Industrial Machine Connectivity (IMC) Kit

Physical-Brownfield Deployment User Guide

**Introduction:**

The physical brownfield deployment is intended to demonstrate the capabilities of the IMC kit in an environment where the end user has an existing edge-based asset modeling software (such as Ignition or KepServer). It is deployed onto physical hardware. After deployment, the physical hardware will run Greengrass software and connect into the edge-based asset modeling software. The IMC kit supports the following OEM devices:

1. **Lenovo**
   1. Model: ThinkCentre M90n IoT
   2. Architecture: Intel® Celeron® 4205U (x86)
   3. URL: <https://www.lenovo.com/us/en/desktops-and-all-in-ones/thinkcentre/m-series-tiny/ThinkCentre-M90n-IoT/p/thinkcentre-m90n-iot>
2. **ADLINK** 
   1. Model: MXE-211
   2. Architecture: Intel Atom® Processor E3900 (x86)
   3. <https://www.adlinktech.com/Products/Industrial_IoT_and_Cloud_solutions/IoTGateway/MXE-210_Series?lang=en>
3. **OnLogic**
   1. Model: Karbon 300 Compact Rugged Computer
   2. Architecture: Intel Atom® E3930 or E3950 processors
   3. URL: <https://onlogic.com/k300/>
4. **Advantech** 
   1. Model: UNO-2372G
   2. Architecture: Intel Atom E3845/Celeron® J1900 Quad-Core Processors
   3. URL: <https://www.advantech.com/products/1-2mlj9a/uno-2372g/mod_f4ff5680-f016-44bd-bff0-e5eddfd82237>
5. **MOXA**
   1. Model: MC-1112-E4-T
   2. Architecture: Intel Atom® Processor E3845 processor
   3. URL: <https://www.moxa.com/en/products/industrial-computing/x86-computers/mc-1100-series/mc-1121-e4-t>

This deployment mode does not come with a configured set of project tags similar to the virtual deployment but does come with a set of device simulations that can be configured to represent a project tag structure similar to the virtual deployment tag structure (or your own structure entirely). This deployment can be configured to work with a physical PLC test harness.

This deployment mode is compatible with either Inductive Automation’s Ignition Server (<https://inductiveautomation.com/ignition/>) or PTCs KEPServerEX (<https://www.kepware.com/en-us/products/kepserverex/>). This deployment option does not bootstrap any partner edge software. The only edge software application that is bootstrapped on the physical hardware as part of the deployment is AWS IoT Greengrass.

**Deployment Options:**

There are several options for the virtual deployment configuration which depend on 2 configuration choices:

1. Data Flow option - There are 3 options for data flow:
   1. Option 1
   2. Option 2a
   3. Option 2b
2. Tag Hierarchy Source
   1. Cirrus Link Module - This option is based on the Cirrus Link MQTT Transmission Module implementing the SparkplugB MQTT spec to publish birth certificates when a node or device is created within the Ignition Designer UI dashboard. The AMC runs automatically without a manual step with this option and results in assets provisioned within AWS IoT SiteWise.
   2. Ignition File Export - This option is based on exporting a JSON tag hierarchy definition file from Ignition Server and uploading it into an S3 bucket. Once the file is uploaded to the S3 bucket, the AMC runs automatically, resulting in assets provisioned within AWS IoT SiteWise.
   3. Kepware File Export - This option is based on exporting a CSV tag hierarchy definition file from KEPServerEX and uploading it into an S3 bucket. Once the file is uploaded to the S3 bucket, the AMC runs automatically, resulting in assets provisioned within AWS IoT SiteWise.

Depending on the choices of these 2 configuration options, there are a total of 6 deployment modes:

## Physical Brownfield Option 1 Ignition File Export

## Physical Brownfield Option 1 KEPServerEX File Export

**Get Started**

1. Ensure that you are preparing pre-requisite resources and launching the CloudFormation stack in one of the 3 supported regions for the IMC kit:
   1. us-east-1
   2. us-west-2
   3. eu-west-1
2. Complete the setup steps defined in the Pre-Requisites section below.
3. Proceed to the deployment steps for the mode you selected in the list above. Each deployment mode has the following sections:
   1. CloudFormation stack launch
   2. Post deployment steps
   3. Troubleshooting - Refer to this section to troubleshoot IMC stack launch and operation.
4. Physical Brownfield Cleanup - Refer to this section to cleanup AWS resources launched for the IMC kit.
5. Physical Brownfield FAQ - Refer to this section for FAQs regarding IMC kit.

The virtual deployment has the following sections:

1. Pre-Requisites
2. CloudFormation stack launch
3. Post deployment steps
4. Troubleshooting
5. Physical Brownfield Cleanup
6. Physical Brownfield FAQ

## Pre-Requisites

* **AWS account with SSO enabled & User created:**
  + **Enable AWS SSO**
    - Navigate to the SSO service in the AWS console
    - A screenshot of a cell phone

      Description automatically generatedClick “Enable AWS SSO”
    - A screenshot of a cell phone

      Description automatically generatedIf you don’t have an AWS organization set up for your account (required for AWS SSO usage), you’ll be prompted to set one up. Click “Create AWS organization”.
    - For extra documentation, visit AWS:
      * <https://docs.aws.amazon.com/singlesignon/latest/userguide/getting-started.html>
  + **Create a Group:**
    - Navigate to the SSO service in the AWS console
    - Click on “Groups” in the left-hand navbar
    - A screenshot of a cell phone

      Description automatically generatedIf none exist, click the blue “Create group” button
    - A screenshot of a cell phone

      Description automatically generatedFill out the required fields, then click “Create”
  + **Create an SSO user:**
    - Navigate to the SSO service in the AWS console
    - Click on “Users” in the left-hand navbar
    - A screenshot of a cell phone

      Description automatically generatedClick the blue “Add user” button
    - A screenshot of a cell phone

      Description automatically generatedFill out the required fields
    - Click “Next: Groups”
    - A screenshot of a cell phone

      Description automatically generatedSelect a group, then click the blue “Add user” button
* **EC2 SSH Key Pair:** If you do not already have an EC2 SSH Key Pair available (pem file format), create one in the region you are launching the CloudFormation stack,

EC2 SSH Key Pair Documentation:<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html>

* **Create IoT SiteWise Service-linked Role Using the AWS CLI:** 
  + Run the following command from the AWS CLI:
    - aws iam create-service-linked-role --aws-service-name iotsitewise.amazonaws.com --description "Service-linked role to support IoT SiteWise"
  + Service-Linked-Role Documentation: <https://docs.aws.amazon.com/iot-sitewise/latest/userguide/using-service-linked-roles.html>
* **Connect to your physical hardware running Ubuntu 18.04, make sure it is connected to the internet, and configure the AWS CLI:**
  + You may achieve this however you please – either via an SSH command or a connection to the device with a keyboard and monitor
  + Ensure you have the ability to connect to the internet
  + Configure the AWS CLI on the hardware to communicate with the AWS account you plan to deploy in: <https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-configure.html>
* **The directory structure of the physical hardware running Ubuntu 18.04 must look like the following:**

home/

ubuntu

* **QuickStart Bucket Preparation:**
  + There are 2 methods of launching the CloudFormation stack using an S3 URL for the template. The recommend approach is option 1, which uses the AWS Quick Start S3 bucket as the source of the CloudFormation artifacts:
    - **Use Existing AWS Quick Start S3 Bucket:** Use the AWS Quick Start S3 bucket and the S3 URLs for the Master or Workload templates. The templates and assets are ready to use to launch a CloudFormation stack.
    - **Manual S3 Bucket Creation:**
      * Download the public IMC Kit AWS Quick Start Github repo as a zip file.
      * Create an S3 bucket and give it a unique name such as “imc-quickstart-bucket-ABC-123”.
      * In that S3 bucket, create a folder called “quickstart-IMC”.
      * Unzip the downloaded file and copy all the contents of the unzipped folder (Github repo contents) into the “quickstart-IMC” folder. The structure will then resemble the structure below:
        + S3 bucket name: “imc-quickstart-bucket-ABC-123”
        + S3 bucket content:

quickstart-IMC/

functions/

scripts/

submodules/

templates/

LICENSE.txt

NOTICE.txt

README.md

Deployment Modes:

## Physical Brownfield Option 1 Ignition File Export

## Physical Brownfield Option 1 KEPServerEX File Export

## Physical Brownfield Option 1 Ignition/KepServer File Export

Open up the AWS Console and navigate to the CloudFormation console. Create a new stack and select “with new resources (standard)”).

### CloudFormation stack launch

**Step 1: Specify templates**

* Pre-Requisite:
  1. Select “Template is ready”
* Specify Template (Choose one of the below):
  1. Template Source: Leave as default – “Amazon S3 URL”. Most users will want to use the “IMC-master.template.yaml” CloudFormation master template that’s in the S3 bucket we just created (i.e. “imc-quickstart-bucket-ABC-123”). This template launches the solution in a new VPC. Note, only 5 VPCs per account are permitted.
     1. The URL should look like this: https://<BUCKETNAME>.s3.amazonaws.com/quickstart-IMC/templates/IMC-master.template.yaml
  2. Alternatively, launch the “IMC-workload.template.yaml” Cloudformation template. The workload template launches the solution in your default VPC.
     1. The URL should look like this: https://<BUCKETNAME>.s3.amazonaws.com/quickstart-IMC/templates/IMC-workload.template.yaml
* Click Next to proceed to Step 2 of the CloudFormation stack launch (Specify stack details)

**Step 2 – Specify Stack Details (if using master template)**

* Stack Name:
  + Stack Name: Give the stack a unique name such as “IMC-Phys” . Make sure the name is no longer than 10 characters. If you don’t adhere to this character limit, your stack deployment may fail due to resource name lengths.
* Parameters
  + Network Configuration
    - VPC CIDR: Defaults to 10.0.0.0/16. Change if desired
    - Public subnet 1 CIDR: Defaults to 10.0.128.0/20. Change if desired
    - VPC Tenancy: Defaults to non-dedicated. Change if desired
  + AWS Quick Start Configuration
    - QuickStart S3 Bucket Name: Use the name of the bucket you created previously in the Pre-Requisites section. We used the example S3 bucket name of “imc-quickstart-bucket-ABC-123”
    - QuickStart S3 Key Prefix: (default: “quickstart-IMC/”) Use the name of the root folder in the S3 bucket you created. In the Pre-Requisites section we named the folder “quickstart-IMC/”
    - QuickStart S3 Bucket Region: Leave as default “us-east-1”
  + Edge Deployment Configuration
    - Name for the edge device**:** You may leave as default or, if desired, you may specify a new name for the edge device. This name will be the name of the Greengrass group that gets created with this stack.
    - Type of Deployment (Virtual or Physical):Physical
    - Deployment Flow: Select “Option 1”.
  + Amazon EC2 Configuration
    - SSH Key Name: For physical deployments, the EC2 Key Pair won’t be used, but one needs to be selected for the stack to deploy properly. If you haven’t, navigate to the EC2 service in the AWS console and create an EC2 key pair. Select the key pair when deploying the stack.
    - Greengrass EC2 Instance Type**:** (default: t3.small) For physical deployments, an EC2 instance won’t be created, but this field must be filled in during the creation of the stack. Leave as the default value.
    - Ignition EC2 Instance Types: (default: t3.large) For physical deployments, an EC2 instance won’t be created, but this field must be filled in during the creation of the stack. Leave as the default value.
    - Select the Asset Model Converter (AMC) Driver: Leave as default “IgnitionCirrusLink”.
    - User Public IP Address: For physical deployments, an EC2 instance won’t be created, but this field must be filled in during the creation of the stack. Input any IP address in the format “x.x.x.x”.

**Step 2: Specify stack details (if using workload template)**

* Stack Name:
  + Stack Name: Give the stack a unique name such as “IMC-Phys” . Make sure the name is no longer than 10 characters. If you don’t adhere to this character limit, your stack deployment may fail due to resource name lengths.
* Parameters:
  + Edge Deployment Configuration
    - Name for the edge device**:** You may leave as default or, if desired, you may specify a new name for the edge device. This name will be the name of the Greengrass group that gets created with this stack.
    - Type of Deployment (Virtual or Physical):Physical
    - Deployment Flow: Select “Option 1”
  + Amazon EC2 Configuration
    - SSH Key Name: For physical deployments, the EC2 Key Pair won’t be used, but one needs to be selected for the stack to deploy properly. If you haven’t, navigate to the EC2 service in the AWS console and create an EC2 key pair. Select the key pair when deploying the stack.
    - VPC ID: Find your default VPC ID and copy it into this field.
    - Greengrass EC2 Instance Type**:** (default: t3.small) For physical deployments, an EC2 instance won’t be created, but this field must be filled in during the creation of the stack. Leave as the default value.
    - Ignition EC2 Instance Types: (default: t3.large) For physical deployments, an EC2 instance won’t be created, but this field must be filled in during the creation of the stack. Leave as the default value.
    - EC2 Subnet: Find the VPC Subnet associated with availability zone [your\_region\_here]a in your account and use that value in this field.
  + AWS Quick Start Configuration
    - QuickStart S3 Bucket Name: Use the name of the bucket you created previously in the Pre-Requisites section. We used the example S3 bucket name of “imc-quickstart-bucket-ABC-123”
    - QuickStart S3 Key Prefix: Use the name of the root folder in the S3 bucket you created. In the Pre-Requisites section, we named the folder “quickstart-IMC/”
    - QuickStart S3 Bucket Region: Leave as default “us-east-1”
    - Select the Asset Model Converter (AMC) Driver: Select “file\_export\_type]”.
    - User Public IP Address: For physical deployments, an EC2 instance won’t be created, but this field must be filled in during the creation of the stack. Input any IP address in the format “x.x.x.x”.
  + Click “Next” to proceed to Step 3 “Configure stack options”

**Step 3: Configure stack options**

* You can accept all defaults
* Click “Next” to proceed to Step 4 “Review”

**Step 4: Review**

* Review and accept the acknowledgements at the bottom of the page
* Click “Create stack” to launch the CloudFormation stack.

**Stack Deployment**

Stack deployment will take approximately 5-10 minutes. You can track the progress of the stack launch by viewing the “Events” tab of the stack.

### Post Deployment Steps

Once the CloudFormation stack is completed, follow the steps to configure the IMC Kit to make it operational.

**Option 1**

1. Retrieve and run the bootup script for the physical hardware device
   1. Open a terminal on the physical hardware
   2. Use the command line to become the root user in your terminal session:
      1. sudo su
   3. Use the command line to retrieve the deployment script from your stack’s S3 bucket, replacing the values in brackets with the corresponding output values from your CloudFormation stack (found on the AWS CloudFormation console, in the stack labeled “NESTED” in the output tab):
      1. aws s3api get-object --bucket [DependenciesBucket] --key [BootupScriptBrownfieldAllOptions] physical-brownfield-all-options.sh
      2. \*\*You must have the AWS CLI configured to point to the AWS account you’re using for the IMC kit.
   4. Use the command line to make the file executable:
      1. chmod +x physical-greenfield-all-options.sh
   5. Retrieve (copy to your clipboard) the command from the “NESTED” CloudFormation output: [FullScriptParamsBrownField]
   6. Use the command line to run the deployment script, which should resemble something like the following (but filled in with your stack-specific values):
      1. ./physical-brownfield-all-options.sh physicalbrownfieldoption1-devicesbucketresource-1ifk8w3a77621 PhysicalBrownfieldOption1/PhysicalBrownfieldOption1Core.tar.gz 866prvj43g us-east-1 PhysicalBrownfieldOption1
2. Ensure that your instance of edge-based asset modeling software is set up with the desired hierarchy of assets.
3. Export the JSON file that describes your project hierarchy in either Ignition or KepServer.
4. Upload the JSON file from step 3 into the following S3 bucket created during deployment to trigger the AMC and creation of models and assets in SiteWise:
   1. [name\_of\_stack]-[amcincomingresource]-[hash]
      1. Upon dropping the JSON file in this S3 bucket, an S3 event trigger will automatically invoke the AMC Lambda function
   2. Wait for the AMC to complete creating your models and assets in SiteWise
5. Update the SiteWise Gateway
   1. Navigate to AWS IoT SiteWise console and select Ingest🡪Gateways
   2. Select the gateway created during the stack launch.
      1. Naming convention: [name\_of\_stack]\_Automated\_Gateway
   3. Edit the "Source Configuration for Automated Gateway Config” section to point to the edge-based asset modeling software.
      1. Ignition format: opc.tcp://<reachable\_IP>:62541
      2. KepServer format: opc.tcp://<reachable\_IP>:49320
   4. Click “Save” at the bottom. No changes are necessary. This action simply activates the SiteWise gateway to ensure data flows from the OPC UA server.
6. Accept SiteWise Certificate in your edge-based asset modeling software: To enable the SiteWise to ingest data over OPC UA from the OPC UA server, you must accept the certificate presented by the SiteWise connector.
7. Validate Incoming PLC Data
   1. Now that you've trusted the certificate, go back to the AWS IoT SiteWise console.
   2. In the SiteWise console, click the icon on the left side of the page, select build🡪assets
   3. In the asset tree on the left, drill down to an asset (i.e. Hauloff or Conveyor), select it and then select “Measurements” tab for that asset.
   4. Verify that the values in the “Latest value” column are updating. This indicates that the Ignition simulation of those virtual devices and sensors is properly sending data through to the SiteWise connector in Greengrass and up to AWS IoT SiteWise in the AWS cloud.

### View SiteWise Portal Data

**Log in to SiteWise Monitor Portal**

1. For a more visual display of the data, navigate to the SiteWise console, select the icon on the left and select Monitor🡪 Portals.
2. Select the hyperlinked "name" of the Portal most recently added (the topmost on the list).
3. Add yourself as an administrator of the Portal by clicking “Assign Users” in the Portal Administrators section
4. Once you are listed as a Portal Administrator, click the hyperlinked URL in the Portal details section under the “URL” column. This URL should have the format <https://[XXXXX....XXXXXX].app.iotsitewise.aws>.
5. Log in with the credentials (username and password) you just created for your administrator account.

**View Data in SiteWise Monitor Portal**

1. Select “Dashboards” tab on the left-hand side, then select the newly created dashboard hyperlink under the “Name” column of the Dashboards page.
   1. Data should be flowing into the line charts for the asset measurement properties
2. You can also see data for individual assets by navigating to the “Asset Library” tab on the left and selecting an asset from the asset tree. Once an asset is selected, you can view its properties.

### Troubleshooting

**Models and assets weren’t created in SiteWise**

Check the Lambda function responsible for creating the models and assets in SiteWise for errors:

1. In the AWS lambda console, navigate to the function named [name\_of\_stack]-AssetModelIngestionLambdaResource-[hash]
2. Hit the “Monitoring” tab
3. Click “View logs in CloudWatch”
4. Click into the most recent Log Stream and find the error message

**Quarantined certificate in Ignition/KepServer doesn't show up, or data doesn’t show up for Option 1 deployments**

If using Ignition, verify that the Ignition trial period (2 hours) has not expired. If that action does not remediate the issue, repeat the process of refreshing the SiteWise Gateway:

1. Navigate to the AWS IoT SiteWise console and select Ingest 🡪 Gateways
2. Select the gateway created during the stack launch:
   1. Naming convention: [name\_of\_stack]\_Automated\_Gateway
3. Click “Edit” in the Source Configuration for Automated Gateway Config section
4. Click “Save” at the bottom. No changes are necessary. This action simply activates the SiteWise gateway to ensure data flows from the OPC UA server.

If it hasn’t already been done, look for and accept the quarantined certificate in Ignition

If using KepServer for Windows, make sure that your default firewalls have been turned off (they prevent the SiteWise Gateway certificates from showing up).

## Physical Brownfield Cleanup

**Follow these steps to clean up the IMC CloudFormation stack deployment:**

Cloud

1. Navigate to the CloudFormation console and delete the base stack (not the stack named "NESTED"), in order to clean up the account as much as possible. Most of the resources will be deleted, but the stack deletion will fail due to non-empty S3 buckets and potentially a deployed Greengrass group (for all Virtual options by default, and for all Physical deployments that have been completed on a piece of hardware. The steps required to delete a stack are:
   1. Empty the S3 buckets:
      1. Navigate to the S3 service in the AWS Console.
      2. In the search bar, enter your stack name.
      3. For each bucket that is associated with the stack (naming convention: [name\_of\_stack]-[bucket\_identifier]-unique hash), select the bucket, and click “Empty” under the search bar.
      4. Here are the following “bucket identifiers” that exist for each deployment:
         1. amcincomingresource
         2. amcoutputresource
         3. devicesbucketresource
         4. imcs3bucket
         5. lambdazipsbucket
   2. Force a reset of the GreenGrass group:
      1. Navigate to the GreenGrass console
      2. Select the GreenGrass group with the “Name for the Edge device” parameter name provided to the stack
      3. Under “Actions”, select “Reset Deployments”
      4. Check the box that asks if you want to force the reset
      5. Click “Reset Deployment”
   3. Navigate back to the CloudFormation console and once again delete the base stack.
      1. There is a master and nested stack. you must delete the master stack (the one that does **not** have “NESTED” in a gray box associated with it.
2. Other resources to clean up after stack deletion (if desired, for cleanliness): SiteWise Portal, SiteWise Gateway, SiteWise Models and Assets, QuickSight dataset.

Edge Hardware

1. Navigate to a terminal on the edge hardware. Become the root user using the “sudo su” command.
2. Stop and remove GreenGrass:
   1. apt remove aws-iot-greengrass-core
   2. rm -rf /greengrass
   3. rm -rf /var/sitewise

## Physical Brownfield FAQs

**Can I update a stack to a different deployment type (Physical, Virtual) or dataflow option (Option1, 2a, 2b)?**

* Updates are currently not supported. To achieve a different deployment type or dataflow type, you'll need to [deploy a new stack.](#_Physical_–_Greenfield) See the details on having more than one stack deployed in an account below, and then decide whether or not you’ll want to [delete your original stack](#_Physical_Greenfield_Cleanup_1) before re-creating another.

**Can I deploy multiple times in the same AWS account?**

Yes, you may deploy multiple stacks in the same account – but be wary of the following:

* For Option 1 deployments, SiteWise may receive data from multiple sources if there are, for example, two instances of Ignition that are publishing data onto the topic “/Tag Providers/default/Line1/CNC/Temperature”.
* If SiteWise models and assets exist in the Cloud from a previous deployment, a new deployment will not re-create the model/asset hierarchy. Instead, the Asset Model Converter operates by recognizing any “deltas” in the existing models/assets and the new deployment hierarchy. For example:
  + If the previously deployed hierarchy is identical to the newly deployed, nothing in SiteWise is changed.
  + If the newly deployed hierarchy has an identical structure to the previously deployed hierarchy **with additional nodes that follow the hierarchy pattern,** the new models/assets will be created in SiteWise**:**
    - Previous:
      * Line 1/CNC1/Temperature
    - New:
      * Line 1/CNC1/Temperature
      * Line 1/CNC2/Temperature
* If the newly deployed hierarchy has a different hierarchy than the previously deployed hierarchy altogether, the Asset Model Converter will not succeed in creating the new models and assets.
* Refer to the instructions in the appendix showing how to [add a line and device with tags](#_Add_a_line) to a project in Ignition using the Ignition Designer.

**Does any hardening of the OS occur during deployment?**

Yes. Fail2ban is installed and enabled on the device, which bans IPs that show the following malicious signs:

* Too many password failures
* Seeking for exploits

Additionally, shared memory on the device is secured via /etc/fstab.

**Can I re-run the AMC to create new models and assets?**

Yes. Follow these steps:

1. If any of the models/assets you want to create share a name with the models/assets created in the first pass of the AMC execution, you’ll need to delete the models and assets in SiteWise.
2. The second action you’ll need to take is clearing out the following DynamoDB tables associated with the initial IMC kit deployment:
   1. [stack-name]-asset-model-table
   2. [stack-name]-asset-table

Depending on your driver of choice, you’ll need to take the follow actions to re-run the AMC:

* IgnitionCirrusLink
  + Initiate a re-birth (NBIRTH,DBIRTH) MQTT message that represents the your project hierarchy.
    - Connect to your Ignition server in the Ignition Designer
    - A screenshot of a cell phone

      Description automatically generatedA birth message is triggered by launching the Ignition Designer, navigating to the tag browser 🡪 tag providers 🡪 MQTT Transmission 🡪 Transmission Control 🡪 clicking the “Refresh” button.
    - This action triggers the IMC kit’s AMC, which creates the models and assets that represent the Ignition hierarchy in SiteWise.
* IgnitionFileExport
  + Obtain your JSON file that represents the new hierarchy and drop it into the following S3 bucket:
    - [name\_of\_stack]-[amcincomingresource]-[hash]
* KepServerFileExport
  + Obtain your JSON file that represents the new hierarchy and drop it into the following S3 bucket:
    - [name\_of\_stack]-[amcincomingresource]-[hash]

# Asset Model Converter

The AMC code is in two separate lambda functions located in the functions/source directory of the repository.

## AssetModelIngestion

This is the ingestion lambda. It’s purpose is to listen to MQTT messages coming from the CirusLink Ignition module and put them into the incoing S3 bucket as objects. The relevant handler file is “assetModelIngestion.py”.

## AssetModelConverter

The AMC really covers two functions – AMC1, and AMC2.

### AMC1

This code is triggered via an S3 Object Creation trigger from the incoming bucket. It has it’s reserved concurrent executions set to 1, to ensure that only a single instance of the lambda can ever fire at once. This has been done because the Ignition CirrusLink module can birth the asset structure as multiple discrete MQTT messages, each of which become file objects in S3.

The core logic of the AMC1 comes in the form of the main handler file, ‘assetModelConverter.py’. It contains the base lambda handler function, which in turn calls into the AssetModelConverter class to execute the handling of a given S3 object creation event. This code in turn calls the relevant Driver code to handle the asset structure files.

#### Drivers

Drivers in AMC1 are responsible for parsing the incoming asset structure file(s), and converting it into a normalized format that is then stored into DynamoDB. Please see the ‘Creating AMC Drivers’ section of the Getting Started guide for more details on creating drivers, and the DynamoDB normalized format.

### AMC2

This relevant code file here is ‘createSitewiseResources.py’ This contains code to take the normalized DynamoDB format, and create relevant AWS IoT Sitewise assets and models from that format. Changes are marked in the DynamoDB table to various assets/models that require updates/creation.

Additionally, all AWS IoT Sitewise interactions take place in the ‘sitewiseUtils.py’ code.

Please note that only additive changes are supported at this time.

The AMC2 process comes in the form of the following steps:

#### **Model Creation**

The DynamoDB models table is queried to look for models that require creation. Those are created in AWS IoT Sitewise.

#### **Asset Creation**

The DynamoDB assets table is also queries, looking for records that are marked as needing creation. Those assets, instances of the models created during ‘Model Creation’ are then created in AWS IoT Sitewise.

#### **Model Hierarchies**

In this step Models are associated with each other in a hierarchy that are tagged as having such a parent/child relationship.

#### **Asset Associations**

Similarly in this final step, assets that are tagged as having a parent/child relationship are associated with each.

## Creating AMC Drivers

### Instructions

1. Write the driver that interprets the incoming hierarchy data from your edge-based asset modeling software and converts it into the AMC-approved format ([see the format here](#_AMC-Approved_DynamoDB_Format)) and puts it into DynamoDB
   1. Refer to the template file for guidance while writing your driver:
      1. /functions/source/AssetModelConverter/drivers/example\_driver\_template.py
   2. \*\* Highly recommended – also refer to the existing drivers:
      1. /functions/source/AssetModelConverter/drivers/igniitonCirrusLinkDriver.py
      2. /functions/source/AssetModelConverter/drivers/ignitionFileDriver.py
      3. /functions/source/AssetModelConverter/drivers/kepserver\_file\_driver.py
2. Edit the entry point file for the AMC (/functions/source/AssetModelConverter/assetModelConverter.py) to use your new driver:
   1. Import your driver
      1. From drivers.[name\_of\_file] import [name\_of\_driver\_class]
   2. Add your driver to the ‘driverTable’ list
      1. ‘[name\_of\_driver]’: [name\_of\_driver\_class]
3. Replace the AssetModelConverter zip file with its new contents:
   1. Zip up the contents of /functions/source/AssetModelConverter/
   2. Name the zip file above “AssetModelConverter.zip”
   3. Replace the old “AssetModelConverter.zip” file (/functions/packages/AssetModelConverter/AssetModelConverter.zip) with the new “AssetModelConverter.zip” file you created in ‘b’ above.
4. Edit the CloudFormation template to include your driver’s name:
   1. /templates/IMC-workload.template.yaml
      1. Add an item to the list of AMCDrivers (parameter section)
         1. - [name\_of\_driver\_here]

# Appendix

## Artifacts

**The following directories and files are necessary for running an IMC kit deployment:**

functions/

scripts/

templates/

LICENSE.txt

NOTICE.txt

README.md

**quickstart**-IMC: The root directory in the S3 bucket, where the rest of the folders live.

**functions**: Contains zipped lambda code that is used for various pieces of the IMC kit.

**scripts**: Contains the scripts that are run on physical hardware if running a physical deployment.

**templates**: Contains the various CloudFormation templates that will be deployed depending on the deployment options selected during stack creation.

## AMC-Approved DynamoDB Format

**Asset Model Table ([name-of-stack]-asset-model-table):**

assetModelEntry = {  
    “assetModelName”: type<string>, # Name of the asset model  
    “parent”: type<string>, # name of the parent asset model, if any  
    “assetModelProperties”: type<list<modelProperty>>, # list of sitewise assetModelProperties as ‘modelProperty’ listed below.  
    “assetModelHierarchies”: type<list>, # sitewise assetModelHierarchies, leave blank []  
    “change”: type<string>, # Should be ‘YES’, indicates in DynamoDB that the record is new or updated.  
}  
    modelProperty = {  
        ‘name’: type<string>, # Name of the property  
        ‘dataType’: type<string>, # Sitewise data type of the property  
        ‘type’: {  
            ‘measurement’: {} # Don’t change this or populate it with anything, used to identify property type in sitewise  
        }  
    }

**Asset Table ([name-of-stack]-asset-table):**

assetEntry = {  
    ‘assetName’: type<string>, # name of the asset  
    ‘modelName’: type<string>, # model name this asset is an instance of  
    ‘change’: type<string>, # Should be ‘YES’, indicates in DynamoDB that the record is new or updated.  
    ‘tags’: type<list<tagEntry>>, # List of tagEntry struct, as specified below  
}    tagEntry = {  
        ‘tagName’: type<string>, # name of the tag  
        ‘tagPath’: type<string>, # Full property alias path for the tag  
    }

## Add a line and device to an Ignition project

* Navigate to the Ignition Designer and connect to your Ignition server.
  + Launch the Designer
  + Click “Add Designer”
  + Click “Manually Add Gateway”
  + Add a Gateway URL in the following format: <http://[ignition_ec2_public_ip]:8088>
  + Under the Gateway tile you just added, click “Launch”
  + Supply the username and password and click “Login”
    - Username: admin
    - Password: password
      * **If you haven’t already, it is recommended that you change your password once you’ve successfully logged into the Ignition web UI.**
* Create a Data Type
  + Nagivate to the Tag Browser, expand “Tags”, right click “Data Types” 🡪 New Tag 🡪 New Data Type
  + Under Properties, name the Data Type “Pump”, and click “Apply”
* Configure the Tags for the Data Type
  + Tag 1: Temperature
    - A screenshot of a cell phone

      Description automatically generatedTo the left of the “Properties” section, click the “Add Tag” button, and select OPC Tag:
    - Edit the “Basic Properties”
      * Name the tag “Temperature”
      * Change its Data Type to float
      * Click the Link icon to the right of “OPC Server”, click “Browse OPC”, right click on Ignition OPC UA Server, and click “Copy Item Path”, then “Commit”. Right click in the space to the right of OPC Server and paste what’s copied in your clipboard.
      * Click the Link icon to the right of OPC Item Path, click “Browse OPC”, expand Ignition OPC UA Server, expand Devices, expand, [Simulation], Line 1, Conveyor. Highlight “Temperature” and press “Commit”.
      * A screenshot of a cell phone

        Description automatically generatedEnsure your tag configuration looks similar to the following:
      * Click “Apply” and “OK” to accept the tag configuration.
  + Tag 2: Pressure
    - To the left of the “Properties” section, click the “Add Tag” button, and select OPC Tag:
    - Edit the “Basic Properties”
      * Name the tag “Pressure”
      * Change its Data Type to float
      * Click the Link icon to the right of “OPC Server”, click “Browse OPC”, right click on Ignition OPC UA Server, and click “Copy Item Path”, then “Commit”. Right click in the space to the right of OPC Server and paste what’s copied in your clipboard.
      * Click the Link icon to the right of OPC Item Path, click “Browse OPC”, expand Ignition OPC UA Server, expand Devices, expand, [Simulation], expand Line 1, expand Stamping Machine. Highlight “Pressure” and press “Commit”.
      * Click “Apply” and “OK” to accept the tag configuration.
  + Tag 3: Vibration
    - To the left of the “Properties” section, click the “Add Tag” button, and select OPC Tag:
    - Edit the “Basic Properties”
      * Name the tag “Vibration”
      * Change its Data Type to float
      * Click the Link icon to the right of “OPC Server”, click “Browse OPC”, right click on Ignition OPC UA Server, and click “Copy Item Path”, then “Commit”. Right click in the space to the right of OPC Server and paste what’s copied in your clipboard.
      * Click the Link icon to the right of OPC Item Path, click “Browse OPC”, expand Ignition OPC UA Server, expand Devices, expand, [Simulation], expand Line 1, expand Conveyor. Highlight “Vibration” and press “Commit”.
      * Click “Apply” and “OK” to accept the tag configuration.
      * Click “Apply” and “OK” to accept the Pump configuration.
    - A screenshot of a cell phone

      Description automatically generatedBy the time you’re finished adding all your tags, the Pump should look like this:
  + Add the line to the project:
    - Under the Tag Browser, expand All Providers, right click on “default” 🡪 New Tag 🡪 New Folder 🡪 Line 4 🡪 OK.
    - Right click “Line 4” 🡪 New Tag 🡪 New Data Type Instance 🡪 Pump
      * Give the instance the name “Pump”, press “Apply” and “OK”.
  + Trigger a birth message:
    - Under the Tag Browser, expand Tag Providers, expand default, expand Sim Controls. To the right of “New Birth”, check the checkbox and “Write Once”.
* This triggers an MQTT message that defines your new hierarchy, with Line 4 and the Pump included. You should see your new models and assets in SiteWise.