ptc

thingworx®

Starter Apps Asset Simulator Guide

8.x

PTC Inc., 140 Kendrick Street, Needham, MA 02494 USA

Contents

About This Guide

The Asset Simulator is used with Asset Advisor to simulate device behavior without having to connect to a physical asset. The Asset Simulator is intended for use with the 8.x release stream.

This *Starter Apps Asset Simulator Guide* describes the following:

- Installation and configuration of the Asset Simulator.
- Using the Asset Simulator.

Comments

PTC welcomes your suggestions and comments on its documentation. To submit your feedback, you can send an email to documentation@ptc.com. To help us more quickly address your concern, include the name of the PTC product and its release number with your comments. If your comments are about this book, include the *Starter Apps Asset Simulator Guide* book title.

1

Asset Simulator Overview

Asset Simulator enables you to simulate the behavior of sensors on an asset without connecting to a real device. The Asset Simulator system has two main components: the Data Generator and the Asset Simulator. The Data Generator allows you to load or generate sensor data readings. Once sensor data is loaded into the Data Generator, you can simulate several sensor behaviors through the Asset Simulator: normal, anomalous, disconnected, failed, and maintenance states.

Note

Disconnected, failed, and maintenance states are treated identically by Asset Advisor.

In addition to installing the Asset Simulator and example assets, you can also create new simulated assets. For information on creating new simulated assets, see Asset Simulator Configuration on page 11.

Note

The provided example assets can be imported only to ThingWorx instances with ThingWorx Service Apps installed. The Asset Simulator can be used with ThingWorx Manufacturing Apps, but simulated assets must be created.

2

Installing the Asset Simulator

The Asset Simulator can simulate actual device behavior without having to connect to a physical asset. It does this by replaying data sequences derived from mathematical distributions or actual asset data imported as CSV files. Virtual assets can be configured to reference these data sequences and expose them as asset behavior. The Asset Simulator communicates with KEPServerEX in the same way a real device does. The simulated asset behavior is controlled through an administration console. Typically, both normal and anomalous behaviors are defined for each sensor on the asset.

The Asset Simulator provides 10 example simulated assets as part of a reference implementation (5 pump assets and 5 robotic arm assets). Importing the provided `demo-assets.xml` creates the 10 example assets as things in ThingWorx. Once the Asset Simulator and KEPServerEX are configured, the properties on these things reflect simulated data.

The following section describes how to install and configure the Asset Simulator to bring life to the example assets. Then, you can view data streaming through each asset and control their respective sensors' status through the Asset Simulator administration console, to illustrate how Asset Advisor visualizes both normal and anomalous behaviors.

KEPServerEX Prerequisites

If you plan to install the Asset Simulator, when prompted to **Select Features** during your KEPServerEX 6.2 or KEPServerEX 6.3 installation, do the following:

- Expand the **Communication Drivers** tree and **OPC Connectivity Suite** subtree. Select **OPC UA Client**.
- Ensure that **OPC Quick Client** is selected for installation in the **Communications Server** tree.

Distribution

The Asset Simulator has two main components: the Data Generator and the Asset Simulator. The Data Generator produces data sequences for the Asset Simulator to consume. The Asset Simulator distribution file, `ThingWorx-Service-Apps-Asset-Simulator-<version>.zip` is available in the distribution location and contains two sub-distributions and two data files:

- `data-generator-<version>.zip`
- `asset-simulator-<version>.zip`
- `KepwareDemoAssetsProject.json`
- `demo-assets.xml`

Download the distribution file and unzip it into a directory of your choice.

Installing the Data Generator

The Data Generator service produces sequences of data points on request. The Asset Simulator takes these data points and exposes them in a protocol that mimics sensor hardware.

Unzip the `data-generator-<version>.zip` file into a directory of your choice. It contains the following files:

- `demo-dataset.csv`
- `load-demo-dataset.bat`
- `load-demo-dataset.sh`
- `ptc-cslm-data-generator.jar`
- `start.bat`
- `start.sh`

Running the Data Generator

Note

For simplicity, it is recommended that a team run a single instance of the Data Generator on a shared server, to be used by all Asset Simulator instances. This makes setup easier, and also allows users to share data sequences.

The Data Generator persists its data sequences into Redis. The Data Generator exposes these data sequences through its REST API.

1. Install Redis—
   - On Windows—
     a. Download the Microsoft Open Tech Redis 3.2.1 installer (`Redis-x64-3.2.100.msi`) from the following URL: https://github.com/MSOpenTech/redis/releases.
     b. Run the Redis on Windows installer. The default settings are normally fine but can be changed at the user's discretion. Redis is now running as a Windows service.
   - On Linux—
     a. For detailed instructions on installing Redis on Linux, refer to their Quick Start guide at the following URL: https://redis.io/topics/quickstart, or install using your distributions package manager.
     b. Start the Redis server according to your installation instructions.

   Note

   Do not store any other data in this Redis installation.

2. Start the Data Generator—
   a. Edit the `start.sh` file (for Linux) or `start.bat` file (for Windows). Replace `localhost` with the actual host name of the Redis server, if different.

   The Data Generator service can be executed with several command-line flags which can be added to the start script. These include:

   - `--server.port=<port>` to change the API/UI port from 8082. (The URLs in these instructions assume that the Data Generator is installed at `localhost:8082`.)
   - `--logging.file=<file>` to log to a file (in addition to standard console logging).

    - `--debug` for more detailed logging.
    - `--spring.redis.host=<host>` to specify a Redis location (default is localhost).
    - `--spring.redis.port=<port>` to specify a Redis port (default is 6379).

   b. Execute the `start.sh` or `start.bat` file.

   c. Visit `http://localhost:8082/swagger-ui.html` to confirm the setup is working.

      If the `swagger-ui` page is loaded, then the Data Generator is up and running.

3. Load the sample data—

   Data sequences can be loaded from files. You can load `demo-dataset.csv` by running `load-demo-dataset.sh` (for Linux) or `load-demo-dataset.bat` (for Windows). Verify that the data is loaded by visiting `http://localhost:8082/Sequences`, and confirm that the page shows data.

   Note

   On Windows, if you do not have curl installed, you can download it from the following URL: https://curl.haxx.se/download.html. Place the folder containing the curl executable in your system PATH environment variable.

Installing the Asset Simulator

Unzip the `asset-simulator-<version>.zip` file into a directory of your choice. It contains the following files:

- `asset-sim-init.json`
- `ptc-cslm-asset-simulator.jar`
- `start.bat`
- `start.sh`

Running the Asset Simulator

1. Start the Asset Simulator—

   a. Execute the `start.sh` file (for Linux) or `start.bat` file (for Windows).

b. Visit `http://localhost:8083/AssetSimulator`. You should see the interface described in Using the Asset Simulator on page 14.

2. Command-line flags—

   The Asset Simulator can be executed with several command-line flags which can be added to the start script. These include:

   - `--server.port=<port>` to change the API/UI port from 8083. (The URLs in these instructions assume that the Asset Simulator is installed at `localhost:8083`.)
   - `--logging.file=<file>` to log to a file (in addition to standard console logging).
   - `--debug` for more detailed logging.
   - `--simulator.config=<file>` to specify a non-default name or location for the configuration file.

Importing the Kepware Project File

Import the `KepwareDemoAssetsProject.json` file into KEPServerEX.

Importing the Example Assets

1. In ThingWorx Composer, navigate to **Import/Export** ▸ **Import** ▸ **From File**.
2. On the **Import From File** window, ensure that **Entities** is selected, and click **Choose File**.
3. Navigate to and select the `demo-assets.xml` file.
4. Click **Import**. A success message appears.
5. Click **Close**.

3

Asset Simulator Configuration

The Asset Simulator configuration file (typically named `asset-sim-init.json`) names the assets to be simulated and their associated sensors, and maps those sensors to Data Generator data sequences. It also describes the location of the Data Generator and the configuration of the Asset Simulator OPC UA server.

Create new simulated assets by adding them to the configuration file.

Note

For information on working with the example assets in the Asset Simulator, see Using the Asset Simulator on page 14.

The configuration file uses JSON format. The configurable properties of the top-level JSON object are listed below. Properties are JSON strings unless otherwise noted.

imports

List of strings. This is the only property not shown in the example configuration. It allows partial configurations to be imported from other files. Example syntax is `"imports": ["import1.json", "import2.json"]`.

sensorTypes

List of objects with the following properties. Each sensor type represents a specific sensor on an asset, not just a kind of sensor. For instance, if an asset is fitted with three vibration sensors, each one needs its own sensor type definition. Multiple sensor types can share the same Data Generator sequences.

- `name`—The name of the OPC node for the sensor, also referenced elsewhere in this file.
- `updateIntervalMs`—Time delay before presenting each new data point, in milliseconds.
- `normalSequenceName`—The name of the Data Generator sequence providing this sensor's data while it is in a normal state.
- `anomalySequenceName`—The name of the Data Generator sequence providing this sensor's data while it is in an anomalous state.

assetTypes

List of objects with the following properties:

- `name`—Name of the asset type. Used by asset definitions elsewhere in this file.
- `sensorTypes`—List of Strings. The names of the sensor types on each asset of this type.
- `sensorGroups`—Optional List of Objects. Named groups of sensors which all become anomalous or failed due to a single underlying cause. This is purely a convenience to allow users to change multiple sensors' states at once. The properties of each sensor group object are `name` and `sensors`.

assets

List of objects with the following properties:

- `name`—The name of the OPC folder representing the asset and containing its sensor nodes.
- `assetType`—The asset type, determining the sensors and sensor groups for the asset.

assetBatches

List of objects with the following properties. All assets created in a batch present the same data for a given sensor in a given state, which allows large numbers of assets to be simulated efficiently. Each asset name has the same prefix and a zero-padded numeric suffix from 1 to the asset count. That is, setting name to "Pump" and count to "1000" creates assets named Pump0001 through Pump1000.

- `name` – Prefix for names of assets to create.
- `assetType` – The asset type, determining the sensors and sensor groups for the assets.
- `count` – The number of assets to create.

dataGeneratorLocation

The host and port of the Data Generator.

opcHostname

The hostname which to be used by OPC clients for the host machine.

dataBufferMinSeconds

To prevent delays, the Asset Simulator buffers data for each sensor. When the buffer size drops below this level (calculated with `updateIntervalMs`), the simulator requests additional data from the Data Generator.

dataBufferMaxSeconds

When the Asset Simulator makes a request to the Data Generator, it requests enough data to fill its buffer to this level.

sequenceLifetimeHours

The Asset Simulator requests that the Data Generator make temporary copies of its data sequences for each sensor, to avoid collisions in time series requests. This controls the lifetime of the temporary copies.

opcApplicationURI

opcBindPort

opcServerName

opcApplicationName

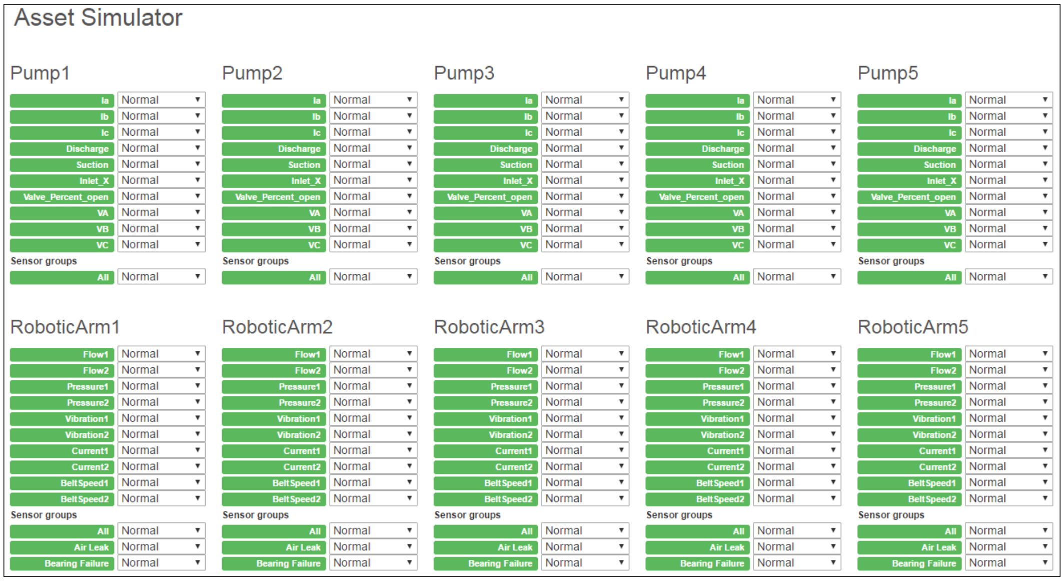
These properties control the identity of the simulated OPC server.

4

Using the Asset Simulator

The Asset Simulator user interface is available at `<host>:<port>/AssetSimulator`, for example: `http://localhost:8083/AssetSimulator`.

The assets, sensors, and sensor groups available in the Asset Simulator UI are largely determined by its configuration file. For more information, see Asset Simulator Configuration on page 11.



Asset Simulator

Pump1

Ia Normal
Ib Normal
Ic Normal
Discharge Normal
Suction Normal
Inlet_X Normal
Valve_Percent_open Normal
VA Normal
VB Normal
VC Normal
Sensor groups
All Normal

Pump2

Ia Normal
Ib Normal
Ic Normal
Discharge Normal
Suction Normal
Inlet_X Normal
Valve_Percent_open Normal
VA Normal
VB Normal
VC Normal
Sensor groups
All Normal

Pump3

Ia Normal
Ib Normal
Ic Normal
Discharge Normal
Suction Normal
Inlet_X Normal
Valve_Percent_open Normal
VA Normal
VB Normal
VC Normal
Sensor groups
All Normal

Pump4

Ia Normal
Ib Normal
Ic Normal
Discharge Normal
Suction Normal
Inlet_X Normal
Valve_Percent_open Normal
VA Normal
VB Normal
VC Normal
Sensor groups
All Normal

Pump5

Ia Normal
Ib Normal
Ic Normal
Discharge Normal
Suction Normal
Inlet_X Normal
Valve_Percent_open Normal
VA Normal
VB Normal
VC Normal
Sensor groups
All Normal

RoboticArm1

Flow1 Normal
Flow2 Normal
Pressure1 Normal
Pressure2 Normal
Vibration1 Normal
Vibration2 Normal
Current1 Normal
Current2 Normal
BeltSpeed1 Normal
BeltSpeed2 Normal
Sensor groups
All Normal
Air Leak Normal
Bearing Failure Normal

RoboticArm2

Flow1 Normal
Flow2 Normal
Pressure1 Normal
Pressure2 Normal
Vibration1 Normal
Vibration2 Normal
Current1 Normal
Current2 Normal
BeltSpeed1 Normal
BeltSpeed2 Normal
Sensor groups
All Normal
Air Leak Normal
Bearing Failure Normal

RoboticArm3

Flow1 Normal
Flow2 Normal
Pressure1 Normal
Pressure2 Normal
Vibration1 Normal
Vibration2 Normal
Current1 Normal
Current2 Normal
BeltSpeed1 Normal
BeltSpeed2 Normal
Sensor groups
All Normal
Air Leak Normal
Bearing Failure Normal

RoboticArm4

Flow1 Normal
Flow2 Normal
Pressure1 Normal
Pressure2 Normal
Vibration1 Normal
Vibration2 Normal
Current1 Normal
Current2 Normal
BeltSpeed1 Normal
BeltSpeed2 Normal
Sensor groups
All Normal
Air Leak Normal
Bearing Failure Normal

RoboticArm5

Flow1 Normal
Flow2 Normal
Pressure1 Normal
Pressure2 Normal
Vibration1 Normal
Vibration2 Normal
Current1 Normal
Current2 Normal
BeltSpeed1 Normal
BeltSpeed2 Normal
Sensor groups
All Normal
Air Leak Normal
Bearing Failure Normal

Changing sensor states

- To set the state of a sensor on an asset, click on its corresponding drop-down list, and select the new state.
- To set the state of all sensors configured in a group on an asset, click on the group's drop-down list, and select the new state.

Sensor behaviors

- **Normal**—Sensor presents normal data, as defined by the configuration file.
- **Anomalous**—Sensor presents anomalous data, as defined by the configuration file.
- **Failed**—Sensor presents the `BadSensorFailure` OPC UA error code.
- **Disconnected**—Sensor presents the `BadDisconnect` OPC UA error code.
- **Maintenance**—Sensor presents the `BadOutOfService` OPC UA error code.