderIcon. If true, the left border icon was clicked, or if false, the right border icon was clicked. The pEvAfterFired parameter is kTrue if clicking the border icon triggered evAfter for the field.

Scroll Shadows

You can now apply a shadow to various lists, grids, and containers to indicate that they have scrollable content – the shadow will appear on the relevant edge of the control to show that there is more content that can be scrolled. (Revision 39756, ST/JS/3714, ST/JS/3741 for the first raft of JS controls, and Revision 40151, ST/JS/3788 scroll shadow added to the Subform control)

Scroll shadows are fading shadows that can be added to various JavaScript components to indicate when part of the content is hidden and can be revealed by scrolling. There are two scroll shadow properties:

$scrollshadowwidth indicates the width (depth) of the shadows in pixels. A zero value (the default) means there are no scroll shadows.

$scrollshadowcolor is a color value including alpha (opacity), selected using the alpha color picker.

The following JavaScript components support scroll shadows:

List Control	Native List	Data Grid
Complex Grid	Droplist	Combo Box
Edit Control	Picture	Paged Pane
Scroll Box	Subform Control	Tile Grid
Tree List		

Component Icons

Feather Icon Set

A new set of SVG icons called **'feather'** has been added to Omnis Studio. (Revision 39034, ST/JS/2798)

A new set of SVG icons called 'feather' has been added to Omnis which can be used for JavaScript components as an alternative to the existing 'material' icon set. To use the feather icon set, you need to add the name 'feather' to the $iconsets library property (via the Property Manager), after which the feather icon set will be available in the **Select an Icon** dialog alongside the existing icon sets including material.

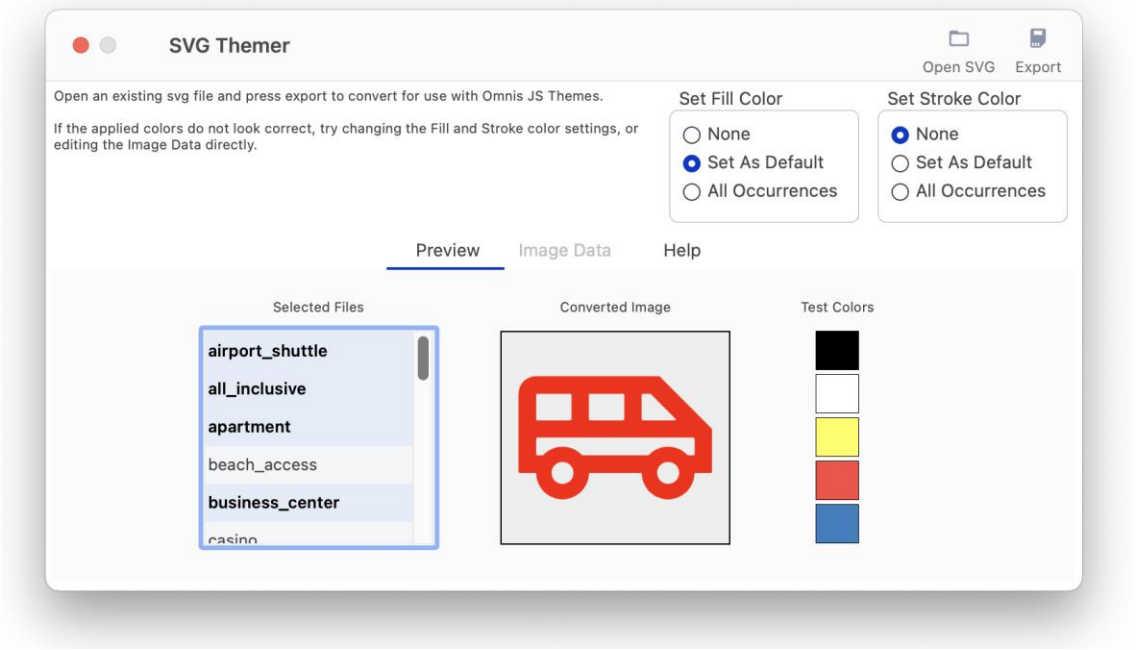
The feather icon set contains over 200 SVG icons which are suitable for a broad range of business use cases. They have been 'themed' using the SVG Themer tool (available in the Tools>>Add Ons menu) and therefore support JS Themes (see also batch theming below).

The icons in the feather icon set are issued under the MIT License, so you are free to use them in your Omnis applications but with the proper attribution in your product licensing.

SVG Themer Tool

You can now theme a batch of SVG icons using the SVG Themer tool (available in the Tools>>Add Ons menu), making it easier and quicker to add a new icon set for use with JS Themes; in previous revisions you could only theme one icon at a time. (Revision 39111, ST/AD/210)

You can theme and export a batch of SVG icons using the SVG Themer tool. In this case, you can open a number of SVG image files, make a selection from the list or select all the files (using Ctrl/Cmnd+A), adjust the Set Fill Color and Set Stroke Color options as required, and click **Export** to theme and export all the files.



IMPORTANT: note that when exporting a batch, any files in the destination folder with the same name as the exported files will be automatically overwritten ***without a prompt to overwrite the files,*** so you may want to place the exported themed files in a separate folder if you wish to retain the original SVG image files.

If you are creating a new icon set, the icons need to be placed in a folder in the 'iconsets' folder in the Omnis tree, and the folder name of the icon set needs to be added to the $iconsets library property. You can then select the icon for a component by setting its $iconid property in the **Select an icon** dialog.

Icon Naming

When specifying $iconid for SVG icons, the *icon set name* is now added to the icon name in the format setname.iconname. (Revision 39060, ST/HE/2073)

When you select an icon from the **Select an Icon** dialog the icon set name is now added to the icon name using the format iconsetname.iconname, where iconsetname is the name of the icon set containing the icon. This avoids any conflict where different icons with the same name may exist in different icon sets. For example, to reference an icon called 'archive' in the 'material' icon set, the full name material.archive is used and Omnis will identify the icon in the correct icon set. This format of naming icons does not apply when using the old PNG icons with numeric icon IDs.

Note when using the new naming, you can still append a custom size to the icon name in the format setname.iconname+WxH when specifying $iconid in the Property Manager or in code, e.g. material.account_box+60x60 to display the icon at 60x60 pixels.

Native List

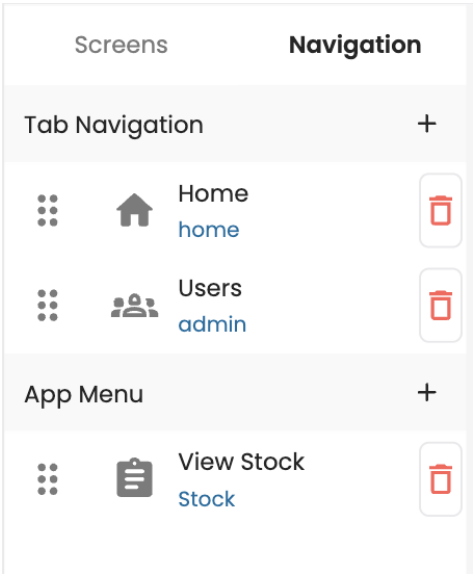
Extra Scroll Height

The **$extrascrollheight** property has been added to the JS Native List. (Revision 39920, ST/JS/3760)

The $extrascrollheight property specifies the extra scroll height added to the end of a Native list. This could be useful to allow space for a Floating Action Button to be displayed over the top of a Native list, for example.

Native List Buttons

You can now add buttons to the group headings in Native Lists, plus button accessories can now take an icon (or image) URL as their content. (Revision 39773, ST/JS/3740 & ST/JS/3748) The following example shows the use of icons buttons and icons in a Native List:



Native Lists have two new properties: **$groupaccessorytypecol** and **$groupaccessorycontentcol**, which can be set to column numbers in your main (grouped) list which contain values for the group heading's accessory type, and its content, respectively (these cannot have the value 1 or 2, as they are reserved for the group name and item list).

You can set $groupaccessorytypecol to kJSNativeListAccessoryTypeButton to add a button to the group heading (no other accessory type is currently supported).

$groupaccessorycontentcol can be set to text or an icon/image URL, the image being detected by the URL ending with a .png/.jpg/.jpeg/.svg extension.

If you are not using a URL generated using the iconurl() function, you can specify a size to render the image at by including a URL parameter specifying this with the format _<w>x<h>.

E.g. https://omnis.net/developers/resources/favicon.png?_20x20

You can specify a separate tint color for themed SVG icons by including a "color" URL parameter, e.g. con(iconurl("add+24x24"),"&color=red"). Otherwise, the icons will use the color resolved as the $buttontextcolor.

When a group heading accessory is clicked, evClick will be fired. The **pWhat** parameter will be kJSNativeListPartGroupAccessory, and **pGroup** will be the group number.

Toolbar Control

The $backbutton property and evBackClick event have been added to the JS Toolbar control to allow you to add a "back" function to the side menu in a JS Toolbar control. (Revision 39791, ST/JS/3753)

You can now add a 'back' button to the left side of a JS Toolbar control by setting **$backbutton** to kTrue. In this case, if the side menu is visible ($sidemenu is kTrue), the hamburger menu is replaced by the back button. When the back button is visible and is clicked, the new **evBackClick** event is triggered.

You could use this property and event to add a "back" function to the side menu in a toolbar. For example, you could use an option in the side menu to navigate to a

different form, change the hamburger menu to show the back button by setting $backbutton, then when this is clicked return to the original form and hide the back button.

Data Grid

Filtering the Grid

The **$filteronkeypress** property has been added to the Data Grid to control when filters are applied to the grid data – in previous revisions, the filtering took place on each key press. (Revision 39953, ST/JS/3618)

If $filteronkeypress is set to true, the grid is filtered on every keypress (the default, to maintain behavior from previous revisions). If false, filtering takes place when the Enter key is pressed or when the focus leaves the field.

Disabling Grid Cells

The **$enablecell** method has been added to the Data Grid control to prevent specific cells from being edited. (Revision 39803, ST/JS/3695)

The $enablecell(pRow,pCol,pDataColumnName) method is called before a cell is made editable, so return kFalse to disable editing for the cell. This method can be client or server executed.

Grid Cell Tooltips

You can now display tooltips for individual cells in a Data Grid. (Revision 39004, ST/JS/3545)

The **$gettooltip()** client-executed method is called when a grid cell is hovered. The returned string is displayed as the tooltip for the cell. The method has the following parameters pHorzCell, pVertCell, pDataColumnName to identify which cell has been hovered over.

Pick Lists

The **$picklistrowheight** property has been added to the Data grid control to allow you to set the row height in pick lists. You could set this to the same value as the row height for the data grid, e.g. 44 which is the default grid row height. The pick list row height will not be less than the font height. (Revision 39006, ST/JS/3492)

Date Picker

If either of the Data Grid properties **$editdatetext** or **$columneditdatetext** is true, the Date picker button is no longer treated as a separate tab stop. You can open the Date picker using Alt + Down arrow, or by clicking on it. (Revision 39082, ST/JS/3685)

If you want the old behavior, where the Date picker is a separate tab stop, you can set the JavaScript variable 'alwaysTabToDatePicker' to true on the data grid. For example:

```
Calculate lDataGrid as $cinst.$objs.datagrid1
JavaScript:lDataGrid.alwaysTabToDatePicker = true;
```

Drag and Drop

The evDrop, evCanDrop and evWillDrop events have a new parameter pColNum to report the column number for the dropped data. (Revision 38661, ST/JS/1345)

The Data Grid control has a new parameter, **pColNum,** representing the column number of the cell in which the data will be dropped. The parameter is available for the evDrop, evCanDrop and evWillDrop events.

JS Chart Control

The $xlabelpadding and $ylabelpadding properties have been added to the JS Chart control to improve the display of the X and Y axis labels on the chart. (Revision 40131, ST/JS/3781 and Revision 40138, ST/JS/3785)

The **$xlabelpadding** and **$ylabelpadding** properties allow you to specify the amount of extra space between labels on the X or Y axes, to allow you to improve the display of chart labels when you have a lot of data points and the labels would otherwise be very cramped. The properties are specified as a number of pixels, but the number is used in conjunction with the default label spacing to arrive at a "best fit" arrangement of the labels.

As labels must be associated with a discrete data point, adding padding by setting $xlabelpadding and $ylabelpadding may result in an uneven distribution of labels, for example, labels may be placed at positions "0, 3, 6, 10, 13", and so on.

In most cases the X axis will always show the first and last label, so the full extent of the range of data points can be seen. However, when using a large value for setting the X axis label padding, the last label may not be shown, so as a special case you can set $xlabelpadding to > 9999 to force only the first and last label to be shown.

Html Link Control

Link URL

When the link URL properties in the HTML Link Control are empty, the URL will now default to the link text. (Revision 39433, ST/JS/3705)

If the $linkurl, $linkurlname, or $linkedobject properties are empty, the URL for the Html Link control defaults to the text from $text or $dataname, provided the text has a valid URL format (http://... or https://...).

Events

The default event behavior for **evClick** in the Html Link control can now be overridden in the $event method using the *Quit event handler (Discard event)* command. (Revision 39413, ST/JS/3717)

Date Pickers

Date picker popups now use the client locale's default for the first day of the week. (Revision 39601, ST/JS/3733)

Date pickers for Entry fields and Data Grids (not the Date picker control) will now use the standard first day of the week from the client browser's locale when in the calendar view. In previous revisions the first day of the week defaulted to Sunday.

Navigation Menu Object

Two new properties $::iconcolor and $cascadeiconcolor have been added to the Navigation Menu Control to allow you set the color of the SVG icons in the control. This enhancement applies to the JavaScript (Remote form) and Window class Nav Menu controls. (Revision 39578, ST/JS/3720)

The **$::iconcolor** property specifies the color of the SVG icon on the top level menu specified in $verticalcascadeiconid.

The **$cascadeiconcolor** property specifies the color of the SVG icons on the cascaded menus, specified in $horizontalcascadeiconid, $closeboxiconid and $hotcloseboxiconid.

In the JS client component, kColorDefault conforms to the default text color rules, while in the Window class control kColorDefault causes the icon colors to follow the text color.

Tree Lists

You can now set the size of the icons in a Tree List using "iconID+k32x32" in the list definition. (Revision 38825, ST/JS/2095)

Droplists

The **$selectonopen** property is now set to kFalse by default in all new Droplists (when added from the Component Store), which means the first line in the droplist is not selected when the form is opened and evClick is not generated. $selectonopen was set to kTrue by default in previous revisions. (Revision 38830, NC/JS/3625)

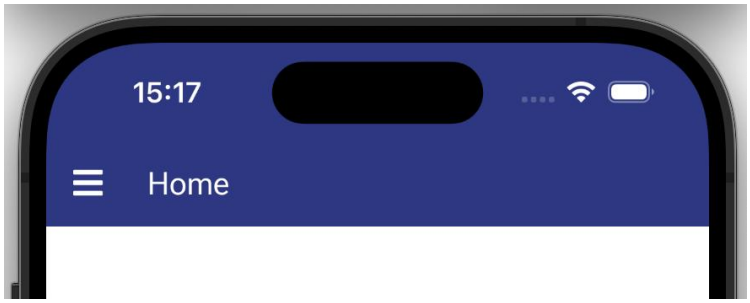
Tab Bar Control

The $currenttab property of the Tab Bar control can now be set to 0, which has the effect of deselecting all tabs. (Revision 39794, ST/JS/3750)

JS Themes

The kJSThemeColorWindowChrome theme color has been added, which can be used to color the window surrounding the JS client when displayed in a browser on a mobile device. (Revision 40152, ST/JS/3789)

The **kJSThemeColorWindowChrome** theme color is used to color the window surround, including the status bar, the 'unsafe area' (around the notch etc), and the address bar if the JS client is displayed in a browser app on a mobile device.



This color is unique in the JS Theme Editor, in that you can clear its value to set it to the Default color, that is, no color will be applied to the window surround.

Note that on Android, this does not take effect if you are using a dark theme on your device.

Component Names

It is possible to assign a numeric name to $name for a component in your code using the notation, but this is not recommended. If you do this, a warning will be added to the trace log, and in this case, you should change your code to not assign numeric names. (Revision 39215, ST/NT/821)

JavaScript component names (and other class components) cannot start with a number and cannot contain only numbers, so when naming a component in the Property Manager (i.e. changing $name) Omnis will not allow numeric names.

JavaScript Remote Forms

Overriding the Browser History

You can now interact with the web browser's history stack using various new form methods, for example, to add navigation events inside your remote form which are triggered when the user clicks the **Back** or **Forward** browser buttons. (Revision 39275, ST/JS/2121)

There is a new sample app called **JS History** in the **Samples** section of the **Hub** in the Studio Browser which shows how you can use the browser history to track clicks in a Tab bar on a remote form.

The History stack

The **History stack** is a list of pages or states that a user navigates through within a browser session. History states can be added to the session history stack or replaced but cannot be removed. Rolling back history therefore involves moving the user back through the history stack, so forward navigation will still be possible after a rollback. Pushing a new state onto the stack does, however, invalidate any states that are ahead of the user's current position in the stack. Note that session history persists between page reloads.

Interacting with the History

The browser history buttons can be overridden by adding a new state to the history stack using the **$pushhistorystate()** remote form method, which can be client- or server-executed. When the user uses the browser **Back** or **Forward** buttons to navigate through the history, the **$applynewhistory** callback method will be called.

- **$pushhistorystate**(*wRow*, [*bReplaceState*=kFalse])
  wRow is a row and should contain the information needed for you to restore your application to this state. This could be as simple as the current tab number of a tab control, as shown in the sample app. You can replace the current history state instead of adding it to the history stack by specifying *bReplaceState* as kTrue.
- **$applynewhistory()**
  receives the row *wRow* from $pushhistorystate as a parameter. Custom code should be added to this form method to determine the behavior to be applied on browser navigation. For example, the tab number could be assigned to the $currenttab property of a tab control, as in the sample app. Return kTrue to indicate you have handled the history change. This will prevent $applynewhistory being called on any other forms for this history navigation event.

Note that $applynewhistory will be called on all open forms until one returns kTrue, and that the order in which forms are called is not guaranteed.

In the sample app, when the end user clicks on a Tab bar the tab number is added to the browser history using $pushhistorystate(), so when the user clicks on the **Back** button in the browser, Omnis cycles back through the tabs in the correct order according to the browser history.

Rolling Back the History

The **$rollbackhistory** form method can be used to move the user back through the custom history states that have been added using $pushhistorystate(). $rollbackhistory can be client- or server-executed.

- **$rollbackhistory**([*iCount*=0, *bPreventNotify*=kFalse])
  iCount is the number of custom states to go back. The default value is zero, which takes the user back to the beginning of custom history. If *bPreventNotify* is specified as kTrue, $applynewhistory will not be called at the end of the rollback.

In the sample app, a home button uses $rollbackhistory without a parameter to jump back to the beginning of the history stack, in effect going back to the initial tab.

Customizing Keyboard Shortcuts

You can now intercept keyboard presses in a remote form or on individual controls and modify them with your own event handling methods. (Revision 39937, ST/JS/3008)

The new **$keyboardshortcut** client-executed method allows keyboard events to be intercepted either in the remote form or on a JS control, which allows you to add custom behavior to a keyboard event where one or more of the modifier keys are pressed. The $keyboardshortcut method can be called on a remote form instance by overriding the default form method or can be added to a JavaScript control.

The $keyboardshortcut method has the following parameters:

- **pSystemKey**
  If the pressed key is a system key, the value is the keyboard constant value representing the pressed key, such as kBack. If the pressed key is not a system key the value is 0.
- **pCharacterKey**
  If the key is a character key the value is the uppercase version of the character pressed. If the pressed key is not a character the value is an empty string.
- **pModifiers**
  Is the sum of the following keyboard modifier constants, representing the state of these modifiers at the point the system or character key was pressed: kJSModShift, kJSModCtrl, kJSModAltOrOption, kJSModCmd.
  These values produce a "bitmask" where each value in the sum sets a different binary bit in the number. You can use the binary *bitand()* function to see whether the kJSMod... values' bits are set. For example, the following code would determine if the control key is pressed, where lControlKey is a Boolean variable:

```
Calculate lControlKey as bitand(pModifiers,kJSModCtrl)<>0
```

- **pFocusedControl**
  Is a reference to the current active control, or null if no control is focused. It cannot be assigned to an instance variable, but its properties, such as pFocusedControl.$name, can be assigned to an instance or local variable

The $keyboardshortcut method should return kTrue to specify that you have handled the keyboard shortcut, and do not want the default behavior to occur. For example, the following code will result in the default keyboard shortcut behavior for ctrl + “S” being overridden and $somemethod() being called instead:

```
If pCharacterKey="S"&lControlKey=kTrue
   Do $cinst.$somemethod()
   Quit method kTrue
End If
```

An attempt is first made to call $keyboardshortcut on the current active control (or top form if no control is active), and successively up the stack of container controls and forms until a control / form is encountered whose $keyboardshortcut method (if it exists) returns kTrue.

Web Browser or Operating System shortcuts using the same key combinations may take precedence. Therefore, you are advised to test your methods thoroughly to ensure your custom shortcuts are not already used by the OS or browser.

Interacting with the Clipboard

Reading Data from the Clipboard

The $readclipboard() and $onclipboardread() methods have been added to remote forms to allow you to read the last item of data on the end user's clipboard. (Revision 39979, ST/JS/3767)

The **$readclipboard()** remote form method allows you to read the last item of data on the end user's clipboard, which in turn calls the **$onclipboardread()** remote form method, passing it three parameters:

- **pSuccess**
  A boolean value indicating whether the clipboard read has been successful
- **pPlainText**
  A character value. If pSuccess is true, pPlainText is the value of the 'text/plain' MIME type of the clipboard content. If there is no plain text data on the clipboard, pPlainText will be an empty string. If pSuccess is false, pPlainText is the associated error message
- **pExtraData**
  A row containing any other MIME types associated with the data (for example 'text/html', 'image/png'). The column headings are the MIME types, with the values being the data value corresponding to the MIME type. Html text data is a Character type, while images are represented as Binary data

Both $readclipboard() and $onclipboardread() can be either client- or server-executed.

The following example code in $onclipboardread() would assign the value of some clipboard plain text to iText if the read is successful, or send an error message to the trace log if not:

```
If pSuccess=kTrue
   Calculate iText as pPlainText
Else
   Send to trace log [con("error: ",pPlainText)]
End If
```

As another example, if the clipboard item is an image, the binary string associated with the image can be assigned to iImageData in the following code:

```
Calculate iImageData as pExtraData.["image/png"]
```

Html text may be sanitized depending on the browser to prevent malicious content from being pasted into the document.

For security reasons, $readclipboard() will only work if called within a secure context and from a user interaction such as a button click. In Chrome, the browser's clipboard permissions may need to be changed to 'Allow' or 'Ask'. In Safari and Firefox, when the method is called a Paste option is displayed, which will need to be clicked by the end user to proceed with the clipboard read.

Copying Data to the Clipboard

The **$copytoclipboard()** method has been added to remote forms to allow you to copy some data to the end user's clipboard. (Revision 39448, ST/JS/3330)

The $copytoclipboard() form method has the following parameters:

- **$copytoclipboard**(vDataToCopy[,cMIMEType='text/plain',bDataIsBinary=kFalse])
  copies vDataToCopy to the user's clipboard, either character or binary depending on the value of bDataIsBinary; cMIMEType is the MIME type of the data which defaults to "text/plain"

If Chrome is being used, the end user can change the permission level for the clipboard in the privacy and security settings. The default is "Ask", but the user could also change the permission level to "Allow" or "Block". However, whatever the permission level that the user has set, the $copytoclipboard method will work without

interruption if it has been initiated by the user, for example by clicking a button – "Ask" or "Block" will only be active if the $copytoclipboard method has been called other than by user interaction. Permissions do not apply in Firefox or Safari.

For example, a character variable iText holds the text entered into an Edit Control. When a button is clicked, the text is copied to the user's clipboard as text. The code behind the button would be:

```
On evClick
   Do $cinst.$copytoclipboard(iText)
```

If the text held by iText is html, which needs to be rendered as rich text, the code behind the button would be:

```
On evClick
   Do $cinst.$copytoclipboard(iText,"text/html")
```

In the following example, a binary variable iImageData holds the binary data of an image. When a button is clicked the image data in iImageData is copied to the clipboard. The code behind the button would be:

```
On evClick
   Do $cinst.$copytoclipboard(iImageData,"image/png",kTrue)
```

Subform Palettes

Palette Arrow

The **kSFSPaletteHideArrow** flag has been added to the subform palette options to hide the arrow if required. (Revision 38681, ST/JS/3571)

When using the "subformpaletteshow" client command to open a subform palette, you can use the kSFSPaletteHideArrow flag in the iPositionFlags row parameter to hide the arrow normally shown on the palette window.

Show Overlay

A cutout has been added around the target for palette dialogs when overlay is shown. (Revision 39808, ST/JS/3757)

If you pass the **bShowOverlay** parameter as kTrue to the "subformpaletteshow" client command ($clientcommand), then if it points at a control (or a sub-element of the control), a cutout will be made in the overlay around that element, to highlight it.

Control Name

Subform Palette dialogs now open centered in the browser window if no valid control name is given. (Revision 38693, ST/JS/3610)

If the control name given in the cControl parameter of the "subformpaletteshow" client command is not found (or you omit the name), the palette dialog will be centered in the browser window, and the arrow is not displayed since there is no control for it to point at.

PDF Printing

Embedding Files and Data

Two new functions $embedfile() and $embeddata() have been added to the Omnis PDF Device to allow you to embed files and data into PDF files. (Revision 38713, ST/EC/1901)

$embedfile()

The $embedfile() function allows you to embed a file in a PDF file, such as an XML file. If cName is omitted, the name of the file in cFilePath is used.

```
Omnis PDF Device.$embedfile(cFilePath[, cName])
```

$embeddata()

The $embeddata() function allows you to embed the data in cData with the given name cName in a PDF file.

```
Omnis PDF Device.$embeddata(cData, cName)
```

Setting the Document Info

The **$setdocinfo** method now allows you to set the producer and creator of the output PDF file. (Revision 39956, ST/EC/1942)

The cProducer and cCreator parameters have been added to the $setdocinfo method to allow you to add the PDF producer and creator. The full syntax for the method is now:

$setdocinfo(cAuthor[,cTitle='',cSubject='',cKeywords='',cLanguage='en-GB',cProducer='Omnis Studio PDF Device',cCreator='Omnis Studio'])

- **cProducer**
  is the software that generated the PDF, the default is Omnis Studio PDF Device.
- **cCreator**
  is the software that created the content of the PDF, the default is Omnis Studio.

Push Notifications

Configuration and error handling for push notifications have been improved with the addition of new parameters for the 'openpush' client command, and a new client method $pushclosed which is called if a push connection closes. (Revision 38643, ST/JS/3561)

The 'openpush' client command (executed using $clientcommand) has two new optional row() parameters iMaxTries and cRetryCodes.

- **iMaxTries**
  The max number of times a request will automatically be tried, if it returns a http status code matched by cRetryCodes. -1 Means unlimited. Default is 5.
- **cRetryCodes**
  HTTP status codes to treat as 'temporary errors'. These will be retried automatically, up to iMaxTries times before reporting an error. This is a string containing comma-separated codes and/or code ranges, e.g. "404,500-599" means any responses with http status code 404 or anywhere between 500 and 599 will automatically be retried. Default is 500-599.

The timeout period of each request is 60 seconds, so it can take multiple minutes for an error to be reported, if you have more than a few iMaxTries. Timeouts are a normal part of push connections – if the server hasn't pushed anything back, the requests will keep being sent and timing out until it does.

In addition, there is a new client-executed method **$pushclosed** which can be implemented for remote forms and is called (on all forms) if a push connection closes. Return kTrue from the method to prevent the default behavior.

$pushclosed has two parameters:

- **pErrorCode**
  Non-zero values indicate it was closed due to an error.
  0 means it was closed by calling the 'closepush' $clientcommand.
  > 1 means pErrorCode is the http status code which caused the error. An error message will be displayed, unless you return kTrue to prevent the default behavior.
  -1 means a transport error, i.e. the request could not be delivered.
- **pWillRestart**
  If true, the default behavior will be to restart the push connection. Return kTrue to prevent this.

Construct Row Variable

Headers Column

A new **headers** column containing the HTTP headers received from the JavaScript client has been added to the construct row variable for remote tasks and forms. (Revision 39230, ST/JS/3690)

The $construct row variable in a remote task now has an additional column named **headers,** containing a row with each column matching a name-value pair for each HTTP header received from the client, e.g. Host: developer.mozilla.org will include the header 'host' as the column name with the 'developer.mozilla.org' as the value. The header name is lowercased and all hyphens are removed, e.g. Accept-Language becomes acceptlanguage.

Note a further enhancement was made in revision 39589 – any spaces at the beginning of the header values are now stripped out.

Any spaces at the beginning of the header names in the **headers** construct row value are now removed. (Revision 39589, ST/PF/1475)

Cookies Column

A new column named **cookies** has been added to the construct row variable that contains the cookies available in the client browser. (Revision 38472, ST/TC/043)

The $construct row parameter in a remote task now has an additional column named cookies, containing a row with each column matching a name-value pair for each cookie, for example cookie1=hello will result in a row with the column name 'cookie' and 'hello' as the value.

Return Methods

A Promise is now returned from calls to server-executed methods from the client if there is no _return method setup. (Revision 38819, ST/JS/3622)

The default behavior now, when calling a server-executed method from the client, is to return a Promise, unless there is a matching ..._return method, in which case it will call back to the return method when complete (the behavior in previous revisions).

The return value from the server method will be passed through as a parameter in the Promise's "then" function. This allows you to chain your asynchronous logic in the same method, maintaining context and continue to use variables defined in the outer method. For example:

```
Do $cinst.$myServerMethod() Returns lPromise
JavaScript: lPromise.then( (returnVal) => {
JavaScript: lRetVal = returnVal; // Assign the parameter to a local variable
   that Omnis code knows about
Do $cinst.$showmessage(lRetVal,"Server Method Complete")
JavaScript: });
```

Date Parsing

The Date Parser in Omnis tries to convert a string representation of a date or date/time value to an Omnis Date/time format. Date string parsing has been improved and now falls back to **Short date/time** formats. (Revision 39079, ST/JS/3684)

The Short date/time format depends on the browser locale, so for GB English it is D/M/y, and for US English it is M/D/y, for example. Therefore, an end user could enter "3/5" into any Date or Date/time field, and it might be converted to "3 May 2024" or “March 5 2024”, for example.

HTTP Server

The HTTP Server built into Omnis now supports IPv6. (Revision 38474, ST/WT/1881)

The HTTP Server in Omnis Studio allows you to test the Remote forms in your web and mobile applications. The HTTP Server on macOS already supports IPv6, but IPv6 support has been added for Omnis running on Windows and Linux.

There is a new item "IPv" in the 'server' group in the Omnis Configuration file (config.json). The values can be 4 for IPv4, 6 for IPv6, and 64 for a dual-stack IPv6 socket, that is, an IPv6 socket which has support for IPv4 on the same socket. By default, IPv4 will be used.

Layout Minimum Height

The minimum value of **$layoutminheight** has been reduced to 10 (the previous minimum value was 100). (Revision 38276, ST/LR/057)

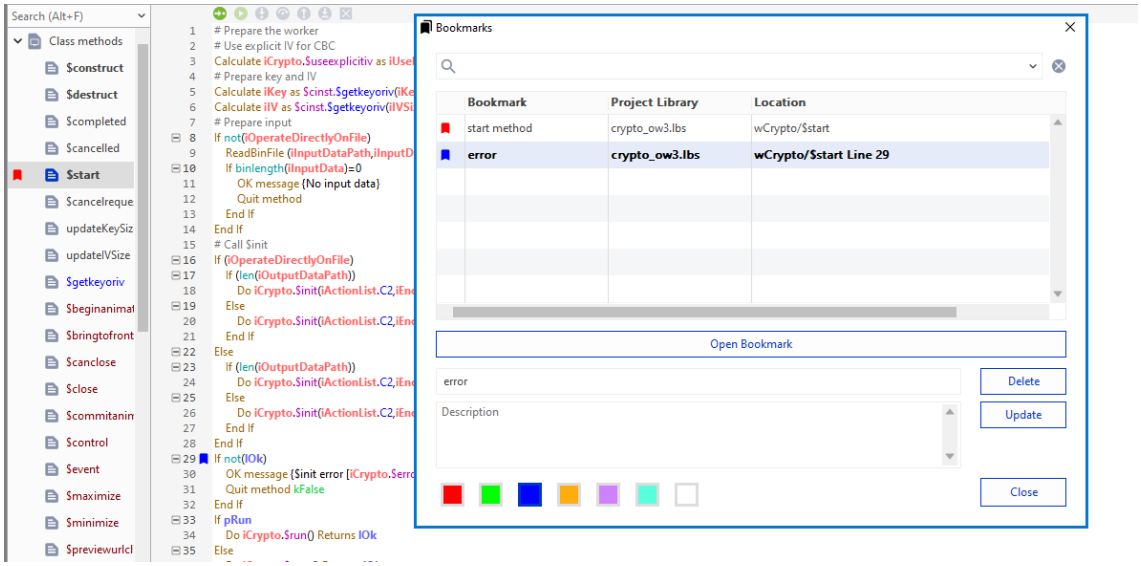
The value range of $layoutminheight is now 10 to 32000 inclusive. Note it is set to 0 by default (as in previous revisions), which means the layout height is 2 pixels below the lower edge of the bottom-most component on the remote form.

Debugging Methods

Method Bookmarks

You can now add **Bookmarks** to methods or to individual method lines which allow you to mark significant places in your code and easily locate them. (Revision 39442, ST/CE/245)

You can show Bookmarks in the Method Editor by checking the **Show Bookmarks** option in the **View** menu (enabled by default). Bookmarks are stored in a new system table called **#BOOKMARKS** which is in the System Classes folder and can be edited in the Bookmarks editor.



To add a bookmark, Right-click on the *method name* or in the margin of a *method line* (where breakpoints are normally shown). When creating a bookmark, you can give the bookmark a name, description and choose a color. Bookmarks added to a method line can be moved up or down using drag and drop.

All Bookmarks for the current library are listed in the Bookmark editor which you can open by double-clicking on a bookmark in the Method Editor, by selecting the 'Open

Bookmarks' option in the View menu, or by double-clicking on the #BOOKMARKS system table in the Studio Browser. You can update, delete, or jump to a bookmark from the Bookmarks list. When jumping to a bookmark, the bookmark is highlighted briefly.

Exporting Method Bookmarks

When exporting a library to JSON, the bookmarks are exported. However, the bookmarks system table #BOOKMARKS is not exported. In this case, bookmarks are exported as part of the method, and when importing the library from JSON, the bookmarks table is rebuilt.

Method Bookmarks and the VCS

When you attempt to add or edit a bookmark in a library that has been checked out of the Omnis VCS, the #BOOKMARKS system table must be checked out. The VCS will do a check when you select either the Bookmark Line or Bookmark Method option, or try to edit a bookmark, to see if #BOOKMARKS is still read-only. If so, the VCS will prompt you to check it out. If the checkout fails, or you cancel the checkout dialog, a message will pop up indicating the bookmark cannot be added or edited while the class is read-only.

If #BOOKMARKS does not exist in the corresponding VCS project (e.g. an existing library which has not yet had the class checked in), the class's $showascheckedout property is toggled and a message confirms this. You then need to ensure that the #BOOKMARKS class is checked into the VCS.

Omnis Environment

macOS Tree Restructure

On macOS the application tree has been restructured to adhere to the Apple best practice guidelines. This will improve verification speeds when Omnis first starts up. (Revision 38518, ST/IN/281)

As a consequence of the macOS tree restructuring, you should be aware that many files have changed location which may have an effect on the functioning of your application or the deployment process you use. You should read the following sections for details of the changes and make any necessary adjustments to your application or deployment setup.

Only the main core Mach-O binary executable is now inside the Contents/MacOS folder.

Any external component previously existing in the external, xcomp and jscomp folders is now placed in the **Contents/PlugIns** folder. An external component will be identified by its file suffix which will be one of u_external, u_xcomp or u_webdesign. The wesecure and weshared bundles are also now placed into PlugIns.

Third-party components should be placed in the Contents/PlugIns folder if part of the application bundle.

Components can still be placed in the legacy xcomp and jscomp folders within a user's application data folder as well as the PlugIns folder. Where there is a component with the same name in the PlugIns folder of the user's application data this will be loaded instead of the duplicate in the legacy xcomp or jscomp folder.

The Contents/Helpers directory now contains any binary which supplements the use of Omnis Studio and is not directly loaded. This will contain binaries for nodejs (the node packages and other files remain in Resources), crashpad, Omnis load sharing and web server plug-ins.

All other non-binary files which were previously present in Contents/MacOS have been moved into Contents/Resources. The original custom directory names will persist, e.g. /Contents/Resources/firstruninstall

See the Appendix at the end of this document for specific changes in the online docs regarding the macOS tree restructuring.

DAMs

The DAM components now locate their client libraries from a set of predetermined paths which are the recommended locations for installing or bundling third party libraries.

For more details see Tech note: TNSQ0043 – Loading External macOS Libraries

Notarization

The **Notarize** option in the Deployment tool has been updated and now uses the notarytool command to notarize your application. (Revision 38542, ST/AD/326)

Older revisions of the Deployment tool use the altool command, but this will no longer work since Apple has removed support for altool. You can however notarize the bundle built with the Deployment tool using the notarytool command as follows:

```
xcrun notarytool submit <path-to>/<app-name>.dmg --apple-id <developer-email>
   --team-id <team-id> --password <app-specific-password> --verbose –wait
```

SQL Query Builder

Query Builder Manager

A **Query Builder Manager** has been added to the SQL Query Builder toolbar and the query time has been added to the Results tab. (Revision 38339, ST/SS/448)

The new Query Builder Manager lists the queries you have saved in the Query Builder and is accessed via a new button next to the list of queries in the Query Builder toolbar. In the Query Builder Manager, you can search for queries using the Find button, and you can amend a query's description.

When you Run a query, the time taken to run the query is now shown in the status bar of the Results window.

Export Options

The **Export Statement** and **Export Results** options have been added to the context menu in the SQL Query Builder. (Revision 39455, ST/SS/469)

Omnis Configuration

The **$getconfigjson()** and **$setconfigjson()** methods ($root.$prefs) have an additional optional boolean parameter **bBaseConfig,** to control whether the methods operate on the userconfig.json (the default) or the config.json configuration file. (Revision 38344, ST/FU/896)

The syntax for the updated methods is:

- **$getconfigjson**([bBaseConfig=kFalse])
  Returns userconfig.json as a row when bBaseConfig is set to kFalse or is omitted (or empty if userconfig.json could not be parsed). If bBaseConfig is kTrue, returns config.json instead.
- **$setconfigjson**(wConfigJson[, bBaseConfig=kFalse])
  Sets userconfig.json to the supplied row when bBaseConfig is set to kFalse or is omitted. If bBaseConfig is kTrue, sets config.json instead.

By default the $get and $set methods will operate on userconfig.json when bBaseConfig is set to kFalse or is omitted. You can operate on the base config.json by

setting bBaseConfig to kTrue; however you should avoid editing config.json directly, so in general you should update userconfig.json.

Note that *userconfig.json overrides settings in config.json,* so if a setting is present in both config.json and userconfig.json, the userconfig.json value will be used.

Menu Theme Colors

The setting **colormenuselectedtext** has been added to the 'menu' section of the appearance.json theme file which controls the color of the text of the currently selected line in a menu. (Revision 39255, ST/MC/279)

The colormenuselectedtext theme color allows you to adjust the color of the text of the currently selected line in a menu (Menu Class). This can be changed by setting the $appearance property (an Omnis preference) in the Property Manager. This is available for Windows only.

Omnis Task Bar

On Windows only, when the task bar is shown, the context menu on the task bar now includes an option **Show FullInames** to show or hide the library part of class names. There is also a new option **taskBarShowsFullClassnames** in the 'windows' section of the Omnis configuration file to set this option, which you can set in the Configuration Editor. (Revision 38883, ST/HE/2068)

Studio Now Sample Libraries

There is a new 'Studio Now' filter in the **Samples** section in the Hub to show sample libraries that have been added to the latest Studio Now release. (Revision 38502, ST/BR/463)

Window Components

Border Icons

Border icons in Entry fields now respond to clicks. This applies to controls that support $bordericonstyle including Entry fields, Token Entry fields, Combo boxes, and Drop lists (Revision 38829, ST/BE/1821)

Clicking on a border icon now generates the event evBorderIconClicked. The event has the parameter pLeftBorderIcon. If true the left border icon was clicked, or if false the right border icon was clicked. For example:

```
On evBorderIconClicked
   If pLeftBorderIcon
      Popup menu MyOptionsMenu
   Else
      Send to trace log Right Icon Clicked
   End If
```

OBrowser

Scroll Zoom

The $allowscrollzoom property has been added to **OBrowser** to toggle the scroll zoom feature. (Revision 39299, ST/EC/1893)

The **$allowscrollzoom** property allows you to toggle the scroll zoom feature within a browser window inside OBrowser. When set to kTrue (the default), the end user can zoom in and out of the browser content by holding down the **Ctrl** key while scrolling the mouse wheel. If $allowscrollzoom is set to false, this zoom functionality is disabled. Note that when a browser page is cached, the current zoom size of the content is also cached. This means that the page will be rendered at that scale, even if the zoom functionality was disabled.

CEF Switches

You can now add cefSwitches with values to the "obrowser" item in the Omnis configuration file, edited using the Configuration Editor. (Revision 38461, ST/EC/1895)

You can define cefSwitches with values in the config.json. For example, to enable web socket connections from any origin, add the following:

```
"cefSwitches": [
        "remote-allow-origins=*"
    ],
```

Entry Fields

Various properties have been added to control the text font size and color of animated content tips. (Revision 39021, ST/WC/600)

The properties **$labelcolor, $labelfontsize,** and **$labelhottextcolor** have been added to Entry fields and Combo boxes to control the text color and size of the floating content tip labels. By default, these map to a font size of 0 and kColorDefault to maintain behavior in previous revisions where the color of the label text is the same as the field text and the font size is 80% of the edit field.

Tab Strip

The scroll speed of a vertical Tab Strip has changed so, by default, the control uses the number of tabs and the size of the control to determine the scroll speed, plus there are some new config items to control the scroll speed and behavior. (Revision 38500, ST/WO/2836)

You can control the scroll speed using the new **tabstripSpeedToScroll** item in the 'defaults' section of config.json. The default is 0 which uses the new scroll speed, or you can enter the number of milliseconds you would like the scroll animation to use; 600ms was the default in previous revisions.

You can control whether the control scrolls when clicked using the new **tabstripScrollsOnClick** item in the 'defaults' section of config.json. When set to false (the default), the control scrolls as the pointer enters the scroll buttons. When set to true the scroll buttons must be clicked to scroll the control. In this case, the Tab Strip will scroll an average tab height per click and is not animated.

Masked Entry Field

The **$passwordchar** property has been added to the Masked Entry Field. (Revision 39799, ST/WO/2860)

You can use '*' in $passwordchar in a Masked Entry Field to replace characters with asterisks to hide the field content (black dots are not supported for masked entry fields).

Pushbuttons

The style of pushbuttons set to kSystemButton in $buttonstyle has changed on macOS. (Revision 39811)

For headed lists and tree lists, $buttonstyle is the style of the header button, either kSystemButton or kUserButton. On macOS, the kSystemButton button style now uses the style of a native system pushbutton with a flexible height. If the height exceeds that of a regular size button the button keeps the same style.

In previous revisions, the behavior was to use the square system style when the height was taller than a regular size button. To enable the legacy behavior, set the new $squaresystembuttons root preference to kTrue.

Headed List

A new property **$showcolumnslinesifempty** has been added to Headed Lists to show column lines when the list is empty; this only applies when $showcolumnlines is also kTrue. (Revision 39451, ST/WO/2858)

Report Programming

Enterable Report Fields

Report entry fields have a new property **$enterable** allowing the end user to enter and save text in a PDF report. (Revision 38452, ST/EC/1849)

You can allow the end user to enter and save text in a field in a PDF report by setting its $enterable property to kTrue. When the end user opens the PDF report, the field is active and enterable and once text has been entered the PDF can be saved. Note this only applies to PDFs that have been generated using the Omnis PDF Device.

Note that if the enterable report field is empty when it is printed, and the $nolineifempty property is set to kTrue, then the whole line including the enterable field will not be included in the report. In this case, it is advisable to set $nolineifempty for enterable report fields to kFalse (which is the default).

The **PRIobjectStruct** has also been updated with a new **mEnterable** member which external component developers could use to make their own enterable fields.

OW3 Worker Objects

HTTP Worker Object

Using AI in Omnis

You can access AI models via their APIs using the Omnis **HTTP Worker.** An example application called **HTTP AI** has been added to the **Samples** section of the **Hub** in the Studio Browser to demonstrate how you can access various AI models. Note that you will need to get an **API key** from the respective AI provider to use their service, which may incur charges, or a free demo key may be available. (Revision 39432, ST/TU/054)

In the example, Omnis uses the HTTP Worker to make restful calls to the API of the respective AI service. Object classes are provided to make calls to Google's Gemini, OpenAI, and Anthropic, which inherit the HTTP Worker object – support for AI Vision is also included which allows you to analyze and interpret the content in image data. In each of these object classes, $runprompt is called to build the request in JSON format and make the restful call. On completion, $completed is called and passes the results to $ai_completed or $ai_error in the calling window instance. An additional object class oAIFunctions is provided where methods prefixed with $ai_ may be added for use with function calls as demonstrated in the example.

Note that the AI configurations that you add in the example app are added to the userconfig.json configuration file in a group named 'ai', including the API key which is stored in plain text.

Header Content Type

The content-type header 'application/json' is now added automatically when content is converted from a row or list parameter. (Revision 39412, ST/EC/1922)

HASH Worker Object

The $initsignature method has a new optional parameter iSignatureType to allow you to select the RSA or ECDSA signature type. (Revision 39415, ST/EC/1902)
In addition, the bBlind parameter is now set to kTrue by default. (Revision 39456, ST/EC/1928)

The syntax for the $initsignature method is now:

- **$initsignature**(vData, iHashType, vPrivateKeyPEM [,bBlind=kTrue, iSignatureType=kOW3signatureRSA])

Where **iSignatureType** can be kOW3signatureRSA (the default) or kOW3signatureECDSA. The latter enables ECDSA signatures, such as ES256, ES384 and ES512 depending on the hashing algorithm selected in iHashType (SHA256 for ES256, SHA384 for ES384, SHA512 for ES512).

When using a ECDSA signature, the PEM private key used for signing must also be an elliptic curve of the correct type for ES256, 384 or 512. The $initverifysignature method can verify both RSA and ECDSA signatures.

When bBlind is kTrue (now the default) the RSA encryption used to generate the signature uses blinding.

OAUTH2 Worker Object

PKCE

The $pkceentropy property has been added to the OAUTH2 Worker to control the code verifier length for PKCE. (Revision 39423, ST/EC/1925)

The **$pkceentropy** property controls the code verifier length for PKCE and takes an integer value between 43 (the default) and 128.

$completed callback method

The pRow in the $completed callback method for the OAUTH2 Worker Object has a new column called *state,* which contains the value you set up in the $oauth2state property. (Revision 38315, ST/EC/1889)

The *state* column contains the value you set up in the $oauth2state property. This helps you to identify which user the callback belongs to. For example, where multiple OAuth2 workers are used to authenticate multiple mobile devices, you could set a UUID in the $oauth2state property and keep a list of username:uuid until the $completed callback, where you can match up the token received.

LDAP Worker Object

The $init method in the LDAP worker can now accept the cUser and cPassword parameters as either Character or Binary; this enhancement is to accommodate names with umlauted characters. (Revision 39596, ST/EC/1921)

When Character is used, Omnis sends cUser and cPassword as UTF-8. When Binary is used, Omnis sends cUser and cPassword using whatever encoding is required by the server, e.g. CP1252 is required by some servers.

Omnis VCS

VCS API

The way you access the VCS API has changed. You should now use the $getapiobject method to access the VCS API methods. (Revision 39610, ST/VC/837)

To call the VCS API methods you first need to use the **$getapiobject()** method to return an object reference to the VCS API, for example:

```
Do $root.$modes.$getapiobject('VCS') Returns iAPIObjRef
```

Once the object reference has been generated, all calls to the API can be made using this reference, allowing a simpler syntax for method calls. For example, to call the $logon method, you can use:

```
Do iAPIObjRef.$logon(cUSERNAME, cPASSWORD, nTokenTime, cToken, cErrors,
rSessionOrSessionPoolRef, cDBName) Returns bStatus
```

This new approach means that you can no longer logon using any defined session templates. Instead, you have to use a reference to either a logged on SQL session or a session pool (e.g. $sessions.MY_VCS_SESSION or $sessionpools.MY_VCS_SESSIONPOOL) specified in the **rSessionOrSessionPoolRef** parameter. This allows you to logon to a session via code.

See VCS API in the online docs for full details including the new syntax for all the VCS API methods.

Note that the old method of accessing the VCS API using $dotoolmethod() and the $x_ API methods will continue to work, but you are advised to use the new method going forward.

$buildProject method

$buildProject VCS API method now allows classes to be locked or excluded in the build. (Revision 39788, ST/VC/842)

The $buildProject method has two new list parameters, **excludeClassList** and **lockClassesList**. Both are defined with **cClassName** and **cClassType** (e,g. wTest,

kWindow) and contain a list of classes to either exclude from the build or to be built locked.

Update from VCS

The **Update from VCS** option is now available when you right-click on individual or selected classes and folders in the Studio Browser. (Revision 38480, ST/VC/829)

A new context menu option **Update from VCS** is available at class level in the Studio Browser. Note the existing Update from VCS hyperlink option checks the whole library. Therefore, if you only want to update the selected class(es) or folder(s), you can use the new context menu option.

Update Local Library

The **Update Local Library** option is now available in the VCS to update the local library relating to the selected project. (Revision 38498, ST/VC/817)

A new hyperlink and context menu option **Update Local Library** allows you to update the local library with latest changes from the current, selected project. It is the equivalent of the 'Update from VCS' option except that it pushes any updates to the relevant local library.

VCS Privileges

The Privileges window in the VCS has been updated and now allows you to set privileges for all the classes in a folder or for a whole project. (Revision 38290, ST/VC/818)

The **Privileges** window in the Omnis VCS now displays a tree of classes, including all classes within a folder. Classes or folders can be selected or deselected individually. There is also a new second pane which lists all the privileges that have been set for the current project. In addition, it is now possible to set privileges *at a project level*, with new hyperlink and context menu options to initiate this. Note that when these options are used, the tree only shows the project name, not the individual classes within it.

Window Programming

Enter Data Mode

A reported issue with the Enter Data command and the window Close box has been resolved in this revision of Omnis Studio 11.2. (Revision 39959, ST/FU/917)

For all versions of Omnis Studio up to and including 10.22, when closing a window, the Enter Data command was left on the command stack if not duly processed. To cancel the command correctly, you had to use the standard OK or Cancel buttons or catch the window close event and execute a command such as *Queue Cancel* to terminate the Enter Data command.

In Omnis Studio 11, a change was made such that the close box automatically cancelled the Enter Data command, but this change could potentially break legacy applications. This latest revision of Omnis Studio 11.2 reverts to the 10.2x behavior.

There is a new configuration item **enterDataCancelOnWindowClose,** in the 'defaults' section of config.json, that controls if Enter Data mode is cancelled when a window closes. This is set to false by default, which is the recommended state. Any code in Studio 11 that used the above anomaly should be modified to process the Enter Data command correctly when closing a window.

The **deferUserWindowClosure** item, also in the ‘defaults’ section of config,json, has been set to false in this revision as this was causing some windows to ghost during a closing process.

Form and Report Wizards

The Omnis Form Wizard and Omnis Report Wizard are now available in the Studio Browser when the **Datafiles** option is enabled in the IDE Options – they were missing from previous revisions of Studio 11.x. (Revision 39098, ST/HE/2085)

The **Omnis Form Wizard** and **Omnis Report Wizard** allow you to create a form (window class) or report class based on an Omnis File class that links to an Omnis datafile. The wizards are only available when the **Datafiles** option is enabled in the IDE Options in the Studio Browser.

Note these wizards and file classes should only be used in legacy apps that use Omnis datafiles, and not for new applications.

Libraries and Classes

Recent Libraries

You can now remove a library from the **Recent Project Libraries** list in the Studio Browser by right-clicking on the item in the list and selecting **Remove from list.** (Revision 38463, ST/BR/460)

There is also a new option ‘Add Samples to 'Recent Project Libraries' in the IDE Options in the Hub to control whether the sample libraries (available under the Samples option in the Hub) are added to the **Recent Project Libraries** list. This is disabled by default so sample libraries are not added to the recent library list.

Exporting Libraries to JSON

When exporting a library to JSON and the fullexportimport preference option is set to kFalse, various VCS-related properties are now excluded from JSON export. (Revision 39976, ST/IE/254)

The following VCS-related properties are excluded from the JSON export when the fullexportimport option (part of the $exportimportjsonoptions root preference) is false:

- Library properties:
  $vcsbuilddate, $vcsbuildersname, $vcsbuildnotes
- Class properties:
  $vcsrevision, $showascheckedout

File Classes

The file class editor now includes userinfo for each field in the file class which can be used to store data of any type for the field. This maps to fileclass.field.$userinfo. (Revision 39035, ST/BE/137)

SQL Programming

Cancelling a long-running fetch

You can cancel a long-running fetch by pressing the system break keys (Ctrl+Break on Windows or Cmnd+. on macOS). (Revision 39995, ST/*A/177)

$fetch() will also update a Working message window, and the Repeat Count will display the number of rows fetched so far. In this mode, the Cancel button will also terminate the fetch. The fetch list will retain any rows that have already been added.

List Programming

Searching Lists

You can now find a line in a list using the $search() method without setting the selected line to the line found using a new parameter bSetCurLineWhenNotSelecting. (Revision 39424, ST/FU/905)

A new optional parameter **bSetCurLineWhenNotSelecting** has been added to the $search() method. The syntax is now:

$search(calculation [,bFromStart=kTrue, bOnlySelected=kFalse, bSelectMatches=kTrue, bDeselectNonMatches=kTrue, bSetCurLineWhenNotSelecting=kTrue])

When bSetCurLineWhenNotSelecting is passed as false (the default is kTrue), and the select/deselect options are false, the search is completed, the found line number is returned but the found line is not selected.

JSON

JSON Object Methods

Adding Missing Members

Some of the JSON Object methods have a new parameter to ensure missing objects are created automatically. (Revision 39226, ST/EC/1799)

The following JSON Object methods have an optional third parameter bAddMissingMembers. When this is passed as kTrue, the method creates the missing members (objects only), i.e. {}; when omitted defaults to kFalse. The following methods have the new parameter:

$setstring(), $setbool(), $setinteger(), $setfloat(), and $setobject(). For example, the syntax for $setstring() is now:

- **$setstring**(cMember,cString[,bAddMissingMembers=kFalse])
  Sets specified member to JSON string with value cString. When bAddMissingMembers is passed as kTrue, creates the missing members (objects only), defaults to kFalse. Returns true for success.

In addition, the following External SDK methods also have a new pAddMembersIfMissing parameter:

qlong ECOupdateJSON(EXTfldval &pJSONList, EXTfldval &pMemberIdFval, EXTfldval *pValueFval, EXTfldval *pMemberNameToAddOrRemove, EXTfldval &pErrorText, qbool pAddMembersIfMissing = qfalse);

qlong ECOupdateJSONEx(EXTfldval& pJSONList, EXTfldval& pMemberIdFval, EXTfldval* pValueFval, EXTfldval* pMemberNameToAddOrRemove, EXTfldval& pErrorText, qlong pFlags = 0, qbool pAddMemberIfMissing = qfalse);

Passing the pAddMemberIfMissing parameter as qtrue will create the missing members (objects only); defaults to qfalse.

Call ECOupdateJSON with non-null pValueFval and pMemberNameToAddOrRemove to add new member pMemberNameToAddOrRemove (with value pValueFval) to existing member identified by pMemberIdFval. With pAddMemberIfMissing set to true to pre-create the missing members.

Column Types

The iOptions parameter and kOJSONoptionAllowAnyType constant have been added to the $listtoobjectarray() JSON Object method to allow you to use column types other than the simple column types supported in previous revisions, namely integer, number, boolean, string. (Revision 39102, ST/FU/735)

The syntax of JSON.$listtoobjectarray() is now:

- **$listtoobjectarray**(lList [,iEncoding=kUniTypeUTF8, &cErrorText, iOptions=kOJSONoptionNone])
  iOptions can be one of the following constants: kOJSONoptionNone (the default) or kOJSONoptionAllowAnyType which would allow you to use column types, such as Row, beyond the simple column types (integer, number, boolean, string) supported in previous revisions

In addition, the ECOsaveToJSONObjectArray_AllowAnyTypes function has been added to the External SDK that works just like ECOsaveToJSONObjectArray, but it allows any type as columns, like the kOJSONoptionAllowAnyType.

Functions

bitand() and bitor()

The *bitand()* and *bitor()* functions can now be executed on the client. (Revision 39932, ST/JS/3761)

bitclear()

The *lastBitNumber* parameter of the *bitclear()* function is now optional. If *lastBitNumber* is omitted, the value in *firstBitNumber* is used allowing you to clear a single bit. (Revision 38248, ST/FU/893)

Syntax

bitclear(*binary*,*firstbitnumber*[,*lastbitnumber=firstbitnumber*])

You should note that bit numbers are *zero-indexed* and bit 0 is the most significant bit.

bool()

There is a new function **bool()** which converts a value to a Boolean. (Revision 38265, ST/FU/891). The *bool()* function can now be executed on the client. (Revision 39946, ST/JS/3762)

Function group	Execute on client	Platform(s)
General	YES	All

Syntax

bool(*pValue*)

Description

Converts *pValue* to Boolean. In effect, it returns the 'truthiness' of a value.

Example

```
Do bool(#NULL)          ## would return kFalse

If kTrue|bool(#NULL)    ## Would resolve to true
 ...
End If

If bool(lObjRef)        ## Would resolve to true if lObjRef was set to an
   instance, and false if unset
 ...
End If
```

coalesce()

Note the *coalesce()* function was added in Revision 38856 (ST/FU/901) but its operation has been changed in this revision; the following is the revised definition. Plus the *coalesceempty()* function has been added. (Revision 39293, ST/FU/907)

The *coalesce()* and coalesceempty() functions can now be executed on the client. (Revision 39949, ST/JS/3763)

Function group	Execute on client	Platform(s)
General	YES	All

Syntax

coalesce([*pvalue* …]])

Description

Returns the first non-NULL parameter from a list of values.

coalesceempty()

Function group	Execute on client	Platform(s)
General	YES	All

Syntax

coalesceempty([*pvalue* …])

Description

Returns the first non-NULL and non-Empty parameter from a list of values.

FileOps.$copy()

Two new static functions **$copy()** and **$move()** have been added to the FileOps external package, while the existing functions $copyfile and $movefile have been deprecated and should not be used for future development. (Revision 39607, ST/FU/909)

The $copy() and $move() functions operate similarly to the old $copyfile and $movefile functions, but offer improved syntax and broader applicability, since they operate on both files and folders. Additionally, they include new parameters to create missing paths, replace existing files, and merge content, which by default are set to false.

Function group	Execute on client	Platform(s)
FileOps	NO	All

Syntax

$copy(*cFromPath*,*cToPath*[,*bCreatePath*=kFalse,*bReplace*=kFalse,*bMerge*=kFalse])

Description

Copies the file or folder specified in *cFromPath* to *cToPath*.

FileOps.$move()

Function group	Execute on client	Platform(s)
FileOps	NO	All

Syntax

$move(*cFromPath*,*cToPath*[,*bCreatePath*=kFalse,*bReplace*=kFalse,*bMerge*=kFalse])

Description

Moves the file or folder specified in *cFromPath* to *cToPath*.

If *bCreatePath* is kTrue, the $copy or $move functions will automatically create any missing folders in the path specified by the pToPath parameter.

When *bMerge* is kTrue, and the destination is a file, the contents of the source file are appended to it. If the destination is a folder, the files and subfolders from the source are merged with those in the destination.

If *bReplace* is kTrue, the destination file or folder is deleted before the source is copied or moved.

When *bReplace* and *bMerge* are both kTrue, and the destination is a file, replace takes precedence over merge, meaning the destination file is removed before the source is copied or moved. If the destination is a folder, merging occurs, but existing files in the destination are overwritten by those from the source.

Note: for destination folders that do not exist to be recognized as folders, they must end in the platform-specific filepath separator, otherwise they will be interpreted as files.

FileOps Worker Objects

The $init functions for the Copy and Move FileOps Worker objects have also been updated, with the addition of the *bCreatePath*, *bReplace*, and *bMerge* parameters.

FileOpsCopyWorker:

$init(*cFromPath*, *cToPath* [,*vTag*,*bCreatePath*=kFalse, *bReplace*=kFalse, *bMerge*=kFalse])
Initializes the object so it is ready to copy the file or folder specified in cFromPath to cToPath.

FileOpsMoveWorker:

$init(*cFromPath*, *cToPath* [,*vTag*,*bCreatePath*=kFalse, *bReplace*=kFalse, *bMerge*=kFalse])
Initializes the object so it is ready to move the file or folder specified in cFromPath to cToPath.

FileOps.$joinpath()

The *$joinpath()* function has been added to the FileOps external package to allow you to join a number of path segments. (Revision 38859, ST/EC/1910)

Function group	Execute on client	Platform(s)
FileOps	NO	All

Syntax

FileOps.$joinpath(*cSegment,* ...)

Description

Joins all given segments in *cSegment* together using the platform-specific separator as delimiter. Zero-length segments are ignored.

On Windows only, if / is found in the final joint path, it will be replaced with the Windows path separator: \\

FileOps Workers

Tag parameter

The **FileOps Workers** (Copy, Delete, Move) now have an extra last parameter vTag added to their $init methods. This works exactly as the tag parameter for the JavaScript and Python Workers. (Revision 38270, ST/EC/1885)

If supplied, the vTag parameter is some data that is passed to $completed in the column __tag of the row parameter. This can be used for example to identify the caller when the worker object is shared by several instances.

Thread Lock

The lock on the main thread caused when cancelling a FileOps worker has been removed. (Revision 38285, ST/EC/1886)

The lock on the main thread when a FileOps worker is cancelled, that is, either directly through a call to $cancel, or through the FileOps worker variable going out of scope, has been removed. However, you should note that the FileOps worker will still execute in the background even when cancelled, as it did before in previous revisions.

FileOps.$writefile()

A new parameter bCreateParentDirectories has been added to the FileOps.$writefile function to create the directories in the path. (Revision 39410, ST/FU/911 & ST/FU/912)

The syntax for FileOps.$writefile is now:

FileOps.$writefile(cFilePath,vVariable [,iEnc=kUniTypeUTF8, bBOM=kTrue, bReplace=kTrue, bCreateParentDirectories=kFalse])

When **bCreateParentDirectories** is kTrue (the default is kFalse), FileOps.$writefile will create the directories in the path before attempting to write the file.

In addition, FileOps.$writefile will use the default parameters when any of the parameters are null.

idletime()

There is a new function **idletime**() that returns the elapsed time in milliseconds since the keyboard or mouse was used. (Revision 38495, ST/FU/897)

Function group	Execute on client	Platform(s)
Date and Time	NO	All

Syntax

idletime()

Description

Returns the number of milliseconds elapsed since the mouse or keyboard was last used.

Omnis PDF Device.$embeddata()

Two new functions $embedfile() and $embeddata() have been added to the Omnis PDF Device to allow you to embed files and data into PDF files. (Revision 38713, ST/EC/1901)

Function group	Execute on client	Platform(s)
Omnis PDF Device	NO	All

Syntax

Omnis PDF Device.$embeddata(*cData*, *cName*)

Description

Embeds the data in *cData* with given name *cName* in the PDF file.

Omnis PDF Device.$embedfile()

Function group	Execute on client	Platform(s)
Omnis PDF Device	NO	All

Syntax

Omnis PDF Device.$embedfile(*cFilePath*[, *cName*])

Description

Embeds the file in the PDF. If *cName* is omitted, the name of the file in *cFilePath* will be used.

OREGEX.$replace()

There is a new external component named OREGEX providing $replace() and $replaceall() regex supported functions. (Revision 38706, ST/EC/1900)

Function group	Execute on client	Platform(s)
REGEX	NO	All

Syntax

OREGEX.$replace(*cSource, cTarget, cReplace* [, *iFlags,* &*cErrorText*])

Description

Searches for a match in *cSource* using pattern in *cTarget* and replaces the matched substring with *cReplace*.

OREGEX.$replaceall()

Function group	Execute on client	Platform(s)
REGEX	NO	All

Syntax

OREGEX.$replaceall(*cSource, cTarget, cReplace* [, *iFlags*, &*cErrorText*])

Description

Searches for all matches in cSource using pattern in *cTarget*, and replaces the matched substrings with *cReplace*.

For both functions, the optional *iFlags* parameter is the sum of kORegEx... constants which determine how the regular expression engine will behave. The flags can be summed together, e.g. kORegExCaseInsensitive+kORegExMatchNotNull. They are:

Constant	Description
kORegExMatchNotNull	Do not match empty sequences
kORegExMatchContinuous	Only match a sub-sequence that begins at first
kORegExFormatNoCopy	Do not copy unmatched strings to the result when replacing
kORegExCaseInsensitive	Character matching performed without regard to case
kORegExMultiline	^ shall match the beginning of a line and $ shall match the end of a line when using ECMAScript engine, not available on Windows

For both functions, the optional *cErrorText* parameter provides further information about any error that is generated.

rcedit.$getapplicationmanifest()

Some new static methods have been added to the **rcedit** external component to return various information about files (note rcedit is available on Windows only). (Revision 38316, ST/EC/1891)

Function group	Execute on client	Platform(s)
rcedit	NO	All

Syntax

$getapplicationmanifest(*cFileName*)

Description

Returns the manifest from the exe/dll in *cFileName*.

rcedit.$getfileversion()

Function group	Execute on client	Platform(s)
rcedit	NO	All

Syntax

$getfileversion(*cFileName*)

Description

Returns the file version from the exe/dll in *cFileName*.

rcedit.$getproductversion()

Function group	Execute on client	Platform(s)
rcedit	NO	All

Syntax

$getproductversion(*cFileName*)

Description

Returns the product version from the exe/dll in *cFileName*.

rcedit.$getresourcestring()

Function group	Execute on client	Platform(s)
rcedit	NO	All

Syntax

$getresourcestring(*cFileName*, *iKey*)

Description

Returns the resource string with key *iKey* from the exe/dll in *cFileName*. You need an integer to identify the string. These can be inspected with tools such as Resource Hacker.

rcedit.$getversionstring()

Function group	Execute on client	Platform(s)
rcedit	NO	All

Syntax

$getversionstring(*cFileName*, *cKey*)

Description

Returns the version string with key *cKey* from the exe/dll in *cFileName*. The key could be some of the standard Windows PE keys, e.g.:

Comments
CompanyName
FileDescription
FileVersion
InternalName
LegalCopyright
LegalTrademarks
OriginalFilename
PrivateBuild
ProductName
ProductVersion
SpecialBuild

replace() and replaceall()

The *replace()* and *replaceall()* functions now have an optional fourth parameter to control whether a case-sensitive or case-insensitive replace is performed. (Revision 38261, ST/FU/892)

Syntax

replace(*source-string*,*target-string*,*replacement-string*[,*case-sensitive*=kTrue])

replaceall(*source-string*,*target-string*,*replacement-string*[,*case-sensitive*=kTrue])

If *case-sensitive* is kTrue (the default), or is omitted, a case-sensitive replace is performed. If passed as kFalse, a case-insensitive replace is performed.

Deploying Your Apps

Server Configuration

The **discardRequestsContaining** item has been added to the 'server' section of the Omnis Configuration file. (Revision 39233, ST/PF/1464)

The **discardRequestsContaining** item is an array of strings, where each string will be matched against a requested URL by the Omnis server and if it matches, the request is discarded. By default, the array contains the string:

```
"discardRequestsContaining": [
   "../"
]
```

This means that any requests received by the Omnis Server that contain ../ anywhere in the URL path will be discarded and an error returned to the client.

Firstruninstall

A hidden option has been added to disable the firstruninstall folder from copying over if the data folder already exists on Windows or macOS. (Revision 38834, ST/IN/296)

Omnis now checks for a file named "disable_fri.txt" in the root of the data folder, and if this file exists, Omnis does not copy files from the firstruninstall folder. Omnis does not read the file's contents, just checks if "disable_fri.txt" exists.

External Component SDK

Functions

A new SDK function **ECOisClosingProgram** has been added which returns a qbool: qtrue if Omnis is currently in its shut-down procedure, else qfalse. (Revision 39784, ST/EC/1940)

Appendix

The following changes have been made to the Omnis Online docs with regards to the macOS Tree Restructuring in Omnis Studio 11.2 Revision 38542. This may assist you in making any necessary adjustments to your application or deployment setup on macOS.

macOS Tree Restructure

Online Documentation changes

Creating Web & Mobile Apps

Chapter 2 - JavaScript Remote Forms

PDF Printing

Supporting Files

The blue text was replaced by the red text.

The PDF Device component is available for Windows and macOS and is located in the ‘xcomp’ folder and is loaded automatically.

is loaded automatically. It is located in the ‘xcomp’ folder on Windows and the ‘PlugIns’ folder on macOS.

Updated doc: 

Chapter 7 - Deploying your Web & Mobile Apps

Setting up the Omnis App Server

Server Logging

logcomp is the name of the logging component to use, that is, "logToFile" which references the logtofile.dll component in the logcomp folder of the Studio tree.

references the logtofile component located in the logcomp folder in the Windows and Linux Studio tree and the PlugIns folder in the macOS tree.

Updated doc: 

Omnis Programming

Chapter 10 - Report Programming

HTML Report Device

The Omnis Studio Print Manager API has been made public, allowing you to create your own custom printing devices as external components and place them in the XCOMP folder.

in the external component folder (PlugIns on macOS and XCOMP on other platforms).

Updated doc:

Chapter 17 - Deployment

Deployment Tool

macOS

The second screen allows you to specify the bundle's startup folder, iconsets (for the library), xcomp and icons folders, as well as the option to pre-serialise the bundle or add a custom read/write directory.

iconsets (for the library), PlugIns and icons folders,

Updated doc:

The label on the 3rd entry field should be Additions to /PlugIns.

Code Signing Omnis

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

The Studio application will only retain a valid signature if the contents of the signed application package remain unchanged, i.e. altering this folder will break the signature,

The code text in the following box should be replaced with this:

Omnis\ Studio\ 11\ x64.app/Contents

Updated doc:

Firstruninstall and Application Support Folders

To do this, when Omnis starts up it will check for the existence of a folder called 'firstruninstall' in the macOS folder in the Omnis package.

called 'firstruninstall' in the Resources folder

Updated doc:

Updating Components

Either type of component can be placed in:

~/Library/Application Support/ Omnis/\Omnis\ Studio\ 11\ x64/PlugIns/

Where there is a component with the same name in PlugIns this will be loaded instead of the duplicate in the legacy folder.

Insert the above before this existing line:

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

Updated doc: 

Patching a signed tree

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

xcomp, jscomp and PlugIns folders

Updated doc: 

Update Manifest Files

When Omnis starts, it reads the contents of the 'version' file in the root of its installation files, that is '/Application/Omnis Studio 11.app/Contents/MacOS/version' on macOS

which is located at '/Application/Omnis Studio 11.app/Contents/Resources/version' on macOS

Updated doc: 

Update on macOS

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

Resources folder

The code text in the following box should be replaced with this:

/Applications/Omnis\ Studio\ 11.2.app/Contents/Resources/version

Updated doc:

The updates are specified in a set of files which should be placed in a folder called “manifest” within the Omnis application's macOS folder.

Resources folder

The code text in the following box should be replaced with this:

/Applications/Omnis\ Studio\ 11.0.1.app/Contents/Resources/manifest/23071

Updated doc: (Scroll to view update)

Extending Omnis

Chapter 7 - OW3 Worker Objects

JavaScript Worker Object

Npm is provided alongside Node.js in Omnis Studio. *Insert after:* On Windows and Linux the nodejs folder is installed within clientserver\server. On macOS the nodejs folder is installed within Resources and the node executable is installed in Helpers.

To launch npm you can run index.js inside the npm folder, e.g.

./node npm/index.js

On Windows and Linux to launch npm you can run index.js inside the npm folder, e.g.

./node npm/index.js

On macOS if inside the npm folder then use the relative path to the node binary, e.g.

../../Helpers/node npm/index.js

Updated doc:

Omnis Studio External Components

Chapter 1 - Omnis External Components

Creating your own External Components

Components in Omnis

Loading Components

On all platforms, external components are loaded from the relevant sub-directory (xcomps, jscomps and logcomps folders) of both the Omnis data folder and the Omnis program / Application folder.

sub-directory (PlugIns, xcomps, jscomps and logcomps folders) of

Updated doc: 

Getting Started with Generic

Testing the Generic Component

To use the component, place a copy in the XCOMP folder of the Studio tree.

XCOMP folder within the Studio tree on Windows and the PlugIns folder within the Studio tree on macOS.

Updated doc:  (scroll to view update)

Debugging on macOS/Linux

Change the target modules command line for macOS: target modules add /Applications/Omnis\ Studio\ 11.app/Contents/MacOS/xcomp/myxcomp.u_xcomp/Contents/MacOS/myxcomp --symfile /Users/user/Documents/

OmnisSDKBuild/DebugSymbols/xcomp/myxcomp.u_xcomp.dSYM

/MacOS/PlugIns/myxcomp.u_xcomp/

Updated doc:  (scroll further to view update)

Extending Generic

After building the generic2 component close Omnis if it is still running, and move the component into your XCOMP folder.

your XCOMP or PlugIns folder

Updated doc:  (scroll to view update)

Chapter 14—DAM API Reference

Developer Guide

Building the DAM

Mac OSX

To debug, the DAM needs to be built into the Omnis.app/Contents/MacOS/xcomp folder.

Omnis.app/Contents/MacOS/PlugIns

Updated doc:

If the component fails to load on starting Omnis, you can verify the integrity of the component by navigating to

Omnis.app/Contents/MacOS/xcomp,

Omnis.app/Contents/MacOS/PlugIns,

Replace the screenshot showing the Contents folder.

Updated doc:

Note the resource files which should be copied into the component package during the build process. Localizable.strings and xcomp.rsrc are generated by the Omnis resource compiler from the .RC files. xcomp.rsrc in particular must be present in order for Omnis to recognise the package as an Omnis external component.

Localizable.strings is generated by the Omnis resource compiler from the .RC files. This file must be present for Omnis to recognise the package as an Omnis external component.

Updated doc: (scroll to view update)

Omnis Resource Compiler

Mac OSX

Replace the code text in the box:

gXcomp: 1
COCOA VERSION 1.0CountResources('OCTY') = 64

with:

gXcomp: 1
COCOA VERSION 2.0

Updated doc: (scroll further to view update)

Base classes

tqfDAMbaseObj

Remove these methods:

tqfDAMbaseObj::setEnvFile()
tqfDAMbaseObj::getEnvFile()

Updated doc:

JavaScript Component SDK

Tutorial

Building Generic

C++ Component

Mac

You then need to copy the built component (**jsgeneric.u_webdesign**) into the **jscomp** folder in the Omnis app package.
into the **PlugIns** folder

For example, if you want to place the component directly into your jscomp **PlugIns** folder, you need to set the Installation Directory to
"**/Applications/Omnis.app/Contents/MacOS/jscomp**".
"**/Applications/Omnis.app/Contents/MacOS/PlugIns**".

Updated doc: (scroll to Mac section)

Debugging

Debugging a Component

Mac

Set the **Debug** value to the full path to the **jscomp** folder in your Omnis app.

Set the **Debug** value to the full path to the **PlugIns** folder in your Omnis app.

Replace the screenshot.

Updated doc: (scroll to view update)