Table of Contents

Getting Started

IoT is driving new opportunities. The 3M IoT Platform accelerates innovation by providing reusable services essential to every IoT solution.

This Getting Started section provide a high-level overview of what the 3M IoT Platform provides and how it can be used. The Guides section offers hands-on related information.

Use the Download to PDF button to take the documentation offline or print it.

TODO: Cards with common needs should go here.

> Note: Need to create custom template for this

3M IoT Platform Overview

3M's IoT Platform on Azure offers an end-to-end solution for acceleration innovation and time-to-market by offering core sources that can be integrated into custom IoT Solutions. Device Management, Security, and cloud resource provisioning and other core requirements of any IoT solution can use the platform's user interface to manage devices. In addition, platform APIs offer direct integration to allow development to focus on their unique business objectives instead of common requirements. The platform combines Azure and custom services to pride a generic implementation ready for any IoT business scenario.

The platform code is available on GitHub where new features can be tracked and requested using the issues list. Currently requests are restricted to 3M. To learn how to request features or report issues, see the documentation TODO.

Logical Architecture

The following diagram outlines the logical components of the platform overlaid on the IoT architecture:
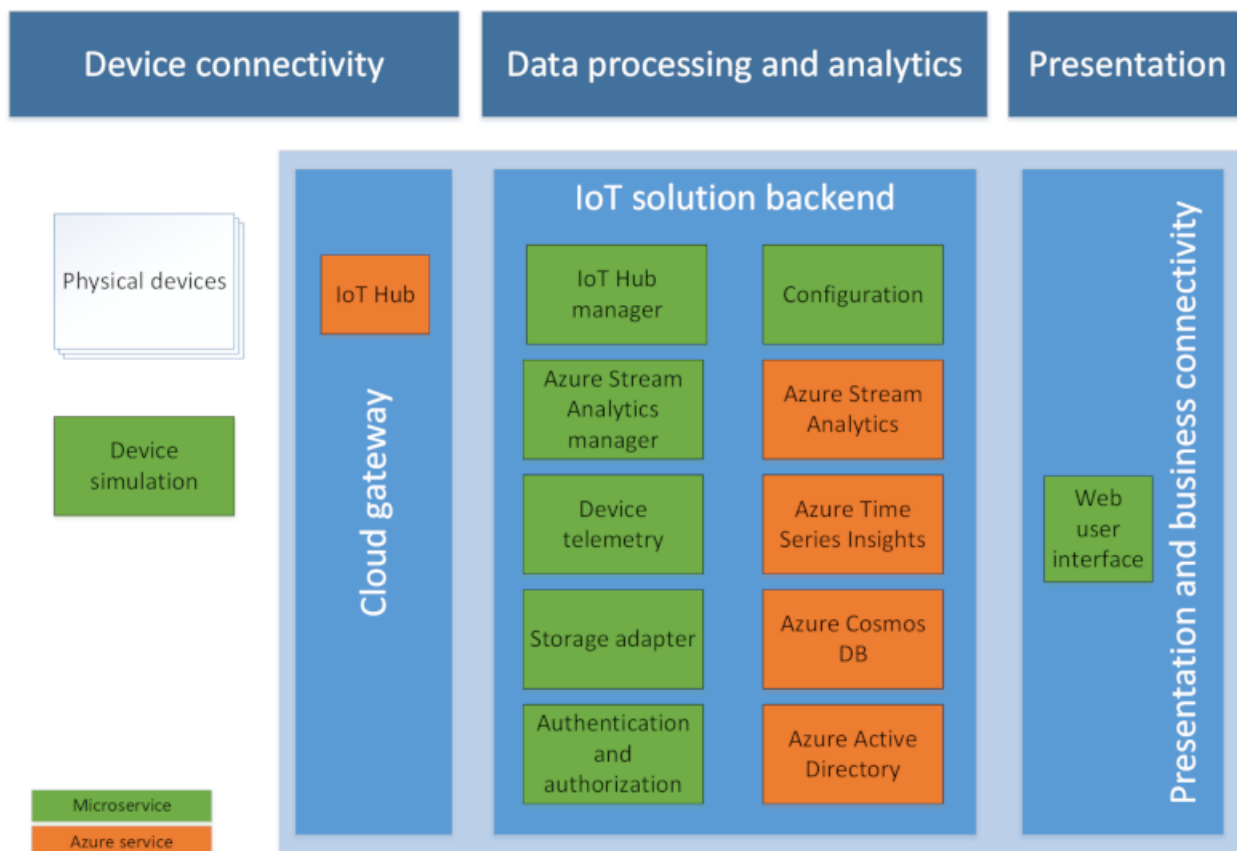
IoT Hub

The IoT Hub is a Microsoft Azure Resource used to ingest telemetry sent from both real and siulated devices into the cloud. The hub makes the teletry available to the services in the IoT solution for processing.

The IoT Hub also provides the following:

- Maintains an identity registry that stores the IDs and authentication keys of all the devices permitted to connect to the portal.
- Invokes methods on your devices on behalf of the solution accelerator.
- Maintains device twins for all registered devices. A device twin stores the property values reported by a device. A device twin also stores desired properties, set in the solution portal, for the device to retrieve when it next connects.
- Schedules jobs to set properties for multiple devices or invoke methods on multiple devices.

Presentation

A web user interface is available for device management. It offers a user friendly way to register and configure devices, deploy firmware, and manage alerts, and provides a dashboard for generally observing device telemetry.

The web user interface is a React Javascript application that:

- Is served to the browser via backend services
- Is styled with Css and SaSS
- Interacts with other backend services like the authentication and authorization service to protect user data and communicate with Azure resources

Backend Services

The platform is made up several services running in Azure Kubernetes Service (AKS). The platform containerizes services to offer enterprise-grade security and scalability. The services - often called *microservices*- are written in .NET (C#) and provide RESTful endpoints that can be used for direct integration in business specific solutions.

Data processing and analytics

The solution includes the following components in the data processing and analytics part of the logical architecture:

IoT Hub manager

The solution includes the IoT Hub manager microservice to handle interactions with your IoT hub such as:

Creating and managing IoT devices. Managing device twins. Invoking methods on devices. Managing IoT credentials. This service also runs IoT Hub queries to retrieve devices belonging to user-defined groups.

The microservice provides a RESTful endpoint to manage devices and device twins, invoke methods, and run IoT Hub queries.

Device telemetry

The device telemetry microservice provides a RESTful endpoint for read access to device telemetry stored in Time Series Insights. The RESTful endpoint also enables CRUD operations on rules and read/write access for alarm definitions from storage.

Storage adapter

The storage adapter microservice manages key-value pairs, abstracting the storage service semantics, and presenting a simple interface to store data of any format using Azure Cosmos DB.

Values are organized in collections. You can work on individual values or fetch entire collections. Complex data structures are serialized by the clients and managed as simple text payload.

The service provides a RESTful endpoint for CRUD operations on key-value pairs. values

Azure Cosmos DB

Deployments use Azure Cosmos DB to store rules, alerts, configuration settings, and all other cold storage.

Azure Stream Analytics

The Azure Stream Analytics manager microservice manages Azure Stream Analytics (ASA) jobs, including setting their configuration,starting and stopping them, and monitoring their status.

The ASA job is supported by two reference data sets. One data set defines rules and one defines device groups. The rules reference data is generated from the information managed by the device telemetry microservice. The Azure Stream Analytics manager microservice transforms telemetry rules into stream processing logic.

The device groups reference data is used to identify which group of rules to apply to an incoming telemetry message. The device groups are managed by the configuration microservice and use Azure IoT Hub device twin queries.

The ASA jobs deliver the telemetry from the connected devices to Time Series Insights for storage and analysis.

Azure Stream Analytics is an event-processing engine that allows you to examine high volumes of data streaming from devices.

Azure Time Series Insights

Azure Time Series Insights stores the telemetry from the devices connected to the solution accelerator. It also enables visualizing and querying device telemetry in the solution web UI.

Configuration microservice

The configuration microservice provides a RESTful endpoint for CRUD operations on device groups, solution settings, and user-settings in the solution accelerator. It works with the storage adapter microservice to persist the configuration data.

Authentication and authorization microservice

The authentication and authorization microservice manages the users authorized to access the solution accelerator. User management supports a variety of identity service provides that support OpenId Connect, including Azure B2C.

Reference content from the following:

https://azure.microsoft.com/en-us/features/iot-accelerators/#iot-accelerators-features
file:///C:/Users/jonat/Desktop/3M%20Temp/Microsoft_Azure_IoT_Reference_Architecture.pdf https://docs.microsoft.com/en-us/azure/iot-accelerators/iot-accelerators-connecting-devices https://docs.microsoft.com/en-us/azure/iot-accelerators/iot-accelerators-remote-monitoring-monitor

https://docs.microsoft.com/en-us/azure/iot-accelerators/iot-accelerators-remote-monitoring-architectural-choices

TODO: Move to appropriate location:

Resources

COLUMN1	COLUMN2
IotT Solution	- Source Code: https://github.mmm.com/MMM/bluebird - Docs: https://github.mmm.com/pages/MMM/bluebird/
Citrix VM	http://3mcitrix.mmm.com
Base URL for Solution:	https://crsliot-aks-dev.centralus.cloudapp.azure.com
Odin Repos for Azure IoT	- https://github.mmm.com/pages/MMM/bluebird/ - https://github.mmm.com/orgs/mmm/teams/valkyrie/members - **Public:** 3M-Company/azure-iot-platform-dotnet
Azure DevOps	https://dev.azure.com/3m-bluebird/azureplatform
Microsoft Team Channels:	- **3M:** Azure Platforms, Azure Platform Development - **ACS:** 3M Digital (GGK + Analyst Channel) Project Team
3M Docker Hub	https://hub.docker.com/azureiot3m

Repositories

GitHub Enterprise

- Code and Infrastructure (https://github.mmm.com/MMM/azure-iot-services-dotnet)
- Azure Functions (https://github.mmm.com/MMM/azure-iot-messaging-functions)
- App Config (https://github.mmm.com/MMM/azure-iot-appconfig-function)
- Device Migration (https://github.mmm.com/MMM/azure-iot-device-migration)
- Functional Testing (https://github.mmm.com/MMM/azure-iot-functional-tests)

TODO

- Add Device model schema: https://docs.microsoft.com/en-us/azure/iot-accelerators/iot-accelerators-remote-monitoring-device-schema
- Add Device model behavior: https://docs.microsoft.com/en-us/azure/iot-accelerators/iot-accelerators-remote-monitoring-device-behavior
- Direct users to Device guides

How to Make Platform Improvements

The 3M IoT Platform on Azure has a robust CI/CD process to accommodate feedback and contributes from the community.

The IoT Platform source code is stored in a public GitHub repo and developed under an open-source model. The following table describes the common ways to initialize improvements to the platform:

RESOURCE	DESCRIPTION
Issues List	Offers the following: - Report Bugs - Request Features - Influence Priorities - Track Progress
Documentation	All documentation offers community contributions.
Development	You can make direct contributions to the code.
Community Involvement	Join the 3M IoT Platform Community.

Use the links below to get more information on each topic.

Issues List

- How to Report a Bug
- How to Request a Feature
- How to Influence Priorities
- How to Track Development
- How to Track Releases

Documentation

- How to Improve Documentation

Development

- How to Make Code Contributions

Community Involvement

- How to Make Code Contributions

Getting Started for Web Developers

This document provides setup instructions for contributing to the 3M IoT Platform as a *Web Developer*. In addition, it outlines relevant skills and helpful links to learn about or brush up on related topics.

Useful Skills

Be familiar with the following:

- JavaScript
- CSS/Sass
- ReactJS (16.6.3)
- Redux (5.1.1)
- IoT
- Azure IoT UX Fluent Controls
- Git

Common Contributions

Web Developers typically contribute to the platform in the following ways:

- Enhancing the web User Interface
- Communicating with the Backend Services

Tools to Install

- VS Code or other IDEs
- Node 10.14.1 (or above)
- NPM 6.4.1 (or above)
- Git

Helpful Resources

- IoT School
- Microsoft IoT
- ReactJS
- Redux (a ReactJS event management tool)
- Fluent UI: IoT Controls
- Git

Getting Started for API Developers

This document provides setup instructions for contributing to the 3M IoT Platform as an *API Developer*. In addition, it outlines relevant skills and helpful links to learn about or brush up on related topics.

Useful Skills

- C#
- .Net core
- IoT
- Azure services- IoTHub, Azure Storage, Cosmos DB
- Containerization
- Dev spaces (optional)
- Azure functions
- Kubernetes (optional)
- Docker (optional)
- Git

Common Contributions

API Developers typically contribute to the platform in the following ways:

- Updating/enhancing back-end services
- Building service to interact with IoTHub to configure devices and provide configuration updates to devices via IoTHub
- Building Analytics to check for anamloies
- Services to store data for Audit

Tools to Install

- Visual Studio 2019 (or above) / VS Code
- Docker desktop (optional)
- Azure CLI (optional)
- Git

Helpful Resources

- IoT School
- Microsoft IoT
- Azure CLI
- Azure Storage
- Cosmos DB
- Azure IoT Hub
- Git

Getting Started for Azure Developers

This document provides setup instructions for contributing to the 3M IoT Platform as a *Azure Developer*. In addition, it outlines relevant skills and helpful links to learn about or brush up on related topics.

Useful Skills

Be familiar with the following list of Azure services

- AKS
- Storage Account
- DPS (Device Provisioning Service)
- Function App
- App Configuration
- Cosomos DB
- Event Hub
- Key Vault
- Send Grid
- IOT Hub
- Stream Analytics Job
- Azure Maps

Common Contributions

Azure Developer typically contribute to the platform in the following ways:

- Should have knowledge on Azure Services and how to deploy code to these services.
- Enhancing the Azure Services and integration between the services.

Tools

- Az cli
- Azure Storage Explorer
- Azure Devspaces cli and extension
- Azure IOT Explorer
- VS Code
- Docker

Helpful Resources

- IoT School
- Microsoft IoT
- Azure Storage Explorer
- Az Cli
- Devspaces

Getting Started for DevOps Engineers

This document provides setup instructions for contributing to the 3M IoT Platform as a *DevOps Engineer*. In addition, it outlines relevant skills and helpful links to learn about or brush up on related topics.

Useful Skills

Helpful Resources

Tools

Getting Started for Document Contributors

This document provides setup instructions for contributing to the 3M IoT Platform as a *Document Contributor*. In addition, it outlines relevant skills and helpful links to learn about or brush up on related topics.

Useful Skills

- Markdown
- IoT (optional)
- Git

Common Contributions

Documentation Contributors typically contribute in the following ways:

- Add new content to the documentation
- Improve or organize the existing documentation

Tools to Install

- VS Code
- Docfx
- Git

Helpful Resources

- Docfx User manual
- Docfx - API Documentation
- Docfx - Rest API
- IoT School
- Microsoft IoT
- Git

3M IoT Development Environment Setup Instructions

This document provides detailed documentation on setting up a development environment to contribute to 3M's IoT Azure Platform, called Bluebird. Note: If you're using a 3M Virtual Machine, it's possible some of the products to install may exist already.

Products to Install

Please install the following products in the order listed below. Click the project name to open the specific steps for each product below. Wherever relevant, OS specific steps will be provided for both Mac and PC. I would like to also point out that we are not allowed to be admins on our 3M machines, we use elevated access. This is accomplished by right clicking and selecting the elevated access option.

Note: Install Issues not specifically addressed in the specific product install guides below should be recorded here. Since we have moved into an open source environment for Odin it is not unreasonable to use our own machines for dev. Sometimes this proves less restrictive and has better up-time.

Product Install Guides (by install order)

- Virtualization (only necessary if developing on Windows)
- .NET Core 3.1
- PowerShell
- Azure CLI (plus extensions)
- Visual Studio Code
- Git / Bash
- NodeJS / npm
- Visual Studio 2019
- Azure Data Studio
- Storage Explorer
- Azure IoT Explorer
- Docker
- Terraform
- Kubernetes
- Helm
- Redux
- Postman
- Source Code

Whats new in VS 2019

Check the details here
Use the community edition if you don't have a licensed version: https://visualstudio.microsoft.com/downloads/

Download

Download Visual Studio 2019 from here

Overview

If your development environment runs on Windows you must setup virtualization in order to run docker containers locally since some of them run on Linux which Windows does not natively support. This is not necessary on a MacOS since it's based on Unix. Which virtualization product you use is a matter of preference and is typically influence by the OS of your host machine. Historically, Hyper-V was popular for servers and VirtualBox on clients.

Virtualization Product

PRODUCT	VENDOR	GUIDES
Hyper-V	Microsoft	MS Install Guide
Virtual Box	Oracle	- Docker for Mac - Using Docker with Virutal Box on Windows

Prepare Virtual Machine using Hyper-V

1. Enable Hyper-V. Click for more details.
2. Create a Virtual Machine. Click for more details. Note: Microsoft offers instances of Ubuntu and an evaluation copy of Windows. Currently the evaluation only last 5 days so it's not a great option unless you have an Win 10 Enterprise Key to use to upgrade. Otherwise you need an ISO and License Key for the OS you intend to install. Using the eval requires a 16GB download.
3. Prepare Installation Media
4. Create Virtual Machine

Prepare Installation Media

If you're going to use your own copy of an operating system, you need to first create an ISO so Hyper-V or VirtualBox can use it to build the Virtual Machine. Microsoft provides a useful tool called **Create Windows 10 installation media** that can be downloaded here https://www.microsoft.com/en-us/software-download/windows10. The link includes instructions on how to use.
The above approach can also be used for Windows 8.1 and Windows 7. Alternatively there are several open source and 3rd party products that can be used to create ISO.

Create Virtual Machine

Once you have your ISO, you can start the create virtual machine process. To continue, follow these steps:
https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/quick-start/quick-create-virtual-machine

Alternative

As untried alternative to virtualization, if you're using Windows you could try the Linux subsystem for Windows.
Note: We have not tried this. Key consideration is whether or not docker images can be instantiated.

Use Windows Subsystem for Linux for production: https://docs.microsoft.com/en-us/windows/nodejs/setup-on-windows#use-windows-subsystem-for-linux-for-production

The Windows Subsystem for Linux, introduced in the Anniversary Update, became a stable feature in the Fall Creators Update. You can now run Ubuntu and openSUSE on Windows, with Fedora and more Linux distributions coming soon.

This documented might be a good guide: https://docs.microsoft.com/en-us/virtualization/windowscontainers/quick-start/set-up-environment?tabs=Windows-10-Client

Overview

.NET Core is a cross-platform successor to the .NET Framework that runs on Windows, Linux, and macOS operating systems. You can download from here. It was open sourced by Microsoft.

TODO

Did you choose 3.1.0 or 3.1.3 (latest as of 4/8)?

Overview

PowerShell is a command-line shell and associated scripting language that can run on on Windows, Linux and macOS as of version 7.

Installation Guides

- Windows
- macOS
- Linux
- Installing PowerShell in Azure Resources via ARM

To access the download packages go here: https://github.com/PowerShell/PowerShell

Additional

Windows PowerShell 7 is automatically part of Windows 10 IoT Enterprise. Additional details look here:
https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell-core-on-windows?view=powershell-7#deploying-on-windows-10-iot-enterprise

Overview

The Azure command-line interface (Azure CLI) is a set of commands used to create and manage Azure resources. The Azure CLI is available across Azure services and is designed to get you working quickly with Azure, with an emphasis on automation.

Click here to get more details Be sure to add additional extensions, see below.

Note: Requires Elevated Permissions on 3M's Virtual Machine

Install the Azure CLI

The Azure CLI is available to install in Windows, macOS and Linux environments. It can also be run in a Docker container and Azure Cloud Shell.

The current version of the Azure CLI is **2.3.1.** For information about the latest release, see the release notes. To find your installed version and see if you need to update, run `az --version`.

- Install on Windows
- Install on macOS
- Install on Linux or Windows Subsystem for Linux (WSL)
    - Install with apt on Debian or Ubuntu
    - Install with yum on RHEL, Fedora, or CentOS
    - Install with zypper on openSUSE or SLE
    - Install from script

- Run in Docker container
- Run in Azure Cloud Shell

Use extensions with Azure CLI

The Azure CLI offers the capability to load extensions. Extensions are Python wheels that aren't shipped as part of the CLI but run as CLI commands. With extensions, you gain access to experimental and pre-release commands along with the ability to write your own CLI interfaces.

Click here to get details on how to use extensions

Additional Extensions

Add the extensions listed below. To do so, use the following command:

```
az extension add --name {extension_name}
```

To see the list of available extensions and their details such as if they're enabed, run the following:

```
az extension list-available
```

Note: if any of your extensions need to be updated, you can swap the **add** commands with **update**.

Dev Spaces

- **Description:** Dev Spaces provides a rapid, iterative Kubernetes development experience for teams.
- **Command:** `az extension add --name dev-spaces`
- Documentation
    - **Note:** there is a preview version as well. dev-spaces-preview

IoT

- **Description:** Comprehensive data-plane functionality to manage Internet of Things (IoT) assets.
- **Command:** `az extension add --name azure-iot`
- [Documentation](#)

DevOps

- **Description:** Manage Azure DevOps organization level operations including pipelines, boards, repos, artifacts, etc.
- **Command:** `az extension add --name azure-devops`
- [Documentation](#)

Azure Kubernetes Service (AKS)

- **Description:** Manage Azure Kubernetes Services.
- **Command:** `az extension add --name aks-preview`
- [Documentation](#)

Introduction

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows. It is designed to handle everything from small to very large projects with speed and efficiency.
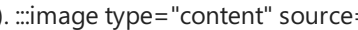
Pre-requisites

Be sure to installed VS Code first. Then you'll have the option of hooking Git into VS Code as you're editor.

> If you're going to use recommended NVM (Node Version Manager) on Windows, you'll need to **Enable symbolic links** during the installation. See settings section, install step **Configuring extra options** below for more information.

Git

Click here to download git for different OS.

Settings

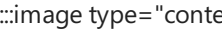Use the default values except on the screenshots below. For those steps, use the settings indicated in the screenshot (assuming Win Install). :::image type="content" source="../../../images/git-settings1.PNG" alt-text="Git settings 1":::

:::image type="content" source="../../../images/git-settings2.PNG" alt-text="Git settings 2":::

:::image type="content" source="../../../images/git-settings3.PNG" alt-text="Git settings 3":::

:::image type="content" source="../../../images/git-settings4.PNG" alt-text="Git settings 4":::

GitHub Desktop (optional)

Click here to download GitHub Desktop for macOS, Win, and Linux.

Connecting Your GitHub Repository to 3M

1. The repository you are look for is https://github.com/3Mcloud/azure-iot-platform-dotnet/
2. You will then select Fork in the top right hand corner as displayed below
   - :::image type="content" source="../../../images/git-repo-fork1.PNG" alt-text="Git repo Fork":::

3. This will then allow you to tie your personal account to the 3m cloud
4. You are able to verify that you are associated by clicking the number next to fork. It will display the different accounts attached as shown below.
   - :::image type="content" source="../../../images/git-repo-fork2.PNG" alt-text="Git repo Fork":::

5. You can see mine is highlighted, using this method you are able to raise PR's and also submit Reviews. Keep in mind we need two reviews for QA purposes.
6. You can also see what your teammates have been working on by clicking on their account.
7. You are able to sync data from your repo to the 3m master. There are 2 main ways to do this.
   - This way is done by the cli.
     - https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/syncing-a-fork

   - This is the way to do it through the web gui. I think this is the ideal way to do this. I will attach a video and screen shots.
     - https://youtu.be/YhwBgYPfoVE

- As you can see my repo is 15 commits behind.
- :::image type="content" source="../../../images/pull-request1.PNG" alt-text="Pull request 1":::
- After comparing it will tell me I am able to merge.
- :::image type="content" source="../../../images/pull-request2.PNG" alt-text="Pull request 2":::
- You would then submit a PR to sync your fork.
- :::image type="content" source="../../../images/pull-request3.PNG" alt-text="Pull request 3":::

CLI Approach to Refresh Local Master with Upstream Changes

Note: This approach will completely replace your local master (forked 3M repo) with changes upstream (3M Cloud):

git remote add upstream /url/to/original/repo
git fetch upstream
git checkout master
git reset --hard upstream/master
git push origin master --force

CLI Approach to Creating New Branch

By on a new copy of "master" (using above approach). Then create a new branch

```
git checkout -b 1245-fix(webui)-deployment-flyout
```

Introduction

Bash is the shell, or command language interpreter, for the GNU operating system. The name is an acronym for the 'Bourne-Again SHell', a pun on Stephen Bourne, the author of the direct ancestor of the current Unix shell sh, which appeared in the Seventh Edition Bell Labs Research version of Unix. Bash is largely compatible with sh and incorporates useful features from the Korn shell ksh and the C shell csh. It is intended to be a conformant implementation of the IEEE POSIX Shell and Tools portion of the IEEE POSIX specification (IEEE Standard 1003.1). It offers functional improvements over sh for both interactive and programming use. While the GNU operating system provides other shells, including a version of csh, Bash is the default shell. Like other GNU software, Bash is quite portable. It currently runs on nearly every version of Unix and a few other operating systems - independently-supported ports exist for MS-DOS, OS/2, and Windows platforms.

The improvements offered by Bash include:

- Command line editing
- Unlimited size command history
- Job Control
- Shell Functions and Aliases
- Indexed arrays of unlimited size
- Integer arithmetic in any base from two to sixty-four

Download

There are many ways to install bash. A couple options are as following:

GNU

Bash can be found on the main GNU ftp server: http://ftp.gnu.org/gnu/bash/ (via HTTP) and ftp://ftp.gnu.org/gnu/bash/ (via FTP). It can also be found on the GNU mirrors; please use a mirror if possible.

Click here for more details

Install with Git

You can get bash on windows by installing GIT. This might be the easiest approach for developers using Windows. During the install, choose Windows Explorer integration: :::image type="content" source="../../../images/git-install.PNG" alt-text="Git install":::

VS Code

Check out this: https://stackoverflow.com/questions/42606837/how-do-i-use-bash-on-windows-from-the-visual-studio-code-integrated-terminal

Overview

NodeJS is a extension from Joyant that builds on Google's V8 Engine that offers a very fast run-time enviornment for solutions built using JavaScript that can run both client-side (web and desktop - via Electron) and server side. The speed comes from Google's non-blocking IO model.

BE SURE TO TARGET VERSIONS: **NPM** 6.4.1 **NODE** 10.14.1 (local) 11.1 (prod)

We recommended using Node Version Manager (NVM) for Windows to be able to support switching between NodeJS versions (which also influences the active NPM version). We have found install issues on some Windows 10 Machines running McAfee. Follow these instructions to avoid install issues: https://medium.com/@tysonpaul89/maintain-multiple-versions-of-node-js-in-windows-operating-system-using-nvm-3c6bf5b63f29

Alternatively, to install NodeJS directly go to https://nodejs.org/, but you want be able to switch versions so be sure to install the correct version (see above).

Additional details about configuring and switching versions using NVM are below

Another helpful guide is: Set up your Node.js development environment directly on Windows

NVM Alternatives

If you have install issues with NVM or are running on a non-windows machine, try the following:

While windows-nvm is currently the most popular version manager for node, there are alternatives to consider:

nvs (Node Version Switcher) is a cross-platform nvm alternative with the ability to integrate with VS Code.
Volta is a new version manager from the LinkedIn team that claims improved speed and cross-platform support.
To install Volta as your version manager (rather than windows-nvm), go to the **Windows Installation** section of their Getting Started guide, then download and run their Windows installer, following the setup instructions.

Important: You must ensure that Developer Mode is enabled on your Windows machine before installing Volta.

To learn more about using Volta to install multiple versions of Node.js on Windows, see the Volta Docs.

Node Version Manager (NVM)

Use Node Version Manager (NVM) to support multiple versions:

- Windows: https://github.com/coreybutler/nvm-windows
- Linux and MAC: https://github.com/nvm-sh/nvm

Note

This solution uses older versions of NodeJS. Uses older versions will cause warning to be triggered like:

```
npm WARN npm npm does not support Node.js vXX.XX.XX These can be ignored.
```

It's best to do uninstall any direct node installs before installing NVM. Even though the install on windows seems to merge existing installs, you'll likely still run into issues with the Node Package Manager (NPM). See Uninstall Existing Node Documentation below:

Once installed, run the following to install the latest version of NodeJS:

Useful NVS Commands

List Versions currently installed

```
"engines" : { "node" : ">=0.12" }
```

```
list
```

Install version

```
nvm install "latest"
or
nvm install 10.1
```

Set current version

```
nvm use 10.1
```

Set NPM Versions

You can change your NPM Versions up or down using the following

Getting Lastest NPM

Node comes with npm installed so you should have a version of npm. However, npm gets updated more frequently than Node does, so you'll want to make sure it's the latest version.

```
npm install npm@latest -g
```

Targeting Specific NPM Version

Just replace @latest with the version number you want to downgrade to. I wanted to downgrade to version 3.10.10, so I used this command:

```
npm install -g npm@3.10.10
```

Note: If you do a lot of version switching, you might run into the following issue:

Error: Node Sass does not yet support your current orted please see: environment: Windows 64-bit with Unsupported runtim1.0e

There's an easy fix:

```
npm rebuild node-sass
```

Additional information can be found [here](). This approach requires python to be installed.

Uninstall Existing Node and NPM Versions

Uninstall existing Node Version

Please note, you need to uninstall any existing versions of node.js before installing NVM for Windows. Also delete any existing nodejs installation directories (e.g., "C:\Program Files\nodejs") that might remain. NVM's generated symlink will not overwrite an existing (even empty) installation directory.

Uninstall existing NPM Version

You should also delete the existing npm install location (e.g. "C:\Users<user>\AppData\Roaming\npm"), so that the nvm install location will be correctly used instead. Backup the global `npmrc` config (e.g. `C:\Users\<user>\AppData\Roaming\npm\etc\npmrc`), if you have some important settings there, or copy the settings to the user config `C:\Users\<user>\.npmrc`.

Helpful Tools

TOOL	DESCRIPTION	NOTES
NPM-Check	Display package versions and if updates are available.	Warning: Careful on making updates. Those are sweeping changes that shouldn't be taken lightly (unless of course you're not installing into the solution or changing dependencies) .

ESLint Issues

Sometimes ESLint causes a lot of issues. This often happens on a fresh clone that came from a repo built by a different OS. This issue can be easily resolved by running:

```
npm run lint -- --fix
```

Overview

npm (Node Package Manager) is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry. The registry is accessed via the client, and the available packages can be browsed and searched via the npm website. The package manager and the registry are managed by npm, Inc.

Command Line Client

npm includes a **CLI** (Command Line Client) that can be used to download and install software:

Windows Example

```
C:\>npm install <package>
```

Mac OS Example

```
>npm install <package>
```

Installing npm

npm is installed with Node.js This means that you have to install Node.js to get npm installed on your computer. Download Node.js from the official Node.js web site: https://nodejs.org

Overview

A light-weight editor that can run on-demand SQL queries, view and save results as text, JSON, or Excel. Edit data, organize your favorite database connections, and browse database objects in a familiar object browsing experience.

Download

Azure Data Studio can be downloaded from here.

Overview

Upload, download, and manage Azure blobs, files, queues, and tables, as well as Azure Cosmos DB and Azure Data Lake Storage entities. Easily access virtual machine disks, and work with either Azure Resource Manager or classic storage accounts. Manage and configure cross-origin resource sharing rules.

Download

Storage Explorer can be downloaded from here

Introduction

The Azure IoT explorer is a graphical tool for interacting with and testing your IoT Plug and Play Preview devices. After installing the tool on your local machine, you can use it to connect to a device. You can use the tool to view the telemetry the device is sending, work with device properties, and call commands.

Download

Azure IoT Explorer can be downloaded from here

Install and use Azure IoT explorer

Click here to know how to install and use Azure IoT explorer

Quick Demo

Check out this video for a quick e2e demo.

Installation guide for Docker

Docker is a set of platform as a service products that uses OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. To install it, following the guides below specific to your operating system.

Docker on Mac

Installation instructions for installing Docker on a Mac can be found here.

Docker on Windows

What to know before installation

- Windows 10 64-bit: Pro, Enterprise, or Education (Build 15063 or later).
- Hyper-V and Containers Windows features must be enabled.
- The following hardware prerequisites are required to successfully run Client Hyper-V on Windows 10:
    - 64-bit processor with Second Level Address Translation (SLAT)
    - 4GB system RAM
    - BIOS-level hardware virtualization support must be enabled in the BIOS settings. For more information, see Virtualization.

For additional information about requirements see: What to know before you install.

Steps to setup Docker on Windows

- Download Docker Desktop https://hub.docker.com/editions/community/docker-ce-desktop-windows/
- Double-click Docker Desktop Installer.exe to run the installer.
- If you haven't already downloaded the installer (Docker Desktop Installer.exe), you can get it from Docker Hub. It typically downloads to your Downloads folder, or you can run it from the recent downloads bar at the bottom of your web browser.
- Follow instructions on the installation wizard to accept the license, authorize the installer, and proceed with the install.
- When prompted, authorize the Docker Desktop Installer with your system password during the install process. Privileged access is needed to install networking components, links to the Docker apps, and manage the Hyper-V VMs.
- Click Finish on the setup complete dialog and launch the Docker Desktop application.

For additional information, see Install Docker Desktop on Windows. To test your installation, follow this guide.

Alternative Setup

It might now be possible to use use Docker on Windows without having to virtualize thanks to the release of the Linux subsystem for Windows. This approach has not been tested to determine if it will work for this solution. It allows you to run Unbuntu and openSUSE on Windows, with Fedora and more Linux distributions coming soon.

As untried alternative to virtualization, if you're using Windows you could try the Linux subsystem for Windows. Note: We have not tried this. Key consideration is whether or not docker images can be instantiated.

This documented might be a good guide: https://docs.microsoft.com/en-us/virtualization/windowscontainers/quick-start/set-up-environment?tabs=Windows-10-Client

For additional information on the Linux Subsystem: The Windows Subsystem for Linux, introduced in the Anniversary Update, became a stable feature in the Fall Creators Update.

Installation guide for Terraform

Steps to set up terraform on Windows:

- To install Terraform, find the appropriate package for your system and download it. Terraform is packaged as a zip archive.
- After downloading Terraform, unzip the package. Terraform runs as a single binary named terraform. Any other files in the package can be safely removed and Terraform will still function.
- The final step is to make sure that the terraform binary is available on the PATH.
- Verify the installation by executing terraform on new terminal session

Installation guide for Kubernetes

Steps to set up a kubectl on Windows

- Download the kubectl.exe using a link and save the file in any folder on windows file system.
- Add the kubectl.exe folder location in path variable - "Advanced System Settings -> Advanced -> Environment Variables -> Path". For example, if you have saved file to C:/kube then add this folder path to the path variable.
- Open a command prompt and type kubectl and you should see all commands supported by kubectl.

Install minikube *(Not Mandatory)

- VT-x or AMD-v virtualization must be enabled in your computer's BIOS.
- Install the virtualization platform such as Virtualbox or KVM. You are not really required to configure the image.
- Download the minikube-windows-amd64 file from here.
- Add this folder path location in path variable: "Advanced System Settings -> Advanced -> Environment Variables -> Path." For example, if you have saved the file to C:/kube then add this folder path to the path variable.
- Open the command prompt and fire a command minikube and you should see all the commands supported by minikube.

On windows, you can get similar kind of linux kind of user experience with Cygwin. Install Cygwin by following the steps listed on its website.

Start minikube :

- Open the Cygwin terminal and run command $ minikube start.
- Run command kubectl version to confirm the working of minikube.

Installation guide for Helm

Prerequisites

- You will need the command line program kubectl installed on your Windows 10 computer and configured to work with a Kubernetes Cluster.
- 7-Zip compression / decompression program is needed to extract the Helm program for Windows from the compressed file-folder from the Helm site. You can download it here: https://www.7-zip.org/download.html

Steps to Install

- Download the latest version of the compressed executable from the Helm GitHub site, https://github.com/kubernetes/helm/releases.
- Navigate to the folder you downloaded the helm-vX.X.X-windows-amd64.tar.gz compressed file from and move the file to its own directory.
- Navigate to the new directory and right click on the tar.gz file and with 7Zip, open the tar.gz archive.
- Double click the single tar file in that directory, helm-v2.7.2-windows-amd64.tar.
- You should now see a windows file folder in the 7Zip window, windows-amd64. Right click on the folder, select Copy To, and select the directory you want to copy the folder to.
- add the helm program to the System File path to make it easily accessible from the command line.
- Open the Control Panel's System panel.
- Select the Advanced system settings link on the left.
- Select Environment Variables.
- Under System variables, select Path, and then select Edit.
- Select the New button and then add the folder path where you copied the helm folder to and then press OK.
- Open a new command line window and type helm on the command line to make sure you have access to helm from the command line.
- Assuming you have the kubectl program configured for your Kubernetes cluster you can now initialize helm.
- Now you are ready to deploy Kubernetes applications to your kube cluster.

Note: Ubuntu and an evaluation copy of Windows. Otherwise you need an ISO and License Key for the OS you intend to install. Using the eval requires a 16GB download.

Overview

Redux is an open-source JavaScript library for managing application state. It is most commonly used with libraries such as React or Angular for building user interfaces. It was created by Dan Abramov and Andrew Clark.

Redux is a predictable state container for JavaScript apps.

It helps you write applications that behave consistently, run in different environments (client, server, and native), and are easy to test. On top of that, it provides a great developer experience, such as live code editing combined with a time traveling debugger.

You can use Redux together with React, or with any other view library. It is tiny (2kB, including dependencies), but has a large ecosystem of addons available.

Installation

To install the stable version:

npm install redux This assumes you are using npm as your package manager.

If you're not, you can access these files on unpkg, download them, or point your package manager to them.

Click here to get more details

Introduction

Postman is a collaboration platform for API development. Postman's features simplify each step of building an API and streamline collaboration so you can create better APIs—faster.

Get more detail here

Installation and updates

Postman is available as a native app for Mac, Windows (32-bit / 64-bit), and Linux (32-bit / 64-bit) operating systems.

To get the latest version of the Postman app, visit the download page and click Download for your platform.

Installing Postman

- Mac
- Windows
- Linux

Installing and Configuring 3M's Azure IoT Source Code

Assuming you've completed the major installs required to run the source code, you're now ready to clone the 3M repo and configure the solution on your local development environment.

To do so, requires the following steps:

1. Clone Repo
2. Configure Environment Settings
3. Update Source Code Dependencies
4. Build and Run

You might also want to watch the setup video: https://www.youtube.com/watch?v=asICCW_1uiI

Grant KeyVault Access to Project

dotnet user-secrets set --project C:{your path}\src\services\common\Services\Services.csproj AppConfigurationConnectionString "Endpoint=https://crsliot-appconfig-dev.azconfig.io;Id=AMGf-l4-s0:rAtTcp3u1hOQszVB49Tg;Secret=R4VYRXHcDx8FqQxc9xNC4F16Mxt0yG4FReBFVxQF7Is="

TODO

Organize content into the sections below and provide coverage of how/when to use: Icons in React JS Fluent Framework: https://developer.microsoft.com/en-us/fluentui#/styles/web/icons

<crsliot-aks-dev.centralus.cloudapp.azure.com>

Web UI:

dotnet user-secrets set --project C:{your path}\src\services\common\Services\Services.csproj AppConfigurationConnectionString "Endpoint=https://crsliot-appconfig-dev.azconfig.io;Id=AMGf-l4-s0:rAtTcp3u1hOQszVB49Tg;Secret=R4VYRXHcDx8FqQxc9xNC4F16Mxt0yG4FReBFVxQF7Is="

```
.env NODE_PATH src/
```

App config:

const baseUrl = "https://crsliot-aks-dev.centralus.cloudapp.azure.com";//process.env.REACT_APP_BASE_SERVICE_URL || '';

dotnet user-secrets set --project C:\DevOps\OpenSource\azure-iot-platform-dotnet\src\services\common\Services\Services.csproj AppConfigurationConnectionString "Endpoint=https://crsliot-appconfig-dev.azconfig.io;Id=AMGf-l4-s0:rAtTcp3u1hOQszVB49Tg;Secret=R4VYRXHcDx8FqQxc9xNC4F16Mxt0yG4FReBFVxQF7Is="

dotnet build mmm.iot.sln C:\DevOps\OpenSource\azure-iot-platform-dotnet\webui

https://github.mmm.com/mmm/azure-iot-services-dotnet

```
npm install

npm start
```

Clone Repo

There are many ways to clone the 3M Repo. Doing so depends on having Git installed.

- Internal Repo: https://github.mmm.com/mmm/azure-iot-services-dotnet
- Public Repo: https://github.com/3M-Company/azure-iot-platform-dotnet/

Configure Environment Settings

Use the relevant sections below to configure environment Settings

Follow recommendations here: https://github.com/3M-Company/azure-iot-platform-dotnet/blob/master/docs/DEVELOPMENT.md More information on configuring environment variables here.

WebUI Environment Variables Use the section below that matches your environment.

- Windows
- MAC
- LINUX

Docker and Kubernetes Settings

WebUI Development Settings

It's necessary to set a few environment settings to get things up and running. Use the section below that matches your environment:

Update Source Code Dependencies

Build and Run

If you have SASS issues run:

```
npm run lint -- --fix
```

Check-in Changes

TODO: https://designmodo.com/react-ci-cd/

Services Development Settings

TODO: 3MC02YM21KJG5J:webui a9q25zz$ export REACT_APP_BASE_SERVICE_URL="https://crsliot-aks-dev.centralus.cloudapp.azure.com/"

References

Azure Iot UX Fluent Controls	https://www.microsoft.com/design/fluent/	https://github.com/Azure/iot-ux-fluent-controls

Library version updates in 3M source code

Web UI section

update "node-sass" version to "4.13.1"

To install, use:

```
npm install node-sass@4.13.1
```

Guide

A PowerShell script for configuring DevSpaces is available here. Works on Windows and macOS. Not sure about Linux. But we also have a bash version (under Files\Tools\DevSpaces). Additional documentation is here.

Here are instructions on how to use:

To use this file, download DevSpaces.ps1 to your machine
Then edit DevSpaces.ps1 line 10 and change the $script:MmmSourceDirectory variable to point to the containing folder of the Serenity repository
Then edit your PowerShell profile and add a line to "dot-source" DevSpaces.ps1 like so:

```
. 'C:\Users\aa30hzz\DevSpaces.ps1'
```

Then, restart your PowerShell Core terminal
Then, create a new DevSpace like so (creates the default/kyle DevSpace):

```
New-DevSpace -Name kyle -Parent default
```

Then, deploy Serenity to the DevSpace like so:

```
Start-SerenityDevSpace -Name kyle

Use Stop-SerenityDevSpace and Remove-DevSpace to stop the DevSpace and remove the DevSpace.
```

Scripts

NAME	WINDOWS	MACOS	LINUX
Configure Dev Spaces	DevSpaces.ps1	DevSpaces.ps1	new_dev_space.sh

Install Guide

You can download the tool from here: https://xunit.net/
XUnit is a unit testing tool for the .NET Framework

3M IoT Platform Guides

The following guides offer insight into how to do day-to-day operations on the platform.

Devices

TODO: Describe this section.

Telementry

TODO: Describe this section.

Overview of IoT Devices on Azure

Conceptually, there are two types of devices that can be used in Azure: 1) Simulated and 2) Real.
Simulated devices are used to test connectivity and offer a quick way to evaluate functionality without needing to procure any hardware.
There are two types of real devices: IoT Edge device and IoT device.

If you'd like to learn more about the types of devices that can work with Azure check out Microsoft's Device Catalog

Device Registration

Before you can use a real or simulated device, you must first register it. Use the following links to get started:

- Register a device
- Register a simulated device

How to Register a Device

For a device to connect to the solution accelerator, it must identify itself to IoT Hub using valid credentials. You have the opportunity to save the device connection string that contains these credentials when you add the device to the solution. You include the device connection string in your client application later in this tutorial.

To add a device to your Remote Monitoring solution, complete the following steps on the Device Explorer page in the solution:

1. Choose + **New device**, and then choose **Real** as the **Device type**:
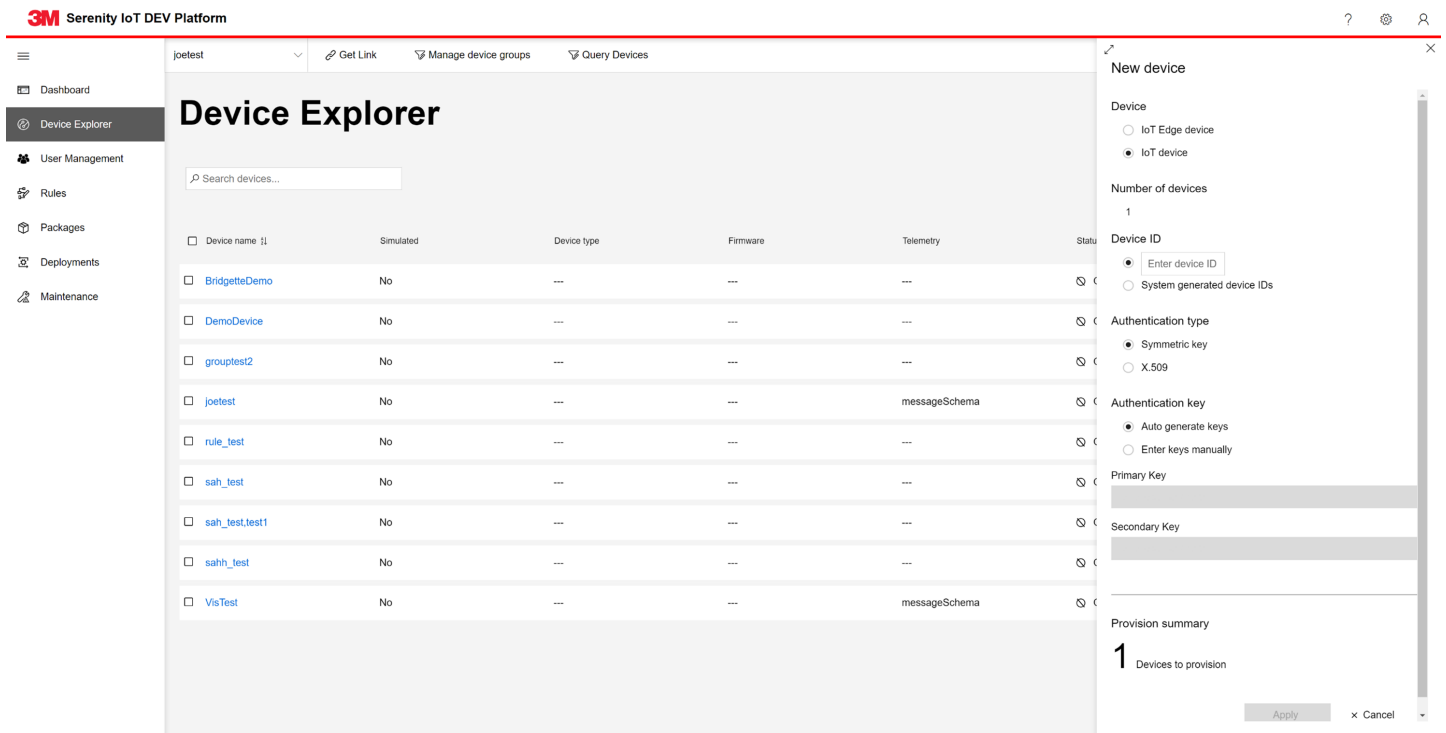
1. Enter a Device ID you wish to use to reference this device or allow the system to generate one for you. The value must be unique and connect include spaces.

2. Select the desired authentication type and either have the system generate authentication keys for you or provide your own. CRSL generally uses a Symmetric Key but your organization might prefer x509. Chose whichever is the most appropriate.

3. Select the preferred option for how to define keys. You have the choice between:

   - Providing your own
   - Auto Generating

How to Register a Simulated Device

Simulating Device Data using Azure Online Simulator

This documentation explains how to simulate device telemetry using Azure Online Simulator i.e., Raspberry Pi Simulator. Note: The example used is to explain how to simulate Device data using connection string of a device. The step#1(Creating a device) can be skipped if the device is already present and connection string is available. Step-1: Creating a Device Create a device and fetch the connection string. Ex: (Using web application)

1. Login to CRSL or PSD portal
2. Go to Device Explorer and click New Device

3. Give the inputs for creating a new device a. Device type: IoT Device/IoT Edge Device b. Device ID: Custom DeviceID or System generated DeviceID c. Authentication type: Symmetric key/ X.509 d. Authentication keys: Autogenerated/Enter manually (Primary and Secondary) e. Click on create/apply

4. Once the device is connected, copy the connection strings (Sample connection string: HostName=iothub-233a1ca2.azure-devices.net;DeviceId=9db19edd-ba26-4e56-80f8-63582f0d1f8c;SharedAccessKey=bjRJEOgHkRxfefNnO4AfakB7No7bzlC3yqeHVlu/F2E=) Note: For creating a device in IoT hub, refer this: https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-raspberry-pi-web-simulator-get-started#register-a-new-device-in-the-iot-hub Step-2: Simulate Device telemetry

5. Launch the Raspberry Pi online Simulator using: https://azure-samples.github.io/raspberry-pi-web-simulator/#getstarted
6. The simulator has three areas

a. Assembly area: Preview of the default circuit b. Coding area: An online code editor where the application to collect and send data in the simulator resides. Note: A default sample application is present in this area which helps to collect sensor data from BME280 sensor and sends to your Azure IoT Hub. This is fully compatible with real Pi devices c. Integrated Console window: Shows the output of your code. And it has three buttons i. Run: Run the application from the coding area ii. Reset: Resets the coding area to the default application by undoing the changes you made iii. Fold/Expand: Used to fold/expand the console window

1. In the coding area, if you are running the default application, scroll to the connectionString constant assignment(line.15) and replace the default placeholder with your device connection string from Step-1.

2. Run code: Select Run from console window or type npm start to run the application

3. Output: You should see the following output that shows the sensor data and the messages that are sent to your IoT hub

Step-3: Verify telemetry data

Telemetry data is sent and can be verified either through Azure portal or web application

References: • https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-raspberry-pi-web-simulator-get-started • https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-vscode-iot-toolkit-cloud-device-messaging