



ptc university

IoT Modeling with ThingWorx Exercise Workbook



M2:4. Create a ThingWorx Project

1. Log on to ThingWorx Composer with your username and password.
2. Click **+NEW** at the top and select **Project** to create a new project.
3. Type MarsInterstellarShipyardProject in the Name field.

***Note:** Because you are using ThingWorx Academic Edition, your username will be automatically appended to any entity you create. For example, if your username is alevine, the entity name will be MarsInterstellarShipyardProject_alevine.*

4. Click **Save**.
5. In the Set Project Context field in the upper-left corner of ThingWorx Composer, type/select **MarsInterstellarShipyardProject**.

***Note:** Once the project has been set in the Set Project Context field, each of the new entities you create will automatically be assigned to this project context.*








M3:8. Create the Shipyard Organization

1. Create a new **Organization** entity.
2. In the Name field, type **MarsInterstellarShipyardOrg**.
3. Click **Save**.

Note: Now we will give all members of the organization visibility to see it exists.

4. In the Entity Tabs section on the top, click **Permissions**.
5. In the Search Organizations field type/select **MarsInterstellarShipyardOrg**.
6. Click **Save**.

M3.9: Create Organizational Units

1. In the MarsInterstellarShipyardOrg entity, navigate to the **Organization** tab.
2. Click Unit 1.
3. Change the name to **MarsInterstellarShipyardOrg**.
4. Click  **Done**.
5. Hover over the MarsInterstellarShipyardOrg button and click the  button underneath it.
6. In the Name field, type **Power**.
7. Click  **Done**.
8. Hover over the MarsInterstellarShipyardOrg button and click the  button between MarsInterstellarShipyardOrg and Power.
9. In the Name field, type **Shipyard**.
10. Click  **Done**.
11. Click **Save**.

M3:10. Creating User Entities

1. Create a new **User** entity.
2. In the Name field, type **klee**.
3. In the New Password field, type **ptcuniversity**.
4. In the Confirm Password field, type **ptcuniversity**.
5. Click **Save**.
6. From the entities tabs, select **User Extensions**.
7. In the Last Name field, type **Lee**.
8. In the First Name field, type **Karen**.



9. In the E-mail Address field, type **klee@ptcuniversity.com**.
10. Click **Save**.
11. Create a user entity with the following details:

User Name	Project	Password	First Name	Last Name	E-mail Address
hconrad	MarsInterstellarShipyardProject	ptcuniversity	Hector	Conrad	hconrad@ptcuniversity.com

M3:12. Creating User Groups

1. Create new user group entities with the following details:

Name	Project
PowerAdminGrp	MarsInterstellarShipyardProject
ShipyardAdminGrp	MarsInterstellarShipyardProject

M3:13. Adding Users to User Groups

1. Click **Manage Members** for the ShipyardAdminGrp user group.
2. Drag **hconrad** from the left panel to the right panel.
3. Click **Save**.
4. Open PowerAdminGrp from the Recent tab.
5. Click **Manage Members** for the PowerAdminGrp user group.
6. Drag **klee** from the left panel to the right panel.
7. Click **Save**.

M3:14. Creating Role Based Groups

1. Create user groups with the following details:

Name	Project	Members
Power.Admin	MarsInterstellarShipyardProject	PowerAdminGrp
Power.Write	MarsInterstellarShipyardProject	Power.Admin
Power.General	MarsInterstellarShipyardProject	Power.Write



M3:15. Adding Role-Based Groups to Organizations

1. Navigate to the Organization page of the MarsInterstellarShipyardOrg from the Recent tab.
2. Click the **Power** box in the Organization chart.
3. In the Members field, type/select **Power.General**.
4. Click ☒ **Done**.

Note: Adding role-based groups as organization members is a best practice, and we are violating that best practice by adding the ShipyardAdminGrp. We're only doing this because you have already created five groups, and there is no instructional value in creating more. In short, we're saving time.

5. Click the **Shipyard** box in the Organization chart.
6. In the Members field, type/select **ShipyardAdminGrp**.
7. Click ☒ **Done**.
8. Click **Save**.



M5:7. Create Thing: Shipyard

Note: When you type an entity, service, or property name, enter it exactly as shown in this guide. Do not change the name or capitalization. These names are used in code that you will copy and paste into your application later in the course, and if your names do not match the copied code, you will have to troubleshoot the code.

1. Create a thing entity with the following details:

Name	Project	Base Thing Template
Shipyard	MarsInterstellarShipyardProject	Timer_Simple

2. Click **Save**.

Note: Next, we add permissions. We will grant the shipyard organizational unit visibility to the Shipyard entity.

3. Navigate to the **Permissions** tab.
4. Verify the **Visibility** tab is selected.
5. Type **M** in the Search Organizations field and click > to expand, but do not select MarsInterstellarShipyardOrg.
6. Click > to expand MarsInterstellarShipyardOrg.
7. Select **Shipyard**.
8. Verify the organization in the visibility table is MarsInterstellarShipyardOrg:Shipyard.
9. Click **Save**.
10. Navigate to the Configuration page of the Shipyard thing.

Note: Notice that the enabled box is checked by default. This means that the timer is active and sending events every updateRate milliseconds (in this case, every 60 seconds). To conserve system resources, ThingWorx Academic Edition will disable your timer if you have been offline for a long period of time. If you log back in and need your timer running, you will have to come back here to re-enable it. Regular, production instances of ThingWorx do not automatically shut down your timers. This is a feature of academic versions designed to support PTC courses.

M5:9. Create Thing Shape: Power Producer



1. Create a Thing Shape entity with the following details:

Name	Project
PowerProducerTS	MarsInterstellarShipyardProject

Note: We will add visibility to the Thing Shape. We'll grant visibility to the power and shipyard organizational units.

2. Click **Save**.
3. Navigate to the Permissions tab of the **PowerProducerTS** thing shape.




4. Verify the Visibility tab is selected.
5. Type **M** in the Search Organizations field and click  to expand, but do not select MarsInterstellarShipyardOrg.
6. Click  to expand MarsInterstellarShipyardOrg.
7. Select **Power**.
8. Likewise, add visibility to the MarsInterstellarShipyardOrg:Shipyard suborganization.
9. Verify the organizations listed in the visibility table are MarsInterstellarShipyardOrg:Power and MarsInterstellarShipyard:Shipyard.
10. Click **Save**.

M5:10. Create Template: Solar Collector


1. Create a Thing Template entity with the following details:

Name	Project	Base Thing Template	Implemented Shapes
SolarCollectorTT	MarsInterstellarShipyardProject	RemoteThingWithFileTransfer	PowerProducerTS

Note: We'll grant visibility and visibility instance to the power and shipyard organizational units. By adding visibility instance, we grant permissions to any solar collectors derived from this thing template.

2. Click **Save**.
3. Navigate to the Permissions tab of the SolarCollectorTT thing template.
4. Verify the Visibility tab is selected.
5. Click the **Instance of Entity**  button to set visibility instance permissions.
6. Grant instance visibility permissions to:

Org or Org Unit
MarsInterstellarShipyardOrg:Power
MarsInterstellarShipyardOrg:Shipyard

7. Click the **Entity**  button
8. Verify the organizations listed in the visibility table are the following:

Org or Org Unit
MarsInterstellarShipyardOrg:Power
MarsInterstellarShipyardOrg:Shipyard

9. Click **Save**.




M5:12. Create Thing: Solar Collector

1. Create a thing entity with the following details:

Name	Project	Base Thing Template
SolarCollector-1	MarsInterstellarShipyardProject	SolarCollectorTT



Note: Next, we will test our visibility using access control reports. Let's check access to the Shipyard.

2. Click **Save**.
3. Click the **Permissions**  tab.
4. Click the **Access Reports** link.
5. Type **k** in the Search Users or Groups field.
6. Select **klee**.
7. Click the **+** in the Search Entities field.
8. Select SolarCollector-1.
9. Click **Apply**.

Note: Notice that Karen has visibility but no run time or design time permissions. This is expected – we gave the Power organizational unit visibility to any instance of the solar collector thing template. Next, let's try Hector.

10. Remove klee from the User or Group field.
11. Click **+** in the Search Users or Groups field.
12. Select **hconrad**.
13. Click **Apply**.

Note: The report correctly states that Hector has visibility. Let's view the details.

14. Click the symbol  →  to view the visibility details report for Hector.
15. Hover over the MarsInterstellarShipyard in the visibility report to view the full text.

Note: The report says that Hector has visibility because he is a member of the MarsInterstellarShipyard's Shipyard organizational unit, and that permission was granted on the SolarCollectorTT thing template, not directly on SolarCollector-1.

M5:13. Duplicate Thing: Solar Collector

1. Select the **Browse** tab.
2. Click **Things** under the MODELING section.
3. Select the check box in the **SolarCollector-1** row from the entities list area.
4. Click Duplicate.
5. Type **SolarCollector-2** in the Name field.



6. Click **Save**.

M5:16. Create Thing Shape: Power Consumer

***Note:** We will create the Power Consumer thing shape by duplicating the Power Producer thing shape. The two shapes are nearly identical, and that way, we keep the access control settings.*

1. Select the **Browse** tab.
2. Click **Thing Shapes** under the MODELING section.
3. Select the check box in the **PowerProducerTS** row.
4. Click Duplicate.
5. Type **PowerConsumerTS** in the Name field.
6. Click **Save**.
7. Click **Permissions** to verify the power and shipyard units have visibility.

M5:17. Iterate Things: Mars Rover

1. Click **SEARCH** on the header of the New Composer. Type/select **BaseRoverTemplate** to open the BaseRoverTemplate thing template.
2. In the Implemented Shapes field, type/select **PowerConsumerTS**.
3. In the Implemented Shapes field, type/select **PowerProducerTS**.
4. Click **Save**.
5. Grant visibility permissions for the BaseRoverTemplate to:

Visibility Permissions	Org or Org Unit
Entity	MarsInterstellarShipyardOrg
Instance	MarsInterstellarShipyardOrg

6. Click **Save**.

***Note:** The rover entities existed before our project, so we will add them to the project.*

7. Open the MarsInterstellarShipyardProject.
8. Select the **Entities** tab.
9. From the Project Header, click **Edit**.
10. From the Available Entities table, for the following seven entities, click the **black arrow** on the right column to bring them into the Project Entities table:
 - BaseRoverTemplate
 - CrewRover1
 - CrewRover2



- CrewRover3
- CrewRoverTemplate
- WeatherRover1
- WeatherRoverTemplate

11. Click **Save**.

Note: Notice that in the browse panel, the base rover template has moved underneath the MarsInterstellarShipyardProject.

M5:20. Create Thing: Power Storage

1. Create a Thing with the following details:

Name	Project	Base Thing Template
PowerStorage	MarsInterstellarShipyardProject	RemoteThing

2. Click **Save**.
3. Grant visibility to the entire **MarsInterstellarShipyardOrg** organization.
4. Click **Save**.




M6:15. Creating Properties: Power System

1. Edit the **PowerStorage** thing.
2. Add a property with the following details:

Name	Base Type	Units	Min Value	Max Value	Persistent	Logged
PowerStored	NUMBER	kWh	0	500000000 (eight zeroes)	True/Checked	True/Checked

3. Expand Advanced Settings and set the following details:


Data Change Type	Data Change Threshold
Value	100

4. Click **Done** .
5. Click **Save**.





Note: We have a logged property, so the thing needs to be associated with a value stream. We'll use the VS value stream which has been created for you.

6. Select the General Information tab.
7. In the Value Stream field, type/select VS.
8. Click Save.

Note: Our thing now has a property. If anyone is to use that property, they need run time access to it. We'll grant the Power.General and ShipyardAdminGrp permission to read all properties for the power storage thing.

9. Navigate to the Run Time permissions page for the PowerStorage thing.
10. Under All Properties, Services, and Events in the Search Users or Groups field, click +.
11. Type/select **Power.General**.
12. Click the + again.
13. Type/select **ShipyardAdminGrp**.
14. Set the Property Read permission for both groups to **Allow** .
15. Click **Save**.

Note: We have a logged property. Let's test that and make sure logging is working as expected. We'll set the value of PowerStored a few times to create log entries.

16. Navigate to the Properties and Alerts page of the PowerStorage thing.
17. In the My Properties table, for the PowerStored property in the Value column, click  to edit the value.
18. Type **11000** in the Set value of property field.
19. Click .
20. Click  for the PowerStored property again.
21. Type **10000** in the Set value of property field.
22. Click .



23. Click  for the PowerStored property a third time.

24. Type **10003** in the Set value of property field.


25. Click .

Note: We'll execute the QueryPropertyHistory service to view the log.

26. Select the thing's **Services** tab.

27. Click **Generic** at the bottom to expand the Generic Services section.

28. Use the scroll bar to scroll down to the QueryPropertyHistory service.

29. Click the **Execute service**  button for the QueryPropertyHistory service.

30. Click **Execute** in the pop-up window.

31. Click **Done**.

Note: We are done with the power storage property. Now, let's create the power produced property on the power producer thing shape.

32. Edit the **PowerProducerTS** thing shape.

33. Add a property with the following details:

Name	Base Type	Units	Min Value	Persistent	Logged
PowerProduced	NUMBER	kW	0	True/Checked	True/Checked

34. Click **Done** .

35. Click **Save**.


Note: We'll grant run time permissions to view properties to the Power.General group. The shipyard only needs to see properties as an aggregate, not individually, so we won't grant permission to the ShipyardAdminGrp.

36. Navigate to the **Run Time** permissions page for the PowerProducerTS thing shape.

37. Click the **Instance of Entity**  button to set Run Time Instance permissions.

38. Under All Properties, Events, and Services, in the Search Users or Groups field, click **+**.

39. Type/select **Power.General**.

40. Set the Property Read permission to **Allow** .

41. Click **Save**.

Note: You can't set the value stream on a thing shape – only from a thing template or thing. So, we'll set it on the solar collector thing template.

42. Edit the **SolarCollectorTT** thing template.

43. In the Value Stream field, type/select **VS**.

44. Click **Save**.

45. Edit the **PowerConsumerTS** thing shape.

46. Add a property with the following details:




Name	Base Type	Units	Min Value	Persistent	Logged
PowerConsumed	NUMBER	kW	0	True/Checked	True/Checked

47. Click **Done** .

48. Click **Save**.


Note: The Power.General group should be able to view the property of every power producer, so we'll set both run time permissions and run time instance permissions.

49. Navigate to the Run Time permissions page.

50. Click the **Instance of Entity**  button to set Run Time Instance permissions.

51. Under All Properties, Events, and Services, in the Search Users or Groups field, click +.

52. Select Power.General.

53. Set the Property Read permission to **Allow** .

54. Click **Save**.

55. Edit the **BaseRoverTemplate** thing template.


56. In the Value Stream field, type/select **VS**.

57. Click **Save**.

Note: Let's test logging for crew rover one.

58. Open the **CrewRover1** thing.

59. Navigate to the Properties and Alerts.

60. For the PowerConsumed property, in the Value column, click  to edit the value.

Note: You may need to refresh your browser window to see the updated properties.

61. Type **5** in the Set value of property field.

62. Click .

63. Click  for the PowerConsumed property again.


64. Type **2** in the Set value of property field.

65. Click .

66. Select the **Services** tab.

67. Click **Generic** to expand the Generic Services section.

68. Scroll down to the QueryPropertyHistory service.

69. Click the **Execute service**  button for the QueryPropertyHistory service.

70. Click **Execute** in the pop-up window.

71. After observing the data, click **Done**.



M6:16. Wrapped Services: Power System

Note: Before we write the get power producers and consumers services, we need to decide which thing to put them in. This service doesn't belong to any one specific thing, so we will create a utility thing to store it.

1. Create a Thing with the following details:

Name	Project	Base Thing Template
PowerSystemServices	MarsInterstellarShipyardProject	GenericThing


Note: The power and shipyard organizational units should be able to see this entity, but other units should not.

2. Grant visibility to the Power and Shipyard organizational units.


Note: Now we can add services. Let's start with the get power producers service.

3. Navigate to the Services tab.
4. Click **Add** to add a service.
5. In the Service Info tab of the service editor, in the Name field, type **GetPowerProducers**.
6. Select the **Output** tab of the service editor.



Note: This service returns structured data – more than one property of more than one power producer. Since the output of a service can only be one base type, it needs to be one that can contain structured data. As a best practice, use the info table for this unless there is a compelling reason to use another base type.

7. From the drop-down list where NOTHING is selected, select **INFOTABLE**.
8. Select the **Me/Entities** tab of the service editor.
9. Select the **Other Entity** radio button.
10. In the Search Entities field, type/select **PowerProducerTS**.
11. In the Choose category drop-down list, select **Queries**.
12. Click **Services** to expand it.
13. Click the **black arrow** ➔ to the right of the **GetImplementingThingsWithData** service name to add the snippet for the service.
14. Click **Check syntax**  at the top of the Script editing area.

Note: Next, we test the service and create a data shape for our result at the same time.

15. Click **Done**.
16. Click **Save**.
17. Click the **Execute service**  button for the GetPowerProducers service.
18. Click **Execute**.
19. Click **+ Data Shape** on the upper-right side of the modal window to automatically create a data shape from this result.
20. In the Name field, type **GetPowerProducersDS**.



21. Click **Save**.
22. Select the **Field Definitions** tab to view the fields in this data shape.
Note: Now that we have our data shape, we apply it to the info table.
23. Navigate to the Services page of the PowerSystemServices thing.
24. Click the **GetPowerProducers** service.
25. Click **Output**.
26. In the **Data Shape** field, type/select **GetPowerProducersDS**.
27. Click **Done**.
28. Click **Save**.
Note: It's nearly identical to GetPowerProducers, so we'll use duplicate.
29. Select the **GetPowerProducers** check box
30. Click Duplicate.
31. In the New Name field, type **GetPowerConsumers**.
32. Click **Done**.
33. Click the **GetPowerConsumers** service.
Note: We need to edit the script to use the power consumers thing shape, and we can't use the same data shape, since the power consumers thing shape won't return the same fields.
34. In the script area, replace PowerProducerTS with PowerConsumerTS.
35. Click **Output**.
36. In the Data Shape field, click the **X** to remove GetPowerProducersDS.
37. Click **Check syntax**  at the top of the Script editing area.
Note: And we create the new data shape.
38. Click **Done**.
39. Click **Save**.
40. For the GetPowerConsumers service, click the **Execute service**  button.
41. Click **Execute**.
42. Click **+ Data Shape** on the right side of the modal window to automatically create a data shape from this result.
43. Type **GetPowerConsumersDS** in the Name field.
44. Click **Save**.
45. Edit the PowerSystemServices thing.
46. Select the **Services** tab.
47. Click the **GetPowerConsumers** service.
48. Click **Output**.
49. In the **Data Shape** field, type/select **GetPowerConsumersDS**.
50. Click **Done**.



51. Click **Save**.

***Note:** As a best practice, you should set permissions for services individually and not grant blanket service execute permissions for the entire thing. Doing so exposes dangerous services to the user that should only be part of the internal API, executed by the code that we write in a way that uses them safely and responsibly. We will give the power.general group permission to execute both services.*


52. Navigate to Run Time permissions.

53. In the Search Properties, Services, or Events field, search for GetPowerProducers.

54. Expand Services.

55. Select **GetPowerProducers**.

56. Under the GetpowerProducers service, in the Search Users or Groups field, type/select **Power.General**.

57. Set the Service Execute permission to **Allow** .

58. In the Search Properties, Services, or Events field, select **GetPowerConsumers**.

59. Under the GetPowerConsumers service, in the Search Users or Groups field, type/select **Power.General**.

60. Set the Service Execute permission to **Allow** .

61. Click **Save**.

62. Click the **Permissions**  tab.

63. Click the **Access Reports** link.

64. In the Search Users or Groups field, type/select **hconrad**.

65. In the Search Entities field, type/select **PowerSystemServices**.

66. Click **Apply**.

***Note:** The report correctly states that Hector was not given permission to execute any services from the PowerSystemServices thing.*

67. Remove hconrad from the User or Group field.

68. In the Search Users or Groups field, type/select **klee**.

69. Click **Apply**.

***Note:** Karen does have permission to execute GetPowerProducers and GetPowerConsumers. Next, let's see if she has permissions to execute the parent service from the internal API – GetImplementingThingsWithData.*

70. Remove PowerSystemServices from the Entity field.

71. In the Search Entities field, type/select **PowerProducerTS**.

72. Click **Apply**.

M6:17. Aggregating Properties

1. Edit the **Shipyard** thing.

2. Add properties with the following details:



Name	Base Type	Units	Persistent	Logged
PowerConsumed	NUMBER	kW	True/Checked	True/Checked
PowerProduced	NUMBER	kW	True/Checked	True/Checked
PowerStored	NUMBER	kWh	False/Unchecked	True/Checked

- Click **Advanced Settings** for the PowerStored property.
- Verify that Value is selected in the Data Change Type field.
- Type **100** in the Data Change Threshold field.
- Click ☒.
- Click **Save**.

***Note:** We are going to do something different for permissions on properties this time. Hector needs to be able to read all properties of the Shipyard thing. Karen needs to read the power properties, but there will be other properties in the future coming from other departments, such as an operational efficiency property from the manufacturing department. Karen shouldn't have access to any new properties, so we will grant her permissions to individual properties, rather than blanket permissions to all properties. However, to see any properties, Karen first needs visibility to the Shipyard thing, which she doesn't currently have.*

- Navigate to the **Visibility** permissions page for the Shipyard thing.
- Grant the MarsInterstellarShipyardOrg:Power organizational unit visibility.
- Select the **Run Time** tab.

***Note:** Let's grant blanket property read to ShipyardAdminGrp.*

- In the Search Users or Groups field type/select **ShipyardAdminGrp**.
- Set the Property Read permission to **Allow** ☒.

***Note:** Now we'll grant permissions to the Power.General group individually.*

- In the Search Properties, Services or Events field, search for **PowerConsumed**.
- Click **Properties** to expand it.
- Select **PowerConsumed**.
- Under the PowerConsumed service, in the Search Users or Groups field type/select **Power.General**.
- Set the **Property Read** permission to **Allow** ☒.
- In the Search Properties, Services or Events field, type/select **PowerProduced**.
- Under the PowerProduced service, in the Search Users or Groups field, type/select **Power.General**.
- Set the **Property Read** permission to **Allow** ☒.
- In the Search Properties, Services or Events field, type/select **PowerStored**.
- Under the PowerStored service, in the Search Users or Groups field, type/select **Power.General**.
- Set the **Property Read** permission to **Allow** ☒. Permissions should look like this:



Permissions | Entities ? Save Done Cancel

Visibility Run Time Design Time

Shipyards_alevine

Run Time ?

▼ All Properties, Services, and Events

Remove Bulk Set Search Users or Groups +

User or Group	Property Read	Property Write	Service Execute	Event Execute	Event Subscribe
ShipyardsAdminDro_alevine	<input checked="" type="checkbox"/> Remove Set	<input checked="" type="checkbox"/> Remove Set	<input checked="" type="checkbox"/> Remove Set	<input checked="" type="checkbox"/> Remove Set	<input checked="" type="checkbox"/> Remove Set

▼ Property, Service, or Event Overrides Search Properties, Services or Events +

▼ Properties

▼ PowerConsumed X

Remove Bulk Set Search Users or Groups +

User or Group	Property Read	Property Write
PowerGeneral_alevine	<input checked="" type="checkbox"/> Remove Set	<input checked="" type="checkbox"/> Remove Set

▼ PowerProduced X

Remove Bulk Set Search Users or Groups +

User or Group	Property Read	Property Write
PowerGeneral_alevine	<input checked="" type="checkbox"/> Remove Set	<input checked="" type="checkbox"/> Remove Set

▼ PowerStored X

Remove Bulk Set Search Users or Groups +

User or Group	Property Read	Property Write
PowerGeneral_alevine	<input checked="" type="checkbox"/> Remove Set	<input checked="" type="checkbox"/> Remove Set

24. Click **Save**.


Note: The shipyard thing now has logged properties, so it needs a value stream specified.

25. Navigate to the General Information page of the Shipyard thing.

26. In the Value Stream field, type/select **VS**.

27. Click **Save**.

M6:18. Binding Properties

1. Navigate to the **Properties and Alerts** page of the Shipyard thing.
2. Click **Manage Bindings**.
3. In the Search Things field, type/select **PowerStorage** from the list.
4. Cursor over  symbol of the PowerStored property and drag PowerStored property for the PowerStorage thing in the left panel to the PowerStored property for the Shipyard thing in the right panel.
5. Click **Done**.
6. Click **Save**.


M6:19. Aggregator Service

Note: Our rollup service will return three values – power produced, consumed, and stored. That means we’re going to need to return an info table, and that info table will need a data shape. So, we’ll create the data shape first. Unlike the last time, we need to create this data shape manually, since there is no other service that returns exactly the fields we need.



1. Create a data shape entity with the following details:

Name	Project
RollupDS	MarsInterstellarShipyardProject

2. Select the Field Definitions tab.
3. Click Add to add the field definition.
4. In the Name field, type **powerProduced**.
5. Type/select NUMBER for the Base Type field.
6. Type kW in the Units field.
7. Click .
8. Create two more field definitions with the following details:

Name	Base Type	Units
powerConsumed	NUMBER	kW
powerStored	NUMBER	kWh

9. Click **Save**.

Note: Now that we have our data shape, we can author the rollup service.

10. Navigate to the Services page of the Shipyard thing.
11. Click **Add**.
12. In the Name field, type **Rollup**.
13. Click **Output**.
14. From the drop-down list where NOTHING is selected, select **INFOTABLE**.
15. In the Search Data Shape field, type/select **RollupDS**.
16. Copy and paste the following code in the Script editing area.

```
/*
*****
Get the current user so we can append _username to entity names
*****
*/
logger.warn("Beginning Shipyard Rollup Service.");
var username = Resources["CurrentSessionInfo"].GetCurrentUser();

/*
*****
Using our GetPowerConsumers() service, get all power
consumers and store them in an infoTable
*****
*/
var sysServThingName = "PowerSystemServices" + "_" + username;
var powerConsumers = Things[sysServThingName].GetPowerConsumers();
var totalPowerConsumed = 0;

/*
*****
Iterate through the table one row at a time, adding to
totalPowerConsumed with each row. This code was derived
from Snippets > Infotable for loop
*****
*/
var tableLength = powerConsumers.rows.length;
for (var x = 0; x < tableLength; x++) {
    var row = powerConsumers.rows[x];
```

```
        totalPowerConsumed = totalPowerConsumed + row.PowerConsumed;
    }


    //Set the Shipyard PowerConsumed property
    me.PowerConsumed = totalPowerConsumed;

    /*****
    Using our GetPowerProducers() service, get all power
    consumers and store them in an infoTable
    *****/
    var powerProducers = Things[sysServThingName].GetPowerProducers();

    /*****
    Use the Aggregate function, available from
    Snippets > InfoTableFunctions > Aggregate
    to sum the column of the totalPowerProduced
    InfoTable.
    *****/
    var params = {
        t: powerProducers /* INFOTABLE */,
        columns: "PowerProduced" /* STRING */,
        aggregates: "SUM" /* STRING */,
        groupByColumns: undefined /* STRING */
    };
    var resultTable = Resources["InfoTableFunctions"].Aggregate(params);
    var totalPowerProduced = resultTable.SUM_PowerProduced;

    //Set the Shipyard PowerProduced property
    me.PowerProduced = totalPowerProduced;

    /*****
    Return info table of power produced/consumed/stored
    Used Snippets > Create Infotable from Datashape and Snippets > Create
    infotable
    entry from datashape as a basis for this code.
    *****/
    params = {
        infoTableName : "InfoTable",
        dataShapeName : "RollupDS" + "_" + username
    };
    var result =
    Resources["InfoTableFunctions"].CreateInfoTableFromDataShape(params);
    var newEntry = new Object();
    newEntry.powerConsumed = totalPowerConsumed; // NUMBER
    newEntry.powerProduced = totalPowerProduced; // NUMBER
    var powStorageThingname = "PowerStorage" + "_" + username;
    newEntry.powerStored = Things[powStorageThingname].PowerStored; // NUMBER
    result.AddRow(newEntry);
    logger.warn("Shipyards Rollup: " + me.PowerConsumed + "C-" +
        me.PowerProduced + "P-" + me.PowerStored + "S");
```

17. Click **Check Syntax**  at the top of the Script editing area.


18. Click **Done**.

19. Click **Save**.



Note: Normally, you test a service immediately after writing it. But first, let's look at our power assets so we know what the result of our rollup service should be.

20. Navigate to the Services tab of the PowerSystemServices thing.


21. In the GetPowerProducers row, click the **Execute Service**  button.

22. Click **Execute**.

23. Click **Done**.

Note: Let's set each of the solar collectors to produce 1000 kW, for a total of 2000 kW for the shipyard, and power produced to 10003 kWh.

24. Navigate to the Properties and Alerts tab of **SolarCollector-1**.

25. In the PowerProduced property row, click .

26. Edit the Set value of property field to **1000**.


27. Click .

28. Set or verify the following property values:

Entity	Property	Property Value
SolarCollector-2	PowerProduced	1000 kW
PowerStorage	PowerStored	10003 kWh
CrewRover1	PowerConsumed	2 kW

Note: If you don't see the property that you want to edit, refresh your browser.

29. Navigate to the Services tab of the PowerSystemServices thing.

30. For the GetPowerConsumers service, click the **Execute Service**  button.

31. Click **Execute**.


32. Verify that at least one of your power producers is consuming more than 0 kW of power. If not, set a property so at least one is producing some power.

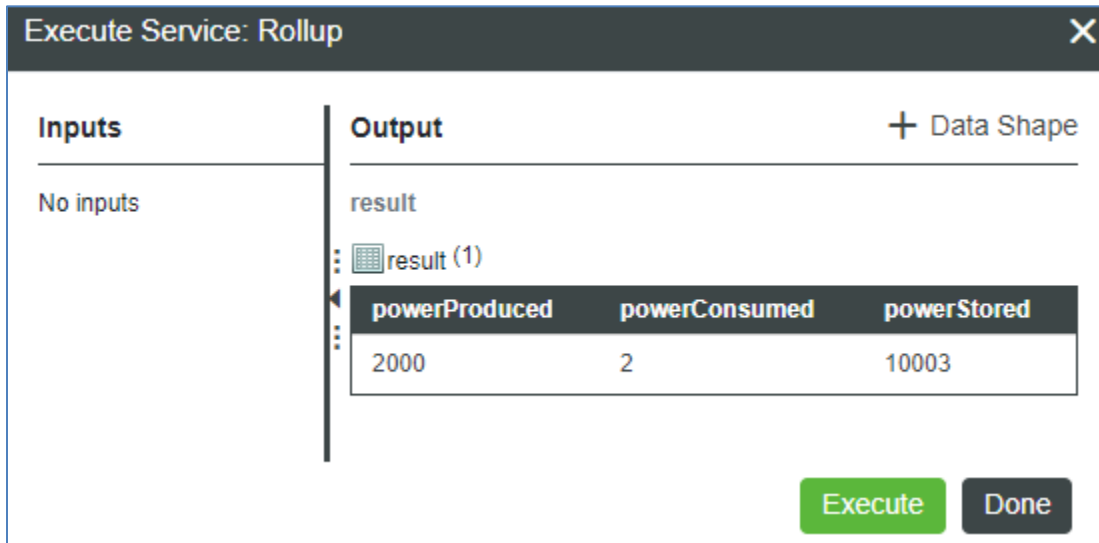
name	tags	PowerConsumed
CrewRover1_testUser	Model Tag not selected	2
CrewRover2_testUser	Model Tag not selected	0
CrewRover3_testUser	Model Tag not selected	0
WeatherRover1_testUser	Model Tag not selected	0

33. Add up and remember the total power produced for the shipyard. In the example above, that would be 2 kW.

34. Click **Done**.

Note: We know our shipyard is producing 2000 kW, consuming 2 kW, and has 10,003 kWh stored. Now let's make sure the Rollup service reflects that.

35. Navigate to the Services page of the Shipyard thing.
36. For the Rollup service, click the **Execute Service**  button.
37. Click **Execute**.



The dialog box titled "Execute Service: Rollup" shows the execution results. The "Inputs" section is empty, labeled "No inputs". The "Output" section shows a table with the following data:

powerProduced	powerConsumed	powerStored
2000	2	10003

At the bottom right, there are two buttons: "Execute" (green) and "Done" (grey).

***Note:** The result info table is correct. However, our service also sets properties on the shipyard. Let's verify that worked as well.*

38. Click **Done**.
39. Select the **Properties** and **Alerts** tab.
40. Verify that the PowerConsumed property is set to **2**, PowerProduced property is set to **2000**, and PowerStored property is set to **10003**. If the property values are not reflecting, then click the **Refresh** button to verify.

M6:20. Subscribing to a Timer

1. Navigate to the Configuration tab of the Shipyard thing.
2. Change the value of the Update Rate field to **10000**.
3. Verify the enabled check box is selected.

***Note:** To conserve system resources, ThingWorx Academic Edition will disable your timer if you have been offline for a long period of time. If you log back in and need your timer running, you will have to come back here to re-enable it.*

4. Select the **Subscriptions** tab.
5. Click **Add** to add your subscription.
6. In the Subscription Info tab of the subscription editor:
 - Verify that Me is selected.
 - Select the **Enabled** check box.



7. Select **Inputs**.
8. In the Event drop-down list, select **Timer**.
9. Type the following code in the Script editing area. This code calls the Rollup service.

```
var result = me.Rollup();
```

10. Click **Done**.
11. Click **Save**.

***Note:** To test this subscription, we will change a few power properties and verify that the shipyard updated appropriately.*

12. Set the following property values:

Thing Name	Property	Property Value
PowerStorage	PowerStored	12000 kWh
SolarCollector-1	PowerProduced	300 kW
CrewRover1	PowerConsumed	150 kW

13. Navigate to the Properties and Alerts tab of the Shipyard thing.
14. Wait for 10 seconds and verify that the PowerConsumed property is set to **150**, PowerProduced property is set to **1300**, and PowerStored property is set to **12000**. You may need to click the Refresh button.

***Note:** The property values change reflecting the values of the underlying power assets.*

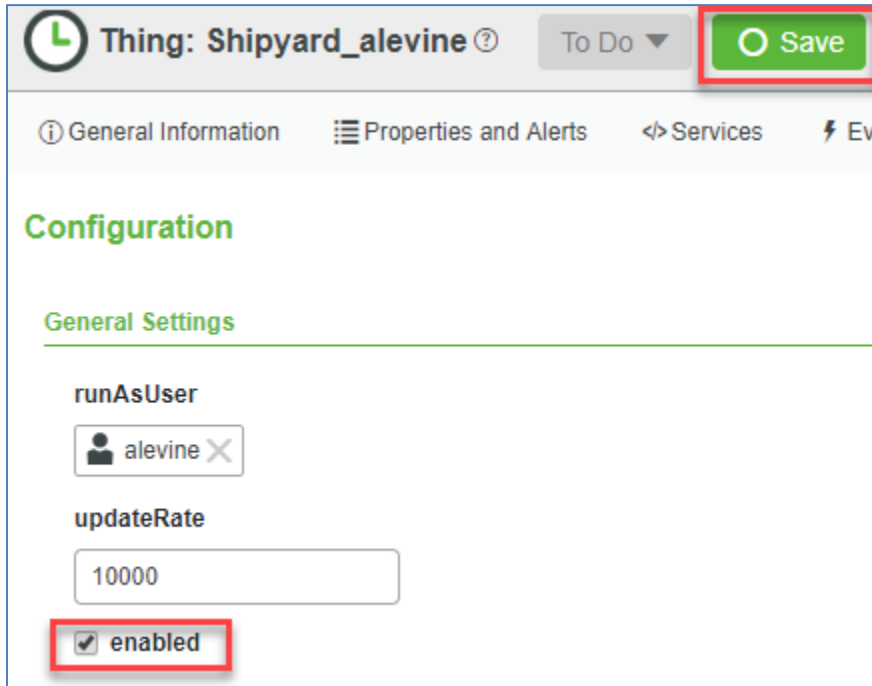
M6:22. Viewing Script Logs

***Note:** If your Rollup service is not executing automatically, check to make sure that the timer in the Shipyard is enabled. We disable timers for less active students to improve system performance.*

This is a feature of the classroom environment and will not occur in a default installation of ThingWorx.

If you suspect your timer has been disabled, do the following:

- *Navigate to the Configuration page of the Shipyard thing.*
- *If the **enabled** check box is not selected, select it.*
- *Click **Save**.*



Thing: Shipyard_alevine ? To Do Save

General Information Properties and Alerts Services Ev

Configuration

General Settings

runAsUser

alevine X

updateRate


10000

☒ enabled

Note: Students do not have access to the standard ThingWorx log viewer due to security concerns. This is an alternate log viewer for ThingWorx Academic Edition. The user interface is different, but it has the same feature set. The most notable difference is that it will only show log entries that originate from your login or a login that you create. Let's look at the script log.

1. Open the **StudentLogViewer** mashup.
2. Click View Mashup.

Note: It is likely that you will need to disable your popup blocker to view your finished mashup.

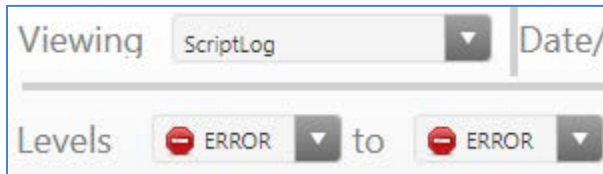
3. In the upper-right corner of Chrome, click the popup blocker icon .
4. Select **Always allow** pop-ups.
5. From the Viewing drop-down list, select **ScriptLog**.

Note: The Rollup service has two log output lines. Since this service is executing every ten seconds, it adds to the log frequently. This is actually very poor design, as logging does consume system resources. However, logs are helpful when troubleshooting issues with the system or your scripts.

6. Click the **User** drop-down list to expand it. This list is used to view the log entries created by users you create, such as Karen or Hector. The rollup service executes as your login user.
7. Expand the Log Level drop-down list.

Note: Setting a log level here will prevent any log entries below that level from being inserted into the script log. Any such entries are never recorded, and permanently lost. So, if our rollup script had debug logs while this is set to warn, those debug log messages would never be recorded. We won't change this. You don't have the access control permissions required to do so.

8. In the Levels drop-down list, select **ERROR**.
9. In the to drop-down list, select **ERROR**.



***Note:** When we filter the log instead of configuring it, we change the log entries that are shown in the table. However, we don't change what is being stored in the actual log file. In this case, warn level logs are still being entered in the log and can be retrieved. We are just filtering them out of the table. Let's create an error.*

10. Click **Update**.
11. In the composer window, navigate to the Services tab of the Shipyard thing.
12. Click the **Rollup** service.
13. Add the following as the last line of code in the service, which will generate an error because the referenced thing doesn't exist:

```
Things[ "NonexistantThing" ].RunFakeService();
```

***Note:** Our rollup service will still work because the error is on the last line. All the useful code has executed before this line is interpreted.*

14. Click **Done**.
15. Click **Save**.
16. Click **Update** in the log viewer window to refresh the script log. The error is being emitted every ten seconds.
17. Select **ALL** in the Levels drop-down list.
18. Select **ALL** in the To drop-down list.

***Note:** We see three log entries every ten seconds, but we don't see entries below WARN level because our system is configured to ignore them. Let's fix our Rollup service.*

19. In the ThingWorx Composer window, navigate to the Services tab of the Shipyard thing.
 20. Click the **Rollup** service.
 21. Remove this last line of code causing the error:
- ```
Things["NonexistantThing"].RunFakeService();
```
22. Click **Done**.
  23. Click **Save**.

## M6:23. Creating Properties

1. Create the following properties:

| Thing Name | Property Name | Base Type | Persistent | Logged |
|------------|---------------|-----------|------------|--------|
|------------|---------------|-----------|------------|--------|



|                 |                      |          |              |              |
|-----------------|----------------------|----------|--------------|--------------|
| Shipyard        | LowPowerState        | BOOLEAN  | True/Checked | True/Checked |
| Shipyard        | NoLowPowerStateUntil | DATETIME | True/Checked | True/Checked |
| PowerConsumerTS | HighPriority         | BOOLEAN  | True/Checked | True/Checked |

**Note:** Let's consider access control on these properties. Our shipyard administrator already has run time read on the shipyard, so he has access. Karen is a more difficult case. She needs to be able to set all these properties, but we can't give her blanket control of all properties in the Shipyard. It will have other properties in the future that are unrelated to Karen's job, such as supply and logistics information.

2. Allow the following permissions:

| Thing Name | Property             | Permission Type         | Group         |
|------------|----------------------|-------------------------|---------------|
| Shipyard   | LowPowerState        | Run Time Property Read  | Power.General |
| Shipyard   | LowPowerState        | Run Time Property Write | Power.Write   |
| Shipyard   | NoLowPowerStateUntil | Run Time Property Read  | Power.General |
| Shipyard   | NoLowPowerStateUntil | Run Time Property Write | Power.Write   |

## M6:24. Writing Services

1. Edit **PowerConsumerTS**.
2. Navigate to the **Services** tab.
3. Click **Add** and select **Local (JavaScript)**.
4. Type **Shutdown** in the Name field.
5. In the Description field, type the following:

This is a generic service intended to be overridden by the thing template or thing that implements the PowerConsumerTS thing shape. It is designed to shut down the power in a safe way to conserve power.

6. Select the **Allow Override** checkbox.
7. Select the **Async** checkbox.


**Note:** This is an asynchronous service; therefore, it requires no input and there is no output. Also, it does not require any code in the script area. We can't test it directly, since it has no code to test.

8. Click **Done**.
9. Click **Save**.

**Note:** Next, we'll write the shutdown service for rovers, placing it on the base rover template. This service will send the rover to the charging station, so it can use the remaining power in its battery to supplement the power grid.

10. Edit the **BaseRoverTemplate** thing template.




11. Navigate to the **Services** tab.
12. For the Shutdown service, click the **Override**  icon.
13. Select the **Me/Entities** tab.
14. Click to expand **Properties**.
15. Click the black arrow ➔ to the right of the Destination property.
16. Alter the end of the code so it reads:

```
me.Destination = "Charging Station";
```

17. Click **Done**.
18. Click **Save**.



*Note: Let's test this service on crew rover one.*

19. Edit the **CrewRover1** thing.
20. Navigate to the **Services** tab.
21. Click **Execute Service**  for the Shutdown service. If necessary, click **Save** to see the Shutdown service.
22. Click **Execute**.

*Note: There is never a result from an asynchronous service, so this no result is normal.*

23. Click **Done**.
24. Select the **Properties** and **Alerts** tab.
25. Notice the Destination property under CrewRoverTemplate is set to Charging Station. If not, click **Refresh** to reflect the Destination property values.

*Note: We'll change the destination because if it is Charging Station, we won't know if a shutdown ran successfully.*

26. For the Destination property, click **Set value of property** .
27. Delete Charging Station and type **Something Else** in the Set value of property field.
28. Click .
29. Edit the **PowerSystemServices** thing.
30. Navigate to the Services tab.
31. Click **Add**.
32. Type **EstHrsToPowerOutage** in the Name field.
33. Click **Output**.
34. From the drop-down list where NOTHING is selected, select **NUMBER**.

35. Copy and paste the following code in the Script editing area:

```
/*

Get the current user so we can append _username to entity names

var username = Resources["CurrentSessionInfo"].GetCurrentUser();
var shipyardEntityName = "Shipyard" + "_" + username;

*/
```

```


* Start by doing rollup to get up-to-the-second data
*****/

Things[shipyardEntityName].Rollup();

/*****
* Calculate hours to dead battery.
* powerDeficit is (PowerConsumed - PowerProduced)
* PowerStored / powerDeficit = hrs to dead battery
*****/

var powerDeficit = Things[shipyardEntityName].PowerConsumed -
Things[shipyardEntityName].PowerProduced;
var hrsToPowerOut;
if (powerDeficit <= 0) { //Set to 10,000 if there is no powerDeficit.
 hrsToPowerOut = 10000;
}
else { // Calculate hours if deficit is positive
 hrsToPowerOut = Things[shipyardEntityName].PowerStored / powerDeficit;
}
result = hrsToPowerOut;

```

36. Click **Check syntax**  at the top of the Script editing area.

37. Click **Done**.

38. Click **Save**.

***Note:** Before we test this service, let's set properties so we know the result. We'll set the system so there is 12 hours of remaining power.*

39. Set or verify the following property values:

| Thing Name       | Property      | Property Value |
|------------------|---------------|----------------|
| PowerStorage     | PowerStored   | 12000 kWh      |
| CrewRover1       | PowerConsumed | 2000 kW        |
| CrewRover2       | PowerConsumed | 0 kW           |
| CrewRover3       | PowerConsumed | 0 kW           |
| SolarCollector-1 | PowerProduced | 1000 kW        |
| SolarCollector-2 | PowerProduced | 0 kW           |

***Note:** 12,000 kWh stored / (1000 kW produced – 2000 kW consumed) = 12 hours*

40. Open **PowerSystemServices**.

41. Navigate to the **Services** tab.

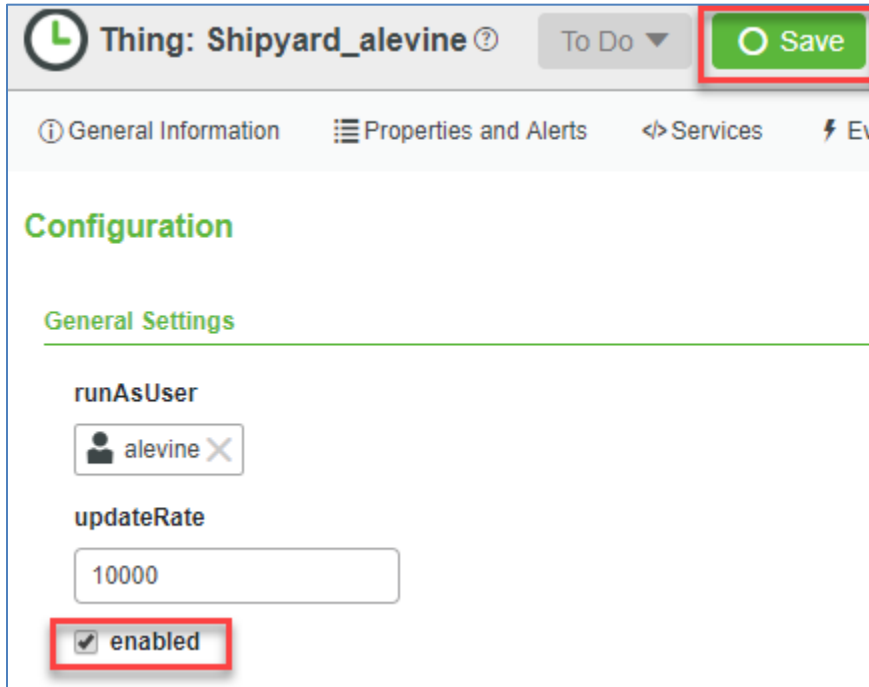
42. For the EstHrsToPowerOutage service, click the **Execute Service**  button.

43. Click **Execute**.

***Note:** We should have 12 hours of energy remaining. If not, your Rollup service may not be executing automatically. Check to make sure that the timer in the Shipyard is enabled. We disable timers for less active students to improve system performance.*

*If you suspect your timer has been disabled, do the following:*

- *Navigate to the Configuration page of the Shipyard thing.*
- *If the **enabled** check box is not selected, select it.*
- *Click **Save**.*



The screenshot shows the configuration interface for a 'Thing' named 'Shipyard\_alevine'. At the top, there is a header bar with a clock icon, the title 'Thing: Shipyard\_alevine', a 'To Do' dropdown, and a green 'Save' button. Below the header, there are tabs for 'General Information', 'Properties and Alerts', 'Services', and 'Events'. The 'Configuration' section is active, showing 'General Settings'. Under 'General Settings', there are two fields: 'runAsUser' with a dropdown menu showing 'alevine' and an 'X' icon, and 'updateRate' with a text input field containing '10000'. At the bottom, there is a checkbox labeled 'enabled' which is checked and highlighted with a red box.

44. Click **Done**.

45. Click **Add**.

46. In the Name field, type **ExitLowPowerState**.

***Note:** This service will take a number as input. This number is the number of hours before a new low power emergency cannot occur. This way, Karen can prevent the automated system from re-starting the low power emergency.*

47. Click **Inputs**.

48. Click **Add**.

49. Type **hrsNoLowPowerState** in the Name field.

50. Select **NUMBER** in the Base Type field.

51. Click **Done**.

52. Click **Output**.

53. Select **STRING** from the drop-down list where NOTHING is selected.

54. Copy and paste the following code in the Script editing area:

```
/*

Get the current user so we can append _username to entity names


*/
var username = Resources["CurrentSessionInfo"].GetCurrentUser();
var shipyardEntityName = "Shipyard" + "_" + username;
```

```

/*****
* If system is not in low power state, don't change state
* and begin creation of result message
*****/
var message;
if (Things[shipyardEntityName].LowPowerState == false) {
 message = "Attempted to exit low power state while " +
 "shipyard is not in a low power state. ";
}
else {

//Otherwise, set low power state to false and
//begin creation of result message
 Things[shipyardEntityName].LowPowerState = false;
 message = "System removed from low power state. ";
}
/*****
* Set hrsNoLowPowerState to current date/time + input
* parameter. Complete message
*****/
var tempDate = Date.now();
tempDate = dateAddHours(tempDate, hrsNoLowPowerState);
Things[shipyardEntityName].NoLowPowerStateUntil = tempDate;
message = message +
 " System will not enter low power state again before " +
 Things[shipyardEntityName].NoLowPowerStateUntil;
/*****
* Send message to result string and script log.
*****/
logger.warn(message);
result = message;

```

55. Click **Check syntax**  at the top of the Script editing area.

***Note:** We can test the service directly from this page.*

56. Below the script window, in the # hrsNoLowPowerState field, type **0**.


57. Click Save and Execute.

***Note:** Notice the date and time listed in the result message. It should match the property in the Shipyard thing. Let's make sure the shipyard is now in a low power state. Now let's set the shipyard to a low power emergency state and try again.*

58. Open the **Shipyard** thing.

59. Select the **Properties** and **Alerts** tab.

60. Verify that the LowPowerState property is false and the NoLowPowerStateUntil property matches the results of the service execution. If not, then click **Refresh** and then verify.


61. For the LowPowerState property, click the **Set value of property**  icon.

62. Select **True** in the Set value of property field.

63. Click .

64. Open the **PowerSystemServices** thing.

65. Navigate to the **Services** tab.

66. For the ExitLowPowerState service, click the **Execute Service**  icon.

67. Type **0** in the hrsNoLowPowerState input parameter.

68. Click **Execute**.

***Note:** Notice the date and time listed in the result message. It should match the property in the Shipyard thing.*

69. Click **Done**.

70. Open the **Shipyard** thing.

71. Select the **Properties and Alerts** tab.

72. Verify that the LowPowerState property is false and the NoLowPowerStateUntil property matches the results of the service execution. If not, then click **Refresh** and then verify.

73. Open the **PowerSystemServices** thing

74. Navigate to the **Services** tab.

75. Click **Add**.

76. Type **EnterLowPowerState** in the Name field.

77. Click **Output**.


78. From the drop-down list where NOTHING is selected, select **STRING**.

79. Copy and paste the following code in the Script editing area:

```
/* *****
Get the current user so we can append _username to entity names
***** */
var username = Resources["CurrentSessionInfo"].GetCurrentUser();
var shipyardEntityName = "Shipyard" + "_" + username;

/* *****
* If already in low power state, do nothing
* except set result message
***** */
var nowDate = Date.now();
if (me.LowPowerState) {
 message = "Attempted to enter low power state while already " +
 "in low power state. No action taken.";
}
else {
 //if date is later than NoLowPowerState, set low power state to true
 //Set result message
 if (nowDate > Things[shipyardEntityName].NoLowPowerStateUntil) {
 Things[shipyardEntityName].LowPowerState=true;
 message = "Entered low power state";
 }
 else {
 //else, don't enter low power state because
 //doing so is blocked by NoLowPowerStateUntil
 //Set result message.
 message = "Cannot enter low power state until " +
 Things[shipyardEntityName].NoLowPowerStateUntil +
```

```
 "Try again later or reset Shipyard.";
 }
}
/*****
* send message to result string and script log
*****/
logger.warn(message);
result = message;
```

80. Click **Check syntax**  at the top of the Script editing area.

81. Click the **Save and Execute** button to test the service.

82. Click **Execute**.

***Note:** We get an entered low power state message. Let's exit the low power state and set the system so it can't re-enter the low power state for one hour.*

83. Click **Done**.

84. For the ExitLowPowerState service, click the **Execute Service**  icon.

85. In the hrsNoLowPowerState input parameter, type **1**.

86. Click **Execute**.

87. Click **Done**.

***Note:** Now let's try to re-enter the low power state and verify that we can't.*

88. For the EnterLowPowerState service, click the **Execute Service**  icon.

89. Click **Execute**.

***Note:** We get a message that we can't enter a low power state. Next, we'll run the ExitLowPowerState service again to reset the NoLowPowerStateUntil property.*

90. Click **Done**.

91. For the ExitLowPowerState service, click the **Execute Service**  icon.

92. in the hrsNoLowPowerState input parameter, type **0**.

93. Click **Execute**.

***Note:** The result should be a message saying that you attempted to exit low power state while shipyard is not in a low power state.*

94. Click **Done**.

95. Click **Save**.







96. Set the following Run Time permissions on the PowerSystemServices thing:

| Service             | Permission Type          | Groups                           |
|---------------------|--------------------------|----------------------------------|
| EstHrsToPowerOutage | Run Time Service Execute | Power.General & ShipyardAdminGrp |
| EnterLowPowerState  | Run Time Service Execute | Power.Admin                      |
| ExitLowPowerState   | Run Time Service Execute | Power.Admin                      |

## M6:25. Events and Subscriptions

***Note:** We need a low power emergency custom event, and every event needs a data shape to define the data that comes with it. We'll start by creating that data shape. It has four fields – The estimated time to power outage, power produced, consumed, and stored. We already have the RollupDS which has three of these fields, so we'll start by duplicating that.*

1. Duplicate **RollupDS**.
2. In the Name field, type **LowPowerEmergencyDS**.
3. Select the **Field Definitions** tab.
4. Click **Add** to add the field definition.
5. In the Name field, type **estHrsToPowerOut**.
6. Select **NUMBER** for the Base Type field.
7. Click .
8. Click **Save**.
9. Open the **Shipyard** thing.
10. Navigate to the **Events** tab.
11. Click **Add** to add an event.
12. In the Name field, type **LowPowerEmergency**.
13. In the Data Shape field, type/select **LowPowerEmergencyDS**.
14. Click .
15. Click **Save**.

***Note:** Our custom event exists, but nothing is subscribed to it. Now we will start implementing the chain of events and subscriptions that cause a low power emergency, starting with the data change event on power stored.*

16. Select the **Subscriptions** tab.
17. Click **Add** to add your subscription.
18. Verify that **Me** is selected.
19. Select the **Enabled** checkbox.
20. Click **Inputs**.

21. In the Event list, select **DataChange**.
22. In the Property list, select **PowerStored**.
23. Type the following code in the Script editing area:

```
/*

Get the current user so we can append _username to entity names


var username = Resources["CurrentSessionInfo"].GetCurrentUser();
var sysServEntityName = "PowerSystemServices" + "_" + username;

//Get hours to power outage and current time
var hrsToPowerOutage = Things[sysServEntityName].EstHrsToPowerOutage();
var nowDate = Date.now();
/*

* We only act if there are less than 12
* hours to power outage and the system
* can enter a lower power state according to the
* NoLowerPowerStateUntil DATETIME parameter
* Otherwise, this subscription does nothing.

if ((hrsToPowerOutage < 12) &&
 (nowDate > me.NoLowPowerStateUntil)) {
 //Emit the LowPowerEmergency event
 var params = {
 powerConsumed: me.PowerConsumed,
 powerStored: me.PowerStored,
 powerProduced: me.PowerProduced,
 estHrsToPowerOut: hrsToPowerOutage
 };
 me.LowPowerEmergency(params);
 //Enter LowPowerState
 Things[sysServEntityName].EnterLowPowerState();
}
```

**Note:** This code calls *estHrsToPowerOutage*. If the answer is less than 12 hours, it emits the *LowPowerEmergency* event and executes the *EnterLowPowerState* service.

24. Click **Check Syntax**  at the top of the Script editing area.
25. Click **Done**.
26. Click **Save**.

**Note:** Let's see if this works. We'll trigger the data change event by changing *PowerStored*. Let's put the shipyard in a state this is not a power emergency. We'll also make Crew Rover 2 a critical power producer.


27. Set or verify the following property values:

**Note:** You can validate power producing and consuming properties faster using the *GetPowerProducers* and *GetPowerConsumers* services than going to each entity individually in Composer. Entries in bold should only need to be validated – you set them to these values in earlier exercises.

| Thing Name       | Property             | Property Value  |
|------------------|----------------------|-----------------|
| PowerStorage     | PowerStored          | 13000 kWh       |
| CrewRover1       | PowerConsumed        | 2000 kW         |
| CrewRover2       | PowerConsumed        | 0 kW            |
| CrewRover2       | HighPriority         | True/Checked    |
| CrewRover3       | PowerConsumed        | 0 kW            |
| SolarCollector-1 | PowerProduced        | 1000kW          |
| SolarCollector-2 | PowerProduced        | 0 kW            |
| Shipyards        | NoLowPowerStateUntil | Yesterday       |
| Shipyards        | LowPowerState        | False/Unchecked |

28. Open the **PowerSystemServices** thing.

29. Navigate to the Services tab.

30. For the EstHrsToPowerOutage service, click the **Execute Service**  icon.

31. Click **Execute**.

**Note:** You will get 13 in the result. That's more than 12 hours, so we aren't in a power emergency. If you do not get 13, your Rollup service may not be executing automatically. Check to make sure that the timer in the Shipyards is enabled. We disable timers for less active students to improve system performance.

If you suspect your timer has been disabled, do the following:

- Navigate to the Configuration page of the Shipyards thing.
- If the **enabled** check box is not selected, select it.
- Click **Save**.

32. Click **Done**.

**Note:** Let's change power stored to 10,000, resulting in 10 hours or power left and a power emergency.

33. In the PowerStorage thing, set the PowerStored property to **10000**.

34. Open the **Shipyards** thing.

35. Navigate to the **Properties and Alerts** tab.

36. Under the My Properties table, verify that the LowPowerState property value is now true (LowPowerState property check box is selected). If not, then click **Refresh** and then verify.

**Note:** The LowPowerState property was set to true by our subscription. But, non-critical power consumers didn't shut down. We'll implement that next with a subscription to the LowPowerEmergency event.

37. Open the **PowerConsumersTS** thing shape.

38. Navigate to the Subscriptions tab.

39. Click **Add**.

40. Select the **Other Entity** radio button.

41. In the Search Entities field, type/select **Shipyards**.




42. Select the **Enabled** check box.

43. Click **Inputs**.

44. From the Event drop-down, select **LowPowerEmergency**.

45. Type the following code to the Script area. This code calls the overridable shutdown service for the power consumer, so a large robot can have a lengthy shutdown procedure, while our rovers can go to a charging station:

```
if (me.HighPriority === false) {
 me.Shutdown();
}
```

46. Click **Check Syntax**  at the top of the Script editing area.

47. Click **Done**.


48. Click **Save**.

*Note: Next, we'll get out of the low power state and try again. This time, our consumers should shut down.*

49. In the PowerStorage thing, set the PowerStored property to **13000**.

50. Open the **PowerSystemServices** thing.

51. Navigate to the **Services** tab.

52. For the ExitLowPowerState service, click the **Execute Service**  button in the Execute column.

53. Type **0** in the hrsNoLowPowerState field.

54. Click **Execute**.

*Note: You should get a message that the System is removed from low power state.*

55. Click **Done**.

56. In the Shipyard thing, verify that the LowPowerState property is false/unchecked.

*Note: The low power state is over. Let's trigger it again.*

57. In the PowerStorage thing, set the PowerStored property to **10000**.

58. In CrewRover1, verify that the Destination property is Charging Station.

*Note: This means that the CrewRover1 is headed to the Charging Station – our automation works!*

59. In CrewRover2, verify that the Destination property is not Charging Station.

*Note: We tagged Crew Rover 2 as a high-priority power consumer that isn't shut down in a power emergency. It will do whatever it was doing before.*