ROMA Repair

Problem Statement

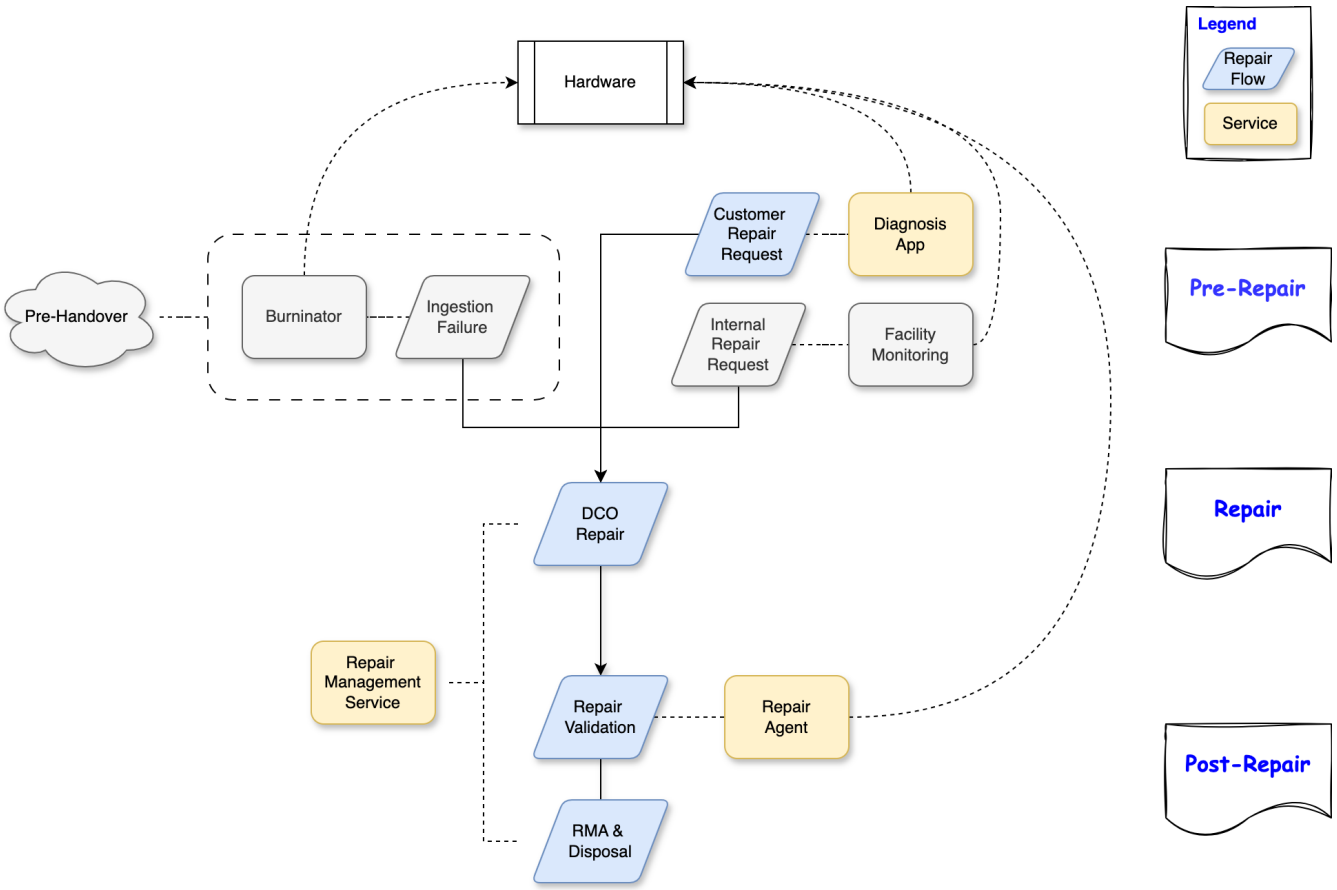
As part of the Project ROMA design, OCI will be physically disconnected from the ROMA customer enclave after handover, resulting in no direct network access. This makes the hardware diagnosis and repair process different from the commercial offering, where OCI usually monitors and manages hardware directly. The ROMA enclave primarily operates GB200 and E5 racks, due to the complexity of these new hardwares, a typical repair requires comprehensive diagnostic support to thoroughly examine each component and FRU within the rack, pinpoint the issue, and guide the DCO repair process. Throughout the process, one of the biggest challenges is coordinating with ROMA to access their fleet. Since OCI does not have direct control over, the ROMA customer must actively participate in certain actions to complete the story.

This page outlines our vision and design for orchestrating this type of repair process between ROMA and OCI, it is derived from the original designs of: Repair Agent - Low level Design and RMC Change Management Low Level Design.

Overview

From very high level, we envision an ideal ROMA repair world, will go through three stages: **Pre-Repair, Repair and Post-Repair**. Note that Pre-Repair involves issue detection and repair request originating, which typically comes from three sources: ingestion failures from burn-in process (pre-handover), customer repair requests from customer diagnosis (post-handover), and internal repair requests from facility monitoring (post-handover). This doc focuses on the customer path, while ingestion failure and internal repair are beyond its scope. Here we will walk through a typical customer repair process, as below:

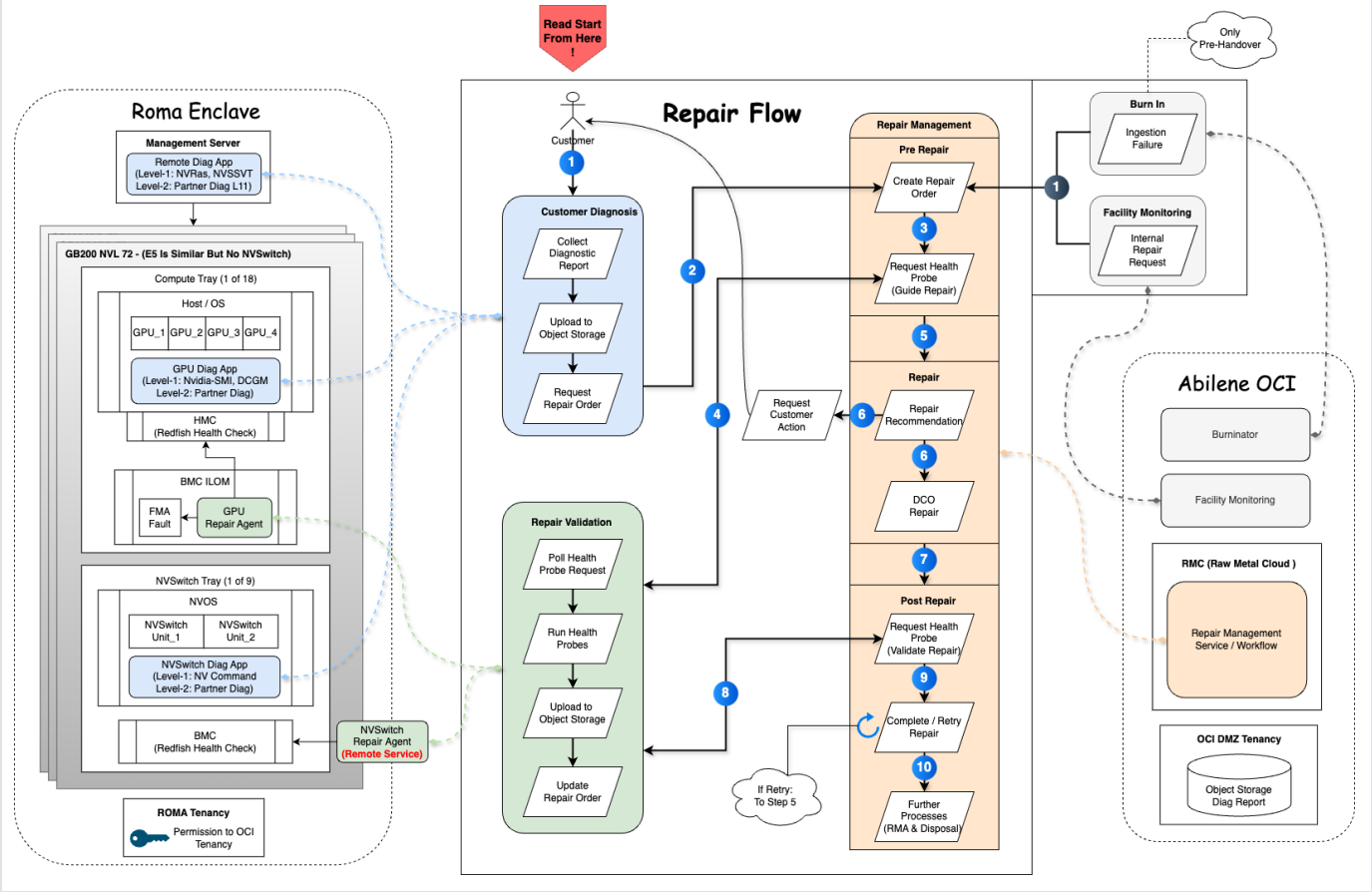
The process begins with customers using the OCI provided Diagnosis App to generate a comprehensive hardware health report through various diagnostic utilities. Once the diagnostic data is collected, customers submit a repair order via the public APIs of the OCI Repair Management Service, which orchestrates the data center repair workflow. Upon registration, the repair order becomes visible in the repair management system, allowing the DCO to manage it by following predefined repair steps. Once repairs are complete, validation is performed by the OCI Repair Agent, which probes the host ILOM and NVSwitch, conducting a sanity health check via Redfish and FMA calls. After all, there are additional disposal and RMA steps for faulty hardware, marking completion of the repair.



Architecture

A comprehensive ROMA repair architecture is outlined as below: Note the Diagnosis App and the Repair Agent are new softwares that need to be developed, the Repair Management would be introduced as a workflow to be implemented within the existing Abilene RMC (Raw Metal Cloud) system.

ROMA Repair Architecture



Below is a textual representation of the architecture, which also reflects the User Experience

1). The ROMA customer suspects hardware failure, and runs the Diagnosis App OCI provided to trigger a repair order.

- The agreement states that the customer will isolate their hardware from their production workload before initiating an OCI repair. OCI repair assumes that all actions performed on the host are safe, have the customer's permission, and will not cause any disruptive impact.

1). An ingestion failure, or a internal center repair request (i.e., internal power issue) can also trigger a repair order.

2). The Diagnosis App collects and parse the comprehensive diagnostic report from various underlying utilities, uploads the results to object storage, and requests OCI to create a repair order.

- Types of diagnostic utilities :
  - GPU Diag: runs NVIDIA-SMI, DCGM and Partner-Diag (Compute tray mode) to collect comprehensive GPU card report.
  - NVSwitch Diag: runs Partner-Diag (NVSwitch tray mode) and NVOS system commands to collect comprehensive NVSwitch and NVLink report.
  - Remote Diag: runs NVRasTool, NVSSVT, Partner-Diag (L11 rack mode) to collect rack level information such as supplement thermal and power coverage for non-NVIDIA products.
- Types of repair order:
  - GPU Server Repair: requests to repair the GPU server node (if GB200).
  - E5 Server Repair: requests to repair the E5 server (if E5).
  - NVSwitch Repair: requests to repair NVSwitch.
  - Disk Repair: requests to replace NVMe SSDs.

3). The repair management in RMC receives the repair order, and initiates an health probe request to validate the hardware correctness, in order to collect additional OCI standard fault data to assist in the repair process.

4). The Repair Agent deployed in ILOM listens and gets the probe request, runs the underlying health probes, uploads the results to object storage, and updates the repair order with the vetted health status.

- Types of Repair Agent:
  - GPU Repair Agent: Runs probes from ILOM FMA module for host hardware fault detection; runs probes from GPU HMC redfish APIs for NVIDIA embedded GPU health data.
  - NVSwitch Repair Agent: Run probes from NVSwitch BMC redfish APIs for NVIDIA authorized NVSwitch health data.

5). Next, the health data is pushed to the repair recommendation component, which analyzes the diagnostic report and maps the fault code to generate a repair recipe.

- In our future plans, we anticipate repair recommendation is a ML based health data analysis engine, and who wires up with GRT and TRS to seamlessly generate repair recipe based on detected faults.

6). The DCO personel follows the repair recipe to fix the issue, or replace faulty components.

- If replacement, OCI needs to upgrade the firmware to the latest for the new units.

6). The repair recipe may indicate that the issue is not a valid repair case (false positive), request additional diagnostic data (e.g., a Level-2 run of Part-Diag), or suggest a resolution such as a customer-side reset or firmware upgrade. In such cases, the repair process will not proceed; instead, an attention request will be sent back to the customer for further action.

7). After the DCO repair operation, another internal health probe request is initiated to validate the system's status and ensure the repair was effective.

8). The step is identical to step 4

9). If the health probe returns a successful health check, the repair order will be closed and marked as complete. If the health probe indicates a failed health check, the process will be retried, returning to step 5.

10). Once the repair order is closed, a separate offline process manages hardware disposal or RMA.

- If an RMA is required, the returned hardware must be accompanied by detailed diagnostic data stored in object storage.

FAQ

Why a customer managed Diagnosis App is essential for GB200 in ROMA?

Several reasons: First, unlike traditional commercial regions where OCI provides built-in hardware monitoring, the GB200 hardware is super complex that demands a different approach where partners should follow NVIDIA's guidance to run a set of NVIDIA provided utilities to gather comprehensive GPU reports. Since OCI is entirely disconnected from ROMA infrastructure after handover thus OCI can't log in and run NVIDIA utilities. Instead, ROMA customers are responsible for running these utilities and sending results back to OCI when troubleshooting or repairs are needed. Second, NVIDIA's diagnostic utilities are not suited for continuous background monitoring. These tools are resource-intensive and disruptive, and should be triggered only on-demand. Third, sometimes hardware vendor like Ingrasys would require these NVIDIA reports to justify RMA, we have to rely on customers to collect and provide these reports on our behalf.

Why do we still need Repair Agent in additional to Diagnosis App?

Repair Agent plays different roles in pre-repair and post-repair, and both are necessary:

In pre-repair, NVIDIA diagnosis report sometime is not sufficient to guide DCO repair, as we need addition OCI standard / recognized fault codes where DCO can map to repair recipes. The fault code will be fetched by Repair Agent scanning through ILOM FMA module and NVIDIA HMV Redfish health APIs. Moreover, for extreme cases that ROMA can't get any diagnosis data when the host is dead entirely, since Repair Agent sits in the standalone hardware (ILOM, Nvidia HMC) that may still be alive, and can be relied on to pass back bare minimum health info. For post-repair, OCI needs a seamless process to validate the repair, as at this stage both ROMA customer and OCI are disconnected from the host, Repair Agent will be the only source to vet the health correctness.

In terms of performance impact and customer agreement, Repair Agent sits in ILOM (for Compute tray) or remotely (for NVSwitch) so it has minimum disruptions to customer workloads. ROMA customers are aligned of embedding these solutions into their fleet.

How do DCO personnel determine the correct repair process from a diagnostic report?

The diagnostic report, generated by the Diagnosis App and Repair Agent, includes a "Fault Code." The Repair Recommendation component analyzes this report in depth and integrates with various RHS repair systems, such as GRT, to query and retrieve the appropriate repair recipe for the customer. In a manual process, DCO personnel can extract the fault code directly from the report and manually input it into GRT to find the corresponding repair instructions.

There are two sources of fault codes: Fault code derived from the ILOM FMA module; Fault code parsed / aggregated from the comprehensive Nvidia diagnostic report, similar to HPC Plugin code.

In Step 6, who ensures that the newly replaced hardware has the latest firmware installed and is operational?

There is a OCI process called - Warminator, will download the latest firmware to be installed into replaced hardware (if hosts), and make sure the hosts are reset to default state and is operational (power on).

During ingestion, does Burinator use the same underlying Nvidia validation mechanism as the Diagnosis App?

Yes and no. While Burinator and the Diagnosis App share some diagnostic logic, such as running Partner-Diag to stress-test verify hardware integrity, but Burinator is a far more intensive process. It runs over multiple days, applying significant stress to the hardware to uncover any pre-delivery issues. Burinator utilizes 10+ software tools from Compute CPV orchestration software to ensure every component is thoroughly vetted. In hardware industry, hardware validation before delivery is inherently more rigorous and meticulous than post-shipping health checks. Thus they are different.

Open Questions

1. We need to determine the correct repair procedure when the FMA fault code returns nothing (i.e., DCO can't receive any data from GRT because no fault code), however, the customer's diagnosis app still reports an unhealthy system. - What should we do for this type of repair? I actually foresee this won't be the edge cases but it would be the majority

2. We should establish a process ensuring customers perform due diligence - such as resetting the host or upgrading firmware - before creating repair orders. - Any systematic level of change you think we should do? Or we just need to define this as a process and deliver a runbook to customer?
3. Who or what service should be responsible for ensuring that replacement hardware is operational, has the latest firmware loaded into ILOM, is powered on, and is network-connected? In commercial regions, this is handled through the HOPS provisioning process and CPV validation. However, for ROMA repairs, there is currently no established process. (Step 7). Should OCI own this process (I've heard the term "Warminator"), or should ROMA customers be responsible for ensuring the host is provisioned, has the latest ILOM firmware, etc.?

MVP Delivery

Rome wasn't built in a day, and similarly, while most of the "colored" components in the diagram will eventually be implemented, our first customer release will come with some compromises and a slightly reduced scope:

- No Repair Recommendation: For the initial release, the DCO will manually retrieve repair recipes by looking up fault codes in the Guided Resolution Tool (GRT) via the UI. Since the repair recommendation is envisioned as a sophisticated ML-based analysis platform, requiring long-term development.
- No Repair Agent for Pre-Repair (Arch diagram: Step 6): It will be temporally / alternatively achieved by customers proactively running the Diagnosis App through ILOM to collect FMA fault codes from Step 2, and send it back to OCI. This is due to the complexity of modifying the repair workflow to synchronously wait for Repair Agent coordination after order creation.

Business Assumptions

- ROMA customers have certain level of expertise in managing basic Python applications (Diagnosis App), and debugging / configuring the Nvidia environment when required.
- Before submitting a repair order, ROMA customers will isolate the hardware to ensure that OCI's repair process is both permitted and non-disruptive.
- ROMA and OCI follow a manual protocol and communication channel to handle edge-case scenarios. This includes facilitating back-and-forth communication for gathering additional diagnostic results or requesting specific customer actions.

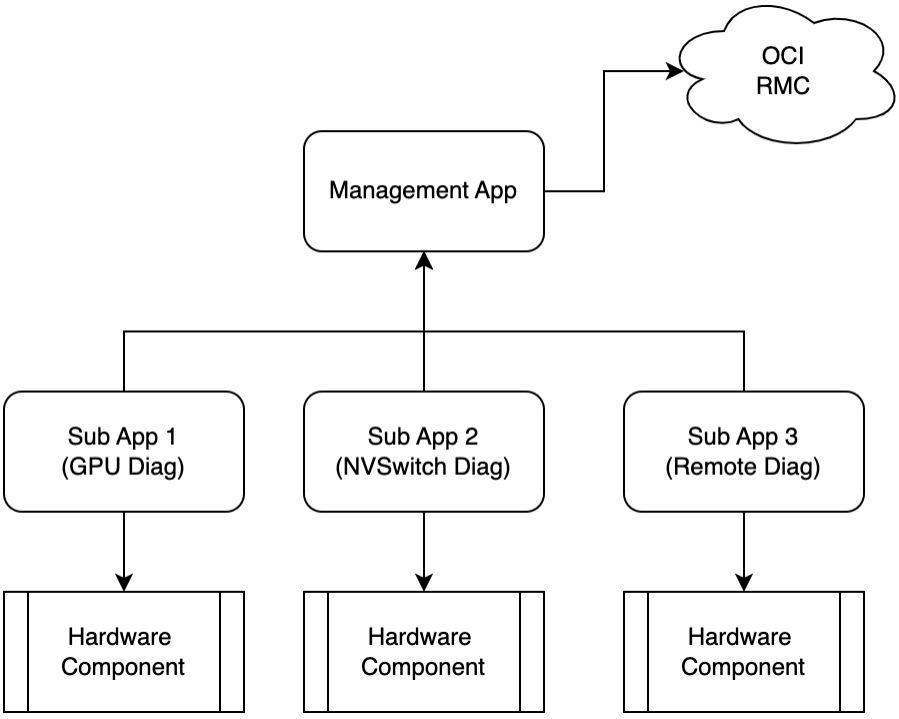
Disclaimer

Below section is still in progress

Design

Diagnosis App

The Diagnosis App is a set of Python packages delivered by OCI to ROMA customers for running hardware diagnostics on their GB200 and E5 servers. It consists of multiple **Sub-Applications** that run associated diagnostic utilities, along with a **Management Application** that collects and manages all diagnostic reports from the sub-applications.



Management Application

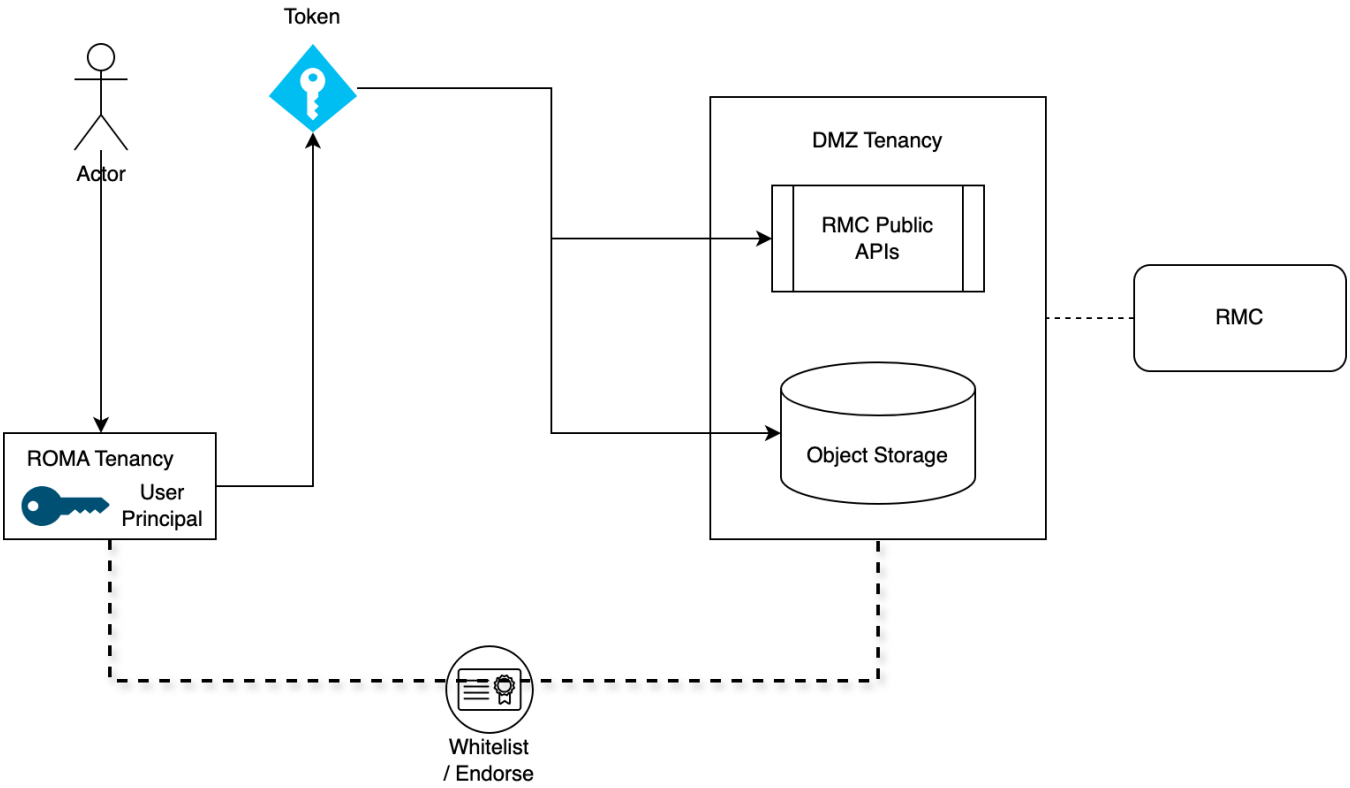
This app runs on a management node or control system, and executes below flows

- Executes diagnostic tools on each sub application.
- Collects and aggregates diagnostic reports from each sub application.
- Uploads raw logs to OCI object storage.
- Parses and generates the compact version of diagnosis report.
- Calls the RMC API to create repair order with attached raw logs object storage URI and compact diagnosis report.

Note that steps 1 and 2 will be largely manual for the MVP delivery. We anticipate that ROMA customers will have the expertise to install the sub-diagnostic apps in the designated locations, execute them, and retrieve the results for the management application.

Auth

The management app operates on the ROMA's public internet and uses a user principal token for authentication when interacting with OCI object storage and RMC APIs. To implement this securely, we establish two separate tenancies: 1. ROMA tenancy - fully owned and managed by ROMA customer; 2. DMZ tenancy - fully controlled by OCI, acting as the broker for data exchange. ROMA will have no direct access to DMZ tenancy as they only configure their API key access in their dedicate tenancy. OCI will have no direct access to ROMA tenancy to ensure complete security and isolation for the customer. Cross-tenancy communication is established, by granting the necessary permissions through Identity endorse and whitelist.



DMZ whitelist policy example:

```
define tenancy customer as ocid1.tenancy.oc1..aaaaaaaa2fydp7bskn56umypp5jb2i3petcin74lbgzmkr4y467hmysc5yuq
admit any-user of tenancy customer to read objectstorage-namespace in COMPARTMENT id ocid1.tenancy.oc1..
aaaaaaaa2zhxeq4vwgtevie7657dh5ak4ega4sjgokeigshe2legco5ornfa
admit any-user of tenancy customer to read object-family in COMPARTMENT id ocid1.tenancy.oc1..
aaaaaaaa2zhxeq4vwgtevie7657dh5ak4ega4sjgokeigshe2legco5ornfa where any { target.bucket.name = 'rmc-export-
bucket', target.bucket.name = 'rmc-export-schema-bucket' }
admit any-user of tenancy customer to manage object-family in COMPARTMENT id ocid1.tenancy.oc1..
aaaaaaaa2zhxeq4vwgtevie7657dh5ak4ega4sjgokeigshe2legco5ornfa where any { target.bucket.name = 'rmc-repair-order-
diagnostics' }
```

ROMA endorse policy example:

```
Define tenancy DMZRoma as ocid1.tenancy.oc1..aaaaaaaa2zhxeq4vwgtevie7657dh5ak4ega4sjgokeigshe2legco5ornfa
Endorse any-user to read object-family in tenancy DMZRoma where any { target.bucket.name = 'rmc-export-bucket',
target.bucket.name = 'rmc-export-schema-bucket'}
Endorse any-user to manage object-family in tenancy DMZRoma where any { target.bucket.name = 'rmc-repair-order-
diagnostics'}
```

User principal config example:

```
[DEFAULT]
user=<your_user_ocid>
fingerprint=<your_fingerprint>
tenancy=<your_tenancy_ocid>
region=<your_region>
key_file=path/to/your/private_key.pem
namespace=<your_namespace>
bucket_name=<your_bucket>
default_endpoint=<your-api-endpoint>
```

Sub Applications

The sub applications act as a wrapper around various system diagnostic utilities. It simply executes the underlying subprocess commands and stores the output in local files, which are later retrieved and passed to the management app.

- **GPU Diag:** Diagnoses GPU hardware and driver issues. It runs NVIDIA-SMI, DCGM and Partner Diag (Compute tray mode) to collect comprehensive GPU card report. - Installed at the host OS within GPU Computer tray
- **NVSwitch Diag:** Checks NVSwitch connectivity and performance. It runs Partner Diag (NVSwitch tray mode) and NVOS system commands to collect comprehensive NVSwitch and NVLink report. - Installed at the NVOS within NVSwitch.
- **Remote Diag:** Performs remote system diagnostics for networked nodes / racks. It runs NVRasTool, NVSSVT, Partner Diag (L11 rack mode) to collect rack level information such as supplement thermal and power coverage for non-NVIDIA products. - Installed at a remote jump box or management server.

Two Levels Diagnosis

All sub-diagnostic applications should support two levels of diagnosis:

- **Level-1 Diagnosis**: Runs by "default". A quick diagnostic process that runs within 5 minutes, gathering essential hardware reports. It is designed to cover approximately 95% of potential hardware issues.
- **Level-2 Diagnosis**: Runs as "optional". A comprehensive, extended diagnostic process that deeply stresses the system to collect detailed test results. This process can run for days and is disabled by default. It is used only when instantiated by the customer or when OCI requires additional data to support the repair process (Arch diagram: Step 6).

Sub App	Level-1	Level-2
GPU Diag	Nvidia-SMI, DCGM	Partner-Diag (Compute Tray Mode)
NVSWitch	NVOS CLI	Partner-Diag (NVSwitch Mode)
Remote Diag	NVRasTool, NVSSVT	Partner-Diag (L11 rack Mode)

Diagnostic Utils

See Appendix.1 for details

Diagnostic Report

The diagnostic report consists of two parts: **Raw Logs** (full detailed output from sub diag apps) and **Compact Report** (summary for management and repair requests). The raw logs will be generated directly by each sub app from running underlying system diagnostic utilities and stored in their original, unaltered format. The compact report will be aggregated by the local parser in the management app and presented in the following format:

```
Report Metadata:
  Report ID: <UUID>
  Generated Timestamp: <YYYY-MM-DD HH:MM:SS UTC>
  Hostname: <Server Name>
  System Type: <GB200 / E5>
  Diagnosis Level: <Level-1 / Level-2>
  Executed By: <Auto / Manual>
  Report Status: <PASS / WARNING / FAIL>
  Uploaded Logs URI: <OCI Object Storage Link>

Summary:
  Overall Status: <PASS / WARNING / FAIL>
  Issues Detected: <Yes / No>
  Critical Errors:
    - Component: <GPU / NVSwitch / Rack>
    - Issue Type: <Hardware Failure / Performance Degradation / Thermal Alert>
    - Error Code: <Error Identifier>

Component Diagnostics:
  GPU:
    - Model: <GB200 / GB300>
    - Serial Number: <Serial Number>
    - Driver Version: <Version>
    - Memory Errors: <Yes / No>
    - Temperature: <XX°C>
    - Power Usage: <XX W>
    - Overall Status: <PASS / FAIL>

  NVSwitch:
    - Serial Number: <Serial Number>
    - Firmware Version: <Version>
    - Link Errors: <Yes / No>
    - Bandwidth: <Gbps>
    - Latency: <ms>
    - Overall Status: <PASS / FAIL>

  Rack:
    - PSU Health: <OK / Warning / Critical>
    - Cooling Efficiency: <XX%>
    - Network Connectivity: <OK / Issues Detected>
    - Overall Status: <PASS / FAIL>

FMA Fault Code:
    - Fault Code: <code>
    - Suggested Actions: <Restart / Replace / Further Testing>
    - OCI Runbook: <Link>
```

Repair Agent

(IN-PROGRESS)

1. Application framework - main business logic of the repair agent
2. What underlying probes repair agent will call
3. Health check coverage
4. Format of the repair agent health check data

Repair Management

(IN-PROGRESS)

1. Repair order type
2. APIs

Release Pipeline

(IN-PROGRESS)

ROMA is a specialized region where customers have full ownership of their fleet, with strict security measures in place to prevent malicious attacks. Meanwhile, OCI operates the data center and maintains a highly cautious stance to prevent potential risks from customer supplied code. As outlined, Diagnosis App and Repair Agent are Python based tools that we deliver to customers. From our agreement with ROMA, we must establish a secure delivery pipeline that ensures protection on both ends: OCI must verify that the packages delivered to ROMA are recognized and signed, while ROMA must confirm that the received packages have been inspected and approved. Both sides must guarantee verification before deployment. This mutual understanding has already been aligned with ROMA customers.

The following section presents the high level design of the release pipeline. It assumes that ROMA places trust in specific segments of OCI's internal pipeline, relying on its security policies and internal network safeguards to protect against hacking and external threats.

Scaling

(IN-PROGRESS)

Resiliency & Disaster Recovery

(IN-PROGRESS)

Engineering Plan

WS2 Detailed Engineering Plan

Appendix

1). Diagnostic Capability

GPU Diag

	Area	Component Diagnosed	Command	Details
Nvidia-SMI	GPU Utilization & Performance	GPU Core, Memory	$nvidia-smi dmon	Monitors real-time GPU, memory, and power utilization
	Temperature & Fan Speed	Thermal System, Fans	$nvidia-smi --query-gpu=temperature.gpu,fan.speed --format=csv	Fetches current GPU temperature and fan speed
	ECC Error Reporting	GPU Memory	$nvidia-smi --query-gpu=memory.ecc.errors --format=csv	Reports single/double-bit ECC errors in GPU memory
	Power & Voltage Monitoring	Power Supply, VRMs	$nvidia-smi --query-gpu=power.draw,voltage --format=csv	Displays power consumption and voltage levels
	Process & Compute Workloads	GPU Scheduler	$nvidia-smi pmon	Shows active GPU processes and compute workloads
DCGM	Health Checks & Fault Detection	GPU Core, Memory, NVLink	$dcgmi diag -r 1	Runs quick diagnostic tests to detect hardware faults
	Tensor Core & Compute Tests	Tensor Cores, FP Units	$dcgmi diag -r 4	Evaluates deep learning performance and floating-point operations
	NVLink & PCIe Bandwidth Tests	Interconnects, PCIe Bus	$dcgmi diag -r 3	Checks NVLink and PCIe bandwidth health
	Stress Tests & Stability Checks	GPU Core, Cooling System	$dcgmi diag -r 3	Simulates high workloads to assess thermal stability
	System-wide Telemetry & Logging	Entire GPU System	$nvidia-smi pmon	Collects historical telemetry for failure prediction

	Area	Duration	Test	Details
Partner-Diag (Compute Tray Mode)	Inventory	40 sec	System-level check of components against expected versions.	Fails if the firmware version does not mathc, or the hardware is missing.
	TegraCpu	12 min	Performs CPU diagnostics testing.	Fails if the CPU operation is not stable.
	TegraMemory	1 min	Saturated the system memory bus with concurrent data traffic generated by High Speed Scrubbing and CPU. This requires a MODS secure partition.	Fails if unable to perform read/write /allocate transactions on all memory channels.

CpuMemorySweep	15 min	Perform reads and writes and correctness checks for CPU memory. This requires a MODS secure partition.	Fails if unable to perform read/write /allocate transactions on all memory channels.
TegraClink	10 min	CPU-CPU NVLink bandwidth, eye diagram tests.	Fails if the link qualify thresholds are not met.
Gpustress	7 min	GPU Stress Tests.	Fails if there is CRC miscompare during computation.
Gpumem	3 min	GPU memory and interface (FBIO) tests.	Fails if the GPU memory is unstable.
Pcie	10.5 min	GPU PCIE Bandwidth, speed switching, and eye diagram tests.	Fails if the GPU PCIe connection is unstable or cannot achieve required functions.
C2C	1 min	CPU-GPU C2C NVLink.	
CPUVDD_PowerStress	30 min	Performs high power stress on the CPUs.	Fails if the thermal limits are exceeded, or the CPU operation is unstable.
CpuGpuConstPower	11 min	Stress power on system components (CPU, GPU).	
Connectivity	8 min	Validates the electrical quality of NVLinks and PCIE link speeds/width match the POR.	Fails if the nvlink connection detects errors or is unstable.
Nvlink	25 min	GPU-GPU NVLink bandwidth, eye diagram tests.	Fails if the link qualify thresholds are not met.
DimmStress	6 min	Stresses CPU DRAM.	
gpufielddiag	3 hr	GPU field diagnostics test suite.	

NVSwitch Diag

(IN-PROGRESS)

Remote Diag

(IN-PROGRESS)