# Environment preparing

Step 1 – Create four devices in the target IoT Hub, with tag ‘OEMName’ and desired property ‘fwversion’

IoTHubConsole /a:AddDevices /d:Dev1 /n:tags.OEMName /v:"ACME Inc." /n:fwversion /v:0.9

IoTHubConsole /a:AddDevices /d:Dev2 /n:tags.OEMName /v:"ACME Inc." /n:fwversion /v:1.0

IoTHubConsole /a:AddDevices /d:Dev3 /n:tags.OEMName /v:"ACME Inc." /n:fwversion /v:1.1

IoTHubConsole /a:AddDevices /d:Dev4 /n:tags.OEMName /v:"MS Inc." /n:fwversion /v:1.2

Step 2 – Add desired property ‘temp’ for each devices

IoTHubConsole /a:UpdateTwin /d:Dev1 /n:temp /v:50

IoTHubConsole /a:UpdateTwin /d:Dev2 /n:temp /v:60

IoTHubConsole /a:UpdateTwin /d:Dev3 /n:temp /v:70

IoTHubConsole /a:UpdateTwin /d:Dev4 /n:temp /v:80

Step 3 - Start device simulators

Command line: node app.js <IoTHub name> <deviceId> <device key>

# Scenario 1: Querying for Devices

Test 1: Query for devices which fwversion >= 1.0 and OEMName = ACME Inc.

IoTHubConsole /a:QueryDevices /q:"select \* from devices where properties.desired.fwversion >= '1.0' AND tags.OEMName = 'ACME Inc.'"

Test 2: Query for devices which desired temp >= 70

IoTHubConsole /a:QueryDevices /q:"select \* from devices where properties.desired.temp >= 70"

Test 3: Query for devices which reported temp >= 70

(Device application will copy desired property to reported. So there will be a reported property temp = 70 now)

IoTHubConsole /a:QueryDevices /q:"select \* from devices where properties.reported.temp >= 70"

# Scenario 2: Device Workflows

Test 1: Invoke firmware update on single device via scheduled job

IoTHubConsole /a:ScheduleDeviceMethod /d:Dev3 /n:firmwareUpdate /v:"{'fwPackageUri':'https://secureurl'}"

Test 2: Invoke firmware update on devices which fwversion >= 1.1, with timeout = 60, start offset = 60

IoTHubConsole /a:ScheduleDeviceMethod /q:"properties.desired.fwversion >= '1.1'" /n:firmwareUpdate /v:"{'fwPackageUri':'https://secureurl'}" /t:60 /o:60

Test 3: Directly invoke firmware update on single device

IoTHubConsole /a:InvokeMethod /d:Dev3 /n:firmwareUpdate /v:"{'fwPackageUri':'https://secureurlDev3'}"

Test 4: Query for jobs with given job ID

IoTHubConsole /a:QueryJobs /q:"select \* from devices.jobs where devices.jobs.jobId = '{JobID}'"

Test 5: Try to schedule firmware update with a false query condition. No device should be updated

IoTHubConsole /a:ScheduleDeviceMethod /q:"properties.desired.fwversion >= '3.1'" /n:firmwareUpdate /v:"{'fwPackageUri':'https://secureurl'}" /t:20

Test 6: Schedule firmware update on offline devices

IoTHubConsole /a:ScheduleDeviceMethod /q:"properties.desired.fwversion <= '1.0'" /n:firmwareUpdate /v:"{'fwPackageUri':'https://secureurl'}" /t:20

Test 7: Directly invoke firmware update on offline device (turn off simulator for Dev1)

IoTHubConsole /a:InvokeMethod /d:Dev1 /n:firmwareUpdate /v:"{'fwPackageUri':'https://secureurlDev3'}"

Exception raised: System.AggregateException: One or more errors occurred. ---> Microsoft.Azure.Devices.Common.Exceptions.ServerErrorException: {"Message":"ErrorCode:GenericTimeout;InternalServerError","ExceptionMessage":"Tracking ID:de5d714f93d94fd99f82e685270f18ad-G:7-TimeStamp:10/21/2016 08:41:51"}

Test 8: Directly invoke firmware update on unregistered device

IoTHubConsole /a:InvokeMethod /d:Dev0 /n:firmwareUpdate /v:"{'fwPackageUri':'https://secureurlDev3'}"

Exception raised: System.AggregateException: One or more errors occurred. ---> Microsoft.Azure.Devices.Common.Exceptions.DeviceNotFoundException: Device {"Message":"ErrorCode:DeviceNotFound;Dev0","ExceptionMessage":"Tracking ID:78d8180a9045468a86a958182089e453-G:9-TimeStamp:10/21/2016 08:41:48"} not registered

Test 9: Query for all jobs

IoTHubConsole /a:QueryJobs

Test 10: Query for device specific jobs

IoTHubConsole /a:QueryJobs /q:"select \* from devices.jobs where devices.jobs.deviceId='Dev3'"

Test 11: Query for type specific jobs

IoTHubConsole /a:QueryJobs /q:"select \* from devices.jobs where devices.jobs.jobType = 'scheduleUpdateTwin'"

Test 12: Query for status specific jobs

IoTHubConsole /a:QueryJobs /q:"select \* from devices.jobs where devices.jobs.status='completed'"

No job returned

\*\*\*Not Found features\*\*\*

Device Workflows (Event based)

7. CRAWL - List available device workflows and the number of devices that support each.

8. WALK - For a specific workflow, show a list of devices that support and do not support that workflow.

9. CRAWL - For a specific device, show a list of workflows that can be executed on that device.

10. RUN - Define a custom workflow using protocol specific device activities.

11. RUN - Register, Update, and Remove a custom workflow.

Test 13: Update twin on single device via scheduled job

IoTHubConsole /a:ScheduleTwinUpdate /d:Dev3 /n:temp /v:100

(Since device simulator was online, the reported property was refreshed as expected)

Test 14: Update twin on devices which fwversion >= 1.1, with timeout and start offset

IoTHubConsole /a:ScheduleTwinUpdate /q:"properties.desired.fwversion >= '1.1'" /n:temp /v:120 /t:60 /o:60

Test 15: Directly update twin on single device

IoTHubConsole /a:UpdateTwin /d:Dev3 /n:temp /v:130

Test 16: Query for jobs with job ID

IoTHubConsole /a:QueryJobs /q:"select \* from devices.jobs where devices.jobs.jobId = '{JobID}'"

Test 17: Schedule twin update job with false query condition. No device should be updated

IoTHubConsole /a:ScheduleTwinUpdate /q:"properties.desired.fwversion >= '3.1'" /n:temp /v:140 /t:20

# Scenario 3: Tags

Test 1: Add tag Location for devices which OEMName = ACME Inc.

IoTHubConsole /a:ScheduleTwinUpdate /q:"tags.OEMName = 'ACME Inc.'" /n:tags.Location /v:Beijing

Test 2: Remove tag Location for devices which fwverion > 1.0

IoTHubConsole /a:ScheduleTwinUpdate /q:"properties.desired.fwversion > '1.0'" /n:tags.Location /v:null

Test 3: Add tag Location on single device

IoTHubConsole.exe /a:UpdateTwin /d:Dev4 /n:tags.Location /v:Beijing

Test 4: Query for devices which Location = Beijing

IoTHubConsole /a:QueryDevices /q:"select \* from devices where tags.Location = 'Beijing'"

Test 5: Query for devices which Location = Beijing and OEMName = ACME Inc.

IoTHubConsole /a:QueryDevices /q:"select \* from devices where tags.Location = 'Beijing' AND tags.OEMName = 'ACME Inc.'"

# Scenario 4: Others

Test 1: Cancel queued job

IoTHubConsole /a:CancelJobs /n:{jobId}

(canceled job will not be recorded)

Test 2: Send C2D message

IoTHubConsole /a:SendMessage /d:Dev1 /m:Hello

# Summary Bugs

|  |  |
| --- | --- |
| Exception on device | Repro Steps:   1. Run a node.js script with "client.open(function (err) {…});" 2. Wait for a couple hours (1 or 2 hours)     Expected:  The device script should be still functional.    Observed:  Got exception on device side as following. And it can't receive message from service any more.    Exception:  c:\Temp\TestDM\DeviceApp\node\_modules\azure-iot-device-mqtt\node\_modules\azure-iot-mqtt-base\lib\mqtt.js:74  this.client = this.mqttprovider.connect(this.\_hostName, this.\_options);  ^    TypeError: this.mqttprovider.connect is not a function  at Mqtt.connect (c:\Temp\TestDM\DeviceApp\node\_modules\azure-iot-device-mqtt\node\_modules\azure-iot-mqtt-base\lib\mqtt.js:74:35)  at Mqtt.connect (c:\Temp\TestDM\DeviceApp\node\_modules\azure-iot-device-mqtt\lib\mqtt.js:49:14)  at Client.open (c:\Temp\TestDM\DeviceApp\node\_modules\azure-iot-device\lib\client.js:323:21)  at null.<anonymous> (c:\Temp\TestDM\DeviceApp\node\_modules\azure-iot-device\lib\client.js:283:14)  at Mqtt.<anonymous> (c:\Temp\TestDM\DeviceApp\node\_modules\azure-iot-device-mqtt\lib\mqtt.js:148:7)  at Store.close (c:\Temp\TestDM\DeviceApp\node\_modules\azure-iot-device-mqtt\node\_modules\azure-iot-mqtt-base\node\_modules\mqtt\lib\store.js:104:5)  at c:\Temp\TestDM\DeviceApp\node\_modules\azure-iot-device-mqtt\node\_modules\azure-iot-mqtt-base\node\_modules\mqtt\lib\client.js:521:26  at Store.close (c:\Temp\TestDM\DeviceApp\node\_modules\azure-iot-device-mqtt\node\_modules\azure-iot-mqtt-base\node\_modules\mqtt\lib\store.js:104:5)  at Immediate.closeStores (c:\Temp\TestDM\DeviceApp\node\_modules\azure-iot-device-mqtt\node\_modules\azure-iot-mqtt-base\node\_modules\mqtt\lib\client.js:520:24) |
| Twin update job failed while no tag in the target twin | Repro steps   1. Create an IoTHub with DM support in Global Azure 2. Add a new device in the new created IoTHub 3. Set a Tag of the new added device, e.g. city = Beijing 4. Schedule a twin update job via the C# SDK to update a desired property, with a query condition on the new added Tag. The updating twin MUST NOT contain any Tag (we expect it means no tag need be changed) 5. Poll the job status until it was completed or failed.     Expected:  The job should be completed after few seconds    Observed:  The job failed with error message: ‘Specified cast is not valid’. And if we added any tag in the updating twin (No matter if the tag was new added. No matter if the tag value is same to current value), the job will complete as expected.    Source code:  var client = JobClient.CreateFromConnectionString(connectionString);  var jobId = Guid.NewGuid().ToString();  var twin = new Twin();  twin.Properties.Desired["targetTTL"] = 150;  // The job will complete if we uncomment any one of the two lines below  // twin.Tags["city"] = "Beijing";  // twin.Tags["country"] = "China";  twin.ETag = "\*";    var job = await client.ScheduleTwinUpdateAsync(jobId, "tags.city = 'Beijing'", twin, DateTime.UtcNow, 3600);    Location: East US |
| Cannot use job status as query condition | Job query “Select \* from devices.jobs where device.jobs.status = ‘completed’” returns no job. While the other queries on jobId, deviceId and jobType works well. |