Example Title:	Utilization Application
Description:	This example provides (in Building Block design pattern) both the infrastructure (shapes, data entities and services) and application for a generic utilization solution that can be used as a demonstration and / or for a proof of concept / pilot. Two implementations are included – one to house data in ThingWorx objects (data tables and a stream) and one to house data in SQL database objects
Key Concepts:	Building Blocks, Utilization, Data Tables, Streams, Database, Thing Shapes
Latest TW Version Used:	9.4.1
Extensions Required:	ThingWorx Solution Framework 1.2.102
Original Tech Sales Engineer:	John Ristey
Example Revision:	• 1.0.1 – First Release with Building Block design pattern on ThingWorx 9.4.1

1. **Supplied Files**
   a. These are the Building Block extension files:
      i. utilization-thingworxobjects-1.0.1 -extension-package.zip
      ii. utilization-sqlobjects-1.0.1 -extension-package.zip
   b. ConceptualUtilizationBuildingBlockApplication.pptx
      i. PowerPoint that runs through the provided functionality
   c. UtilizationApplicationUserManual.1.0.1.pdf – this document
   d. TableAndDataInitializationScripts.sql – scripts for creating SQL server data infrastructure
   e. PTCDTS.Base.Permissions-extension-1.2.103.zip – a set of permissions that can be imported for utilization application

2. **Building Block Design Pattern**
   a. The following building blocks were created for the application

Building Block Name	**Building Block Type**	**Description**
PTCSC.Utilization	Abstract	Services building block
PTCSC.UtilizationTWImpl	Implementation	Implementation of PTCSC.Utilization using ThingWorx data objects
PTCSC.UtilizationSQLImpl	Implementation	Implementation of PTCSC.Utilization using SQL Server data objects
PTCSC.UtilizationUI	UI	Utilization UI components
PTCSC.UtilizationAdminUI	UI	Building block to drive the Utilization Admin menu options

   b. PTCSC.Utilization Details
      i. This is the abstract building block
      ii. Data Shapes are defined in this building block
      iii. Services are defined in the PTCSC.Utilization.Management_TS of this building block
      iv. A PTCSC.Utilization.ModelLogic_TS is defined to be applied to model entities in the solution
   c. PTCSC.UtilizationTWImpl Details
      i. This is the implementation building block that uses ThingWorx data objects
      ii. Service code lives in the PTCSC.UtilizationTWImpl.Manager_TT
      iii. ThingWorx Data Objects
         1. PTCSC.UtilizationTWImpl.MachineCode_DT
         2. PTCSC.UtilizationTWImpl.Reason_DT
         3. PTCSC.UtilizationTWImpl.ReasonGroup_DT

        4. PTCSC.UtilizationTWImpl.UtilizationState_DT
        5. PTCSC.UtilizationTWImpl.Utilization_SM
   d. PTCSC.UtilizationSQLImpl Details
      i. This is the implementation building block that uses SQL data objects
      ii. Service code lives in the PTCSC.UtilizationSQLImpl.Manager_TT
      iii. The services call into the PTCSC.UtilizationSQLImpl.SQLDatabase thing
      iv. The PTCSC.UtilizationSQLImpl.Database thing has its services defined on the PTCSC.UtilizationSQLImpl.SQLDatabase_TT which is derived from a SQLThing
      v. Since the PTCSC.UtilizationSQLImpl.Database thing is derived from a SQLThing, we need to tie it to an existing persistence provider. By using an existing persistence provider, we can cut down on the number of database connections.

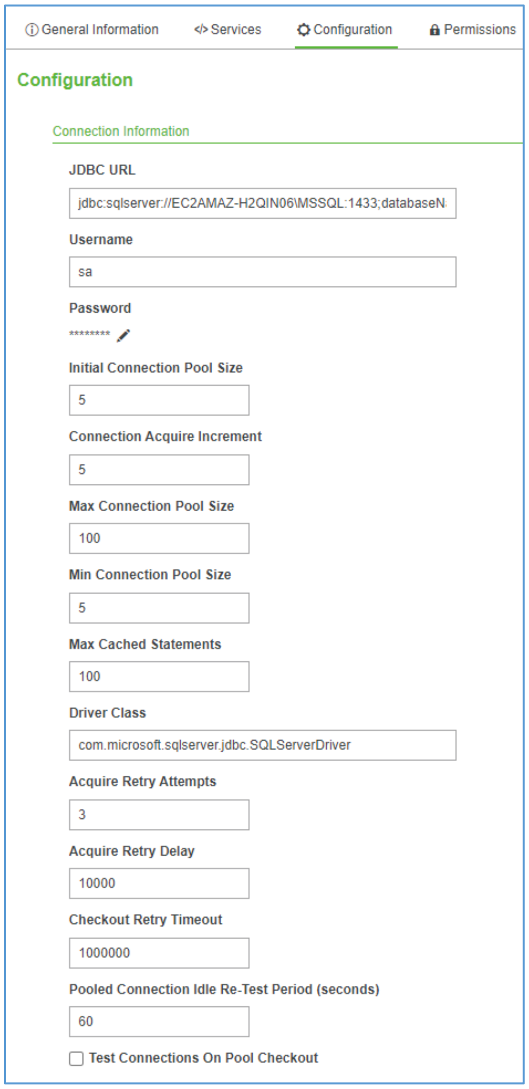
3. **Installation Steps**
   a. Determine which implementation you will utilize
      i. PTCSC.UtilizationTWImpl – configuration data stored in 4 ThingWorx data tables and historical data stored in a ThingWorx stream object (not a value stream, but a stream with schema)
      ii. PTCSC.UtilizationSQLImpl – configuration and historical data stored in SQL Server
   b. Install and initialize Solution Framework 1.2.102
      i. Link for Solution Framework 1.2.102
      ii. Install instructions are included in the above link for the Solution Framework. We simply:
         1. Downloaded the solutionframework-1.2.102-extension-package.zip
         2. Imported the file into our ThingWorx server as an extension
         3. Ran the InitializeSolution service on PTC.Base.Manager with the proper JSON input
         4. This was the JSON for a postgres ThingWorx installation:

```
{

"twxAdminUserName" : "Administrator",

"twxAdminPassword" : "passwordhere",

"twxURL" : "https://serveraddress/Thingworx",

"databaseUser": "twadmin",

"overrideComponentDeploymentState": false,

"databasePassword": "twadmin",

"databaseJDBCString": "jdbc:postgresql://localhost:5432/thingworx",

"databaseThing": "PTC.DBConnection.PostgresDatabase"

}
```

   c. Import the desired implementation extension package file in Composer

Extension Package Name	**Data Structure Type**
utilization-thingworxobjects-1.0.1-extension-package.zip	ThingWorx data objects
utilization-sqlobjects-1.0.1-extension-package.zip	SQL server data objects

   d. If using SQL Server data objects
      i. Go to your SQL Server
      ii. Create a database to house your Utilization objects
      iii. Rn the provided SQLTableAndProcedureScripts.sql against that database to create the required database tables and stored procedures
      iv. In Composer, make sure that you have a persistence provider configured to your SQL database
         1. If you create a new persistence provider, make sure to check the configuration settings. These were the settings on a test Persistence Provider:



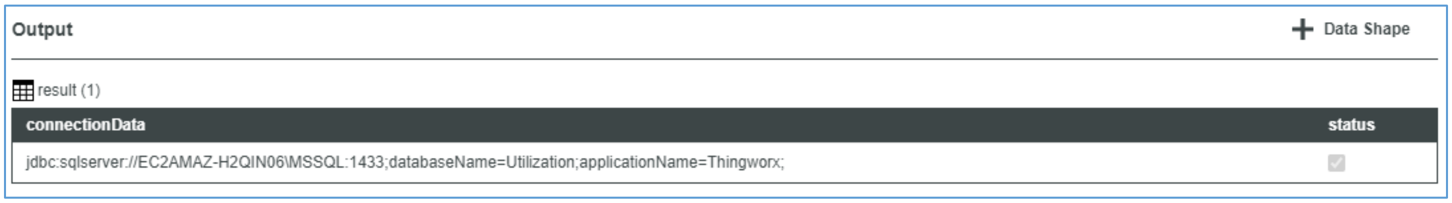
Configuration

Connection Information

JDBC URL	jdbc:sqlserver://EC2AMAZ-H2QIN06\MSSQL:1433;databaseN
Username	sa
Password	********
Initial Connection Pool Size	5
Connection Acquire Increment	5
Max Connection Pool Size	100
Min Connection Pool Size	5
Max Cached Statements	100
Driver Class	com.microsoft.sqlserver.jdbc.SQLServerDriver
Acquire Retry Attempts	3
Acquire Retry Delay	10000
Checkout Retry Timeout	1000000
Pooled Connection Idle Re-Test Period (seconds)	60

☐ Test Connections On Pool Checkout

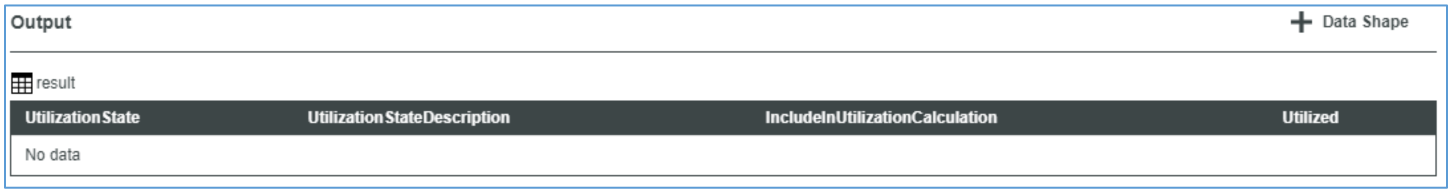
2. After configuration of the persistence provider, if you run the TestConnectivity service you should get true:



result (1)

connectionData	status
jdbc:sqlserver://EC2AMAZ-H2QIN06\MSSQL:1433;databaseName=Utilization;applicationName=Thingworx;	✓

v. Go to the PTCSC.UtilizationSQLImpl.SQLDatabase thing
vi. Go to Configuration, choose the appropriate persistence provider that connects to your SQL database
vii. Once you set and save this configuration, it you run the GetUtilizationStates service on the PTCSC.UtilizationSQLImpl.SQLDatabase thing, you should get zero rows, but it should connect and list the columns:



result

UtilizationState	UtilizationStateDescription	IncludeInUtilizationCalculation	Utilized
No data			

e. Deploy the Utilization Building Blocks
   i. Navigate to the PTC.Base.Manager thing
   ii. Go to services

        iii. Run the InitializeSolution service with an empty JSON object as the deploymentConfig input {}
    f. Navigate to the specific implementation manager that was chosen (PTCSC.UtilizationSQLImpl.Manager or PTCSC.UtilizationTWImpl.Manager)
        i. Go the configuration tab
        ii. Enter or edit configuration values as needed:
            1. DaysToKeepWithPurging – integer (7 was the tested value)
            2. HoursToLookBackOnUtilScreens – integer (6 was the tested value)
            3. ScreenRefreshRate – integer (30 seconds was the tested value)
            4. UtilizationCriticalPercent – integer value at which formatted utilization value goes from red to yellow (50 was the tested value)
            5. UtilizationWarningPercent – integer value at which formatted utilization value goes from yellow to green (63 was the tested value)
        iii. Save the changes if edited
    g. Import the Utilization permissions extension package
        i. Import the provided PTCDTS.Base.Permissions-extension-1.2.103.zip file
            1. This file has the required permissions for the utilization application. It is a package that was created on the solution framework. Note that it needs to have a later build that the permissions extension currently installed on your ThingWorx server. If required, you may have to edit the metadata.xml file.
        ii. After the extension import, navigate to the PTC.Base.Manager thing
        iii. Go to services
        iv. Run the InitializeSolution service with an empty JSON object as the deploymentConfig input {}

4. **Setup for Demo**
    a. We will need a template to create model objects from that we want to collect utilization data from
    b. Simply create a new template (i.e. PTCSC.Utilization.Lathe_TT)
    c. For the created template, give Visibility permissions for the template and instances to:
        i. PTCDTS.BasePermissions.Default_OR
    d. Give it the PTCSC.Utilization.ModelLogic_TS thing shape
    e. Also give it the PTC.Base.ConfigManagement_TS thing shape
    f. Create some instances from the template (i.e. PTCSC.Utilization.Lathe1, PTCSC.Utilization.Lathe2, etc.)
    g. The description will be the name displayed on the mashups
    h. For each thing that is created that will not use a machine code lookup to drive utilization sate, go to the PTCSCSimulationEnabled property and set it to true.
    i. Additional Machine code configuration steps
        i. An integer machine code can be used to drive a lookup to determine Utilization State, Reason Group and Reason
        ii. If you would like to add this to your demo data, complete the following:
            1. On the PTCSC.UtilizationTWImpl.Manager or PTCSC.UtilizationSQLImpl.Manager thing, run the AddDemoMachineCodeDataForEquipment service for each piece of equipment (use the thingname as the EquipmentID parameter) you would like to use MachineCode to drive the state lookup
            2. Set the property PTCSCUseMachineCode on the thing to true (or use the Machine Code Admin mashup to set this property)
    j. Review the provided Power Point file for functionality overview
    k. Additional Machine(s)
        i. If additional machine(s) are desired, simply create another / more instance(s) from the Thing Template that was created above and configure appropriately. Nothing else is required.
    l. Data Simulation
        i. The subscription on the PTCSC.Utilization.ModelLogic_TS thing shape to the PTCSC.Utilization.Simualtion_TM timer event is what runs the utilization data simulation. The timer event is configured to run every minute. If simulation records are not appearing, make

sure that the entity is configured to use simulation and that the subscription is enabled / timer is enabled.

m. Utilization Record Purging
    i. The PTCSC.Utilization.RecordPurge_TM timer thing is set at 86400000 ms (1 day)
    ii. The subscription to the timer event is on the implementation manager thing (PTCSC.UtilizationTWImpl.Manager or PTCSC.UtilizationSQLImpl.Manager) and will delete utilization records older than the configured threshold (7 days is the default).
    iii. If utilization records are not being purged, check to make sure that this subscription and the timer is enabled.

5. **Permissions for Utilization Application**
    a. Based on the permissions package that was imported, we can create the following types of users for the application:
        i. Administrator Users: need to be in the PTCDTS.Base.Permissions.ApplicationAdmin_UG
            1. Testing with utiladmin / utiladmin123456
        ii. Non-Admin Users: need to be in the PTCDTS.Base.Permissions.Default_UG
            1. Testing with utiluser / utiluser123456

6. **Setup for a Proof of Concept / Pilot**
    a. Disable he subscription on the PTCSC.Utilization.ModelLogic_TS thing shape to the PTCSC.Utilization.Simualtion_TM event.
    b. Navigate to the specific implementation manager that was chosen (PTCSC.UtilizationSQLImpl.Manager or PTCSC.UtilizationTWImpl.Manager)
        i. Go the configuration tab
    c. Edit the DaysToKeepWithPurging parameter if more than / less than 7 days of history is required
    d. Create a thing template to create your thing instances from
        i. If you will be connecting to Kepware, this template would use the RemoteThing thing template as a base template
        ii. For the created template, give Visibility permissions for the template and instances to:
            1. PTCDTS.BasePermissions.Default_OR
        iii. Give it the PTCSC.Utilization.ModelLogic_TS thing shape
        iv. Also give it the PTC.Base.ConfigManagement_TS thing shape
    e. Determine the required Utilization States, Reason Groups, Reasons required for the POC
    f. Make any changes to these items in the Utilization Admin, General Admin administration screen
    g. If the Utilization States are altered, several StateDefinitions related to utilization states will need to be adjusted and reimported appropriately on the mashups – not that big of a deal, but hopefully these 5 states are adequate
        i. If these changes are required, you will need to duplicate the affected mashups and State Definitions and change your menu configuration.
    h. Create instances from your thing template for the machines for the POC
    i. If needed, create additional properties on the template or individual things for mapped Kepware tags
    j. Install Kepware, and create a corresponding Industrial Gateway Thing in ThingWorx
    k. Add the IndustrialThingShape on the thing template that was created
    l. Point the instance IndustrialThing property to the Industrial Connector for the Kepware server
    m. Determine if a single integer machine code can drive utilization state, reason group and reason. If so, use these steps:
        i. Map the integer automation tag to the thing property PTCSCMachineCode
        ii. Use the Utilization Admin, Machine Code administration screen to create the lookup table by machine that maps the integer machine code to a state, reason group and reason
        iii. Ensure the that "Use Machine Code to Determine Reason Code" check box is checked and the "Set Properties" button is clicked.

    iv. The subscription should already be in place and active to lookup the required state, reason group and reason when the PTCSCMachineCode value changes. This subscription is on the PTCSC.Utilization.ModelLogic_TS

n. If multiple tags drive the state reason:
    i. Map any “tags of interest” that will factor into business logic for change of state / reason code.
    ii. Create a UtilizationStateChange service (may be different for specific thing instances) that is triggered when applicable tags change that may signal a utilization state change
    iii. The current logic is that the utilization record is recorded when the reason changes.
    iv. In the custom service, ensure that the utilization state and reason group properties are set prior to changing the reason
    v. In the custom service make sure to set the PTCSCEventStart property to the current time
    vi. When the reason is changed, the utilization record will be created by the existing subscription / service
    vii. Here is an example of a super simple UtilizationStateChange service:

```
if (me["PLC_RUNNING"] == true) {
        me.PTCSCReasonGroup = "Running";
        me.PTCSCUtilizationState = "Running";
        me.PTCSCReason = "Good Production";
}
else if (me["PLC_RUNNING"] == false) {
        me. PTCSCReasonGroup = "Downtime";
        me. PTCSCUtilizationState = "Down";
        me.PTCSCReason = "Unknown";
}
var timestamp = new Date();
me.PTCSCEventStart = timestamp;
```