Spider9 Energy Management System

Table of Contents

Introduction

The Spider 9 software comprises an array of functionality including:

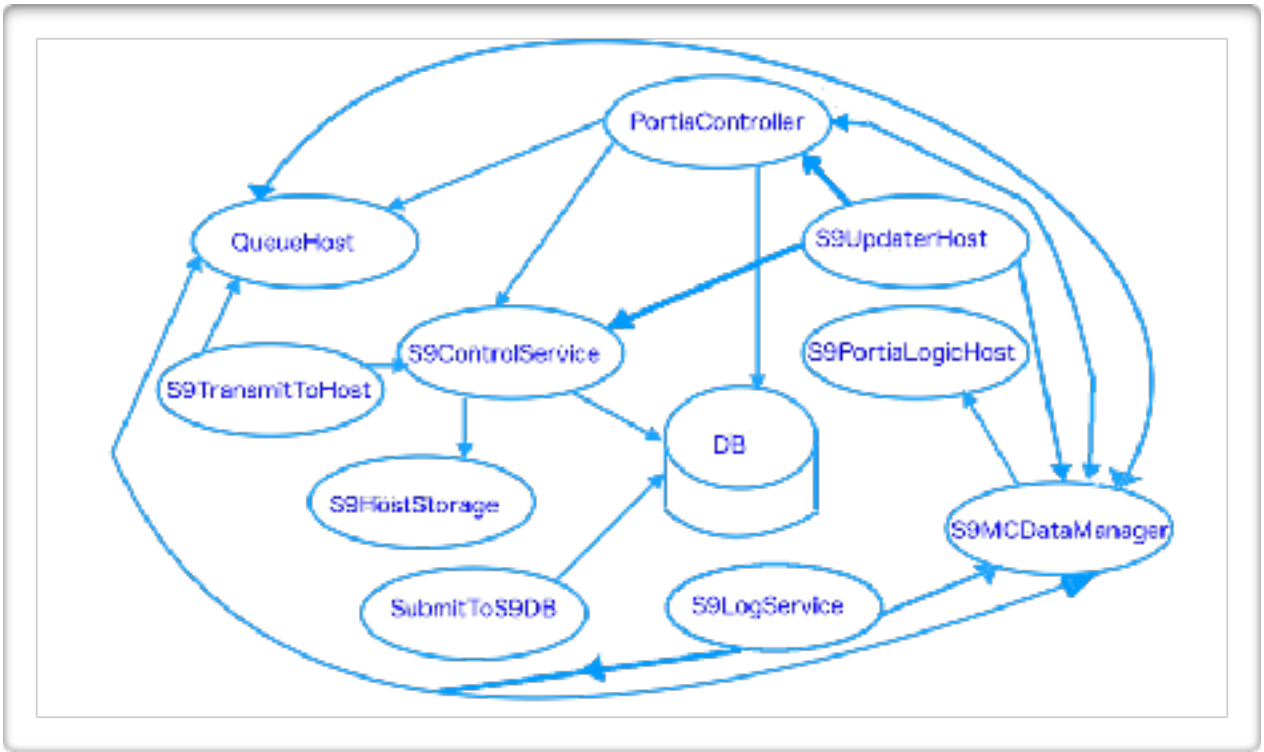
1. management of the voltage of the batteries at the cell level,
2. a driver for the inverter,
3. drivers to read sensor data from the batteries,
4. recording of sensor data in the database,
5. scheduling of charge/discharge cycles according to the rates of the costs of power from the grid to which the battery pack is connected.

This document aims to describe the technical details of the Energy Management portions in the Spider 9 software, specifically item 5 above. Notes on the other items may appear to provide context.

Concepts

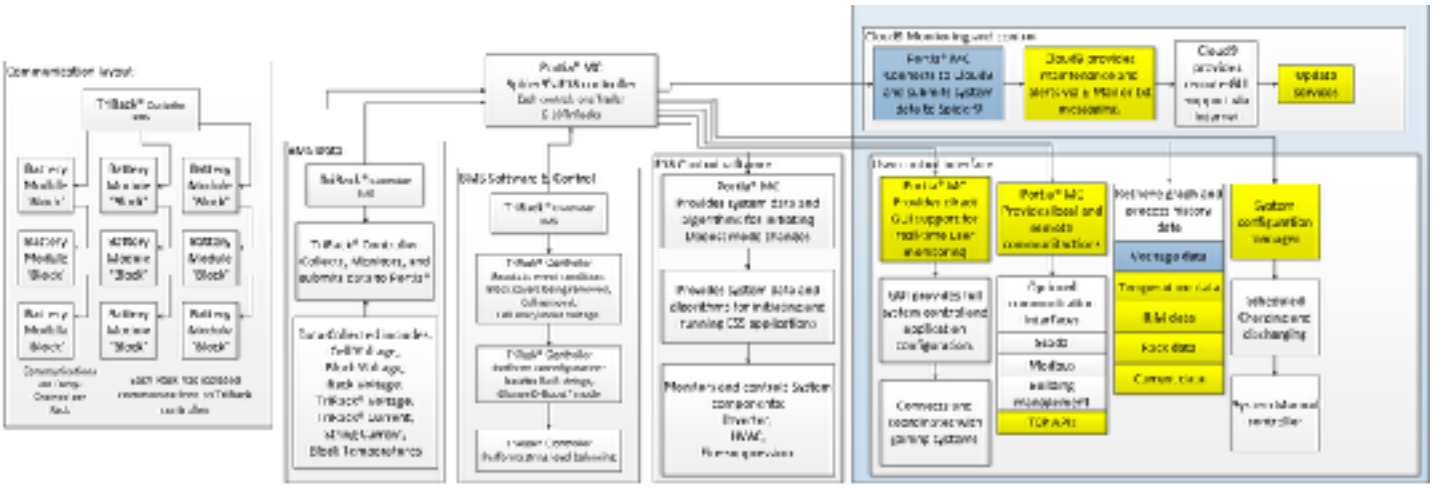
The Spider9 system is a collection of executables running as services in Windows™. The services communicate over sockets. Here is a rough topology:

- S9HostStorage
- NIControlAndMonitorService
- S9LogService
- S9MCDataManager
- S9PortiaLogic
- PortiaController
- S9QueueHost
- S9Updater
- PowerSenseConnect
- SubmitToS9DB
- S9TransferToHostService



SOCKET CONNECTIONS BETWEEN SPIDER9 SERVICES.
Arrows indicate origin and target of the connection.

The following diagram from “Portia System and code overview” offers useful illustration of the logical layout of the system, including batteries, battery management system (BMS), energy storage system(?) (ESS).



For each type of hardware device, eg. inverter, solar panel, there is a separate Windows™ service which acts as the driver through which the Portia Controller communicates. The following are the names of services for Inverters and other devices.

- ECControlService - Cubloc EC Driver
- EDAInverter - EDA Modbus
- InverterController - EDA Driver
- InverterController - GTIB Driver
- InverterDebugService
- InverterDrivers - Inverter Controls DRI-10 driver
- InverterController
- ECControlService - Dummy EC driver
- DummyInverter - DummyInverter driver
- DummySolarDriver - DummySolar driver
- DynaPowerInverterService - Dynapower Inverter driver
- EPCInverterService - EPC Inverter driver
- GeneratorService - Generator Service driver
- ECControlService - Haiti EC driver
- HybridDummyInverter - Hybrid - DummyInverter driver
- ECControlService - Lead Acid EC driver
- RhombusInverterController - Rhombus Inverter driver
- SatConInverterController - SatCon Inverter driver
- SolarDriver - SolarUpdate driver
- SharkDriverService - Shark driver

The BMS service SubmitToS9DB, periodically, on a timer, receives the following status data, (and more) from a queue, and logs it to the database:

- Cell data
- Rack data
- RIM data
- Block data
- Inverter data
- MIC data
- Temperature data
- Notification data
- Portia data
- LogFR data

As it logs the status data to the DB, it simultaneously sets it in a storage service (S9MCDataManager). Throughout runtime operation of the Portia system, the storage service provides values of the elements of this status data to the Power logic and Schedule logic, amongst other modules.

Databases

There are 4 databases in the Spider9 system:

```
/// Database for the Host to queue incoming data
S9HostQueue,
/// Database for the Master controller to queue outgoing data
S9MCQueue,
/// Database for the Master controller to store local data
S9LocalStorage,
/// V1.0 Host Queue Database for the Host to queue incoming data
S9Queue
```

The document “Portia System and code overview” explains the interaction with the databases in the Spider 9 system:

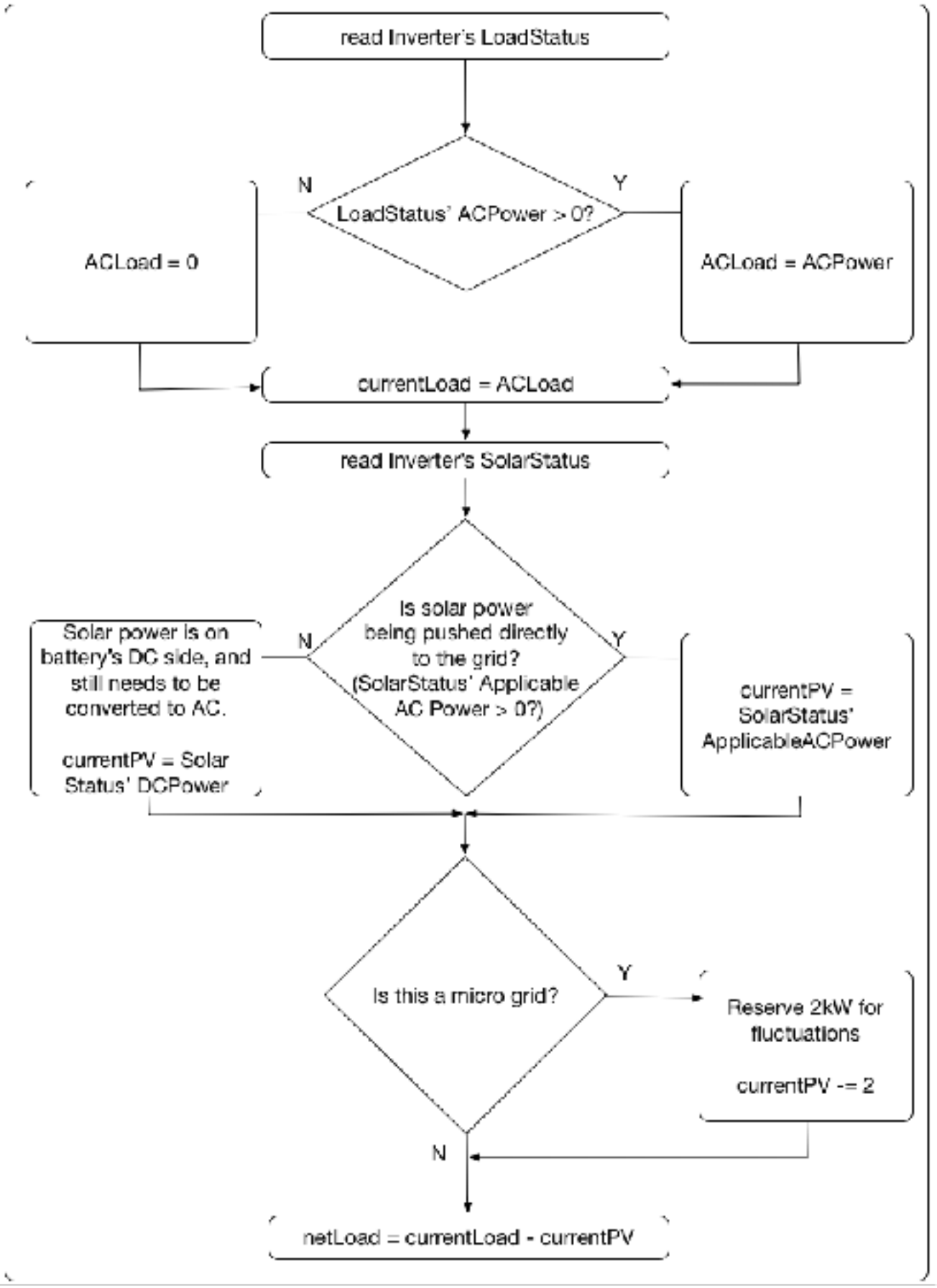
> “The data manager is constantly being updated with the latest data from the system. The logger will tick and grab the current data, and send it off to the Queue service. The queue service will save the data either in a MSMQ or SQL server database. The transmit to S9Host service will grab the data that the Queue service stored and transmit it to our host computer to be logged permanently. The logic service acts independent of these services. It is responsible for the systems schedule, and reacting to over/under triggers.”

CalculateNetLoad

The system calculates the net load as the kilowatts being drawn from the battery. It calculates this by reading the amount of solar power on the AC side of the inverter. It queries the inverter for this information by requesting the ApplicableACPower from the inverter's driver. This value is then subtracted from the LoadPowerReading. LoadPowerReading is also read from the inverter's driver.

See the source for the methods ACLoad, CalculateNetLoad and GetSolarApplicableACPower in DLLs/Logic Classes/Methods.cs. For the inverter drivers, see the directory Services/Drivers.

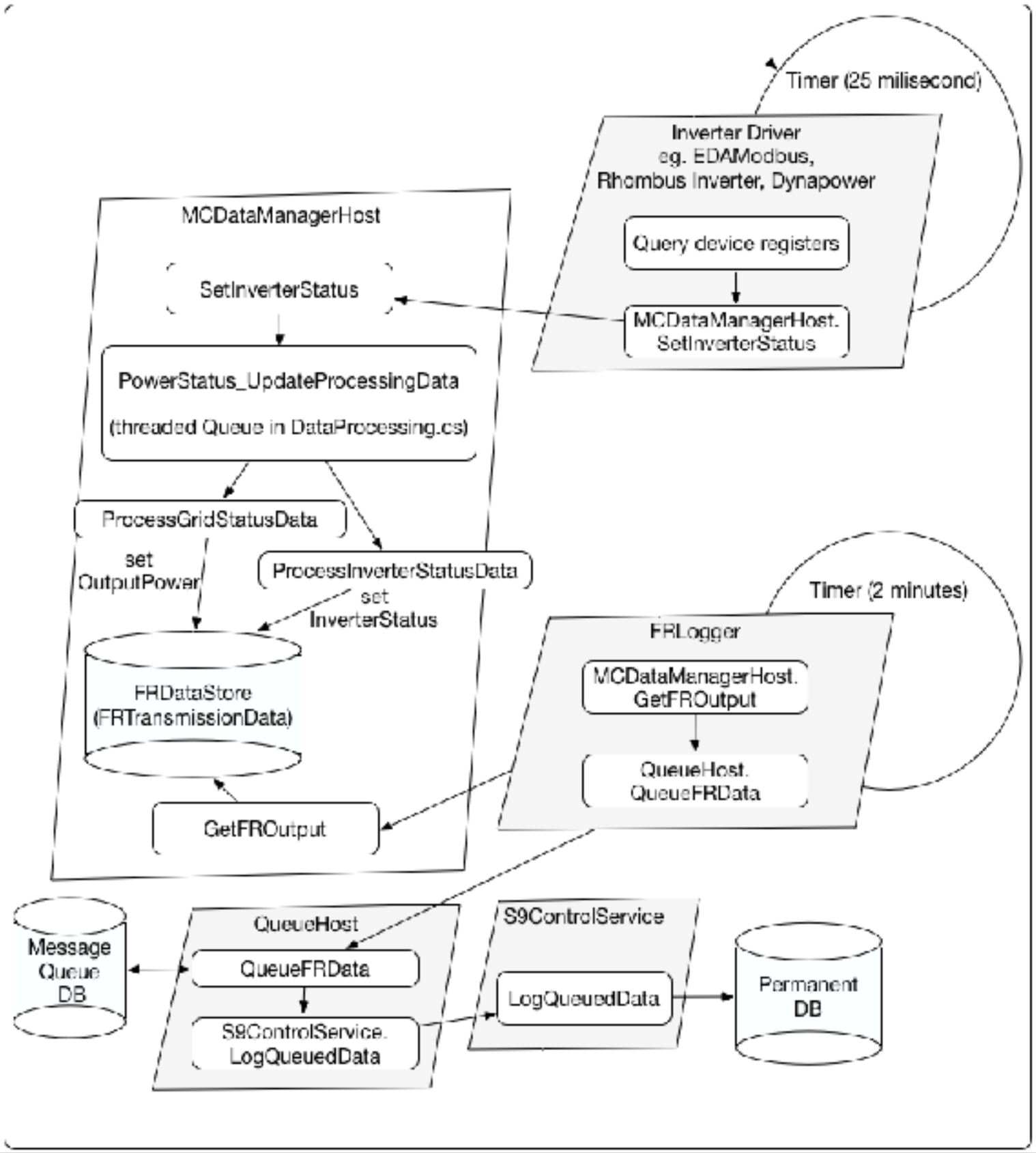
CalculateNetLoad



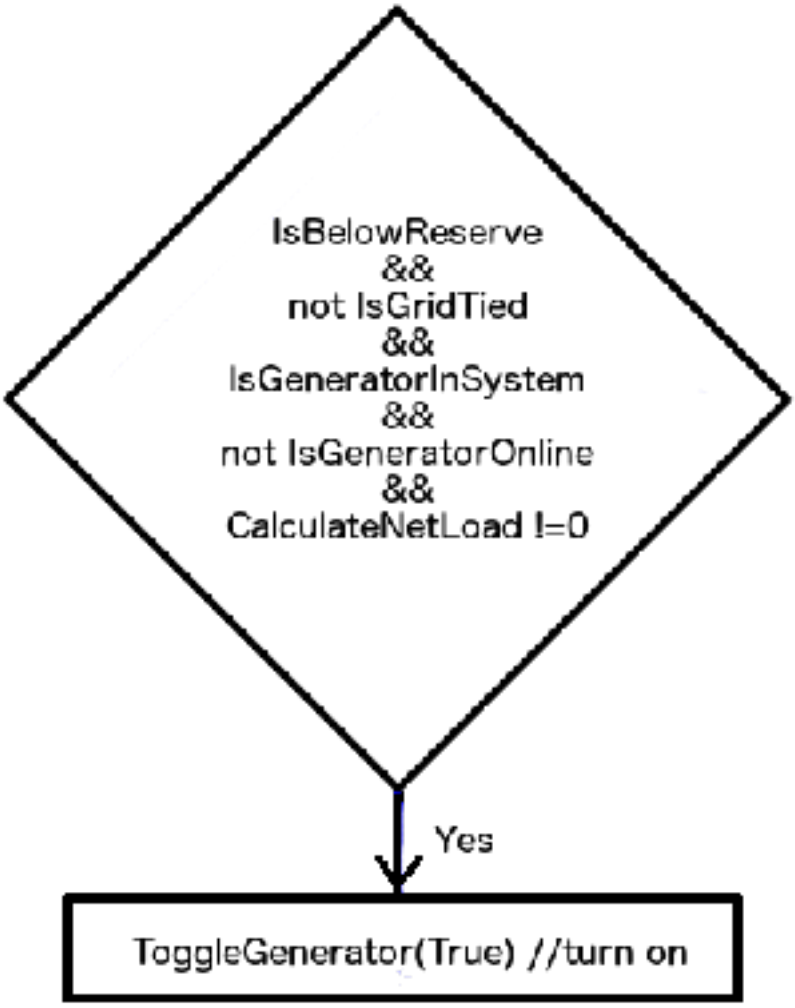
Frequency Regulation

When requested by an administrator (eg. through the UI) or an application (eg. grep the Test directory for FRSignal), the Spider9 system will perform frequency regulation. This causes a timer set for every two minutes to

FrequencyRegulation



ProcessBackupMode



Terminology

PV - Photovoltaic cells, commonly known as solar panels

SOC - State Of Charge of a battery, in %. The system will limit power if its below (eg.) 60% State-Of-Charge or above (eg.) 90%. This only limits if the requested power is greater than the limit. If a charge cycle is too long and the battery gets to 100% State-Of-Charge the system will simply remain at 0 power.

If a discharge is below 60% it'll start limiting power if the requested is greater than the limit. If discharge is near 0% State-Of-Charge power will be reduced to 0 and any further requests for discharge will result in a 0 power response.

FR - Frequency Regulation. An administrator or app may request frequency regulation, and Spider 9 will set the power (kW) of the inverter accordingly. The system may also opt out of performing frequency regulation for a variety of reasons:

```
//from “Golbal Enumerators.cs” (yes, the file name is misspelt)

        public enum OptOutReason {
           /// Temperature rise could possibly restrict system performance.
           Temperature,
           /// The inverter is not responding or is offline for some reason.
           Inverter,
           /// The manual disconnect appears to be open.
           Disconnect,
           /// SOC is out of the operating range of Frequency Regulation.
           Soc,
           /// The Frequency Regulation signal has not been updated for more
then 2 minutes.
           FRSignal,
           /// At least one rack is offline and will restrict Frequency Regula-
tion performance.
           Rack,
           /// A manual signal has been received to opt out.
           Manual,
           /// A manual request for power changed the system operation mode out
of Frequency Regulation
           Mode,
           /// The Power On button was disengaged and the system is offline.
           POButton,
           /// Opt out while system charges to an acceptable range.
           Charging,
           /// Opt out while system discharges to an acceptable range.
```

Example: If the Max FR power is 100kW and the FR signal is 0.10 then the system will adjust power to be close to 10kW +- about 0.1kWatts.

As long as the requested power is within the system power limit (based on system voltage) then the actual power will match the requested FR signal power within the inverter tolerance.

See PowerLogic.cs, wherein the method FRPowerRequest will set the requested system power when Frequency Regulation (FRSignal) is enabled. FRPower

DTG - Date Time Group

S9Filter - is an important utility class used throughout the Spider9 system for collecting and averaging sensor data. It is internally an array of 64-bit integers which is averaged with outlier values filtered, rejected and restricted. See S9Filter.cs, and SavedMaxPower in PowerLogic.cs for an example of application.

RIM - Rack Interconnect Module. Battery packs are housed in racks, and the racks are connected together and managed by a Rack Interconnect (Interface) Module. Status data is reported at each level: Cells, Racks, RIM, Block and Inverter.

SL Mode - Short and Large mode. Short racks and Large racks of the RIM are separate/unconnected. This is the Normal mode.

DBoost Mode - Short and Large racks are connected together to boost voltage of the RIMMax-ChargeLimit - Is the maximum power permitted during charging of the system.

System Power - The scheduler, administrator, or an application, request power. On such request, the system determines the system's present maximum power. If requested power is below the max, it will set power to this level. If requested power is greater than max power, it determines whether or not it's safe to raise the maximum. If it can't raise the maximum, it sets requested power to the maximum. After determining what power it is going to set, it will either change power right away or do it in steps if current power and requested power are not close. Logic will remember what power was requested and if the max is lowered for some reason, it will eventually try to bring power back to what was originally requested. It returns the value to which that power has been set

MaxPower - The min of CalculatedMax, MaxPowerAvailable, maxCharge/discharge limit, and requested power, as an absolute value (kW). It is returned from GetMaxPower in PowerLogic.cs

SavedMaxPower - MaxPower as described above, and saved. It is an S9Filter object.

SolarRampRate - The RampRate to which solar firming will limit change in PV, in Watts/min. The system will try to offset the rate of change of the grid by supplementing it with the battery. So if a cloud were to pass by, instead of a sharp drop in power output, the battery will provide some power to gradually lower it.

SolarRampLimit

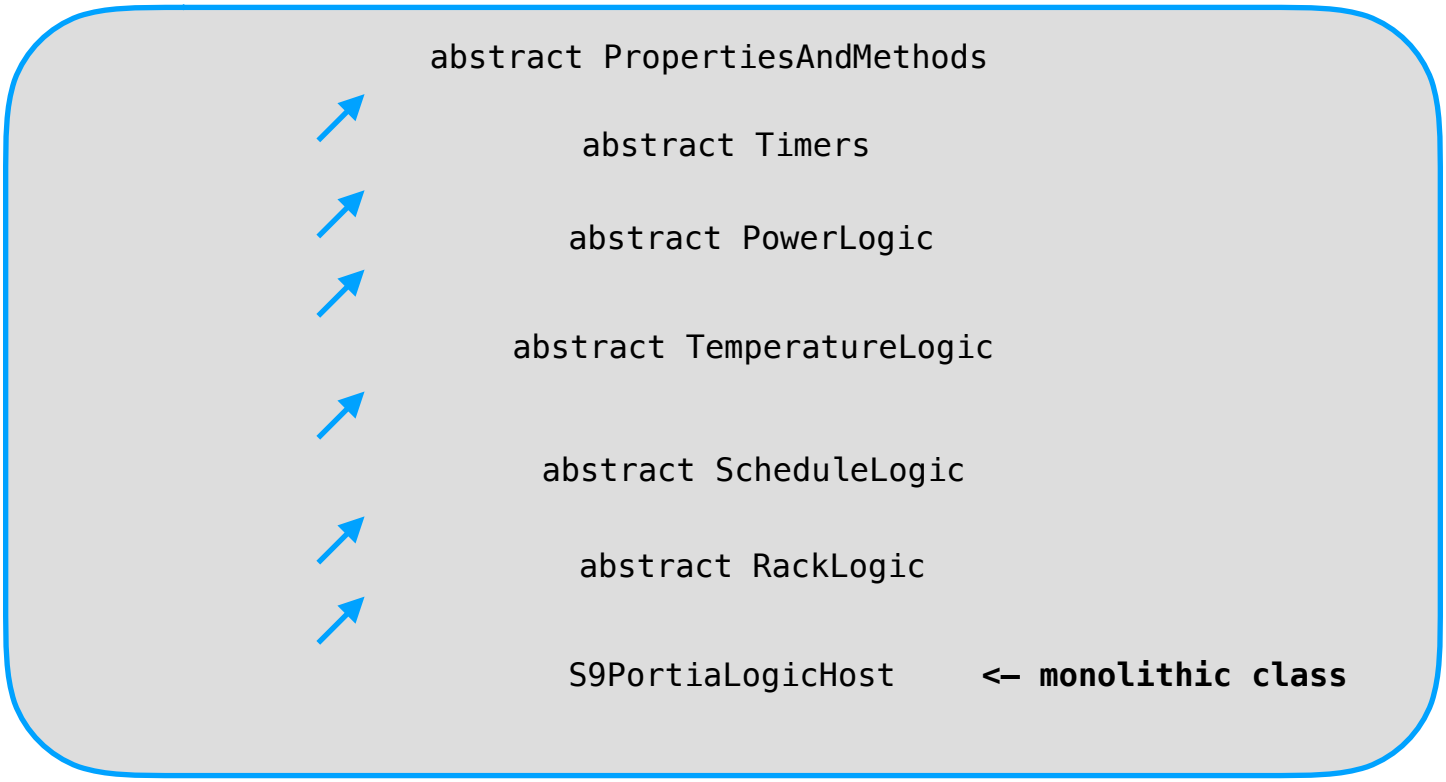
PVPowerReading.DCPower - the power in kW at the DC port of the photovoltaic cells

PVPowerReading.ApplicableACPower - kW from the photovoltaic cells which is being pushed directly to the grid (not being stored in the battery)

CellsAboveMax

CellsBelowMin

Logic Hierarchy:



Scheduling

The system was designed to be configured with a schedule according to the variable rates for power provided by the grid. The general data for which the schedule logic operates is managed in a ScheduleData structure.

DLLs/S9Schedule/ScheduleData.cs

```
 1:    9    public class ScheduleData
 2:   19         public ChargeState State { get; set; }
 3:   25         public TimeSpan EndOfWindow { get; set; }
 4:   30         public float MaxPowerKW { get; set; }
 5:   35         public float DischargeAmount { get; set; }
 6:   40         public float PeakShavingChargeLimit { get; set; }
 7:   45         public float PeakShavingDischargeLimit { get; set; }
 8:   50         public bool DoPriceShifting { get; set; }
 9:   55         public bool DoLoadFollowing { get; set; }
10:   60         public bool DoPeakShaving { get; set; }
11:   65         public bool DoSolarFirming { get; set; }
12:   70         public float GridLimit { get; set; }
```

As documented in the code, and further in that for TimeSlot, (see below), the fields of ScheduleData are as follows:

State: The current state of charge for this schedule period; Either, Charging, Discharging, Neither, Maintain. State defaults to Either in the ScheduleData structure.
EndOfWindow: The time until the end of the Current Window. Only matters if current charge state is Charging. Used to calculate the power needed to allow the cells to use a full charge cycle for balancing. TimeSpan represents a time interval, and is part of the C# platform.
MaxPowerKW: The max power for the Current Window. Default is 100 kW. This is an absolute value and must always positive.
DischargeAmount: The amount in kW that the system will discharge when an application requests discharge. Used in Peak Shaving, PV Firming, and Price Shifting
PeakShavingChargeLimit: While PeakShaving is enabled, when Load is below this value (kW), the system will Charge.
PeakShavingDischargeLimit: While PeakShaving is enabled, when Load is above this value (kW), the system will Discharge.
DoPriceShifting: if true, the system is following a schedule
DoLoadFollowing: if true, the system will try and maintain a specific net load
DoPeakShaving: if true the system will perform peak shaving
DoSolarFirming: if true the system will perform solar firming
GridLimit: Schedule logic will never allow the grid to go above this limit in kW
DesiredSoc: Desired SOC that the system will try to maintain during SOC maintenence. Expressed as a percent, i.e. 50 == 50%

The schedule is made up of an array of WeekDays, the first element, (element 0) being Sunday.

DLLs/S9Schedule/Schedule.cs

```
    public Schedule()
    {
        Week = new Weekday[7] { new Weekday(), new Weekday(), new
Weekday(), new Weekday(), new Weekday(), new Weekday(), new Weekday() };
```

Each WeekDay consists of a list of non-overlapping TimeSlots.

DLLs/S9Schedule/TimeSlot.cs

```
  1:    9     public class Timeslot
  2:   19           public Guid Id { get; set; }
  3:   24           public int StartHour { get; set; }
  4:   28           public int StartMinute { get; set; }
  5:   32           public int EndHour { get; set; }
  6:   36           public int EndMinute { get; set; }
  7:   41           public ChargeState State { get; set; }
  8:   49           public float ScheduledChargeAmount { get; set; }
  9:   57           public float ScheduledDischargeAmount { get; set; }
 10:   62           public float PeakShavingChargeLimit { get; set; }
 11:   67           public float PeakShavingDischargeLimit { get; set; }
 12:   72           public bool DoPriceShifting { get; set; }
 13:   77           public bool DoLoadFollowing { get; set; }
 14:   82           public bool DoPeakShaving { get; set; }
 15:   87           public bool DoSolarFirming { get; set; }
 16:   92           public float GridLimit { get; set; }
 17:   97           public bool IsValid { get; set; }
 18:  102           public float DesiredSoc { get; set; }
```

Each TimeSlot structure has the following fields (as is well documented in the code):

ChargeState: The desired system state for this TimeSlot.
ScheduledChargeAmount: The maximum amount of power (kW) the system will use when the schedule requires a charge. If this is less than the system limit, it will use this amount, otherwise it will use the system limit. A scheduled charge will charge up to this amount (but may use less power to attempt to use the entire charge window. Price shifting will charge by this amount. Other applications will treat it as a maximum for charge requests.
ScheduledDischargeAmount: The maximum amount of power (kW) the system will use when the schedule requires a discharge. If this is less than the system limit, it will use this amount, otherwise it will use the system limit. A scheduled discharge will discharge at this amount. Price shifting will discharge by this amount. Other applications will treat it as a maximum for discharge requests.
PeakShavingChargeLimit: While PeakShaving is enabled, when Load is below this value (kW), the system will Charge.
PeakShavingDischargeLimit: While PeakShaving is enabled, when Load is above this value (kW), the system will Discharge.
DoPriceShifting: If true, the system will charge when price is low and discharge when price is high.
DoLoadFollowing: If true, the system will try and maintain the current net load.
DoPeakShaving: If true the system will discharge when load is high and charge when load is low with the intent of keeping the net load lower.
DoSolarFirming: If true the system will reduce the effect of spikes and drops in solar power.
GridLimit: Schedule logic will never allow the grid to go above this limit (kW)
IsValid: This value will be false until the start and end times are validated. If true, the times are within the required range and the end is after the start.
DesiredSoc: Desired State Of Charge that the system will try to maintain during SOC maintenance. Expressed as a percent, i.e. 50 == 50%

Solar Firming

Solar Firming is the process whereby power from the system is kept at a constant rate, regardless of spikes and drops in that supplied from the solar panels. The Spider9 system accomplishes this according to the following algorithm.

Peak Shaving

Price Shifting

Load Following

Appendices

Appendix 1 - Cell Data

CellData: These are the data maintained for each battery cell.

DLLs/PCL Transmission Classes/CellTransmissionData.cs:

```
        /// Date Time Group for this data point
        public ulong DTG
        /// Property to see if the stated property is being changed.  True if
changed.
        public bool DTGChanged
        /// [int] Percent
        /// State of Charge of the Cell in hundredths of a percent (100% SOC =
SOC/10000 = 1).
        public ushort SOC
        /// mAHrs available energy State of health.
        public int Capacity
        /// Property to see if the stated property is being changed.  True if
changed.
        public bool CapacityChanged
        /// current cell voltage in mVolts.
        public short Voltage
        /// Property to see if the stated property is being changed.  True if
changed.
        public bool VoltageChanged
        /// current cell current in mAmps.
        public int Ampere
        /// Property to see if the stated property is being changed.  True if
changed.
        public bool AmpereChanged
        /// State of health internal cell resistance in mOhms.
        public short Resistance
        /// Property to see if the stated property is being changed.  True if
changed.
        public bool ResistanceChanged
        /// Status of balance circuit:
        /// On = true : Off = false.
        public bool BalanceCircuitStatus
        /// Property to see if the stated property is being changed.  True if
changed.
        public bool BalanceCircuitStatusChanged
        /// Command sent to Force balancing.
        public Enumerators.Cell.ForceBalance ForceBalanceCommand
        /// Property to see if the stated property is being changed.  True if
changed.
        public bool ForceBalanceCommandChanged
```

```
        /// True if the cell voltage is over the max voltage limit
        public bool Over
        /// Property to see if the stated property is being changed.  True if
changed.
        public bool OverChanged
        /// True if the cell voltage is under the min voltage limit
        public bool Under
        /// Property to see if the stated property is being changed.  True if
changed.
        public bool UnderChanged
        /// Indicates if its time to update SharePoint.
        public bool DoUpdateSP
        /// Property to see if the stated property is being changed.  True if
changed.
        public bool DoUpdateSPChanged
        /// State of health of a cell in hundredths of a percent.
        public ushort SOH
        /// Property to see if the stated property is being changed.  True if
changed.
        public bool SOHChanged
        /// Means of virtual cycling. this is not logged
        public int SOCCapacity
        /// Property to see if the stated property is being changed.  True if
changed.
        public bool SOCCapacityChanged
        /// Cell Identification information.
        public IDs Info
```

Appendix 2 – Cell States

These are the states of a cell's 'Cycle'

DLLs/Algorithms/ToC_EOD.cs:

```
                switch (Cycle)

                    case Enumerators.Controller.Mode.Charge:
                        ProcessDuringChargeCycle(value);
                        break;
                    case Enumerators.Controller.Mode.Discharge:
                        ProcessDuringDischargeChargeCycle(value);
                        break;
                    case Enumerators.Controller.Mode.Offline:
                        ProcessWhileOffline(value);
                        break;
```

Appendix 3 – Logical State of a TimeSlot

These are the possible logical states that a TimeSlot can schedule.

DLLs/S9Schedule/ChargeState.cs

```
/// Enumeration for the charge states that are available for scheduling.


       public enum ChargeState
        {
                /// <summary>
                /// The Master Controller will allow charging and discharging;
whichever is requested.  This must be selected when using Apps.
                /// </summary>
                Either,
                /// <summary>
                /// The master controller will attempt to charge the cells if
they are below max voltage.
                /// </summary>
                Charging,
                /// <summary>
                /// The master controller will attempt to discharge the cells if
they are above min voltage.
                /// </summary>
                Discharging,
                /// <summary>
                /// The master controller will not allow any charging or dis-
charging.
                /// </summary>
                Neither,
                /// <summary>
                /// Maintains a specific SOC. Discharges when above the SOC,
charges when below.
                /// </summary>
                Maintain
```

Appendix 4 – Rack Data

DLLs/Services/AccessData.cs

```
  1:  296                     if (_cellData[info.RIMId][info.Rack][info.BlockId]
[info.MICId][info.CellIndex].isDataExpired)
  2:  344                     if (_micData[info.RIMId][info.Rack][info.BlockId]
[info.MICId].isDataExpired)
  3:  391                     if (_blockData[info.RIMId][info.Rack]
[info.BlockId].isDataExpired)
  4:  433                     if (_blockCells[info.RIMId][info.Rack][info.Block-
Id].isDataExpired)
  5:  482                     if (_rackData[info.RIMId][info.Rack].isDataExpired)
  6:  525                     if (_rimData[info.RIMId].isDataExpired)
```

```
  7:  559                     if (_onlineRackCount.isDataExpired)
  8:  586                     if (_inverterData[info.RIMId].isDataExpired)
  9:  623                     if (_powerStatus[statusItem].isDataExpired)
 10:  780                     if (_runningAvgPV.isDataExpired)
 11:  957                     if (_bmtData.isDataExpired)
 12:  976                     if (_price.isDataExpired)
 13:  998                     if (_usedRackNames.isDataExpired)
 14: 1043                     if (_usedRackNamesByMode.isDataExpired)
 15: 1088                     if (_customerImage.isDataExpired)
 16: 1110                     if (_xlsFile.isDataExpired)
 17: 1285                     if (_EPOStatus.isDataExpired)
 18: 1330                     if (_maxChargeLimit.isDataExpired)
 19: 1352                     if (_maxDischargeLimit.isDataExpired)
 20: 1375                     if (_autoCyclingEnabled.isDataExpired)
 21: 1393                     if (_rampRateControlEnabled.isDataExpired)
 22: 1411                     if (_isInBackupMode.isDataExpired)
 23: 1429                     if (_allRIMs.isDataExpired)
 24: 1476                     if (_notifications.isDataExpired)
 25: 1526                     if (_bmfrData.isDataExpired)
 26: 1546                     if (_hasChanges[info.RIMId].isDataExpired)
 27: 1745                 if (_LocalRIMdata.isDataExpired)
 28: 1778                         _LocalRIMdata.isDataExpired = true;
 29: 1848                     if (IsGeneratorOnline.isDataExpired)
```

Appendix 5

```
        private float DoSchedule(RIMData rim)
        {
            //rv is max power
            float RV = 0;
             // … more stuff
                if (!CurrentChargeState.DoLoadFollowing && !CurrentChargeState.-
DoPeakShaving && !CurrentChargeState.DoPriceShifting && !CurrentChargeState.DoSo-
larFirming)
                {
                    RV = RequestedPower;   <-- what is the purpose of RV here?
                    RV = DoEither(rim);
```

Appendix 6 – Methods of ScheduleData

DLLs/S9Schedule/ScheduleData.cs

```
  1:    9         public class ScheduleData
  2:   19         public ChargeState State { get; set; }
  3:   25         public TimeSpan EndOfWindow { get; set; }
  4:   30         public float MaxPowerKW { get; set; }
  5:   35         public float DischargeAmount { get; set; }
  6:   40         public float PeakShavingChargeLimit { get; set; }
  7:   45         public float PeakShavingDischargeLimit { get; set; }
  8:   50         public bool DoPriceShifting { get; set; }
  9:   55         public bool DoLoadFollowing { get; set; }
```

```
 10:   60          public bool DoPeakShaving { get; set; }
 11:   65          public bool DoSolarFirming { get; set; }
 12:   70          public float GridLimit { get; set; }
 13:   75          public float DesiredSoc { get; set; }
```

```
                                          //time-slot found
                                           RV.State = slot.State;
                                           RV.MaxPowerKW = slot.ScheduledCharge-
Amount;
                                           RV.PeakShavingChargeLimit = slot.Peak-
ShavingChargeLimit;

public ScheduleData GetCurrentWindow(DateTime time, float maxPowerKW)
```

```
    RV.DischargeAmount = slot.ScheduledDischargeAmount;
    RV.PeakShavingDischargeLimit = slot.PeakShavingDischargeLimit;
    RV.DoLoadFollowing = slot.DoLoadFollowing;
    RV.DoPriceShifting = slot.DoPriceShifting;
    RV.DoPeakShaving = slot.DoPeakShaving;
    RV.DoSolarFirming = slot.DoSolarFirming;
    RV.GridLimit = slot.GridLimit;
    RV.DesiredSoc = slot.DesiredSoc;
    timeSlotFound = true;
```

Appendix 7 – Complete List of Service Executables

This is a list of the service executables (daemons) which comprise the runtime Portia system:

```
grep -r --include \*.cs InstanceContextMode ./

.//Clients/Debug Viewer/WcfServiceLibrary1/DebugHost.cs:    [ServiceBehaviorAt-
tribute(InstanceContextMode = InstanceContextMode.Single)]
.//Clients/PowersenseClient/PowersenseServices/PowersenseService.cs:    [Service-
Behavior(InstanceContextMode = InstanceContextMode.Single)]
.//DLLs/DataManagerClasses/MCDataManagerHost.cs:    [ServiceBehavior(InstanceCon-
textMode = InstanceContextMode.Single)]
.//DLLs/DebugLibrary/DebugHost.cs:    [ServiceBehaviorAttribute(InstanceContext-
Mode = InstanceContextMode.Single)]
.//DLLs/Logic Classes/S9PortiaLogicHost.cs:    [ServiceBehaviorAttribute(In-
stanceContextMode = InstanceContextMode.Single)]
.//DLLs/Services/AccessConfiguration.cs:    [ServiceBehaviorAttribute(Instance-
ContextMode = InstanceContextMode.Single)]
.//DLLs/Services/AccessControl.cs:    [ServiceBehaviorAttribute(InstanceContext-
Mode = InstanceContextMode.Single)]
.//DLLs/Services/AccessData.cs:    [ServiceBehaviorAttribute(InstanceContextMode
= InstanceContextMode.Single)]
```

```
.//DLLs/Services/GeneratorController.cs:    [ServiceBehaviorAttribute(Instance-
ContextMode = InstanceContextMode.Single)]
.//DLLs/Services/QueueHost.cs:    [ServiceBehaviorAttribute(InstanceContextMode =
InstanceContextMode.PerCall)]
.//DLLs/Services/S9UpdaterHost.cs:    [ServiceBehaviorAttribute(InstanceContext-
Mode = InstanceContextMode.Single)]
.//DLLs/Services/srInverterControls.cs:    [ServiceBehaviorAttribute(InstanceCon-
textMode = InstanceContextMode.Single)]
.//DLLs/Services/srPortiaController.cs:    [ServiceBehaviorAttribute(InstanceCon-
textMode = InstanceContextMode.Single)]
.//DLLs/Services/srRIMControls.cs:    [ServiceBehaviorAttribute(InstanceContext-
Mode = InstanceContextMode.Single)]
.//Hosts/Spider9Host - V1.0/S9Host Storage/S9HostStorageService.cs:    [Service-
BehaviorAttribute(InstanceContextMode = InstanceContextMode.Single)]
```

Appendix 9

The command below will generate an exhaustive list of the targets(projects) in the “PC RIM Data” project within Team Foundation Server:

```
grep -r --include \*.csproj AssemblyName ./ | gsed "s/^\
([^:]*\).*AssemblyName>\([^<]*\)<.*/\1:\2/"
```

Notes

Solar firming:

From here: http://new.abb.com/substations/energy-storage-applications/capacity-firming

Capacity firming
The variable, intermittent power output from a renewable power generation plant, such as wind or solar, can be maintained at a committed level for a period of time.
The energy storage system smoothes the output and controls the ramp rate (MW/min) to eliminate rapid voltage and power swings on the electrical grid.

Spent an hour to discover the enums in these two files differ, but are deceptively similar:

```
 31 #    "DLLs/PCL Transmission Classes/GolbalEnumerators.cs" line 1447
 32 %a   "APIs/DataManagerClientAPI/DataManagerClientPublicApi/PublicEnums.cs"
line 360
```

Specifically, PowerStatus is only in the former. PowerStatus is used in DoSolarFirming(), in DLLs/Logic Classes/ScheduleLogic.cs
A list of ServiceNames is also in GolbalEnumerators.cs

GetMaxPower sets SavedMaxPower in PowerLogic.cs

To find the dlls upon which a project depends, grep for HintPath in the .csproj file:

```
grep HintPath Hosts/Spider9Host\ -\ V1.0//Spider9Host//Spider9Host.csproj

      <HintPath>..\..\..\DLLs\Compiled\Clients Interfaces.dll</HintPath>
      <HintPath>..\..\..\DLLs\Compiled\Clients.dll</HintPath>
      <HintPath>..\..\..\DLLs\Compiled\Common Namespace.dll</HintPath>
      <HintPath>..\..\..\DLLs\Compiled\Database Namspace.dll</HintPath>
      <HintPath>..\..\..\DLLs\Compiled\DBAccess.dll</HintPath>
      <HintPath>..\..\..\DLLs\Compiled\DebugLibrary.dll</HintPath>
      <HintPath>..\..\..\DLLs\Compiled\MCDBAccess.dll</HintPath>
      <HintPath>..\packages\Microsoft.Data.Edm.5.0.0\lib\net40\Microsoft.Data.Edm.dll</HintPath>
      <HintPath>..\packages\Microsoft.Data.OData.5.0.0\lib\net40\Microsoft.Data.OData.dll</HintPath>
      <HintPath>..\packages\Microsoft.Data.Services.Client.5.0.0\lib\net40\Microsoft.Data.Services.Client.dll</HintPath>
      <HintPath>..\..\..
\DLLs\Compiled\Microsoft.Deployment.WindowsInstaller.dll</HintPath>
      <HintPath>..\..\..\DLLs\Compiled\Notifications.dll</HintPath>
      <HintPath>..\..\..\DLLs\Compiled\PCL Transmission Classes.dll</HintPath>
      <HintPath>..\..\..\DLLs\Compiled\S9Classes.dll</HintPath>
      <HintPath>..\..\..\DLLs\Compiled\Service Interfaces.dll</HintPath>
      <HintPath>..\..\..\DLLs\Compiled\SmBMSCore.dll</HintPath>
      <HintPath>..\..\..\DLLs\Compiled\StaunchCore.dll</HintPath>
      <HintPath>..\packages\System.Spatial.5.0.0\lib\net40\System.Spatial.dll</
HintPath>
```

Here are the list of files I used thus far in code investigation:

```
:ls
  2      "Services/PortiaController/PortiaController/PortiaController.cs" line 45
  3      "DLLs/Logic Classes/ScheduleLogic.cs" line 130
  4      "DLLs/Logic Classes/TemperatureLogic.cs" line 62
  5      "DLLs/S9Schedule/Schedule.cs"  line 103
  6      "DLLs/Logic Classes/Methods.cs" line 350
  7      "DLLs/Logic Classes/PowerLogic.cs" line 64
  8      "DLLs/ToolChest/DataBuffer.cs" line 14
  9      "DLLs/S9Schedule/ScheduleData.cs" line 28
 10      "DLLs/S9Schedule/WeekDay.cs"   line 69
 11      "DLLs/Algorithms/ToC_EOD.cs"   line 68
 12      "DLLs/LogicClasses/PhysicalRackconnection.cs" line 609
 13      "DLLs/LogicClasses/MeasurementIC.cs" line 107
 14      "DLLs/LogicClasses/CellCommsLogic.cs" line 69
 15      "DLLs/Win8 S9 Classes/CellData.cs" line 83
 16      "DLLs/Non Portable/S9 Classes/S9 Classes/CellData.cs" line 132
 17      "DLLs/PCL Transmission Classes/CellTransmissionData.cs" line 120
 18      "DLLs/S9Schedule/TimeSlot.cs"  line 1
 19      "DLLs/S9Schedule/ChargeState.cs" line 7
 20      "DLLs/PCL Transmission Classes/TimeTracking.cs" line 1
 21      "DLLs/Logic Classes/Timers.cs" line 37
 22      "DLLs/Non Portable/S9 Classes/S9 Classes/PowerStatusData.cs" line 347
 23      "Test/PriceShifting/PriceShifting/Program.cs" line 29
 24      "DLLs/Logic Classes/S9PortiaLogicHost.cs" line 309
 25      "Services/Portia Logic/S9PortiaLogic.cs" line 60
 26      "DLLs/Non Portable/S9 Classes/S9 Classes/S9 Classes.csproj" line 80
 27      "DLLs/Services/srPortiaController.cs" line 69
 28      "DLLs/Service Interfaces/IsrPortiaController.cs" line 86
 29      "DLLs/PCL Transmission Classes/GolbalEnumerators.cs" line 4334
 30      "APIs/DataManagerClientAPI/DataManagerClientPublicApi/PublicEnums.cs"
line 360
 31      "DLLs/Win8 S9 Classes/InverterData.cs" line 1
 32      "DLLs/Non Portable/nonPCL-WP8 S9 transmission Classes/GolbalEnumera-
tors.cs" line 1280
 33      "DLLs/Logic Classes/PropertiesAndFields.cs" line 46
 34      "DLLs/DataManagerClasses/MCDataManagerHost.cs" line 21
 35      "DLLs/Clients/PortiaLogicClient.cs" line 627
 36      "DLLs/Clients/ClientBase.cs"   line 135
 37      "DLLs/Clients/Service References/srPortiaLogic/Reference.cs" line 236
 38      "Services/Drivers/SatConDrivers/SatConDrivers.cs" line 69
 39      "DLLs/Logic Classes/RackLogic.cs" line 720
 40      "DLLs/DataManagerClasses/CellLogic_DataProcessingAndMethods.cs" line 224
 41      "DLLs/DataManagerClasses/CellLogic_EventHandlingAndTrigerProcessing.cs"
line 41
 42      "DLLs/DataManagerClasses/DMBase_EventsHandlers.cs" line 1
 43      "DLLs/DataManagerClasses/DMBase_EventsTriggers.cs" line 147
 44      "DLLs/Non Portable/S9 Classes/S9 Classes/S9Filter.cs" line 272
 45      "DLLs/Clients/DataManagerClient.cs" line 1
 46      "Services/SubmitToS9DB/SubmitToS9DB/LogData.cs" line 704
```

```
 47      "Services/SubmitToS9DB/SubmitToS9DB/SubmitToDBHelper.cs" line 17
 48      "DLLs/Clients/HostStorageClient.cs" line 202
 49      "Services/MC Datamanager Gen 2.0/MC Logger Gen 2.0/MCDataManager.cs"
line 67
 51      "Hosts/Spider9Host - V1.0/Storage Classes/RackStorage.cs" line 1
 54      "Services/TransmitToS9Host/TransmitToS9Host.cs" line 144
 55      "Services/TransmitToS9Host/S9TTHService.cs" line 42
 57      "DLLs/Clients/S9HostClient.cs" line 69
 62      "Hosts/Spider9Host - V1.0/Spider9Host/S9ControlService.svc.cs" line 1266
 65      "Services/Portia Logic/Program.cs" line 1
 66      "Services/Portia Logic/App.config" line 1
 80      "Hosts/Spider9Host - V1.0/S9Host Storage/S9HostStorage.cs" line 46
 84      "Hosts/Spider9Host - V1.0/S9Host Storage/S9HostStorageService.cs" line
270
 85      "Hosts/Spider9Host - V1.0/Storage Classes/RIMStorage.cs" line 72
 88      "Services/SubmitToS9DB/SubmitToS9DB/SubmitToS9DB.cs" line 9
 94      "Services/Logging - Gen 1.0/Logging/S9LogService.cs" line 17
 95      "DLLs/PCL Win8 Clients/DataManagerClient.cs" line 144
100      "Services/QueueHost/S9QueueHost.cs" line 19
102      "Services/Updater/S9Updater.cs" line 475
106      "DLLs/Algorithms/DBoost.cs"    line 77
107      "DLLs/Non Portable/S9 Classes/S9 Classes/GlobalVariables.cs" line 606
108      "DLLs/Services/AccessConfiguration.cs" line 3553
109      "DLLs/Services/QueueHost.cs"   line 1
113      "DLLs/DBaccess/MCDBAccess/packages.config" line 1
118      "DLLs/DBaccess/MCDBAccess/Properties/AssemblyInfo.cs" line 35
120      "DLLs/Services/Properties/AssemblyInfo.cs" line 1
121 %a   "Hosts/Spider9Host - V1.0/Spider9Host/Spider9Host.csproj" line 95
```

`InitializeHost()` sets initial values for CurrentSystemPower, loads the Schedule, sets the PriceLimitDischarge, and some other important initialization procedures. It is called in:

- Methods.cs,
- ScheduleLogic.cs,
- TemperatureLogic.cs,
- PowerLogic.cs,
- RackLogic.cs

Sprint Presentation Jan 18, 2017

- code investigation exercise requires many hours of lexical analysis.
- still discovering new aspects of the system
- primary tools, vim, grep, find.
- Even it is written in Visual Studio, the IDE is unnecessary for this archeological exercise, and can inhibit productivity. (IDE's and projects files are mostly irrelevant to static code analysis, yet have dependencies which if unsatisfied will break project settings)
- got started with the Portia Notes document from Brian
- While we only want to know the Energy Management Logic, and not Battery Management Logic, keep in mind the Spider9 system was conceived by the need for a system capable of managing rechargeable battery clusters connected to photo voltaic panels. Battery Management was the original objective of the Spider 9 system.
- There is some EMS logic, (eg. The Scheduler), but much of the logic that I've seen is for managing the charging and discharging of the batteries.
- The two sets of logic are not clearly distinct.
- Understanding the logic, much of the time also requires understanding the data structures.
- I'll attempt to focus on logic, but details on data structure and handling (defined in code) were and will sometimes be the only way I'm able to understand and explain at this rudimentary stage of my knowledge of the system.
- Imagine a slug mapping the ecosystem of a forest.

Interesting Facts:

- 1591 .cs (C#) source files in Spider9 code base
  - Includes 173 source files for tests
  - My investigation has taken me through approximately 150
- 160 projects (excluding Tests and Unused)

abstract ScheduleLogic