

Asynchronous Network Management System (ANMS) Product Guide

**DOC-005444, Prepared by The Johns Hopkins Applied
Physics Laboratory**

Copyright © 2023-2025 The Johns Hopkins University Applied Physics Laboratory LLC

License

This document is part of the Asynchronous Network Management System (ANMS).

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

This work was performed for the Jet Propulsion Laboratory, California Institute of Technology, sponsored by the United States Government under the prime contract 80NM0018D0004 between the Caltech and NASA under subcontract 1658085.

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
Initial	30 August 2023	Initial issue of document for ANMS v1.0.0	
A	28 August 2024	Updates for ANMS v1.1.0	
B	31 July 2025	Updates for ANMS v2.0	

Contents

Introduction	viii
Identification	viii
Scope	viii
Terminology	viii
References	ix
1 ANMS Architecture	1
1.1 ANMS Components	1
1.1.1 Light Components	3
1.2 Deployment	4
1.2.1 Using a CAM Gateway Emulator	5
1.3 Containers	6
1.4 Filesystem	7
1.5 Networking	8
1.6 Security	8
1.7 Long-Term Databases	8
2 Development Procedures	9
2.1 Test Build and Deployment	9
2.2 Production-Like Build and Installation	9
3 Production Procedures	11
3.1 Building	11
3.2 Installation	12
3.2.1 TLS Configuration Volume	13
3.3 Upgrading	13
3.4 Resetting Container State	13

3.5	Monitoring	14
3.6	Long-Term Database Backup and Restore	14
3.6.1	Container State and Logs	14
3.6.2	SELinux Audit Events	15
3.7	Checkout Procedures	15
3.7.1	Frontend Communication	15
3.7.2	AMP Manager Proxy Connection	16
3.7.3	BP Agent Communication	16
4	Product Support	18
4.1	Troubleshooting	18
4.1.1	Installation	18
4.1.1.1	SELinux Blocked Behavior	18
4.1.2	Operations	18
4.1.2.1	Grafana Containers	19
4.1.2.2	Agent Registration Issues on ANMS Startup	19
4.1.2.3	New Agent Registration Issues	19
4.1.2.4	AMP Database Querying	19
4.1.2.5	ANMS-UI is not visible at hostname:9030	20
4.1.2.6	ANMS-UI is not visible at hostname	20
4.1.2.7	Other container startup issues	20
4.1.3	Advanced	20
4.1.3.1	Override Configuration Files with Volume Mounts	20
4.1.3.2	General Networking Issues	21
4.2	Contacting or Contributing	21

List of Figures

1.1	ANMS Components with Protocol Associations	2
1.2	ANMS Primary Components with Logical Associations	3
1.3	ANMS Light Components with Protocol Associations	4

List of Tables

1	Applicable JPL Rules Documents	ix
2	Applicable MGSS Documents	ix
3	Applicable Other Documents	ix
1.1	Target host packages	5

Introduction

This Product Guide provides architectural and maintenance details about the Asynchronous Network Management System (ANMS), which is part of the Advanced Multi Mission Operations System (AMMOS) suite of tools.

Identification

Property	Value
Configuration ID (CI)	631.17
Element	Multi-Mission Control System (MMCS)
Program Set	Asynchronous Network Management System (ANMS)
Version	3.0

Scope

This document describes technical details about the ANMS installation, upgrade, monitoring, and maintenance. For details about the user interface and workflows of the ANMS see the [ANMS User Guide](#).

Terminology

Asynchronous Management Protocol (AMP)

The application protocol used to communicate between ANMS and its managed agents.

Bundle Protocol (BP)

The overlay network protocol used to transport AMP messages between ANMS and its managed agents.

BP Agent (BPA)

The instantiation of a BP node with a unique administrative Endpoint ID.

BP Endpoint

The source or destination of a BP bundle.

Container

An isolated unit of runtime state within a host.

Convergence Layer Adapter (CLA)

A specific mechanism used to transport bundles within an underlay (IP) network.

Endpoint ID (EID)

The identifier of a BP Endpoint; names the source and destination for a BP bundle.

Host

A single node on the network and a single instance of an operating system. One host can have many interfaces and many IP addresses, but only one canonical host name.

Internet Protocol (IP)

The network protocol used for inter-container communication and for BP convergence layer messaging with AMP Agents.

References

Title	Document Number
Software Development	57653 rev 10

Table 1: Applicable JPL Rules Documents

Title	Document Number
MGSS Implementation and Maintenance Task Requirements	DOC-001455 rev G
Common Access Manager (CAM) Product Guide (PG)	DOC-005065
ANMS Architecture Description Document	DOC-005089
ANMS Software Design Document	DOC-005445
ANMS Software Interface Specification	DOC-005446
ANMS User Guide	DOC-005443 rev B

Table 2: Applicable MGSS Documents

Title	Reference
Installing Puppet	puppet-agent
Installing Bolt	puppet-bolt
Using SELinux	rhel9-selinux
ANMS Source	anms-source
ANMS Guide Document Source	anms-docs

Table 3: Applicable Other Documents

Chapter 1

ANMS Architecture

The ANMS is designed with a microservice architecture using containers for isolation and network protocols to communicate between services. Many of the service interfaces use stateless HTTP exchanges, some use MQTT for pub-sub interactions and some use PostgreSQL for long-term configuration and data warehousing. Although some of the subsystems have a preferred start-up order (to ensure initial configuration is valid and consistent) the containers can be restarted in any order with a few specific exceptions.

1.1 ANMS Components

The subsystems (containers) of the ANMS are illustrated as gray blocks within the "ANMS Instance" group in the diagram of Figure 1.1. The entire ANMS instance is made to be run on a single host, with future plans to allow installing in a more distributed environment. Currently the ANMS provides security at the boundary of the instance but not between containers (see Section 1.6 for details), which would be required for a distributed installation.

A higher-level logical view of the ANMS is shown in Figure 1.2 where some of the internal infrastructure containers (e.g., PostgreSQL database, MQTT broker) are removed for clarity.

The User Agents in both diagrams are how a user can interact with the ANMS, which is solely via HTTP exchanges. Most of the ANMS API follows RESTful principals of stateless service interactions, while some of the API is more browser-oriented to provide UI visuals.

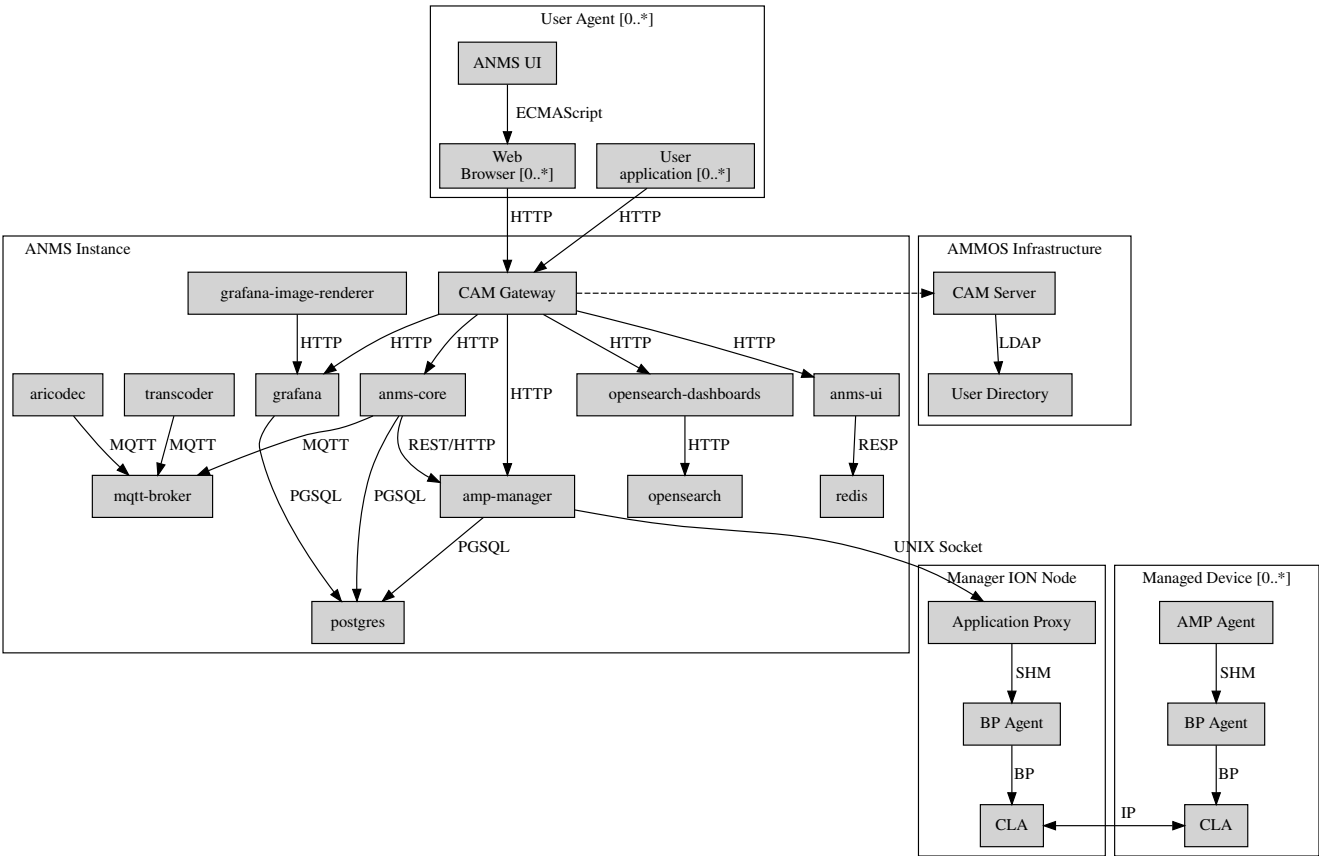


Figure 1.1: ANMS Components with Protocol Associations

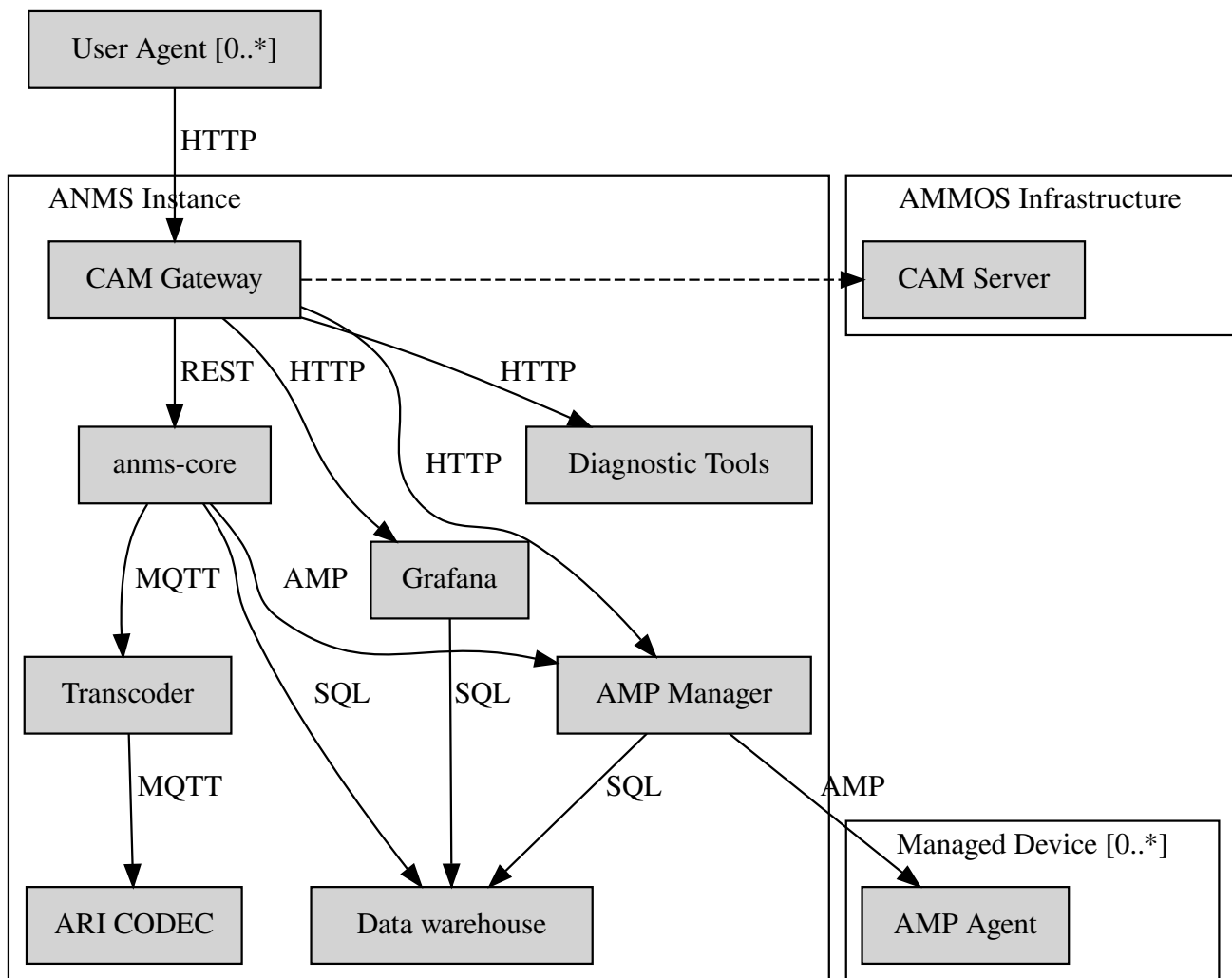


Figure 1.2: ANMS Primary Components with Logical Associations

1.1.1 Light Components

A lightweight ("light") configuration of the ANMS involves removing containers related to web browser user interfaces, including the `anms-ui` file server and its `redis` cache, the `grafana` plotting UI and its renderer, the `adminer` database inspection UI, and the `OpenSearch` dashboard UI.

This is depicted in the set of containers in the diagram of Figure 1.3. The expectation of this light configuration is that there will not be a web browser user agent interacting with the ANMS, but instead some **HTTP** client application operated by another operations/management system external to ANMS.

In the light configuration, the ANMS will still support loading of ADMs needed to operate the transcoder service, sending immediate execution messages to agents, receiving and archiving report messages from agents, and allowing direct access to report contents as well as the postgres data warehouse (which would normally be used by the `Grafana` plotting UI).

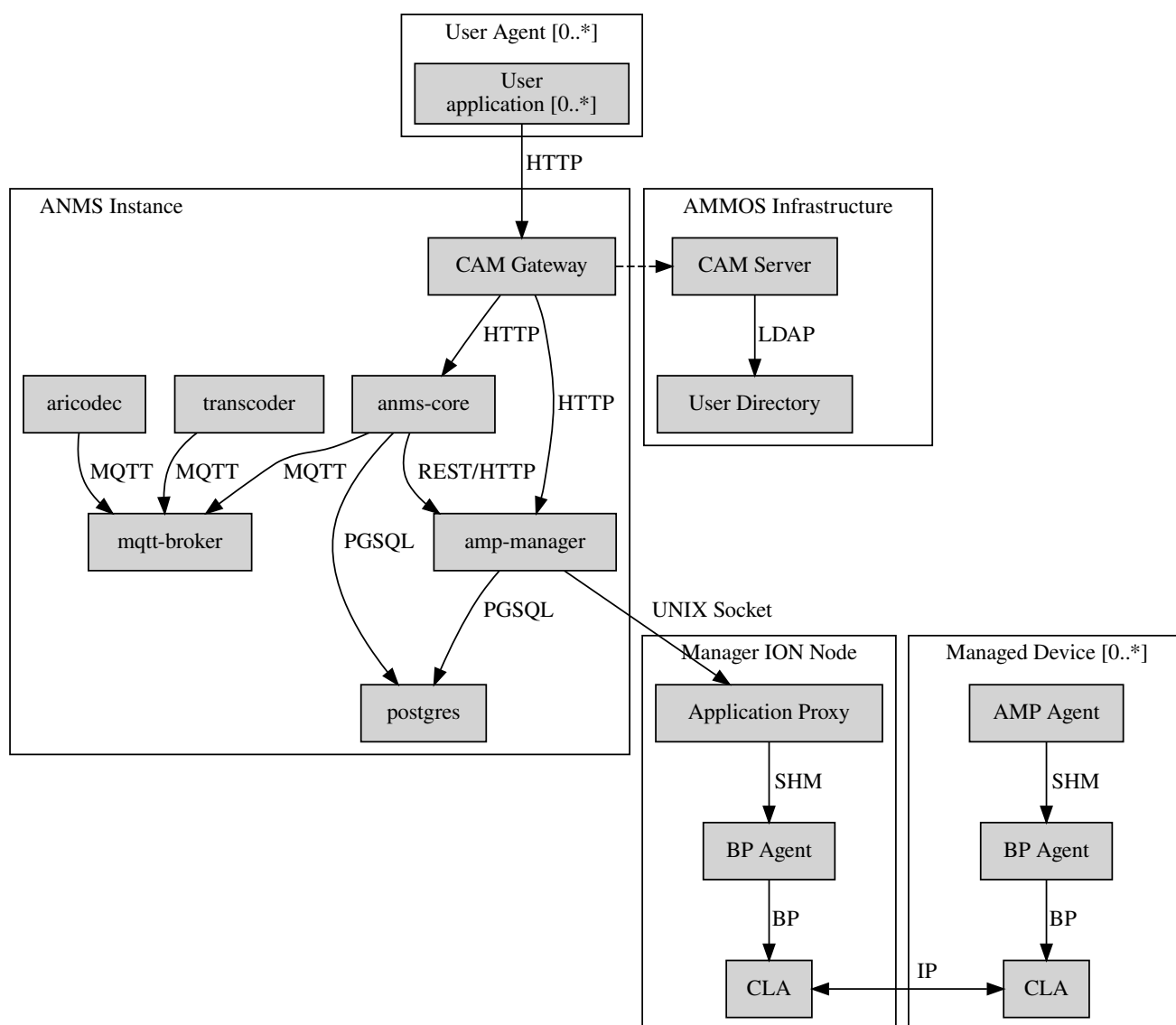


Figure 1.3: ANMS Light Components with Protocol Associations

1.2 Deployment

The target host will be running the RedHat Enterprise Linux (RHEL) version 9 (RHEL-9) with network interfaces configured, and IP addressing and DNS configured along with a running local firewall.

The ANMS is intended to be deployed using the Puppet orchestrator, either from a local Puppet apply execution or configured from a central Puppet server. Part of the ANMS source is a Puppet module "anms" to automate the configuration of an ANMS deployment. Specific procedures for performing an installation using a local Puppet apply are in Section 3.2.

Conditions for installing the ANMS are a host with packages identified in Table 1.1, at least 7 GiB of filesystem space for

container image storage, and additional space for long-term data warehouse storage. The total amount of storage needed depends on the mission use of reports, specifically the average size and rate of reported data.

Note

Usage of podman is recommended. Docker should continue to function as a drop-in replacement, however only podman deployments will be directly supported by the ANMS team.

Note

It is recommended to use docker-compose (which is fully compatible with podman). docker-compose is a distinct package from Docker, while podman-compose is a fully compatible alternative. Older versions of compose were called directly (*i.e.*, `docker-compose`), while current versions are invoked as a sub-command (*i.e.*, `docker compose` or `podman compose`).

Package Name	Version Minimum
podman	5.2+
docker-compose (preferred) OR podman-compose.	2.29+ OR 5.3+
systemd	252
Puppet	7
Puppet Bolt	4

Table 1.1: Target host packages

The ANMS is designed to operate on a network where the MGSS Common Access Manager (CAM) is used to manage user accounts and a CAM Gateway is used as a reverse proxy within the ANMS installation to enforce user login sessions and access permissions. The ANMS has been exercised with CAM v5.1.0 in a test environment outside of the MGSS environment. To deploy the ANMS in an environment without a CAM instance available (or without using it) the ANMS can be built with a CAM Gateway emulator as described in Section 1.2.1. In any case, deployment and configuration of CAM itself is outside the scope of this document and is described in detail in the [CAM Product Guide](#).

1.2.1 Using a CAM Gateway Emulator

To allow the ANMS to be tested in environments where a CAM Server is unavailable or too burdensome to set up, the ANMS can be built with an emulator of the CAM Gateway which uses static accounts, credentials, and access controls. This is now known as the "demo" configuration.

Auth mode is determined by the environment variable `AUTHNZ_EMU` during build (see Section 3.1). Set `AUTHNZ_EMU=cam-gateway` to utilize the CAM Server, or `AUTHNZ_EMU=demo` to utilize the CAM Gateway emulation (a basic authentication demo configuration).


Caution

The CAM Gateway emulator is for demonstration only and must not be present in a production installation.

The static accounts available in the emulator, defined in an `htpasswd` file, are:

test With password `test`, is able to access all typical ANMS UI and features.

admin With password `admin`, is able to access all typical ANMS UI and features as well as the `/adminer/...` and `/nm/...` APIs.

1.3 Containers

The containers defined by the ANMS compose configuration in Section 1.4 are as follows in alphabetical order. Associations between these containers are illustrated in Figure 1.1.

adminer Administrative access to the PostgreSQL database, which requires special authorization. Exposes TCP port 8080 for HTTP.

authnz The CAM Gateway reverse proxy for authentication, authorization, and auditing (AAA); also the endpoint of user agent TLS connections. This container uses the external `amos-tls` volume for TLS configuration (see Section 3.2.1). This provides HTTP routing access to `anms-ui`, `anms-core`, and `grafana` containers. Exposes TCP port 443 for HTTPS and 80 for HTTP, both mapped to the same host port numbers. This container can be remapped to use the `cam-gateway` or `demo` (basic http auth) configuration.

anms-core The ANMS backend REST services. Exposes TCP port 5555 for HTTP.

anms-ui The ANMS frontend REST services and browser client UI. Exposes TCP port 9030 for HTTP.

grafana The data warehouse plotting engine. This uses the `grafana-data` volume for storage. Exposes TCP port 3000 for HTTP.

grafana-image-renderer Image renderer for the `grafana` subsystem. Exposes TCP port 8081 for internal APIs.

ion-manager A combination of the AMP Manager used by ANMS and the BP Agent used for message transport. Exposes UDP port 1113 for LTPCL and port 4556 for UDPCL, and TCP port 8089 for HTTP API; the CL ports are mapped to the same host port numbers.

aricodec A service to convert ARI between text and compressed binary form based on available ADMs.

mqtt-broker The broker host for MQTT pub-sub messaging. Exposes TCP port 1883 for MQTT.

opensearch Log aggregator for the ANMS. This uses the `opensearch` volume for storage. Exposes TCP port 9200 and 9600 for internal APIs.

opensearch-dashboards A user interface for accessing the `opensearch` logs. Exposes TCP port 5601 for HTTP.

postgres Persistent database for the ANMS. This uses the `postgres-data` volume for storage. Exposes TCP port 5432 for PSQL.

redis A database for state keeping from the ANMS UI. Exposes TCP port 6379 for redis API.

transcoder An intermediate service to bookkeep transcoding requests from the ANMS to the ARI CODEC engine.

1.4 Filesystem

Because the ANMS is deployed as a Compose configuration, the only primary files present on the host are to configure the containerized project, its use as a system service, and the system firewall.

The principal directories and files used by ANMS are:

/etc/systemd/system Directory for systemd local configuration units. Its contents for ANMS are:

podman-compose@.service A template systemd service for all compose projects represented as files below.

/etc/containers/compose/projects Compose project top-level service configuration files. Each file is used by the `podman-compose@` template service. Its contents for ANMS are:

anms.env The ANMS service configuration.

testenv.env The test environment service configuration.

/ammos/anms The project-specific deployment path for compose configurations. Its contents are:

.env The compose-level environment configuration for all of the ANMS and testenv containers, including host port mapping options (see Section 1.3).

anms-compose.yml Compose configuration for the ANMS containers, volumes, and networks. The compose runtime is managed by the `podman-compose@anms` template service.

testenv-compose.yml Compose configuration for the test environment containers and network. The compose runtime is managed by the `podman-compose@testenv` template service.

/run/anms The temporary file directory used at runtime. Its contents are:

proxy.sock The transport proxy socket needed by the AMP Manager in order to message to and from agents.

Secondary files related to the ANMS deployment are:

/etc/containers/containers.conf Configured to enable compose provider and SELinux for containers.

/etc/containers/compose/projects/anms.env The systemd environment for the `anms` project, used by the systemd service `podman-compose@anms`. This points to `/ammos/anms/` for its compose configuration.

/etc/systemd/system/podman-compose@.service A parameterized service interface which reads compose environment from under `/etc/containers/compose/projects/`

/ammos/etc/pki/tls Host-level PKI security configuration from which the ANMS user-agent TLS configuration is derived. Its contents for ANMS are:

private/ammos-server-key.pem The private key for the local TLS endpoint.

certs/ammos-server-cert.pem The corresponding public key certificate for the local TLS endpoint.

certs/ammos-ca-bundle.crt The trusted CA bundle for all TLS endpoints (including users).

/var/cache/puppet/puppet-selinux/modules The containing directory for SELinux modules for the ANMS containers (see Section 1.6).

1.5 Networking

The target host will be running RHEL-9 with network interfaces configured, and IP addressing and DNS configured along with a running local firewall.

The container network configuration (under `/ammos/anms`) for the ANMS includes host port forwarding for the following services:

HTTPS Default port 443 forwarded to the `authnz` container for HTTP use.

As depicted in Figure 1.1, the ANMS deployment itself does not include a BP Agent and does not either contain or require any specific configuration of a BP Agent or its associated convergence layers.

Note

The ANMS "testenv" compose configuration includes an optional AMP Manager and a set of local AMP Agents to use to test with. These use a separate "testenv" Docker network to communicate between each other, while real remote agents will require external network configuration to include port forwarding and host name resolution for the Manager-associated BP Agent. How those are configured is outside the scope of this document.

1.6 Security

The host on which the ANMS instance runs is expected to have FIPS-140 mode enabled and SELinux enabled and in enforcing mode. Part of the ANMS deployment includes an SELinux module for each of the component containers which allow all necessary inter-service communication. If issues with SELinux are suspected in a deployment, follow the procedures in Section 3.6.2 to find any audit events related to the ANMS.

The host is also expected to have a running OS-default firewall which will be configured by the Puppet module to allow HTTPS for user agent access.

The interface between ANMS and its User Agents is TLS-secured HTTP with a PKIX certificate supplied by the host network management and chained to the CA hierarchy of the network.

1.7 Long-Term Databases

The ANMS uses an internal PostgreSQL database for following purposes, all within the same schema `amp_core`:

User Configuration Each user account authorized to access the ANMS can have parameters associated with their account, mostly related to UI parameters.

ADM configuration The most static configuration of the ANMS is the set of ADMs available to all agents managed by that ANMS instance.

Agent configuration The Agent configuration consists of agents which are known to, and managed by, the ANMS which are parameterized by their AMP messaging BP EID and their associated CL parameters (network name/address and port).

Reported Data Warehouse When reports arrive from managed agents and are associated with known ADMs they are disassembled and stored as object-values in the historical data warehouse.

Chapter 2

Development Procedures

This chapter includes procedures related to development and pre-production testing of the ANMS. These procedures assume that the user has a working copy of the ANMS source tree and intends to make some changes to the source or the configuration outside of a normal production deployment procedure (where build happens on a different host from the ultimate installation).

2.1 Test Build and Deployment

TBD

2.2 Production-Like Build and Installation

This procedure builds images in the root user's podman context and then uses puppet to install to the local host.

More TBD

1. Images for all of the ANMS production and test containers can be built with the following:

```
sudo AUTHNZ_EMU=authnz-emu podman compose -p anms -f docker-compose.yml --podman-build-args='--format docker' build
sudo podman compose -p agents -f agent-compose.yml --podman-build-args='--format docker' build
```

1. The deployment configuration is set by editing the file `puppet/data/override.yaml` to contain similar to:

```
anms::docker_image_prefix: "" # Left empty to use locally built images
anms::docker_image_tag: "latest"
```

2. Pull the necessary upstream Puppet modules with:

```
./puppet/prep.sh
```

3. Perform a dry-run of the puppet apply with:

```
sudo PATH=/opt/puppetlabs/bin:$PATH ./puppet/apply_local.sh --test --noop
```

and verify that there are no unexpected changes.

4. Perform the actual puppet apply with:

```
sudo PATH=/opt/puppetlabs/bin:$PATH ./puppet/apply_local.sh --test
```

Chapter 3

Production Procedures

This chapter includes specific procedures related to managing an ANMS production instance.

3.1 Building

The ANMS source is composed of a top-level repository `anms` and a number of submodule repositories; all of them are required for building the ANMS.

Before beginning, ensure that either Docker or Podman (preferred) is installed and functional on your system along with `docker-compose`. A hello-world image can be run to verify functionality, *i.e.* `docker run --rm hello-world` or `podman run --rm hello-world`. Compose can be verified with `podman compose ps` - If compose is not properly installed it will report 'unrecognized command'

1. The top-level checkout can be done with:

```
git clone --recursive --branch <TAGNAME> https://github.com/NASA-AMMOS/anms.git
```

2. Optional: switching to a different tag or branch can be done with the sequence:

```
git checkout <TAGNAME>
git submodule update --init --recursive
```

3. If running containers as the root user (not recommended. Podman defaults to rootless), it may be necessary to add the local user to the `docker` access group with the following. :

```
sudo usermod -a -G docker ${USER}
```

4. Update `.env` file as needed. Fields that may need to be updated include `DOCKER_IMAGE_PREFIX` and for rootless podman the port mappings for AUTHNZ may need to be un-commented to avoid permissions issues
5. Build images using either: (Note: The `podman-build-args` flag is required for health checks to function under podman).

```
docker compose -f docker-compose.yml -f testenv-compose.yml build
podman compose --podman-build-args='--format docker' -f docker-compose.yml -f testenv- ↵
compose.yml build
```

Note

To build an ANMS that uses an emulator for the CAM Gateway (which means that the ANMS will not require a CAM server), have the following environment set in the build step above. (Note: The default value that uses the CAM Gateway is `cam-gateway`):

```
export AUTHNZ_EMU=demo
```

3.2 Installation

The ANMS uses Puppet version 7 [\[puppet-agent\]](#) to install requisite system packages and configure system files and services. In addition, Bolt [\[puppet-bolt\]](#) is needed to install needed Puppet modules and run the Puppet agent remotely.

**Caution**

The example TLS configuration in this procedure is for demonstration only and must not be present in a production installation. Details for creating a proper TLS volume are in [Section 3.2.1](#).

To install the ANMS on the local host perform the following:

1. A TLS configuration must be embedded in a volume mounted by the `authnz` frontend container with contents described in [Section 3.2.1](#). This can be done with a boilerplate test-only CA and certificates by running:

```
./create_volume.sh ./puppet/modules/apl_test/files/anms/tls
```

2. The deployment configuration is set by editing the file `puppet/data/override.yaml` to contain similar to:

```
anms::docker_image_prefix: "" # Matching the DOCKER_IMAGE_PREFIX from build procedure
anms::docker_image_tag: "latest" # Matching the tag name from build procedure
anms::docker_registry_user: ""
anms::docker_registry_pass: ""
```

3. Pull the necessary upstream Puppet modules with:

```
./puppet/prep.sh
```

4. Perform a dry-run of the puppet apply with:

```
sudo PATH=/opt/puppetlabs/bin:$PATH ./puppet/apply_local.sh --test --noop
```

and verify that there are no unexpected changes.

5. Perform the actual puppet apply with:

```
sudo PATH=/opt/puppetlabs/bin:$PATH ./puppet/apply_local.sh --test
```

3.2.1 TLS Configuration Volume

The docker volume mounted into the `authnz` container follows the AMMOS conventions for TLS certificate, private key, and CA file paths and contents.

The volume must contain the specific files:

/certs/ammos-server-cert.pem The PEM-encoded certificate for the ANMS frontend itself. It must have extended key use of `id-kp-serverAuth` for a web server.

/private/ammos-server-key.pem The PEM-encoded non-password-protected private key corresponding to the server certificate.

/certs/ammos-ca-bundle.crt The PEM-encoded CA bundle containing at least the CA chain used to sign the server certificate.

3.3 Upgrading

Because the ANMS is deployed as a series of containers managed by compose with associated environment variables and configuration, an upgrade involves rebuilding and restarting affected containers.

An upgrade can be performed using the same procedure as Section 3.2, where Puppet will make any required changes for the upgrade and restart services and containers as necessary. Individual ANMS releases may identify pre-upgrade or post-upgrade steps in their specific Release Description Document (RDD).

3.4 Resetting Container State



Warning

The following will reset all database state, including user profiles, ADM configuration, and all historical report data. This should only be used for test hosts or after performing a full Postgres DB backup (see Section 3.6).

To force containers and volumes (containing long-term database files) to be cleared, a maintainer can run the following from the host.

```
podman stop --all
podman system prune --all --volumes
```

To verify artifacts have been removed you may use the following. The prune command (or object-specific prune command) may need to be reissued if not all relevant artifacts have been cleared.

```
podman image ls
podman container ls
podman volume ls
podman network ls
```

Tip

In some cases, issues may arise due to system cache files on the host system not being cleared (a potential issue with select podman versions). It is recommended to restart the host system after clearing objects to ensure a clean start.

After clearing containers and volumes, the normal `apply_local` step of Section 3.2 should be performed to re-install and start the containers.

3.5 Monitoring

To enable Wireshark logging with patched AMP dissector, run similar to the following:

```
wireshark -i br-anms -f 'port 1113 or port 4556' -k
```

3.6 Long-Term Database Backup and Restore

Although the docker volume `anms_postgres` contains the raw database state, this will not allow backup of or transferring that state to other hosts.

To perform an online backup (keeping the database running) run the following on the host:

```
podman compose -p anms exec -T postgres pg_dump -Ft -d amp_core | gzip -c >~/anms-backup.↵  
tar.gz
```

which can the later be restored using:

```
gunzip -c <~/anms-backup.tar.gz | podman compose -p anms exec -T postgres pg_restore --↵  
clean -d amp_core
```

3.6.1 Container State and Logs

Because of the Compose configuration described in Section 1.4, accessing container state and logs requires running podman with a command similar to the following:

```
podman compose -p anms [action] ...
```

The state of all containers in the ANMS project can be observed with:

```
podman compose -p anms ps
```

which will report a "State" column either as "Up" for simple services or "Up (healthy)" for health-checked services.

And observing logs from specific containers requires running a command similar to:

```
podman compose -p anms logs [service-name]
```

3.6.2 SELinux Audit Events

The procedures in this section are a summary of more detail provided in Chapter 5 of the RedHat [\[rhel9-selinux\]](#) document.

By default, the `setroubleshootd` service is running, which intercepts SELinux audit events

To observe the system audit log in a formatted way run:

```
sudo sealert -l '*'
```

Some SELinux denials are marked as "don't audit" which suppresses normal audit logging when they occur. They are often associated with network access requests which would flood an audit log if they happen often and repeatedly. To enable logging of dontaudit events run:

```
sudo semanage dontaudit off
```

3.7 Checkout Procedures

Each of the following checkout procedures makes progressively more detailed and more normal-operations-like tests of the external interfaces with the ANMS.

In many fault cases, the procedure will work for the first steps and then fail on a specific step and thereafter. This is taken advantage of for the purposes of troubleshooting and failure reporting; the specific procedure(s) run and step(s) that fail are valuable to include in issue reports related to the ANMS.

To make the procedures more readable, the ANMS host is assumed to have the resolvable host name `anms-serv`. For checkout steps occurring on a "client host" it is assumed to be running RHEL-9 or equivalent from the perspective of commands available.

3.7.1 Frontend Communication

This procedure checks the mechanism that a user agent can communicate with the ANMS just as a browser or user application would.

The checkout procedure is as follows:

1. From the ANMS host verify firewall access with:

```
sudo firewall-cmd --zone public --list-services
```

which should include the services "https".

2. From a client host check the port is open with:

```
nmap anms-serv -p80,443
```

which should show a result similar to

PORT	STATE	SERVICE
80/tcp	closed	http
443/tcp	open	https

3. From a client host check HTTP access with:
-


```
curl --head https://anms-serv/
```

which should show a result containing lines similar to

```
HTTP/1.1 302 Found
Server: Apache/2.4.37 (Red Hat Enterprise Linux) OpenSSL/1.1.1k
Location: /authn/login.html
```

4. From a client host check a test login account with:

```
curl --head --user test https://anms-serv/
```

along with the credentials for that account, which should show a result containing lines similar to

```
HTTP/1.1 302 Found
Server: Apache/2.4.37 (Red Hat Enterprise Linux) OpenSSL/1.1.1k
Location: /authn/login.html
```

3.7.2 AMP Manager Proxy Connection

This procedure checks whether the AMP Manager within the ANMS can connect to its transport proxy.

The checkout procedure is as follows:

1. From the ANMS host verify the proxy socket exists with:

```
ls -l /run/anms/
```

which should include the file "proxy.sock".

2. From the ANMS host check the REFDM daemon logs with:

```
docker compose -p anms exec -T amp-manager journalctl --unit refdm-proxy | grep -B4 ↔
Connected
```

3.7.3 BP Agent Communication

This procedure checks whether the BPA in the "testenv" network can communicate with the BPA of a specific managed device.

The checkout procedure is as follows:

1. From the ANMS host verify firewall access with:

```
sudo firewall-cmd --zone public --list-services
```

which should include the services "ltp" and "dtn-bundle-udp".

2. From any RHEL-9 host on the agent network run the following:

```
sudo nmap anms-serv -sU -p4556
```

which should show a result similar to

PORT	STATE	SERVICE
4556/udp	filtered	dtm-bundle-udp

3. From the ANMS host run the following, substituting the host name/address of any valid BP Agent:

```
sudo nmap amp-agent -sU -p4556
```

which should show a result similar to

PORT	STATE	SERVICE
4556/udp	filtered	dtm-bundle-udp

4. From the ANMS host run the following:

```
podman compose -p testenv exec ion-manager ion_ping_peers 1 2 3
```

Chapter 4

Product Support

There are two levels of support for the ANMS: troubleshooting by the administrator or user attempting to install or operate the ANMS, which is detailed in Section 4.1, and upstream support via the ANMS public GitHub project, accessible as described in Section 4.2. Attempts to troubleshoot should be made before submitting issue tickets to the upstream project.

4.1 Troubleshooting

4.1.1 Installation

This section covers issues that can occur during installation (see Section 3.2) of the ANMS.

4.1.1.1 SELinux Blocked Behavior

If there are errors related to the SELinux modules for the ANMS containers during installation of the ANMS on the local host, as discussed in Section 3.2, add the following line to the Puppet `common.yaml` file, typically found at `puppet/data/common.yaml`, and redeploy.

```
selinux::mode: permissive
```

This will result in the host being in permissive mode which allows activity not defined in SELinux modules but records those events to the system audit log. See Section 3.6.2 for details on observing the audit log events.

**Caution**

The SELinux permissive mode is for troubleshooting only and must not be present in a production installation.

4.1.2 Operations

This section covers issues that can occur after successful installation (see Section 3.2) and checkout (see Section 3.7) of the ANMS.

4.1.2.1 Grafana Containers

If the Grafana panels in the Monitor tab displays Connection was reset errors, the Grafana container may not have started successfully.

Restart the container with `podman compose up grafana` (run from within the `anms/` folder).

If restarting the container does not resolve the problem, and the Grafana startup contains errors related to only having read-only access to the database, permissions on various files in the source code will need to be updated for Grafana to run.

The following permissions example is for typical a Docker system with its privileged daemon. Rootless podman may require different permissions.

For both the `docker_data/grafana_vol/` folder and the `docker_data/grafana_vol/grafana.db` file, change the group to `docker` and the permissions to `777`:

```
$ sudo chgrp docker docker_data/grafana_vol
$ sudo chgrp docker docker_data/grafana_vol/grafana.db
$ sudo chmod 777 docker_data/grafana_vol
$ sudo chmod 777 docker_data/grafana_vol/grafana.db
```

After changing these permissions, run `podman compose up grafana` again, and the Grafana container should start successfully.

4.1.2.2 Agent Registration Issues on ANMS Startup

If an Agent is not present in the Agents tab on start up, it is likely due to an error in one of the ION containers and their connection to the underlying database.

To resolve the issue, restart the ION containers using `podman compose restart n1 n2`.

4.1.2.3 New Agent Registration Issues

If registering a new Agent does not result in an update to the displayed Agents in the ANMS Agent tab, check that it has been registered to the Manager via the `nm-manager` CLI. The `nm-manager` CLI is accessible from a terminal, and this check can be done using a command such as:

```
podman compose -p anms exec amp-manager journalctl -f --unit refdm-proxy
```

If the results confirm that the Agent is registered but it still does not show on the Agents tab of the ANMS, there may be an issue with connection between the Manager and ANMS database.

This can be manually resolved by adding the Agent via the adminer DB tool that is deployed as part of the docker-compose tool at <http://localhost/>. The connection information is described in Section 4.1.2.4.

4.1.2.4 AMP Database Querying

To see what is present in the underlying AMP database, use the adminer access point.

With ANMS running, go to `localhost : 8080` and log in to the database with: - System: PostgreSQL - Server: postgres - Username: root - Password: root - Database `amp_core`

4.1.2.5 ANMS-UI is not visible at hostname:9030

This error may indicate that the `anms-ui` docker is experiencing issues receiving HTTP requests. This is most likely related to the `host` or `bind` address specified in `anms-ui/server/shared/config.py`, an environment variable that overrides this, or a firewall issue.

4.1.2.6 ANMS-UI is not visible at hostname

If `http://hostname:9030` (replace `hostname` with the server's hostname) displays the ANMS UI, but `http://hostname` does not render the same page, this indicates an issue with the frontend HTTP router.

Check the status of the `authnz` container in the compose services list. It may be necessary to restart the container using: `sudo podman compose -p anms restart authnz`.

Port numbers can be overridden through environment variables (see `.env` file). Check that ports are mapped to the expected ports and are not being blocked by your system's firewall (if applicable).

When running rootless podman the container may fail to start if the user does not have permission to bind on the configured port(s). Users typically cannot bind to low numbered ports, including 80 (http) and 443 (https). If this is the issue, try to set `AUTHNZ_PORT` and `AUTHNZ_HTTPS_PORT` to higher values and test at the specified port (*i.e.*, In `.env` set `AUTHNZ_PORT=9080` and test at `http://hostname:9080`).

Check that the `authnz` container is running with `podman compose ps`.

Logs can be viewed with `podman compose logs authnz` to identify potential issues.

4.1.2.7 Other container startup issues

Following an upgrade or failed installation step, it is possible for the system to be in an inconsistent state resulting in containers failing to start, initialize properly, or using outdated caches.

As a first debug step, it is recommended to fully restart the host system and all containers. If that fails to resolve the issue, backup any existing data and proceed with a full reset of your installation files. See Section 3.4 for details and ensure you restart the host system prior to rebuilding.

4.1.3 Advanced

4.1.3.1 Override Configuration Files with Volume Mounts

It is sometimes necessary to override configuration files or scripts built into the container environment. Volume mounts can be used to override select files or directories without rebuilding containers.

For example, to override the ION NM Manager configuration file add the following to the `volumes` section of `ion-manager` in `docker-compose.yml`. Adjust the source file to any relative or absolute path desired. For this change to take effect, you must restart all containers with `podman compose down` and then restart with `podman compose up`. Simply restarting the affected container may not refresh changes made to the compose file.

```
- ./ion/configs/simple-2-node/mgr.rc:/etc/ion.rc
```

This approach can also be used to override sample agent configurations (see `agent-compose.yml`), extend authentication settings in the selected auth container, or for developers to quickly test changes.

4.1.3.2 General Networking Issues

If services cannot be accessed from remote machines, verify that it is not being blocked by your system's firewall.

In rare cases, it may be useful to utilize Wireshark or Tshark to verify network traffic within the containers, particularly if debugging NM agents within the container network. The following is a quick guide to installing and running these tools within the container if you do not have the ability to view activity from the host system. This example is for the ion-manager container, but can be adapted to any.

To temporarily install in the ion-manager container run `podman compose exec ion-manager yum install tshark`. Alternatively, modify the ion/Dockerfile to permanently add it to your installation and rebuild. It can then be run with `podman compose exec ion-manager tshark -i any`, with tshark arguments modified as needed.

To use the GUI wireshark, install with `podman compose exec ion-manager yum install wireshark`. To easily run, create a file `novnc-compose.yml` with the definition below. Start it with `podman compose -f novnc-compose.yml up`. From the shell run `podman compose exec ion-manager bash` followed by `DISPLAY=novnc:0.0 wireshark` to launch wireshark in the VNC session accessible from <http://hostname:9081>

```
# noVNC provides browser access to network to aide debugging.
# This container is NOT intended for production usage.

networks:
  # This network is created by docker-compose.yml
  default:
    name: ${DOCKER_CTR_PREFIX}anms
    external: true

services:
  novnc:
    image: chrome-novnc
    build:
      dockerfile: novnc.Dockerfile
    environment:
      # Adjust to your screen size
      - DISPLAY_WIDTH=1920
      - DISPLAY_HEIGHT=1080
      - RUN_XTERM=yes
    ports:
      - "9081:8080"
```

See Section 4.1.3.1 for an example of volume mounting a local directory for easier saving of captures for later analysis. In this case, you may wish to mount a directory instead of a single file, such as `- ./logs:/logs` and append to the tshark command `-w /logs/log.pcap`

4.2 Contacting or Contributing

The ANMS is hosted on a GitHub repository [\[anms-source\]](#) with submodule references to several other repositories. There is a [CONTRIBUTING.md](#) document in the ANMS repository which describes detailed procedures for submitting tickets to identify defects and suggest enhancements.

Separate from the source for the ANMS proper, the ANMS Product Guide and User Guide are hosted on a GitHub repository [\[anms-docs\]](#), with its own [CONTRIBUTING.md](#) document for submitting tickets about either the Product Guide or User Guide.

While the GitHub repositories are the primary means by which users should submit detailed tickets, other inquiries can be made directly via email to the the support address dtmma-support@jhuapl.edu.