Aspire Browser

User Guide

# Introduction

The Aspire Browser provides a user interface (UI) to a running Aspire environment. It is built atop Aspire Studio and presents itself much as many other Integrated Development Environments. The application window is composed of several panes of docked views. Each docked view provides insight into a particular area of the Browser’s internal state. After a brief tour of the Aspire Studio layout, the Aspire Browser will be discussed.

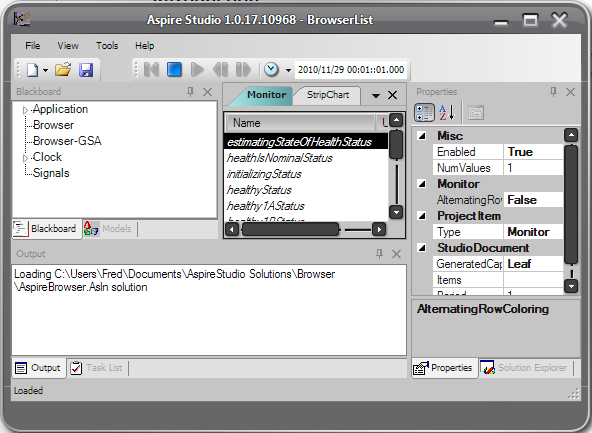


Figure Aspire Browser application window

# Aspire Studio layout

Here is a brief list of the user interface elements that you see in Figure 1.

* Title – Shows “Aspire Studio”, the version <major>.<minor>.<build> and the scenario within the solution.
* The main menus, used variously
* The File toolbar, used to create a new docked view, open/save a file
* The Execution/Time toolbar is used to control time much the same as you would find on a VCR. It also displays current time in one of the following formats
  + Mission time – elapsed time from scenario initial time [D HH:MM::SS]
  + UTC
  + Elapsed seconds
* Left docked views
  + Blackboard view is used to display the hierarchical Blackboard
  + Models view allows you to peruse and edit the running models
* Center for Document views
  + A Monitor view provides dynamic updates of blackboard variable values. Built interactively and saved in the current solution
* Right docked views
  + Solution Explorer allows you to browse and configure multiple scenarios within the current solution
  + Properties view provides a property grid for the currently selected object in the running system. This can be a blackboard object or some other object on a view.
* Lower docked views
  + Output view is essentially a text console for messages
  + Task List view allows you to add and track tasks to do for your current activity.
* Status line shows various status annunciations.

# Docking

You can dock and undock the windows, add and remove specific views. Use the View main menu to re-enable a docked view that you might have turned off. Click the ‘x’ in the upper right of the view to remove it. Use the ‘pin’ to hold the view open or let it collapse against the border it is associated with. Double clicking the title bar will undock a view.

When a view is undocked, you can drag it to various docking sites that become visible when you start dragging a views title bar, seen in Figure 2. Drop the view on the docking site to move it. Drag a view’s border to resize it. You will accumulate multiple tabs in a docking pane for each of the views that is in that pane. Drag the pane into to appropriate order to reorder the tabs.

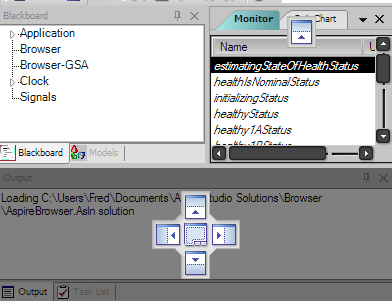


Figure Docking sites

# The Blackboard and its View

The Blackboard is a namespace that all the models running in Aspire Studio can publish their constituent elements to so that the UI can make them available for interactive binding to other UI presentation and control elements. The blackboard view presents the blackboard items in a tree view that mimics the hierarchical structure of the Blackboard itself. The Blackboard and the view are fully dynamic, allowing items to be added and removed at any time.

Each item on the view may have an icon and/or sub-items. You click the ‘plus’ on the left of an item to crack it open to see the sub-items inside. You can also double click the text to accomplish the same thing. Using the right mouse button, a context menu appears that is unique to each item. The context menu (CM) is useful for providing item-unique actions. One menu item that is common to all is to paste the full path of the item to the clipboard. It is then available to paste into any clipboard target that might receive plain text.

Each blackboard item may be dragged to other UI elements, specifically document-backed views that are visible in the center panel. This is the method whereby the user can interactively bind UI elements to the Blackboard. More discussion follows in Monitors and Strip Charts.

# Property View

The property view populates a property grid with the public properties of selected objects. The active selected object is quite dynamic, but there is usually some indication in the other views which object is actively displayed in the property view.

The property view updates every second when no properties are selected on it and the time base is running. If you select a property, the property view pauses its updates until some other UI element outside the property view is selected.

Properties on the property grid display as name, value pairs, grouped under categories. Many categories are initialized closed, so if you can’t find a property, remember to crack open closed categories.

The value cell on the property grid does many things, depending on the type of the datum it contains. For instance, if the value is a Boolean, double clicking the Name cell toggles the value between true and false. If the value is an enumeration, double clicking the Name cell increments the value cyclically through the enumerations that are valid.

Sometimes the value is a class or an array and there will be a ‘plus’ adjacent to it. Cracking it open reveals sub-structure. Sometime there is an editor associated with the type and a button (“…”) will appear in the value cell after the first click. Clicking the button opens the editor.

# Solution Explorer

The Solution Explorer (SE) presents a hierarchy of the objects loaded in the current solution. This includes Scenarios, Monitors and Strip Charts at the moment. Its use will be greatly expanded in future versions of Aspire Studio.

Each scenario in the solution is listed at the first level of hierarchy. Scenarios may be set to the active scenario using the CM. New scenarios may be created using the CM, but it is quite cumbersome at this time.

Cracking open a scenario reveals the Monitors and Strip Charts that it has defined. These may be actively displayed in the document windows center pane, but they may not. Cross reference the names to confirm. These persistent forms of a document view are created automatically whenever a new view is first created and populated. They are automatically saved in the solution.

If a document view is not displayed in the center pane, double clicking its icon in the SE will cause it to be displayed. You can also remove the document with the CM.

# Monitors

A monitor is a real time display of the values of the blackboard items that have been dragged onto it. There is an existing one shown in Figure 1, but let’s start from scratch with an example.

* Using either File,New,Monitor or clicking the New tool on the File toolbar, then Monitor, create a new Monitor. It will be called Monitor2 since there is already a Monitor1.
* Switch the right pane to the SE and crack open BrowserList. You’ll see several document views depicted.
* Crack open the Clock in the Blackboard view and drag Seconds from the Blackboard view to the new Monitor. You will notice that a new Monitor2 icon appears under the scenario. This happens when the first item is dragged onto the monitor. Also notice that Seconds is updating, showing the current value of Clock.Seconds.
* Switch back to the Properties view (Props). You will see the properties for Seconds. Notice that LeafName is Seconds and the Units are sec. These are the blackboard item properties used to initialize the Monitor item’s properties as shown on the new Monitor.
* Click anywhere on the Seconds row in Monitor2 and the MonitorItem’s properties are shown in Props. Notice that Caption says Seconds. There several caption generation rules that might be in effect, but this shows the simplest one: use the Leaf name to initialize the Caption. You can change the Caption to anything you like and it will be preserved.
* Drag MicroSeconds from the BV to Monitor2. Notice that MicroSeconds does not appear to change. This is because the Monitor is only updated 1/sec.
* Double click on the Monitor2. Now the whole Monitor’s properties are visible.
* Change the Period from 1 to 0.1. Now MicroSeconds is updating quite nicely.
* Press the Stop VCR button.
* Press the single step forward button to see MicroSeconds update every 0.125s frame. Try it multiple times. Then press the single step back to see time count down.
* Restore the time base to the running state by pressing the play button.
* It might be somewhat tedious to drag each item from the BV to the monitor, but you drag higher level constructs from the BV to the monitor. Try dragging the whole Clock.
* Seconds and MicroSeconds appear a second time, but GpsSeconds from the Aspire sub-clock also appear.
* Double click the AlternatingRowColor property to clarify the monitor. If there were many items on the monitor, this would make them easier to read.
* Using the Solution Explorer, right click on Monitor2 and Delete.

# Strip Charts

Let’s take a guided tour of the strip chart now. Many concepts are similar to the Monitor in that we are creating displayed items from blackboard items dragged onto the view. So, here we go.

* Create a new StripChart
* Drag MicroSeconds onto it
* You will see something like the Figure 3.
* Using SE, remove StripChart2. Add a new StripChart.
* Using CM, Add sine to the Signals blackboard item. Crack open Signals and select sin0. Change its Amplitude to .1 and its Frequency to .1 in Props. Drag its Value blackboard item to the strip chart. You will see a standard sine wave. Switch to Props and double click the strip chart’s view and change the TimeWindow from 10 to 25
* Notice the period is set to 0.125, the same as the Clock’s StepSize. Change this to 1 and back to see the effects of sub-sampling.
* Toggle the ShowLegend and ShowYAxisTitle properties false and back to true. You can do the same thing by using the CM when clicking over the legend or Y-Axis. You should see Figure 4.
* Right click in the chart area and add a new ChartArea. Drag MicroSeconds onto the lower half of the view.

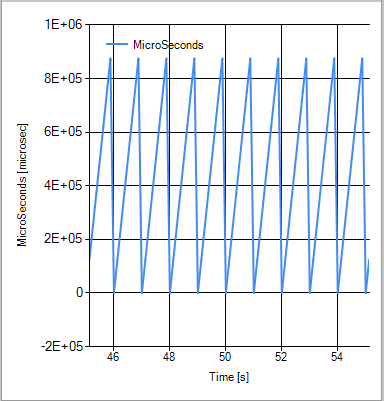


Figure Strip chart of MicroSeconds

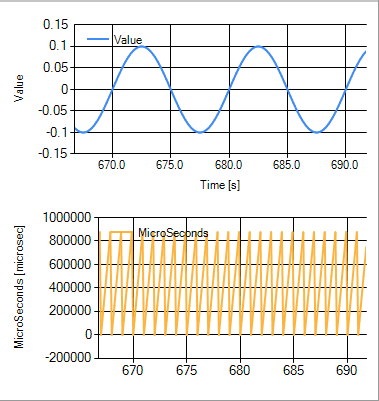


Figure Two ChartAreas on a single strip chart

# Browser

The Aspire Browser is built on top of Aspire Studio using plugin components. It acts as a UI to an Aspire environment running on the same machine. So, let’s start up Aspire and see what happens to our running Browser. Just execute the StartAspire.bat file in Aspire’s installed bin directory. You should see a ‘plus’ adjacent the ‘Browser’ Blackboard item. If not, read the console windows of ProcessorManager or Directory for clues.

Assuming the Browser registered and populated, select the Browser. Its NumComponents property should have a value of 3. Crack open the Browser in the Blackboard and you will see that they are Browser, Directory and ProcessorManager (PM).

Let’s explore PM just a bit. Crack it open and you will see that it provides two interfaces, iCoreComponentMgmt and iStatus. Looking in iStatus, we see the GetTimeStats message. The two arrows in the icons indicate that this is a Request that we can send to PM and it will send a reply back. Let’s do that. Using the CM on the GetTimeStats message, Send Aspire command. How do we see what happened? Crack it open and notice that Sent has a value of 1. This is a counter that is on every outgoing message. Crack open the reply message, TimeStats. Received is also a 1, indicating that we received a reply to our request. The two Received counters have non-0 values, indicating that PM is receiving our Time at Tone and PPS messages. Great.

Try exploring other components and dropping the Component, Interface or Message on the monitor. All of the aggregates variables will be put on the Monitor.

The Browser is a model running in Aspire Studio. The default scenario in the Browser solution is called BrowserList, because there are actually two Browsers in the model list: Browser and Browser-GSA. The distinction between the two is the domain that each registers with. In Aspire, a Directory hosts a domain, so any Aspire component that registers with a particular Directory is said to be in that Directory’s domain. Running an application without specifying a domain results in that application registering with the ‘Default’ domain. In order for the Browser-GSA to register with the GSA domain, you would need to start a Directory with –dGSA in its command line.

When Browser detects a local Processor Manager, it gets the address of the Directory that matches its domain. Currently, you have to start a Browser for each domain that you are interested in. That domain may be hosted on a Directory that is local or on a remote machine in the LAN. You can have as many Browsers as your Aspire domain topology requires. Simply edit the BrowserList.xml file in MyDocuments/AspireStudio Solutions/Browser. Here is an excerpt:

<Model xsi:type="AspireBrowser" name="Browser" />

<Model xsi:type="AspireBrowser" name="Browser-GSA" domain="GSA" />

Notice that the domain is simply specified as an XML attribute on the Browser model. You can use any naming convention, but we typically append the domain as a suffix.

After Browser registers with a Directory, it synchronizes its component list with Directory’s. Each Browser component, those listed under Browser on the Blackboard, represents a consumer’s perspective of the xTEDS provided by each of the Aspire components that has registered with Directory. Thus the icons on the messages and the Sent/Received counters show information flow to the provider from a consumer’s perspective.

There is a complementary model that can be added to a scenario that implements the provider’s perspective on an xTEDS called the AspireShell. You can add an AspireShell to the model configuration file like Browsers or you a=can add it interactively.

One approach to software development is to specify the interfaces between software components and then build providers and consumers of the methods described in the interface. In Aspire, the interface is contained in the extensible Transducer Electronic Data Sheet (xTEDS), which is described with XML in a file whose type is ‘.xteds’. The Aspire plugins to AspireStudio parse and use the xTEDS to generate the actual operational interfaces used in the Browser components and the AspireShell. The Browser gets the xTEDS for the component when it syncs to Directory. The AspireShell needs to know where to find a local xTEDS and although an xTEDS file can be anywhere, we find it convenient to keep them in one place. There is an xTEDS sub-directory in the solution folder where there is an example xTEDS that we will use for the AspireShell example.

Suppose the xTEDS has been created and the development of two Aspire components is being done in parallel. The provider developer can use the Browser to give themselves a UI into their application during development. Similarly, the consumer developer can use the AspireShell to mimic the provider’s use of the xTEDS to test the consumer under development. There are other use cases: maybe you just have the xTEDS and can’t get the source of a provider. Your consumer has a bug and you just want to send it a message sequence that triggers that bug.

So, let’s create the AspireShell and get on with it. Using the CM on Browser, select AspireShell. Browse to an xTEDS that the shell will use as a scaffold. Use

“MyDocuments/AspireStudio Solutions/Browser/xTEDS/StatechartTestHarness.xteds”. Just accept the domain that matches the Browser that you started with. (You can start on Browser-GSA and the domain will initialize to GSA). Click OK. Now you will notice StatechartTest shell on the Blackboard. Make the StripChart prominent in the center pane and you will notice that there is a Temperature trace that is updating now. That’s because the strip chart was saved with that configuration and it has finally seen.

“StatechartTest shell.IStatechartTest.ThermalStatusMsg.Temperature” as a valid Blackboard item. It’s pretty uninteresting though. Do this:

* Using CM on Signals, Add sine. Crack open Signals and select sin0. Set its Amplitude and Frequency properties to 0.1 on the Properties view.
* Crack open sin0. Crack open “StatechartTest shell.IStatechartTest.ThermalStatusMsg”.
* Using CM on ThermalStatusMsg’s Temperature item in the Blackboard and select “Bind to”. The cursor changes to a big cross. Click on the sin0’s Value item. This binds Temperature’s value to the one generated in the sin wave generator. Very useful for testing.

Now let’s look at what happens in a sampled data system like Aspire.

* Create a new strip chart
* Drag StatechartTest shell.IStatechartTest.ThermalStatusMsg.Temperature to it. Notice that is has the same name as the providers, so let’s change it to Sampled Temp. Click on the ThermalStatusMsg.Temperature1 in the legend and edit the Caption to say ‘Sampled Temp’. Hit enter.
* Set the strip charts time window to 25
* Drag Browser.StatechartTest.IStatechartTest.ThermalStatusMsg.Temperature to the strip chart. Notice the 1 second zero order holds. This is because the StatechartTest xTEDS specified the ThermalStatusMsg as PERIODIC with a msgRate = 1 per second. Let’s see the effect of changing the xTEDS:
* Select the StatechartTest shell in the Blackboard.
* Crack open PeriodicMessages, then ThermalStatusMsg. Change the MsgRate to 2 and toggle Enabled from true to false to true. (Increasing the sample rate tracks better at the cost of higher message traffic) Now the sampled signal tracks the 10 Hz signal more accurately. This provides an interactive way to validate the xTEDS. Of course, it’s up to you to know what reasonable test values are, but you don’t have to edit the xTEDS and reload to find out. When all done, you would the edit the xTEDS with the final values. Even though it’s possible to save the tuned xTEDS, any comments that were in the xTEDS would be lost when saved