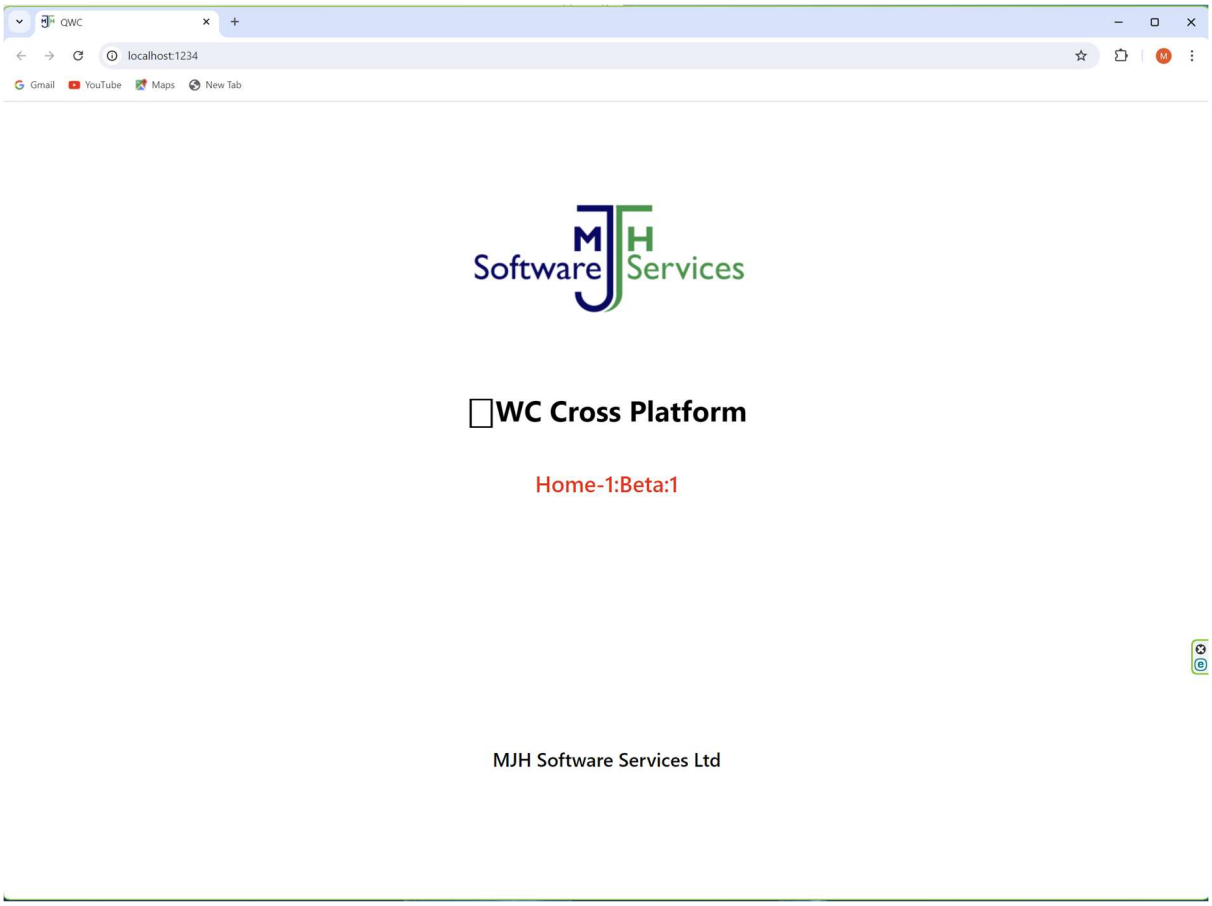

QWC
localhost:1234
Gmail
YouTube
Maps
New Tab
Software
MJH
Services
WC Cross Platform
Home-1:Beta:1
MJH Software Services Ltd

Introduction

□WC has been the staple GUI for Dyalog APL applications running under Windows for many years. Although □WC is a superb product in its own right, it suffers from six major short comings;

1. It only runs on the Windows version of Dyalog APL.
2. It requires Dyalog APL and the application using □WC to be on the same machine.
3. It is dependent on Win32 which is legacy technology.
4. Applying this to the web or cloud is becoming increasingly difficult. Its no longer easy to display your results or receive User input in the new world of distributed computing.
5. It is strictly single user, with no co-operation possible.
6. Does not allow the addition of User-defined or any User adaptation of □WC controls.

qWC addresses all of these issues in a simple and easy to use way. It aims to allow almost any □WC application to run across any operating system which supports Dyalog APL and to display on any modern standalone web browser or HTML Renderer, including Dyalog's. Dyalog APL is not needed on the display machine. This decoupling allows the major step forward towards true distributed cloud computing.

Release 1 Beta 2 is still not a complete version of the □WC objects but the qWC architecture is now well established and proven. It has also been confirmed that every non-Windows specific object can be replicated by qWC.

The gaps in support will now be back filled, the order and speed depending on need and feedback. New objects will continue to be added, again depending on need and feedback. New objects already added are; a SideBar to the Session, Form or Group, Charts, File Uploader, External Events (to feed events from external systems into qWC) with Ribbon and Data Manager (link between qWC controls and SQL databases) still to be completed for the next beta release.

Some current properties have been extended to ease or extend their use, such as BCol and FCol, which have been extend to allow the use of CCS colours like 'Red' or 'Blue' instead of (255 0 0) and (0 0 255) while maintaining backwards compatibility.

qWC consists of two parts;

1. main queue process or framework which is long established and proven.
2. a separate definition for each control, allowing a User easy addition of or replacement by their own controls.

The control definitions consist of five simple text files of a fixed structure – meaning any JavaScript Widget or Blazor (coming with Release 2) can be added to the qWC system. It will therefore be possible for Users to write these definitions to include these new controls or to replace others fully or partially as they wish.

The future is already moving fast towards Web Assembly or WASM, MJH Software are sure anything written in JavaScript will soon be falling behind and moving to legacy status. Web Assembly runs many times faster, is more robust and more secure.

The next full release of qWC will be a Web Assembly version as MJH Software believe this is very much the future and will ensure support for qWC (and so □WC) continues into the future of modern cloud applications. MJH Software has already tested a prototype of qWC running using Web Assembly. It plans on making this release available in early 2025. The Web Assembly version will be able to run both native Blazor controls and legacy Javascript controls giving a smooth seamless transition towards a full Web Assemby version.

qWC requires no installation at the display end.

qWC installs as a single directory and one or two standard Dyalog User command files, depending on the version of qWC being used.

This makes qWC very simple to install and use. The use is identical to □WC either in using the qW*, qDQ and qNEW functions or the GUI namespaces directly.

Everything is controlled by User commands and standard Dyalog .dcfg configuration files. qWC comes in three versions, with two modes;

1. Home (probably to be renamed Standard).
   This is the simplest and allows one user to use one instance of qWC. This behaves exactly like □WC now, with both qWC access and direct namespace manipulation. It allows the APL user to get the same □WC experience on any Operating system supporting Dyalog APL.
2. Professional. This is the next level of sophistication, and provides one queue for processing multiple qWC sessions with a Web Server in front. The web server can be Jarvis, the NGINX container provided (or a custom web server of the User's choice).
3. Enterprise. This is the scaling solution which extends to multiple queues, supporting an almost unlimited number of qWC sessions. All behind a single web address allowing the NGINX web server container provided (or similar according to taste) to load balance the various queues and sessions behind the web server. Queues can be started, stopped and managed depending on load without interrupting use. All controlled with User commands. Including soft management of servers.
   It is intended to place these User Commands behind a control panel using qWC.
   to automate the management of load by soft updating the web server from the APL sessions themselves.

Each version of qWC will come in two modes, Client enabled (non re-entrant) and Client not enabled (re-entrant). The "Client not enabled" mode will still be able to use "Client enabled" access in a similar manner to the current Ride with a Dyalog session for debugging, etc.

Home/Standard is client enabled by default.
Pro/Enterprise will be client not enabled by default.

Client enabled separates the queue into a queue and client process (a workspace as now with □WC). Where the client process is a normal APL workspace containing the application and qWC functions. It allows direct access to the GUI namespaces. It runs exactly the same as □WC so gives applications an extremely easy migration to qWC.

Client not enabled is intended for new lighter weight, re-entrant applications, where direct access to the GUI namespaces is not required. Namespaces can still be used indirectly via an access function but cannot be based on the fixed base #, as all globals prevent a system being re-entrant. Accessing a GUI namespace #.f.etc is not-rentrant so can only safely be used with a client enabled system. The exception is Home/Standard which only supports a single user (as does □WC).

All versions will support both modes, but it is envisaged the defaults are likely to be adhered to except where it is too difficult to convert a □WC system to re-entrant.

Its important to note that all types of □WC and new non-□WC usage has been considered and nothing is precluded.

Installation

Every qWC version has the same installation procedure and they can live alongside each other.

Download the qWC system from; *needs address*
The application consists of

A single directory containing the startup workspace which is the self-referencing to its containing directory. This directory is normally called Output, an historical name as it is the output from the qWC compiler. This directory can be renamed from Output and placed anywhere accessible to Dyalog APL.
It is best practice to place it on an easily accessible SSD drive for efficiency.
Any changes to the name and Output directory location will need to be reflected in the config files described later in this section.

One or two User Command files, according to version.

QWC.dyalog
GS.dyalog

These should be placed according to the User Command file locations configured for your Dyalog APL installation. See Dyalog's User Command manual for details.

Once downloaded, and assuming the directory is kept as the default "c:\Dyalog\Output" and config files are unchanged then qWC Home/Standard can be started in a Dyalog APL session with;

```
]QWCInit
```

This starts;

1. the client workspace
2. the queue workspace (which should probably be an APL service but is by default an ordinary APL session in a workspace.
3. The browser/HTML renderer of choice

If other options are required – the config files listed in the next section should be reviewed before trying]QWCInit.

Configuration files

General

There are at two configuration files for all versions.

QWC.dcfg stored in the directory with user command file QWC.dyalog.

Config.dcfg stored in the Output directory as <Output>\Config\Config.dcfg.

If the Enterprise version of qWC is in use then GS.dcfg is stored in the directory with the User Command file GS.dyalog.

All config entries are optional, the values listed are those assumed if the config entry is omitted.

QWC.dcfg (for QWC.dyalog)

```
{
 Output:              “c:/Dyalog/Cmd/QWC/Output”,
 Connection:          “localhost”,
 Port                 1234   (0 means no port)
 Secure               0      (0 = use Http://  1= use Https://)
 Client:              “Yes”,
 QStyle:              “Min”,

 Only applicable to Home, ignored by others
 DefaultBrowser:      "Chrome",
 ClientPort:          1236,
}
```

The url to see the display from qWC built from the defaults above would be Http://localhost:1234. This is loaded into the browser with]QWCINIT by the home version.

```
 ]QWCInit   [H|P|E] [Output=<Output>] [URL=<Connection>] [Client=<Client>]
            [QStyle=<QStyle>] [DefaultBrowser=<DefaultBrowser>]
            [ClientPort=<ClientPort>]
```

The user command options take precedence over the configuration file values.

Output	Base location of the qWC system
DataDir	Sub-directory below Output and contains Application data
UserDir	Sub-directory below Output and contains User data. Any dyalog workspace or directory *.lws (Link directory) which is copied in when qWC is started. Once all the User application is loaded, if there is an
Only applicable to Home, ignored by others	

DefaultBrowser	Browser started by]QWCInit for Home edition Currently one of Chrome, Edge, Firefox, Local (HTMLRenderer) – others can be added on request
ClientPort	Port the client server connects
QStyle	Whether the Queue workspace is Minimized, Maximized, etc. Default is minimized

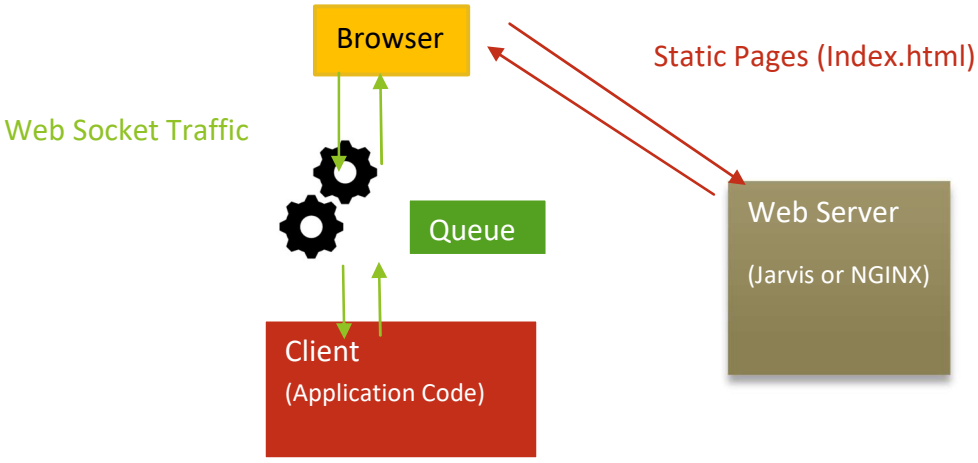
Config.dcfg (for QWC application)

```
{
 Output:          "c:/Dyalog/Cmd/QWC/Output"   Output directory
 Connection:      “localhost”,                 url
 Client:          “Yes”,                       Client mode enabled
 QStyle:          “Min”,                       Style of Queue window
 Port             12345,                       Queue port
 DefaultBrowser:  "Chrome",
 ClientPort:      1236,                        Client Workspace Port
 UseZip           0                            Zip or don’t zip connection
 UserDir          User/                        User directory name, where
                                               application code is held
 UseJarvis        1                            Use Jarvis as Web Server
 UseZipJarvis     0                            Does Jarvis zip data
 JarvisPort       1234                         Port used by Jarvis
 UseRest          1                            Use Jarvis server in Rest mode
 RestPort         1235                         Port used by Rest server
 RestUseZip       0                            Does Rest Server zip data
}
```

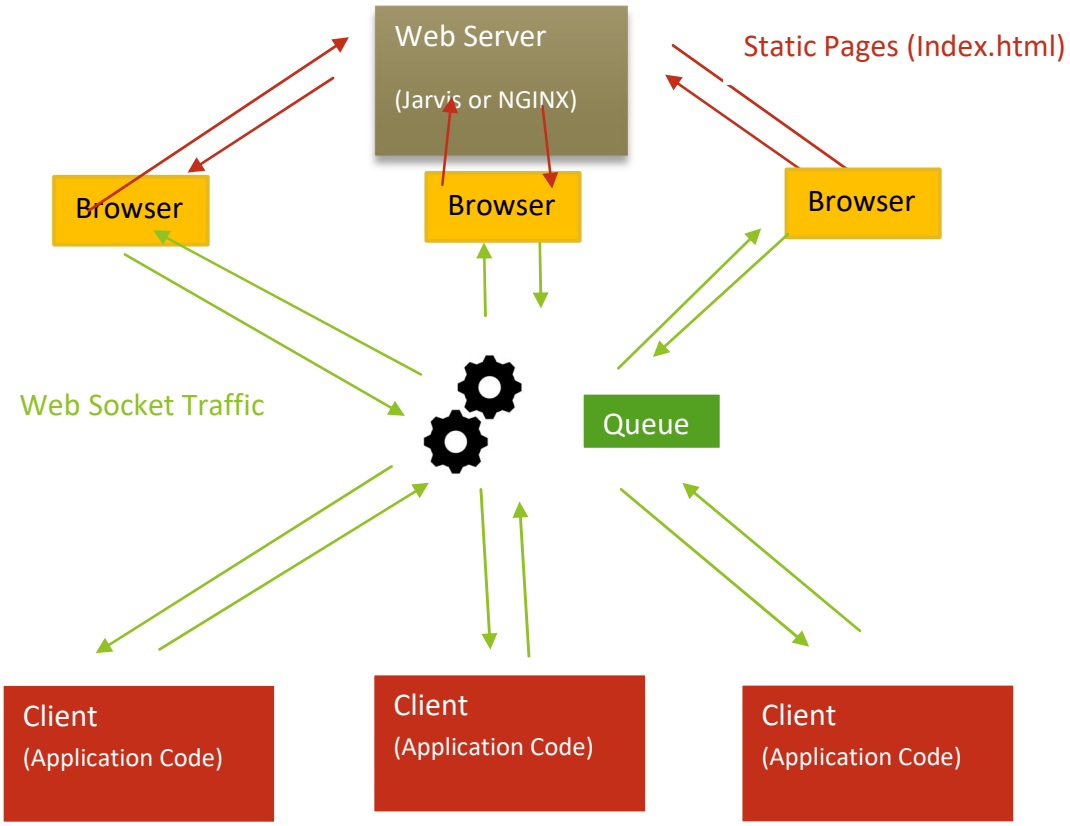
GS.dcfg (for Global Server, used by Enterprise version)

```
{
}
```

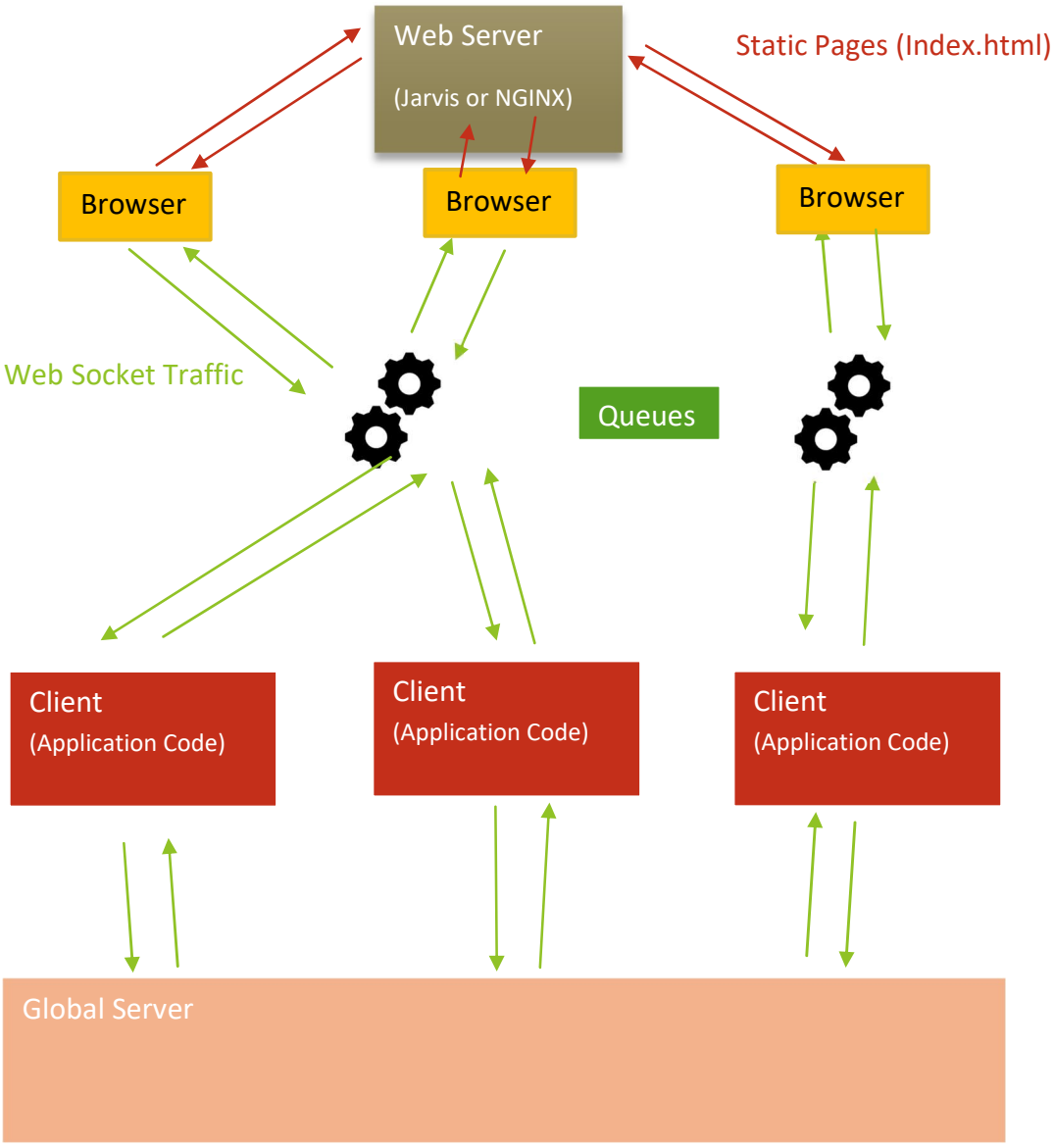
Architecture
Home



Pro


Web Server
(Jarvis or NGINX)
Static Pages (Index.html)
Browser
Browser
Browser
Web Socket Traffic
Queue
Client
(Application Code)
Client
(Application Code)
Client
(Application Code)

Enterprise


Web Server
(Jarvis or NGINX)
Static Pages (Index.html)
Browser
Browser
Browser
Web Socket Traffic
Queues
Client
(Application Code)
Client
(Application Code)
Client
(Application Code)
Global Server

Release Contents and Scope

The system is designed to match the ⎕WC system as closely as possible.

Currently only two minor differences exist between qWC and ⎕WC, both are due to qWC being a set of normal APL functions grounded in namespace # while the ⎕WC functions are system quad functions not grounded in any namespace. This difference cannot be changed but there is a work around.

1. #qWC is the only capitalization allowed, unlike both ⎕WC or ⎕wc being valid. Thos can be changed by the user by assigning #.qWC, etc. to functions of required capitalization but this isn't really recommended.

2. #.namesp1.namesp2.qWC does not work as the code cannot be located in any namespace other than #. This can be emulated with a qWC operator called #.qWith as in

   #.qWC #.qWith #.namesp1.namesp2.

Great care has been taken to enable the migration of a ⎕WC system on Dyalog APL for Windows to any platform supporting Dyalog APL. The display can now be separated from the application providing they are linked via a network or the internet. The display can be any modern HTML renderer or browser.

⎕W* fns

Every function below is supported and should behave as described by the Dyalog reference manuals.

⎕WC	#.qWC	Windows Create
⎕WG	#.qWG	Windows Get
⎕WS	#.qWS	Windows Set
⎕NQ	#.qNQ	Enqueue method or event
⎕WN	#.qWN	Names of the Children
⎕WD	#.qWD	Delete Windows objects
⎕DQ	#.qDQ	DeQueue Windows message queue
⎕New	#.qNEW	Behaves like ⎕New as far as GUI objects are concerned

:With

These functions all work as described – (see example Ex1).
They all work with the :With control structure (see example Ex1w).

#.qWith operator

So 'fm' A.BBB.⎕WC 'Form' should be replaced with 'fm' #.qWC 'Form' #.qWith A.BBB.
qWC provides a utility to scan workspaces to automate this change.

Additionally, when the mode is set to Client Enabled = "Yes" then #.qNEW also works as expected to create GUI namespaces.

⎕NEW	#.qNEW	Creates a new object (similar to #.qWC but without instance name)

#.qInsertEvent
Onloaded
OnExternal
#.qFor

Supported □WC features
Supported properties/methods across most □WC/qWC objects

Data	Store any APL data object
Children	Children of the current object
Posn	Standard Posn of an object
Size	Standard Size of an object
Coord	Coordinates of the object, the inheritance of the value is inherited as it is in □WC. Pixel and Prop are both supported. The others are not but are a simple add. They will be if requested.
Event	Setting event handlers

Root (#) properties/methods

Caption	Sets or retrieves the text on the tab of the browser document
ScreenSize	Size of the browser client area
DateToIDN	Converts date to Date number
IDNToDate	Converts Date Number to Date

Button properties/methods
Properties

ChildList	Parent object name
Coord	
Name	Current object name
Parent	

Buttons with Style property set to Check, CommandLink, Push, Radio, Split, Toggle,
Calendar properties/methods

Combo

DatePicker

DateTimePicker

Edit

Font

Form

Grid

Group

Image

Label

List

ListView

Menu

MenuItem

MenuBar

MsgBox

Ribbon
Separator

SubForm

TabButton

TabButton1

TabControl

Text

TreeView

New features not supported by □WC

Root (#) properties/methods

FontList	List of fonts
FontFamily	Font Family lists
FullScreenSize	Size of the entire screen when if full screen mode
FullScreenId	Name of the object in Full Screen mode, should be the Form instance name
URL	Full URL used to start the browser session
URLS	List of URLs for the other tabs
Captions	List of Captions for the other tabs
Colors	Names of the CCS colours supported by BCOL and other colour properties
ColDefs	# colour definitions aligned with Colors
SetCaption(n,caption)	Sets the text on the n^{th} tab to caption
GetCaption(n)	Gets the text on the n^{th} tab
SetURL(n,url)	Sets the URL on the n^{th} tab to url
GetURL(n)	Gets the URL on the n^{th} tab
Focus(n)	Puts focus on nth tab
SendHTML(n,text)	Sends text (html) to nth tab
NewTab()	
CloseTab(n)	Closes nth tab

Semi or almost supported □WC features
Supported properties/methods across most □WC/qWC objects

Data	Store any APL data object
Children	Children of the current object
Posn	Standard Posn of an object
FontObj	The FontObj is supported and inherited as it is in □WC but has not been fully connected to the objects. So settings made to the FontObj aren't reflected. So Font and FontSize properties aren't honoured. This will hopefully be corrected fairly soon.

Clipboard
A partial implementation of the clipboard package
This still needs work.
It will be completed and described later, time permitting.

New objects not supported by □WC
BridgeScore
A specialized control for playing bridge.
This is a fairly easy self explanatory control built from other qWC components.
It will be described later, time permitting.
Charts
A partial implementation of a charting package
This still needs work.
It will be completed and described later, time permitting.
CircularGuage

ColorPicker

Dashboard
A partial implementation of the clipboard package
This still needs work.
It will be completed and described later, time permitting.
DataManager

Expander

LinearGuage

Ribbon
SideBar

Uploader

Structure of a qWC object
qWC consists of several components.
A Web Server for serving up static data and web files
(can be Jarvis or NGINX contained supplied)
(can be an alternative favoured by the User)
A Load Balancer (NGINX container supplied)
or can be an alternative preferred by the User
One or more Queue processing servers
One or more client workspaces
The four or three features (the Home/Standard edition doesn't require a load balancer, and NGINX can combine the Web Server with the load balancing) make up the qWC framework. This framework then runs and controls the qWC objects. Currently these are either native controls built using Xaml and WPF or are taken from the Syncfusion library of EJ2 JavaScript controls.
However, **ANY** JavaScript widget will work, with Blazor widgets added from Release 2.
Whatever the source of the underlying control, the structure is the same. This allows Users to design and include their own new controls or to replace any of the pre-supplied controls.

There are 5 simple files that make up a qWC control. The Grid is probably the most complex control in qWC but has been added to qWC using the EJ2 JavaScript Spreadsheet control. The files below will show how easy it is to add new controls to qWC.

Files structure

qwcGrid.txt Class and Property definitions

```
:Class qwcGrid_Data M                                                    1
:include qwcCommon                                                       2
// Main structure – property definitions                                 3
// Name, data type, initial value, mode                                  3
PropList   String[]  "'Type' 'Caption' ,                                 4
             'Posn'  'Size' 'Style'   "  C
FontObj   string     ""          IB                                      5
Type       String    Button    C                                        5
Margin     Margin    "0 0 0 0"  B                                       5
Height     Double    25                                                 5
```

See Appendix 1 for full file

```
:Event Select Click                                                      6
:BasedOn qwcNewObj APLNewObj ds.Instance                                 7
Caption Content                                                          8
BCol  Background "" "" APLColorConverter                                 9
FCol Foreground "" "" APLColConverter                                    9
:NQ FontChanged                                                          10
```

1	M = Instance class, S = Static, visible by all clients
2	Includes another *.txt file Can have multiple entries
3	// starts a comment
4	Property Name = PropList (shortened example, usually more entries) Order determines how qWC decides order of non named properties. Data Type is vector of strings " only needed if string contains spaces Line can be continued with , C=Mode. Details of Mode and Data type at the end.

```
FontObj   string     ""          IB
Type       String    Button    C
Margin     Margin    "0 0 0 0"  B
Height     Double    25
```

Etc – examples can be provided for full set

:Event Select Click Maps qWC event Select to internal Click
:BasedOn qwcNewObj APLNewObj ds.Instance
  States NewObj is based on the Class
  qwcNewObj based on APLNewObj
Caption Content
BCol Background “” “”
APLColorConverter
FCol Foreground “” “” APLColConverter
:NQ FontChanged

APLGrid.xaml Base object (can be shared)
Standard WPF Xaml definition

```
<GButton x:Class="APLControls.APLGrid"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Visibility="{Binding Visibility}" HorizontalAlignment="Left"
VerticalAlignment="Top"
        Height="{Binding Height}" Width="{Binding Width}"  Canvas.ZIndex="{Binding
ZIndex}"
        ClipToBounds="true" Opacity="{Binding Opacity}"
/>
```

APLNewObj.xaml.cs Base object

```
using System;
using System.Collections.ObjectModel;
using CSHTML5;
using System.Collections.Generic;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Data;
using APLExtension;
using Windows.System;
using System.Reflection;
//
namespace APLControls
{
   public partial class APLButton : Button
   {
      public string InstanceName { get; set; }
      public string Id {get; set; }
```

```
      public string SerialNo { get; set; }
      public string APLStyle { get; set; } = "";
      public string APLDataInstance { get; set; } = "";
      public string APLClassInstance { get; set; } = "";
      public string GridText { get; set; } = "";
      public bool InGrid { get; set; } = false;
      public object PopupParent { get; set; }

      public APLButton()
      {
          RunInit("");
      }
      public APLButton(string instance)
      {
          RunInit(instance);
      }
      public void RunInit(string instance)
      {
         InitializeComponent();
         InstanceName = instance;
      }
  }
}
```

qwc<NewObj>_Insert.txt

```
      public void Init(string ns, qwcButtonPush_Data instance, TransferString ds)
      {
         Button btn = (Button)instance.MJH_Self;
         btn.HorizontalAlignment = HorizontalAlignment.Left;
         btn.VerticalAlignment = VerticalAlignment.Top;
         btn.ClipToBounds = true;
      }

       private void AddAsChild(string pcl, string parent, qwcButtonPush_Data instance)
       {
           if (pcl == "qwcForm_Data")
           {
               qwcForm_Data pinst = (qwcForm_Data)APL.GetInstance("DataBinding",
"qwcForm_Data", parent, this);
            APLForm par = (APLForm)pinst.MJH_Self;
            Grid g = (Grid)par.Content;
                                        Button btn = (Button)instance.MJH_Self;
            g.Children.Add(btn);
                  pinst.Children.Add(instance.MJH_Instance);
            pinst.NotifyPropertyChanged("Children");
         }
         else if (pcl == "qwcExpander_Data")
         {
```

```
            qwcExpander_Data pinst = (qwcExpander_Data)APL.GetInstance("DataBinding",
"qwcExpander_Data", parent, this);
            Expander par = (Expander)pinst.MJH_Self;
            Button btn = (Button)instance.MJH_Self;
            StackPanel sp = (StackPanel)par.Content;
            sp.Children.Add(btn);
                  pinst.Children.Add(instance.MJH_Instance);
            pinst.NotifyPropertyChanged("Children");
         }
         else if (pcl == "qwcGroup_Data")
         {
            Button btn = (Button)instance.MJH_Self;
            qwcGroup_Data pinst = (qwcGroup_Data)APL.GetInstance("DataBinding",
"qwcGroup_Data", parent, this);
            APLGroup par = (APLGroup)pinst.MJH_Self;
            Grid g = (Grid)par.Children[5];
            g.Children.Add(btn);
                  pinst.Children.Add(instance.MJH_Instance);
            pinst.NotifyPropertyChanged("Children");
         }
         else if (pcl == "qwcGrid_Data")
         {
            qwcGrid_Data pinst = (qwcGrid_Data)APL.GetInstance("DataBinding",
"qwcGrid_Data", parent, this);
            APLGrid par = (APLGrid)pinst.MJH_Self;
            Button btn = (Button)instance.MJH_Self;
            Grid g = (Grid)par.Inputs;
            g.Children.Add(btn);
                  pinst.Children.Add(instance.MJH_Instance);
            pinst.NotifyPropertyChanged("Children");
         }
         else if (pcl == "qwcSubForm_Data")
         {
            qwcSubForm_Data pinst = (qwcSubForm_Data)APL.GetInstance("DataBinding",
"qwcSubForm_Data", parent, this);
            APLForm par = (APLForm)pinst.MJH_Self;
                                    Grid g = (Grid)par.Content;
            Button btn = (Button)instance.MJH_Self;
            g.Children.Add(btn);
            pinst.Children.Add(instance.MJH_Instance);
            pinst.NotifyPropertyChanged("Children");
                              } else if (pcl == "qwcSideBar_Data")
         {
            qwcSideBar_Data pinst = (qwcSideBar_Data)APL.GetInstance("DataBinding",
"qwcSideBar_Data", parent, this);
            if (!pinst.Children.Contains(instance.MJH_Instance))
                                    {
```

```
                                        Grid g = (Grid)APL.GetObjectByName("QWCGrid");
                                        Button btn = (Button)instance.MJH_Self;
                                        g.Children.Add(btn);
                                        pinst.Children.Add(instance.MJH_Instance);
                                        pinst.NotifyPropertyChanged("Children");
                                    }
                                } else throw (new System.ArgumentException("Invalid type <" + pcl +
"> as parent to qwcButtonPush"));
        }

        private void RemoveChild(string pcl, string parent, TransferString ds)
        {
            qwcButtonPush_Data instance =
(qwcButtonPush_Data)APL.GetInstance(ds.SerialNo);
            if (null != instance)
            {
                if (pcl == "qwcForm_Data")
                {
                    qwcForm_Data pinst = (qwcForm_Data)APL.GetInstance("DataBinding",
"qwcForm_Data", parent, this);
                    APLForm par = (APLForm)pinst.MJH_Self;
                    Grid g = (Grid)par.Content;
                    Button btn = (Button)instance.MJH_Self;
                    if (null != g && g.Children.Contains(btn)) g.Children.Remove(btn);
                    if (pinst.Children.Contains(instance.MJH_Instance)) {
                                                pinst.Children.Remove(instance.MJH_Instance);
                                                pinst.NotifyPropertyChanged("Children");
                    }
                                        APL.DeleteInstance(instance.MJH_Instance,
instance.SerialNo);
                } else if (pcl == "qwcExpander_Data")
                {
                    qwcExpander_Data pinst = (qwcExpander_Data)APL.GetInstance("DataBinding",
"qwcExpander_Data", parent, this);
                    Expander par = (Expander)pinst.MJH_Self;
                    Button btn = (Button)instance.MJH_Self;
                    if (null != par)
                    {
                        StackPanel sp = (StackPanel)par.Content;
                        if (sp.Children.Contains(btn)) sp.Children.Remove(btn);
                    }
                    if (pinst.Children.Contains(instance.MJH_Instance)) {
                                                pinst.Children.Remove(instance.MJH_Instance);
                                                pinst.NotifyPropertyChanged("Children");
                    }
                                        APL.DeleteInstance(instance.MJH_Instance,
instance.SerialNo);
```

```
        } else if (pcl == "qwcGroup_Data")
        {
            Button btn = (Button)instance.MJH_Self;
            qwcGroup_Data pinst = (qwcGroup_Data)APL.GetInstance("DataBinding",
"qwcGroup_Data", parent, this);
            APLGroup par = (APLGroup)pinst.MJH_Self;
            Grid g = (Grid)par.Children[5];
            if (g.Children.Contains(btn)) g.Children.Remove(btn);
            if (pinst.Children.Contains(instance.MJH_Instance)) {
                                        pinst.Children.Remove(instance.MJH_Instance);
                                        pinst.NotifyPropertyChanged("Children");
            }
                                    APL.DeleteInstance(instance.MJH_Instance,
instance.SerialNo);
        } else if (pcl == "qwcGrid_Data")
        {
            qwcGrid_Data pinst = (qwcGrid_Data)APL.GetInstance("DataBinding",
"qwcGrid_Data", parent, this);
            APLGrid par = (APLGrid)pinst.MJH_Self;
            Button btn = (Button)instance.MJH_Self;
            Grid g = (Grid)par.Inputs;
            if (null != g && g.Children.Contains(btn)) g.Children.Remove(btn);
            if (pinst.Children.Contains(instance.MJH_Instance)) {
                                        pinst.Children.Remove(instance.MJH_Instance);
                                        pinst.NotifyPropertyChanged("Children");
                                    }
                                    APL.DeleteInstance(instance.MJH_Instance,
instance.SerialNo);
                                } else if (pcl == "qwcSubForm_Data")
        {
            qwcSubForm_Data pinst = (qwcSubForm_Data)APL.GetInstance("DataBinding",
"qwcSubForm_Data", parent, this);
            APLForm par = (APLForm)pinst.MJH_Self;
            Grid g = (Grid)par.Content;
            Button btn = (Button)instance.MJH_Self;
            if (null != g && g.Children.Contains(btn)) g.Children.Remove(btn);
            if (pinst.Children.Contains(instance.MJH_Instance)) {
                                        pinst.Children.Remove(instance.MJH_Instance);
                                        pinst.NotifyPropertyChanged("Children");
            }
                                    APL.DeleteInstance(instance.MJH_Instance,
instance.SerialNo);
        } else if (pcl == "qwcSideBar_Data")
                                {
                                    qwcSideBar_Data pinst =
(qwcSideBar_Data)APL.GetInstance("DataBinding", "qwcSideBar_Data", parent, this);
                                    if (pinst.Children.Contains(instance.MJH_Instance))
```

```
                                        {
                                                int ind = (int)
pinst.Children.IndexOf(instance.MJH_Instance);
                                                Object g =
Interop.ExecuteJavaScript("document.getElementById($0)",parent);
                                            Object div =
Interop.ExecuteJavaScript("$0.children[$1]",parent, ind);

        Interop.ExecuteJavaScript("$0.removeChild($1)",g ,div);
                                                if
(pinst.Children.Contains(instance.MJH_Instance)) {

        pinst.Children.Remove(instance.MJH_Instance);

        pinst.NotifyPropertyChanged("Children");
                                                }
                                        APL.DeleteInstance(instance.MJH_Instance,
instance.SerialNo);
                                        }
                                    }
        }
    }
```

Appendix 1
qwcGrid.txt

```
:class qwcGrid_Data M            // M = Multiple versions (ie) Individual to each Browser   // S
= Shared, one copy for every browser
:include qwcCommon
// Fields = Name  Type Default Attributes
// Attributes
//                  A = APL only
//     B = Update both ways, equivalent to using ST
//     C = Constant (no set allowed after first)
//     D = Defined - values set and read from via function
//     I = Inherited
//     O = Reflects ⎕IO value of calling evnvironment
//     P = Private
//     S = Source updated direction
//     T = Target updated direction
//     X = Process even if same value as last time

HPropList sv    “'MJH_Size' 'HeightOfTitle' 'HPropList' 'Height' 'Width' 'MaxIndex' 'HeadRows'
,
  'HeadCols'  'BorderBrush' 'BorderThickness' 'FontSize' 'ProcessList' "                  C
PropList sv       "'Type' 'Values' 'Posn' 'Size' 'FCol' 'BCol' 'Coord' 'Border' 'Active' 'Visible'
'Event' ,
'VScroll' , ‘HScroll' 'SelItems' 'Sizeable' 'Dragable' 'FontObj' 'CursorObj' ,          'AutoConf'
'Index' 'YRange'  'XRange' 'Data' 'Attach' 'TextSize' 'EdgeStyle',
'Handle' 'Hint' 'HintObj' 'Tip' 'TipObj' 'FormatString' 'RowTitles' 'ColTitles' , 'CurCell'
'TitleWidth' 'CellHeights' 'CellWidths' 'TitleHeight' 'CellFonts' 'Input' , 'CellTypes'
'AutoExpand' 'CellSelect' 'ResizeRows' 'ResizeCols' ,
'ResizeRowTitles' 'ResizeColTitles' 'ClipCells' 'InputModeKey' 'InputMode' , 'GridFCol'
'GridBCol' 'ShowInput' 'CellSet' 'RowTitleFCol' 'ColTitleFCol' , 'RowTitleDepth' 'ColTitleDepth'
'RowTitleAlign'  'ColTitleAlign' ,
'OverflowChar' 'AlignChar' 'GridLineFCol' 'GridLineWidth' 'RowLineTypes' ,
'ColLineTypes' 'EnterReadOnlyCells' 'RowTitleBCol' 'ColTitleBCol' ,   'RowTreeDepth'
'RowTreeStyle'  'RowTreeImages' 'ColSortImages' , 'SelectionColor' 'SelectionColorAlpha'
'SelectionBorderWidth' ,  'HighlightHeaders' 'Translate' 'Accelerator' 'AcceptFiles'
'KeepOnClose' ,
'Redraw' 'InputProperties' 'TabIndex' 'AlwaysShowSelection' ,
'AlwaysShowBorder' 'RowHiddenDepth' 'MethodList' 'ChildList' ,                'EventList'
'PropList' 'TitleInput' 'TitleCellTypes' 'RowTitleTypes' ,
'ColTitleypes' , 'TopTitleType' "                                                  C
EventList sv      "'Close' 'Create' 'FontOK' 'FontCancel' 'DragDrop' 'Configure' 'ContextMenu'
,
   'DropFiles' 'DropObjects'  'Expose' 'Help' 'KeyPress' 'GotFocus' 'LostFocus' ,
   'MouseEnter' 'MouseLeave' 'GridKeyPress' ‘'CellChange' 'CellMove' ,
   'AddRow' 'AddCol' 'CellError' 'CellOver' 'CellDown' 'CellUp' 'CellDblClick' ,
'CellChanged' 'GridSelect' 'Expanding' 'Retracting' 'SetRowSize' 'SetColSize' ,
```

'GridCut' 'GridCopy' 'GridPaste' 'GridDelete' 'GridPasteError' 'GridDropSel', 'GridCopyError'
'IndexChanged' 'ShowComment' 'VisibleComment' , 'ClickComment' 'Select' 'CellClick'"
C

MethodList sv "'Detach' 'ChooseFont' 'GetTextSize' 'Animate' 'GetFocus' 'ShowSIP' ,
'GetFocusObj' 'DelRow' 'DelCol' 'SetCellType' 'RowChange' 'ColChange'
'Undo' 'SetCellSet' 'RowSetVisibleDepth' 'ColSorted' 'DuplicateRow' ,
'DuplicateColumn' 'CellFromPoint' 'GetCellRect' 'LockRows' 'LockColumns' ,
'AddComment' 'DelComment' 'GetComment'" C

ChildList sv "'Bitmap' 'BrowseBox' 'Button' 'Circle' 'ColorButton' 'Combo' 'Cursor' ,
'DateTimePicker' 'Edit' 'Ellipse' 'FileBox' 'Font' 'Icon' 'Image' 'Label' 'Marker' , 'Menu'
'MsgBox' 'Poly' 'Rect' 'Spinner' 'Text' 'Timer' 'TrackBar'" C

// X indicates always process even if same as last

ProcessList	int	0	BX
HMethodList	sv	"'FontObjChanged'"	C
FontObj	string	""	IB
Type	String	Grid	C
DefaultSize	Size	"50 50"	C
DefaultWidth	int	40	B
DefaultHeight	int	30	B
DefaultState	int	0	B
IsEJ2	bool	1	C
ZIndex	int	0	B
AfterCreate	bool	0	XY
Posn	position	GetPosnGen,SetPosnGen	DB
Size	size	GetSizeGen,SetSizeGen	DB
HeadRows	int	0	B
HeadCols	int	0	B
BCol	APLItem<>	""	B
GridBCol	APLItem	"White"	B
FCol	APLItem<>	""	B
Values	APLItem<,>	""	B
CurCell	int[2]	"1 1"	BO
OldCurCell	int[2]	"0 0"	BO
CurCellIsReadOnly	bool	1	B

// Hide RowHeader ColumnHeader Cell

HideActive	int[]	0	B
Index	int[2]	"1 1"	BO
Input	string<>	""	B
TitleInput	string<>	""	B
Border	int	1	B

BorderCol	APLItem	"Black"	B
BorderStyle	String	solid	B
CellHeights	double<>	GetSizeExtendY,SetSizeExtendY	DB
CellWidths	double<>	GetSizeExtendX,SetSizeExtendX	DB
CellTypes	int<,>	0	B
ClipCells	bool	1	B
ColTitleAlign	string<>	""	B
ColTitleBCol	APLItem<>	"lightgray"	B
ColTitleCSS	string<>	""	B
ColTitleDepth	int 1		B
ColTitleFCol	APLItem<>	"black"	
B			
ColTitleInput	string<>	""	B
ColTitles	string<>	""	B
ColTitleTypes	int<>	0	B
ColTreeDepth	int<>	0	
B			
ColTitleActiveFCol	APLItem	"white"	B
ColTitleActiveBCol	APLItem	"blue"	B
RowTitleAlign	string<>	""	
B			
RowTitleBCol	APLItem<>	"lightgray"	B
RowTitleCSS	string<>	"black"	
B			
RowTitleDepth	int	1	B
RowTitleFCol	APLItem<>	""	B
RowTitles	string<>	""	B
RowTitleTypes	int<>	0	
B			
RowTreeDepth	int<>	0	B
RowTitleActiveFCol	APLItem	"white"	B
RowTitleActiveBCol	APLItem	"blue"	B
TopAlign	string<>	""	B
TopBCol	APLItem	"lightgray"	B
TopFCol	APLItem	"black"	
B			
TopTitle	string	""	B
TopMerge	int<>	0	B
TitleHeight	double<>	GetSizeY,SetSizeY	BD
TitleWidth	double<>	GetSizeX,SetSizeX	BD
TitleDepth	int	0	B

```
VScroll                 int         ¯1
        B
HScroll             int         ¯1                                          B

ShowInput           bool<>      0                                           B
Active              bool        1                                           B

Input               string<>    ""                                          B
HeightOfTitle       double              0
        BC
// Height of Title bar - here so Parent.HeightOfTitle can be used for Size

Margin                  Margin              "4↑⌽⌊0.5+(.25×GetSize Parent)-
2↑Parent #.qWG 'HeightOfTitle'"     B
Height                  Double      "⌊0.5+HeightOfTitle+.5×1⊃GetSize Parent"
                                    B
Width                   Double      "⌊0.5+.5×2⊃GetSize Parent"
                                        B

MaxIndex                int         0
            B       // Max Index of children to store data
FontSize                int         10
            B

:EventObj CellChange            obj  EJ2
:EventObj IndexChanged      obj  EJ2
:EventObj CellOver      obj   JS
:EventObj CellDown      obj   JS
:EventObj CellUp        obj   JS
:Event Return Always        CellChange          beforeCellUpdate
        object
:Event NoReturn             CellChanged
:Event NoReturn                 CellDblClick
:Event NoReturn         CellError
:Event Always NoReturn      CellDown        mousedown
        object
:Event NoReturn                             CellUp          mouseup
            object
:Event NoReturn         CellOver    mousemove           object
:Event NoReturn                     CellMove
:Event NoReturn                     GridKeyPress   KeyUp
        KeyRouted
:Event NoReturn                             IndexChanged
        actionComplete:stopScroll   object
:Event NoReturn                     KeyPress    KeyDown                 KeyRouted
:BasedOn qwcGrid APLGrid ds.Instance
:NQ Delete
```

:NQ FontChanged

:EventObj CellChange obj EJ2
:Event Return Always CellChange beforeCellUpdate
object