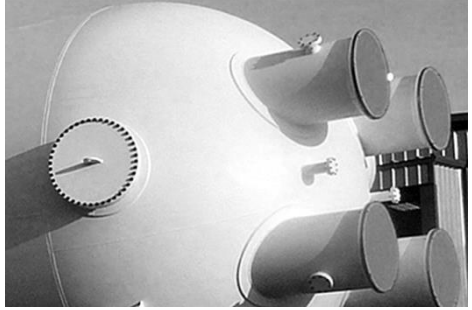
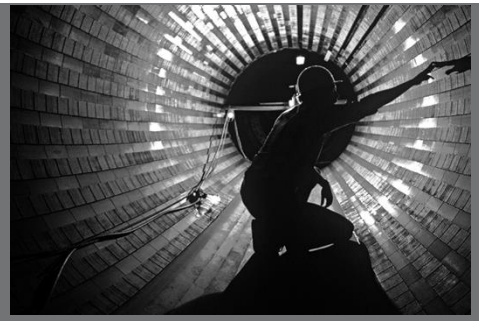
CURTISS-WRIGHT

Nuclear Division





Introduction to Viewer 5.0

Kevin Coble
Jan 2020

CURTISS-WRIGHT

Reasons to Replace Viewer 4.x

The reasons for replacing the current display system (the MMI/Build/Netapp/NetSrvr/External Applications system) are numerous:

- **Deprecated or near-deprecated technology used as the backbone**
- **Issues that have arisen from the current methodology that can only be resolved by changing the methodology**
- **New desired features that don't fit into the current methodology**
- **Unable to take advantage of performance improvements on new hardware due to not taking advantage of newer graphic card technology**

R*TIME Viewer 5.0 Goals and Objectives

- **Address issues caused by 30-year-old design decisions**
  - Resizing
  - Fixed limits (8 ranges, 32 trend lines, 1 program, etc.)
  - Client side configuration for Server data (alarm colors, etc.)
  - Multiple data hops (extapp-NetSrvr-Netapp-MMI) slowing data
- **Allow almost any client request to be addressed**
  - Dynamically defined entities
  - Configurable menus, status bar, pop-ups
- **Remove soon-to-be-deprecated technology**
  - gdi+
  - MFC
  - QuickTime (already deprecated on PC)
  - INI files (Microsoft has switched to XML base configuration
  - 32-Bit programs

R*TIME Viewer 5.0 Replacement Technology

- **The following replacement technologies are used in R*TIME Viewer 5.0:**
  - WPF – Windows Presentation Foundation (one of two current Microsoft windowing libraries)
    - The other one, Windows Forms, is used for some of the auxiliary programs
  - DirectX – A current graphics library that fully exploits hardware acceleration
  - XML Files – Allows more than two levels of organization and has fast parsing routines built into .NET
  - 64 bit programs – no longer requires the program to be run in compatibility mode on new OS's
  - C# source code (no pointers or byte buffers used, so more stable code base)

Solving Current Viewer Issues

The following issues were explicitly addressed in the design:

Issue	Solution with R*TIME Viewer 5.0
Resizing	Affine-Transform graphics that allow stretching of all graphic elements independent of aspect rations, including text
Shape Piece Connections	Shapes can now be created using Paths, which are attached segments, rather than a set of discrete lines, arcs, ellipses, etc.
Limited Number of Items in Ranges or object lists	All groups of items are contained in non-limited lists, rather than fixed size arrays
Alarm colors different on multiple units	Display state information is now kept on the server, and passed to the Viewer with the data, rather than a single color set configuration kept on each client
$LOCAL Usage conflicts	Named variables rather than indexed – no limit on number. Global and local list, with locals having current display scope

Solving Current Viewer Issues (Continued)

Issue	Solution with R*TIME Viewer 5.0
Local Configuration Updates on Multiple Workstations	Password and Display State information is now located on the servers, and sent to the Viewers dynamically
Limited Blinking Colors	48 bit color definitions allow paired 24-bit colors for any blink. (Full 32-bit ARGB colors for non-blinking items)
Having to edit multiple entities on multiple display for basic format changes	The 'Collection' model allows changes to the collections to affect all displays that use elements of the collection immediately (think re-usable subroutines)
Previous button only goes back one level	The Forward/Backward model used by web browsers is used instead
'Ghost' external applications/slow interaction with external applications	Along with an external application model that no longer uses polling through shared memory, there is an option to use 'Display Extensions' (described later)

New Features

- **The following features were also a major component of the design:**
  - Movable/sizable/rotatable entities
  - All parameters can be specified to be dynamic
    - Constants, point values, range selection, variables, etc.
  - Cut/Paste partial entity definitions between items on a display
    - i.e. – copying the color ranges defined on one entity into another
  - Multiple external applications
  - Better security
    - Expiring passwords
    - Database specific login
    - Functional rather than level based

New Features (Continued)

- **More features that affected the design:**
  - Better Display Calculations
    - If-Then-Else capability
    - More of a ‘linear’ description for the calculation
  - Live Data Editing
    - Ability to insert test data while building displays to see what the effect will be
  - Configurable look at all levels
    - Window dressing
    - Menus
    - Status bar
    - Pop-up menus

Replacement, not Update

- **Removing old foundation technology and adding the new features meant we needed to replace Viewer 4.0, not just update it. This has good and bad implications:**
  - The good:
    - Removes limitations caused by the old technology
    - Both Viewer 4.0 and 5.0 can be installed on the same box
      - Since we don't have replacements for Audible Alarms, Excel interface, etc. yet, this is a likely scenario
  - The bad
    - Can not use existing displays
      - Converter is being made, but see discussion later
    - Can not use existing external applications
      - Although existing C# applications can be converted quickly

Let's take a look…

- **With the design basis out of the way, let us take a look at what Viewer 5.0 is. We will discuss in more detail the following topics:**
  - Server components and setup
    - User Accounts/Passwords
    - Display States
  - Client side (Viewer only component)
    - Configuration options
    - Building Displays
  - Miscellaneous Topics
    - Display converter
    - External applications/extensions

R*TIME Viewer 5.0 Server Components

- **ViewerData (NetSrvr equivalent)**
- **Viewer Applications (similar, but different from current C# display applications**
- **Viewer Display Extensions (new)**
- **Support/Configuration Programs**
  - PasswordManager
  - DisplayStateEditor
- **Common Assemblies**
  - Loaded into the GAC

Server-Side Assemblies

- **The following new assemblies will be need to be added to the Global Assembly Cache (GAC)**
  - ServerClientCommon
    - Defines structure of messages and data passed between the server and the client
  - ViewerExtensions
    - Defines the interface protocol for Viewer Display Extensions
  - DisplayState
    - Defines the Display State structures and file format

Server-Side Directories

- **Only one new directory required**
  - Extensions
    - Location for Viewer Display Extensions (dll files)
    - May become configurable
- **Will use standard R*TIME directories**
  - bin
    - ViewerDisplay Applications will be here – they are C# applications (created by ProgramCreator)
  - data
    - Configuration files
    - DisplayState files
    - Password files

R*TIME Viewer 5.0 Client Side Components

- **Viewer**
- **… and nothing else!**
  - No ‘NetApp’ like program, each Viewer instance gets its own direct socket to the server(s)
  - No audible alarms, archive report, or Excel interface either
    - If you need the functionality of these, you will have to have the legacy Viewer installed as well. Both can operate on the same system without issues

Environmental Variables

- **Only one needed on the client side**
  - **RTIMEViewer**
    - Contains the directory that has the default configuration file (ViewerConfig.xml) and the language-dependent strings file (ViewerStrings.xml)
    - All other directories are specified in the configuration file (no environmental variable for the temporary file directory)
- **Server side has no new environmental variables**
  - **RTIMEHOME is used like most other server-side programs**

Client-Side Assemblies

- **The following new assemblies will be need to be added to the Global Assembly Cache (GAC) on each client**
  - ServerClientCommon
    - Defines structure of messages and data passed between the server and the client
  - DisplayState
    - Defines the Display State structures and file format
- **These are the same assemblies as those on the server side, but may have different version numbers if things diverge in the future.**
  - The assemblies pass version numbers with all data formats, so that backwards compatibility can be maintained.
  - Multiple versions of an assembly can be in the GAC, so server/client combo machines may need to do this if the client/server versions diverge later.

Client-Side Directories

- **All client-side directories are specified in the configuration file, and therefore do not need to be in any specific hierarchy, although that will be recommended so others can easily find things**
- **The following directories must be present**
  - Writable : the temporary file directory. This is the only one that (non display-building) users must have write access to
  - Media : the directory for any graphic files that are not displays built with the Viewer
    - Gif, jpeg, svg, etc are all supported (anything supported by Microsoft's media library should work)
  - Collections : this is where Collection files are stored. Collections are like libraries of pre-defined entities (or displays) are kept
- **A main display directory should be defined as well, and may be required for things like page navigation, etc.**
- **Additional display directories can also be specified. These may be used for things like shapes, frames, etc. – although there is no such distinction between these in Viewer 5.0, everything is a display (including the status bar, pop-up menus, etc.)**

ViewerData Configuration

- **The ViewerData program performs the functionality done by NetSrvr in the previous versions of the Viewer**
- **It uses the RTIME assembly and looks like most other C# applications on the R*TIME server**
- **Some configuration options**
  - Port #, maximum connections
  - Encryption – if it will accept unencrypted connections or not
  - Application directories

Colors

- **Starting with full RGB colors this time, no Colors.ini file**
- **Alpha channel available on all colors (transparency, translucency) can be used in every object**
  - Displays are not drawn on a changed-only basis, the entire display is redrawn and composited each update, so transparency works like expected
- **Color is stored as a single number, and can be put into points, calculations, etc.**
- **8-bit channels for ARGB on non-blinking colors, 6-bit channels for each part of a blinking color**
  - Recommended that most colors used in standard data displays that may blink based on a display state stick with 6-bit compatible color numbers so there isn't a slight difference when the state changes.
- **Custom colors can be saved into a (local) file, for easy selection later during display building**

Display States

- **Display States replace the Alarm Colors – renamed because more than alarms affect the state of a point (quality, user status bits, etc.)**
- **The definition file contains not only the rule set, but the strings, colors, and state indices together – not split between ftch_alm_ind.ini and mmi.ini**
  - Kept on the server, so no need to update clients when changed
  - Can be different for each database, and clients connected to multiple databases (even displays with points from multiple databases) will use the database specific rule set for each point
  - The resulting index number, state string, foreground color, background color, and plot color can be used to drive any part of any defined entity.

Display State Editor

- **Display State definition files are created, maintained, and tested using the Display State Editor program.**
  - GUI for rule editing
  - Allows testing against a manually entered state (quality, alarm info, user bits)
    - Testing shows the rule (or default) that would result from the entered state
  - Allows testing against an actual point in the database
    - Because of this, it must have access to the CVT, so must be executed on a running R*TIME server
    - The points' status word is read and used to determine the rule that would trigger for the current state of that point
  - Can read one database's rule set, and save as another DB (convenient for getting a starting rule set for a new project
  - <Demo>

User Accounts

- **User accounts for the new Viewer are kept on the server**
- **Accounts are database specific. The Viewer can be logged into multiple databases at the same time**
  - If logged into one database as a specific user, the Viewer will attempt to log into subsequent databases using the same account, only asking the user for an account entry if that fails.
- **Access levels for a user are individual bit fields for each level (0-31)**
  - If bit for the actions specified level is on, the user is authorized for that action
  - Setting the highest bit the user is allowed, and all bits underneath, simulates the legacy Viewer authorization levels
- **Account passwords can be set to time-out after a given period**
- **Active Directory can be used for passwords, or account/passwords for default users**
  - If user is in Viewer password file (for the database), it can be set to use AD for password/expiration functions. Authorization levels can be individually set for these users
  - If a default authorization level is set in the password file for the database, any valid AD account can log in, even if the account is not specified in the password file. Those users will get the default authorization levels
  - Active Directory Server must be specified in the ViewerData.xml configuration file.

Password Files

- **Password files are database and machine specific, and are kept in the R*TIME data directory with names Passwords_<machine name>_<database>.pswd, where the server machine name and the database the file is for are part of the name**
  - File is fully 256-bit encrypted
  - File will only work on the machine in the name, and for the database specified (they are encrypted inside the file, so renaming won't help)
    - Can use PasswordManager program to convert a file from one server to another, or one database to another
- **Each user in the file can be set to either use the file entries for password and expiration dates, or Active Directory**
- **Created and managed with the PasswordManager program**
  - <Demo>

Viewer Command line

- **The Viewer can be started with a set of optional command line parameters that can be used to obviate the need for multiple configuration files, especially on multi-monitor machines**
- **These command-line options override any setting for that parameter that is in the configuration file**
  - '-config' : the next parameter is interpreted as a path to an alternate Viewer configuration XML file. This file is used instead of ViewerConfiguration.xml in the RTIMEViewer directory
  - '-alias' : the next parameter is used as the string that identifies this Viewer in the send-display dialog boxes
  - 'initialDisplay' : the next parameter is interpreted as a display name to be activated on Viewer startup

Viewer Configuration

- **Below are a few of the configuration options for the Viewer that highlight some of the new options:**
  - CanEdit : The Viewer is both the display builder and the display presenter. This option determines if the display building functions are disabled.
  - BackgroundColor : Displays do not have a background unless you add a filled rectangle. This option allows a configurable background for all displays.
  - AdditionalDisplayDirectory : Multiple display directories can be defined
  - The Printing section contains print options
    - AutoRotate – Yes/No. Rotates display 90 degrees for printing (makes it unnecessary to change printer to landscape)
    - Margins for the print (the display can stretch to any size, so will fill the print page without margins)
  - The Status Bar section defines the (optional) status bar that can be placed at the bottom of the Viewer window
    - The status bar is just another Container, and must be built like any other display
    - Named variables are provided for the current login, display name, and points underneath the cursor. These can be used in the status bar display.

Viewer Configuration (continued)

- **The Connections section of the Viewer configuration file is the equivalent of the DataServ.ini file in the legacy Viewer. It has the following entries:**
  - DefaultConnectionDB : Initial connection database. There must be a 'Connection' entry with this database in the section
  - Connection : The definition for a connection for a database. Any number of these can be in the Connections section. The child elements for a Connection are as follows:
    - Database – the database this connection is for. If there are more than one database on a server (that are being used by displays), more than one connection definition will be needed
    - Type – the connection type. One of the following
      - FirstToAnswer – first server in the list to answer connection gets to feed us data
      - Master – first server in the list that responds that it is the master computer gets to feed us data. Will start to reconnect if currently connected server ever says it is no longer master (no points needed with client machine names!)
      - Redundant – first server in the list that responds that is a redundant computer gets to feed us data
    - UpdateFromOtherServerTime – amount of time (milliseconds) that can elapse without data from current connection, before the next server (even if not master/redundant) will be allowed to update this client
    - Server – Source machine name and port for connection. More than one of these entries are allowed in a Connection
    - Encryption – "Yes" or "No". If yes, all data is encrypted before transfer. Server must be configured to accept encrypted connections if this option is used, or to accept non-encrypted connections if this option is off.

Viewer Configuration (continued)

- **More options:**
  - Completely user-defined menu structure
    - Edit-mode menus fixed, but all View-mode menus come from configuration file
    - Allows for user customization, user menus, security removal of functions
    - A standard default set will be provided
  - Completely user defined shortcut keys
    - Edit-mode keyboard shortcuts fixed, but all View-mode shortcuts come from the configuration file
    - Any command, action list, or display can be assigned to any key
    - A standard default set will be provided
  - Window style (WPF supports three styles)

Send-Display

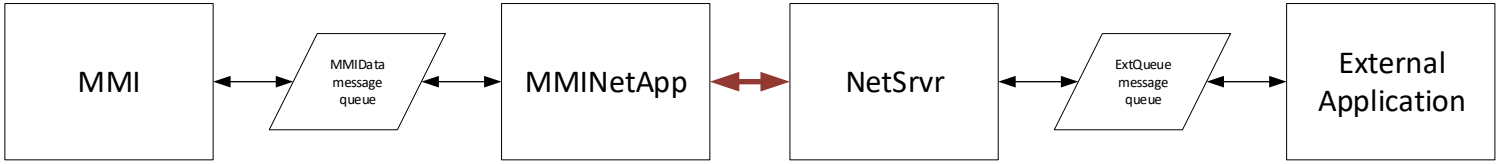
- **Some differences on the Send Display**
  - The Viewer ‘registers’ itself with all servers it is connected to, not just the default server. It will therefore show up in the display pick-list multiple times if multiple databases are available
  - Any Display State for sending the ‘current display’ is kept as part of send-display message, not a separate file. Therefore, there does not need to be a writable directory for DisplayStates for the send operation (but may need one if the user can create them manually)

External Applications

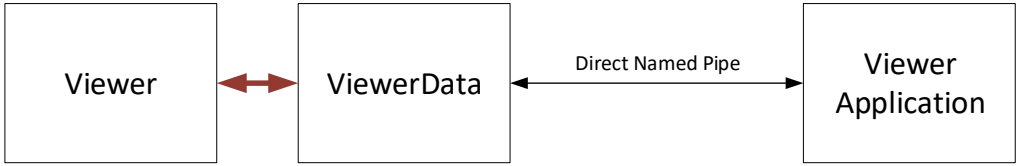
- **External Display Applications now come in two types**
  - ViewerApplication
    - Similar to the C# DisplayApp, including the use of the R*TIME assembly. Differences noted here:
      - Methods get called by name (not function code), and get passed named variables from the display
      - No ‘Field’, ‘Local’, or ‘DrawArea’ objects – these are all accomplished with named variables
  - ViewerExtension
    - An assembly that is loaded into ViewerData so its’ methods can be called directly
    - No time-extensive operations (reports, etc.)
    - Display actions that require responses (dialogs, variable reads, etc.) are handled with callback functions – there is no threading or pausing.

External Applications

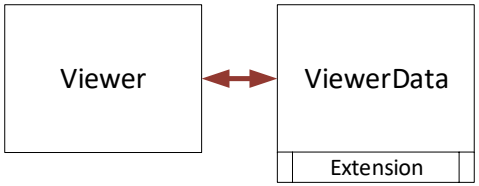
- **Currently**



- **ViewerApplication**



- **ViewerExtension**



Break…

- **Let's take a 5 minute coffee/bathroom break before diving into the displays themselves…**

Displays

- **The following design guidelines were used to guide development of the new Viewers' display format:**
  - Anything that can be drawn, can be used on a display dynamically
    - WPF 2D-graphics have been wrapped with tools to allow almost all parameters to come from data. If WPF can draw it, the display can put it on there with conditional data
  - No limits on any items
    - This includes ranges, trend points, entity counts, menus, frames, etc.
  - Re-usability of components
    - The ability to define an item, save it, and re-use it in multiple places
  - Updating not dependent on draw order – full compositing
  - Customer defined layout
    - Including menus, status bar, pop-ups, window dressing, etc.

Displays (cont.)

- **WPF primitives that can be put on a display**
  - Text
    - Consisting on one or more ‘Runs’ of text. Full font, size, color, attribute control for each run, full layout control for text item
  - Graphics
    - Lines
    - Rectangles
    - Ellipses
    - Paths
      - Constructed of connected line segments, arc segments, and Bezier curves
    - Image
      - Any Microsoft media format should be supported (GIF, jpeg, SVG, etc.)

Displays (cont.)

- **R*TIME primitives that can be put on a display**
  - Trend Line
    - A single trend line with an X, Y, and Time axis. Time can be duplicated into the X or Y axis for time-based trends
  - Controls
    - Click Area – runs an action script when clicked on
    - Text Entry – opens a text entry field with appropriate input parsing
    - Slider – allows the user to drag the cursor along a path to set a value from a range

And that is all. The last two pages are a complete list of drawable items that can go on a display

So you ask: "How do I put something like a bar-chart on a display?"

For that, there are Containers and Collections. But first, we must discuss the parameters for these primitives…

Displays (cont.)

- **Each item that can go on a display has a base set of parameters**
  - Position
  - Size
  - Rotation
  - Visibility
- **And often a set of item-specific attributes**
  - Line and fill attributes for graphics
  - Text runs for a text object
  - Formatting for text inputs
  - Etc.

DisplayData

- **Many of those parameters can be broken down further. For example, the position contains both an X and Y axis location for the object.**
- **But all parameters eventually work their way down to a set of numbers or text strings. For example a text run has a string to be displayed (text), a font name (text), a font size (number), a color (number), etc.**
- **Each text or number item parameter in the new Viewer comes from a class called DisplayData. A DisplayData can define the text or number as a data item from one of several sources**
  - Constant Number – a number defined at build time
  - Constant Text – a text string defined at build time
  - Formatted Number - a text string formatted from a supplied number
  - Formatted Time – a text string formatted from a supplied number, interpreted as a time value
  - Range – selection of a number or text item from a list with ranges, or a default value if no range matches the provided selection value
  - Point Reference – a number or text string from a point in an R*TIME database
  - Variable – a number or text string stored in a named location for the display
    - Can be filled with a value from other items, programs, or calculations
- **Many of the DisplayData source types have multiple parameters. For example, the Range type has a selector value, a default value, and a number or text for each range. Each of these are themselves defined as DisplayData objects**
  - Recursion – but this allows multiple items to drive one parameter (i.e., quality and value both driving the color of a pump shape) without calculations or programs

Composite Objects

So, back to that Bar Chart…

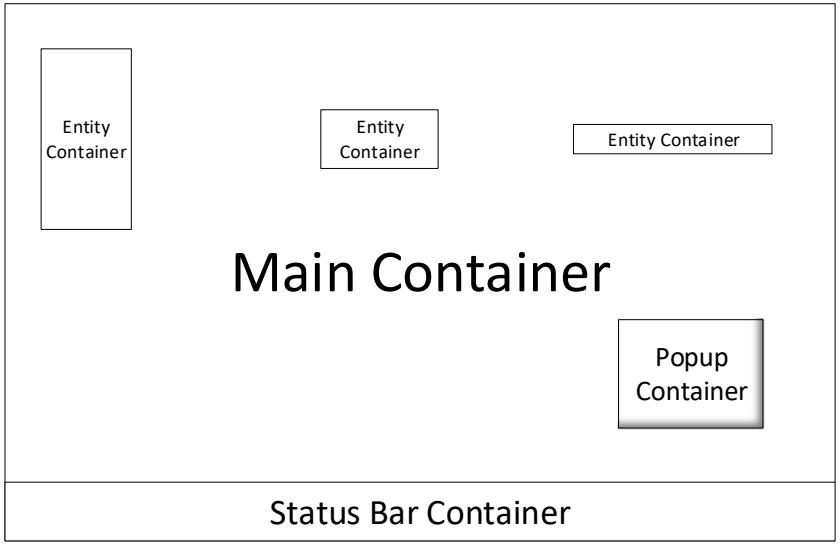
- **A Bar Chart is just a rectangle that grows in the specified direction based on the value, minimum and maximum settings, and a frame rectangle. So, we just have to:**
  - Draw the frame rectangle
  - Set the minimum and maximum values into variables
  - Create a calculation that defines the size of the rectangle
  - If not growing in the down or right direction (position is upper-left), then another calculation for the starting position of the rectangle
  - Add a rectangle, with the preceding calculations for position and size providing those parameters
- **If you want a more professional bar-chart**
  - Add tick-marks
  - Make the border size configurable – this will affect the sizing calculations
  - Make the color change based on the driving point Display State
  - Etc.
- **… But that is a lot of work for one bar chart!**
  - Yes, but we have full control over all aspects of the bar chart, not just what Scientech thought needed to go into a bar-chart entity
  - And this is where the 'reusability' design guideline comes in. Let's build a bar chart the way we want it, and make it so we can re-use that work for all other bar charts.

Composite Objects (cont.)

- Size a new window to have an aspect ratio close to that of the bar chart we want
- Create the bar chart as we did before, but make it fill the entire window
- Save the display to a file (or to a Collection, discussed later)
- Now that bar-chart definition can be used on other displays, in something called a ‘Container’

Display Layouts

- **Displays are based on ‘Containers’. A Container is a rectangular display area that has its’ own display definition (from file or Collection), local variable set, and sizing/positioning parameters**
  - The main display area of the window is a Container, the only required one
  - The status bar, if defined, is a Container
  - Any pop-ups are Containers
  - Additional Containers can be put on the display to show pre-built sub-displays (frames, entities, etc.)



Display Layouts

- **Containers can have their own local variables, calculations, sub-containers**
  - Each Container can be named for reference by programs, other containers, calculations
    - For example, if you want your program to set the minimum value of a trend entity container you have named ‘Trend1’, if that value is in a local variable called ‘Minimum’, your program would reference Trend1.Minimum to set it. The same entity could be in another Container, where you would used Trend2.Minimum to set the same local variable in that Container (if you named it ‘Trend2’)
  - Each Container can have an external application or extension associated with it.
    - This allows ‘overlays’ that have application functionality, such as ‘last five messages’, etc.
    - Entities can be built that require special data from the server, and have their own copy of an extension to drive it.

Containers

- **So we can open a new display to edit, click on the add Container menu command, and a new Container object is added to the display**
  - Like any other object, you can set position, size, rotation, and visibility for the Container object
  - You can assign a point mapping from the Container point list to the enclosing Containers' point list (in this case the parent Container is the main window)
  - You can override variable settings, such as the minimum and maximum bar chart limits in our definition

Containers (Continued)

- **Point Assignment**
  - Each Container has a list of points used in the display definition inside
    - References are made by index. Therefore, if a point name changes, only the point list needs to be updated, as all entities using that point will reference the index within the point list.
  - The main Container (the main display area) contains the point list that will be sent to the ViewerData program to have current values sent periodically, or as they change.
  - Sub-Containers (entities put on displays, status bar, pop-ups) are usually built with 'PlaceHolder' for the points
  - The Containers then have a cross-reference map, mapping the PlaceHolder in the Container to the index of the point list of its' parent Container.
    - When filling in the cross-reference map, the point can be selected from a point selection list, the point will be added to the parent (if it is not already in the parent's point list), and the appropriate index put into the map.

Containers (Continued)

- **When building a sub-display for use as an entity on the display, you can use named Variables for values that you want to have configurable later.**
  - Give it an initial value that will be the default setting for that Variable
- **A Container (other than the main one) has a named Variable override list**
  - Any named Variable in the sub-display used by the Container will show up in the definition dialog allowing an override value to be specified for it
  - If an override is specified, the override value will be put into the named Variable when the sub-display is loaded
- **Between point mapping and Variable overrides, Containers can behave a lot like entities from the version 4.x Viewer**
  - Now we need a way to easily access them…

Collections

- **A Collection is a library of saved displays**
  - Each entry gets a name
  - Each Collection has a name
  - Any number of Collections can exist and be referenced on a display
  - A Container has an entry for a Collection name and entry name to reference the sub-display that will be shown in the Container
    - A Collection name of '*File*' indicates the display definition comes from a file rather than a named Collection
  - A list of used Collections is presented when the Viewer is in edit mode
    - This list can be pre-filled based on entries in the configuration file
  - A Collection can be selected from the Collection dialog (in edit mode), and a list of entries is shown. Selecting and entry and clicking on 'Add to Display' adds that Collection entry as a Container on the current display definition

Converting Version 4.x Displays

- **There is a display converter for 4.10->5.0, but it is not recommended for anything but a first draft**
  - The two Viewers are vastly different. The converter has to try to get the 5.0 Viewer to act like the 4.10 MMI program, and that is not what it was made to do.
    - Shapes that have multiple parts that were not joined in 4.10 will not be joined by the converter. This means that the resizing problem where shape parts separate when being resized will still happen.(though it won't be as bad due to line-width sizing)
    - No entity re-use, so the display load/convert time will be larger than a display built with reusable components
    - No way to tell if multiple entities are different, or just 'stacked', where the stacked entities should become a single item in the new Viewer.
    - Has to assume any differences in ranges are not due to 'overlapping', and so will create a different sub-display for each different entity that has even the slightest difference in the specified ranges.


G
N