Industrial Machine Connectivity (IMC) Kit

Master User Guide

Table of Contents

[Industrial Machine Connectivity QuickStart Getting Started Guide 3](#_Toc44933622)

[Deployment Types 4](#_Toc44933623)

[Virtual: 4](#_Toc44933624)

[Physical 4](#_Toc44933625)

[Physical - Greenfield 4](#_Toc44933626)

[Physical - Brownfield 4](#_Toc44933627)

[Data Flow Options 5](#_Toc44933628)

[Option 1 5](#_Toc44933629)

[Option 2a 5](#_Toc44933630)

[Option 2b 5](#_Toc44933631)

[Creating AMC Drivers 6](#_Toc44933632)

[Instructions 6](#_Toc44933633)

[Appendix 7](#_Toc44933634)

[Artifacts 7](#_Toc44933635)

[AMC-Approved DynamoDB Format 7](#_Toc44933636)

[Add a line and device to an Ignition project 8](#_Toc44933637)

# Industrial Machine Connectivity (IMC) QuickStart Getting Started Guide

Introduction

The IMC framework is designed to enable customers and partners to get data from their assets to AWS in a simple, structured process so they can rapidly realize the business value that is derived from that data. The IMC Quick Start has the capability to convert customers’ existing asset hierarchy definitions (i.e. factory, lines, machines, tags, etc.) defined in partner edge applications like Inductive Automation’s Ignition Server or PTC’s KEPServerEX to the equivalent asset hierarchy within AWS IoT SiteWise. This capability is enabled by the Asset Model Converter (AMC), a component of the IMC architecture. With asset hierarchies defined within IoT SiteWise, customer data can be ingested continuously to the AWS cloud and all the pertinent metadata is readily accessible for applications that will use that data to deliver business value, such as asset condition monitoring dashboards.

With multiple ingestion patterns from edge to the cloud, customers and partners may choose the IMC Framework path that suits the needs of their specific use case. The first path is via the IoT SiteWise Connector running in an AWS IoT Greengrass core (via OPC-UA) to IoT SiteWise in the cloud. The second path is sending data directly from edge applications (i.e. Ignition Server with Cirrus Link MQTT Transmission module) to AWS IoT Core via MQTT. The third path is sending data from the edge application (i.e. Ignition Server or KEPServerEX) to IoT Greengrass where it can then be processed/filtered by customer-defined lambda functions, used for local machine learning model inference or consumed by a containerized Docker application processing. Raw and processed data can then be transmitted to the AWS cloud through a number of different paths including IoT Core, Kinesis Data Streams, or AWS IoT Analytics. These edge processing capabilities are not part of the IMC kit and partners/customers can add those components as needed.

**Configuration and deployment process -** The IMC kit will include the software and a list of hardware required to connect a customer's assets to the cloud and visualize their data. The primary objective of the framework is to enable partners to deliver a fast, inexpensive proof of value so that the customer can see real business value from the IIoT solution and will want to develop and deploy a complete production solution that addresses all of their critical use cases. One example of a proof of value deployment a partner could deliver would be helping enable a customer to visualize near real-time operational metrics and perform Root Cause Analysis (RCA) of their assets when a line goes down. The IMC software will be packaged into CloudFormation templates that simplify the deployment process. There will be two deployment modes of the framework: **virtual (for evaluation and training)** and **physical (for real customer deployments)**. The CloudFormation templates will provision the cloud resources and generate the scripts required to bootstrap physical edge devices. All documentation required to deploy this framework will be included with the CloudFormation templates.

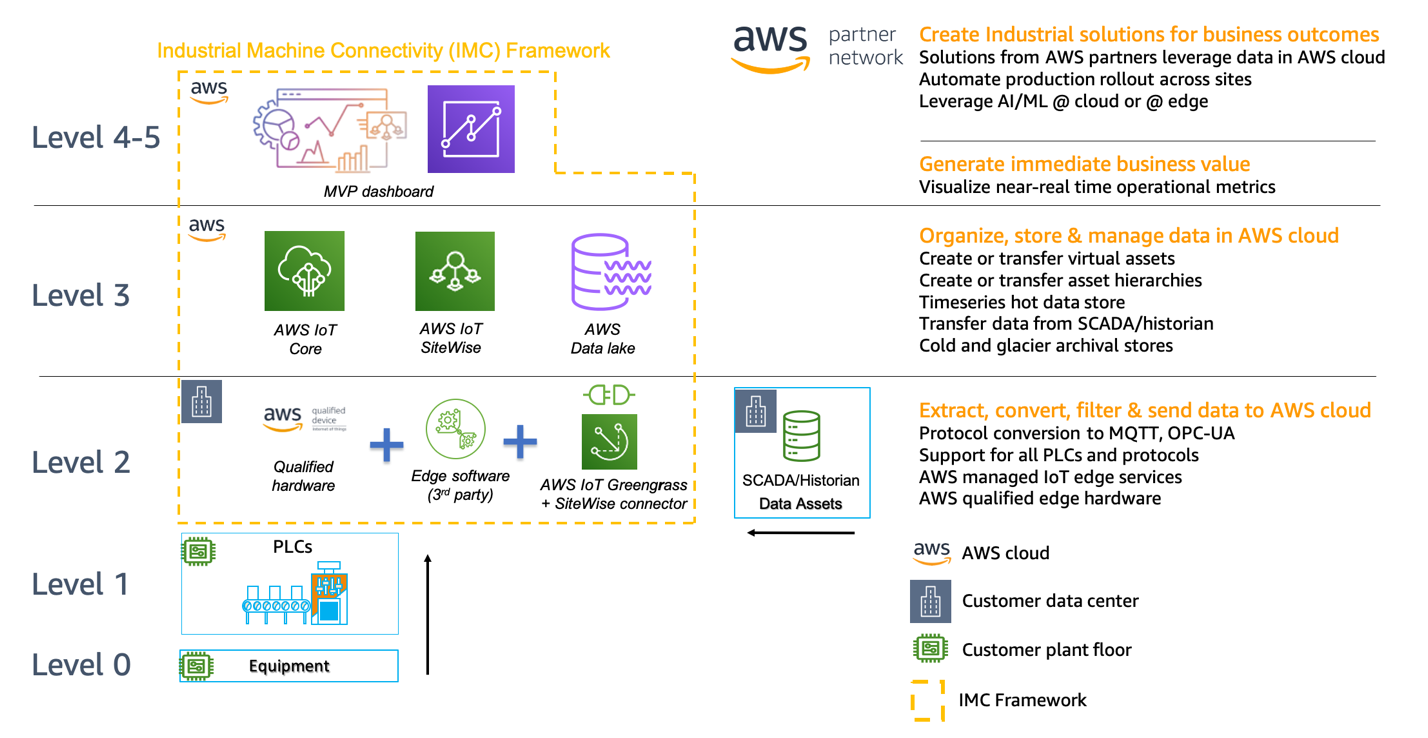
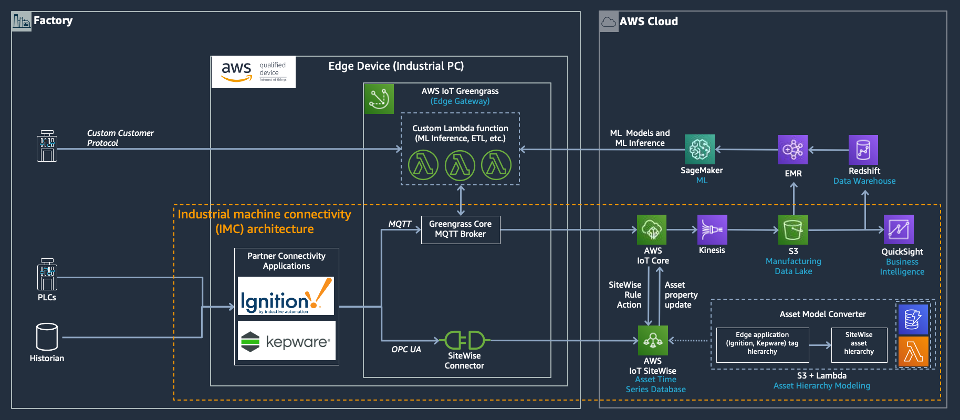


Figure 1: IMC Framework Diagram

The Industrial Machine Connectivity (IMC) kit enables connecting industrial assets into AWS cloud services along with visualizing data using AWS IoT SiteWise Monitor and QuickSight dashboards. This kit integrates with AWS IoT Partner edge software such as Inductive Automation’s Ignition Server and PTC’s KEPServerEX. These partner edge software applications handle the industrial protocol translation from the PLCs, other devices, historians and SCADA systems they connect to. The IMC architecture integrates both the data collected by the edge software applications and the asset model hierarchy that they maintain. The IMC architecture is shown below:



**Figure 2: IMC Reference Architecture**

A key component of this kit is the Asset Model Converter (AMC). The AMC is a serverless, module-based framework supporting mapping edge-based asset modeling software conventions into AWS IoT SiteWise models and assets.

Currently as part of this kit, we provide sample modules (drivers) for:

* Inductive Automation (Ignition Server)
  + Cirrus Link Module MQTT driver
  + Ignition Project File Export driver
* PTC (KepServer)
  + KepServer Project File Export driver

Based on the driver selected, the AMC will ingest the edge software’s (i.e. Ignition or KEPServerEX) native asset model definition(s) and automatically provision the matching asset hierarchy within AWS IoT SiteWise. This automatic mapping enables application builders, whether they be customer’s own developers, SIs, GSIs or AWS ProServe team, to have immediate access to the customer’s asset hierarchy within a managed service in the AWS Cloud (AWS IoT SiteWise).

The objective of the IMC kit is to accelerate industrial machine connectivity to the AWS cloud so solution builder teams can demonstrate real business value to customers faster and with lower integration cost and effort than is currently possible.

# Deployment Types

The IMC kit can be deployed in 3 configurations:

## Virtual:

The virtual deployment is intended for demonstration, training and evaluation of the Kit’s capabilities. EC2 instances will be launched to simulate edge gateway hardware but in all other respects the experience will mirror that of the real physical deployment. This deployment mode relies on simulated tag values generated by the partner edge software. There are no physical PLCs or sensors that are being connected.

## Physical

Physical deployment of the IMC kit enables users to deploy edge software (i.e. AWS IoT Greengrass and partner edge software) on physical industrial PCs that are ready to connect to physical devices (I.e. PLCs)/historians/SCADA systems on the customers plant floor. The physical deployment has two flavors:

### Physical - Greenfield

AWS IoT Greengrass and the partner edge software will be running on a single industrial PC.

### Physical - Brownfield

AWS IoT Greengrass will run standalone on an industrial PC and will connect to partner edge software application that is already running on the customers premises (i.e. on a VM in the server room of a manufacturing plant). We assume our access to the hardware is limited or non-existent, and our ability to reconfigure the Edge Software Application Server is limited to additive changes only.

-------------------------------------------------------------------------------------------------------------------------------

The type of deployment (Virtual or Physical) determines whether to use physical edge hardware (Physical industrial PCs) or virtual edge hardware (EC2) and how connectivity and security is configured. All other cloud-based resources are largely the same.

# Data Flow Options

In addition to the Virtual/Physical edge hardware distinction, the IMC kit supports three types of data flow architectures. Each type outlines different methods of data ingestion from the edge environment into the AWS Cloud.

## Option 1

**Edge Software Application OPCUA Server -> AWS IoT SiteWise:**

In this variant, we have the AWS Greengrass SiteWise Connector configured to connect to the Edge Software Application OPCUA Server. All telemetry data will flow directly into AWS IoT SiteWise.

## Option 2a

**Edge Software Application Server -> AWS IoT Core:**

In this variant, the Edge Software Application Server has some kind functionality to connect to IoT Core via MQTT. All telemetry data is pushed from the Edge Software Application Server to AWS IoT Core, and from there usually pushed to S3 or a similar data lake for processing.

## Option 2b

**Edge Software Application Server -> AWS Greengrass Core -> IoT Core -> S3:**

Option 2b is almost identical to option 2a, except we instead have the Edge Software Application Server pushing MQTT data messages to AWS Greengrass Core first, and then those messages are forwarded on to AWS IoT Core.

# Getting Started

To get started, choose which of the 3 deployment types you would like to launch:

1. Virtual
   1. User guide: “IMC - Virtual Deployment User Guide”
2. Physical - Greenfield
   1. User guide: “IMC - Physical - Greenfield Deployment User Guide”
3. Physical - Brownfield
   1. User guide: “IMC - Physical - Brownfield Deployment User Guide”

Select the appropriate user guide document for that specific launch configuration and follow the instructions. Ensure that you are preparing pre-requisite resources and launching the CloudFormation stack in one of the 3 supported regions for the IMC kit:

* us-east-1
* us-west-2
* eu-west-1

See the appendix below for additional information about:

1. Creating new AMC drivers
   1. This is relevant for partners or customer that wish to integrate a new edge software application with the IMC kit
2. Artifacts
   1. List of artifacts that are required to launch the CloudFormation templates
3. AMC-Approved DynamoDB Format
   1. Details the format of the 2 DynamoDB tables that the AMC uses to store the SiteWise asset model and asset information needed to provision resources in SiteWise.
4. Add a line and a device to an Ignition Project
   1. These instructions show a user how to add an additional device to an Ignition Server project and how that new device will be provisioned in AWS IoT SiteWise via the AMC.

# Appendix

# 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]

## 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
* 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.