Quick Start Guide

intel

Network and Edge Reference System Architectures -Industrial Controller

Develop and verify Industrial Controller solutions using BMRA and VMRA on Intel Atom[®] and Intel[®] Core[™] platforms.

		6 - C C C C C C C C	
l r	ntrod	luction	

defined solutions.

Abhijit Sinha	The Intel Network and Edge Reference System Architectures (Reference System ¹) are forward-
Alek Du	looking cloud-native reference platforms aiming to ease bare-metal and Kubernetes* cluster
Mathieu Sobrero	automatically deployed using Ansible* playbooks that are designed to optimally support diverse use
Shu Ren	cases across network locations.
	This document is a quick start guide to configure the Container Bare Metal and Virtual Machine Reference System Architecture (BMRA and VMRA) on Intel Atom® and Intel® Core™ processor - based platforms for Industrial Controller workloads .
	The Reference System has a variety of configuration profile settings for different network traffic workloads. This quick start guide enables the Industrial Controller solution using the On-Premises Software Defined Factory Profile. This document demonstrates the implementation of a Mixed Deployment using the BMRA and VMRA, supporting Industrial Process Automation and Machine Controller use cases by adding an optimized RA Configuration Profile.
	This guide highlights the deployment model based on using Intel® Edge Controls for Industrial (Intel® ECI) software for industrial process automation and machine controller deployment mechanism using Reference Systems to transition the industrial control systems to software-

Hardware BOM

Authors

Following is the list of the hardware components that are required for setting up Reference Systems:

Ansible host	Laptop or server/VM running a UNIX base distribution with internet connectivity
Target platforms (Process Automation)	Control Plane Node: Any Intel® Xeon® Gold processor-based platform DCN Node: 12th Gen Intel® Core™ i7-12700TE processor Remote IO Emulator Node: 12th Gen Intel® Core™ i7-12700E processor or Intel® Atom® processor x6000 series
Target platform (Machine Controller)	12th Gen Intel® Core™
BIOS	Use the default BIOS settings

¹ In this document, "Reference System" refers to the Network and Edge Reference System Architecture.

Software BOM

Following is the list of the software components that are required for setting up Reference Systems:

Edge Controls for Industrial (ECI) services	Edge Control for Industrial v3.0.2 Process Automation, Codesys OPC UA client, EtherCAT, Machine Controller, Open PLC (programmable logic controller)	
Container Runtime	Docker	
os	Ubuntu 22.04.2 Desktop (on DCN and Remote IO node) Ubuntu 22.04.2 Server (on Control Plane node)	

For details of the software versions for the **On-Premises Software Defined Factory Profile**, refer to the BMRA/VMRA User Guides listed in the <u>Reference Documentation</u> section.

Getting Started

Prerequisites and Deployment Model

Before starting the deployment, perform the following steps:

- A fresh OS installation is expected on the controller and target nodes to avoid a conflict between the RA deployment
 process with the existing software packages. To deploy RA on the existing OS, ensure that there are no prior Docker or
 Kubernetes* (K8s) installations on the servers.
- The hostname must be in lowercase, numerals, and hyphen -.

For example: wrk-8 is acceptable; wrk_8, WRK8, Wrk^8 are not accepted as hostnames.

• The platforms are Network Time Protocol (NTP) synced, i.e., they must have the correct date and time.



Figure 1: RA Deployment Setup for Factory Process Automation



Figure 2: Industrial Machine Controller Using VMRA

Installation Flow for RA Deployment

Ansible playbooks are used to deploy the Reference Systems using the on_prem_sw_defined_factory profile. Before the playbooks can be run, there are a few steps to prepare the environment and change relevant configuration options.



Figure 3: RA Deployment Flow Using Ansible Playbooks

Step 1 – Set Up the System

The following steps assume that both the Ansible host and target server are running Ubuntu as the operating system. For RHEL, use yum or dnf as the package manager instead of apt.

Ansible Host

- 1. Install necessary packages (some might already be installed):
 # sudo apt update
 # sudo apt install -y python3 python3-pip libselinux-python3 openssh-server git
 # pip3 install -upgrade pip
- 2. Generate a SSH keypair if needed (check /root/.ssh/): # ssh-keygen -t rsa -b 4096 -N "" -f ~/.ssh/id_rsa
- 3. Copy the public key to the target servers controller and worker nodes: # ssh-copy-id root@<target node IP>
- 4. Verify passwordless connectivity to the target servers:
 # ssh root@<target node IP>

System Setup



Target Server

Install necessary packages (some might already be installed): # sudo apt install -y python3 openssh-server lshw

Step 2 – Download and Install

Ansible Host

1. Download the source code from the GitHub repository for the Reference System server:

```
# git clone https://github.com/intel/container-experience-kits/
# cd container-experience-kits
```

- # git checkout v23.10
- 2. Set up Python* virtual environment and install dependencies:

```
# python3 -m venv venv
# source venv/bin/activate
# pip3 install -r requirements.txt
```

3. Install Ansible dependencies for the Reference System:

ansible-galaxy install -r collections/requirements.yml

Step 3 – Configure

The **On-Premises Software Defined Factory** configuration profile (on_prem_sw_defined_factory) is used for both industrial process automation and industrial machine controller deployment.

Industrial Process Automation Configuration

A mixed cluster deployment comprised of VMs and bare metal (BM) hosts is used for the industrial process automation deployment.

Target Host

To enable the TCC feature in the BIOS, navigate to Intel Advanced Menu > Intel® Time Coordinated Computing (Intel® TCC). Set Intel® TCC Mode to <Enabled>. Save your changes and exit the BIOS. The system reboots.

Ansible Host

1. Generate the configuration files:

```
# export PROFILE=on_prem_sw_defined_factory
# export ARCH=core
# make examples
# cp examples/vm/${PROFILE}/inventory.ini .
# cp -r examples/vm/${PROFILE}/group_vars examples/vm/${PROFILE}/host_vars .
# cp -r examples/k8s/${PROFILE}/host_vars .
```

2. Update the *inventory.ini* file to match the deployment setup. The values for <* hostname> and <* IP> must be updated to match the target systems in the BMRA cluster, as shown in <u>Figure 1</u>.

In this example, we use vm-ctrl-1 and vm-work-1 for the VMRA configuration on the "Control Node". The difference is we also defined a "DCN" and "Remote-IO" host, which will be a kube_node in the coming mixed cluster.

```
# cd container-experience-kits
# vim inventory.ini
[all]
host-for-vms-1 ansible_host=<Controller IP> ip=<Controller IP> ansible_user=root
<DCN hostname> ansible_host=<DCN IP> ip=<DCN IP> ansible_user=root
<Remote-IO hostname> ansible_host=<Remote-IO IP> ip=<Remote-IO IP> ansible_user=root
localhost ansible_connection=local ansible_python_interpreter=/usr/bin/python3
[vm_host]
host-for-vms-1
[kube_control_plane]
#vm-ctrl-1
```



Configure



Download & Install

```
[etcd]
#vm-ctrl-1
[kube_node]
#vm-work-1
<DCN hostname>
<Remote-IO hostname>
[k8s_cluster:children]
kube_control_plane
kube_node
[all:vars]
ansible python interpreter=/usr/bin/python3
```

3. Update the host vars filename of BM hosts with the target machine's hostname:

```
# cp host_vars/node1.yml host_vars/<DCN hostname>.yml
# cp host_vars/node1.yml host_vars/<Remote-IO hostname>.yml
```

4. For a mixed BM host, we need to add the VXLAN gateway IP and the VXLAN physical network info to the *host_vars/host-for-vms-1.yml*. The user is required to set up a proper VXLAN physical network based on their real situation. The subnet must be the same as is used for vm_hosts.

Note: vxlan_gw_ip must belong to the same subnet as vm_host.

```
# vim host_vars/host-for-vms-1.yml
vxlan_gw_ip: "172.31.0.101/24"
vxlan physical network: "11.0.0.0/8"
```

The $vxlan_gw_ip$ variable defines the VXLAN bridge IP on the mixed BM host. This is also the IP that the cluster will "see" the node. The $vxlan_physical_network$ is used to auto select the physical network adapter that will carry on the VXLAN transport. These two parameters are very similar to the ones on VM host host_vars unless only the first VM host requires $vxlan_gw_ip$.

5. Update the *host_vars/host-for-vms-1.ym*/ with the following settings. This file creates the three VMs on the Control Node with the specified configuration.

```
# vim host vars/host-for-vms-1.yml
vms:
- type: "ctrl"
  name: "vm-ctrl-1"
  cpu total: 8
  memory: 20480
  vxlan: 120
- type: "work"
  name: "vm-work-1"
  cpu total: 16
  memory: 61440
  vxlan: 120
 type: "vm"
  name: "vm-1"
  cpu total: 4
  memory: 61440
  vxlan: 120
```

6. Update the *host_vars/<DCN hostname>.yml* with the following settings. Here we enable the eci-processautomation and set the DCN node as Codesys OPCUA client.

```
## Intel ECI (Edge Controls for Industrial)
intel_eci_enabled: true # if true, deploy Intel ECI
intel_eci:
    eci-process-automation: true
    eci-manufacturing-equipment: false
    eci-discrete-manufacturing: false
```

excat_dp_enabled : true

```
opcua_framework:
   codesys_opcua_client: true
   standalone_opcua_server: false
cat_enable: true
## Change the CAT config as per your CPU architecture
cat_define: "llc:0=0x0f;llc:1=0xf0"
cat_affinity: "llc:0=0;llc:1=1,3"
ethercat mac: ""
```

7. Update the *host_vars/<Remote-IO hostname>.yml* with the following settings. Here we enable the eci-processautomation and set the Remote-IO node as Codesys OPCUA server.

```
## Intel ECI (Edge Controls for Industrial)
intel_eci_enabled: true # if true, deploy Intel ECI
intel_eci:
    eci-process-automation: true
    eci-manufacturing-equipment: false
    eci-discrete-manufacturing: false
excat_dp_enabled: true
opcua_framework :
    codesys_opcua_client : false
    standalone_opcua_server: true
cat_enable: true
## Change the CAT config as per your CPU architecture
cat_define: "llc:0=0x0f;llc:1=0xf0"
cat_affinity: "llc:0=0;llc:1=1,3"
```

ethercat_mac: ""

8. Download and copy the Intel ECI csl-excat-v3.0.0.tar.gz file to the Ansible host /tmp folder.

```
## Download edge_controls_industrial.zip from eci.intel.com (will redirect to
https://edgesoftware.intel.com/edge_controls_industrial, select ubuntu 22.04 and version
3.02 to download)
# Install the package on Ansible host
unzip edge_controls_industrial.zip
cd edge_controls_industrial
chmod +x edgesoftware
./edgesoftware download
cd Edge_Controls_for_Industrial_3.0.2/Edge_Controls_for_Industrial
unzip release-eci_3.02.zip
cp ./release-eci_3.0.2/Support/Edge-Orchestration/csl-excat-v3.0.0.tar.gz /tmp
```

Machine Controller Configuration

VMRA is used for the deployment of the machine controller.

Ansible Host

1. Generate the configuration files:

```
# export PROFILE=on_prem_sw_defined_factory
# export ARCH=core
# make examples
# cp examples/vm/${PROFILE}/inventory.ini .
# cp -r examples/vm/${PROFILE}/group_vars examples/vm/${PROFILE}/host_vars .
```

2. Update the *inventory.ini* file to match your deployment setup. The <* IP> must be updated to match the target system in the VMRA cluster, as shown in Figure 2. In this example, we use vm-1 for the VMRA configuration on the Machine Controller Node.

```
# cd container-experience-kits
# vim inventory.ini
```

```
[all]
host-for-vms-1
                       ansible host=<Controller IP> ip=<Controller IP> ansible user=root
localhost
                       ansible connection=local ansible python interpreter=/usr/bin/python3
[vms]
[vm host]
host-for-vms-1
[kube control plane]
#vm-1
[etcd]
#√m−1
[kube node]
#∨m-1
[k8s_cluster:children]
kube control plane
kube node
[all:vars]
ansible python interpreter=/usr/bin/python3
```

3. Update the *host_vars/host-for-vms-1.yml* with the following settings:

Set hashed password for root user inside VMs. Current value is just placeholder. # To create hashed password use e.g.: openssl passwd -6 -salt SaltSalt <your password> vm hashed passwd: 'xxxxxxx' # Fill your physical network mask to this field. vxlan_physical_network: "xxxxxx" # Fill your CAT config to these two fields. Example refers to the vm configuration below. cat define: "llc:0=0x0f;llc:1=0xf0" cat affinity: "llc:0=0;llc:1=1,3" # Update "vms" list with generic VM config used for ECI. Use example below as a guide and follow the comments to change the resources. # Example assumes that there is only one numa on the host machine. vms: - type: "vm" name: "vm-1" cpus: '2,3' # CPUs allocated to VM numa: 0 cpu total: 2 # The number of CPUs, consistent with "cpus" field memory: 4096 # Memory allocated to VM vxlan: 120 pci: - "03:00.0" # BDF of PF device to be passthrough to VM for ethercat.

Mac address of this PF device needs to be configured in "ethercat_mac" variable inside host_vars file for this VM. # Other PCI devices (BDFs) can be configured here as well

4. Update the host vars/vm-ctrl-1 filename vm-l:

mv host vars/vm-ctrl-1.yml host vars/vm-1.yml

5. Disable Kubernetes in *group_vars/all.yml*:

```
# vim group_vars/all.yml
kubernetes: false
```

The following group var configurations are common to both Process Automation and Machine Controller use cases:

1. Add the CPU SKU number of the controller node to the group_vars/all.yml:

```
# vim group_vars/all.yml
unconfirmed cpu models: ["<ADD CPU SKU number here, eg:6238L>"]
```

2. Add the Intel[®] ECI repository link to group_vars/all.yml. Contact eci-support@intel.com for information on how to access this repository.

```
# vim group vars/all.yml
intel_eci_repo: <a href="https://eci.intel.com">https://eci.intel.com</a>
```

3. If the server is behind a proxy, update group_vars/all.ym/ by updating and uncommenting the lines for http proxy, https proxy, and additional no proxy.

```
# vim group vars/all.vml
## Proxy configuration ##
http_proxy: "http://proxy.example.com:port"
https_proxy: "https://proxy.example.com:port"
additional_no_proxy: ".example.com,mirror_ip"
```

4. (Required) Apply required patches for Kubespray:

ansible-playbook -i inventory.ini playbooks/k8s/patch kubespray.yml

5. (Optional) It is recommended that you check the dependencies of components enabled in group_vars and host_vars with the package dependency checker:

ansible-playbook -i inventory.ini playbooks/preflight.yml

6. (Optional) Verify that Ansible can connect to the target server by running the following command and checking the output generated in the *all_system_facts.txt* file:

ansible -i inventory.ini -m setup all > all system facts.txt

Step 4 – Deploy

Ansible Host

The Reference System can be deployed by using the following command:

Industrial Process Automation Deployment

ansible-playbook -i inventory.ini playbooks/on prem sw defined factory.yml --flush-cache

Machine Controller Deployment

ansible-playbook -i inventory.ini playbooks/vm.yml -e "eci package= eci-manufacturing-equipment"

Note: The eci package supports three packages: eci-process-automation, eci-manufacturing-equipment, ecidiscrete-manufacturing. The intel ecifield in vm-l.ym/ can also specify the Intel ECI packages installed. However, the package specified by -e eci package= overwrites the intel ecifield in host-for-vms.yml.

Cleanup: If the playbook fails or if you want to clean up the environment to run a new deployment, you can optionally use the provided Cluster Removal Playbook to remove any previously installed Kubernetes and related plugins.

ansible-playbook -i inventory.ini playbooks/redeploy cleanup.yml

Step 5 – Validate

Industrial Process Automation Verification

Below are the steps to validate the Industrial Process Automation use case.

Ansible Host

1 To interact with the Kubernetes CLI (kubectl), start by connecting to the controller VM node in the cluster, which can be done using the following commands:

ssh root@vm-ctrl-1

2. Once connected, the status of the Kubernetes cluster can be checked:

kubectl get nodes -A

Deploy



V	_
-	-1
-	-1

Validate

root@vm-ctrl-1:	<pre>v# kubect1</pre>	get node -A		
NAME	STATUS	ROLES	AGE	VERSION
intel-adl-tank	Ready	<none></none>	4d5h	v1.27.1
intel-tank-rpl	Ready	<none></none>	4d5h	v1.27.1
vm-ctrl-1	Ready	control-plane	4d5h	v1.27.1
vm-work-1	Ready	<none></none>	4d5h	v1.27.1

Note: The intel-adl-tank is the DCN node and the intel-tank-rpl is the Remote_IO node.

3. The ExCAT feature can be verified on the DCN and Remote-IO nodes:

kubectl describe node <node_name> | grep exact



4. Launching of industrial services was completed in <u>Step 4</u>, and the Open Platform Communications United Architecture (OPC UA) Client Human Machine Interface (HMI) can be accessed through a browser at http://<DCN_IP_ADDRESS>:8080. <u>Configure and Run OPC UA Client Benchmark</u> is a complete guide on the HMI and how to connect it to the OPC UA server instance.

UA Server Address Space —		CPC UA Connection Settings
Browsed Node StaticVariab	les	Server URL opc.tcp://127.0.0.1;4841 Connect intel
Children		
Nodename	nodeClass	Security Policy integraphicanaalionarg/ou/security Disconnect
UInt64	Variable	Secure Time (1771) 22/
UInt32	Variable	Security Mode Dpc0a MessageSecurityMode_Nonk ~ Usemame Publishing Interval [ms] 50 Security Mode Client Time (17:02:24
Uint16	Variable	Time Offset UTCL TAA
String	Variable	User Policy Anonymous V Password *** Subscription Priority 0 time Order (Order, Free
SByte	Variable	
nt64	Variable	A factor of the second s
int32	Variable	Monitoreo tierns
int16	Variable	Name Value IOWaltLatency [ns] t4(SvrTx) t3(SvrSampi) t2(SvrAk) t1(Cl
loat	Variable	0 Uint64 11534 42155096 20:53:58.311428 20:53:58.262152 20:53:58.26172
Double	Variable	
ByteString	Variable	
Byte	Variable	
Soolean	Variable	
Selected Node Attributes		Timing Metrics
Selected Node Attributes		Timing Metrics
Selected Node Attributes	Value ns=2;=Uint64	Timing Metrics Train Duration: T#1m Start of Min / Average / Max (us) 2078 3941 14245 3054
Selected Node Attributes	Value ns=23=Ulnt64 2	Timing Metrice T Traing Metrice Start of Current Cycle 2876 3641 14245 T Tak Min / Average / Max (uz) 1296 3054 Duration: Terim Cycle: 1197 X/Rax Reserve: 1196 Next Cycle Start of Next Cycle Start of Next Cycle
Selected Node Attributes	Value rs=23=Uint64 2 rs=2Uint64	Timing Metrics Train Metrics Start of Current Cycle Control to the start of the
Selected Node Attributes	Value ns=21=UInt64 2 ns=2UInt64 UInt64 UInt64	Timing Metrics Totalc T
Selected Node Attributes Node ID ns=2;s=UInt64 Attributes Nodeld NodeClass BrowseName DisplayName Desription	Value ns=23=Uin54 2 ns=2Uin54 Uin54 Uin54 Uin54	Timing Metrics Train Current Cycle Start of Current Cycle Server Tx Calculation Client Tx Jitter Working Interime 11116 Variance 21116 Varian
Eelected Node Attributes	Value ns=2;s=Uint64 2 ns=2;Uint64 Uint64 Uint64 0	Timing Metrics Totalc Train (Min / Average / Max (u)) Total (u) To
selected Node Attributes	Value ns=23=UIn64 2 ns=2UIn64 UIn64 UIn64 0 0	Timing Metrics Totalc T
elected Node Attributes	Value ns=2;s=Uint64 2 uint64 Uint64 Uint64 0 0 0 0 0	Timing Metrics Train of Current Cycle Server Tx Calculation Calculation Client Tx Value rs=2;s=Ult54 T T T T T T T T T T T T T T T T T T T
elected Node Attributes	Value ns=2;s=Uint64 Uint64 Uint64 Uint64 Uint64 Occub_BadAttributedinvaid Occub_BadAttributedinvaid	Timing Metrics Totalc T
Node ID ns=2;3=Uint64 Attributes Voceidi Voceidi SrowseName DisplayName Displ	Value ns=21=Uin64 2 ns=2Uin64 Uin64 0 0 0x0x8.8x84ttributedinxeid Opcu8_8x84ttributedinxeid Opcu8_8x84ttributedinxeid	Timing Metrics Train Start of Current Cycle Server Tx Calculation Client Tx Understors + 12116 Understors +
elected Node Attributes	Value ns=2;s=Uint64 2 ns=2;Uint64 Uint64 Uint64 0 0 0 0 0 0 0 0 0 0 0 0 0	Timing Metrics Turning Metrics Totalc Train / Average / Max (ui) Start of Current Cycle Z078 3641 14245 Use / Max (ui) Server Tx Calculation Client Tx Jitter Tmin_exec Z256 Tmin_exec Z256 Timin_exec Z256 Timin_exec Timin
Vode Attributes	Value ns=21=Uin64 2 ns=2Uin64 Uin64 Uin64 Uin64 Oocula_BadAttriouteldinvaid Oocula_BadAttriouteldinvaid Oocula_BadAttriouteldinvaid Oocula_BadAttriouteldinvaid Oocula_BadAttriouteldinvaid	Timing Metrics Start of Current Cycle Server Tx Jitter Timin_gexec Calculation Timin_gexec Calculation Timin_gexec Calculation Timin_gexec Calculation Timin_gexec Calculation Timin_gexec Calculation Timin_gexec Calculation Timin_gexec Calculation Timin_gexec Calculation Timin_gexec Calculation Timin_gexec Calculation Timin_gexec Calculation Timin_gexec Calculation Timin_gexec Calculation Timin_gexec Calculation Timin_gexec Calculation Timin_gexec Calculation Timin_gexec Timin_gex
elected Node Attributes	Value ns=2;s=Uint64 2 ns=2;Uint64 Uint64 Uint64 0 0 OpcUa_BadAttributeidinvaid OpcUa_BadAttributeidinvaid OpcUa_BadAttributeidinvaid OpcUa_BadAttributeidinvaid OpcUa_BadAttributeidinvaid Is38	Start of Current Cycle Tcalc Min Tcalc Average Tx Max (ut) Duration: T#1m Cycle: 1197 Start of Next Cycle Server Tx Calculation Client Tx Volverine=2280 Duration: T#1m Cycle: 1197 Start of Next Cycle Server Tx Calculation Client Tx Volverine=2280 Take Take Jitter Tmin_exec 2258 Tjitter 1395 Average Min (us) Average Tmin_exec 2258 Tjitter 1395 Average Min (us) Average Tave_exec 6939 4939 4939
Selected Node Attributes	Value ns=2s=Uln64 2 ns=2Uln64 Uln64 Uln64 Uln64 0 0 0 0 0 0 0 0 0 0 0 0 0	Timing Metrics Turing Metrics Toring Metrics Tart of Current Cycle Tart of Current Cycle Tart Calculation Client Tx Usue ns=2;=Uint64 Tipitter Tipitter Timin_exec Esset Tave_exec Esset Tave_exec Tave_exec Tave_exec Tave
Node ID n=2:3=Uint64 Attributes Nodell NodeClas RowseName DisplayName Description WriteNwisk Sabstrat Sabstrat Sabstrat ContainsNeLoops EventNotIfier Value DataType ValueBank	Value ns=2z=Uin54 2 ns=2Uint64 Uint64 Uint64 0 <td>Timing Metrics Train Min / Average / Max (ur) Start of Current Cycle 2070 3041 14245 Calculation Client Tx Value rs=2;p-Uint54 Unit (ur) 7 average Min (ur) 7 average Min (ur) 7 average Min (ur) 7 average Teaching Table Tab</td>	Timing Metrics Train Min / Average / Max (ur) Start of Current Cycle 2070 3041 14245 Calculation Client Tx Value rs=2;p-Uint54 Unit (ur) 7 average Min (ur) 7 average Min (ur) 7 average Min (ur) 7 average Teaching Table Tab
Node ID ns=2:s=Uint64 Attributes Nodelid StoedClass BrowseName Discription WriteMask UserWriteMask Sakostrack Symmetric OratainNoLoops EventNotIfier ValuePank WalePank WalePank	Value ns=2;=-Uint64 2 ns=2;Uint64 Uint64 Uint64 Uint64 0 0 0 0 0 0 0 0 0 0 0 0 0	Timing Metrics Turning Metrics Duration T#Im Cycle 1197 Start of Next Cycle Received: 1195 Start of Next Cycle Received: 1195 Next Cycle Received: 119 Turning Metrics Client Tx Value: ns=2;s=Uint64 Turning Metrics Turning Metrics Turning Metrics Next Cycle
Selected Node Attributes Node ID ns=2;s=Lint64 Attributes NodeClas BrowseName Description WriteMask Skabstract Symmetric InverseName ContainavoLoops ContainavoL	Value ns=21=Uin54 2 ns=2Uin54 Uin54 Uin54 Uin54 Uin54 Oocub_BadAttributedinvaid Oocub_BadAttributedinvaid Oocub_BadAttributedinvaid Oocub_BadAttributedinvaid Oocub_BadAttributedinvaid 1638 ns=0 =9 -2array and matrix not convertible -3	Thing Metrics Start of Current Cycle Server Tx Jitter Tx Calculation Client Tx Workload kerstions + 12116 Control Logic & Executon Control Logic & Executon Control Logic & Executon Control Logic & Executon
Selected Node Attributes	Value ns=2s=Unt64 2 ns=2Unt64 Unt64 Unt64 Unt64 0 0 0 0 0 0 0 0 0 0 0 0 0	Trining Metrics Trining Metrics Start of Current Cycle Server Tx Jitter Min (u) Calculation Calculation Current Cycle Calculation Control Logic & Execution Control Logic & Execution
Node ID ns=2;s=Uint64 Attributes Nodella Nodella Nodella StrowseName Description WriteMask Substrat Symmetric ContainNoLoops EventNotifier Value DataType ValueRank ArrayDimensions AccessLevel MinimumSamplingInterval	Value ns=21=Uin54 2 ns=2Uin54 Uin54 Uin54 Uin54 0 0 00cub_84AttributeIdinvaid 00cub_84AttributeIdinvaid 00cub_84AttributeIdinvaid 00cub_84AttributeIdinvaid 00cub_84AttributeIdinvaid 00cub_84AttributeIdinvaid 1535 ns=0j=9 -2 array and matrix not convertible - 3 3 3 0.0	Thing Metrics Start of Current Cycle Server Tx Jitter Typitter Typitter Typitter Typitter Calculation Typitter Typi
Selected Node Attributes	Value ns=21=Uin54 2 ns=2Uint64 Uint64 Uint64 0 1638 1539 12	Trining Metrics Trining Metrics Start of Current Cycle Server Tx Calculation Server Tx Calculation Trinin_exec 2350 Client Tx Volvioid Iterations + 12116 Volvioid Iterations + 12

Figure 4: Example of an HMI Dashboard for the OPC UA Client

Machine Controller Verification

To run the machine controller app, use PLCopen sample applications. The PLCopen sample applications are isochronous. Following are the steps to validate the Industrial Machine Controller use case.

1. To interact with the application, start by connecting to the controller VM host terminal console using SSH:

ssh root@ host-for-vms-1

2. Then connect to the controller VM-1 terminal console using SSH:

ssh root@vm-1

3. In case the EtherCAT servo is not available, run the following application:

sudo /opt/plcopen/multi-axis-monitor

- 4. In case the EtherCAT servo is ready, use these steps to run the machine controller:
 - # sudo /opt/plcopen/ethercat sl

sudo taskset -c 1 /opt/plcopen/six_rtmotion_demo -n /opt/plcopen/inovance_six_1ms.xml -i 1000

Additional feature verification tests can be found here:

https://github.com/intel/container-experience-kits/tree/master/docs/eci_guide.md https://github.com/intel/container-experience-kits/tree/master/validation/verification-manual

Reference Documentation

The <u>Network and Edge Bare Metal Reference System Architecture User Guide</u> provides information and a full set of installation instructions for a BMRA.

The <u>Network and Edge Virtual Machine Reference System Architecture User Guide</u> provides information and installation instructions for a VMRA.

The <u>Network and Edge Reference System Architectures Portfolio User Manual</u> provides additional information for the Reference System including a complete list of reference documents.

Other collaterals, including technical guides and solution briefs that explain in detail the technologies enabled in the Reference Systems, are available in the following location: <u>Network & Edge Platform Experience Kits</u>.

Document Revision History

REVISION	DATE	DESCRIPTION
001	July 2023	Initial release.
002	October 2023	Updated BMRA version to 23.10. Added the industrial machine controller feature deployment using a mixed deployment of BMRA and VMRA.
003	October 2023	Added the industrial machine controller configuration and deployment using VMRA.



No product or component can be absolutely secure.

Intel technologies may require enabled hardware, software, or service activation.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

1023/DN/WIT/PDF

784548-003US