Advanced Service Interface

The **RiverSystem.Service.Cluster.dll** and **RiverSystem.Service.Cluster.Host.exe** provides a mechanism to run many different servers without the Source interface . All the features of the Command Line Runner interface are available but the cluster has no ability to be used without external code being written. The Command Line Runner should be used for ad-hoc server creation and batch jobs. The cluster allows for automatically starting and stopping Command Line Runner server instances.

The **RiverSystem.Service.Cluster.dll** can be hosted in IIS. We also provide a simple self hosting executable called **RiverSystem.Service.Cluster.Host.exe**. Running this will start server with the default endpoint of

http://localhost:8001/Source

to specify the endpoint you can provide a parameter during startup

```
RiverSystem.Service.Cluster.Host.exe http://somemachine.domain.com:8001/Source
```

RiverSystem.Service.Cluster.Host.exe and RiverSystem.Service.Cluster.dll are included with Source.

The cluster is used in custom situations where you need to start and stop services on a remote machine:

- for custom development and customisation where running source in being automated

The cluster can only be used in a server configuration. Accessed it can automatically create new servers on the host machine to run source models and destroy them when they are completed.

All communication with the cluster are performed over http. The follow table outlines the calls that can me made. All calls addresses should be considered as appended to the base address of the service.

eg. if the service is hosted at http://localhost:8001/Source a call to the echo command would look like http://localhost:8001/Source/echo/AMessage and the result would return the string "You Typed: AMessage". This echo endpoint can be very useful to verify that the cluster server is up and responsive without changing any state on the server.

Items shown between curly braces are parameters that you would configure as part of the call.

Interface Address	Function	Result
/echo/{message}	A test location that will return the message: `You Typed: {message}` for whatever was pass in. This location is useful for checking the server is up and responsive.	`You Typed: {message}`
/projectDirectory/{*projectDirectory}	Directory in which to look for projects	`true or false`
/create	Starts a new server and returns a {instance} string that should be passed to most other calls to communicate with this created server.	`{instance} identifer`
/load?instance={instance}&project={projectName}&scenario={scenarioName}	Path to the project, or project name on server and optional scenario within that project.	`true or false`
/saveas?instance={instance}&project={projectNewName}&folderPath={folderPath}	Save the loaded project after the run is complete to the specified path and file. The string argument cannot be empty, and the path provided must be an existing directory and have saving permissions for the current user.	`true or false`
/units?instance={instance}&si={useSIUnits}	Ignore units set within project and output base SI units	`true or false`
/project?instance={instance}	Returns the project file an scenario name of the loaded project	`{ "Project" : "projectName" "Scenario" : "scenarioName" }`

/run?instance={instance}	Executes the active scenario from its current location to the end	`true or false`
/step?instance={instance}	Moves forward 1 time step in the active scenario	`true or false`
/getStep?instance={instance}	Returns the current step and the total number of steps in the active scenario	`{ "Current" : 0 "Total" : 100 }`
/metaParameters?instance={instance}	Get the active Functions (also calledmeta-parameter for calibration)	`true or false`
/metaParameter?instance={instance}&name={parameterName}&value={value}	Set the value of a Function (also calledmeta-parameter for calibration)	`true or false`

/allTimeSeries?instance={instance}	Returns all the result timeseries as an array
	
	A Timeseries jSON object contains
	
	Metadata - Key value array
	
	Name - name of timeseries
	
	Units - units string of timeseries
	
	Readings - array of Time, Result objects
		[
		{
		"Metadata": [
		{
		"Key" : "RunTime",
		"Value" : "16/08/2016 2:22:22 PM"
		},
		{
		"Key" : "ScenarioName",
		"Value" : "Scenario 1"
		}
		
		],
		
		"Name": "Gauge\\Gauge 16\\Downstream Flow\\Flow",
		"Readings": [
		{
		"Result": 0.2,
		"Time": "/Date (631112400000+1100)/"
		},
		{
		"Result": 0.2,
		"Time": "/Date (631198800000+1100)/"
		}
		],
		
		"Units": "ML/d"
		
		},
		
		{
		"Metadata": [
		{
		
		"Key" : "RunTime",
		"Value" : "16/08/2016 2:22:22 PM"
		},
		{
		"Key" : "ScenarioName",
		"Value" : "Scenario 1"
		}
		
		],
		
		"Name": "Gauge\\Gauge 16\\Rules Based Orders\\Rules Based Orders@Orders",
		"Readings": [
		{
		"Result": 0,
		"Time": "/Date (631112400000+1100)/"
		},
		{
		"Result": 0,
		"Time": "/Date (631198800000+1100)/"
		}
		],
		"Units": "ML"
		}
		]

/timeSeries?instance={instance}&name={parameterName}	Returns a specific timeseries
	
	A Timeseries jSON object contains
	
	Metadata - Key value array
	
	Name - name of timeseries
	
	Units - units string of timeseries
	
	Readings - array of Time, Result objects
		<pre>{
		"Metadata": [
		{
		"Key" : "RunTime",
		"Value" : "16/08
		/2016 2:22:22 PM"
		},
		{
		"Key" :
		"ScenarioName",
		"Value" : "Scenario
		1"
		}
		
		],
		
		"Name":
		"Gauge\\Gauge
		16\\Downstream
		Flow\\Flow",
		"Readings": [
		{
		"Result": 0.2,
		"Time": "/Date
		(631112400000+1100)
		/"
		},
		{
		"Result": 0.2,
		"Time": "/Date
		(631198800000+1100)
		/"
		}
		],
		
		"Units": "ML/d"
		
		}</pre>

/timeStepValue?instance={instance}&name={parameterName}	returns the last value of a particular timeseries	`double value eg. 1.2`
/reset?instance={instance}	Reset the loaded project.	`true or false`
/finish?instance={instance}	Destroys the service associated with the specified instance identifer	`true or false`

An example session of http requests and responses might look like this

Request	Response
`http://localhost:8001/Source/create`	`1`
`http://localhost:8001/Source/load?instance=1&project=AllNodes.rsproj`	`true`
`http://localhost:8001/Source/run?instance=1`	`true`

http://localhost:8001/Source/allTimeSeries?instance=1	[
	{
	"Metadata": [
	{
	"Key" : "RunTime",
	"Value" : "16/08/2016 2:22:22 PM"
	},
	{
	"Key" : "ScenarioName",
	"Value" : "Scenario 1"
	}
	],
	"Name": "Gauge\\Gauge 16\\Downstream Flow\\Flow",
	"Readings": [
	{
	"Result": 0.2,
	"Time": "/Date(631112400000+1100)/"
	},
	{
	"Result": 0.2,
	"Time": "/Date(631198800000+1100)/"
	}
	],
	"Units": "ML/d"
	},
	{
	"Metadata": [
	{
	"Key" : "RunTime",
	"Value" : "16/08/2016 2:22:22 PM"
	},
	{
	"Key" : "ScenarioName",
	"Value" : "Scenario 1"
	}
	],
	"Name": "Gauge\\Gauge 16\\Rules Based Orders\\Rules Based Orders@Orders",
	"Readings": [
	{
	"Result": 0,
	"Time": "/Date(631112400000+1100)/"
	},
	{
	"Result": 0,
	"Time": "/Date(631198800000+1100)/"
	}
	],
	"Units": "ML"
	}
	]

http://localhost:8001/Source/finish?instance=1	true