75th International Astronautical Congress (IAC), Milan, Italy, 14-18 October 2024.

IAC-24,B3,7,2,x89114

The Columbus Data Management Infrastructure (CDMI): A cloud above the sky on the ISS

Jan Tekülve[a]*, Alexander Balgavý[b], Christian Altenschmidt[a], Catriona Bruce[d], Markus Daugs[a], Jens Ender[a], Christine Gläßer[c], Nora Newie[d], Nicole Roshardt[e], Loric Vandentempel[b]

[a] *CGI Deutschland B.V. & Co. KG., 44803 Bochum, Germany,* jan.tekuelve@cgi.com, christian.altenschmidt@cgi.com, markus.daugs@cgi.com, jens.ender@cgi.com
[b] *Space Applications Services NV/SA, 1932 Sint-Stevens-Woluwe (Brussels Area), Belgium,* alexander.balgavy@spaceapplications.com, loric.vandentempel@spaceapplications.com
[c] *CGI Deutschland B.V. & Co. KG., 64295 Darmstadt, Germany*, christine.glaesser@cgi.com
[d] *GMV GmbH, 82205 Gilching, Germany,* catriona.bruce@gmv.com, nnewie@gmv.com
[e] *European Space Agency (ESA) ESTEC, 2201 AZ Noordwijk, Netherlands*, nicole.roshardt@esa.int
* Corresponding Author

Abstract

Ease of communication between payloads and their ground users is critical to the effectiveness of the International Space Station (ISS) Columbus module as a platform for commercial and ESA payloads. The Multi-Purpose Communications Computer (MPCC) has made IP-based communication between Columbus and the Columbus Control Center (Col-CC) possible through both the NASA Ku-IPS service and the Columbus Ka-band service communication channels. This paper introduces the Columbus Data Management Infrastructure (CDMI), a replacement for MPCC aiming to enhance existing capabilities of MPCC by introducing new features and improving system resilience on both the hardware and software level.

CDMI comprises four single-board computers to be installed on board Columbus, along with a set of virtual machines hosted at Col-CC. The CDMI computers consist of CompactPCI COTS elements, which are an industrial standard providing robustness, scalability, hot swapping, versatility, and long lifecycle support. Their setup is particularly designed to deal with space-specific challenges of power limitations, cooling methods, and radiation susceptibility. They host CDMI's flight services as a cluster of hypervisors based on the Proxmox Virtual Environment (PVE). The PVE cluster enables a modular and redundant setup by containerizing services and replicating storage between nodes. The core services of CDMI ensure the continued functionality previously provided by MPCC. The IP Communications Service abstracts from the underlying Ka- and Ku-band communication channels, while the File Exchange Service provides fast and resilient data transfer across both communication channels. In addition, each payload will have its own redundant storage area within the cluster. This storage area is integrated with a flight SFTP server and Nextcloud on the ground making data easily accessible. Moreover, the use of a hypervisor enables payload users to establish personal virtual machines within CDMI. This has the potential to streamline flight data processing and minimize the requirement for data downlinks.

The design of the ground component follows the principle of distributing services across dedicated virtual machines with a general preference for COTS software whenever possible. Monitoring services include Zabbix and Yamcs, which provide operators and payload users individual insights into CDMI's status. Operators can modify the system configuration using Ansible via the AnsibleForms web interface. Furthermore, Ansible is crucial for CDMI's deployment process, whose infrastructure is entirely represented as code. This enables fast release cycles by including security testing in our DevSecOps process. CDMI's services can be effortlessly updated and configured using Ansible both on the ground and in flight.

Keywords: ISS, MPCC, CDMI, DevSecOps, Infrastructure-as-Code, Ku-IPS

Acronyms/Abbreviations

CDMI	Columbus Data Management Infrastructure	**CI**	Continuous Integration
CDMI-F	CDMI Flight segment	**COTS**	Commercial Off-The-Shelf
CDMI-G	CDMI Ground segment	**Col-CC**	Columbus Control Center
CD	Continuous Deployment	**DaSS**	Data Services Subsystem
CFDP	CCSDS File Delivery Protocol	**DevSecOps**	Development, Security, and Operations

EAC	European Astronaut Centre
EDRS	European Data Relay Satellite
ESA	European Space Agency
FES	File Exchange Service
ISS	International Space Station
Ku-IPS	Ku-band Internet Protocol Service
LM	Link Manager
LXC	Linux Container
MPCC	Multi-Purpose Communication Computer
NASA	National Aeronautics and Space Administration
ORU	On-board Replaceable Unit
PCDS	Power Conditioning and Distribution System
PI	Principal Investigator
PVE	Proxmox Virtual Environment
SCAO	Starboard Cone Aft Overhead
VM	Virtual Machine
VPU	Virtual Processing Unit

1. Introduction

The International Space Station (ISS), established through international cooperation between various space agencies, is the biggest man-made laboratory flying in space. One part of it is the ESA Columbus module, which focuses on conducting experiments in the unique conditions of zero gravity [1, 2, 3, 4, 5]. Data generated by such experiments needs to be downlinked reliably and in a timely manner. In addition, experiment software and data collection regimes may change over time, thus requiring experimenters to update their setup from the ground. To meet those needs, a direct Ku-band Internet Protocol Service (Ku-IPS) link, provided and maintained by NASA, was introduced alongside a gateway for European experimenters called Multi-Purpose Communications Computer (MPCC) [6, 7]. Since 2015, MPCC provides a straightforward approach for European experimenters on the ground to interact with their payload deployed in the ESA Columbus module. On board, MPCC consists of the European IP Communication Laptop (EICL) and the Columbus Monitoring and Administration Unit (CMAU), while their counterparts on the ground are hosted by the Columbus Control Center (Col-CC) in Oberpfaffenhofen, Germany. MPCC supports up to ten connected payloads, each paired with a designated Principal Investigator (PI) user account. In addition to the Ku-IPS link, a Ka-band terminal on Columbus has been installed to provide a direct communication path between Col-CC and the Columbus module that operates independently of shared communication resources.

Although MPCC has proven to be a helpful asset to ensure direct communication between PIs and their payload, the system may not be prepared for required future extensions. Internal predictions show that future payload development is geared increasingly towards IP-based communication, rather than making use of legacy payload interfaces of the Columbus Data Management System (COL-DMS). In order to adapt to the increased load that the integration of future payloads may bring, it is advisable to place additional emphasis on the following factors: (1) Increasing reliability through redundant hardware and software; (2) Enabling extensibility with rapid software release cycles; (3) Improving maintainability by increasing the modularity of the system.

To allow for future expansion of Columbus through additional payloads with increased data link and storage requirements and considering the above factors as primary drivers of the design, it was decided to replace MPCC with the Columbus Data Management Infrastructure (CDMI).

1.1 The CDMI Concept

CDMI is envisioned to provide a reliable and modular platform to integrate payloads into the Columbus module using modern virtualization technology, with a launch planned for the fall of 2025. CDMI's flight component, CDMI-F, comprises four single-board computers, forming three redundant virtual processing units (VPU). Each VPU serves as a hypervisor, running the Proxmox Virtual Environment in a cluster configuration (see Section 2.2).

From the hardware perspective, CDMI-F makes extensive use of CompactPCI COTS elements allowing it to benefit from modern industry standards and cost-efficient redundant parts (see Section 2.3). The approach of using COTS hardware in space aligns with the latest trends in the industry, for example, Hewlett Packard Enterprise's Spaceborne computer [8, 9]. CDMI-F is fully conduction cooled via coldplates, rejecting heat through the thermal control system in Columbus. CDMI-F's three VPUs are installed on top of the starboard cone aft overhead (SCAO) coldplate of the Columbus module and do not protrude into the cabin (see Figure 1).

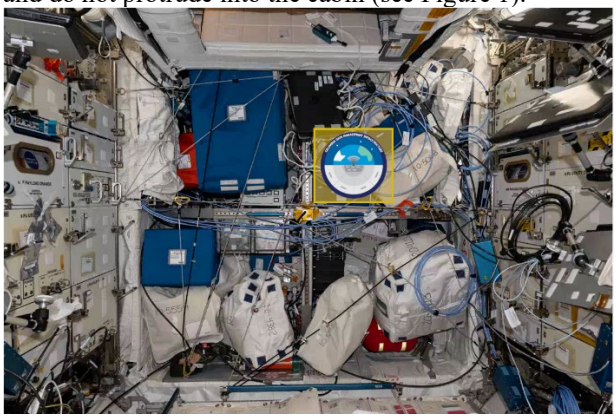

Figure 1: Planned hardware location of CDMI on the starboard side of the Columbus module. ©ESA, adapted with permission.

Additionally, replacement parts for COTS hardware will be shipped with CDMI to mitigate potential radiation damage.

In terms of software, CDMI follows a similar approach, whereby software processes are conceptualized as independent modular services and are realized through COTS software whenever possible. Through its hypervisor architecture, CDMI-F can realize both legacy MPCC functionality and new software components as isolated containerized services (see Section 3). Functions of CDMI will include enabling of payload communications, file transfers, and file storage, but also monitoring of Columbus hardware and provisioning of standard network services (e.g., NTP and DNS). CDMI's ground component, CDMI-G, is distributed across multiple Virtual Machines (VMs) hosted at Col-CC. CDMI-G contains services to monitor and control CDMI as well as the ground counterparts to CDMI-F's services. Ground services can be accessed through web portals that offer custom interfaces for operators and PIs.

The design of CDMI seeks to adhere to DevSecOps principles with the objective of facilitating the implementation of robust and fast software release cycles. To this end, automated testing and validation are conducted using identical setups on Earth, encompassing both digital and physical environments. Continuous integration and deployment (CI/CD) ensures continuous improvement and rapid prototyping to address new challenges. The infrastructure is configurable, allowing rapid adaptation of both software and hardware. CDMI is designed around 'file-based operations', meaning that all configuration changes and therefore all control of the behavior and state of CDMI is achieved through changes in configuration files. Ultimately the goal of our approach is to operate CDMI like a terrestrial IT system, using contemporary monitoring and configuration tools.

Section 3 explains the CDMI services and its virtualization infrastructure including the underlying DevSecOps concept in more detail. Section 4 highlights preliminary results achieved with CDMI's development prototype. The final two sections discuss and summarize the impact of CDMI on operators and PIs.

2. Material and Methods

The following section provides an overview of the most important open-source COTS software components used in CDMI and the hardware setup of CDMI-F. Additionally, it explains the unique nature of the two space-to-ground channels available to CDMI.

2.1 Monitoring and Configuration Software

CDMI configures its services through Ansible, monitors its services and hardware through Zabbix, and offers payload monitoring through Yamcs.

2.1.1 Ansible

Ansible [1] is an automation tool that facilitates configuration management, application deployment, and task automation across large-scale IT environments. Its main building block is the playbook, which declaratively defines desired system states and processes that are to be applied on a set of target hosts. Ansible operates without an agent, requiring no additional software on target hosts, which enhances security and simplifies setup. Communication with managed hosts is typically achieved over SSH, enabling seamless integration with diverse systems. Its idempotent design ensures repeated executions yield consistent results, minimizing the risk of unintended changes. Thus, Ansible is well suited to streamline CI/CD pipelines, fostering efficient and reliable system management and software delivery.

Playbooks are written in the YAML format; the backend of Ansible is implemented in Python. Its modular architecture supports extensibility, allowing users to create custom modules and plugins to meet specific needs. Ansible's ecosystem includes Ansible Galaxy, a platform for code exchange, further promoting collaboration and reuse within the community.

Ansible playbooks are typically executed via the command-line interface, while their parameters are defined through a set of YAML configuration files. The open-source project AnsibleForms [2] provides a web interface for playbook execution. Its form-based approach simplifies the process for users not familiar with the intricacies of Ansible syntax, as it eliminates the need to edit playbooks. Through AnsibleForms, it becomes possible to use Ansible for the deployment and run-time configuration of CDMI.

2.1.2 Zabbix

Zabbix[3] monitors the performance and availability of IT infrastructure, networks, and applications. It offers comprehensive monitoring capabilities, including real-time data collection, alerting, and graphing, making it suitable for complex environments. Zabbix supports multiple data collection methods, either via the Zabbix agent, or without an agent through, e.g., HTTP, SNMP or SSH. This enables Zabbix to centralize monitoring in diverse systems comprising a wide range of platforms and devices. Zabbix proxies may act as intermediate data collectors in remote parts of the IT infrastructure, thus adding resilience to the monitoring architecture.

[1] https://www.ansible.com/
[2] https://ansibleforms.com/
[3] https://www.zabbix.com/

One of Zabbix's key strengths is its highly configurable nature, allowing users to tailor the monitoring setup to their specific needs through custom scripts, templates, and user-defined parameters. Its web interfaces provide various visualization tools, including historical data analysis, customizable dashboards, and custom web pages. Monitored data can also be made accessible for further processing through the Zabbix API.

2.1.3 Yamcs

Yamcs [10] is a scalable framework for mission control created by Space Applications Services that allows commanding and monitoring of flight space missions such as satellites, spacecrafts, and payloads, and their related ground equipment and stations. Yamcs has already been successfully applied in MPCC payload missions [1]. The base software supports a variety of features including gathering telemetry, sending telecommands, creating alarms with custom triggers, data archiving, mission replaying, and file transfers. It is also equipped with an easy-to-use web interface, allowing operators to monitor and control their mission as they see fit. Yamcs also comes with the possibility of creating custom UI displays to showcase parameters and controls visually to provide an intuitive interface.

Its strength lies in its full customizability, both in configuration and in flexible plugin support. Based on Java, Yamcs can be configured to make use of external classes to further extend its capabilities for tailored needs, for example, custom data links, processors, and file transfers.

2.2 Flight Software Foundation

CDMI ensures the robustness of its modular flight infrastructure through the Proxmox Virtual Environment and the ZFS filesystem.

2.2.1 Proxmox

Proxmox Virtual Environment (PVE)[4] is an open-source platform based on Debian that has been designed for efficient and scalable management of virtualized infrastructures. It offers a comprehensive solution for the deployment and management of virtual machines and containers, integrating two virtualization methods, KVM (Kernel-based Virtual Machine) and LXC (Linux Containers). The former provides full virtualization with complete hardware abstraction, while the latter offers lightweight, container-based virtualization, which shares the host system's kernel, resulting in lower overhead and better performance.

Notably, PVE supports high availability clustering, enabling continuous operation of services in the event of hardware failures, as well as features such as live migration and integrated backup mechanisms. Additionally, it offers a wide range of robust storage solutions ranging from software-based RAID to shared network storage.

A virtualization platform, like Proxmox, allows multiple virtual hardware instances to run on a single physical hardware platform. Virtualization enables more effective resource management by allocating the necessary resources in a dynamic manner resulting in reduced operational costs and enhanced flexibility. Furthermore, as a scalable system, PVE can adapt to changing computation, storage or networking requirements both in the short and long term. Managing computation resources of services individually allows for the flexible reduction of required resources to a minimum in the face of unforeseen circumstances. Over time, the long-term benefits of virtualization include the capacity to cater to the different needs of a continuously changing user base.

2.2.2 Software RAID – ZFS

ZFS is a robust filesystem that uniquely integrates file system features with physical volume management, offering comprehensive data protection and efficiency. By binding multiple disks together in a software RAID, ZFS efficiently duplicates data, ensuring redundancy and resilience. Its RAID-Z technology allows systems to remain operational even if a disk fails. Additionally, it employs a self-healing scrubbing process, which detects and corrects data corruption caused by random bit flips, maintaining data integrity over time.

The file system uses a copy-on-write mechanism, meaning that when data is modified, ZFS writes the new data to a new location rather than overwriting the existing data. Only after the write is successful, ZFS updates the metadata to point to the new data block. This approach ensures that data is never left in an inconsistent state, even in the event of a system crash or power failure during the write process. Copy-on-write has the added benefit of allowing incremental snapshots to be created with minimal overhead - ideal for backups and versioning. However, these advanced features come with a trade-off: ZFS requires a significant amount of memory to perform optimally.

2.3 CDMI Hardware

The flight hardware system of CDMI is situated in the starboard cone of the Columbus module and is based on the CompactPCI Serial[5] standard architecture. To allow easy on-orbit maintenance and upgrades, the flight hardware uses state-of-the-art COTS electronics components where possible. It comprises three virtual

[4] https://www.proxmox.com/en/proxmox-virtual-environment/

[5] https://www.picmg.org/openstandards/compactpci/

processing units (VPUs), containing four single-board computers in total. Each VPU also includes its own power conditioning and distribution system (PCDS) and is equipped with a commercial CPU comprising eight 2.6 GHz cores and 64 GB of DDR4 error-correcting code (ECC) RAM. The VPUs host multiple smaller SSD drives, thereby providing redundant storage with relatively low power requirements. Each VPU is equipped with a dedicated switchboard, facilitating interconnections between the different VPUs and the Columbus network switches. These Ethernet switchboards operate autonomously from the CPU, thus ensuring redundant connectivity within the Columbus systems and payloads. Moreover, each VPU is furnished with Intel Active Management Technology (AMT) for remote administration and monitoring during system malfunctions. In non-nominal scenarios, it is possible to connect a portable monitor and keyboard for direct crew access.

The properties of each VPU are identical to allow for easier manufacturing and interchangeability, and all active VPU components are on-board replaceable units (ORUs). This also allows for a possible future upgrade of ORUs with more performant components. There is no interdependence between VPUs on the hardware level, achieving a system that is at least single failure tolerant, while using the available resources to build the most flexible, extensible, and failure-tolerant system under the given constraints.

Due to its location, the hardware is designed and configured to comply with rigorous power and thermal limitations. One of the main constraints is the available thermal budget and the need for a conduction-cooled solution. To this end, the CDMI hardware is cooled by an internal cold plate between two VPUs (in Figure 2, between VPU2 and VPU3). It also uses the SCAO cold plate, which is part of the starboard cross assembly in Columbus, to cool VPU1 (purple in Figure 2).

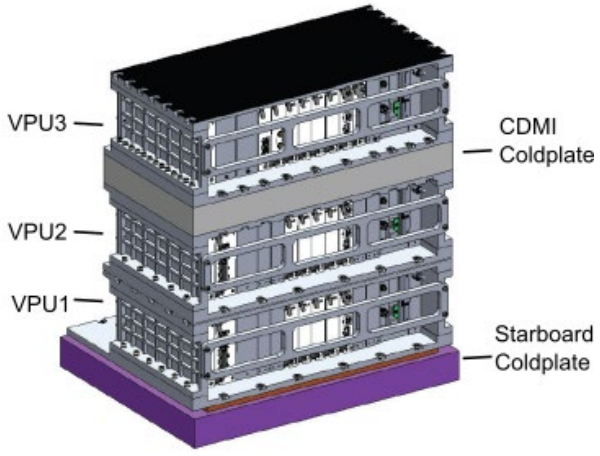

Figure 2: CDMI-F Hardware Components. ©Space Applications Services NV/SA, adapted with permission.

Moreover, while the PCDS for each VPU can provide up to 300W power output, the boards are selected such that even under maximum power, the VPU cannot consume more than its allocated thermal budget. Given the restrictions of the operational environment, the design of the hardware also incorporates the use of custom-developed components, including the internal cold plate, power harness, fluidic harness, and computer enclosures, which have been tailored to meet the specific power, thermal, and structural requirements.

2.4 Space-to-Ground Link

To enable communication between PIs on ground and their payloads in space, CDMI builds on top of the existing communications infrastructure used by Columbus, providing reliability and ease of use.

Columbus systems communicate with ground via S-band, Ku-band, and Ka-band frequencies on the radio spectrum. Data from the S-band and Ku-band antennas is routed via the Tracking and Data Relay Satellite System (TDRSS) network [11], down to the White Sands Ground Terminal, and through NASA infrastructure. Data sent via the Ka-band antenna (mounted externally on Columbus) is routed via the European Data Relay Satellite (EDRS) network [12], down to EDRS Ground stations, and through ESA infrastructure. On ground, ESA and NASA infrastructure relays data via the Interconnecting Ground Subnet (IGS).

CDMI creates a layer of abstraction over this communications system: the IP Communications Service (IPCS), described in Section 3.1.2. The service routes data across any configured links, e.g. Ku and Ka, forming a gateway for bidirectional space-to-ground communication on flight and ground. Since the communication links are not only used by CDMI, but also by other Columbus and ISS components, the IPCS allows operators to configure bandwidth restrictions per link, which limit all CDMI communication between space and ground. Operators can also specify which link a payload can use.

Through this abstraction provided by the IPCS, the space-to-ground link becomes transparent for payload users: a user only has to connect to the IP address of their payload, and CDMI handles the underlying mechanisms. This approach also provides a benefit to CDMI itself: any other internal component that needs to communicate between space and ground can be treated similarly to a payload and can have its traffic managed by and routed via the IPCS.

3. Columbus Data Management Infrastructure

This section focuses on CDMI's hallmark properties: Its modular service infrastructure, its user interface for operators and PIs, and its DevSecOps concept.

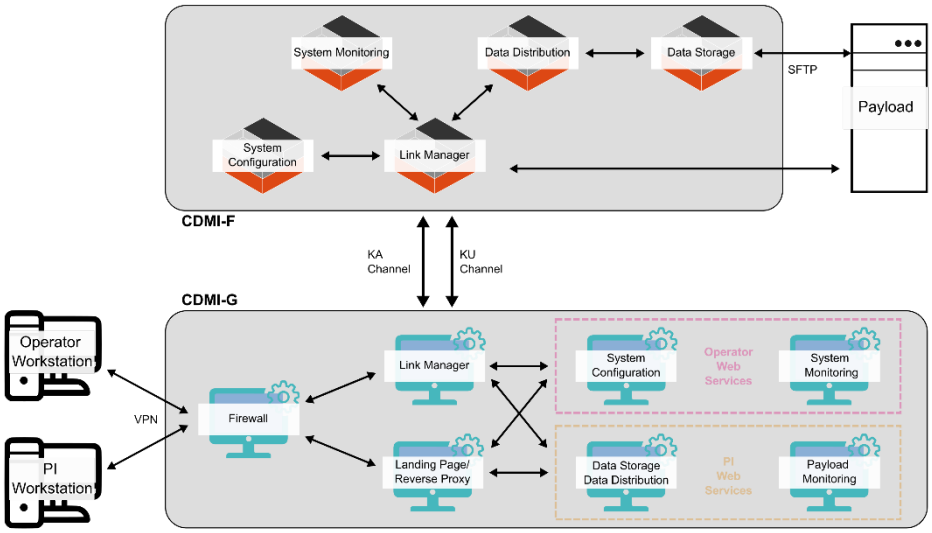

Figure 3: High-level diagram of the CDMI Service Infrastructure

3.1 Service Infrastructure

CDMI's overall functionality is best described through its different services. They are distributed between CDMI-G and CDMI-F and are typically implemented in a single VM or LXC. An overview of the most important VMs and Containers and the related dataflow for PIs and operators can be seen in Figure 3.

3.1.1 Identity Service

User identity management is critical for CDMI because its functionality varies for different user groups. All users enter CDMI through its landing page, which acts as a gateway for all other CDMI web interfaces. When they reach the page, they are prompted to log into CDMI via its single sign-on service, implemented with Keycloak. Keycloak maps the user to a set of roles and permissions that shape all other aspects of CDMI. For example, the landing page will only show a selected number of services to a PI, whereas it will show all services to a CDMI administrator. Once logged in, the user can now visit services such as the Ground Data Service or the Payload Monitoring Service and be presented with their data immediately, without the need for further authentication. The Identity Service, consisting of the landing page and Keycloak, ensures that the underlying distributed system looks and feels like a single application.

3.1.2 IP Communications Service

Central to communication within CDMI, the IP Communications Service (IPCS) provides end-to-end connectivity between ground users and their payloads on the ISS, and for any CDMI services communicating between flight and ground. This communication is transparent to all IPCS clients: a ground user connects to their payload's onboard IP address and the IPCS handles all routing and address translation. The service consists of two components: the firewalls, and the Link Manager (LM).

The firewalls determine what traffic can enter CDMI, and potentially continue to the LM and into space. They allow fine-grained control over network communication permissions, e.g., which ground IP addresses are allowed to access which onboard IP addresses or networks. In this way, a ground user may access their files in the ground storage, without being able to reach any infrastructure in orbit.

The LM monitors the space-to-ground links, establishes communication tunnels, and configures routes between ground and flight. It is split between two instances: one on the ground (LM-G), and one in flight on the Station (LM-F). LM-G periodically sends a ping across each configured link and waits for LM-F to respond. Depending on the latency of the ping responses, and how frequently they arrive, the LM determines that a link is up and can be used for communication (AOS status), or that it is down, and no communication is possible (LOS status). It also broadcasts this information on a local TCP stream that other services, such as the File Exchange Service, can use to determine a link's status. If a link is up, the LM creates a tunnel on the link, which can be configured as either Foo Over UDP (FOU) [13] or IPSEC[6]. It then creates routes for access to clients in its configuration (both payloads and system clients), using the Netlink interface [14] to manipulate networking facilities in the Linux kernel[7]. A client can be set to use

[6] https://strongswan.org/

[7] https://kernel.org/

any of the configured links, which allows an operator to balance the communication load across all available links. Furthermore, since the LM uses the kernel's network stack, a system client (e.g. FES) can automatically failover from a link that lost signal to a different link.

3.1.3 File Exchange Service

The File Exchange Service (FES) allows an arbitrary number of payloads to asynchronously transfer files by mirroring them bidirectionally between flight and ground, similar to MPCC's Dropbox [6]. A user of a payload places the data they wish to transfer into a specific directory via the Flight or Ground Data Storage service, which the FES periodically scans. When the FES sees new files, it adds them to a queue of files to be transferred. When a transfer slot becomes free for a payload, the FES takes files from this queue and begins transferring them. As part of the configuration, a payload has a maximum number of current transfers defined. It can also have file prioritization enabled, which means the user of a given payload can place files in a high-priority directory; these files will begin transferring before any non-high-priority files of the payload.

Internally, FES uses the CCSDS File Delivery Protocol (CFDP) [15] for data transfer. There are four main parts to the FES: the file store, the FES server, the CFDP daemon, and the FES transport. The file store abstracts operations on a part of the payloads' storage filesystems, via primitives specified by CFDP [15], and is made available to users through the Flight Data and Ground Data Distribution services. The FES server connects the file store with the CFDP daemon, and reports telemetry (e.g., number of transfers, the latest complete transfer, file checksums). The FES transport handles sending data over the space-to-ground links: it is configured for both the Ka and Ku links on Columbus, with bandwidth limits set for each. It uses all configured links in parallel: it receives AOS/LOS status from the Link Manager and sends CFDP Protocol Data Units (PDUs) on whichever link is available, through the IP Communications Service. Finally, the CFDP daemon sits between the FES server and transport, managing file transfers as CFDP transactions. A file is added for transfer by sending a message to the daemon. The daemon then transfers the file in the form of CFDP PDUs via the FES transport, retransmitting parts of a file when necessary, and calculates and verifies a checksum over the file. As such, it enables reliable, correct, and complete file delivery.

3.1.4 Storage Services

CDMI provides storage space to its users on the ground and in flight, with different emphases in their implementation. The Ground Data Storage Service is designed to facilitate convenient and secure access to flight-related data on the ground. To this end, a Nextcloud instance is integrated with the FES and connected to the redundant storage share at Col-CC. Nextcloud users can conveniently manage data from their home workstations and share data with collaborators working with the same payload. CDMI's Identity Service ensures that user permissions are correctly applied to all data stored on the ground.

The primary objective of the Flight Data Storage Service is to ensure the integrity of stored data, providing a secure repository for experimental data. The integrity of the data is ensured through multiple layers of protection, including a software-based RAID on the individual cluster nodes and data replication across multiple nodes within the cluster. Each PVE node hosts a dedicated data storage LXC container, which provides an SFTP share accessible by payloads connected to CDMI. The data storage container mounts a RAID-Z2 ZFS volume for each configured payload, thereby ensuring that the data is protected against the loss of two drives of the hosting PVE node. For particularly sensitive data, it is also possible to replicate the ZFS volume to other cluster nodes in fixed intervals, safeguarding it against the loss of an entire PVE node. The use of SFTP gives payloads a straightforward path to transfer data to the ZFS volume, where it is stored securely until it is either processed on-board or transferred to ground. To facilitate this transfer, each payload volume contains a separate storage section with the FES directory structure where files are picked up or moved to automatically by the FES.

3.1.5 Monitoring Services

CDMI's monitoring services consist of Zabbix for self-monitoring and Yamcs for payload monitoring. The core of the Zabbix setup is a server hosted in a VM on the ground, complemented by a proxy running in an LXC deployed in-flight. Yamcs is also hosted in a VM on the ground and exchanges data with both Zabbix and the Data Services Subsystem (DaSS), Col-CC's monitoring database.

The in-flight proxy aggregates metrics from all other in-flight containers, the Proxmox cluster hosts, and the on-board switches. For hardware, this includes critical parameters such as temperature and voltage. Additionally, the Zabbix proxy conducts routine reachability assessments of payloads by performing regular ping operations. Data collected by the proxy is periodically transmitted to the server on ground, where it is stored in a dedicated database.

On the ground, the Zabbix server monitors its neighbouring VMs but also extends its monitoring capabilities to custom parameters of various operational services, including the IP Communications, File Exchange, and Identity services. Thus, monitoring not only provides valuable insights into the health and

integrity of the system, but also detailed information on service utilization patterns, enabling informed decision-making and resource management.

Data collected by the Zabbix infrastructure is made accessible to operators via the Zabbix web GUI. The GUI is customizable in several ways: user-based customization, a native feature of Zabbix allowing operators to create custom dashboards, and global dynamic configuration, adapting selected parts of the Zabbix GUI to CDMI's latest configuration. For example, whenever a payload is added to the system, the Configuration Service adds a custom page for that payload to the GUI, populating it with its respective collected metrics.

By contrast, Yamcs aims to provide CDMI related metrics to PIs. This is achieved by making relevant Zabbix and DaSS parameters available using custom Yamcs data links. It is ensured that a payload's metrics can only be accessed by their respective users. Furthermore, payload monitoring data can be viewed on custom displays as part of the Yamcs web UI or obtained by PI external interfaces through Yamcs data streams.

Additionally, Yamcs is CDMI's interface to the DaSS, therefore retaining compatibility with existing Col-CC ground monitoring tools. Yamcs retrieves a subset of Zabbix items and publishes them to the DaSS, to provide the required metrics to Col-CC. Vice versa, data available in the DaSS can reach CDMI services through Yamcs.

3.1.6 Log Management Service

The Log Management Service centralizes log files from CDMI hosts in the ground storage and preserves the files on the originating hosts for a configurable period. This allows easy retrieval and analysis of log messages by developers and operators on ground, while enabling accountability and traceability. Log collection is achieved using rsyslog[8] servers on flight and ground with all other CDMI hosts acting as their clients. Each client forwards a copy of log messages to their nearest server, i.e., flight services to the flight server, and analogously for ground. Log files are rotated to new files based on file size and/or time limits using the logrotate[9] utility. Old files are compressed and purged after a configurable number of rotations.

The FES (see Section 3.1.3) transfers logs from flight to ground, thus centralizing them in the ground storage. Most log files are downlinked once they have been rotated, while a selection of critical files is transferred in real-time, with new log messages written to these files being downlinked as they occur.

3.1.7 Configuration Service

CDMI is orchestrated and maintained through a comprehensive Ansible-based deployment and operations system, providing improved consistency, automation, and scalability. This Configuration Service serves as the backbone of CDMI, ensuring that all components are consistently deployed and managed through a unified configuration management approach. The system is driven by configuration files, meaning that changes to the infrastructure or services are achieved by making simple adjustments to these configurations, reducing complexity and the risk of error.

The architecture consists of two Ansible controllers: one operating on the ground within a VM and another in-flight housed within a container. The ground-based Ansible controller is tightly integrated with the AnsibleForms web service, providing an intuitive interface for operators to interact with the system. The in-flight Ansible controller, while primarily triggered by its ground counterpart, possesses the capability to autonomously execute playbooks based on sent configuration files, ensuring continued operation and system management even in the event of LOS.

A key feature of the ground Ansible controller is its ability to maintain an up-to-date local representation of the system configuration. This local state is not only critical for pre-populating forms on AnsibleForms for operators, but also facilitates the detection of discrepancies between the desired and actual state of the system when compared to monitoring data observed in Zabbix.

3.1.8 Security Architecture

The security architecture of CDMI follows a defence-in-depth strategy. It is aligned with a risk management process and incorporates derived security controls. The security architecture is built on several key elements designed to ensure comprehensive protection across all systems:

(1) Network security is achieved through strict network segmentation, consistent enforcement of firewall policies in both the ground and flight segments and the implementation of intrusion detection systems. (2) Identity and access management is provided by role-based access controls and centralized single sign-on mechanisms utilizing state-of-the-art protocols. (3) Data security is guaranteed by safeguarding data both in transit and at rest, incorporating strict access control measures and secure communication protocols for data exchange within both flight and ground networks. (4) Endpoint security is ensured through integrating robust anti-malware defences.

The above-mentioned elements are supported by CDMI's CI/CD approach enabling fast patching and

[8] https://www.rsyslog.com/

[9] https://github.com/logrotate/logrotate

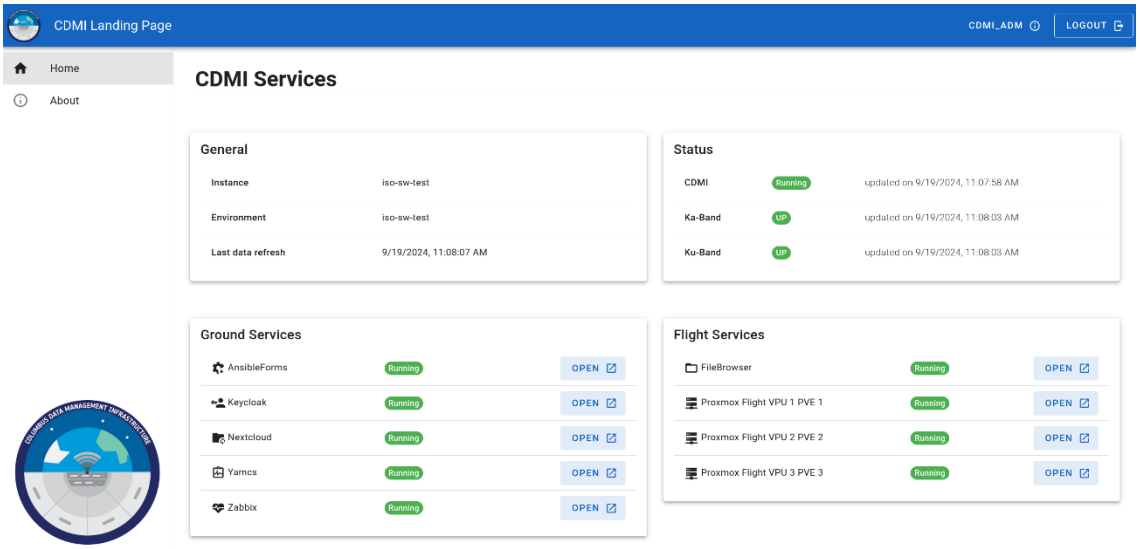

Figure 4: Screenshot of the CDMI landing page

update mechanisms. The infrastructure-as-code approach allows the maintenance of an automated Software Bill of Materials (SBOM). The SBOM enhances application and supply chain security, by enabling automated scanning for common vulnerabilities and exposures which is integrated into our patching policies. Finally, Software development security is automatically applied to both compiled and interpreted code, utilizing language-specific static application security testing and linters to maintain software integrity.

3.2 CDMI's User Interfaces

CDMI's infrastructure is distributed across multiple VMs and LXCs, yet it is accessible to users via a unified web interface. This interface, composed of a landing page and a reverse proxy, streamlines interactions with the entire infrastructure (see Figure 4).

3.2.1 Monitoring and Control

Operators control CDMI primarily through configuration changes. These changes are triggered by filling out the appropriate predefined form in AnsibleForms, ensuring ease of use while minimizing the risk of misconfiguration. For example, by filling out a single form, operators can dynamically register and unregister payloads from the system — a change that Ansible propagates throughout the entire infrastructure, including ground and flight machines. Operators can also configure CDMI's ground and flight firewalls, which allows them to granularly control who can access which parts of the system. In addition, CDMI's resources, such as storage and link bandwidth, can be dynamically configured for each payload, allowing tailored resource management based on individual requirements. For example, it is thus possible to temporarily increase a payload's storage space or its downlink bandwidth in the event of a resource-intensive experiment.

Monitoring through Zabbix further supports CDMI's resource management by providing an overview of current resource usage and issuing alerts when pre-defined limits are reached, ensuring efficient oversight. Overall, the administration of individual machines is kept to a minimum through CDMI's Infrastructure-as-Code approach, promoting consistency and reducing the need for manual intervention.

3.2.2 Using CDMI's services

PIs engage with CDMI through two discrete interfaces. The first interface is the landing page, which provides PIs access to Ground Storage via Nextcloud, Payload Monitoring through Yamcs, and comprehensive information on the overall status of CDMI.

Nextcloud enables PIs to share data with their collaborators, although it mainly serves as a gateway for data upload and download by providing access to the FES storage area (see Section 3.1.3). In contrast, Yamcs provides comprehensive information regarding the status and health of their payload. Through this interface, PIs can monitor essential data related to their payload, including real-time health metrics and current configuration within CDMI (e.g. bandwidth allocation, pending file transfers, and storage quota usage).

Additionally, Yamcs notifies PIs of the availability of the secondary interface, namely direct IP communications via the IP Communications Service (IPCS). Once an operator has granted access by enabling the IPCS with a specified bandwidth, PIs are able to establish a secure connection to their payload through a VPN tunnel. From their payload, PIs may interact directly with selected portions of the CDMI flight

infrastructure, such as accessing the flight data storage via SFTP.

3.3 CDMI's DevSecOps Concept

Fast software release cycles are highly desirable for CDMI, as it provides services to multiple independent PIs. One driver for fast release cycles is the adoption of new experiments or changes to existing ones, which require changes in the underlying service infrastructure. Another driver is the necessity for flexible adaptation of CDMI to future changes of software technologies or additional hardware becoming available on board.

Prerequisites for fast release cycles are lightweight verification processes reducing the overhead traditionally observed in the aerospace industry, as well as a suitable automated toolchain.

Two main pillars of the toolchain in CDMI are GitLab CI for building reliable CI/CD pipelines, and Ansible for complete automation of service deployment and configuration (see Section 2.1.1). Together with automated infrastructure provisioning these tools allow for quick and reproducible bootstrapping and deployment of complete CDMI environments comprised of all required flight nodes (CDMI-F) and ground nodes (CDMI-G).

The CI/CD pipelines automatically produce new CDMI versions by applying the following steps: (1) Code quality checks and static code analysis (linting) on source code and configuration templates; (2) Building of custom software components; (3) Unit testing of source code and configuration templates; (4) Deployment to a digital twin by applying Terraform and Ansible configurations; (5) Running integration and system tests on the digital twin.

Since this procedure takes a few hours only, it is used to provide fresh up-to-date environments to developers and testers every night, allowing for quick identification of integration issues.

To a degree, the deployment in step 4 and the testing in step 5 can be further shortened to do a reconfiguration and shallow testing on an existing environment only, making it also suitable for producing intermediate versions during ongoing development.

3.3.1 Configuration is running the show

CDMI's custom Ansible configuration files describe its entire infrastructure in code form comprising all services and their location within the system. Separate configuration files describe different aspects of CDMI, e.g. its VMs/LXCs, networks, service definitions, and storage configurations. Configurations are based on YAML schemas and may reference each other. This allows a simple representation of basic configuration parameters like hostnames, IP addresses, and ports, but also enables the dynamic representation of emergent parameters that require a combination of multiple pieces of information.

Since configuration changes are rolled out via Ansible only, any change to the host definitions, network structure, or service configuration, will automatically be reflected in dependent services including the monitoring systems, thus significantly reducing maintenance effort.

3.3.2 Environment twins

Multiple environments can be set up sharing the same basic configuration or at least the same structure, allowing for instantiating digital or physical twins of the operational environment.

Five environment classes are defined for CDMI. The *Isolated Software Development* environments are digital twins, i.e., isolated environments situated in ESA's private cloud infrastructure, used for development and automated testing. The *Software Development* environments combine development hardware located at the European Astronaut Centre (EAC) with digital twins of CDMI-G in the ESA cloud to produce (nearly) identical physical twins of the operational CDMI-F segment planned to run in the Columbus module. The *Software Test* environment utilizes a CDMI-G segment deployed in the Col-CC infrastructure with development hardware at EAC resembling CDMI-F. A similar high-fidelity hardware setup will act as the *System Reference Facility* used for training and integrated testing. Finally, there will be the *Payload Reference Facility* (PRF) environment, which represents a digital twin of CDMI that will be made available to payload developers. With the PRF it is possible to test a payload's integration into CDMI through remote network access.

In the development phase, these environments are used for demonstrating new features, enabling fast feedback from Col-CC operators. In preparation of delivery, manual and automatic validation and verification can be dry run on a physical twin to produce realistic test results, thus decreasing the number of iterations necessary to make a testing campaign successful.

4. CDMI Concept Verification

As CDMI is still under active development, there are no operational results available at this stage. However, preliminary tests have been conducted within the isolated development environment and the test environment to verify the feasibility of CDMI's core concepts.

These environments were demonstrated to future operators, showcasing a live user interface of CDMI. The demonstration focussed on a critical procedure: the

addition of a payload to the infrastructure. This procedure was selected because it historically required significant effort and touches nearly every service within CDMI.

Triggered by a straightforward Ansible Form, a single Ansible playbook successfully configured the entire system to accommodate a new payload. Once configured, the newly added payload was able to utilize a simulated space-to-ground link, enabling remote login and file transfer operations (see Figure 5).

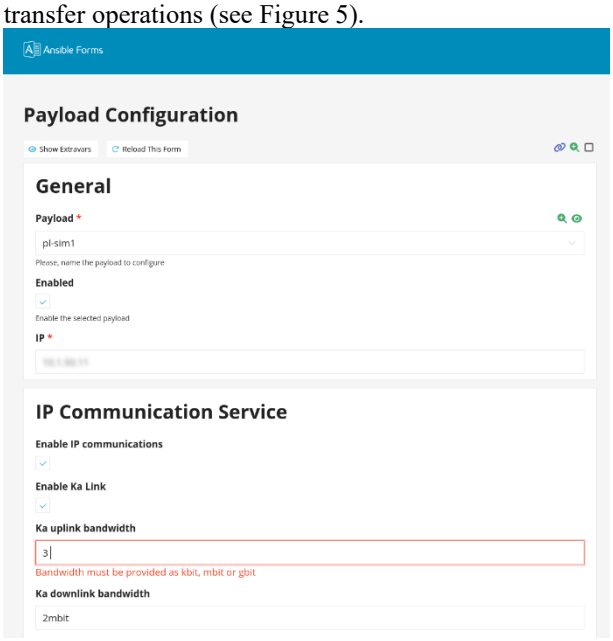

Figure 5: Example screenshot of the AnsibleForms Web UI. The IP-Comms for Payload Simulator 1 (pl-sim1) is configured. AnsibleForms ensures that input is in the correct format.

Importantly, the file transfer process demonstrated robust performance, even under adverse conditions. In the simulation, one of the two utilized space-to-ground channels was deliberately removed, yet the system continued to perform the file transfer without interruption, showcasing CDMI's resilience in handling potential loss of signal. Figure 6 shows the measured bits per second sent by the FES through the space-to-ground link during two file transfers.

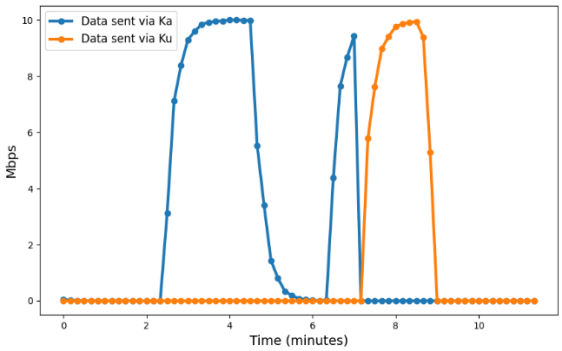

Figure 6: Ka and Ku link usage of the FES during two file transfers.

Starting at the two-minute mark, a first file was transferred entirely through the simulated Ka link without interruption. A second file was sent at the six-minute mark, but the simulated Ka link was removed during transfer. The FES autonomously changed to the Ku link and finished the remaining transfer.

These results, while preliminary, confirm the feasibility of CDMI's approach and provide a solid foundation for further development.

5. Discussion

This work presents CDMI, an advanced infrastructure of virtualized environments that integrates a suite of services distributed across both ground and flight segments. The preliminary results indicate that CDMI successfully combines an intuitive and user-friendly interface with robust, reliable functionality.

CDMI retains the core capabilities of its predecessor, MPCC, particularly in enabling PIs to remotely connect to their payloads and efficiently downlink data. Moreover, CDMI puts additional emphasis on reliability through redundancy, extensibility through rapid software release cycles, and maintainability through modular service components.

A key feature of CDMI is its foundation on hypervisor technology, which opens a range of possibilities for future development. This design not only enhances its current capabilities but also allows for the seamless integration of new services and even custom virtual machines tailored to the specific needs of PIs. Moreover, implementing a DevSecOps approach within CDMI further strengthens its extensibility, enabling rapid development, testing, and deployment of future extensions. This approach ensures that new features and services can be added efficiently and reliably, maintaining the system's robustness while continuously evolving to meet emerging requirements.

5.1 Improving Operations

The new CDMI user interface is expected to reduce operator intervention and chance for human error in comparison to the procedures currently in use. On the one hand, the utilization of AnsibleForms offers a user-friendly interface with input sanitization. Conversely, the integration of Ansible as a single platform that interacts with numerous services should diminish the necessity for operator intervention.

Furthermore, CDMI's open configuration concept facilitates planning at an incremental level allowing operators to easily implement short-term planning changes without the need for extensive reconfiguration. This, along with the straightforward patching process facilitated by Git, should minimize planned downtime by speeding up maintenance. With smaller downtime, patches are expected to be applied more quickly as there is less impact on ongoing science, thus simplifying

scheduling. The goal is to make rolling out patches, particularly security-related ones, as fast and painless as possible. For example, adding or removing a payload from CDMI, a task which can require several configuration changes to the existing system, will be realizable through the execution of a single playbook. The integration of new payloads into the configuration of CDMI's services is expected to occur seamlessly, with no anticipated downtime.

As CDMI will feature a far greater number of hosts compared to MPCC and has the scope to increase in complexity, human error could become a bigger factor if not mitigated by strong controls within the system. The big advantage of Ansible for the CDMI operator is that it provides a management platform for the execution of playbooks. Rather than executing numerous commands touching multiple hosts, the operator relies on Ansible to appropriately configure the entire system and must only review the job output and analyse changes in telemetry. Although routine operator tasks can be automated without Ansible through simple scripting, the Ansible platform manages the execution of playbooks for the user in several ways. Ansible populates the playbook template for the user at execution time: the set of hosts to be managed is tracked in Ansible's dynamic host inventory, particularly suitable in virtual and containerized environments. Ansible also provides easily digestible information to the operator about the outcome of each task on each host. In cases where a routine task fails a large part of the diagnostic analysis is already done for the operator by Ansible. Additionally, AnsibleForms offers the functionality to schedule playbook runs for future execution, thus allowing operators to manage their time more efficiently and reducing the concentration of tasks during high-pressure periods. The traditional monitoring and control system in use at Col-CC handles commands atomically, so that even though automated routines can be created, the routines are static and cannot customize themselves to the running environment.

5.2 Improving Reliability

Operators and PIs alike will benefit from CDMI's enhanced reliability features at both the hardware and service levels. For example, in terms of file downlink, CDMI's File Exchange Service provides improved data integrity. It uses CFDP's native checksum functionality, where checksums are created and verified for each file transfer. For payloads requiring the highest levels of data integrity, MPCC relied on manual checks and user intervention, which resulted in a high administrative burden. Since checksum verification is now a part of the transfer process, it can allow automated mitigation, such as re-transferring a corrupted file.

With its three VPUs, CDMI implements redundancy at multiple levels, improving the achievement of mission objectives. The CDMI VPUs use different power sources, minimizing the risk of a complete CDMI power failure, which would not only affect payload communications and payload data storage, but also access to critical Columbus components such as the Columbus LAN Switches or the on-board terminal of the Columbus Ka-band service. In the event of a power failure affecting one or two VPUs, most operations can be resumed without crew intervention through standard ground reconfigurations achieved through VPU hot redundancy. These reconfigurations are largely transparent to the payload users, ensuring uninterrupted payload communications.

The Intel AMT allows a powered VPU to be booted remotely, even if the operating system is not running, to facilitate troubleshooting or power-down operations. The different VPUs will have redundant connections not only within the CDMI and the Columbus network, but also to the NASA data interfaces, making CDMI robust against failure or power loss of devices in its communication chain, and improving the availability of equipment that depends on CDMI. The CDMI software can automatically adapt to losses within the VPU configuration and transfer active services to other VPUs, making CDMI a more reliable data management system. With the improved redundancy and more robust hardware and software design, crew interaction with CDMI is expected to be reduced to a minimum of non-nominal cases. The most likely case of crew intervention will be to replace an SSD that has been damaged by prolonged exposure to radiation. The use of ZFS RAID software, and the spare parts supplied with CDMI, should allow a seamless replacement procedure.

5.3 Improving Extensibility

One of the most compelling aspects of CDMI's architecture is its extensibility, largely enabled by the underlying hypervisor technology. CDMI has been provisioned with significantly more computing resources than are currently required, providing ample capacity for future expansion. This foresight allows CDMI to evolve beyond its initial capabilities in two primary ways: by adding more flight services and by enabling PIs to host their own private VMs on the platform, effectively transforming CDMI into a "cloud above the sky".

Future services that could be offered by CDMI are, for example, dedicated compression services for images or video data, reducing the need for extensive file downlinks by optimizing data before transmission. Another potential service could be a flight package mirror, which would facilitate fast and secure software updates for devices connected to CDMI.

Furthermore, the ability for PIs to deploy private VMs within CDMI could have significant long-term implications for payload development. By offloading resource-intensive tasks to these virtualized environments, future payloads may be designed with

fewer onboard computational resources, reducing both complexity and cost. This capability not only enhances the flexibility and utility of CDMI but also promotes a more efficient use of resources in space missions.

6. Conclusion

CDMI demonstrates how modern principles and paradigms from Earth-based server architecture can be effectively applied to space infrastructure. By adhering to the concept of using COTS software and hardware whenever possible, CDMI reduces both development and maintenance costs. The adoption of a DevSecOps approach ensures fast release cycles, enabling the incorporation of short-term feedback from its user base. CDMI's modernized operator interfaces can be continuously extended to cater to future extensions of the system. Especially as CDMI's hypervisor-based setup facilitates long-term extensibility, allowing the platform to evolve and expand its capabilities over time.

As CDMI looks forward to its planned launch in 2025, it aims to positively influence the design of future space systems. Further lessons learned from the operational phase are anticipated, which will continue to refine and enhance the framework.

Acknowledgements

This project was carried out jointly by ALTEC S.p.A., CGI Deutschland B.V. & Co. KG, Space Applications Services NV/SA, and ESA. The authors would like to thank Stephen Ennis and Han Wessels at ESA and Mathieu Schmitt, Nicolae Mihalache, Massimiliano Signori, Martin Ursik and the CDMI Hardware Team at Space Applications Services for their support.

References

[1] H. Stenuit and M. Ricci, "ICE Cubes---International Commercial Experiment Service for Fast-Track, Simple and Affordable Access to Space for Research---Status and Evolution," in *Space Capacity Building in the XXI Century*, Springer International Publishing, 2020, pp. 95--107.

[2] M. Gisi, L. Pfeiffer, A. Stettner, R. Seurig, M. Wahle, A. Honne, K. Kaspersen, K. Bakke, J. Thielemann and A. Liverud, "ANITA2 Trace Gas Analyser for the ISS-Flight Model Finalisation, Ground Test Results, and ANITA-X for future exploration missions," in *50th International Conference on Environmental Systems*, Lisbon, 2021.

[3] J. Love, "Science in Space: Week of Sept. 8, 2023 – The Immune System in Space," National Aeronautics and Space Administration, 11 09 2023. [Online]. Available: https://www.nasa.gov/missions/station/iss-research/science-in-space-week-of-sept-8-2023-the-immune-system-in-space/. [Accessed 18 09 2024].

[4] The European Space Agency, "Muninn Launch Kit," 2023. [Online]. Available: https://esamultimedia.esa.int /docs/HRE/Muninn_launchkit.pdf. [Accessed 18 09 2024].

[5] D. Honess and O. Quinlan, "Astro pi: Running your code aboard the international space station," *Acta Astronautica,* vol. 138, pp. 43-52, 2017.

[6] T. Müller, D. Burdulis and C. Corsten, "MPCC and Ku-IPS, New Ways to Control the Next Generation of Columbus Payloads-Ground Segment Aspects," in *International Astronautical Congress (IAC)*, Adelaide, 2017.

[7] A. Schlerf, D. Sabath, G. Soellner and I. Verzola, "Implementation of an additional command system, pathing the way for new tasks at Col-CC," in *International Astronautical Congress (IAC)*, Adelaide, 2017.

[8] G. Brunetti, G. Campiti, M. Tagliente and C. Ciminelli, "COTS Devices for Space Missions in LEO," *IEEE Access,* vol. 12, pp. 76478-76514, 2024.

[9] Hewlett Packard Enterprise, "HPE Spaceborne Computer," Hewlett Packard Enterprise, 30 1 2024. [Online]. Available: https://www.hpe.com/us/en/compute/hpc/ supercomputing/spaceborne.html. [Accessed 19 9 2024].

[10] Space Applications Services, "About Yamcs," Space Applications Services, 30 August 2024. [Online]. Available: https://yamcs.org/about. [Accessed 30 August 2024].

[11] The European Space Agency, "Annex 1: Additional technical Information on ISS capabilities and background information," 2011.

[12] The European Space Agency, "Data-relay system connects astronauts direct to Europe," The European Space Agency, 17 01 2022. [Online]. Available: https://www.esa.int/Applications/ Connectivity_and_Secure_Communications/ Data-relay_system_connects_astronauts_direct _to_Europe. [Accessed 02 09 2024].

[13] J. Corbet, "Foo over UDP," LWN.net, 1 10 2014. [Online]. Available: https://lwn.net/Articles/614348/. [Accessed 06 09 2024].

[14] P. N. Ayuso, R. M. Gasca and L. Lefevre, "Communicating between the kernel and user-space in Linux using Netlink sockets," John Wiley & Sons, Ltd., 2010.

[15] The Consultative Committee for Space Data Systems, "CCSDS File Delivery Protocol (CFDP)," CCSDS Secretariat, National Aeronautics and Space Administration, Washington, DC, 2020.