What’s New in Omnis Studio 12

Omnis Software
October 2025
76-102025-01

Table of Contents

About This Manual

This document describes the new features and enhancements in **Omnis Studio 12**. See the **Studio Now** tab in the Studio Browser for details of bug fixes in this revision.

See the **Install.txt** file to find out System Requirements for running the Development and Server versions of Omnis Studio 12.

NOTE: Where a new feature or an enhancement relates to an Enhancement Request or Customer reported fault, the fault reference 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 12.**

Library and Datafile Conversion

Converting 11.x Libraries

All Omnis Studio 11.X or earlier libraries need to be converted to run in Omnis Studio 12.X. ONCE A STUDIO 11.X or 10.X LIBRARY HAS BEEN OPENED WITH OMNIS STUDIO 12.X IT CANNOT BE OPENED WITH AN EARLIER VERSION – THE CONVERSION PROCESS IS IRREVERSIBLE.

Converting 8.x or earlier Libraries

ALL VERSIONS OF OMNIS STUDIO 12.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.*

Omnis Studio Licensing

The way Omnis Studio is licensed in version 12 has changed. Please see the section ‘Licensing and Activation‘ below, or for more information see the 'Licensing and Activation' document (pdf) accompanying this release, under the Studio NOW panel > Documentation section.

You will require a new software license to run Omnis Studio 12. Contact your local sales office to buy a license or obtain an upgrade license under your current support program; go to the Contacts page on the Omnis Studio website: www.omnis.net

oXML

Various third-party libraries used in the Omnis Studio oXML external component have been updated. (Revision 42271, ST/EC/1962)

- ❑ xerces (parsing and manipulating XML data) updated to 3.3.0
- ❑ ws (WebSocket library) updated to 8.18.3
- ❑ pbkdf2 (authentication) updated to 3.1.3

What's New in Omnis Studio 12 Revision 42449

The following enhancements have been added to **Omnis Studio 12 Revision 42449.** See the **Studio Now** panel in the Studio Browser for information about faults fixed in this release and links to the online docs.

Licensing and Activation

The way Omnis Studio is licensed in version 12 has changed, as we move from 'lifetime' license types to subscription-based licensing. Implementing online licensing and activation offers a range of benefits for you as developers when deploying your applications, including:

- ❑ License validation ensures only authorized users can use your software.
- ❑ Activation keys tied to specific devices or accounts reduce unauthorized distribution.
- ❑ Supports tracking of license usage, activations, and expirations.
- ❑ Allows deactivation or reallocation of licenses when needed.
- ❑ Helps enforce compliance with terms of service or usage limits.

When you first start Omnis Studio **Professional Edition** you no longer enter a serial number, rather you must **Activate** your software license to use Omnis Studio – the activation process requires that you have to be connected to the internet and create your own Omnis Studio ID (although offline activation is available in special cases where end user computers are never connected to the internet).

Similarly, activation is required for **Desktop User** licenses (these were called Runtime or Client licenses) and **App Server** deployment licenses, although activation 'tickets' can be generated and embedded into your deployment software to remove the need for end users to activate your software.

Software Development Licenses can be managed online in a new Licensing Dashboard at: dashboard.omnis.net, as well as Desktop and Server deployment licenses.

Please read the 'Omnis Studio Licensing and Activation' document (pdf) for further information about the new licensing and activation in Omnis Studio, which is available under the **Studio Now** > Documentation option in the Studio Browser.

JavaScript Remote Forms

Subform Palettes

Pointing a Subform Palette to a sub-element

You now have greater control over the target of a Subform Palette dialog when pointing at more complex controls, for example, a Data grid where you can point to a specific line or cell in the grid. (Revision 42175, ST/JS/3611)

For controls that contain sub-elements, such as lists or grids, the subform palette window can be positioned relative to a particular item or line within the control rather than to the control itself. This is achieved by adding a **selector** to the control name in **cControl,** separated by a colon (:), to determine the target sub-element or line (cControl is used in the *subformpaletteshow* client command to point the subform palette at a control).

For example, for a Data Grid control, the subform palette could point at a row in the grid with the selector comprising a rowSelector only, this being either a row number or a search term such as “Price=299.99". The corresponding cControl string would be something like: “DataGrid:3” to target row 3 of the grid, or “DataGrid:Price=299.99" to target the row whose Price value is “299.99”.

Additionally, the subform palette could point at a specific grid cell, with the selector being a data column number or name (the columnSelector), followed by a dot then the rowSelector. The corresponding cControl string would be something like: “DataGrid:Product.3” to target the product cell in row 3 of the grid, or “DataGrid:Product.Price=299.99" to target the product cell of the row whose Price value is “299.99”.

The **selector** syntax for each of the controls that have sub-elements is as follows:

- **Complex Grid**
  [targetSelector.]<rowSelector> where targetSelector is the name of a control within the complex grid, and rowSelector is a number or search term
- **Data Grid**
  [<column number or name>.]<rowSelector> where rowSelector is a number or search term
- **List**
  <rowSelector> which is a number or search term indicating the target line in the list
- **Native List**
  <rowSelector> which can be of the form:
  "5" to target row 5 of a flat list. or "g2:5" to target row 5 of group 2 in a grouped list.
  "myCol=Something" to target the first row where the 'myCol' column has a value of "Something".
  For grouped lists, use "g2:myCol=Something"
- **Segmented Bar**
  The selector is the number of the target segment
- **Tab Bar**
  The selector is the number of the target tab
- **Tile Grid**
  [targetSelector.]<rowSelector> where targetSelector is the string “button". If the targetSelector is absent the tile will be pointed at. If it is present, the button of the tile will be pointed at. The rowSelector is a number or search term
- **Toolbar**
  The selector is the number of the target toolbar item
- **Tree List**
  A Tree List allows for a subform palette to target an individual node provided it has

already been opened, otherwise the node element will not yet exist to point at. For a flat tree the selector is the ident, for a dynamic tree the selector is the node ident

Setting cControl

When using the **subformpaletteshow** client command to open a subform palette you can use "<controlName>:0" (zero) in the cControl parameter to point to the control as a whole, rather than an element within the control. (Revision 42300, ST/JS/3845)

Return Values Using a Promise

The promise returned from the **subformdialog** and **paletteshow** client commands, and when loading a subform, now returns an object with errorText and form members – this is potentially a breaking change, so you should review the code where you have used a promise. (Revision 42276, ST/JS/3838)

When the promise representing a subform loading resolves (the 'then()' function fires), either in the subformdialog and paletteshow client commands, or when assigning a subform control's $classname, it was previously passing an 'errorText' parameter.

However, it is now passed an object as its parameter, which has an 'errorText' member, and a 'form' member. The 'form' member is a reference to the form instance which has loaded. As before, if the 'errorText' member is populated, this indicates an error loading the form. For example:

```
Do $cinst.$clientcommand("subformdialogshow",row(...) Returns lPromise

JavaScript:lPromise.then( (result) => {
JavaScript:    lForm = result.form;
Do lForm.$myMethod()
JavaScript:});
```

Subform Sets

The 'subformSetParamsAreLocalized' item has been added to the Omnis Studio configuration file (config.json) to allow parameters passed to subform sets to use the server's localized function and decimal separators. (Revision 42312, ST/JS/3846)

There is a new item 'subformSetParamsAreLocalized' in the 'jsClient' group in the Omnis Studio configuration file. If true, parameters passed to Subform Set forms use the Omnis Studio server's localized parameter separator character to separate the parameters. For example, setting $language to Italian will set the parameter separator to ";" (colon), and decimal point character as "," (comma). If 'subformSetParamsAreLocalized' is false, commas are used as the separator, regardless of the localized parameter setting.

Due to this change in behavior, you should review any code that sets parameters passed to subform sets and where you may have set the localized parameter separator.

Note you can return the parameter separator using the sys(93) function, which you could use to embed into your list of parameters passed to subforms.

JavaScript Components

Data Grid Control

Drag and drop for data cells

The **$celldraganddrop** property has been added to the Data Grid control to allow you to drag and drop individual cells of data, rather than rows which was the case in previous versions. (Revision 42262, ST/JS/3772)

For Data Grids you can set $celldraganddrop to **kJSDataDragCells** to allow the data in a cell to be dragged out for the data grid, and dropped onto a list, for example. In this case, pDragValue will contain the data in the dragged cell, and not the whole row.

The Data grid also has the option **kJSDataDropCells** in the $celldraganddrop property. When this is set to true the current destination cell is highlighted instead of the row when data is dropped onto the data grid.

Drop event parameters

A parameter pDropColName has been added to drop events for Data Grids to indicate the data column name of the target column. (Revision 42332, ST/JS/3826)

The **pDropColName** event parameter has been added to the **evCanDrop, evWillDrop** and **evDrop** events for Data Grids to indicate the name of the list column on which the drop is to occur.

Drag and Drop

Drag data type

The pDragDataType parameter has been added to the evDrop and other drag and drop events to indicate the data type of the dragged data. (Revision 42269, ST/JS/3822)

The evDrag, evDrop, evWillDrop, and evCanDrop events have a new parameter **pDragDataType,** which is the Omnis Studio data type of the dragged data, e.g. kCharacter, kList, etc.

evDrag parameters

Some new parameters have been added to the evDrag event to provide more information about what data was dragged from list-based controls. (Revision 42281, ST/JS/3833)

The following parameters have been added to the **evDrag** event:

- **pDragRow**
  The row of the Complex Grid from which the drag occurs. Zero if the control does not belong to a complex grid.
- **pDragId**
  The identifier of the area of the control from which the drag has occurred. Either a line number or ident, or zero if the control is not list-based. In the case of a list-based control with multiple lines selected, the value is in the form of a list of line numbers.
- **pDragCol**
  Data grid only. The column number of the cell from which the data has been dragged. Zero for other controls.

Tree List Control

Icon Color

The **$::iconcolor** property has been added to the JS Tree List control allowing you to set the color of SVG icons shown on the nodes in the tree. When set to **kColorDefault** (the default), the icons are the same color as the node text (assuming the icons are themed using the SVG Themer tool). (Revision 42226, ST/JS/3787)

Icon Spacing

The **$icontextspacing** property has been added to the JS Tree control allowing you to set the spacing between the node icon and the text; the default is 4 pixels. (Revision 42360, ST/JS/3867)

Toolbar Control

The **$showoverflowicons** property has been added to the Toolbar control to allow you to display icons in the overflow menu items. (Revision 42211, ST/JS/3801)

When the $showoverflowicons property is set to kTrue (default is kFalse), the overflow menu items can include icons (icons were not shown in the overflow menu in previous versions).

Rich Text Editor

You can now insert tables when editing content using the Rich Text Editor. (Revision 42334, ST/JS/3843)

You can insert a table into the content edited in the Rich Text Editor using the new table button on the editor's toolbar. The button is hidden when opening libraries created in a previous version of Omnis Studio.

To support tables in the Rich Text Editor, some extra files have been added to the html folder, as follows:

```
html/scripts
   quill-table-better.js
html/css
   quill-table-better.css
```

Markdown Control

Scroll Shadows

The **$scrollshadowwidth** and **$scrollshadowcolor** properties have been added to the JS Markdown control allowing you to add a shadow to the control when there is more data to be scrolled – the shadow will appear on the relevant edge of the control to show that there is more content that can be scrolled, either in a vertical or horizontal direction. (Revision 42301, ST/JS/3836) This enhancement also applies to the Markdown window class control.

Scroll to position

The **$scrolltoposition** property has been added to the JS Markdown control which means the control will be scrolled to the specified position when new data is added; it can be set to one of the following constans: kJSMarkdownMaintainScroll (the default), kJSMarkdownScrollToTop, or kJSMarkdownScrollToBottom. (Revision 42359, ST/EC/1969)

This enhancement also applies to the Markdown window class control, where scrolltoposition is set in the $htmlcontroloptions property; in this case, negative number values indicate scrolling to the top, positive numbers to the bottom, and zero maintains the current scroll position.

Segmented Bar Control

The **$segmenttooltip** property has been added to the Segmented Bar control to allow you to add a tooltip to each segment. (Revision 42278, ST/JS/3840)

Label Object

The **$issingleline** property has been added to the Label Object. (Revision 42293, ST/JS/3844)

If $issingleline is true, the text will not wrap and will show ellipses if it does not fit in the control's bounds.

Libraries and Classes

Omnis Studio Data Types

A new item **numericCompareDps** has been added to the configuration file (config.json) which specifies the number of decimal digits that should be tested when comparing Number types. (Revision 42342, ST/PC/597)

For Number types 0..14dp and floating dp, Omnis Studio considers two values to be equal only if they have exactly the same fractional part (stored using 52 bits). There is a new entry in the config.json file; **numericCompareDps** which can be used to specify the number of decimal digits that should be tested when comparing Number types. Although zero (disabled) for backward compatibility, numericCompareDps accepts values in the range 0..19. For example, if numericCompareDps is set to 8, Omnis Studio will consider the following two values to be equal:

```
Calculate lNum1 as 3.14159265
Calculate lNum2 as #PI
Calculate #F as lNum2=lNum1
```

Omnis Studio Environment

User Components Library

You can now store your own components and wizard templates in a separate library and display them alongside the built-in components in the Component Store. In previous versions, your own components had to be added to the built-in Component Store library. Such user components will appear in the same groups as the built-in Component Store, or in a separate group of your own. (Revision 42162)

Your own components and wizards can now be stored in a **User Components Library,** which is an Omnis Studio library called usercomps.lbs, and placed in the Studio folder in the Omnis Studio tree. If such a library is present, Omnis Studio will load its components and wizards immediately after those in the built-in Component Store library – so the Component Store now displays the combined contents of comps.lbs and usercomps.lbs.

To add your own components, create a new library called **usercomps.lbs** in the Studio folder, alongside the existing comps.lbs. You must restart Omnis Studio before Omnis Studio will recognize this as a User Components library. Similarly, you must restart Omnis Studio after copying usercomps.lbs into the Studio folder.

To show the component libraries, select the **Show Component Library** option from the Component Store context menu (Right-click or Ctrl-click on its background). Showing and hiding the component library now shows both comps.lbs and usercomps.lbs (if it is present).

To create your own components in usercomps.lbs, you could copy existing components from the built-in comps.lbs into your usercomps.lbs, and amend those, to save time. For example, to do this for JS components, create a new Remote form in usercomps.lbs, open the remote form class JSFormComponents in comps.lbs, then drag an existing component from this remote form and drop it onto your new remote form in usercomps.lbs. Then adjust its properties as required including the name.

The type, name and Component store group of your own components need to be set. Classes and components in usercomps.lbs library have the **$componenttype** and **$componentinfo** properties, as in previous versions (note these properties are only visible if the usercomps.lbs has been loaded using the Show Component Library

option, not using the standard Open Library option). For further information about creating components and setting these properties, see the online docs: Editing the Component Store Library.

If there are duplicate components, those in usercomps take precedence, and replace those in comps.lbs. A duplicate in this case is considered as follows: for objects, such as window objects, a duplicate is an object with the same group and name (where case is not significant). For classes, new class default is a duplicate if the class type is the same; otherwise, wizards and templates are duplicated if the class name is the same. Note that wizards and templates in usercomps.lbs are identified in the browser Class Wizard panel by having the library name prefix "User Component Library".

Field styles in usercomps.lbs should have different names to those in comps.lbs – this only becomes an issue if a style in comps.lbs is different to one in usercomps.lbs, since when adding an object from the Component Store, Omnis Studio only copies the object's field style from the applicable component library into the destination library if the style is not present.

There is a new hidden notation property, $modes.$usercompstore (similar to $modes.$compstore) that returns an item reference to the User Component Store library, or it returns null if no user component store library is present.

A class marked as kCompStoreNewClassDefault can contain a $setup() method that is called as the final step in creating a new class. The first parameter of this method is an item reference to comps.lbs or usercomps.lbs, depending on the library containing the new class template.

As usercomps.lbs is a separate library, you can store and manage it in the Omnis Studio VCS, like any other Omnis Studio library. You can also export it to JSON, including the value of $componenttype (it exports kCompStoreHidden for all non-component libraries); this was not possible with the existing comps.lbs.

Keyboard Shortcuts

The following shortcut keys have been added. (Revision 42425, ST/HI/2069)

The custom menu item F11 on Windows and standard Window menu item Cmnd-` (backtick) on macOS have been added to allow you to cycle forward through open windows in Omnis Studio. The Shift modifier will cycle backward.

On macOS this menu item is standard and will show in the Finder. If the keyboard language is changed, e.g. to German then the short cut key will also change (Cmnd-<). This is likely to be the same key on the keyboard. The Shift modifier is not supported in this case since that key combination does not map to a valid character. Therefore, in some languages only cycle forward is supported.

Window Components

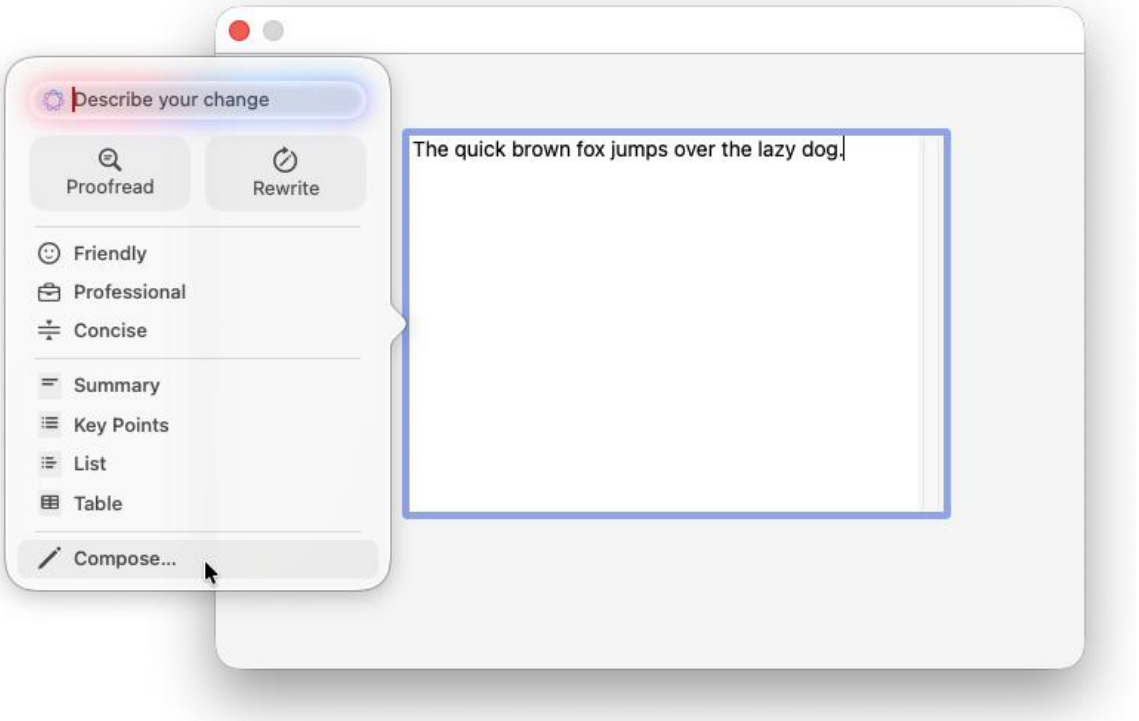
Writing Tools and Image Playground

On macOS, support for Apple's **Writing Tools** has been added to all Edit fields (Single line edit, Multi-line edit, Combo box, etc) to allow you to proofread, rewrite, and summarize text in the field, or to generate text using AI to insert into the field. In addition, **Image Playground** is available using a new method for AI-assisted image generation based on text prompts and descriptions. (Revision 42440, ST/HE/2130)

Apple Intelligence is required and must be enabled to use Writing Tools and Image Playground, which is only available on Apple Silicon (M) series computers running macOS Sequoia 15.1 or later.

Writing Tools

The Writing Tools are available on the main Omnis Studio Edit menu on macOS when entering text into a Single Line Edit Field, plus you can right-click (Cmnd-click) on the field to show the **Writing Tools** and access the **Services** menu. Selecting the **Show Writing Tools** option opens the tools popover allowing the end user to proofread or rewrite the text in the field.



The **Compose** menu option allows the end user to generate some text using ChatGPT based on a text prompt or question. Having inserted the generated text, you can edit it or refine it using the Writing Tools from the field context menu.

The **Services** menu option allows the selected text to be passed to an external service such as a Text editor. To enable items on the Services menu, use the Keyboard shortcuts in the main Settings panel.

You can disable the Writing Tools for the whole application by setting the **$writingtools** Omnis preference ($root.$prefs.$writingtools) to kFalse. To disable the Writing Tools for an individual edit field, set its **$disablewritingtools** property to kTrue.

Image Playground

Support for **Image Playground** on macOS allows you to add AI-assisted image generation into your applications.

The **$canuseimageplayground** Omnis preference ($root.$prefs) returns kTrue if Image Playground is available or kFalse if not. When Apple Intelligence is not supported or is turned off this will return kFalse.

The **$getplaygroundimage()** method allows you to generate an image using Image Playground and has the following parameters:

- $root.$getplaygroundimage(*cDescription,&iImageBinary[,&cFilePathURL*])
  Displays a standard system interface to generate an image from a description in *cDescription*. The image is returned as a PNG in *iImageBinary* with a temporary file URL in *cFilePathURL* if required.

The returned image can be inserted into a Picture field or stored in a file.

In addition, if Image Playground is available, when you right-click (Cmnd-click) on the background of a design window the **Image Playground** option will be shown, allowing you to generate an image which is added to the background of the window class.

OBrowser

Cursor Tracking

The **$enablecursortracking** property has been added to the OBrowser window component to allow cursor changes within browser content; affects macOS only. (Revision 42402, ST/EC/1971)

You can set the $enablecursortracking property to kTrue to allow embedded web content to control cursor updates via system cursor tracking. For example, to allow CSS managed cursor state in the browser. This defaults to kFalse to avoid conflicts with the Omnis-managed cursor state. Once enabled this will affect all cursor tracking within the same window.

Component Order

The **$disablecomponentorder** property has been added to the OBrowser window component to allow other window controls to be placed over it by preventing the OBrowser component coming to the front when it gets the focus. (Revision 42182, ST/WO/2532)

When set to kTrue, the $disablecomponentorder property disables any component order changes in the window. Therefore, the OBrowser component will not be brought to the front when it gets the focus, allowing any other components placed over the OBrowser component to stay on top. The property is kFalse by default to maintain the usual component ordering.

FishEye Control

Content Scrolling

The **Fisheye** control now allows content to be clipped to the window and for content to be scrolled using improved scroll support. (Revision 42230, ST/EC/1879)

When set to kTrue, the new **$cliptosize** property ensures that content is shown only within the dimensions of the window containing the control. The default is kFalse, which means large content areas could potentially be displayed beyond the window or screen border (which is the behavior in previous versions).

Content can now be scrolled to bring items into view. You can use the standard scroll gesture with a trackpad, use the left mouse button to click and drag with a mouse, or use the mouse wheel. You can set the new **$usewheeldelta** property to kTrue to increase granularity of mouse wheel scrolling (defaults to false). If using a mouse with a wheel on Windows then enable $usewheeldelta to make the content scroll according to the wheel delta, which will improve the granularity of the scrolling.

In addition, content in the FishEye control will now be visible and scrollable if not tracking when $magnifyall and the new property $magnifyallscroll are true. (Revision 42291, ST/EC/1963)

The $resetonresize property has also been added which when set to false will preserve the position of scrolled content when the field is resized.

SQL Classes

Table Instance Notation

The **$excludefromselect** property has been added to table instances. (Revision 42433, ST/TA/029)

When $excludefromselect is set to kTrue, the column is omitted from the result of $selectnames() (either using the table instance method, or using a DAM/session object method) and from results generated by $select(). When $fetch() is called, any excluded columns will contain NULL values. For example, the following will exclude all binary columns:

```
Do $cinst.$cols.$sendall($ref.$excludefromselect.$assign(kTrue),
   $ref.$coltype=kBinary)  ## excludes binary columns
```

OW3 Worker Objects

HTTP Worker

The HTTP Worker has a new property **$streamresponse** and **$ondata()** callback method allowing you to stream content (data) to the server rather than waiting for the whole response to be returned before calling $completed. (Revision 42354, ST/EC/1967)

$streamresponse

If $streamresponse is set to true, data received from the server will be sent through to the $ondata callback as it is received, rather than waiting for the whole response before calling $completed. If $streamresponse is true, $completed will still be called on completion, but data will no longer be sent in the responseContent column of $completed (to reduce memory usage).

$ondata

The $ondata callback method receives a row as its single parameter, which varies based on the Content-Type of the response. The row has the following columns:

- **httpStatusCode:** The HTTP status code of the response, usually 200 for a good response
- **httpStatusText:** Any status text sent with the response

The final column of the row depends on the Content Type of the response:

- **events:** If the Content-Type of the response indicates that this is a Server-Sent Event (SSE) stream (“text/event-stream”), the row will have an ‘events’ column of type List. This is a list of ‘event’ rows, where each column is a Character column matching the event ‘field’
- **ndjson:** If the Content-Type of the response indicates that this is a newline-delimited JSON response (“application/ndjson” or “application/jsonl”), the row will have an ‘ndjson’ column of type List. This is a list of ‘data’ rows, each row being an individual JSON object converted to an Omnis Studio row
- **content:** Any other Content-Types will be returned as raw binary data in a ‘content’ column

Important Considerations

As $ondata may be called many times in quick succession, so you should make sure to minimize the amount of work you do from this callback. If your execution takes longer than the time until the next $ondata call, these will start to back up.

Similarly, adding a breakpoint to an $ondata method will not prevent further calls to $ondata being added to the event queue. In this case, these calls will back up and could potentially overwhelm the event queue.

If you want to debug $ondata, or do a lot of work from it, it is recommended that you increase $minstreamdelay to a higher value so fewer calls to $ondata are made.

$minstreamdelay

Data can be returned from the connection very quickly, and if calls to $ondata back on the main thread were made too frequently, performance could suffer, or in extreme cases the Omnis Studio event queue could become overwhelmed, effectively hanging Omnis Studio.

The $minstreamdelay property specifies the minimum number of milliseconds to wait between calls to $ondata (the default is 100ms). Any data received during this time will be buffered and sent with the next $ondata call.

OAUTH2 Worker

Using OpenID tokens

The $openidtoken property has been added to the OAUTH2 Worker object allowing you to return the "openid" with your authorization. (Revision 42217, ST/EC/1958)

The **$openidtoken** property allows you to add another layer to authorization. If your authorization endpoint supports *OpenID Connect* (OIDC), and you include "openid" in the scope (along with other relevant scopes, such as 'email', 'profile', etc), the $openidtoken property will be populated with the OpenID token after calling $authorize(). The $openidtoken property is also saved and loaded automatically with the $save() and $load() methods.

The content in $openidtoken is the raw OpenID Connect *JSON Web Token* (JWT). It is then your responsibility to parse this and verify its signature, if it's important that it could not have been tampered with.

A JWT comprises the following components: payload, headers, and signature:

- **payload** is the content
- **headers** contain metadata about the token, and crucially, information about the way the signature was generated:
  'alg': the algorithm used to generate the signature, e.g. "RS256"
  'kid': (Key ID) the id of the public key you should use to verify the signature.

The provider will generally have some HTTP endpoint you can query to get the current keys, then you can find the one which matches the 'kid'.

You can use the $initverifysignature() method in the Hash Worker to verify the signature, once you have the appropriate public key.

SMTP Worker

When sending emails with the OW3 SMTP Worker, the Body Part Headers can now be provided in the vContent parameter MIMEList in the $init() method. (Revision 42174, ST/EC/1952)

The MIMEList sent as vContent in the $init() method for the OW3 SMTP Worker now supports a tenth column bodypartheaders of type Row, representing the Body Part Headers. Each column in this row will be added as a header, whose name matches the column name, and the value matches the column value (columns should be of type Character). This is the same as the tenth column returned from OW3.$splitmultipartdata().

This allows you to set any header, such as "Content-Id" or "Content-Description". If you set the Content-Id, bear in mind that it should be of the form "<local part>@<domain part>". A common convention is just to use "cid" as the domain, e.g. "myfile_id@cid".

OW3.$createmimemessage()

The $createmimemessage() function has been added to the OW3 package to allow you to create a raw MIME email message to then send, for example, via a third-party API such as provided by GMail. (Revision 42202, ST/EC/1955)

Function group	Execute on client	Platform(s)
OW3	NO	All

Syntax

OW3.$createmimemessage(*vFrom,lTo,lCc,lBcc,cSubject,iPriority,lHeaders,vContent*)
Returns lBinaryData

Description

Creates a raw MIME email message from given input. Returns Binary data. The parameters are as follows (the same as the SMTP Worker's $init() method, minus those for the SMTP connection/credentials):

Parameter	Description
vFrom	The email address of the message sender. Either a character value e.g. user@test.com or a row with 2 columns where column 1 is the email address e.g. user@test.com and column 2 is descriptive text for the sender, typically their name
lTo	A one or two column list where each row identifies a primary recipient of the message. Column 1 contains the email address e.g. user@test.com and column 2 if present contains descriptive text for the recipient, typically their name
lCc	Empty if there are no CC recipients, or a one or two column list where each row identifies a carbon copied recipient of the message. Column 1 contains the email address e.g. user@test.com and column 2 if present contains descriptive text for the recipient, typically their name
lBcc	Empty if there are no BCC recipients, or a single column list where each row contains the email address of a blind carbon copied recipient of the message e.g. user@test.com. Unlike lTo and lCc, lBcc does not allow more than 1 column, as blind carbon copied recipients are not added to the message header and therefore the descriptive text is not required
cSubject	The subject of the message
iPriority	A kOW3msgPriority... constant that specifies the priority of the message
lHeaders	A two-column list where each row is an additional SMTP header to send with the message. Column 1 is the header name e.g. 'X-OriginalArrivalTime' and column 2 is the header value e.g. ’23:02'
vContent	Message content. Either binary raw content, or a list to be sent as MIME. See the documentation for the MailSplit command to see how a MIME list is structured; however note that the charset in the OW3 MIME list is a kUniType... constant

Example

When using or testing this function, you may like to use the RESTful API provided by Gmail, which requires you to send raw MIME content. Once you've authorized with the oauth2 worker, you can attach the oauth2 worker to a HTTP worker and send the email message using something like the following:

```
Do lMimeList.$define(lLevel, lContentType, lContentSubType, lFileName,
   lCharData, lBinData, lCharSet, lEncoding, lContentDisposition,
   lBodyPartHeaders)
Do lMimeList.$add(0,'text','plain',,lText,,"utf-8","base64",)


Do lToList.$define("")
Do lToList.$add("user@gmail.com")


Do OW3.$createmimemessage("user@gmail.com", lToList,,,"My test email",
   kOW3msgPriorityHigh, lHeaders, lMimeList) Returns lBinData


Do OXML.$base64encode(lBinData,lErr) Returns lB64
Do bintobase64(lBinData,kTrue,kFalse) Returns lB64
Do lContentRow.$define()
Do lContentRow.$cols.$add("raw",kCharacter,kSimplechar)
Calculate lContentRow.raw as utf8tochar(lB64)


Do
   iHTTP.$init("https://gmail.googleapis.com/gmail/v1/users/me/messages/send",k
   OW3httpMethodPost,,lContentRow) Returns #F
Do iHTTP.$start()
```

Omnis Studio VCS

Time Zones (Postgres)

If you are using Postgres as your VCS repository, and you have set the **$usetimezone** session property, you can no longer set your time zone in the **VCS Options > Display Options** window. In this case, times are returned from the server with no further conversion. (Revision 42200, ST/VC/845 & ST/VC/847)

VCS API

The $unlockClass method has been added to the VCS API. (Revision 42321, ST/VC/838)

Unlock Class

The **$unlockClass** method unlocks a class or list of classes in the specified library.

```
Do iAPIObjRef.$unlockClass(cProject,lClassList,rLibRef,cToken,cErrors) Returns
   bStatus
```

cProject is the project name. **lClassList** is a single column list of the classes you want to unlock. **rLibRef** is a reference to the open library that holds the classes which are shown as locked.

Functions

OIMAGE.$crop()

The OIMAGE.$crop() function has been added to allow you to crop an image. (Revision 42422, ST/FU/945)

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

Syntax

OIMAGE.$crop(*xImage, iX, iY, iWidth, iHeight, &xNewImage* [, *wParams = #NULL, &cErrorText*])

Description

Crops *xImage* to *iWidth* by *iHeight* at *iX* and *iY*. Returns Boolean true and *xNewImage* on success, or false and *cErrorText* if an error occurs.

wParams is an optional row variable of parameters. The following columns can be specified in *wParams*:

- **sampler**
  The sampling method to be used when cropping. Either kOIMAGEsamplerBilinear or kOIMAGEsamplerNearestNeighbour. Defaults to kOIMAGEsamplerBilinear if this column is not present, or if wParams is omitted.
- **gray**
  A Boolean that indicates if the new image is to be a grayscale image. Defaults to kFalse if this column is not present, or if wParams is omitted.
- **quality**
  If the input image type is JPEG this column contains the JPEG image quality (1 to 100) of the new image. Defaults to 80 if this column is not present, or if wParams is omitted.

sys()

sys(295) has been added and returns the pathname of the folder containing non-executable support folders and files, e.g. htmlcontrols and iconsets. (Revision 42365, ST/FU/941)

join()

You can now use a column name as a parameter for the *join()* function, as well as the column number, as in previous versions. (Revision 42245, ST/FU/930)