WithSecure Atlant

Contents

Chapter
1

Introduction

Topics:

WithSecure Atlant is a platform for building applications that are able to scan and detect malicious files.

Atlant provides a REST API for scanning files and managing the product configuration. Applications and services can use the API resources to analyze content using always up-to-date heuristics and statistical techniques.

Atlant provides a scanning service to protect clients against viruses and other harmful files, exploits, network-based attacks, and other security threats. It is based on the ICAP protocol and provides content scanning and reputation services to any application on the network. You can use the scanning service with a HTTP proxy that supports ICAP protocol or create your own ICAP client for it.

This documentation gives you instructions on getting started with using Atlant as well as a reference of the available API resources.

1.1 Features and benefits

The development kit offers the scanning functionality that utilizes multiple scanning engines and cloud-based file and network reputation services.

When deployed, the scanning service:

- scans files, emails and URLs for harmful content,
- protects content in real-time using cloud-based scanning,
- includes cloud-based reputation services for both files and network addresses,
- scans both HTTP requests and responses,
- includes multiple scanning engines, and
- is always up to date.

Benefits

- Multiple scanning engines provide a layered protection against harmful content.
- WithSecure Cloud technology provides real-time and always up-to-date protection for both files and URLs.
- Provides URL filtering that supports multiple different content categories.
- Updates can be configured to meet your needs.
- Easy to deploy and configure for a wide range of purposes.
- The scanning service can be easily integrated with ICAP-compatible software.
- Easy to integrate with your own solutions to provide virus and spam scanning.

1.2 Cloud-based reputation services

Security Cloud is a cloud network that houses the various databases and automated analysis systems, which support and enhance the performance of WithSecure security products. The infrastructure for this network is hosted on servers in multiple data centers around the world.

Services connect to Security Cloud to retrieve the most up-to-date details of threats seen in the wild by other protected machines, which makes the response more efficient and effective. The service queries the reputation details of all objects, such as files and URLs. These queries contain anonymous metadata about the object, such as file size and anonymized path, are sent to the Security Cloud for combined data analysis. Security Cloud does not collect IP addresses or other private information, queries are completely anonymous to maintain the privacy.

By evaluating the combined metadata with information drawn from the in-house databases and various other sources, the automated analysis systems provide a fully-informed, up-to-date risk assessment for the object, immediately blocking threats that have been seen previously by any other service or device that is connected to Security Cloud. This also removes the need to perform any further analysis of the object, which reduces the impact on the system resources that the service uses.

Security Cloud also allows Response Labs analysts to provide critical human intelligence and judgment to complement the automated systems and on-host scanning technology. In addition to creating and maintaining the rules that underpin the databases and automated analysis systems, analysts actively monitor the latest threats and research malware characteristics and behavior patterns to find the most effective ways to identify malicious programs.

1.3 About ICAP

The scanning service supports the Internet Content Adaption Protocol (ICAP). The ICAP is a protocol for sending HTTP requests and responses for another server for modifications.

The ICAP allows any client to pass messages to services for processing. The scanning service processes and modifies these messages when necessary and then sends back responses to clients, which then use the modified content instead of the original.

You can use existing ICAP compatible applications and proxies with the scanning service or create your own ICAP client to scan, remove, and block harmful content and filter unwanted web pages. The scanning service includes scan results as extended ICAP response headers and when used to scan the web traffic, it can show HTML block pages to users when they try to access infected content.

The ICAP protocol is documented in RFC 3507. For more information, see Internet Content Adaptation Protocol (ICAP).

Related concepts

ICAP extension reference on page 254

Chapter
2

Installing WithSecure Atlant

Topics:

This section outlines how to install WithSecure Atlant as well as the supported Linux distributions and required dependencies.

2.1 System requirements

The supported Linux distributions and required dependencies for Atlant are listed here.

Note: WithSecure supports only operating systems that are currently supported by their vendor. Learn more about WithSecure's operating system support policies at Product updates and supported versions. If you are interested in long-term support for a platform that vendors no longer support, contact your sales representative.

Atlant supports the following Linux distributions:

- AlmaLinux 8
- AlmaLinux 9
- Amazon Linux 2
- Debian 11
- Debian 12
- Oracle Linux 8
- Oracle Linux 9
- RHEL 8
- RHEL 9
- Rocky Linux 8
- Rocky Linux 9
- SUSE Linux Enterprise Server 15 (Service Pack 6)
- Ubuntu 20.04
- Ubuntu 22.04
- Ubuntu 24.04

Note: You can use the `--override-distro` option to install the product on a distribution that is not officially supported. For example, `--override-distro rhel:8.6`. Note that WithSecure cannot offer support for any issues with unsupported distributions.

Note: All operating systems must have TLS 1.2 configured and enabled.

Atlant requires the following packages to be installed before installing the product:

- Amazon Linux 2: `libcurl`, `python`
- AlmaLinux 8, Oracle Linux 8, RHEL 8, Rocky Linux 8: `libcurl`, `python36` or `python39`
- AlmaLinux 9, Oracle Linux 9, RHEL 9 , Rocky Linux 9: `libcurl`, `python3`
- Debian 11, Debian 12, Ubuntu 20.04, Ubuntu 22.04, Ubuntu 24.04: `libcurl4`, `python3`
- SUSE Linux Enterprise Server 15: `libcurl4`, `python3`

The hardware requirements for WithSecure Atlant depend on the use case. You can install Atlant on a computer with the following minimum requirements:

- Processor: x86-64 compatible CPU
- Memory: 4 GB RAM
- Disk space: At least 3 GB recommended

  Having sufficient swap memory is highly recommended.

  Additionally, sufficient disk space must be available for the temporary storage of all content submitted for malware analysis.

Atlant requires a connection to cloud services for database updates and for the subscription validation. If the subscription cannot be validated for two weeks, the product stops working.

Support for SELinux

The product supports Security-Enhanced Linux with the following distributions:

- AlmaLinux 8, 9 *
- Debian 11 , 12 **

- Oracle Linux 8, 9 *
- RHEL 8, 9 *
- Rocky Linux 8, 9 *

* on systems running the "targeted" SELinux system policy

** the product is compatible with the "default" SELinux system policy

If the distribution is not supported, SELinux must be disabled.

Note: Major distribution upgrades are not supported while Atlant is installed on a host with SELinux enabled. We recommend uninstalling the product before upgrading the distribution and then reinstall Atlant.

2.2 Deployment methods

Atlant supports many deployment options. This section provides a summary of the different forms of deployment and discusses when each one of them might be appropriate.

All the different deployment methods offer the same scanning capabilities but differ on the way Atlant's configuration is managed and how the initial installation process happens.

Feature	Standalone	Policy Manager Managed	Container
Installation Package	DEB or RPM file	JAR file	Container Image
REST Scanning API	X	X	X
ICAP Scanning API	X	X	X
REST Management API	X	X	
Centralized management with Policy Manager		X	
OAuth Authentication	X	X	
API Key Authentication			X

2.2.1 Standalone installation

When deployed in standalone mode, Atlant is installed from a DEB or an RPM package.

In the standalone mode, Atlant can be configured locally using `atlantctl` command-line utility or remotely using Atlant's Management API.

Clients must authenticate with Atlant before they can scan files. In the standalone mode, authentication happens using `OAuth`. The list of allowed clients is managed locally with the `atlantctl` utility.

1. Download the installation package from the WithSecure Atlant download page.
2. Run the installation package.
   - For distributions that belong to the Red Hat family, double click the `rpm` installation package or run the following command with root privileges:

     ```
     rpm -Uvh atlant-installer-XXX.x86_64.rpm
     ```

   - For Debian or Ubuntu distributions, run the following command with root privileges:

     ```
     dpkg -i atlant_installer-XXX.amd64.deb
     ```

3. Activate the product license.

   Important: Once the product is activated, it automatically starts listening on port `1344`.

The command for activating the product depends on the kind of license you are using.

- If you are using a license key (a short string similar to `AAA-BBB-CCC-DDD`) to activate the product, run the following command:

```
# /opt/f-secure/atlant/atlant/bin/activate --license-key LICENSE-KEY
```

- If you are using a file-based license to activate the product, run the following command:

```
# /opt/f-secure/atlant/atlant/bin/activate --license-file PATH-TO-LICENSE-FILE
```

If you want to keep using a specific version of the product without automatically updating to the latest available version, use the `--product-version` option to indicate the version that you want in the `activate` command.

Note: To use an HTTP proxy during the activation process, add the `--http-proxy` command line option. You can use this option in the following formats:

- `--http-proxy=host:port`: This configures the product to use the given `host` and `port` as the network proxy without any authentication. If no port number is given, port number 3128 is used by default. Example: `--http-proxy=proxy.example.com:8080`.
- `--http-proxy=username:password@host:port`: This configures the product to use the given `host` and `port` as the network proxy, and the given `username` and `password` as the credentials for authentication. Use URL encoding for any special characters in the username or password; for example, if the password contains an @ character, enter it as `%40`. Example: `--http-proxy=abc:x%40y%40z@proxy.example.com:8080`.

After activating the product license, the ICAP service is running and configured by default.

Delaying product activation

Instead of activating the product immediately during installation or deployment, you can set the product activation to take place the next time the computer is turned on.

For example, you may want to delay activation if you are installing the product within a virtual machine template that is used for multiple virtual machine instances. In such a setup, you could install the product in the template and schedule activation for the next time the virtual machine is started. This approach allows individual virtual machines to be activated separately as they are taken into use.

1. To set activation to take place the next time the computer starts up, use the `--next-boot` argument in the `activate` command.

   For example:

```
# /opt/f-secure/atlant/atlant/bin/activate --next-boot --license-key LICENSE-KEY
```

   This sets Atlant to be activated when the computer is next started.
2. The product activates automatically when the system is rebooted. No user interaction is necessary during this activation process.

2.2.2 Policy Manager managed installation

In the Policy Manager managed mode, Atlant's configuration is centrally managed using Policy Manager Console, which allows you to configure multiple instances at the same time.

In Policy Manager managed mode, clients authenticate with Atlant using `OAuth`. The list of approved clients can be managed centrally using Policy Manager Console. When a client that has been created using Policy Manager acquires an OAuth access token from one Atlant instance, the access token is also valid on other Atlant instances that are part of the same management domain. This makes Policy Manager managed setups more suitable for deployment scenarios involving load balancers, where clients might not know which specific Atlant instance they are communicating with.

Note: In Policy Manager managed mode, changes to individual instances configuration can be still made locally using the `atlantctl` command-line utility or remotely using Atlant's Management API.

The first step in deploying Atlant in the Policy Manager managed mode is to import Atlant Policy Manager installer jar file into Policy Manager Console. Once the package has been imported, a finalized installation zip package can be exported from Policy Manager Console. Follow these steps:

1. In Policy Manager Console, select **Tools** > **Installation packages** from the menu.
   This opens the **Installation packages** window.
2. Click **Import**.
3. Select the Atlant installation package that you want to use, then click **Import**.
4. Select the imported installation package in the packages list and click **Export**.
5. Enter a name and select a folder for the exported `zip` file.
   A **Remote Installation Wizard** window opens.
6. Click **Next**.
7. Enter your license keycode for the product, then click **Next**.

   Note: This must be the Policy Manager Atlant installer activation keycode. If you do not yet have this keycode, contact customer support.
8. Enter the address for your Policy Manager Server and modify the ports to use for both HTTP and HTTPS communication if necessary.
9. Click **Finish**.

Note: For more information on using Policy Manager, see the Policy Manager administrator's guide.

Related tasks

Virtual appliance

Deploying the installation package on a Linux host

Follow these instructions to deploy the installation package that you created in Policy Manager on a Linux computer.

1. Copy the exported `zip` installation package to the Linux hosts in your network.

   For any isolated hosts, also copy the `zip` content package that includes the components that are normally downloaded during installation.
2. Install the product on each host:
   a) Log in to the Linux host as `root`.
   b) For checking that the required dependencies are installed, see System requirements on page 9.
   c) Extract the installation package that you exported from Policy Manager Console to a fresh, empty directory.
   d) Run the installation command.

      The command to use depends on whether or not the host can connect to WithSecure backend systems over the network.

      - To install the latest version of the product over the network, run:

        ```
        bash atlant-pm-installer/atlant-installer
        ```

        This command installs the latest version of the product and configures it to be updated automatically.
      - To install and keep using a specific version of the product without updating it automatically, use the `--product-version` option to indicate the version that you want in the installer command.
      - To install the product on an isolated host using a content package (`withsecure-updates.zip`), run:

        ```
        bash atlant-pm-installer/atlant-installer --package=/PATH/TO/withsecure-updates.zip
        --automatic-updates=none
        ```

        On an isolated host, the product does not download or install any product or malware definition database updates automatically. If you want to install the product using a content package but

still let the product fetch updates automatically over the network, remove the `--automatic-updates=none` option from the `atlant-installer` command.

The installation has finished successfully when you see the location of the license agreement.

In centrally managed installations, the host is shown in the **Pending hosts** list in Policy Manager Console after the installation is complete.

2.2.3 Container

Atlant can be deployed as a container, which offers the same scanning and detection capabilities as other variants of Atlant. However, the deployment, configuration, and updating processes differ for the containerized version. The access to Atlant container's scanning API can be optionally restricted with API keys.

There are some differences between the Atlant container and other versions of Atlant:

- By default, the Atlant container does not include any HTTP or ICAP scanning endpoints. Other Atlant variants have a single ICAP endpoint enabled on port 1344 by default.
- The authentication is optional with the Atlant container. HTTP-based scanning endpoints can be configured to accept scan requests without any authentication. Other Atlant variants always require authentication when accessing HTTP scanning endpoints.
- The Atlant container supports API key-based authentication for HTTP-based scanning endpoints. Other Atlant variants employ OAuth 2.0 client credentials-based authentication.
- The Atlant container makes HTTPS optional. HTTP-based scanning endpoints can be served over plain HTTP. Other Atlant variants always require HTTPS.
- Configuration for the Atlant container is done through a single JSON configuration file. Other Atlant variants can be configured either locally using the `atlantctl` command or centrally using the WithSecure Policy Manager.

Configuration

The configuration of the Atlant container is done with a single `JSON` configuration file. This file can be mounted from the host system into a predefined location within the container or it can be directly copied into that location when a new container image that is based on the Atlant base image is created.

Place the `config.json` configuration inside the `/etc/opt/withsecure/atlant/config` directory within the container. If the configuration file refers to any additional files (for example, license files or certificates), copy these files to the same directory.

Important: The configuration of the container cannot be changed once it has been started.

An example configuration file for Atlant container that activates the container using a subscription key and configures a single HTTP scanning endpoint:

```
{
  "subscription_key": "ATLANT-SUBSCRIPTION-KEY",
  "scanning": {
    "http_endpoints": [
      {
        "address": "0.0.0.0",
        "port": 8081
      }
    ]
  }
}
```

See Container Configuration Reference for the full list of available settings.

Deploying the Container

Atlant container images are available on Amazon ECR Public Gallery. New versions of the image are published regularly and each release is tagged with its version number. The latest version is always available using the latest tag.

Assuming the host system has Atlant's `config.json` configuration file available in the `/etc/atlant` directory, a new Atlant container can be launched using Docker with the following command:

```
docker run -v /etc/atlant:/etc/opt/withsecure/atlant/config:ro \
           -p 8081:8081 \
           public.ecr.aws/withsecure/atlant:latest
```

This command pulls the latest Atlant container image from the repository, assuming it is not already available on the system. Then it creates a new container from the image mounting host's `/etc/atlant` directory inside the container as `/etc/opt/withsecure/atlant/config`. The configuration directory is mounted as read-only. Finally, the command exposes the port 8081 from the container.

Updates

When the Atlant container is started, it automatically updates the detection engines to the latest available versions. The container continually updates the engines to ensure it always has the most up-to-date versions. However, the Atlant container itself does not update automatically. To update to a new version of Atlant, pull a new container image version from the repository and deploy it. You have a complete control over when to deploy updates, and the container remains mostly stateless and immutable.

2.3 Creating the content package for isolated environments

To install Atlant in an isolated environment with limited network connectivity, you need to prepare an additional content package for the product installer.

Note: To generate the content package, use a computer that has network access to WithSecure servers and either Policy Manager Server (Linux or Windows) installed or a separately available `fspm-definitions-update-tool` downloaded (Linux).

The installation package that you create using Policy Manager Console installs the latest available version of the product by downloading it over the network. Skip these steps if you are only deploying the installation package to hosts that can connect to the network to download data during installation.

The additional package for isolated hosts provides the necessary components that are usually downloaded from the network during installation. Use this with the installation package on each isolated host to which you deploy the product.

Follow these instructions to create the content package.

Note: These instructions are for Linux. You can find the command path and the syntax for Windows in the Policy Manager administrator's guide.

1. Make sure that you have the current version of `fspm-definitions-update-tool`.

   If you are not using Policy Manager 16, or if the version of your `fspm-definitions-update-tool` is uncertain, download the latest version from the product support page.

   Note: If you are using `fspm-definitions-update-tool` from Policy Manager Server 16, it is up-to-date.

2. Open a command line.
3. Select the directory where you want to extract the Policy Manager definitions update tool.

   Make sure that you use an existing directory. The examples in these steps use `<DIR>` as a placeholder for this directory.

4. Extract the definitions update tool.
   - If you have Policy Manager Server, run the following command:

     ```
     /opt/f-secure/fspms/bin/prepare-fspm-definitions-update-tool <DIR>
     ```

   - If you have downloaded the `fspm-definitions-update-tool.tar.gz` file, run the following command:

     ```
     tar -C <DIR> -xf /PATH/TO/fspm-definitions-update-tool.tar.gz
     ```

5. Open the following file in a text editor:
<DIR>/fspm-definitions-update-tool/conf/channels.json.
6. Replace the file with the following content:

```
[
    "fsbspamd-100-linux-x86_64",
    "aqualnx64",
    "fmlibunix64",
    "hydra-linux64",
    "baseguard-100-linux-x86_64",
    "fsbg-100-linux-x86_64",
    "atlant-100-linux-x86_64"
]
```

7. Make sure that the <DIR>/fspm-definitions-update-tool/data/ directory is empty or does not exist by running the following command:

```
rm -rf <DIR>/fspm-definitions-update-tool/data/
```

8. Run the following command to create the content package:

```
<DIR>/fspm-definitions-update-tool/fspm-definitions-update-tool
```

This command creates two ZIP files, f-secure-updates.zip and withsecure-updates.zip, to the <DIR>/fspm-definitions-update-tool/data/ directory.

Use the <DIR>/fspm-definitions-update-tool/data/withsecure-updates.zip for creating new installations of the product (for more information, see Deploying the installation package on a Linux host on page 12). The f-secure-updates.zip file is not needed for creating new installations.

Note: To create another content package with newer updates, repeat steps 6 and 7. An empty data directory is required to create packages that are valid for use with WithSecure Atlant.

2.4 Pinning the product version

By pinning the product version, the installation is locked to a specific version of the product. The pinned version still receives new engine and definition updates but does not install new product version updates.

In general, each pinnable product version is supported for one year from the release date of the subsequent pinnable version. Exceptions may be outlined in the change log of the pinnable version release.

2.5 Updating the product manually using an update archive

In isolated environments with limited network connectivity, you can update the product and virus definition databases by creating update archives and deploying them on the hosts.

Note: To generate update archives, use a computer that has network access to WithSecure servers and either Policy Manager Server (Linux or Windows) installed or a separately available fspm-definitions-update-tool downloaded (Linux).

Before you can install the updates, create update archives first by following the instructions in Creating the content package for isolated environments.

Follow these instructions to install the updates.

1. Log on to the host on which you wish to install the updates.
2. Run the command /opt/f-secure/atlant/fsbg/bin/withsecure-migrator status. Check the last line of the command's output to choose the ZIP file to use for the update installation.
   - If the system reports that the command does not exist, or the command reports not-performed or not-ready, use the f-secure-updates.zip archive for installing updates on the host.
   - If the command reports migrated, use the withsecure-updates.zip file to install the updates.

3. Copy the appropriate ZIP file to the host and run the following command as the root user:
   `/opt/f-secure/atlant/fsbg/bin/offline-update /PATH/TO/ZIP-FILE`

   The command prepares the updates ready for installation. It displays the channel names while it extracts the updates from the archive and registers them for installation.

   Once registered, updates are installed right away, regardless of any schedule set in the product's settings.
4. Repeat the previous steps to copy and install the updates on each host where you want to install them.

Related concepts

Starting and stopping services on page 248
The product contains a number of background services, which you can control using the `/opt/f-secure/atlant/fsbg/bin/master-switch` command.

Related tasks

Updating the product manually using an update archive on page 15
In isolated environments with limited network connectivity, you can update the product and virus definition databases by creating update archives and deploying them on the hosts.

2.6 Uninstallation

This topic describes how to uninstall the product.

1. Log in to the Linux host as `root`.
2. Run the uninstallation command:

   Note: The command that is given here is for the current version of the product. If you are uninstalling an older version, use `f-secure-atlant` as the package name instead.

   - RHEL-based distributions: `rpm -e atlant-installer`
   - Debian-based distributions: `dpkg -r atlant-installer`

Chapter
3

Setting up WithSecure Atlant

Topics:

This section outlines the steps needed for a minimal Atlant setup.

The steps given here assume that you have already activated Atlant. You also need a valid TLS certificate and corresponding PEM key file.

3.1 Set up HTTPS

All communication with Atlant's REST API happens over secure HTTPS connections, so the first step in setting up Atlant is to provide it with a TLS certificate and a key.

Note: The command path given here is for the current version of the product. If you are using an older version, use `/opt/f-secure/atlant/bin/<command>` as the path instead.

Run the following command to set up TLS:

```
/opt/f-secure/atlant/atlant/bin/atlantctl set tls '{"certificate": "/root/cert.pem", "key":
"/root/key.pem"}'
```

This command example assumes that the path for your certificate is `/root/cert.pem` and the path for the key is `/root/key.pem`.

Note: Make sure that the certificate is trusted by the clients.

3.1.1 About TLS/SSL certificates

A TLS certificate is required for handling the HTTPS communication with Atlant's REST API.

Certificates that are used by public web servers are usually created by some known, trusted certificate authority (CA). For internal servers, you can be your own trusted authority and use a self-signed root certificate that all internal clients trust. As all other certificates rely on the trust on the root certificate, make sure that it is very well secured.

Typically, root certificates are used to sign intermediate CA certificates, which are then used to sign other certificates. Signing certificates this way means that you do not need to access the root private key very often, which keeps it more secure.

Certificates can be revoked if they get compromised. Web browsers should warn users about revoked certificates and even block the access to web sites using them. However, web browsers do not always receive the information about revoked certificates reliably so they may still show that the certificate is valid even when it has been revoked. To prevent this, use short validity times for certificates and replace them often. If a certificate gets compromised, it invalidates itself when its validity expires.

When attaching a certificate to a server, you usually provide a certificate chain and not just the leaf certificate. A certificate chain consists of the needed certificate and the certificates that were used to sign it up to the root certificate. The chain of trust ensures that the leaf certificate and all the intermediate certificates are trusted because the root certificate is trusted.

3.2 Set up Atlant services

When your TLS certificate is set up, the next step is to enable the services that clients use.

Note: The command path given here is for the current version of the product. If you are using an older version, use `/opt/f-secure/atlant/bin/<command>` as the path instead.

Clients can authenticate with Atlant's internal authorization server using OAuth 2.0. Before that, you have to bind the internal authorization server to an address to enable it.

Note: For more complicated setups, you can configure Atlant to use an external authorization server. This can be useful for scenarios where you want to use an external client database.

1. Run the following command to bind the internal authorization server to an address:

   ```
   /opt/f-secure/atlant/atlant/bin/atlantctl add authorization https_endpoints '{"address":
   "localhost", "port": 8081}'
   ```

   This command enables the authorization server to listen on port `8081`. The server uses the TLS certificate specified previously.

2. Run the following command to enable the management server, which allows clients to change Atlant's configuration:

```
/opt/f-secure/atlant/atlant/bin/atlantctl add management https_endpoints '{"address":
"localhost", "port": 8082}'
```

   This command enables the management server to listen on port `8082`. The server uses the TLS certificate specified previously.
3. Run the following command to enable Atlant's scanning server:

```
/opt/f-secure/atlant/atlant/bin/atlantctl add scanning https_endpoints '{"address": "localhost",
 "port": 8083}'
```

   The scanning server is required for clients to scan files. This command enables the scanning server to listen on port `8083`. The server uses the TLS certificate specified previously.

3.3 Create the first client

After you have enabled all the services, you can create the first API client.

Note: The command path given here is for the current version of the product. If you are using an older version, use `/opt/f-secure/atlant/bin/<command>` as the path instead.

Run the following command to create the initial client:

```
/opt/f-secure/atlant/atlant/bin/atlantctl client create '{"scopes": ["management", "scan"]}'
```

This command adds a new client that can inspect and change the Atlant configuration and scan files. You can also specify only the `scan` scope to create a client that scans files but cannot edit the Atlant configuration. The command will return the following type of response:

```
{
  "client_id": "5dbfe97de42bf53a0ae73bff9eba4ecb",
  "client_secret": "87e7b3a61e7fdcdfc03162fdcab82a942b9c62715ff3d612e15adf32d1b1500b"
}
```

The response contains the client ID and the secret of the new client. Store the client secret as this is the only time it is given to you. These are the credentials that the client uses to request access tokens from the authorization server.

Chapter
4

API reference

Topics:

This section contains the details for all API resources available in WithSecure Atlant.

WithSecure Atlant provides a comprehensive REST API that can be used to perform most tasks with the product. The API is divided into three different services: Scanning API, Management API, and Authentication API. Each API is implemented as a separate service and must be made available on separate ports.

Examples for use with the Atlant API are available online at https://github.com/WithSecureOpenSource/atlant-api.

4.1 Authentication API

Authentication API is used to authenticate with Atlant. Authentication is necessary before the scanning API or management API can be used.

4.1.1 OAuth-based Authentication

Clients must authenticate with Atlant before they scan files or manage configurations using the REST API. Atlant supports the OAuth 2.0 client credentials-based authentication flow, where each client has its own client ID and secret. Clients use these credentials to obtain short-lived access tokens to access authenticated APIs.

Note: OAuth-based authentication is not available with Atlant container. See API key-based Authentication for how authentication can be configured with Atlant container.

Before a client can authenticate with Atlant, it requires a client ID and a client secret. These are created either locally on the Atlant instance using the included `atlantctl` command-line utility or centrally using Policy Manager Console if Atlant is managed with Policy Manager.

Both the `atlantctl` utility and Policy Manager Console can be used concurrently to create clients. The `atlantctl` utility is used for creating clients local to a specific Atlant instance, while Policy Manager Console can be used simultaneously to centrally create additional clients with access to the same instance.

Note: `atlantctl` can only delete clients created with `atlantctl`, and Policy Manager Console can only delete clients created with Policy Manager Console.

Authentication

OAuth 2.0 token endpoint for requesting new access tokens. Client credentials is the only supported grant type.

All requests to Atlant REST API require authentication. When making a request to the API, client must include `Authorization` header containing a valid access token. Client can request an access token from an authorization server. Atlant installation can use either the internal authorization server that ships with the product or an external authorization server.

The authentication request must include an `audience` parameter, which must match with the client:

- For clients that have been created locally either by `atlantctl` or with Atlant's REST API, the audience parameter is `f-secure-atlant`
- For clients that have been created using Policy Manager Console, the audience parameter is `policy-manager`

Note: `atlantctl` and REST API handle only clients that have been created locally. External clients or clients that have been created with Policy Manager are not visible to atlantctl or REST API.

Resource URI: `/api/token/v1`

Methods: `POST`

Data type: `object`

POST

Request a new access token using client credentials grant type. Response will contain access token that must be included in the `Authorization` header when making requests to other Atlant APIs. By default, the returned token will enable token bearer to access all the scopes available to the client.

Example request:

```
POST /api/token/v1 HTTP/1.1
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&audience=f-secure-atlant&client_id=CLIENT_ID&client_secret=CLIENT_SECRET
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
  "access_token": "ACCESS_TOKEN",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

4.1.2 API key-based authentication

The Atlant container supports API key-based authentication. In API key-based authentication, clients must include a secret API key with each request using the `X-Api-Key` HTTP header.

Note: API key-based authentication is currently only available with the Atlant container.

Configuring API key authentication

API key-based authentication is configured by specifying the required API key using the ***authentication/api_key*** configuration property. If the property is set, then scanning requests must include the specified API key using the ***X-Api-Key*** header.

The API keys must be between 30 and 128 bytes in size.

A sample configuration file for Atlant container that enables API key based authentication for the scanning service:

```
{
  "subscription_key": "ATLANT-LICENSE-KEY",
  "scanning": {
    "http_endpoints": [
      {
        "address": "0.0.0.0",
        "port": 8081
      }
    ]
  },
  "authentication": {
    "api_key": "f7ff02989217f102353eac2289728e380d5c15b233f828e1eb0028fabdd97ce7"
  },
  "tls": {
    "certificate": "atlant.pem",
    "key": "atlant.key"
  }
}
```

This configuration activates the container using a subscription key, configures a single HTTP-based scanning endpoint, enables TLS for the scanning endpoint, and enables API key-based authentication by specifying the API key (in this example, `f7ff02989217f102353eac2289728e380d5c15b233f828e1eb0028fabdd97ce7`) that clients must include in their requests.

Authenticating with an API key

If Atlant container has been configured to use API Key-based authentication, each request to the scanning service must include the API key using the `X-Api-Key` header.

An example command that scans a file using the `curl` command-line utility.

```
curl -H "X-Api-Key: f7ff02989217f102353eac2289728e380d5c15b233f828e1eb0028fabdd97ce7" \
     -F 'metadata={};type=application/json' \
     -F 'data=@/path/to/some/file' \
     https://${ATLANT_HOST}:${ATLANT_SCAN_PORT}/api/scan/v1
```

The command authenticates with the Atlant instance by passing API key via header using `curl`'s `-H` option.

This example assumes that the address and the port for Atlant's scanning service have been stored respectively in the ***ATLANT_HOST*** and ***ATLANT_SCAN_PORT*** shell variables.

4.2 Scanning API

Scanning API can be used to scan files and other types of content.

4.2.1 Scanning overview

WithSecure Atlant offers a wide range scanning options through its REST API.

Using Atlant's scanning API, you are able to:

- Scan files for malicious content.
- Scan files using only the hashes of their content. In this mode, Atlant determines whether the file is malicious without having access to the full contents of the file.
- Scan emails for spam.
- Scan URLs and classify them based on the type of content available on the URL. Additionally, Atlant can disallow certain categories of content.

All the different types of scans are performed by making `HTTP POST` requests to the API endpoint at ***/api/scan/v1***. The body of the scan request must be encoded as ***multipart/form-data*** and contain either one or two parts.

The first part of the scanning request body must always be present and contain a `JSON` object with scan settings. The default values are used if you provide an empty `JSON` object. This first part of the request should use ***application/json*** content type. Change the contents of the scan settings `JSON` object to perform different types of scans.

The second part of the scanning request body is optional and - if present - must contain the scanned content in binary format.

Accessing the REST scanning API may require authentication depending on how Atlant has been configured. In the case of non-container versions of Atlant, authentication is always mandatory and uses OAuth 2.0 client credentials flow. With Atlant container, authentication is optional and uses an API key based scheme, if enabled.

ICAP-based scanning

Atlant also supports scanning via a standardized protocol ICAP. Using ICAP, Atlant can be integrated with existing solutions that support it. Atlant expands the ICAP protocol with additional capabilities using a number of custom protocol extensions.

Scanning files

Files are scanned using Atlant's REST API by making `HTTP POST` requests to the API endpoint located at ***/api/scan/v1***. The body of the scan request must be encoded as ***multipart/form-data*** and, in the case of file scans, must contain two parts.

The first part of the scanning request body must always be present and contain a `JSON` object with scan settings. The default values are used if you provide an empty `JSON` object. This first part of the request should use ***application/json*** content type. Change the contents of the scan settings `JSON` object to perform different types of scans.

The second part of the scanning request body must contain the scanned content in binary format.

A basic file scan can be performed with the following `curl` command, assuming Atlant uses OAuth based authentication:

```
curl -H "Authorization: Bearer $ACCESS_TOKEN" \
     -F 'metadata={};type=application/json' \
     -F 'data=@/path/to/some/file' \
     https://${ATLANT_HOST}:${ATLANT_SCAN_PORT}/api/scan/v1
```

- The example command assumes that the OAuth access token has been stored in `ACCESS_TOKEN` shell variable. The command includes the access token into request's `Authorization` header.

- The command specifies that the `POST` request body should contain two parts using curls `-F` command-line option.
- The first part of the request specifies scan metadata as an empty json object {} and states that the part should use application/json content type.

  Note: See Scan request reference for more details.

- The second part of the request body contains the file contents to be scanned. The example includes the contents of the file `/path/to/some/file` directly into the request by using curl's `@FILE` syntax.

  Note: See cURL manual for more details on how to include file contents into a request.

- Additionally, the example assumes that the scanning service address has been stored in ***ATLANT_HOST*** shell variable and that the port of the scanning service has been stored in ***ATLANT_SCAN_PORT*** shell variable.

If API key-based authentication is used, instead of specifying the OAuth access token using the Authorization request header, the API key should be included into the request using the X-Api-Key request header:

```
curl -H "X-Api-Key: $API_KEY" \
     -F 'metadata={};type=application/json' \
     -F 'data=@/path/to/some/file' \
     https://${ATLANT_HOST}:${ATLANT_SCAN_PORT}/api/scan/v1
```

If Atlant has been configured to use no authentication, `Authorization` or `X-Api-Key` headers do not need to be specified.

Scan responses

Atlant responds to scan requests with HTTP status codes:

Status code	Description
200	Scan was successful and fully completed.
202	Scan was successful but the results are not ready yet.
401	Either Atlant's license is not valid or authentication failed. If OAuth authentication is used, this could also indicate that the access token has expired.
500	Internal error.

If the scan was successful, the response body is a `JSON` object.

An example scan response for a scan which did not find any detections:

```
{
  "scan_result": "clean",
  "detections": [],
  "warnings": {
    "corrupted": false,
    "encrypted": false,
    "max_nested": false,
    "max_results": false,
    "max_scan_time": false,
    "need_content": false
  },
  "status": "complete"
}
```

- The scan response object has the result of the scan in `scan_result` property, in this case indicating that nothing malicious was found.
- The response includes an array of detections, which is empty in this example.
- Additionally, the response contains a number of warning flags, all of which are set to false, indicating that no warnings were emitted.

  Note: See Scan response specification for more information on the scan response format.

- Finally, the status field indicates that the scan was fully completed.

Scanning with a hash

Files can be scanned using only the SHA-1 hash of their content. This mode of scanning uses the hash of the file to check if the file is known to be clean or malicious. Some files cannot be scanned using the hash alone, in which case Atlant indicates that the complete file should be sent for scanning.

Similarly to the regular file scanning, scanning a file using its hash is made with an `HTTP POST` request to the API endpoint located at ***/api/scan/v1***. The body of the scan request must be encoded as ***multipart/form-data*** and contain just a single part with the scan settings as a `JSON` object. This part must use application/json content type.

To scan a file using a hash of its contents, the scan settings json object must include the SHA-1 hash of the file.

An example of scan settings that specify the hash of the file to be identified using the sha1 property of the ***content_meta*** object:

```
{
    "content_meta": {
        "sha1": "da39a3ee5e6b4b0d3255bfef95601890afd80709"
    }
}
```

An example `curl` command of a scan request with this hash:

```
curl -H "Authorization: Bearer $ACCESS_TOKEN" \
     -F 'metadata=@metadata.json;type=application/json' \
     https://${ATLANT_HOST}:${ATLANT_SCAN_PORT}/api/scan/v1
```

- The example command assumes that the OAuth access token has been stored in ***ACCESS_TOKEN*** shell variable. The command includes the access token into request's Authorization header.
- The command specifies that the `POST` request body should contain one part using curls `-F` command-line option. The example assumes that the scan settings with the file hash have been saved in a file named `metadata.json`. The command includes the contents of that file to the first part of the request body using curl's `@FILE` syntax.

  Note: See cURL manual for more details on how to include file contents into a request.

- Additionally, the example assumes that the scanning service address has been stored in ***ATLANT_HOST*** shell variable and that the port of the scanning service has been stored in ***ATLANT_SCAN_PORT*** shell variable.

An example response to this request:

```
{
  "scan_result": "whitelisted",
  "detections": [],
  "warnings": {
    "corrupted": false,
    "encrypted": false,
    "max_nested": false,
    "max_results": false,
    "max_scan_time": false,
    "need_content": false
  },
  "status": "complete"
}
```

In this example, Atlant classified the file as ***whitelisted***.

Detecting spam

The scanning service can be used to detect spam emails.

To detect spam, send the contents of the email to Atlant along with optional metadata about the sender and the recipients of the email.

Similarly to the regular file scanning, detecting spam is made with an `HTTP POST` request to the API endpoint located at ***/api/scan/v1***. The body of the scan request must be encoded as ***multipart/form-data*** and contain two parts.

The first part of the request body must be a `json` object with settings for the scan. The part must use the content type ***application/json***. For spam detection, the ***scan_settings*** object must have the ***antispam*** property set to ***true***. The detection accuracy can be improved by providing additional metadata about the message by using the ***ip***, ***sender***, and ***recipients*** properties of the ***content_meta*** object.

Note: See Scan request reference for more details.

The second part of the request body must contain the contents of the email to be scanned in binary format.

An example of scan settings to detect spam:

```
{
    "scan_settings": {
        "antispam": true
    },
    "content_meta": {
       "ip": "127.0.0.1",
       "sender": "john@example.com",
       "recipients": [
          "jane@example.com",
          "bob@example.com"
       ]
    }
}
```

- In this example, the scan request turns on spam detection by setting the ***antispam*** property to ***true*** inside the ***scan_settings*** object.
- The request supplies Atlant with additional metadata about the message by specifying message's source IP address using the ***ip*** property and details about the sender and the recipients of the message using their respective properties.

An example `curl` command of a scan request:

```
curl -H "Authorization: Bearer $ACCESS_TOKEN" \
     -F 'metadata=@metadata.json;type=application/json' \
     -F 'data=@/path/to/some/email.txt' \
     https://${ATLANT_HOST}:${ATLANT_SCAN_PORT}/api/scan/v1
```

- The example command assumes that the OAuth access token has been stored in ***ACCESS_TOKEN*** shell variable. The command includes the access token into request's `Authorization` header.
- The command specifies that the `POST` request body should contain one part using curls `-F` command-line option. The example assumes that the scan settings with the file hash have been saved in a file named `metadata.json` and that the email to be scanned is stored in the `/path/to/some/email.txt` file. The command includes the contents of these files to the request body using curl's `@FILE` syntax.

  Note: See cURL manual for more details on how to include file contents into a request.

- Additionally, the example assumes that the scanning service address has been stored in ***ATLANT_HOST*** shell variable and that the port of the scanning service has been stored in ***ATLANT_SCAN_PORT*** shell variable.

An example response to this request:

```
{
  "scan_result": "spam",
  "detections": [
    {
      "category": "spam",
      "name": "Email_Spam"
    }
  ],
  "warnings": {
    "corrupted": false,
    "encrypted": false,
    "max_nested": false,
    "max_results": false,
    "max_scan_time": false,
    "need_content": false
```

```
  },
  "status": "complete"
}
```

- In this example, the ***scan_result*** property of the response indicates that the message was classified as ***spam*** and the detections array confirms this by listing a single ***Email_Spam*** detection.

Scanning URLs

The scanning service can detect harmful URL and classify URLs into categories based on their content. In addition, certain categories of URLs can be blocked with rules. The URL scanning can be used to scan individual URLs or to scan URLs embedded in documents.

Similarly to the regular file scanning, scanning a URL is made with an `HTTP POST` request to the API endpoint located at ***/api/scan/v1***. The body of the scan request must be encoded as ***multipart/form-data*** and, when scanning an individual URL, contain just a single part with the scan settings as a `JSON` object. This part must use ***application/json*** content type.

To classify a URL, the scan settings `JSON` object must include the URL.

An example scan settings object that includes the URL to be classified using the ***uri*** property of the ***content_meta*** object:

```
{
    "content_meta": {
        "uri": "https://google.com"
    }
}
```

An example `curl` command of a scan request:

```
curl -H "Authorization: Bearer $ACCESS_TOKEN" \
     -F 'metadata=@metadata.json;type=application/json' \
     https://${ATLANT_HOST}:${ATLANT_SCAN_PORT}/api/scan/v1
```

- The example command assumes that the OAuth access token has been stored in ***ACCESS_TOKEN*** shell variable. The command includes the access token into request's `Authorization` header.
- The command specifies that the `POST` request body should contain one part using curls `-F` command-line option. The example assumes that the scan settings with the file hash have been saved in a file named `metadata.json`. The command includes the contents of this file to the request body using curl's `@FILE` syntax.

  Note: See cURL manual for more details on how to include file contents into a request.

- Additionally, the example assumes that the scanning service address has been stored in ***ATLANT_HOST*** shell variable and that the port of the scanning service has been stored in ***ATLANT_SCAN_PORT*** shell variable.

An example response to this request:

```
{
  "scan_result": "clean",
  "detections": [],
  "uri_categories": [
    "search_engines"
  ],
  "warnings": {
    "corrupted": false,
    "encrypted": false,
    "max_nested": false,
    "max_results": false,
    "max_scan_time": false,
    "need_content": true
  },
  "status": "complete"
}
```

- In this example, Atlant classifies the URL and provides a list of categories assigned to it in the ***uri_categories*** property of the response. This property may not be present every time if the category information for the URL is not available.

URL categories can be marked as forbidden. If the category of a scanned URL is forbidden, the scan result is set to ***harmful***. To specify forbidden URL categories, use the ***forbidden_uri_categories*** property of the ***scan_settings*** object.

An example scan request that specifies that URLs in the category ***search_engines*** should be forbidden:

```
{
    "scan_settings": {
        "forbidden_uri_categories": [
            "search_engines"
        ]
    },
    "content_meta": {
        "uri": "https://google.com"
    }
}
```

An example response to this scan request:

```
{
  "scan_result": "harmful",
  "detections": [
    {
      "category": "harmful",
      "name": "ORSP NRS: forbidden category search_engines"
    }
  ],
  "uri_categories": [
    "search_engines"
  ],
  "warnings": {
    "corrupted": false,
    "encrypted": false,
    "max_nested": false,
    "max_results": false,
    "max_scan_time": false,
    "need_content": true
  },
  "status": "complete"
}
```

- In this example, the scan result indicates that the URL as harmful because it belongs to one of the forbidden categories.

Scanning URLs embedded in documents

Atlant can scan URLs embedded in documents. In this mode, Atlant looks for every URL that is present in the document and scans it.

This feature is currently available for the following types of documents:

- Plain text files (text/plain),
- HTML files (text/html), and
- RTF files (text/rtf, application/rtf, application/x-rtf)

When scanning URLs embedded in a document, the scan request body should contain two parts. The first part of the request should contain scan settings in a JSON format and the second part should contain the scanned document.

To scan for embedded URLs, set the ***scan_embedded_urls*** property in the ***scan_settings*** object to `true` and specify the content type of document with the ***content_type*** property inside the ***content_meta*** object.

For HTML and plain text files, specify the used character encoding with the ***charset*** property of the ***content_meta*** object. Use the `iconv --list` command to view the list of character encodings that can be used with the ***charset*** property. If the ***charset*** is not specified, Atlant assumes that the input uses ***latin1*** encoding.

Handling slow scans

In some situations, Atlant may not be able to complete the scan right away. In such cases, Atlant provides a partial response that includes information on how the client can check for more results at a later time.

When a client sends a scan request to the Atlant API by using an `HTTP POST` request to the endpoint ***/api/scan/v1***, there are cases where the scan cannot be completed immediately. In such situations, Atlant responds with an HTTP status code `202`, which means that the scan has been accepted for processing.

The response from Atlant includes two additional HTTP headers:

- The ***Location*** header contains a server path that the client can use to request additional results of the scan.
- The ***Retry-After*** header gives a recommended waiting time in seconds that the client should wait before requesting additional results.

In addition, the `JSON` object in the response body includes a ***Status*** property that is set to ***pending***, which indicated that the scan is still being processed and the client should wait for further updates.

When the client receives a `202` response, it is expected to initiate a polling process for obtaining the full scan result. This is done by sending repeated `HTTP GET` requests to the path that is provided in the ***Location*** response header. Before initiating each polling request, the client should wait for the number of seconds mentioned in the ***Retry-After*** header.

The client should keep polling for results until the response received from the API indicates that the ***Status*** property has been set to ***complete***. This indicates that the scan has finished processing and the final results are available.

4.2.2 Scan request reference

With Atlant's REST API, scans are performed by making `HTTP POST` requests to the API endpoint located at ***/api/scan/v1***. The body of the scan request must be encoded as `multipart/form-data` and contain one or two parts.

The first part must always be present and contain scan settings as a `JSON` object. The part should use ***application/json*** content type. A json schema description of the scan settings object:

```
{
  "type": "object",
  "properties": {
    "scan_settings": {
      "description": "Scan settings.",
      "type": "object",
      "properties": {
        "scan_archives": {
          "description": "Enable archive scanning.",
          "type": "boolean",
          "default": true
        },
        "max_nested": {
          "description": "Scan nested archives up to this level.",
          "type": "integer",
          "minimum": 0,
          "default": 5
        },
        "max_scan_time": {
          "description": "Maximum scan time in seconds.",
          "type": "integer",
          "minimum": 1,
          "default": 600
        },
        "stop_on_first": {
          "description": "Stop scanning after the first detection.",
          "type": "boolean",
          "default": true
        },
        "allow_upstream_metadata": {
          "description": "Allow upstreaming of file metadata.",
          "type": "boolean",
          "default": true
        },
        "antispam": {
          "description": "Scan content for spam.",
          "type": "boolean",
```

```
          "default": false
        },
        "scan_embedded_urls": {
          "description": "Scan URLs embedded in the content.",
          "type": "boolean",
          "default": false
        },
        "security_cloud": {
          "description": "Use advanced cloud analysis.",
          "type": "object",
          "properties": {
            "allow_upstream_application_files": {
              "description": "Allow upstreaming of application files.",
              "type": "boolean",
              "default": true
            },
            "allow_upstream_data_files": {
              "description": "Allow upstreaming of data files.",
              "type": "boolean",
              "default": false
            }
          }
        },
        "forbidden_uri_categories": {
          "description": "Detect if given \"uri\" metadata or found embedded urls match any of
these categories.",
          "type": "array",
          "items": {
            "type": "string"
          }
        }
      }
    },
    "content_meta": {
      "description": "Content metadata.",
      "type": "object",
      "properties": {
        "sha1": {
          "description": "The content sha1 digest.",
          "type": "string"
        },
        "uri": {
          "description": "The content source URI.",
          "type": "string"
        },
        "content_length": {
          "description": "The content length.",
          "type": "integer"
        },
        "content_type": {
          "description": "The content type.",
          "type": "string"G
        },
        "charset": {
          "description": "The content character set.",
          "type": "string"
        },
        "ip": {
          "description": "The content source IP.",
          "type": "string"
        },
        "sender": {
          "description": "The email address of the sender of the content.",
          "type": "string"
        },
        "recipients": {
          "description": "The email addresses of the recipients of the content.",
          "type": "array",
          "items": {
            "type": "string"
          }
        }
      }
    }
  }
}
```

Scan settings

The ***scan_settings*** property of the scan request JSON object specifies settings for the scan. All ***scan_settings*** properties are optional. Default values are used if the request object does not specify the ***scan_settings*** property.

scan_archives	A boolean setting that controls whether archives are extracted when they are scanned. If the archive scanning is enabled, Atlant extracts the archive and analyzes all archived items individually. If the scanned archive is encrypted, the scan response has the ***encrypted*** warning set to ***true***. If the scanned archive is corrupted and cannot be extracted, the scan response has the ***corrupted*** warning set to ***true***.
max_nested	An integer setting that controls the maximum number of nested archives, which are extracted if the archive scanning is enabled. The setting is used only if the archive scanning is enabled. If archive nesting limit is reached, the scan response has the ***max_nested*** warning set to ***true***.
max_scan_time	An integer setting that controls the maximum time that is spent scanning the content. **Note:** This time limit does not include the time that it takes for Atlant to receive the content to be scanned. The timer starts counting when Atlant has received the entire file. If the ***max_scan_time*** limit is hit, the scan response has the ***max_scan_time*** warning set to ***true***.
stop_on_first	A boolean setting controls whether scanning should be stopped when the first detection is made. During the archive scanning, it might not be necessary to continue scanning once the first detection is made. To get a full list of results, set ***stop_on_first*** to false.
allow_upstream_metadata	A boolean setting controls whether Atlant is allowed to send metadata about the scan to the cloud for additional analysis.
antispam	A boolean setting controls whether the spam detection is enabled.
scan_embedded_urls	A boolean setting controls whether URLs that are embedded in the request content are analyzed.
security_cloud	Enables the advanced cloud analysis using Security Cloud. If the property is left out, Security Cloud is not used.
forbidden_uri_categories	A list of strings representing URL content categories that are blocked during the URL scanning.

Content metadata

The ***content_meta*** property of the scan request JSON object specifies additional metadata about the scanned content. If present, the property should be a JSON object. All ***content_meta*** properties are optional.

sha1	A string specifying the SHA-1 hash of the content. Used for scanning with a hash.
uri	A string specifying the URL for the content. Used for scanning URLs.
content_length	An integer specifying the size of the content in bytes.
content_type	A string specifying the content type. Used when scanning URLs embedded in the content.
charset	A string specifying the character encoding that is used in the content.
ip	A string specifying the IP address of the content source or originating client.
sender	A string specifying the email address of the sender of the content.
recipients	An array of strings specifying the email addresses of the recipients of the content.

4.2.3 Scan response reference


```
{
  "type": "object",
  "properties": {
    "scan_result": {
      "description": "Scan result.",
      "type": "string",
      "enum": [
        "clean",
        "whitelisted",
        "suspicious",
        "PUA",
        "UA",
        "harmful",
        "spam"
      ]
    },
    "detections": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "category": {
            "description": "Detection category.",
            "type": "string",
            "enum": [
              "suspicious",
              "PUA",
              "UA",
              "harmful",
              "spam"
            ]
          },
          "name": {
            "description": "Detection name.",
            "type": "string"
          },
          "member_name": {
            "description": "Logical path to the archive member that triggered the detection.",
            "type": "string"
          }
        },
        "required": [
          "category",
          "name"
        ]
      }
    },
    "uri_categories": {
      "description": "The known categories of the content URI specified in the metadata.",
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "status": {
      "description": "Task status.",
      "type": "string",
      "enum": [
        "pending",
        "complete"
      ]
    },
    "warnings": {
      "type": "object",
      "properties": {
        "corrupted": {
          "description": "The server was unable to fully analyze the content because some data
is corrupted",
          "type": "boolean"
        },
        "encrypted": {
          "description": "The server was unable to fully analyze the content because some data
is encrypted",
          "type": "boolean"
        },
        "max_nested": {
         "description": "The server was unable to fully analyze the content with the given level
 limit for nested archives",
          "type": "boolean"
        },
        "max_results": {
```

```
        "description": "The server was unable to fully analyze the content since it has reached
 the maximum number of detections",
          "type": "boolean"
        },
        "max_scan_time": {
          "description": "The server was unable to fully analyze the content with the given scan
 time limit",
          "type": "boolean"
        },
        "need_content": {
          "description": "The client should re-send the request with the content included",
          "type": "boolean"
        }
      }
    }
  },
  "required": [
    "scan_result",
    "status"
  ]
}
```

4.2.4 Container configuration reference

Atlant container is configured using a single JSON file.

Below is a json schema description of the configuration file format:

```
{
  "type": "object",
  "oneOf": [
    {
      "required": "subscription_key"
    },
    {
      "required": "license_file"
    }
  ],
  "properties": {
    "subscription_key": {
      "type": "string"
    },
    "license_file": {
      "type": "string"
    },
    "http_proxy": {
      "type": "object",
      "properties": {
        "host": {
          "type": "string"
        },
        "port": {
          "type": "integer",
          "minimum": 0,
          "maximum": 65535
        },
        "username": {
          "type": "string"
        },
        "password": {
          "type": "string"
        }
      },
      "required": [
        "host",
        "port"
      ]
    },
    "scanning": {
      "type": "object",
      "properties": {
        "icap_endpoints": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "address": {
                "type": "string"
              },
              "port": {
                "type": "integer",
```

```
            "minimum": 0,
            "maximum": 65535
          }
        },
        "required": [
          "address",
          "port"
        ]
      }
    },
    "http_endpoints": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "address": {
            "type": "string"
          },
          "port": {
            "type": "integer",
            "minimum": 0,
            "maximum": 65535
          }
        },
        "required": [
          "address",
          "port"
        ]
      }
    },
    "archives_max_nested": {
      "type": "integer",
      "minimum": 0,
      "default": 5
    },
    "max_scan_time": {
      "type": "integer",
      "minimum": 0,
      "default": 90
    },
    "keepalive_time": {
      "type": "integer",
      "minimum": 1,
      "default": 600
    },
    "max_conns": {
      "type": "integer",
      "minimum": 1,
      "default": 500
    },
    "reputation": {
      "type": "object",
      "properties": {
        "url_check": {
          "type": "boolean",
          "default": true
        },
        "file_check": {
          "type": "boolean",
          "default": true
        },
        "timeout": {
          "type": "integer",
          "minimum": 0,
          "default": 5
        }
      }
    },
    "trickle": {
      "type": "object",
      "properties": {
        "interval": {
          "type": "integer",
          "default": 30
        },
        "max_bytes": {
          "type": "integer",
          "default": 120
        },
        "start_delay": {
          "type": "integer",
          "default": 0
        }
      }
```

```
      },
      "icap": {
        "type": "object",
        "properties": {
          "scan_archives": {
            "type": "boolean",
            "default": true
          },
          "block_archive_max_nested": {
            "type": "boolean",
            "default": false
          },
          "block_encrypted_archives": {
            "type": "boolean",
            "default": false
          },
          "detect_pua": {
            "type": "boolean",
            "default": false
          },
          "detection_template": {
            "type": "string"
          }
        }
      },
      "logging": {
        "type": "object",
        "properties": {
          "ip_address": {
            "type": "boolean",
            "default": false
          },
          "http": {
            "type": "object",
            "properties": {
              "request_headers": {
                "type": "array",
                "items": {
                  "type": "string"
                }
              }
            }
          },
          "icap": {
            "type": "object",
            "properties": {
              "request_headers": {
                "type": "array",
                "items": {
                  "type": "string"
                }
              },
              "encapsulated_request_headers": {
                "type": "array",
                "items": {
                  "type": "string"
                }
              },
              "encapsulated_response_headers": {
                "type": "array",
                "items": {
                  "type": "string"
                }
              }
            }
          }
        }
      },
      "forbidden_uri_categories": {
        "type": "array",
        "items": {
          "type": "string"
        }
      }
    }
  },
  "authentication": {
    "type": "object",
    "properties": {
      "method": {
        "type": "string",
        "enum": [
          "none",
          "api_key"
```

```
          ]
        },
        "api_key": {
          "type": "string"
        }
      }
    },
    "tls": {
      "type": "object",
      "properties": {
        "certificate": {
          "type": "string"
        },
        "key": {
          "type": "string"
        }
      },
      "required": [
        "certificate",
        "key"
      ]
    }
  }
}
```

Atlant container license settings

These settings control the license that Atlant uses. Either `subscription_key` or `license_file` must be specified.

subscription_key	Atlant subscription key.
license_file	The path to a license file. Relative paths are interpreted as relative to the directory that contains the configuration file. If the license file is in the same directory as the configuration file, only the name of the license file is required.

Atlant container scanning settings

These settings control the behavior of the scanning service.

scanning/icap_endpoints	An array of objects where each object specifies an address and a port that the ICAP scanning service listens. Each item in the array must have a string address property and an integer port property.
scanning/http_endpoints	An array of objects where each object specifies an address and a port that the REST scanning service listens. Each item in the array must have a string address property and an integer port property.
scanning/ archives_max_nested	An integer property that specifies the maximum number of nested archives that Atlant tries to extract if archive scanning is enabled.
scanning/max_scan_time	An integer that specifies the maximum number of seconds that scans can take. **Note:** This time limit does not include the time it takes for the scanning service to receive the content so the total time that it takes to receive and scan the file can be longer than the time limit specified in this setting.
scanning/keepalive_time	
scanning/max_conns	An integer that specifies the maximum number of simultaneous connections that scanning service accepts.

Reputation settings

scanning/reputation/url_check	A boolean that specifies whether the online reputation service is used to scan URLs.
scanning/reputation/file_check	A boolean that specifies whether the online reputation service is used to identify files based on the hash of their content.
scanning/reputation/timeout	An integer that sets the time limit for requests to the online reputation service.

Reputation settings

scanning/forbidden_uri_categories	A list of strings that specifies blocked URL categories.

ICAP trickle settings

scanning/trickle	An object that sets whether the ICAP response trickling is enabled. When enabled, clients receive a part of the content before it is scanned. Since content is sent before the scan has been finished, it is possible that the client may receive bits of harmful content and there is no guarantee that truncating harmful content in this way makes it safe. Consider whether this trade-off is acceptable for your use case before you enable trickling. We recommend that you have additional protection at endpoints when trickling is enabled. If trickling is enabled, Atlant sends ICAP response headers and any possible encapsulated HTTP headers to the client immediately. Then, it starts trickling the encapsulated body data one byte at a time. After the scan is finished, either the remaining unmodified body data is sent, or if there is an infection, the ICAP connection is closed. If the property is omitted, trickling is not used with ICAP.
scanning/trickle/ interval	An integer that specifies the number of seconds that Atlant waits between sending bytes of content to the client when trickling is enabled. Atlant starts sending the content of the file that is being scanned to the client one byte at a time, waiting for the specified number of seconds between each byte that is sent.
scanning/trickle/ max_bytes	An integer that specifies the maximum number of bytes that Atlant sends to the client before scan results are ready.
scanning/trickle/ start_delay	An integer that specifies the number of seconds that Atlant waits before it starts trickling. Setting this to zero (the default value) causes trickling to be enabled for all requests.

ICAP scanning settings

scanning/icap/scan_archives	A boolean that specifies whether archives are extracted when scanned.
scanning/icap/ block_archive_max_nested	A boolean that specifies whether archives containing more levels of nested archives than specified in the ***scanning/ archives_max_nested*** setting are blocked.
scanning/icap/ block_encrypted_archives	A boolean that specifies whether encrypted archives are blocked.
scanning/icap/detect_pua	A boolean that specifies whether potentially unwanted applications are blocked by the ICAP service.
scanning/icap/detection_template	The path to an ICAP detection template that is used when the content is blocked by the service. Relative paths are interpreted as relative to the directory that contains the configuration file.

Scan logging settings

scanning/logging/ip_address	A boolean that specifies whether client's IP address is logged when it makes a scan request.
scanning/logging/http/ request_headers	An array of strings that specifies HTTP request headers to be logged from scan requests made to the REST scanning API.
scanning/logging/icap/ request_headers	An array of strings that specifies ICAP request headers to be logged from scan requests made to the ICAP scanning API.
scanning/logging/icap/ encapsulated_request_headers	An array of strings that specifies the HTTP request headers to be logged from the encapsulated HTTP requests within scan requests made to the ICAP scanning API.

Scan logging settings	
scanning/logging/icap/ encapsulated_response_headers	An array of strings that specifies the HTTP response headers to be logged from the encapsulated HTTP responses within scan requests made to the ICAP scanning API.

Atlant container authentication settings

These settings control whether the REST scanning API requires authentication.

If the authentication object is omitted from the configuration file, the authentication is not required for accessing the scanning API.

authentication/method	A string that specifies the authentication method to be used. Possible values for this are `none` when no authentication is required, and `api_key`, indicating API key-based authentication is required.
authentication/api_key	A string that specifies the API key that clients must include to their requests when accessing the scanning API. This is required when ***authentication/ method*** is set to `api_key`.

Atlant container TLS settings

These settings control whether the REST scanning API is exposed over HTTPS.

If the TLS object is omitted from the configuration file, HTTPS is not used. If the TLS object is present, it must contain the properties certificate and key.

tls/certificate	The path to a PEM formatted certificate for the scanning API. Relative paths are interpreted as relative to the directory that contains the configuration file.
tls/key	The path to a key file associated with the certificate. Relative paths are interpreted as relative to the directory that contains the configuration file.

Atlant container proxy settings

These settings control whether Atlant uses an HTTP proxy when connecting to the Internet. If the ***http_proxy*** object is omitted from the configuration file, a proxy is not used when making connections.

http_proxy/host	A string that specifies the HTTP proxy address.
http_proxy/port	An integer that specifies the HTTP proxy port.
http_proxy/username	An URL-encoded string that specifies the username for the proxy if the proxy requires authentication.
http_proxy/password	An URL-encoded string that specifies the password for the proxy if the proxy requires authentication. If ***http_proxy/username*** and ***http_proxy/password*** are not specified, Atlant does not use authentication when accessing the proxy.

4.2.5 Checking the engine status

The status API enables clients to query engine and server versions.

Resource URI: `/api/status/v1`

Methods: `GET`

This retrieves the status information. The response contains an `application/json` body structured according to the following schema:

```
{
    "type": "object",
    "properties": {
        "engines": {
            "description": "Engines.",
            "type": "array",
```

```
            "items": {
                "type": "object",
                "properties": {
                    "name": {
                        "description": "Engine name.",
                        "type": "string"
                    },
                    "version": {
                        "description": "Engine version.",
                        "type": "string"
                    },
                    "db_version": {
                        "descripton": "Engine database version",
                        "type": "string"
                    },
                    "release_timestamp": {
                        "description": "Engine timestamp",
                        "type": "integer"
                    }
                }
            },
            "required": [
                "name",
                "version",
                "db_version",
                "release_timestamp"
            ]
        },
        "version": {
            "description": "Server version.",
            "type": "string"
        }
    },
    "required": [
        "engines"
    ]
}
```

4.3 Management API

Management API can be used to inspect and change Atlant's configuration.

Note: This API is not available on Atlant container.

4.3.1 Client management

Client management endpoint allows the creation, deletion, and listing of API clients.

Client management is only available when using the internal authorization server.

Resource URI: `/api/atlant/clients/v1`

Methods: `GET POST DELETE`

Data type: `object`

GET

Return a list of clients and their scopes.

Example request:

```
GET /api/atlant/clients/v1/ HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "clients": [
    {
      "client_id": "CLIENT_ID_1",
      "scopes": ["scan"]
```

```
    },
    {
      "client_id": "CLIENT_ID_2",
      "scopes": ["scan", "management"]
    }
  ]
}
```

POST

Create a client. Caller needs to specify a list of scopes that the client is allowed to access. Response will contain a newly-generated client ID and client secret for the client.

Example request:

```
POST /api/atlant/clients/v1/ HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "scopes": ["scan", "management"]
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
  "client_id": "CLIENT_ID",
  "client_secret": "CLIENT_SECRET"
}
```

DELETE

Remove a client. Caller needs to specify the client ID of the client to be removed.

Example request:

```
DELETE /api/atlant/clients/v1/ HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "client_id": "CLIENT_ID"
}
```

Example response:

```
HTTP/1.1 204 No Content
```

4.3.2 Updating the product

REST API requests can be used to download and install new product, malware definition, and scanning engine updates for the product.

Resource URI: `/api/atlant/actions/v1/update`

Methods: `GET` `POST`

Data type: `object`

POST

Install all product, malware definition, and scanning engine updates.

Note:

- Installing updates through a request made through the management server REST API is not supported in isolated installations.
- If a specific product version has been set in the Atlant configuration, the product will not be updated beyond that version, only malware definition and scanning engine updates (if any) will be installed.

The request resource URI may include the `wait` query parameter to choose whether the client expects a response from the management server only after the installation of updates has finished (synchronous mode), or whether the client intends to poll for the completion of the installation through the REST API later (asynchronous mode). By default, the management server responds to update requests in asynchronous mode.

Example request (asynchronous mode):

```
POST /api/atlant/actions/v1/update HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{}
```

Example request (synchronous mode):

```
POST /api/atlant/actions/v1/update?wait=true HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{}
```

Example response that indicates the successful completion of the update:

```
HTTP/1.1 200 OK
Content-Type: application/json
Location: URL

{
    "status": "success",
    "result": {
        "updates_available": true
    }
}
```

The value of the `updates_available` field indicates whether the request resulted in installing one or more updates.

Example response (asynchronous mode only):

```
HTTP/1.1 201 Created
Location: URL
Retry-After: 3
```

The client should use the URL given in the `Location` header for polling for the completion of the update later. The `Retry-After` header gives a suggestion on how many seconds the client should wait before polling.

GET

Get the result of a previously started update operation.

Example request:

```
GET URL HTTP/1.1
Authorization: Bearer TOKEN
```

URL should match the URL included in the `Location` header of a previous POST request.

Example response (asynchronous mode only):

```
HTTP/1.1 202 Accepted
Retry-After: 5
```

A 202 response means that the request is still being processed on the server, and the client should continue polling.

Example response (update was completed):

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "status": "success",
    "result": {
        "updates_available": false
    }
}
```

Error handling

A request to update the product may result in the restarting of the management server or other parts of Atlant as part of the update. For this reason, the client should be prepared to handle temporary communication failures with the management server, and explicit HTTP responses with status code `410 Gone` or `503 Service Temporarily Unavailable`.

In case the management server closes the connection without sending a response, or sends a response with one of the above status codes, the client should keep restarting the update operation by sending new POST requests until it receives a `200 OK` response to indicate that the operation was completed. A response with `success` status means that all components of the product are then up to date.

4.3.3 Product license

Resource URI: `/api/atlant/config/v1/license`

Methods: GET PUT PATCH

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/license HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "content": "XXXX-XXXX-XXXX-XXXX-XXXX",
  "remote_license_key": "XXXX-XXXX-XXXX-XXXX-XXXX",
  "type": "key"
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/license HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "content": "XXXX-XXXX-XXXX-XXXX-XXXX",
  "remote_license_key": "XXXX-XXXX-XXXX-XXXX-XXXX",
  "type": "key"
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "content": "XXXX-XXXX-XXXX-XXXX-XXXX",
  "remote_license_key": "XXXX-XXXX-XXXX-XXXX-XXXX",
  "type": "key"
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/license HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "content": "XXXX-XXXX-XXXX-XXXX-XXXX",
  "remote_license_key": "XXXX-XXXX-XXXX-XXXX-XXXX",
  "type": "key"
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "content": "XXXX-XXXX-XXXX-XXXX-XXXX",
  "remote_license_key": "XXXX-XXXX-XXXX-XXXX-XXXX",
  "type": "key"
}
```

4.3.4 Authorization server settings

Resource URI: `/api/atlant/config/v1/authorization`

Methods: GET PUT PATCH DELETE

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/authorization HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "external_domains": [
    {
      "audience": "example-audience",
      "issuer": "example-issuer",
      "signing_method": "HS256",
      "verification_key": "cGFzc3dvcmQK"
    }
  ],
  "https_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 8080
    }
  ]
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/authorization HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "external_domains": [
    {
      "audience": "example-audience",
      "issuer": "example-issuer",
      "signing_method": "HS256",
      "verification_key": "cGFzc3dvcmQK"
    }
  ],
  "https_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 8080
    }
  ]
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "external_domains": [
    {
      "audience": "example-audience",
      "issuer": "example-issuer",
      "signing_method": "HS256",
      "verification_key": "cGFzc3dvcmQK"
    }
  ],
  "https_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 8080
    }
  ]
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/authorization HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "external_domains": [
    {
      "audience": "example-audience",
      "issuer": "example-issuer",
      "signing_method": "HS256",
      "verification_key": "cGFzc3dvcmQK"
    }
  ],
  "https_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 8080
    }
  ]
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "external_domains": [
```

```
    {
      "audience": "example-audience",
      "issuer": "example-issuer",
      "signing_method": "HS256",
      "verification_key": "cGFzc3dvcmQK"
    }
  ],
  "https_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 8080
    }
  ]
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/authorization HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

List of external authorization domains

Resource URI: `/api/atlant/config/v1/authorization/external_domains`

Methods: GET PUT PATCH POST DELETE

Data type: `array`

GET

Example request:

```
GET /api/atlant/config/v1/authorization/external_domains HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "audience": "example-audience",
    "issuer": "example-issuer",
    "signing_method": "HS256",
    "verification_key": "cGFzc3dvcmQK"
  }
]
```

PUT

Example request:

```
PUT /api/atlant/config/v1/authorization/external_domains HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  {
    "audience": "example-audience",
    "issuer": "example-issuer",
    "signing_method": "HS256",
    "verification_key": "cGFzc3dvcmQK"
  }
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "audience": "example-audience",
    "issuer": "example-issuer",
    "signing_method": "HS256",
    "verification_key": "cGFzc3dvcmQK"
  }
]
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/authorization/external_domains HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  {
    "audience": "example-audience",
    "issuer": "example-issuer",
    "signing_method": "HS256",
    "verification_key": "cGFzc3dvcmQK"
  }
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "audience": "example-audience",
    "issuer": "example-issuer",
    "signing_method": "HS256",
    "verification_key": "cGFzc3dvcmQK"
  }
]
```

POST

Example request:

```
POST /api/atlant/config/v1/authorization/external_domains HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "audience": "example-audience",
  "issuer": "example-issuer",
  "signing_method": "HS256",
  "verification_key": "cGFzc3dvcmQK"
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: URL

{
  "audience": "example-audience",
  "issuer": "example-issuer",
  "signing_method": "HS256",
  "verification_key": "cGFzc3dvcmQK"
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/authorization/external_domains HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

External authorization domain

Keys:

- DOMAIN

  Example: `"string"`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/authorization/external_domains/DOMAIN`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/authorization/external_domains/DOMAIN HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "audience": "example-audience",
  "issuer": "example-issuer",
  "signing_method": "HS256",
  "verification_key": "cGFzc3dvcmQK"
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/authorization/external_domains/DOMAIN HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "audience": "example-audience",
  "issuer": "example-issuer",
  "signing_method": "HS256",
  "verification_key": "cGFzc3dvcmQK"
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "audience": "example-audience",
  "issuer": "example-issuer",
  "signing_method": "HS256",
  "verification_key": "cGFzc3dvcmQK"
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/authorization/external_domains/DOMAIN HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "audience": "example-audience",
  "issuer": "example-issuer",
  "signing_method": "HS256",
  "verification_key": "cGFzc3dvcmQK"
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "audience": "example-audience",
  "issuer": "example-issuer",
  "signing_method": "HS256",
  "verification_key": "cGFzc3dvcmQK"
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/authorization/external_domains/DOMAIN HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Domain audience

Keys:

- DOMAIN

  Example: `"string"`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/authorization/external_domains/DOMAIN/audience`

Methods: `GET` `PUT` `PATCH`

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/authorization/external_domains/DOMAIN/audience HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"example-audience"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/authorization/external_domains/DOMAIN/audience HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"example-audience"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"example-audience"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/authorization/external_domains/DOMAIN/audience HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"example-audience"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"example-audience"
```

Domain issuer

Keys:

- DOMAIN

  Example: `"string"`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/authorization/external_domains/DOMAIN/issuer`

Methods: GET PUT PATCH

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/authorization/external_domains/DOMAIN/issuer HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"example-issuer"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/authorization/external_domains/DOMAIN/issuer HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN
```

```
"example-issuer"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"example-issuer"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/authorization/external_domains/DOMAIN/issuer HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"example-issuer"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"example-issuer"
```

Domain signing method

Value must be one of `"HS256"`, `"RS256"`

Keys:

- DOMAIN

  Example: `"string"`

Key should be URL-encoded

Resource URI:
`/api/atlant/config/v1/authorization/external_domains/DOMAIN/signing_method`

Methods: `GET` `PUT` `PATCH`

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/authorization/external_domains/DOMAIN/signing_method HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"HS256"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/authorization/external_domains/DOMAIN/signing_method HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"HS256"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"HS256"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/authorization/external_domains/DOMAIN/signing_method HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"HS256"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"HS256"
```

Domain verification key

Keys:

- DOMAIN

  Example: `"string"`

Key should be URL-encoded

Resource URI:
`/api/atlant/config/v1/authorization/external_domains/DOMAIN/verification_key`

Methods: GET PUT PATCH

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/authorization/external_domains/DOMAIN/verification_key HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"cGFzc3dvcmQK"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/authorization/external_domains/DOMAIN/verification_key HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"cGFzc3dvcmQK"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"cGFzc3dvcmQK"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/authorization/external_domains/DOMAIN/verification_key HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"cGFzc3dvcmQK"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"cGFzc3dvcmQK"
```

List of authorization service HTTP endpoints

Resource URI: `/api/atlant/config/v1/authorization/https_endpoints`

Methods: `GET` `PUT` `PATCH` `POST` `DELETE`

Data type: `array`

GET

Example request:

```
GET /api/atlant/config/v1/authorization/https_endpoints HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "address": "127.0.0.1",
    "port": 8080
  }
]
```

PUT

Example request:

```
PUT /api/atlant/config/v1/authorization/https_endpoints HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  {
    "address": "127.0.0.1",
    "port": 8080
  }
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "address": "127.0.0.1",
    "port": 8080
  }
]
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/authorization/https_endpoints HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  {
    "address": "127.0.0.1",
    "port": 8080
  }
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "address": "127.0.0.1",
    "port": 8080
  }
]
```

POST

Example request:

```
POST /api/atlant/config/v1/authorization/https_endpoints HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "address": "127.0.0.1",
  "port": 8080
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: URL

{
  "address": "127.0.0.1",
  "port": 8080
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/authorization/https_endpoints HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Authorization service HTTP endpoint

Keys:

- ENDPOINT

  Example: `{"address":"string","port":0}`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/authorization/https_endpoints/ENDPOINT`

Methods: GET PUT PATCH DELETE

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/authorization/https_endpoints/ENDPOINT HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "address": "127.0.0.1",
  "port": 8080
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/authorization/https_endpoints/ENDPOINT HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "address": "127.0.0.1",
  "port": 8080
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "address": "127.0.0.1",
  "port": 8080
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/authorization/https_endpoints/ENDPOINT HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "address": "127.0.0.1",
  "port": 8080
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "address": "127.0.0.1",
  "port": 8080
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/authorization/https_endpoints/ENDPOINT HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Authorization service HTTP endpoint address

Keys:

- ENDPOINT

  Example: `{"address":"string","port":0}`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/authorization/https_endpoints/ENDPOINT/address`

Methods: GET PUT PATCH

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/authorization/https_endpoints/ENDPOINT/address HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"127.0.0.1"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/authorization/https_endpoints/ENDPOINT/address HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"127.0.0.1"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"127.0.0.1"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/authorization/https_endpoints/ENDPOINT/address HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"127.0.0.1"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"127.0.0.1"
```

Authorization service HTTP endpoint port

Keys:

- ENDPOINT

  Example: `{"address":"string","port":0}`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/authorization/https_endpoints/ENDPOINT/port`

Methods: `GET` `PUT` `PATCH`

Data type: `number`

GET

Example request:

```
GET /api/atlant/config/v1/authorization/https_endpoints/ENDPOINT/port HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

8080
```

PUT

Example request:

```
PUT /api/atlant/config/v1/authorization/https_endpoints/ENDPOINT/port HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

8080
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

8080
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/authorization/https_endpoints/ENDPOINT/port HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

8080
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

8080
```

4.3.5 TLS data

Resource URI: `/api/atlant/config/v1/tls`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/tls HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "certificate": "a2V5Cg==",
  "key": "Y2VydGlmaWNhdGU="
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/tls HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "certificate": "a2V5Cg==",
  "key": "Y2VydGlmaWNhdGU="
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "certificate": "a2V5Cg==",
  "key": "Y2VydGlmaWNhdGU="
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/tls HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "certificate": "a2V5Cg==",
  "key": "Y2VydGlmaWNhdGU="
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "certificate": "a2V5Cg==",
  "key": "Y2VydGlmaWNhdGU="
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/tls HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

TLS certificate

Resource URI: `/api/atlant/config/v1/tls/certificate`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/tls/certificate HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"a2V5Cg=="
```

PUT

Example request:

```
PUT /api/atlant/config/v1/tls/certificate HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"a2V5Cg=="
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"a2V5Cg=="
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/tls/certificate HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"a2V5Cg=="
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"a2V5Cg=="
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/tls/certificate HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

TLS key

Resource URI: `/api/atlant/config/v1/tls/key`

Methods: GET PUT PATCH DELETE

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/tls/key HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"Y2VydGlmaWNhdGU="
```

PUT

Example request:

```
PUT /api/atlant/config/v1/tls/key HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"Y2VydGlmaWNhdGU="
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"Y2VydGlmaWNhdGU="
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/tls/key HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"Y2VydGlmaWNhdGU="
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"Y2VydGlmaWNhdGU="
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/tls/key HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

4.3.6 Management server settings

Resource URI: `/api/atlant/config/v1/management`

Methods: `GET PUT PATCH DELETE`

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/management HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "https_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 8081
    }
  ]
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/management HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "https_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 8081
    }
  ]
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "https_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 8081
    }
  ]
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/management HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
```

```
  "https_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 8081
    }
  ]
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "https_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 8081
    }
  ]
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/management HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

List of management server endpoints

Resource URI: `/api/atlant/config/v1/management/https_endpoints`

Methods: GET PUT PATCH POST DELETE

Data type: `array`

GET

Example request:

```
GET /api/atlant/config/v1/management/https_endpoints HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "address": "127.0.0.1",
    "port": 8081
  }
]
```

PUT

Example request:

```
PUT /api/atlant/config/v1/management/https_endpoints HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  {
    "address": "127.0.0.1",
    "port": 8081
```

```
  }
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "address": "127.0.0.1",
    "port": 8081
  }
]
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/management/https_endpoints HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  {
    "address": "127.0.0.1",
    "port": 8081
  }
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "address": "127.0.0.1",
    "port": 8081
  }
]
```

POST

Example request:

```
POST /api/atlant/config/v1/management/https_endpoints HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "address": "127.0.0.1",
  "port": 8081
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: URL

{
  "address": "127.0.0.1",
  "port": 8081
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/management/https_endpoints HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Management server endpoint

Keys:

- ENDPOINT

  Example: `{"address":"string","port":0}`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/management/https_endpoints/ENDPOINT`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/management/https_endpoints/ENDPOINT HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "address": "127.0.0.1",
  "port": 8081
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/management/https_endpoints/ENDPOINT HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "address": "127.0.0.1",
  "port": 8081
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "address": "127.0.0.1",
  "port": 8081
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/management/https_endpoints/ENDPOINT HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "address": "127.0.0.1",
  "port": 8081
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "address": "127.0.0.1",
  "port": 8081
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/management/https_endpoints/ENDPOINT HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Management server endpoint address

Keys:

- ENDPOINT

  Example: `{"address":"string","port":0}`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/management/https_endpoints/ENDPOINT/address`

Methods: GET PUT PATCH

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/management/https_endpoints/ENDPOINT/address HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"127.0.0.1"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/management/https_endpoints/ENDPOINT/address HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"127.0.0.1"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"127.0.0.1"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/management/https_endpoints/ENDPOINT/address HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"127.0.0.1"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"127.0.0.1"
```

Management server endpoint port

Keys:

- ENDPOINT

  Example: `{"address":"string","port":0}`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/management/https_endpoints/ENDPOINT/port`

Methods: `GET` `PUT` `PATCH`

Data type: `number`

GET

Example request:

```
GET /api/atlant/config/v1/management/https_endpoints/ENDPOINT/port HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

8081
```

PUT

Example request:

```
PUT /api/atlant/config/v1/management/https_endpoints/ENDPOINT/port HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

8081
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

8081
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/management/https_endpoints/ENDPOINT/port HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN
```

```
8081
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

8081
```

4.3.7 Settings for controlling the use of online services

Resource URI: `/api/atlant/config/v1/online_services`

Methods: GET PUT PATCH DELETE

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/online_services HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": true
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/online_services HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "enabled": true
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": true
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/online_services HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "enabled": true
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "enabled": true
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/online_services HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Enable the use of online services

Resource URI: `/api/atlant/config/v1/online_services/enabled`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/online_services/enabled HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/online_services/enabled HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/online_services/enabled HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/online_services/enabled HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

4.3.8 HTTP proxy settings

Resource URI: `/api/atlant/config/v1/http_proxy`

Methods: GET PUT PATCH DELETE

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/http_proxy HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": true,
  "host": "example.com",
  "password": "example-password",
  "port": 3128,
  "username": "example-user"
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/http_proxy HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "enabled": true,
  "host": "example.com",
  "password": "example-password",
  "port": 3128,
  "username": "example-user"
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": true,
  "host": "example.com",
  "password": "example-password",
  "port": 3128,
  "username": "example-user"
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/http_proxy HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "enabled": true,
  "host": "example.com",
  "password": "example-password",
  "port": 3128,
  "username": "example-user"
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": true,
  "host": "example.com",
  "password": "example-password",
  "port": 3128,
  "username": "example-user"
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/http_proxy HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

HTTP proxy enabled flag

Resource URI: `/api/atlant/config/v1/http_proxy/enabled`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/http_proxy/enabled HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/http_proxy/enabled HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/http_proxy/enabled HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/http_proxy/enabled HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

HTTP proxy host

Resource URI: `/api/atlant/config/v1/http_proxy/host`

Methods: GET PUT PATCH DELETE

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/http_proxy/host HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"example.com"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/http_proxy/host HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"example.com"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"example.com"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/http_proxy/host HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"example.com"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"example.com"
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/http_proxy/host HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

HTTP proxy authentication password

Resource URI: `/api/atlant/config/v1/http_proxy/password`

Methods: GET PUT PATCH DELETE

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/http_proxy/password HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"example-password"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/http_proxy/password HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"example-password"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"example-password"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/http_proxy/password HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"example-password"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"example-password"
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/http_proxy/password HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

HTTP proxy port

Resource URI: `/api/atlant/config/v1/http_proxy/port`

Methods: GET PUT PATCH DELETE

Data type: `number`

GET

Example request:

```
GET /api/atlant/config/v1/http_proxy/port HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

3128
```

PUT

Example request:

```
PUT /api/atlant/config/v1/http_proxy/port HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

3128
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

3128
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/http_proxy/port HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

3128
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

3128
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/http_proxy/port HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

HTTP proxy authentication user name. If empty, authentication is disabled.

Resource URI: `/api/atlant/config/v1/http_proxy/username`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/http_proxy/username HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"example-user"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/http_proxy/username HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"example-user"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"example-user"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/http_proxy/username HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"example-user"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"example-user"
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/http_proxy/username HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

4.3.9 Scanning server settings

Resource URI: `/api/atlant/config/v1/scanning`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/scanning HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "archive_max_nested": 5,
  "forbidden_uri_categories": {
    "abortion": true,
    "adserving": true,
    "adult": true,
    "alcohol_and_tobacco": true,
    "anonymizers": true,
    "auctions": true,
    "banking": true,
    "blogs": true,
    "chat": true,
    "dating": true,
    "disturbing": true,
    "drugs": true,
    "entertainment": true,
    "file_sharing": true,
```

```
    "forum": true,
    "gambling": true,
    "games": true,
    "hate": true,
    "illegal": true,
    "job_search": true,
    "paymentservice": true,
    "search_engines": true,
    "shopping": true,
    "social_networking": true,
    "software_download": true,
    "spam": true,
    "streaming_media": true,
    "tracking_cookie": true,
    "tracking_domain": true,
    "tracking_object": true,
    "tracking_script": true,
    "travel": true,
    "violence": true,
    "warez": true,
    "weapons": true,
    "webmail": true
  },
  "https_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 8082
    }
  ],
  "icap": {
    "block_archive_max_nested": true,
    "block_encrypted_archives": true,
    "detect_pua": true,
    "scan_archives": true,
    "trickle": {
      "enabled": true,
      "interval": 30,
      "max_bytes": 120,
      "start_delay": 0
    }
  },
  "icap_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 1344
    }
  ],
  "icaps_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 11344
    }
  ],
  "keepalive_time": 600,
  "logging": {
    "https": {
      "request_headers": [
        "User-Agent"
      ]
    },
    "icap": {
      "encapsulated_request_headers": [
        "User-Agent"
      ],
      "encapsulated_response_headers": [
        "ETag"
      ],
      "request_headers": [
        "X-Client-Username"
      ]
    },
    "ip_address": true,
    "rotation": {
      "file_size": {
        "enabled": true,
        "limit": 10000000
      },
      "max_bytes": {
        "enabled": false,
        "limit": 100000000
      },
      "max_days": {
        "enabled": false,
        "limit": 30
```

```
      },
      "max_files": {
        "enabled": true,
        "limit": 10
      }
    }
  },
  "max_scan_size": {
    "enabled": true,
    "limit": 2147483648
  },
  "max_scan_time": 90,
  "reputation": {
    "file_check": true,
    "timeout": 5,
    "url_check": true
  }
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "archive_max_nested": 5,
  "forbidden_uri_categories": {
    "abortion": true,
    "adserving": true,
    "adult": true,
    "alcohol_and_tobacco": true,
    "anonymizers": true,
    "auctions": true,
    "banking": true,
    "blogs": true,
    "chat": true,
    "dating": true,
    "disturbing": true,
    "drugs": true,
    "entertainment": true,
    "file_sharing": true,
    "forum": true,
    "gambling": true,
    "games": true,
    "hate": true,
    "illegal": true,
    "job_search": true,
    "paymentservice": true,
    "search_engines": true,
    "shopping": true,
    "social_networking": true,
    "software_download": true,
    "spam": true,
    "streaming_media": true,
    "tracking_cookie": true,
    "tracking_domain": true,
    "tracking_object": true,
    "tracking_script": true,
    "travel": true,
    "violence": true,
    "warez": true,
    "weapons": true,
    "webmail": true
  },
  "https_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 8082
    }
  ],
  "icap": {
    "block_archive_max_nested": true,
    "block_encrypted_archives": true,
    "detect_pua": true,
    "scan_archives": true,
    "trickle": {
      "enabled": true,
      "interval": 30,
      "max_bytes": 120,
```

```
      "start_delay": 0
    }
  },
  "icap_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 1344
    }
  ],
  "icaps_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 11344
    }
  ],
  "keepalive_time": 600,
  "logging": {
    "https": {
      "request_headers": [
        "User-Agent"
      ]
    },
    "icap": {
      "encapsulated_request_headers": [
        "User-Agent"
      ],
      "encapsulated_response_headers": [
        "ETag"
      ],
      "request_headers": [
        "X-Client-Username"
      ]
    },
    "ip_address": true,
    "rotation": {
      "file_size": {
        "enabled": true,
        "limit": 10000000
      },
      "max_bytes": {
        "enabled": false,
        "limit": 100000000
      },
      "max_days": {
        "enabled": false,
        "limit": 30
      },
      "max_files": {
        "enabled": true,
        "limit": 10
      }
    }
  },
  "max_scan_size": {
    "enabled": true,
    "limit": 2147483648
  },
  "max_scan_time": 90,
  "reputation": {
    "file_check": true,
    "timeout": 5,
    "url_check": true
  }
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "archive_max_nested": 5,
  "forbidden_uri_categories": {
    "abortion": true,
    "adserving": true,
    "adult": true,
    "alcohol_and_tobacco": true,
    "anonymizers": true,
    "auctions": true,
    "banking": true,
    "blogs": true,
    "chat": true,
    "dating": true,
```

```
    "disturbing": true,
    "drugs": true,
    "entertainment": true,
    "file_sharing": true,
    "forum": true,
    "gambling": true,
    "games": true,
    "hate": true,
    "illegal": true,
    "job_search": true,
    "paymentservice": true,
    "search_engines": true,
    "shopping": true,
    "social_networking": true,
    "software_download": true,
    "spam": true,
    "streaming_media": true,
    "tracking_cookie": true,
    "tracking_domain": true,
    "tracking_object": true,
    "tracking_script": true,
    "travel": true,
    "violence": true,
    "warez": true,
    "weapons": true,
    "webmail": true
  },
  "https_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 8082
    }
  ],
  "icap": {
    "block_archive_max_nested": true,
    "block_encrypted_archives": true,
    "detect_pua": true,
    "scan_archives": true,
    "trickle": {
      "enabled": true,
      "interval": 30,
      "max_bytes": 120,
      "start_delay": 0
    }
  },
  "icap_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 1344
    }
  ],
  "icaps_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 11344
    }
  ],
  "keepalive_time": 600,
  "logging": {
    "https": {
      "request_headers": [
        "User-Agent"
      ]
    },
    "icap": {
      "encapsulated_request_headers": [
        "User-Agent"
      ],
      "encapsulated_response_headers": [
        "ETag"
      ],
      "request_headers": [
        "X-Client-Username"
      ]
    },
    "ip_address": true,
    "rotation": {
      "file_size": {
        "enabled": true,
        "limit": 10000000
      },
      "max_bytes": {
        "enabled": false,
        "limit": 100000000
```

```
      },
      "max_days": {
        "enabled": false,
        "limit": 30
      },
      "max_files": {
        "enabled": true,
        "limit": 10
      }
    }
  },
  "max_scan_size": {
    "enabled": true,
    "limit": 2147483648
  },
  "max_scan_time": 90,
  "reputation": {
    "file_check": true,
    "timeout": 5,
    "url_check": true
  }
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "archive_max_nested": 5,
  "forbidden_uri_categories": {
    "abortion": true,
    "adserving": true,
    "adult": true,
    "alcohol_and_tobacco": true,
    "anonymizers": true,
    "auctions": true,
    "banking": true,
    "blogs": true,
    "chat": true,
    "dating": true,
    "disturbing": true,
    "drugs": true,
    "entertainment": true,
    "file_sharing": true,
    "forum": true,
    "gambling": true,
    "games": true,
    "hate": true,
    "illegal": true,
    "job_search": true,
    "paymentservice": true,
    "search_engines": true,
    "shopping": true,
    "social_networking": true,
    "software_download": true,
    "spam": true,
    "streaming_media": true,
    "tracking_cookie": true,
    "tracking_domain": true,
    "tracking_object": true,
    "tracking_script": true,
    "travel": true,
    "violence": true,
    "warez": true,
    "weapons": true,
    "webmail": true
  },
  "https_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 8082
    }
  ],
  "icap": {
    "block_archive_max_nested": true,
    "block_encrypted_archives": true,
    "detect_pua": true,
    "scan_archives": true,
```

```
    "trickle": {
      "enabled": true,
      "interval": 30,
      "max_bytes": 120,
      "start_delay": 0
    }
  },
  "icap_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 1344
    }
  ],
  "icaps_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 11344
    }
  ],
  "keepalive_time": 600,
  "logging": {
    "https": {
      "request_headers": [
        "User-Agent"
      ]
    },
    "icap": {
      "encapsulated_request_headers": [
        "User-Agent"
      ],
      "encapsulated_response_headers": [
        "ETag"
      ],
      "request_headers": [
        "X-Client-Username"
      ]
    },
    "ip_address": true,
    "rotation": {
      "file_size": {
        "enabled": true,
        "limit": 10000000
      },
      "max_bytes": {
        "enabled": false,
        "limit": 100000000
      },
      "max_days": {
        "enabled": false,
        "limit": 30
      },
      "max_files": {
        "enabled": true,
        "limit": 10
      }
    }
  },
  "max_scan_size": {
    "enabled": true,
    "limit": 2147483648
  },
  "max_scan_time": 90,
  "reputation": {
    "file_check": true,
    "timeout": 5,
    "url_check": true
  }
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "archive_max_nested": 5,
  "forbidden_uri_categories": {
    "abortion": true,
    "adserving": true,
    "adult": true,
    "alcohol_and_tobacco": true,
    "anonymizers": true,
    "auctions": true,
```

```
    "banking": true,
    "blogs": true,
    "chat": true,
    "dating": true,
    "disturbing": true,
    "drugs": true,
    "entertainment": true,
    "file_sharing": true,
    "forum": true,
    "gambling": true,
    "games": true,
    "hate": true,
    "illegal": true,
    "job_search": true,
    "paymentservice": true,
    "search_engines": true,
    "shopping": true,
    "social_networking": true,
    "software_download": true,
    "spam": true,
    "streaming_media": true,
    "tracking_cookie": true,
    "tracking_domain": true,
    "tracking_object": true,
    "tracking_script": true,
    "travel": true,
    "violence": true,
    "warez": true,
    "weapons": true,
    "webmail": true
  },
  "https_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 8082
    }
  ],
  "icap": {
    "block_archive_max_nested": true,
    "block_encrypted_archives": true,
    "detect_pua": true,
    "scan_archives": true,
    "trickle": {
      "enabled": true,
      "interval": 30,
      "max_bytes": 120,
      "start_delay": 0
    }
  },
  "icap_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 1344
    }
  ],
  "icaps_endpoints": [
    {
      "address": "127.0.0.1",
      "port": 11344
    }
  ],
  "keepalive_time": 600,
  "logging": {
    "https": {
      "request_headers": [
        "User-Agent"
      ]
    },
    "icap": {
      "encapsulated_request_headers": [
        "User-Agent"
      ],
      "encapsulated_response_headers": [
        "ETag"
      ],
      "request_headers": [
        "X-Client-Username"
      ]
    },
    "ip_address": true,
    "rotation": {
      "file_size": {
        "enabled": true,
        "limit": 10000000
```

```
      },
      "max_bytes": {
        "enabled": false,
        "limit": 100000000
      },
      "max_days": {
        "enabled": false,
        "limit": 30
      },
      "max_files": {
        "enabled": true,
        "limit": 10
      }
    }
  },
  "max_scan_size": {
    "enabled": true,
    "limit": 2147483648
  },
  "max_scan_time": 90,
  "reputation": {
    "file_check": true,
    "timeout": 5,
    "url_check": true
  }
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect if scanned urls match any of these categories

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories`

Methods: GET PUT PATCH DELETE

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "abortion": true,
  "adserving": true,
  "adult": true,
  "alcohol_and_tobacco": true,
  "anonymizers": true,
  "auctions": true,
  "banking": true,
  "blogs": true,
  "chat": true,
  "dating": true,
  "disturbing": true,
  "drugs": true,
  "entertainment": true,
  "file_sharing": true,
  "forum": true,
  "gambling": true,
  "games": true,
```

```
  "hate": true,
  "illegal": true,
  "job_search": true,
  "paymentservice": true,
  "search_engines": true,
  "shopping": true,
  "social_networking": true,
  "software_download": true,
  "spam": true,
  "streaming_media": true,
  "tracking_cookie": true,
  "tracking_domain": true,
  "tracking_object": true,
  "tracking_script": true,
  "travel": true,
  "violence": true,
  "warez": true,
  "weapons": true,
  "webmail": true
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "abortion": true,
  "adserving": true,
  "adult": true,
  "alcohol_and_tobacco": true,
  "anonymizers": true,
  "auctions": true,
  "banking": true,
  "blogs": true,
  "chat": true,
  "dating": true,
  "disturbing": true,
  "drugs": true,
  "entertainment": true,
  "file_sharing": true,
  "forum": true,
  "gambling": true,
  "games": true,
  "hate": true,
  "illegal": true,
  "job_search": true,
  "paymentservice": true,
  "search_engines": true,
  "shopping": true,
  "social_networking": true,
  "software_download": true,
  "spam": true,
  "streaming_media": true,
  "tracking_cookie": true,
  "tracking_domain": true,
  "tracking_object": true,
  "tracking_script": true,
  "travel": true,
  "violence": true,
  "warez": true,
  "weapons": true,
  "webmail": true
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "abortion": true,
  "adserving": true,
  "adult": true,
  "alcohol_and_tobacco": true,
  "anonymizers": true,
  "auctions": true,
  "banking": true,
```

```
  "blogs": true,
  "chat": true,
  "dating": true,
  "disturbing": true,
  "drugs": true,
  "entertainment": true,
  "file_sharing": true,
  "forum": true,
  "gambling": true,
  "games": true,
  "hate": true,
  "illegal": true,
  "job_search": true,
  "paymentservice": true,
  "search_engines": true,
  "shopping": true,
  "social_networking": true,
  "software_download": true,
  "spam": true,
  "streaming_media": true,
  "tracking_cookie": true,
  "tracking_domain": true,
  "tracking_object": true,
  "tracking_script": true,
  "travel": true,
  "violence": true,
  "warez": true,
  "weapons": true,
  "webmail": true
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "abortion": true,
  "adserving": true,
  "adult": true,
  "alcohol_and_tobacco": true,
  "anonymizers": true,
  "auctions": true,
  "banking": true,
  "blogs": true,
  "chat": true,
  "dating": true,
  "disturbing": true,
  "drugs": true,
  "entertainment": true,
  "file_sharing": true,
  "forum": true,
  "gambling": true,
  "games": true,
  "hate": true,
  "illegal": true,
  "job_search": true,
  "paymentservice": true,
  "search_engines": true,
  "shopping": true,
  "social_networking": true,
  "software_download": true,
  "spam": true,
  "streaming_media": true,
  "tracking_cookie": true,
  "tracking_domain": true,
  "tracking_object": true,
  "tracking_script": true,
  "travel": true,
  "violence": true,
  "warez": true,
  "weapons": true,
  "webmail": true
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "abortion": true,
  "adserving": true,
  "adult": true,
  "alcohol_and_tobacco": true,
  "anonymizers": true,
  "auctions": true,
  "banking": true,
  "blogs": true,
  "chat": true,
  "dating": true,
  "disturbing": true,
  "drugs": true,
  "entertainment": true,
  "file_sharing": true,
  "forum": true,
  "gambling": true,
  "games": true,
  "hate": true,
  "illegal": true,
  "job_search": true,
  "paymentservice": true,
  "search_engines": true,
  "shopping": true,
  "social_networking": true,
  "software_download": true,
  "spam": true,
  "streaming_media": true,
  "tracking_cookie": true,
  "tracking_domain": true,
  "tracking_object": true,
  "tracking_script": true,
  "travel": true,
  "violence": true,
  "warez": true,
  "weapons": true,
  "webmail": true
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "abortion" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/abortion`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/abortion HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/abortion HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/abortion HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/abortion HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "abortion" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/abortion`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/abortion HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/abortion HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/abortion HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/abortion HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "adserving" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/adserving`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/adserving HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/adserving HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/adserving HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/adserving HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "adult" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/adult`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/adult HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/adult HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/adult HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/adult HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "alcohol_and_tobacco" category

Resource URI:
`/api/atlant/config/v1/scanning/forbidden_uri_categories/alcohol_and_tobacco`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/alcohol_and_tobacco HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/alcohol_and_tobacco HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/alcohol_and_tobacco HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/alcohol_and_tobacco HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "anonymizers" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/anonymizers`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/anonymizers HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/anonymizers HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/anonymizers HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/anonymizers HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "auctions" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/auctions`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/auctions HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/auctions HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/auctions HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/auctions HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "banking" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/banking`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/banking HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/banking HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/banking HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/banking HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "blogs" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/blogs`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/blogs HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/blogs HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/blogs HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/blogs HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "chat" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/chat`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/chat HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/chat HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/chat HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/chat HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "dating" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/dating`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/dating HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/dating HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/dating HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/dating HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "disturbing" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/disturbing`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/disturbing HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/disturbing HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/disturbing HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/disturbing HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "drugs" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/drugs`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/drugs HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/drugs HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/drugs HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/drugs HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "entertainment" category

Resource URI:
`/api/atlant/config/v1/scanning/forbidden_uri_categories/entertainment`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/entertainment HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/entertainment HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/entertainment HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/entertainment HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "file_sharing" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/file_sharing`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/file_sharing HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/file_sharing HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/file_sharing HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/file_sharing HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "forum" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/forum`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/forum HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/forum HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/forum HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/forum HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "gambling" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/gambling`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/gambling HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/gambling HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/gambling HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/gambling HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "games" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/games`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/games HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/games HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/games HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/games HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "hate" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/hate`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/hate HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/hate HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/hate HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/hate HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "illegal" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/illegal`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/illegal HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/illegal HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/illegal HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/illegal HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "job_search" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/job_search`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/job_search HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/job_search HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/job_search HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/job_search HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "paymentservice" category

Resource URI:
`/api/atlant/config/v1/scanning/forbidden_uri_categories/paymentservice`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/paymentservice HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/paymentservice HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/paymentservice HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/paymentservice HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "search_engines" category

Resource URI:
`/api/atlant/config/v1/scanning/forbidden_uri_categories/search_engines`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/search_engines HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/search_engines HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/search_engines HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/search_engines HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "shopping" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/shopping`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/shopping HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/shopping HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/shopping HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/shopping HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "social_networking" category

Resource URI:
`/api/atlant/config/v1/scanning/forbidden_uri_categories/social_networking`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/social_networking HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/social_networking HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/social_networking HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/social_networking HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "software_download" category

Resource URI:
`/api/atlant/config/v1/scanning/forbidden_uri_categories/software_download`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/software_download HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/software_download HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/software_download HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/software_download HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "spam" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/spam`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/spam HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/spam HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/spam HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/spam HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "streaming_media" category

Resource URI:
`/api/atlant/config/v1/scanning/forbidden_uri_categories/streaming_media`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/streaming_media HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/streaming_media HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/streaming_media HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/streaming_media HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "tracking_cookie" category

Resource URI:
`/api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_cookie`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_cookie HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_cookie HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_cookie HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_cookie HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "tracking_domain" category

Resource URI:
`/api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_domain`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_domain HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_domain HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_domain HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_domain HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "tracking_object" category

Resource URI:
`/api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_object`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_object HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_object HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_object HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_object HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "tracking_script" category

Resource URI:
`/api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_script`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_script HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_script HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_script HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/tracking_script HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "travel" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/travel`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/travel HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/travel HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/travel HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/travel HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "violence" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/violence`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/violence HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/violence HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/violence HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/violence HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "warez" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/warez`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/warez HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/warez HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/warez HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/warez HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "weapons" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/weapons`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/weapons HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/weapons HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/weapons HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/weapons HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect the content matching "webmail" category

Resource URI: `/api/atlant/config/v1/scanning/forbidden_uri_categories/webmail`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/forbidden_uri_categories/webmail HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/forbidden_uri_categories/webmail HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/forbidden_uri_categories/webmail HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/forbidden_uri_categories/webmail HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

List of scan HTTP endpoints

Resource URI: `/api/atlant/config/v1/scanning/https_endpoints`

Methods: GET PUT PATCH POST DELETE

Data type: `array`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/https_endpoints HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "address": "127.0.0.1",
    "port": 8082
  }
]
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/https_endpoints HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  {
    "address": "127.0.0.1",
    "port": 8082
  }
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "address": "127.0.0.1",
    "port": 8082
  }
]
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/https_endpoints HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  {
    "address": "127.0.0.1",
    "port": 8082
  }
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "address": "127.0.0.1",
    "port": 8082
  }
]
```

POST

Example request:

```
POST /api/atlant/config/v1/scanning/https_endpoints HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "address": "127.0.0.1",
  "port": 8082
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: URL

{
  "address": "127.0.0.1",
```

```
  "port": 8082
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/https_endpoints HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Scan HTTP endpoint

Keys:

- ENDPOINT

  Example: `{"address":"string","port":0}`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/scanning/https_endpoints/ENDPOINT`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/https_endpoints/ENDPOINT HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "address": "127.0.0.1",
  "port": 8082
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/https_endpoints/ENDPOINT HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "address": "127.0.0.1",
  "port": 8082
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "address": "127.0.0.1",
  "port": 8082
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/https_endpoints/ENDPOINT HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "address": "127.0.0.1",
  "port": 8082
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "address": "127.0.0.1",
  "port": 8082
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/https_endpoints/ENDPOINT HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Scan HTTP endpoint address

Keys:

- ENDPOINT

  Example: `{"address":"string","port":0}`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/scanning/https_endpoints/ENDPOINT/address`

Methods: GET PUT PATCH

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/https_endpoints/ENDPOINT/address HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"127.0.0.1"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/https_endpoints/ENDPOINT/address HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN
```

```
"127.0.0.1"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"127.0.0.1"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/https_endpoints/ENDPOINT/address HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"127.0.0.1"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"127.0.0.1"
```

Scan HTTP endpoint port

Keys:

- ENDPOINT

  Example: `{"address":"string","port":0}`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/scanning/https_endpoints/ENDPOINT/port`

Methods: `GET` `PUT` `PATCH`

Data type: `number`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/https_endpoints/ENDPOINT/port HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

8082
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/https_endpoints/ENDPOINT/port HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

8082
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

8082
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/https_endpoints/ENDPOINT/port HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

8082
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

8082
```

ICAP API settings

Resource URI: `/api/atlant/config/v1/scanning/icap`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/icap HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "block_archive_max_nested": true,
  "block_encrypted_archives": true,
  "detect_pua": true,
  "scan_archives": true,
  "trickle": {
    "enabled": true,
    "interval": 30,
    "max_bytes": 120,
    "start_delay": 0
  }
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/icap HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "block_archive_max_nested": true,
  "block_encrypted_archives": true,
  "detect_pua": true,
  "scan_archives": true,
  "trickle": {
    "enabled": true,
    "interval": 30,
```

```
    "max_bytes": 120,
    "start_delay": 0
  }
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "block_archive_max_nested": true,
  "block_encrypted_archives": true,
  "detect_pua": true,
  "scan_archives": true,
  "trickle": {
    "enabled": true,
    "interval": 30,
    "max_bytes": 120,
    "start_delay": 0
  }
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/icap HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "block_archive_max_nested": true,
  "block_encrypted_archives": true,
  "detect_pua": true,
  "scan_archives": true,
  "trickle": {
    "enabled": true,
    "interval": 30,
    "max_bytes": 120,
    "start_delay": 0
  }
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "block_archive_max_nested": true,
  "block_encrypted_archives": true,
  "detect_pua": true,
  "scan_archives": true,
  "trickle": {
    "enabled": true,
    "interval": 30,
    "max_bytes": 120,
    "start_delay": 0
  }
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/icap HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Block archives that are nested too deeply

Resource URI: `/api/atlant/config/v1/scanning/icap/block_archive_max_nested`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/icap/block_archive_max_nested HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/icap/block_archive_max_nested HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/icap/block_archive_max_nested HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/icap/block_archive_max_nested HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Block encrypted archives

Resource URI: `/api/atlant/config/v1/scanning/icap/block_encrypted_archives`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/icap/block_encrypted_archives HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/icap/block_encrypted_archives HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/icap/block_encrypted_archives HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/icap/block_encrypted_archives HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect potentially unwanted applications

Resource URI: `/api/atlant/config/v1/scanning/icap/detect_pua`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/icap/detect_pua HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/icap/detect_pua HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/icap/detect_pua HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/icap/detect_pua HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Scan content inside archive files

Resource URI: `/api/atlant/config/v1/scanning/icap/scan_archives`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/icap/scan_archives HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/icap/scan_archives HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/icap/scan_archives HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/icap/scan_archives HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

List of scan ICAP endpoints

Resource URI: `/api/atlant/config/v1/scanning/icap_endpoints`

Methods: `GET` `PUT` `PATCH` `POST` `DELETE`

Data type: `array`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/icap_endpoints HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "address": "127.0.0.1",
    "port": 1344
  }
]
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/icap_endpoints HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  {
    "address": "127.0.0.1",
    "port": 1344
  }
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "address": "127.0.0.1",
    "port": 1344
  }
]
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/icap_endpoints HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  {
    "address": "127.0.0.1",
    "port": 1344
  }
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
[
  {
    "address": "127.0.0.1",
    "port": 1344
  }
]
```

POST

Example request:

```
POST /api/atlant/config/v1/scanning/icap_endpoints HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "address": "127.0.0.1",
  "port": 1344
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: URL

{
  "address": "127.0.0.1",
  "port": 1344
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/icap_endpoints HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Scan ICAP endpoint

Keys:

- ENDPOINT

  Example: `{"address":"string","port":0}`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/scanning/icap_endpoints/ENDPOINT`

Methods: GET PUT PATCH DELETE

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/icap_endpoints/ENDPOINT HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
```

```
  "address": "127.0.0.1",
  "port": 1344
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/icap_endpoints/ENDPOINT HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "address": "127.0.0.1",
  "port": 1344
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "address": "127.0.0.1",
  "port": 1344
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/icap_endpoints/ENDPOINT HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "address": "127.0.0.1",
  "port": 1344
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "address": "127.0.0.1",
  "port": 1344
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/icap_endpoints/ENDPOINT HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Scan ICAP endpoint address

Keys:

- ENDPOINT

  Example: `{"address":"string","port":0}`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/scanning/icap_endpoints/ENDPOINT/address`

Methods: `GET` `PUT` `PATCH`

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/icap_endpoints/ENDPOINT/address HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"127.0.0.1"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/icap_endpoints/ENDPOINT/address HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"127.0.0.1"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"127.0.0.1"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/icap_endpoints/ENDPOINT/address HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"127.0.0.1"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"127.0.0.1"
```

Scan ICAP endpoint port

Keys:

- ENDPOINT

  Example: `{"address":"string","port":0}`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/scanning/icap_endpoints/ENDPOINT/port`

Methods: `GET` `PUT` `PATCH`

Data type: `number`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/icap_endpoints/ENDPOINT/port HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

1344
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/icap_endpoints/ENDPOINT/port HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

1344
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

1344
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/icap_endpoints/ENDPOINT/port HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

1344
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

1344
```

Configures trickling to slowly pass through a little bit of content when scan is still in progress

Resource URI: `/api/atlant/config/v1/scanning/icap/trickle`

Methods: GET PUT PATCH DELETE

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/icap/trickle HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
```

```
  "enabled": true,
  "interval": 30,
  "max_bytes": 120,
  "start_delay": 0
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/icap/trickle HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "enabled": true,
  "interval": 30,
  "max_bytes": 120,
  "start_delay": 0
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": true,
  "interval": 30,
  "max_bytes": 120,
  "start_delay": 0
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/icap/trickle HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "enabled": true,
  "interval": 30,
  "max_bytes": 120,
  "start_delay": 0
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": true,
  "interval": 30,
  "max_bytes": 120,
  "start_delay": 0
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/icap/trickle HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Enable ICAP response trickling

Resource URI: `/api/atlant/config/v1/scanning/icap/trickle/enabled`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/icap/trickle/enabled HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/icap/trickle/enabled HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/icap/trickle/enabled HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/icap/trickle/enabled HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Emit single bytes of data at this interval (seconds)

Resource URI: `/api/atlant/config/v1/scanning/icap/trickle/interval`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `number`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/icap/trickle/interval HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

30
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/icap/trickle/interval HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

30
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

30
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/icap/trickle/interval HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

30
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

30
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/icap/trickle/interval HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Trickle at most this many bytes before scan result is available

Resource URI: `/api/atlant/config/v1/scanning/icap/trickle/max_bytes`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `number`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/icap/trickle/max_bytes HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

120
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/icap/trickle/max_bytes HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

120
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

120
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/icap/trickle/max_bytes HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

120
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

120
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/icap/trickle/max_bytes HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Delay starting of trickling by this many seconds

Resource URI: `/api/atlant/config/v1/scanning/icap/trickle/start_delay`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `number`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/icap/trickle/start_delay HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/icap/trickle/start_delay HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

0
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/icap/trickle/start_delay HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

0
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/icap/trickle/start_delay HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

List of scan ICAPS endpoints

Resource URI: `/api/atlant/config/v1/scanning/icaps_endpoints`

Methods: `GET` `PUT` `PATCH` `POST` `DELETE`

Data type: `array`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/icaps_endpoints HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "address": "127.0.0.1",
    "port": 11344
  }
]
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/icaps_endpoints HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  {
    "address": "127.0.0.1",
    "port": 11344
  }
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "address": "127.0.0.1",
    "port": 11344
  }
]
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/icaps_endpoints HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  {
    "address": "127.0.0.1",
    "port": 11344
  }
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
[
  {
    "address": "127.0.0.1",
    "port": 11344
  }
]
```

POST

Example request:

```
POST /api/atlant/config/v1/scanning/icaps_endpoints HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "address": "127.0.0.1",
  "port": 11344
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: URL

{
  "address": "127.0.0.1",
  "port": 11344
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/icaps_endpoints HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Scan ICAPS endpoint

Keys:

- ENDPOINT

  Example: `{"address":"string","port":0}`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/scanning/icaps_endpoints/ENDPOINT`

Methods: GET PUT PATCH DELETE

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/icaps_endpoints/ENDPOINT HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
```

```
  "address": "127.0.0.1",
  "port": 11344
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/icaps_endpoints/ENDPOINT HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "address": "127.0.0.1",
  "port": 11344
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "address": "127.0.0.1",
  "port": 11344
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/icaps_endpoints/ENDPOINT HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "address": "127.0.0.1",
  "port": 11344
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "address": "127.0.0.1",
  "port": 11344
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/icaps_endpoints/ENDPOINT HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Scan ICAPS endpoint address

Keys:

- ENDPOINT

  Example: `{"address":"string","port":0}`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/scanning/icaps_endpoints/ENDPOINT/address`

Methods: `GET` `PUT` `PATCH`

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/icaps_endpoints/ENDPOINT/address HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"127.0.0.1"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/icaps_endpoints/ENDPOINT/address HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"127.0.0.1"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"127.0.0.1"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/icaps_endpoints/ENDPOINT/address HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"127.0.0.1"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"127.0.0.1"
```

Scan ICAPS endpoint port

Keys:

- ENDPOINT

  Example: `{"address":"string","port":0}`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/scanning/icaps_endpoints/ENDPOINT/port`

Methods: `GET` `PUT` `PATCH`

Data type: `number`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/icaps_endpoints/ENDPOINT/port HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

11344
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/icaps_endpoints/ENDPOINT/port HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

11344
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

11344
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/icaps_endpoints/ENDPOINT/port HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

11344
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

11344
```

Reputation check settings

Resource URI: `/api/atlant/config/v1/scanning/reputation`

Methods: GET PUT PATCH DELETE

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/reputation HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "file_check": true,
  "timeout": 5,
```

```
  "url_check": true
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/reputation HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "file_check": true,
  "timeout": 5,
  "url_check": true
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "file_check": true,
  "timeout": 5,
  "url_check": true
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/reputation HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "file_check": true,
  "timeout": 5,
  "url_check": true
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "file_check": true,
  "timeout": 5,
  "url_check": true
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/reputation HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Enable file reputation checks

Resource URI: `/api/atlant/config/v1/scanning/reputation/file_check`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/reputation/file_check HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/reputation/file_check HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/reputation/file_check HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/reputation/file_check HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Enable URL reputation checks

Resource URI: `/api/atlant/config/v1/scanning/reputation/url_check`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/reputation/url_check HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/reputation/url_check HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/reputation/url_check HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/reputation/url_check HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Timeout for reputation requests in seconds

Resource URI: `/api/atlant/config/v1/scanning/reputation/timeout`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `number`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/reputation/timeout HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

5
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/reputation/timeout HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

5
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

5
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/reputation/timeout HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

5
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

5
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/reputation/timeout HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Maximum scan time in seconds

Resource URI: `/api/atlant/config/v1/scanning/max_scan_time`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `number`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/max_scan_time HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

90
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/max_scan_time HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

90
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

90
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/max_scan_time HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

90
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

90
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/max_scan_time HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Keep-alive time in seconds

Resource URI: `/api/atlant/config/v1/scanning/keepalive_time`

Methods: GET PUT PATCH DELETE

Data type: `number`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/keepalive_time HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

600
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/keepalive_time HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

600
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

600
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/keepalive_time HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

600
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

600
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/keepalive_time HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Scan nested archives up to this depth

Resource URI: `/api/atlant/config/v1/scanning/archive_max_nested`

Methods: GET PUT PATCH DELETE

Data type: `number`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/archive_max_nested HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

5
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/archive_max_nested HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

5
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

5
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/archive_max_nested HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

5
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

5
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/archive_max_nested HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Settings for limiting maximum size of scanned files

Resource URI: `/api/atlant/config/v1/scanning/max_scan_size`

Methods: GET PUT PATCH DELETE

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/max_scan_size HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": true,
  "limit": 2147483648
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/max_scan_size HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "enabled": true,
  "limit": 2147483648
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": true,
  "limit": 2147483648
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/max_scan_size HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "enabled": true,
  "limit": 2147483648
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": true,
  "limit": 2147483648
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/max_scan_size HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Enable size limits for scanned files

Resource URI: `/api/atlant/config/v1/scanning/max_scan_size/enabled`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/max_scan_size/enabled HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/max_scan_size/enabled HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/max_scan_size/enabled HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/max_scan_size/enabled HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Maximum size for scanned files

Resource URI: `/api/atlant/config/v1/scanning/max_scan_size/limit`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `number`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/max_scan_size/limit HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

2147483648
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/max_scan_size/limit HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

2147483648
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

2147483648
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/max_scan_size/limit HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

2147483648
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

2147483648
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/max_scan_size/limit HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

4.3.10 Settings for manual scanning

Resource URI: `/api/atlant/config/v1/fsanalyze`

Methods: GET PUT PATCH DELETE

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/fsanalyze HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "actions": {
    "malware": "remove",
    "pua": "rename",
    "spam": "none",
    "suspected": "none"
  },
  "archive_max_nested": 5,
  "block_archive_max_nested": true,
  "block_encrypted_archives": true,
  "detect_pua": true,
  "detect_spam": true,
  "scan_archives": true
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/fsanalyze HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "actions": {
    "malware": "remove",
    "pua": "rename",
    "spam": "none",
    "suspected": "none"
  },
  "archive_max_nested": 5,
  "block_archive_max_nested": true,
  "block_encrypted_archives": true,
  "detect_pua": true,
  "detect_spam": true,
  "scan_archives": true
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "actions": {
    "malware": "remove",
    "pua": "rename",
    "spam": "none",
    "suspected": "none"
  },
  "archive_max_nested": 5,
  "block_archive_max_nested": true,
  "block_encrypted_archives": true,
  "detect_pua": true,
  "detect_spam": true,
  "scan_archives": true
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/fsanalyze HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "actions": {
    "malware": "remove",
    "pua": "rename",
    "spam": "none",
    "suspected": "none"
  },
  "archive_max_nested": 5,
  "block_archive_max_nested": true,
  "block_encrypted_archives": true,
  "detect_pua": true,
  "detect_spam": true,
  "scan_archives": true
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "actions": {
    "malware": "remove",
    "pua": "rename",
    "spam": "none",
    "suspected": "none"
  },
  "archive_max_nested": 5,
  "block_archive_max_nested": true,
  "block_encrypted_archives": true,
  "detect_pua": true,
  "detect_spam": true,
  "scan_archives": true
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/fsanalyze HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

What to do with different kinds of malicious files

Resource URI: `/api/atlant/config/v1/fsanalyze/actions`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/fsanalyze/actions HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "malware": "remove",
  "pua": "rename",
  "spam": "none",
  "suspected": "none"
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/fsanalyze/actions HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "malware": "remove",
  "pua": "rename",
  "spam": "none",
  "suspected": "none"
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "malware": "remove",
  "pua": "rename",
  "spam": "none",
  "suspected": "none"
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/fsanalyze/actions HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "malware": "remove",
  "pua": "rename",
  "spam": "none",
  "suspected": "none"
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "malware": "remove",
  "pua": "rename",
  "spam": "none",
  "suspected": "none"
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/fsanalyze/actions HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Manual scanning action for malware

Value must be one of `"none"`, `"rename"`, `"remove"`

Resource URI: `/api/atlant/config/v1/fsanalyze/actions/malware`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/fsanalyze/actions/malware HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"remove"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/fsanalyze/actions/malware HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"remove"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"remove"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/fsanalyze/actions/malware HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"remove"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"remove"
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/fsanalyze/actions/malware HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Manual scanning action for PUA

Value must be one of `"none"`, `"rename"`, `"remove"`

Resource URI: `/api/atlant/config/v1/fsanalyze/actions/pua`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/fsanalyze/actions/pua HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"rename"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/fsanalyze/actions/pua HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"rename"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"rename"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/fsanalyze/actions/pua HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"rename"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"rename"
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/fsanalyze/actions/pua HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Manual scanning action for suspicious files

Value must be one of `"none"`, `"rename"`, `"remove"`

Resource URI: `/api/atlant/config/v1/fsanalyze/actions/suspected`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/fsanalyze/actions/suspected HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"none"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/fsanalyze/actions/suspected HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"none"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"none"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/fsanalyze/actions/suspected HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"none"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"none"
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/fsanalyze/actions/suspected HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Manual scanning action for files with spam and phishing content

Value must be one of `"none"`, `"rename"`, `"remove"`

Resource URI: `/api/atlant/config/v1/fsanalyze/actions/spam`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/fsanalyze/actions/spam HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"none"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/fsanalyze/actions/spam HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"none"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"none"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/fsanalyze/actions/spam HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"none"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"none"
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/fsanalyze/actions/spam HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Extract and scan files from archives

Resource URI: `/api/atlant/config/v1/fsanalyze/scan_archives`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/fsanalyze/scan_archives HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/fsanalyze/scan_archives HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/fsanalyze/scan_archives HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/fsanalyze/scan_archives HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Scan nested archives up to this depth

Resource URI: `/api/atlant/config/v1/fsanalyze/archive_max_nested`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `number`

GET

Example request:

```
GET /api/atlant/config/v1/fsanalyze/archive_max_nested HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

5
```

PUT

Example request:

```
PUT /api/atlant/config/v1/fsanalyze/archive_max_nested HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

5
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

5
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/fsanalyze/archive_max_nested HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

5
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

5
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/fsanalyze/archive_max_nested HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Block archives that are not fully scanned because of the configured archive_max_nested value

Resource URI: `/api/atlant/config/v1/fsanalyze/block_archive_max_nested`

Methods: `GET PUT PATCH DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/fsanalyze/block_archive_max_nested HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/fsanalyze/block_archive_max_nested HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/fsanalyze/block_archive_max_nested HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/fsanalyze/block_archive_max_nested HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Block encrypted archives

Resource URI: `/api/atlant/config/v1/fsanalyze/block_encrypted_archives`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/fsanalyze/block_encrypted_archives HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/fsanalyze/block_encrypted_archives HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/fsanalyze/block_encrypted_archives HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/fsanalyze/block_encrypted_archives HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect potentially unwanted applications

Resource URI: `/api/atlant/config/v1/fsanalyze/detect_pua`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/fsanalyze/detect_pua HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/fsanalyze/detect_pua HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/fsanalyze/detect_pua HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/fsanalyze/detect_pua HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Detect spam and phishing content

Resource URI: `/api/atlant/config/v1/fsanalyze/detect_spam`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/fsanalyze/detect_spam HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/fsanalyze/detect_spam HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/fsanalyze/detect_spam HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/fsanalyze/detect_spam HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

4.3.11 Logging settings

Resource URI: `/api/atlant/config/v1/scanning/logging`

Methods: GET PUT PATCH DELETE

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "https": {
    "request_headers": [
      "User-Agent"
    ]
  },
  "icap": {
    "encapsulated_request_headers": [
      "User-Agent"
    ],
    "encapsulated_response_headers": [
      "ETag"
    ],
    "request_headers": [
      "X-Client-Username"
    ]
  },
  "ip_address": true,
  "rotation": {
    "file_size": {
      "enabled": true,
      "limit": 10000000
    },
    "max_bytes": {
      "enabled": false,
      "limit": 100000000
    },
    "max_days": {
      "enabled": false,
      "limit": 30
    },
    "max_files": {
      "enabled": true,
      "limit": 10
    }
  }
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "https": {
    "request_headers": [
      "User-Agent"
    ]
  },
  "icap": {
    "encapsulated_request_headers": [
      "User-Agent"
    ],
    "encapsulated_response_headers": [
      "ETag"
```

```
    ],
    "request_headers": [
      "X-Client-Username"
    ]
  },
  "ip_address": true,
  "rotation": {
    "file_size": {
      "enabled": true,
      "limit": 10000000
    },
    "max_bytes": {
      "enabled": false,
      "limit": 100000000
    },
    "max_days": {
      "enabled": false,
      "limit": 30
    },
    "max_files": {
      "enabled": true,
      "limit": 10
    }
  }
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "https": {
    "request_headers": [
      "User-Agent"
    ]
  },
  "icap": {
    "encapsulated_request_headers": [
      "User-Agent"
    ],
    "encapsulated_response_headers": [
      "ETag"
    ],
    "request_headers": [
      "X-Client-Username"
    ]
  },
  "ip_address": true,
  "rotation": {
    "file_size": {
      "enabled": true,
      "limit": 10000000
    },
    "max_bytes": {
      "enabled": false,
      "limit": 100000000
    },
    "max_days": {
      "enabled": false,
      "limit": 30
    },
    "max_files": {
      "enabled": true,
      "limit": 10
    }
  }
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "https": {
    "request_headers": [
      "User-Agent"
```

```
    ]
  },
  "icap": {
    "encapsulated_request_headers": [
      "User-Agent"
    ],
    "encapsulated_response_headers": [
      "ETag"
    ],
    "request_headers": [
      "X-Client-Username"
    ]
  },
  "ip_address": true,
  "rotation": {
    "file_size": {
      "enabled": true,
      "limit": 10000000
    },
    "max_bytes": {
      "enabled": false,
      "limit": 100000000
    },
    "max_days": {
      "enabled": false,
      "limit": 30
    },
    "max_files": {
      "enabled": true,
      "limit": 10
    }
  }
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "https": {
    "request_headers": [
      "User-Agent"
    ]
  },
  "icap": {
    "encapsulated_request_headers": [
      "User-Agent"
    ],
    "encapsulated_response_headers": [
      "ETag"
    ],
    "request_headers": [
      "X-Client-Username"
    ]
  },
  "ip_address": true,
  "rotation": {
    "file_size": {
      "enabled": true,
      "limit": 10000000
    },
    "max_bytes": {
      "enabled": false,
      "limit": 100000000
    },
    "max_days": {
      "enabled": false,
      "limit": 30
    },
    "max_files": {
      "enabled": true,
      "limit": 10
    }
  }
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Logging settings for REST scanning API

Resource URI: `/api/atlant/config/v1/scanning/logging/https`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/https HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "request_headers": [
    "User-Agent"
  ]
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/https HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "request_headers": [
    "User-Agent"
  ]
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "request_headers": [
    "User-Agent"
  ]
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/https HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "request_headers": [
```

```
    "User-Agent"
  ]
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "request_headers": [
    "User-Agent"
  ]
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/https HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

List of HTTP request headers to log

Resource URI: `/api/atlant/config/v1/scanning/logging/https/request_headers`

Methods: GET PUT PATCH POST DELETE

Data type: `array`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/https/request_headers HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  "User-Agent"
]
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/https/request_headers HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  "User-Agent"
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  "User-Agent"
]
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/https/request_headers HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  "User-Agent"
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  "User-Agent"
]
```

POST

Example request:

```
POST /api/atlant/config/v1/scanning/logging/https/request_headers HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"User-Agent"
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: URL

"User-Agent"
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/https/request_headers HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

HTTP header to log

Keys:

- HEADER

  Example: `"string"`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/scanning/logging/https/request_headers/HEADER`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/https/request_headers/HEADER HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"User-Agent"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/https/request_headers/HEADER HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"User-Agent"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"User-Agent"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/https/request_headers/HEADER HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"User-Agent"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"User-Agent"
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/https/request_headers/HEADER HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Logging settings for ICAP scanning API

Resource URI: `/api/atlant/config/v1/scanning/logging/icap`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/icap HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "encapsulated_request_headers": [
    "User-Agent"
  ],
  "encapsulated_response_headers": [
    "ETag"
  ],
  "request_headers": [
    "X-Client-Username"
  ]
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/icap HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "encapsulated_request_headers": [
    "User-Agent"
  ],
  "encapsulated_response_headers": [
    "ETag"
  ],
  "request_headers": [
    "X-Client-Username"
  ]
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "encapsulated_request_headers": [
    "User-Agent"
  ],
  "encapsulated_response_headers": [
    "ETag"
  ],
  "request_headers": [
    "X-Client-Username"
  ]
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/icap HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "encapsulated_request_headers": [
    "User-Agent"
  ],
  "encapsulated_response_headers": [
    "ETag"
  ],
  "request_headers": [
```

```
    "X-Client-Username"
  ]
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "encapsulated_request_headers": [
    "User-Agent"
  ],
  "encapsulated_response_headers": [
    "ETag"
  ],
  "request_headers": [
    "X-Client-Username"
  ]
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/icap HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

List of encapsulated request headers to log

Resource URI:
`/api/atlant/config/v1/scanning/logging/icap/encapsulated_request_headers`

Methods: GET PUT PATCH POST DELETE

Data type: `array`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/icap/encapsulated_request_headers HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  "User-Agent"
]
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/icap/encapsulated_request_headers HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  "User-Agent"
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  "User-Agent"
]
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/icap/encapsulated_request_headers HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  "User-Agent"
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  "User-Agent"
]
```

POST

Example request:

```
POST /api/atlant/config/v1/scanning/logging/icap/encapsulated_request_headers HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"User-Agent"
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: URL

"User-Agent"
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/icap/encapsulated_request_headers HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

HTTP header to log

Keys:

- HEADER

  Example: `"string"`

Key should be URL-encoded

Resource URI:
`/api/atlant/config/v1/scanning/logging/icap/encapsulated_request_headers/HEADER`

Methods: GET PUT PATCH DELETE

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/icap/encapsulated_request_headers/HEADER HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"User-Agent"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/icap/encapsulated_request_headers/HEADER HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"User-Agent"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"User-Agent"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/icap/encapsulated_request_headers/HEADER HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"User-Agent"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"User-Agent"
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/icap/encapsulated_request_headers/HEADER HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

List of encapsulated response headers to log

Resource URI:
`/api/atlant/config/v1/scanning/logging/icap/encapsulated_response_headers`

Methods: GET PUT PATCH POST DELETE

Data type: `array`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/icap/encapsulated_response_headers HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  "ETag"
]
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/icap/encapsulated_response_headers HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  "ETag"
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  "ETag"
]
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/icap/encapsulated_response_headers HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  "ETag"
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  "ETag"
]
```

POST

Example request:

```
POST /api/atlant/config/v1/scanning/logging/icap/encapsulated_response_headers HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"ETag"
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: URL

"ETag"
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/icap/encapsulated_response_headers HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

HTTP header to log

Keys:

- HEADER

  Example: `"string"`

Key should be URL-encoded

Resource URI:
`/api/atlant/config/v1/scanning/logging/icap/encapsulated_response_headers/HEADER`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/icap/encapsulated_response_headers/HEADER HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"ETag"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/icap/encapsulated_response_headers/HEADER HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"ETag"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"ETag"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/icap/encapsulated_response_headers/HEADER HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"ETag"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"ETag"
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/icap/encapsulated_response_headers/HEADER HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

List of ICAP headers to log

Resource URI: `/api/atlant/config/v1/scanning/logging/icap/request_headers`

Methods: `GET` `PUT` `PATCH` `POST` `DELETE`

Data type: `array`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/icap/request_headers HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  "X-Client-Username"
]
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/icap/request_headers HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  "X-Client-Username"
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
```

```
  "X-Client-Username"
]
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/icap/request_headers HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

[
  "X-Client-Username"
]
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  "X-Client-Username"
]
```

POST

Example request:

```
POST /api/atlant/config/v1/scanning/logging/icap/request_headers HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"X-Client-Username"
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: URL

"X-Client-Username"
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/icap/request_headers HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

ICAP header to log

Keys:

- HEADER

  Example: `"string"`

Key should be URL-encoded

Resource URI: `/api/atlant/config/v1/scanning/logging/icap/request_headers/HEADER`

Methods: `GET PUT PATCH DELETE`

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/icap/request_headers/HEADER HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"X-Client-Username"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/icap/request_headers/HEADER HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"X-Client-Username"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"X-Client-Username"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/icap/request_headers/HEADER HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"X-Client-Username"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"X-Client-Username"
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/icap/request_headers/HEADER HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Enable logging of client IP addresses

Resource URI: `/api/atlant/config/v1/scanning/logging/ip_address`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/ip_address HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/ip_address HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/ip_address HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/ip_address HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Log rotation settings

Resource URI: `/api/atlant/config/v1/scanning/logging/rotation`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/rotation HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "file_size": {
    "enabled": true,
    "limit": 10000000
  },
  "max_bytes": {
    "enabled": false,
    "limit": 100000000
  },
  "max_days": {
    "enabled": false,
    "limit": 30
  },
  "max_files": {
    "enabled": true,
    "limit": 10
  }
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/rotation HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "file_size": {
    "enabled": true,
    "limit": 10000000
  },
  "max_bytes": {
    "enabled": false,
    "limit": 100000000
  },
  "max_days": {
    "enabled": false,
    "limit": 30
  },
  "max_files": {
    "enabled": true,
    "limit": 10
  }
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "file_size": {
    "enabled": true,
    "limit": 10000000
  },
  "max_bytes": {
    "enabled": false,
    "limit": 100000000
  },
  "max_days": {
    "enabled": false,
    "limit": 30
  },
  "max_files": {
    "enabled": true,
    "limit": 10
```

```
  }
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/rotation HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "file_size": {
    "enabled": true,
    "limit": 10000000
  },
  "max_bytes": {
    "enabled": false,
    "limit": 100000000
  },
  "max_days": {
    "enabled": false,
    "limit": 30
  },
  "max_files": {
    "enabled": true,
    "limit": 10
  }
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "file_size": {
    "enabled": true,
    "limit": 10000000
  },
  "max_bytes": {
    "enabled": false,
    "limit": 100000000
  },
  "max_days": {
    "enabled": false,
    "limit": 30
  },
  "max_files": {
    "enabled": true,
    "limit": 10
  }
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/rotation HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Log file rotation size settings

Resource URI: `/api/atlant/config/v1/scanning/logging/rotation/file_size`

Methods: GET PUT PATCH DELETE

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/rotation/file_size HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": true,
  "limit": 10000000
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/rotation/file_size HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "enabled": true,
  "limit": 10000000
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": true,
  "limit": 10000000
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/rotation/file_size HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "enabled": true,
  "limit": 10000000
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": true,
  "limit": 10000000
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/rotation/file_size HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Enable access log rotation

Resource URI: `/api/atlant/config/v1/scanning/logging/rotation/file_size/enabled`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/rotation/file_size/enabled HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/rotation/file_size/enabled HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/rotation/file_size/enabled HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/rotation/file_size/enabled HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Access log rotation size in bytes

Resource URI: `/api/atlant/config/v1/scanning/logging/rotation/file_size/limit`

Methods: GET PUT PATCH DELETE

Data type: number

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/rotation/file_size/limit HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

10000000
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/rotation/file_size/limit HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

10000000
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

10000000
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/rotation/file_size/limit HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

10000000
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

10000000
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/rotation/file_size/limit HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Settings for controlling the maximum cumulative size of rotated access log files.

Resource URI: /api/atlant/config/v1/scanning/logging/rotation/max_bytes

Methods: GET PUT PATCH DELETE

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/rotation/max_bytes HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": false,
  "limit": 100000000
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/rotation/max_bytes HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "enabled": false,
  "limit": 100000000
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": false,
  "limit": 100000000
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/rotation/max_bytes HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "enabled": false,
  "limit": 100000000
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": false,
  "limit": 100000000
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/rotation/max_bytes HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Enable limit on the maximum cumulative size of rotated access log files (in bytes)

Resource URI: `/api/atlant/config/v1/scanning/logging/rotation/max_bytes/enabled`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/rotation/max_bytes/enabled HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

false
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/rotation/max_bytes/enabled HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

false
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

false
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/rotation/max_bytes/enabled HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

false
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

false
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/rotation/max_bytes/enabled HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Limit on the maximum cumulative size of rotated access log files (in bytes)

Resource URI: `/api/atlant/config/v1/scanning/logging/rotation/max_bytes/limit`

Methods: GET PUT PATCH DELETE

Data type: `number`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/rotation/max_bytes/limit HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

100000000
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/rotation/max_bytes/limit HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

100000000
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

100000000
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/rotation/max_bytes/limit HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

100000000
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

100000000
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/rotation/max_bytes/limit HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Settings for controlling the maximum number of days that rotated access log files.

Resource URI: `/api/atlant/config/v1/scanning/logging/rotation/max_days`

Methods: GET PUT PATCH DELETE

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/rotation/max_days HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": false,
  "limit": 30
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/rotation/max_days HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "enabled": false,
  "limit": 30
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": false,
  "limit": 30
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/rotation/max_days HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "enabled": false,
  "limit": 30
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "enabled": false,
  "limit": 30
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/rotation/max_days HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Enable limit on the maximum number of days that rotated access log files will be kept

Resource URI: `/api/atlant/config/v1/scanning/logging/rotation/max_days/enabled`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/rotation/max_days/enabled HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

false
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/rotation/max_days/enabled HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

false
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

false
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/rotation/max_days/enabled HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

false
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

false
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/rotation/max_days/enabled HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Limit on the maximum number of days that rotated access log files will be kept

Resource URI: `/api/atlant/config/v1/scanning/logging/rotation/max_days/limit`

Methods: GET PUT PATCH DELETE

Data type: `number`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/rotation/max_days/limit HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

30
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/rotation/max_days/limit HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

30
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

30
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/rotation/max_days/limit HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

30
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

30
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/rotation/max_days/limit HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Settings for controlling the maximum number of rotated access log files.

Resource URI: `/api/atlant/config/v1/scanning/logging/rotation/max_files`

Methods: GET PUT PATCH DELETE

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/rotation/max_files HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": true,
  "limit": 10
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/rotation/max_files HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "enabled": true,
  "limit": 10
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": true,
  "limit": 10
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/rotation/max_files HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "enabled": true,
  "limit": 10
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": true,
  "limit": 10
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/rotation/max_files HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Enable limit on the maximum number of rotated access log files

Resource URI: `/api/atlant/config/v1/scanning/logging/rotation/max_files/enabled`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/rotation/max_files/enabled HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/rotation/max_files/enabled HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/rotation/max_files/enabled HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/rotation/max_files/enabled HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Limit on the maximum number of rotated access log files

Resource URI: `/api/atlant/config/v1/scanning/logging/rotation/max_files/limit`

Methods: GET PUT PATCH DELETE

Data type: `number`

GET

Example request:

```
GET /api/atlant/config/v1/scanning/logging/rotation/max_files/limit HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

10
```

PUT

Example request:

```
PUT /api/atlant/config/v1/scanning/logging/rotation/max_files/limit HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

10
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

10
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/scanning/logging/rotation/max_files/limit HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

10
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

10
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/scanning/logging/rotation/max_files/limit HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

4.3.12 Scanning platform statistics

Resource URI: `/api/atlant/stats/v1/scan_service`

Methods: `GET`

Data type: `object`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "avg_transaction_latency": {
    "daily": 0,
    "monthly": 0,
    "weekly": 0
  },
  "connection_count": 0,
  "detection_count": {
    "daily": 0,
    "monthly": 0,
    "total": 0,
    "weekly": 0
  },
  "last_detection": "",
  "max_connection_count": {
    "daily": 0,
    "monthly": 0,
    "weekly": 0
  },
  "max_transaction_latency": {
    "daily": 0,
    "monthly": 0,
    "weekly": 0
  },
  "min_transaction_latency": {
    "daily": 0,
```

```
    "monthly": 0,
    "weekly": 0
  },
  "transaction_count": {
    "daily": 0,
    "monthly": 0,
    "total": 0,
    "weekly": 0
  }
}
```

Average transaction latency (milliseconds) in different periods

Resource URI: `/api/atlant/stats/v1/scan_service/avg_transaction_latency`

Methods: `GET`

Data type: `object`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/avg_transaction_latency HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "daily": 0,
  "monthly": 0,
  "weekly": 0
}
```

Average transaction latency (milliseconds) in the last 24 hours

Resource URI: `/api/atlant/stats/v1/scan_service/avg_transaction_latency/daily`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/avg_transaction_latency/daily HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Average transaction latency (milliseconds) in the last month

Resource URI: `/api/atlant/stats/v1/scan_service/avg_transaction_latency/monthly`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/avg_transaction_latency/monthly HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Average transaction latency (milliseconds) in the last week

Resource URI: `/api/atlant/stats/v1/scan_service/avg_transaction_latency/weekly`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/avg_transaction_latency/weekly HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Number of connected hosts

Resource URI: `/api/atlant/stats/v1/scan_service/connection_count`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/connection_count HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Number of detections in different periods

Resource URI: `/api/atlant/stats/v1/scan_service/detection_count`

Methods: `GET`

Data type: `object`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/detection_count HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "daily": 0,
  "monthly": 0,
  "total": 0,
  "weekly": 0
}
```

Number of detections in the last 24 hours

Resource URI: `/api/atlant/stats/v1/scan_service/detection_count/daily`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/detection_count/daily HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Number of detections in the last month

Resource URI: `/api/atlant/stats/v1/scan_service/detection_count/monthly`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/detection_count/monthly HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Current number of detections

Resource URI: `/api/atlant/stats/v1/scan_service/detection_count/total`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/detection_count/total HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Number of detections in the last week

Resource URI: `/api/atlant/stats/v1/scan_service/detection_count/weekly`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/detection_count/weekly HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Name of the last detection

Resource URI: `/api/atlant/stats/v1/scan_service/last_detection`

Methods: `GET`

Data type: `string`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/last_detection HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

""
```

Maximum number of connected hosts in different periods

Resource URI: `/api/atlant/stats/v1/scan_service/max_connection_count`

Methods: `GET`

Data type: `object`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/max_connection_count HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "daily": 0,
  "monthly": 0,
  "weekly": 0
}
```

Maximum number of connected hosts in the last 24 hours

Resource URI: `/api/atlant/stats/v1/scan_service/max_connection_count/daily`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/max_connection_count/daily HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Maximum number of connected hosts in the last month

Resource URI: `/api/atlant/stats/v1/scan_service/max_connection_count/monthly`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/max_connection_count/monthly HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Maximum number of connected hosts in the last week

Resource URI: `/api/atlant/stats/v1/scan_service/max_connection_count/weekly`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/max_connection_count/weekly HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Maximum transaction latency (milliseconds) in different periods

Resource URI: `/api/atlant/stats/v1/scan_service/max_transaction_latency`

Methods: `GET`

Data type: `object`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/max_transaction_latency HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "daily": 0,
  "monthly": 0,
  "weekly": 0
}
```

Maximum transaction latency (milliseconds) in the last 24 hours

Resource URI: `/api/atlant/stats/v1/scan_service/max_transaction_latency/daily`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/max_transaction_latency/daily HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Maximum transaction latency (milliseconds) in the last month

Resource URI: `/api/atlant/stats/v1/scan_service/max_transaction_latency/monthly`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/max_transaction_latency/monthly HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Maximum transaction latency (milliseconds) in the last week

Resource URI: `/api/atlant/stats/v1/scan_service/max_transaction_latency/weekly`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/max_transaction_latency/weekly HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Minimum transaction latency (milliseconds) in different periods

Resource URI: `/api/atlant/stats/v1/scan_service/min_transaction_latency`

Methods: `GET`

Data type: `object`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/min_transaction_latency HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "daily": 0,
  "monthly": 0,
  "weekly": 0
}
```

Minimum transaction latency (milliseconds) in the last 24 hours

Resource URI: `/api/atlant/stats/v1/scan_service/min_transaction_latency/daily`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/min_transaction_latency/daily HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Minimum transaction latency (milliseconds) in the last month

Resource URI: `/api/atlant/stats/v1/scan_service/min_transaction_latency/monthly`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/min_transaction_latency/monthly HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Minimum transaction latency (milliseconds) in the last week

Resource URI: `/api/atlant/stats/v1/scan_service/min_transaction_latency/weekly`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/min_transaction_latency/weekly HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Number of transactions in different periods

Resource URI: `/api/atlant/stats/v1/scan_service/transaction_count`

Methods: `GET`

Data type: `object`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/transaction_count HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "daily": 0,
  "monthly": 0,
  "total": 0,
  "weekly": 0
}
```

Number of transactions in the last 24 hours

Resource URI: `/api/atlant/stats/v1/scan_service/transaction_count/daily`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/transaction_count/daily HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Number of transactions in the last month

Resource URI: `/api/atlant/stats/v1/scan_service/transaction_count/monthly`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/transaction_count/monthly HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Current number of transactions

Resource URI: `/api/atlant/stats/v1/scan_service/transaction_count/total`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/transaction_count/total HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Number of transactions in the last week

Resource URI: `/api/atlant/stats/v1/scan_service/transaction_count/weekly`

Methods: `GET`

Data type: `number`

GET

Example request:

```
GET /api/atlant/stats/v1/scan_service/transaction_count/weekly HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

0
```

Latest definition update status

Resource URI: `/api/atlant/stats/v1/latest_definition_update`

Methods: `GET`

Data type: `object`

GET

Example request:

```
GET /api/atlant/stats/v1/latest_definition_update HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "time": "2000-01-01T00:00:00Z"
}
```

Time and date of the latest definition update in ISO 8601 format

Resource URI: `/api/atlant/stats/v1/latest_definition_update/time`

Methods: `GET`

Data type: `string`

GET

Example request:

```
GET /api/atlant/stats/v1/latest_definition_update/time HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"2000-01-01T00:00:00Z"
```

Subscription status

Resource URI: `/api/atlant/stats/v1/subscription`

Methods: `GET`

Data type: `object`

GET

Example request:

```
GET /api/atlant/stats/v1/subscription HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "expiry_date": "2000-01-01T00:00:00Z",
  "is_valid": true,
  "product_variant": ""
}
```

Expiration date in ISO 8601 format

Resource URI: `/api/atlant/stats/v1/subscription/expiry_date`

Methods: `GET`

Data type: `string`

GET

Example request:

```
GET /api/atlant/stats/v1/subscription/expiry_date HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"2000-01-01T00:00:00Z"
```

Is subscription valid

Resource URI: `/api/atlant/stats/v1/subscription/is_valid`

Methods: `GET`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/stats/v1/subscription/is_valid HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

Product variant

Resource URI: `/api/atlant/stats/v1/subscription/product_variant`

Methods: `GET`

Data type: `string`

GET

Example request:

```
GET /api/atlant/stats/v1/subscription/product_variant HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

""
```

4.3.13 Pinned product version

Resource URI: `/api/atlant/stats/v1/product_version`

Methods: `GET`

Data type: `object`

GET

Example request:

```
GET /api/atlant/stats/v1/product_version HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "name": "example-product-version",
  "status": "applied"
}
```

Product version name

Resource URI: `/api/atlant/stats/v1/product_version/name`

Methods: `GET`

Data type: `string`

GET

Example request:

```
GET /api/atlant/stats/v1/product_version/name HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"example-product-version"
```

Whether or not the product version is successfully applied

Value can be one of `"unset"`, `"applied"`, `"in-progress"`, `"error"`

Resource URI: `/api/atlant/stats/v1/product_version/status`

Methods: `GET`

Data type: `string`

GET

Example request:

```
GET /api/atlant/stats/v1/product_version/status HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"applied"
```

4.3.14 Product updates settings

Resource URI: `/api/atlant/config/v1/updates`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/updates HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": true,
  "product_version": "deferred",
  "schedule": {
    "at_date_schedule": 1566565044,
    "regime": "on_arrival",
    "repetition_schedule": {
      "day": "monday",
      "time_of_day": "23:45"
    }
  },
  "use_tls": true
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/updates HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "enabled": true,
  "product_version": "deferred",
  "schedule": {
    "at_date_schedule": 1566565044,
    "regime": "on_arrival",
    "repetition_schedule": {
      "day": "monday",
      "time_of_day": "23:45"
    }
  },
  "use_tls": true
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": true,
  "product_version": "deferred",
  "schedule": {
    "at_date_schedule": 1566565044,
    "regime": "on_arrival",
    "repetition_schedule": {
      "day": "monday",
      "time_of_day": "23:45"
    }
  },
  "use_tls": true
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/updates HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "enabled": true,
  "product_version": "deferred",
  "schedule": {
    "at_date_schedule": 1566565044,
    "regime": "on_arrival",
    "repetition_schedule": {
      "day": "monday",
      "time_of_day": "23:45"
    }
  },
  "use_tls": true
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "enabled": true,
  "product_version": "deferred",
  "schedule": {
    "at_date_schedule": 1566565044,
    "regime": "on_arrival",
    "repetition_schedule": {
      "day": "monday",
      "time_of_day": "23:45"
    }
  },
```

```
  "use_tls": true
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/updates HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Toggle automatic updates

Resource URI: `/api/atlant/config/v1/updates/enabled`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/updates/enabled HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/updates/enabled HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/updates/enabled HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/updates/enabled HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Pin the specified product version.

Resource URI: `/api/atlant/config/v1/updates/product_version`

Methods: `GET` `PUT` `PATCH`

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/updates/product_version HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"deferred"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/updates/product_version HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"deferred"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"deferred"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/updates/product_version HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"deferred"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"deferred"
```

Update schedule

Resource URI: `/api/atlant/config/v1/updates/schedule`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/updates/schedule HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "at_date_schedule": 1566565044,
  "regime": "on_arrival",
  "repetition_schedule": {
    "day": "monday",
    "time_of_day": "23:45"
  }
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/updates/schedule HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "at_date_schedule": 1566565044,
  "regime": "on_arrival",
  "repetition_schedule": {
    "day": "monday",
    "time_of_day": "23:45"
  }
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "at_date_schedule": 1566565044,
  "regime": "on_arrival",
  "repetition_schedule": {
    "day": "monday",
    "time_of_day": "23:45"
  }
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/updates/schedule HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "at_date_schedule": 1566565044,
  "regime": "on_arrival",
  "repetition_schedule": {
    "day": "monday",
    "time_of_day": "23:45"
```

```
  }
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "at_date_schedule": 1566565044,
  "regime": "on_arrival",
  "repetition_schedule": {
    "day": "monday",
    "time_of_day": "23:45"
  }
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/updates/schedule HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Date of update at epoch time format

Resource URI: `/api/atlant/config/v1/updates/schedule/at_date_schedule`

Methods: GET PUT PATCH DELETE

Data type: `number`

GET

Example request:

```
GET /api/atlant/config/v1/updates/schedule/at_date_schedule HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

1566565044
```

PUT

Example request:

```
PUT /api/atlant/config/v1/updates/schedule/at_date_schedule HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

1566565044
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

1566565044
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/updates/schedule/at_date_schedule HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

1566565044
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

1566565044
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/updates/schedule/at_date_schedule HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Scheduling type for updates

Value must be one of `"on_arrival"`, `"at_date"`, `"with_repetition"`

Resource URI: `/api/atlant/config/v1/updates/schedule/regime`

Methods: GET PUT PATCH DELETE

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/updates/schedule/regime HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"on_arrival"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/updates/schedule/regime HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"on_arrival"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"on_arrival"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/updates/schedule/regime HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"on_arrival"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"on_arrival"
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/updates/schedule/regime HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Update repetition schedule

Resource URI: `/api/atlant/config/v1/updates/schedule/repetition_schedule`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `object`

GET

Example request:

```
GET /api/atlant/config/v1/updates/schedule/repetition_schedule HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "day": "monday",
  "time_of_day": "23:45"
}
```

PUT

Example request:

```
PUT /api/atlant/config/v1/updates/schedule/repetition_schedule HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "day": "monday",
  "time_of_day": "23:45"
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "day": "monday",
  "time_of_day": "23:45"
}
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/updates/schedule/repetition_schedule HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "day": "monday",
  "time_of_day": "23:45"
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "day": "monday",
  "time_of_day": "23:45"
}
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/updates/schedule/repetition_schedule HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Scheduled updates day

Value must be one of `"monday"`, `"tuesday"`, `"wednesday"`, `"thursday"`, `"friday"`, `"saturday"`, `"sunday"`, `"daily"`

Resource URI: `/api/atlant/config/v1/updates/schedule/repetition_schedule/day`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/updates/schedule/repetition_schedule/day HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"monday"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/updates/schedule/repetition_schedule/day HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"monday"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"monday"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/updates/schedule/repetition_schedule/day HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"monday"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"monday"
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/updates/schedule/repetition_schedule/day HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Scheduled updates time

Resource URI:
`/api/atlant/config/v1/updates/schedule/repetition_schedule/time_of_day`

Methods: `GET` `PUT` `PATCH` `DELETE`

Data type: `string`

GET

Example request:

```
GET /api/atlant/config/v1/updates/schedule/repetition_schedule/time_of_day HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"23:45"
```

PUT

Example request:

```
PUT /api/atlant/config/v1/updates/schedule/repetition_schedule/time_of_day HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"23:45"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"23:45"
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/updates/schedule/repetition_schedule/time_of_day HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

"23:45"
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

"23:45"
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/updates/schedule/repetition_schedule/time_of_day HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Use TLS to communicate with the update service

Resource URI: `/api/atlant/config/v1/updates/use_tls`

Methods: GET PUT PATCH DELETE

Data type: `boolean`

GET

Example request:

```
GET /api/atlant/config/v1/updates/use_tls HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PUT

Example request:

```
PUT /api/atlant/config/v1/updates/use_tls HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

PATCH

Example request:

```
PATCH /api/atlant/config/v1/updates/use_tls HTTP/1.1
Content-Type: application/json
Authorization: Bearer TOKEN

true
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

DELETE

Example request:

```
DELETE /api/atlant/config/v1/updates/use_tls HTTP/1.1
Authorization: Bearer TOKEN
```

Example response:

```
HTTP/1.1 204 No Content
```

Chapter
5

Configuring Atlant

Topics:

This section outlines how to configure WithSecure Atlant.

- In the standalone mode, Atlant can be configured locally using `atlantctl` command-line utility or remotely using Atlant's Management API.
- In the Policy Manager managed mode, Atlant's configuration is centrally managed using Policy Manager Console, which allows you to configure multiple instances at the same time.

Note: In the Policy Manager managed mode, changes to individual instances configuration can be still made locally using the `atlantctl` command-line utility or remotely using Atlant's Management API.

For more information on Atlant's Management API see Management API on page 39.

5.1 Configuring settings with the atlantctl utility

In the standalone mode, Atlant can be configured locally using atlantctl command-line utility.

5.1.1 Configuring settings with the atlantctl utility

You can check and edit the product settings using the `atlantctl` command-line utility.

Note: The command path given here is for the current version of the product. If you are using an older version, use `/opt/f-secure/atlant/bin/<command>` as the path instead.

To use the `atlantctl` utility:

1. Log in to the Linux host as `root`.
2. Use the following commands:
   - To see the current value of a setting, run `/opt/f-secure/atlant/atlant/bin/atlantctl get [OPTIONS] [SETTING]`
   - To assign a new value to a setting, run `/opt/f-secure/atlant/atlant/bin/atlantctl set [OPTIONS] [SETTING] [VALUE]`
   - To add an entry to a setting that holds several values, run `/opt/f-secure/atlant/atlant/bin/atlantctl add [OPTIONS] [SETTING] [VALUE]`
   - To remove an entry from a setting that holds several values, run `/opt/f-secure/atlant/atlant/bin/atlantctl del [OPTIONS] [SETTING] [KEY]`
   - To clear the value of a setting, run `/opt/f-secure/atlant/atlant/bin/atlantctl reset [OPTION] [SETTING]`
   - To see a list of the operations available for a setting, run `/opt/f-secure/atlant/atlant/bin/atlantctl help [OPTIONS] [SETTING]`
   - To load product settings from a backup, run `/opt/f-secure/atlant/atlant/bin/atlantctl load [OPTIONS] [SETTING] [VALUE]`
   - To get a list of all available settings, run `/opt/f-secure/atlant/atlant/bin/atlantctl -h`

The `get`, `set`, `add`, `del`, and `load` commands support the following options:

--json, -j	Treat input as JSON and produce output in JSON format.
--raw, -r	This is currently the default. Raw is an alternative input/output format that is similar to JSON with the following changes: • String values are accepted and returned without surrounding quotes. • Boolean values accept additional expressions such as `yes` and `no` in addition to `true` and `false`.

Note: The default input/output format for `atlantctl` is subject to change. If you rely on the format of `atlantctl`, specify either `--json` or `--raw` explicitly when `atlantctl` is run. In addition to setting values, the keys for array-type settings are considered as input and need to be entered in the correct format.

The `get` command accepts the following additional option:

--sensitive, -s	Show all values for settings, including any that may contain private information. If this option is not included in the command, values that are likely to contain private information are not shown.

The `set`, `add`, and `load` commands accept the following options:

--prompt, -p	Prompt for input using a text editor.
--file, -f [PATH]	Read the input value from a file. If the path is set to `-`, the input value is read from the standard input.

If you use the `--prompt` or `--file` options, do not enter a `[VALUE]` for the command.

Note: The `set` and `load` commands differ in how they handle settings that are not included in the input. The `set` command does not change the current values for any settings that are missing from the input, whereas the `load` command resets those settings to their default values.

5.1.2 Loading product settings from a file

You can use the `atlantctl load` command to restore settings from a previously saved JSON file.

This allows you to save your tested, functioning product configuration as a backup and then restore it if necessary. To save the current product configuration, you could use the following command, for example:

```
atlantctl get --json --sensitive > [PATH]
```

For this command, replace `[PATH]` with the full path and filename for the JSON backup file.

To restore the product settings, run the following command:

```
atlantctl load --json --file [PATH]
```

This applies all the settings that are available in the backup file specified as `[PATH]` to your product configuration. Any settings that are missing are reset to their default value.

5.1.3 Inspecting the state of the product

You can use the `atlantctl` command-line utility to inspect the state of the product and to view various scanning and performance statistics with the `status` sub-command.

Inspecting different status values works similarly to inspecting setting values:

- To get a status value, run `atlantctl status get [--json | --raw] STATUS...`
- To see the help for a status, run `atlantctl status help [--json | --raw] STATUS...`

scan_service	Type: JSON object All statistics related to the scanning service.
scan_service connection_count	Type: integer The number of connected hosts.
scan_service last_detection	Type: string The name of the last detection.
scan_service max_connection_count	Type: JSON object All statistics related to the maximum number of connected hosts.
scan_service max_connection_count daily	Type: integer The maximum number of connected hosts in the last 24 hours.
scan_service max_connection_count weekly	Type: integer The maximum number of connected hosts in the last week.
scan_service max_connection_count monthly	Type: integer The maximum number of connected hosts in the last month.
scan_service transaction_count	Type: JSON object All statistics related to the scanning transaction figures.
scan_service transaction_count total	Type: integer The total number of transactions.
scan_service transaction_count daily	Type: integer The number of transactions in the last 24 hours.
scan_service transaction_count weekly	Type: integer

	The number of transactions in the last week.
scan_service transaction_count monthly	Type: integer The number of transactions in the last month.
scan_service detection_count	Type: JSON object All statistics related to the detection figures.
scan_service detection_count total	Type: integer The number of detections.
scan_service detection_count daily	Type: integer The number of detections in the last 24 hours.
scan_service detection_count weekly	Type: integer The number of detections in the last week.
scan_service detection_count monthly	Type: integer The number of detections in the last month.
scan_service min_transaction_latency	Type: JSON object All statistics related to the minimum latency in scanning transactions.
scan_service min_transaction_latency daily	Type: integer The minimum transaction latency (milliseconds) in the last 24 hours.
scan_service min_transaction_latency weekly	Type: integer The minimum transaction latency (milliseconds) in the last week.
scan_service min_transaction_latency monthly	Type: integer The minimum transaction latency (milliseconds) in the last month.
scan_service max_transaction_latency	Type: JSON object All statistics related to the maximum latency in scanning transactions.
scan_service max_transaction_latency daily	Type: integer The maximum transaction latency (milliseconds) in the last 24 hours.
scan_service max_transaction_latency weekly	Type: integer The maximum transaction latency (milliseconds) in the last week.
scan_service max_transaction_latency monthly	Type: integer The maximum transaction latency (milliseconds) in the last month.
scan_service avg_transaction_latency	Type: JSON object All statistics related to the average latency in scanning transactions.
scan_service avg_transaction_latency daily	Type: integer The average transaction latency (milliseconds) in the last 24 hours.
scan_service avg_transaction_latency weekly	Type: integer The average transaction latency (milliseconds) in the last week.

scan_service avg_transaction_latency monthly
Type: integer
The average transaction latency (milliseconds) in the last month.

product_version
Type: JSON object
All information related to the product version.

product_version name
Type: string
The name of the pinned product version. This can be an empty string if no pinned product version has been specified.

Note: If you use a pinned version of the product, it works for one year from the date when the new pinnable product version is released. You should update the product to the new version during that time.

product_version status
Type: string
A string representing the status of the pinned product version. The value can be `unset`, `error`, `in-progress`, or `applied`.

5.1.4 Command-line settings for product license management

The settings related to product license management that are available for `atlantctl`.

Tip: To see an example of the structure for any of the settings, you can check the output for the following command: `/opt/f-secure/atlant/atlant/bin/atlantctl get [--json|--raw] [--sensitive] [SETTING]`

license
Type: JSON object
The license `type` and `content`. The `type` value must be one of `file` or `key`.
For `set` commands, the `content` value should be the path to the license file (if `type` is `file`) or the code in data URL form (if `type` is `key`).
Examples:

```
atlantctl set license '{"type": "file", "content": "path_to_local_license_file"}'
```

```
atlantctl set license '{"type": "key", "content": "data:,aaa-bbb-ccc"}'
```

The response for `get` commands includes the product license content in data URL form.

Note: Unlike most of the `atlantctl` commands, there are no separate command options for the individual components of the `license` setting objects.

5.1.5 Command-line settings for client management

The settings related to client management that are available for `atlantctl`.

Note: The command path given here is for the current version of the product. If you are using an older version, use `/opt/f-secure/atlant/bin/<command>` as the path instead.

Note: The `client` command uses a slightly different syntax than the other `atlantctl` commands.

atlantctl client create [SCOPES]
Type: JSON object
Creates a new client with the specified scopes. The available scopes are:

- `management` - allows the client to inspect and change the Atlant configuration
- `scan` - allows the client to scan files

The response is a JSON object containing the `client_id` and `client_secret` values, which are the credentials that the client uses to request access tokens from the authorization server.

Example:

```
/opt/f-secure/atlant/atlant/bin/atlantctl client create '{"scopes":
["management", "scan"]}'
```

Example response:

```
{
  "client_id": "5dbfe97de42bf53a0ae73bff9eba4ecb",
  "client_secret":
"87e7b3a61e7fdcdfc03162fdcab82a942b9c62715ff3d612e15adf32d1b1500b"
}
```

atlantctl client list

Type: JSON object

Returns a list of the clients and the scopes for each of them.

atlantctl client delete [CLIENT_ID]

Type: JSON object

Delete a client. The request must specify the `client_id` value for the client that you want to remove.

Example:

```
/opt/f-secure/atlant/atlant/bin/atlantctl client delete
5dbfe97de42bf53a0ae73bff9eba4ecb
```

5.1.6 Command-line settings for authorization server

The settings related to authorization that are available for `atlantctl`.

Tip: To see an example of the structure for any of the settings, you can check the output for the following command: `/opt/f-secure/atlant/atlant/bin/atlantctl get [--json|--raw] [--sensitive] [SETTING]`

authorization

Type: JSON object

All authorization server settings.

authorization external_domains

Type: JSON array

The list of external authorization domains.

authorization external_domains [AUDIENCE]

Type: JSON object

All information for the external authorization domain specified in `[AUDIENCE]`.

authorization external_domains [AUDIENCE] audience

Type: string

The audience for the external authorization domain specified in `[AUDIENCE]`.

authorization external_domains [AUDIENCE] issuer

Type: string

The issuer for the external authorization domain specified in `[AUDIENCE]`.

authorization external_domains [AUDIENCE] signing_method

Type: string

The signing method for the external authorization domain specified in `[AUDIENCE]`. The value must be either `HS256` or `RS256`.

authorization external_domains [AUDIENCE] verification_key

Type: string

The verification key for the external authorization domain specified in `[AUDIENCE]`.

authorization https_endpoints

Type: JSON array

The list of authentication server HTTP endpoints.

authorization https_endpoints [ENDPOINT]
Type: JSON object
All information for the HTTP endpoint specified in `[ENDPOINT]`.

authorization https_endpoints [ENDPOINT] address
Type: string
The address for the HTTP endpoint specified in `[ENDPOINT]`.

authorization https_endpoints [ENDPOINT] port
Type: integer
The port for the HTTP endpoint specified in `[ENDPOINT]`.

5.1.7 Command-line settings for TLS

The settings related to TLS that are available for `atlantctl`.

Note: The command path given here is for the current version of the product. If you are using an older version, use `/opt/f-secure/atlant/bin/<command>` as the path instead.

Tip: To see an example of the structure for any of the settings, you can check the output for the following command: `/opt/f-secure/atlant/atlant/bin/atlantctl get [--json|--raw] [--sensitive] [SETTING]`

tls
Type: JSON object
All TLS-related settings used in Atlant.

tls certificate
Type: string
The TLS certificate (public key) used in Atlant.
For `set` commands, enter either the local file path to the certificate or the base64-encoded data URL for the certificate contents.
The response for `get` commands includes the base64-encoded data URL with the contents of the certificate.
Examples:

```
/opt/f-secure/atlant/atlant/bin/atlantctl set tls certificate
<local_path_to_certificate>
```

```
/opt/f-secure/atlant/atlant/bin/atlantctl set tls certificate
'data;base64,Y2VydGlmaWNhdGU='
```

tls key
Type: string
The TLS private key used in Atlant.
For `set` commands, enter either the local file path to the key or the base64-encoded data URL for the key contents.
The response for `get` commands includes the base64-encoded data URL with the contents of the key.
Example:

```
/opt/f-secure/atlant/atlant/bin/atlantctl set tls key <local_path_to_private_key>
```

5.1.8 Command-line settings for management server

The settings related to management server setup that are available for `atlantctl`.

Tip: To see an example of the structure for any of the settings, you can check the output for the following command: `/opt/f-secure/atlant/atlant/bin/atlantctl get [--json|--raw] [--sensitive] [SETTING]`

management
Type: JSON object
All management server settings.

management https_endpoints	Type: JSON array The list of management server endpoints.
management https_endpoints [ENDPOINT]	Type: JSON object All information for the management server endpoint specified in `[ENDPOINT]`.
management https_endpoints [ENDPOINT] address	Type: string The address for the management server endpoint specified in `[ENDPOINT]`.
management https_endpoints [ENDPOINT] port	Type: integer The port for the management server endpoint specified in `[ENDPOINT]`.

5.1.9 HTTP proxy settings

The settings related to HTTP proxy usage that are available for `atlantctl`.

Tip: To see an example of the structure for any of the settings, you can check the output for the following command: `/opt/f-secure/atlant/atlant/bin/atlantctl get [--json|--raw] [--sensitive] [SETTING]`

http_proxy	Type: JSON object All HTTP proxy settings.
http_proxy enabled	Type: boolean Controls whether or not to use an HTTP proxy for features that require network access.
http_proxy host	Type: string The host name of the HTTP proxy.
http_proxy password	Type: string The password for basic HTTP proxy authentication.
http_proxy port	Type: integer The HTTP proxy port. The value must be a number between `1` and `65535`.
http_proxy username	Type: string The user name for basic HTTP proxy authentication. **Note:** If this setting is empty, no authentication is used.

5.1.10 Command-line settings for scanning server

The settings related to the scanning server that are available for `atlantctl`.

Tip: To see an example of the structure for any of the settings, you can check the output for the following command: `/opt/f-secure/atlant/atlant/bin/atlantctl get [--json|--raw] [--sensitive] [SETTING]`

scanning	Type: JSON object Contains all scanning server settings.
scanning https_endpoints	Type: JSON array List of HTTP scanning endpoints.
scanning https_endpoints [ENDPOINT]	Type: JSON object Settings for the specified HTTP endpoint.

scanning https_endpoints [ENDPOINT] address
Type: string
The endpoint address.

scanning https_endpoints [ENDPOINT] port
Type: integer
The endpoint port.

scanning icap
Type: JSON object
Settings for ICAP scanning API.

scanning icap_endpoints
Type: JSON array
List of ICAP scanning endpoints.

scanning icap_endpoints [ENDPOINT]
Type: JSON object
Settings for the specified ICAP endpoint.

scanning icap_endpoints [ENDPOINT] address
Type: string
The endpoint address.

scanning icap_endpoints [ENDPOINT] port
Type: integer
The endpoint port.

scanning icap scan_archives
Type: boolean
Enable scanning of archived files.

scanning icap trickle
Type: JSON object
ICAP trickling settings.

scanning icap trickle enabled
Type: boolean
Enable ICAP response trickling.

When enabled, Atlant sends ICAP and HTTP headers immediately, then trickles body data one byte at a time. After scanning completes, remaining data is sent or the connection is closed if infection is detected.

Important: Trickling can expose clients to harmful content because some data is sent before scanning finishes. Truncating infected content does not make it safe. Use trickling only if this trade-off is acceptable and ensure endpoints have additional protection.

scanning icap trickle interval
Type: integer
Interval (in seconds) between trickled bytes.

scanning icap trickle max_bytes
Type: integer
Maximum bytes to trickle before scan results are available.

scanning icap trickle start_delay
Type: integer
Delay (in seconds) before trickling starts. `0` (default) starts immediately.

scanning icap block_archive_max_nested
Type: boolean
Block archives with excessive nesting.

scanning icap block_encrypted_archives
Type: boolean
Block encrypted archives.

scanning icap detect_pua
Type: boolean
Block potentially unwanted applications (PUAs).

scanning icaps_endpoints
Type: JSON array
List of ICAPS (ICAP over TLS) endpoints.

scanning icaps_endpoints [ENDPOINT]
Type: JSON object
Settings for the specified ICAPS endpoint.

scanning icaps_endpoints [ENDPOINT] address	Type: string The endpoint address.
scanning icaps_endpoints [ENDPOINT] port	Type: integer The endpoint port.
scanning logging	Type: JSON object Logging settings.
scanning logging ip_address	Type: boolean Log client IP addresses.
scanning logging https	Type: JSON object Logging settings for REST scanning API.
scanning logging https request_headers	Type: array of strings HTTP request headers to log.
scanning logging icap	Type: JSON object Logging settings for ICAP scanning API.
scanning logging icap request_headers	Type: array of strings List of ICAP request headers to log.
scanning logging icap encapsulated_request_headers	Type: array of strings Encapsulated request headers to log.
scanning logging icap encapsulated_response_headers	Type: array of strings Encapsulated response headers to log.
scanning logging rotation	Type: JSON object Log rotation settings for `access.log`.
scanning logging rotation file_size	Type: JSON object Rotate based on file size.
scanning logging rotation file_size enabled	Type: boolean Enable rotation when file exceeds size limit. If disabled, `access.log` grows indefinitely.
scanning logging rotation file_size limit	Type: integer Size threshold (in bytes).
scanning logging rotation max_bytes	Type: JSON object Limit total size of rotated logs.
scanning logging rotation max_bytes enabled	Type: boolean Enable cumulative size limit.
scanning logging rotation max_bytes limit	Type: integer Maximum cumulative size (in bytes).
scanning logging rotation max_days	Type: JSON object Limit retention by age.
scanning logging rotation max_days enabled	Type: boolean Enable retention limit.
scanning logging rotation max_days limit	Type: integer Maximum number of days to keep logs.
scanning logging rotation max_files	Type: JSON object Limit number of rotated files.

scanning logging rotation max_files enabled	Type: boolean Enable file count limit.
scanning logging rotation max_files limit	Type: integer Maximum number of rotated files.
scanning keepalive_time	Type: integer Seconds to wait for client to complete request before returning an error and closing the connection.
scanning max_scan_time	Type: integer Maximum scan duration (in seconds). If exceeded, partial results are returned.
scanning max_scan_size	Type: JSON object Limit scanned file size.
scanning max_scan_size enabled	Type: boolean Enable size limit.
scanning max_scan_size limit	Type: integer Maximum file size (in bytes).
scanning archive_max_nested	Type: integer Maximum depth for nested archives.
scanning forbidden_uri_categories	Type: JSON object Detect URLs matching specified categories.
scanning forbidden_uri_categories [CATEGORY]	Type: boolean Enable detection for the given category. [CATEGORY] can be any of the predefined categories from the URL categories.

5.1.11 Command-line settings for online services

The settings related to online services that are available for `atlantctl`.

Tip: To see an example of the structure for any of the settings, you can check the output for the following command: `/opt/f-secure/atlant/atlant/bin/atlantctl get [--json|--raw] [--sensitive] [SETTING]`

online_services	Type: JSON object All online services settings.
online_services enabled	Type: boolean Enables or disables online services. • When `true`, features that need internet access are available. • When `false`, these features are turned off. The product can then run in environments with little or no internet access.

5.1.12 Command-line settings for automatic updates

The settings related to automatic updates that are available for `atlantctl`.

Tip: To see an example of the structure for any of the settings, you can check the output for the following command: `/opt/f-secure/atlant/atlant/bin/atlantctl get [--json|--raw] [--sensitive] [SETTING]`

updates	Type: JSON object All settings related to automatic updates.

updates enabled	Type: boolean Controls whether automatic updates are switched on or off.
updates product_version	Type: string If not empty, this sets a specific product version to use instead of automatically updating to the latest available version.
updates schedule	Type: JSON object The automatic update schedule.
updates schedule regime	Type: string The type of scheduling for automatic updates. The allowed values are: • `on_arrival` - updates are applied as soon as they become available • `at_date` - updates are postponed until the time specified in the `at_date_schedule` setting • `with_repetition` - updates are applied regularly according to the `repetition_schedule` settings
updates schedule at_date_schedule	Type: integer The time to apply updates when `regime` is set to `at_date`. The value must be a timestamp in epoch time format.
updates schedule repetition_schedule	Type: JSON object Settings related to automatic updates with a fixed repetition schedule.
updates schedule repetition_schedule day	Type: string The scheduled day for repeated updates. Allowed values are `daily`, to check for updates every day at the specific time, or `monday`, `tuesday`, `wednesday`, `thursday`, `friday`, `saturday`, or `sunday` to check for updates on the given day each week.
updates schedule repetition_schedule time_of_day	Type: string The scheduled time of day for applying updates. The value is a 24-hour UTC time value (`HH:MM`).
updates use_tls	Type: boolean Controls whether Transport Layer Security (TLS) is used when downloading updates.

5.2 Configuring Atlant settings with WithSecure Policy Manager

WithSecure Policy Manager can be used to configure and change settings in Atlant.

In the Policy Manager Console, go to **Root** > **Settings** > **Atlant** to view the options.

For more information about WithSecure Policy Manager, see the latest Policy Manager Admin guide.

5.2.1 Setting up HTTPS connections in Atlant with Policy Manager

All communication with Atlant's REST API use HTTP over Transport Layer Security (TLS).

You can use Policy Manager to set up secure HTTPS connections with Atlant.

To set up HTTPS connections, this requires a TLS certificate and private key, which can be added in Policy Manager.

Adding a TLS certificate and private key for HTTP endpoints in Policy Manager

You can use Policy Manager to add a TLS certificate and private key.

You can add a TLS certificate and private key in the same view.

In the standard view of the Policy Manager Console:

1. Select the target domain.
2. Go to the **Settings** > **Atlant** > **Centralized management** page.
3. Under **Default TLS settings**, select one of the following:
   - To add a TLS certificate, next to **TLS certificate for HTTP endpoints**, select **Choose file**, and then select the certificate to be used.
   - To add a TLS private key, next to **TLS private key for HTTP endpoints**, select **Choose file**, and then select the key to be used.

   Both the TLS certificate and private key are added for access to Atlant's REST API.
4. Click the following icon to distribute the policy: 

5.2.2 Setting up HTTP endpoints with Policy Manager

In Policy Manager, it is possible to list the HTTP endpoints that an Atlant instance listens to.

The three HTTP endpoints are as follows:

- **Management HTTP Endpoints** can be used to configure the product and manage per-instance users.
- **Scanning HTTP Endpoints** can be used to scan content.
- **Authorization HTTP Endpoints** can be used to authenticate with an Atlant instance.

Each endpoint is a combination of address and port.

Adding HTTP endpoints for Atlant in Policy Manager

This topic describes how to add HTTP endpoints for an Atlant instance in Policy Manager.

To set up HTTP endpoints for Atlant's services, in the standard view of the Policy Manager Console, do the following:

1. Select the target domain.
2. Go to the **Settings** > **Atlant** > **Centralized management** page.
3. Select the respective service:
   - **Management HTTP Endpoints**
   - **Scanning HTTP Endpoints**
   - **Authorization HTTP Endpoints**

   a) Select the **Address** tab, then select **Add**.
   b) Add the address in the field.
   c) Select the **Port** tab, then select **Add**.
   d) Add the port.

   The endpoints are added to the Policy Manager.
4. Click the following icon to distribute the policy: 

5.2.3 Setting up external authorization domains for Atlant with Policy Manager

This topic explains about setting up external authorization domains for Atlant with Policy Manager.

As an alternative, Atlant can be configured to rely on third-party external authorization services that can be used to authenticate with Atlant. In Policy Manager, this option can be beneficial if there are multiple Atlant instances managed by Policy Manager.

To authenticate external authorization services, Atlant needs the following details:

- The type of cryptography that the authorization server uses when creating the tokens, either RS256 or HS256.

  If RS256 is used, Atlant needs to have the server's public key in PEM format.

  If HS256 is used, Atlant needs to know the shared secret.

- The issuer
- The audience

Adding external authorization domains

This topic explains how to add external authorization domains for Atlant with Policy Manager.

To add the necessary details, in the standard view of the Policy Manager Console, do the following:

1. Select the target domain.
2. Go to the **Settings** > **Atlant** > **Centralized management** page.
3. Add the details under **External authorization domains**.
   a) Select the **Audience** tab to add the audience information, then select **Add**.
   b) Select the **Issuer** tab to add the issuer information, then select **Add**.
   c) Select the **Signing method** tab to add the signing method (RS256 or HS256), then select **Add**.
   d) Select the **Verification key**tab to add the verification key, then select **Add**.
4. Click the following icon to distribute the policy: 

5.2.4 Configuring the list of content categories for URL scanning

In Policy Manager, it is possible to configure the list of content categories for Atlant. Based on the list of disallowed content categories, Atlant determines if a URL should be blocked or not.

To select the categories, in the standard view of the Policy Manager Console, do the following:

1. Select the target domain.
2. Go to the **Settings** > **Atlant** > **Centralized management** page.
3. Under **Disallowed site categories**, in the **Site category** column select which category to include, then select the corresponding box in the **Disallowed** column.
4. Select **Add**.
   The content category has successfully been added.
5. Click the following icon to distribute the policy: 

5.2.5 Client access management with Policy Manager to the Atlant REST API

Policy Manager can be used to manage client access to Atlant's REST API.

Clients need to authenticate with Atlant's internal authorization server before getting access to Atlant's management and scanning services.

To authenticate, clients need to provide Atlant with a client ID and a client secret. If both credentials are valid, Atlant issues a short-lived authentication token, which can be used for subsequent requests.

Adding client credentials in Policy Manager

This topic describes how to add client IDs and client secrets to Policy Manager in order to access Atlant's REST API.

To add client IDs and client secrets to Policy Manager, in the standard view of the Policy Manager Console, do the following:

1. Select the target domain.
2. Go to the **Settings** > **Atlant** > **Centralized management** page.
3. Under **Remote domain clients**, do the following:
   - Select the **Client ID** tab, add the client ID, then select **Add**.
   - Select the **Scope identifiers** tab, add the client secret, then select **Add**.

   The client credentials have been added to Policy Manager Console.
4. Click the following icon to distribute the policy: 

5.2.6 Central management for Atlant in Policy Manager

This section describes about the central management options available in Policy Manager for managing Atlant.

In Policy Manager, there are certain settings that can be managed centrally as follows:

- Internet connections: these are settings that manage how the product connects to the internet; for example, the use of a proxy.
- Automatic updates: these settings manage if and when the product should automatically install updates.
- Product security: this setting manages changes to the local admin settings.

Configuring manual scanning settings for Atlant in Policy Manager

In Policy Manager, it is possible to configure settings for Atlant's manual scanning.

To configure the different settings for manual scanning, in the Standard view in the Policy Manager Console, do the following:

1. Select the target domain.
2. Go to the **Settings** > **Atlant** > **Centralized management** page.
3. Under **Manual Scanning**, to turn on a setting, select the box next to the setting:
   - **Scan for potentially unwanted applications**
   - **Scan for spam**
4. Under **Handling archives in manual scanning**, to turn on a setting, select the box next to the setting:
   - **Scan inside archives**
   - **Treat encrypted archives as unsafe**. Once selected, next to **Scan archive up to nesting level** , set the nesting level in the dropdown.
   - **Treat archives that exceed the maximum nesting level as unsafe**

5. Under **Actions for manual scanning**, set the type and the action for the setting:
   - **Action for malware**
     - **Rename**: Rename the infected file. The renamed file has a `.malware` or `.suspected` extension depending on the type of detection.
     - **Delete**: Remove the infected file.
     - **Do nothing**: Do not perform any action on the infected file.
   - **Action for potentially unwanted applications**
     - **Rename**: Rename the infected file. The renamed file has a `.malware` or `.suspected` extension depending on the type of detection.
     - **Delete**: Remove the infected file.
     - **Do nothing**: Do not perform any action on the infected file.
   - **Actions for suspicious files**
     - **Rename**: Rename the infected file. The renamed file has a `.malware` or `.suspected` extension depending on the type of detection.
     - **Delete**: Remove the infected file.
     - **Do nothing**: Do not perform any action on the infected file.
   - **Actions for spam**
     - **Rename**: Rename the infected file. The renamed file has a `.malware` or `.suspected` extension depending on the type of detection.
     - **Delete**: Remove the infected file.
     - **Do nothing**: Do not perform any action on the infected file.
6. Click the following icon to distribute the policy: 

Configuring automatic update options for Atlant in Policy Manager

This topic explains how to configure automatic updates in Policy Manager for Atlant.

WithSecure uses dedicated service connections between the backend systems and installed products, referred to as channels, to deliver updates to the malware definition databases, scanning engines, and the actual products. Each channel provides updates for a specific product component or function.

The settings for automatic updates apply to the three WithSecure product channels (`atlant`, `fsbg`, and `baseguard`) that the product uses. The automatic update settings do not affect updates for other channels, such as the scanning engines, as they are applied immediately when they are available. The exception to this is the main feature switch for automatic updates: turning automatic updates off disables the updates for all channels.

In addition, product channel updates are applied one week after the update has been published at the latest.

To configure automatic updates, in the Standard view of the Policy Manager Console, do the following:

1. Select the target domain.
2. Go to the **Settings** > **Atlant** > **Centralized management** page.
3. Make sure that **Enable automated product and security updates** is selected.
4. Configure the scheduling for the automatic product updates:
   - **On arrival**: This setting is the default, and product and virus definition updates are applied as soon as they become available. When this is selected, **Date**, **Time**, and **Day** are ignored.
   - **Once**: Updates are postponed until the specified **Date** and **Time**. When this is selected, **Day** is ignored. The time is given in Policy Manager's local timezone.
   - **Daily**: Updates are applied at the specified **Time**. When this is selected, **Date** and **Day** are ignored.

- **Weekly**: Updates are applied at the specified **Time** on the specified **Day**. When this is selected, **Date** is ignored.

Note: Each update has an associated expiry time. Once the expiry time is reached, the update is applied regardless of the date and time setting. This means that users do not have to worry about late updates jeopardizing security.

5. If you want to keep using a specific version of the product, select **Use a specific product version instead of the latest version available** and enter the version that you want to use in the **Product version** field.

   Normally, all hosts are automatically updated to the latest available version, which means that there can be some changes to the product features without explicit notification. If you want to use a specific version, check the release notes for that version to find the text to enter here.

6. Select **Send alerts for failed updates** to generate product alerts for any failed updates.

7. Click the following icon to distribute the policy: 

Granting permission to change settings locally

This topic explains how to give the local administrator permission to change settings locally.

If the local administrator is given rights to change settings locally, this overrides the settings specified in Policy Manager.

To change this setting, in the Standard view in the Policy Manager Console, do the following:

1. Select the target domain.
2. Go to the **Settings** > **Atlant** > **Centralized management** page.
3. Under **Bypassing product security**, next to **Allow user to unload products**, select **Not allowed** or **Allowed** in the drop-down menu.

4. Click the following icon to distribute the policy: 

5.2.7 Setting up the Atlant scanning service for Virtual Security in Policy Manager

This topic explains how to set up the Atlant scanning service in Policy Manager for Virtual Security.

In the Policy Manager Console, it is possible to add the Atlant subscription key and configure general scanning settings.

To set up the Atlant scanning service for Virtual security, in the Standard view in the Policy Manager Console, do the following:

1. Select the target domain.
2. Go to the **Settings** > **Virtual Security** > **Atlant scanning service** page.
3. Under **Atlant scanning service** > **Atlant subscription**, add the subscription key in the boxes provided.

   Note: Adding a subscription key is only needed to enable the advanced features not covered by a Business Suite license.

4. Under **General**, select the boxes for **Check file reputation** and **Check URL reputation**, then configure the following settings:

   - Timeout for reputation requests
   - Maximum scanning time
   - Keep client connections alive
   - Scan archives up to nesting level

5. Under **Endpoints**:

- Select the **Address** tab, add the address, then select **Add**.
- Select the **Port** tab, add the port number, then select **Add**.

6. Click the following icon to distribute the policy: 

Chapter
6

Command line usage

Topics:

This section describes the commands that you can run from the command line in the product.

6.1 Scanning the computer manually from the command line

You can scan the computer for malware manually from the command line of a Linux host that has the product installed.

Note: The command path given here is for the current version of the product. If you are using an older version, use `/opt/f-secure/atlant/bin/<command>` as the path instead.

The `fsanalyze` program scans files with the privileges of the user who runs it. The user must have read access to files, and read and execute access to all directories to be scanned. To rename or remove infected files, the user must have write access to the directories where the files are located.

The `fsanalyze` command scans specified targets (files or directories) and reports any malicious code it detects.

If no files are specified on the command line, `fsanalyze` reads the standard input for content to analyze. Otherwise, every given file is analyzed and every given directory is scanned recursively, with the following exceptions:

- Special files (such as socket, FIFO, or device files), and files on the `/proc` and `/sys` special file systems are skipped.
- Recursive scanning does not follow symbolic links unless this is explicitly requested with the `--follow` command line option.
- Recursive scanning does not automatically cross file system boundaries. To scan files mounted on other file systems, specify the mount points explicitly on the command line.

Run the following command from the shell to start the manual scan:
`/opt/f-secure/atlant/atlant/bin/fsanalyze [OPTIONS] [FILE...]`

Usage: `fsanalyze [OPTIONS] [FILE...]`

Analyze each `FILE` for potential malicious content. If no file is specified, analyze standard input, in which case the only supported action is `none`.

-h, --help	Show help.
--malware=ACTION	Action to take when a file is detected as malware. If ACTION is `remove` the file is removed; if ACTION is `rename` the file is renamed adding the detection type as an extension; if ACTION is `none`, the infection is only reported on standard output. Default: `rename`
--pua=ACTION	Action to take when a file is detected as a potentially unwanted application. Options for the action are the same as for malware. Default: `none`
--suspected=ACTION	Action to take when a file is detected as suspicious. Options for the action are the same as for malware. Default: `none`
-L, --follow	Follow symbolic links.
-l, --list	List all scanned files.
--preserve-atime=VALUE	Preserve the access time of files that are scanned. Value can be `yes` or `no`. Default: `no`
-q, --quiet	Only output the scan results.
--scan-archives=VALUE	Enable archive scanning. Value can be `yes` or `no`. Default: `no`
--max-nested=NUMBER	Set the maximum allowed nesting level of scanned archives. Default: 5

--detect-max-nested=VALUE	If enabled, treat archives that exceed the maximum depth as malware. Value can be `yes` or `no`. Default: `no`
--detect-encrypted-archives=VALUE	Treat encrypted archives as malware. Value can be `yes` or `no`. Default: `no`
--detect-pua=VALUE	If disabled, potentially unwanted applications are ignored. Value can be `yes` or `no`. Default: `yes`
--use-orsp=VALUE	If enabled, use the reputation service (ORSP) from Security Cloud. Value can be `yes` or `no`. Default: `yes`
-v, --version	Display program version and exit.
--quoting-style=STYLE	Set how to quote the output. If STYLE is `url`, the output is URL-encoded; if STYLE is `escape`, ASCII control characters in the output are escaped as `<NAME>`, and if the output is not redirected to a file, they are also highlighted; if STYLE is `literal`, the output is printed as is. Default: `url`
--spam=ACTION	Action to take when a file is detected as spam or phishing content. Options for the action are the same as for malware. Default: `none`
--detect-spam=VALUE	If enabled, detect spam and phishing content. Value can be `yes` or `no`. Default: `no`
--ip=ADDRESS	The IP address of the content source or originating client.
--sender=EMAIL	The email address of the sender of the content.
--recipient=EMAIL	The email address of the recipient of the content. This option can appear multiple times.

The scanning exit codes are as follows:

- `0` = scanning was completed with no errors, and no malware, PUA, or suspicious content was detected
- `1` = scanning failed for one or more files
- `2` = scanning was completed with no errors, but malware, PUA, or suspicious content was detected

Scanning detections are printed for each file on the standard output as separate lines in the following format:

```
<FILE>: result=<RESULT> infection=<DETECTION NAME> member-name=<MEMBER>
```

where:

- `<FILE>` is the path name of the scanned file.
- `<RESULT>` is one of `clean`, `infected`, `pua`, or `suspected`. Clean files are reported in the output only if the `--list` option is specified on the command line.
- `<DETECTION NAME>` is the name of the infection that was detected in the file. If the file is clean, the `infection` field is omitted.
- `<MEMBER>` is the name of the file scanned in an archive (if `<FILE>` is an archive file). If `<FILE>` is not an archive, the `member-name` field is omitted.

Multiple detections on the same file are printed on separate lines of output.

Note: Nonprintable or non-ASCII characters in all file names shown in the output are URL-encoded by default. For example, the space character is represented as a plus sign (+) and the ÃƒÂ© character is

represented as %C3%A9. You can change the output format for path names with the `--quoting-style` option.

Unless the action assigned to the type of detection is `none`, the detection information is followed by a line in the following form:

```
<FILE>: action=<ACTION>
```

where:

- <FILE> is the name of the file that was scanned
- <ACTION> is either `deleted` or `renamed`

Sample output for detections:

```
archive.tgz: result=infected infection=EICAR_Test_File member-name=eicar.txt
archive.tgz: action=renamed
cloudcar.exe: result=infected infection=Malware:W32/Cloudcartestfile
cloudcar.exe: action=renamed
```

If the `--quiet` option was specified on the command line, `fsanalyze` does not produce any further output. Otherwise, the list of detections is followed by a list of engines used for scanning and a summary of scan results in the following form:

```
Engine versions: F-Secure Corporation Aquarius/18.0.676/2020-04-14_03 F-Secure Corporation
Hydra/6.0.216/2020-04-13_01 F-Secure Corporation FMLib/17.0.607.475  (cf1875a)/2020-02-04_01
fsicapd/2.0.84

15 files scanned
7 files infected
3 files contain PUA
1 files suspected
2 files could not be scanned
```

The summary always includes the number of input files that were scanned successfully. If there were no infected files, files with PUA, suspected files, or files that could not be scanned, the corresponding lines are omitted. Archive files count as single input files.

Scanning errors are reported on the standard error using messages in the following form:

```
fsanalyze: <MESSAGE> '<FILE>' (<DETAILS>)
```

where:

- <MESSAGE> identifies the operation that failed
- <FILE> is the name of the file for which the operation failed
- <DETAILS> gives the operating system-level details about the error

Note: The `fsanalyze` program fills the same purpose as the `fsav` program included in WithSecure Linux Security 11. The main differences compared to the output of `fsav` are that `fsanalyze` shows path names with nonprintable or non-ASCII characters as URL-encoded by default, does not display the name of the scanning engine separately for every detection, and does not display time stamps for the scanning process.

6.2 Starting and stopping services

The product contains a number of background services, which you can control using the `/opt/f-secure/atlant/fsbg/bin/master-switch` command.

Note: The command path given here is for the current version of the product. If you are using an older version, use `/opt/f-secure/fsbg/bin/<command>` as the path instead.

You can use the `master-switch` command to control all the services related to the product. The syntax for the command is:

```
/opt/f-secure/atlant/fsbg/bin/master-switch ( on | off | status | enable | disable | is-enabled
 )
```

Note: If you are managing your Atlant installation with WithSecure Policy Manager, you can choose whether or not the `master-switch` command can be used for turning off WithSecure services in Policy Manager Console. Change this using the **Windows** > **Centralized management** > **Allow users to unload products** setting.

Use the `off` sub-command to temporarily stop all WithSecure services:

```
/opt/f-secure/atlant/fsbg/bin/master-switch off
```

The `on` sub-command restarts the stopped services:

```
/opt/f-secure/atlant/fsbg/bin/master-switch on
```

To view the status of WithSecure services, use the `status` sub-command:

```
/opt/f-secure/atlant/fsbg/bin/master-switch status
```

The `disable` sub-command turns off all WithSecure services. The services stay turned off after you restart the computer. The `enable` sub-command turns on all WithSecure services again. Use the `is-enabled` sub-command to check if WithSecure services are turned on.

6.3 Updating the product manually from the command line

You can update the product, the malware definition databases, and the scanning engines manually to their latest versions using the `atlantctl` program.

Note: The instructions in this topic do not apply to isolated product installations.

You can update the product from the command line by running the following command as the root user:

```
/opt/f-secure/atlant/atlant/bin/atlantctl update
```

This command checks for the availability of new updates over the network, and downloads and installs them on the system to make sure that everything is up to date.

The `--timeout` option can be used to specify the period of time to wait, in seconds, for the update to complete before letting it finish in the background. The value `0`, which is used as the default if the option is not specified, disables the timeout.

Note: There are a few things to consider before and during the update:

- Before running the command, make sure that the product services are running.
- Once the update has started, downloading and installing the updates cannot be canceled. When using the `--timeout` option, the operation is left to complete in the background after the timeout.
- If a specific product version has been set in the configuration, the product is not updated past that version. Only the malware definitions and the scanning engine updates are installed.

Related concepts

Starting and stopping services on page 248
The product contains a number of background services, which you can control using the `/opt/f-secure/atlant/fsbg/bin/master-switch` command.

Related tasks

Updating the product manually using an update archive on page 15
In isolated environments with limited network connectivity, you can update the product and virus definition databases by creating update archives and deploying them on the hosts.

Chapter
7

Using Atlant as a replacement for WithSecure Scanning and Reputation Server

Topics:

You can use Atlant to provide scanning and content reputation services to WithSecure client software installed in virtual environments.

You can use Atlant to minimize the impact on the performance of virtual desktops and servers. This approach offloads the malware scanning and content reputation checking to a dedicated server that runs Atlant, instead of performing the requests directly on each virtual machine.

To deploy Atlant to a Linux host for use in this role:

1. Create the installation package in WithSecure Policy Manager.
2. Install the package on the target Linux host.

After deployment, you then need to configure the settings in Policy Manager Console.

Related concepts

Using Atlant as a virtual appliance

7.1 Configuring Atlant for virtual environments

To set up Atlant for use as a replacement for Scanning and Reputation Server, you need to configure it in Policy Manager Console to handle the offloaded requests from virtual desktops and servers.

After you have created the installation package in Policy Manager and installed Atlant:

1. In the Standard view of Policy Manager Console, select the **Root** domain.
2. Go to the **Settings** tab and select **Virtual security** > **Offload Scanning Agent**.
3. Select **Offload file scanning** and enter the address for your Atlant host.
4. Select **Virtual security** > **Atlant scanning service**.
5. Enter your Atlant for Offload Scanning subscription key.
6. Configure the **General** settings and timeout values as needed.
7. Under **Endpoints**:
   - Click **Add** if you want add more ICAP service endpoints.
   - Select any of the entries and click **Edit** if you need to change the address or port for the endpoint.

   By default, Policy Manager creates a single ICAP scanning endpoint that uses the address `0.0.0.0` and port `1344`.
8. Click the following icon to distribute the policy: 

Appendix A

Services installed with Atlant

Topics:

WithSecure Atlant installs several services that are used to handle various product features.

Note: The services listed here are subject to change without advance notice.

Important: Do not manually start or stop any of the services listed here.

A.1 Active services

These services are loaded by the product.

f-secure-atlant-atlant-atlantpmd.service	Locally distributes centrally managed settings to Atlant services.
f-secure-atlant-atlant-webserver.service	Provides an HTTP API for centrally managed settings.
f-secure-atlant-baseguard-as.service	A BaseGuard facility for email spam scanning.
f-secure-atlant-baseguard-authorize.service	An OAuth 2 authorization server used with HTTP APIs. This service is inactive unless you have configured authorization endpoints.
f-secure-atlant-baseguard-cleanup.service	Makes sure that older channel updates are cleaned up to save disk space.
f-secure-atlant-baseguard-doormand.service	Facilitates handling subscriptions, registration and cloud service lookups.
f-secure-atlant-baseguard-icap.service	The malware analysis service used for realtime, scheduled and manual scanning.
f-secure-atlant-baseguard-telemetry.service	Collects information for monitoring and analysis to keep the product features working as they should.
f-secure-atlant-baseguard-tokenverify.service	An OAuth 2 authentication manager used with HTTP APIs.
fsbg-atlant-updated.service	Schedules the installation of online channel updates.
fsbg-atlant-statusd.service	Collects status and statistics information from BaseGuard services and relays them to the central management agent (`fsma2`).
fsbg-atlant-pmd.service	Locally distributes centrally managed settings to BaseGuard services.
atlant-fsma2.service	Handles centrally managed settings and communication.
fsbg-hotfixctl.service	WithSecure hotfix service that checks if there are hotfixes available. Usually in an inactive state.
fsbg-hotfixctl.timer	WithSecure hotfix service timer. Runs fsbg-hotfixctl.service to check hotfixes once per hour.

A.2 Inactive services

These services are included during installation, but are not loaded by the product.

f-secure-atlant-baseguard-accd.service	Real-time scanner service that handles file access notifications.
f-secure-atlant-baseguard-sensor.service	Handles the functionality of WithSecure Elements Endpoint Detection and Response. If the Endpoint Detection and Response feature is enabled for the subscription key in use, this service is active, but it is inactive by default.

Appendix B

ICAP extension reference

Topics:

The ICAP service in Atlant extends the standard protocol with a number of extensions.

B.1 URI options

The REQMOD and RESPMOD requests can include the following URI options.

Option	Values	Default	Description
`allow_upstream_metadata`	`[01]`	`1`	Allow logging and upstreaming of metadata.
`allow_upstream_application_files`	`[01]`	`1`	Allow upstreaming of application files.
`allow_upstream_data_files`	`[01]`	`0`	Allow upstreaming of data files.
`antispam`	`[01]`	`0`	Scan content for spam.
`scan_embedded_urls`	`[01]`	`0`	Scan URLs embedded in the content.
`security_cloud`	`[01]`	`0`	1 to allow call to Security Cloud to process request.

B.2 Request headers

For REQMOD and RESPMOD requests, the client can use the following extension headers.

X-Forbidden-URI-Categories If the URLs that are checked in this request match any of the provided categories, a detection is reported. See **URL categories** for a list of supported categories.

```
value = token *( ',' token )
```

X-Meta-Charset The MIME content character set (for example `UTF8`). Several commonly used alternate spellings are supported.

```
value = token ; RFC 7230
```

X-Meta-Content-Length The content size.

```
value = 1*DIGIT
```

X-Meta-Content-Type The MIME type of the content (for example `text/html`).

```
value = token “/” token ; RFC 7230
```

X-Meta-SHA1 The SHA1 digest for the content.

```
value = 40*HEXDIG
```

X-Meta-URI The source URI for the content.

```
value = URI
```

X-Meta-IP

The IP address of the content source or originating client.

```
value = IPv4address ; RFC 3986
```

X-Meta-Sender

The email address of the sender of the content.

```
value = addr-spec ; RFC 822
```

X-Meta-Recipients

The email addresses of the recipients of the content.

```
value = addr-spec *( ‘,’ addr-spec )
```

B.3 Response headers

These extended response headers are included by default in the ICAP responses.

X-FSecure-Scan-Result

Reports the final verdict of the scanning process. This header is included in all REQMOD and RESPMOD responses. The following values are possible:

- `clean`: No detection found
- `infected`: Content detected as harmful
- `suspected`: Content detected as suspicious
- `grayware`: Content detected as PUA/UA
- `spam`: Content detected as spam

Example:

```
X-FSecure-Scan-Result: infected
```

X-FSecure-Infection-Name

Reports the infection name. This header is not included if the scan result is `clean`. The infection name always appears in the quoted format described in RFC 2822, surrounded by double quotes, with `"` and `'` characters escaped with an extra `'`.

Example:

```
X-FSecure-Infection-Name: “Eicar_Test_File”
```

X-FSecure-Transaction-Duration

Reports the total time used to process a single request. This is the number of seconds between the time the server finished receiving the ICAP request headers and the time the ICAP response headers were generated. This is more than the sum of scan operation durations, because it also includes the time taken to parse requests and the time spent by the request in various processing queues.

Example:

```
X-FSecure-Transaction-Duration: 0.43920
```

X-FSecure-Infected-Filename

If file is detected as `clean`, `suspected`, or `grayware`, this header may report the name of the file that caused the detection. This header is omitted if the name of the file is not known. Currently, the file name can be reported only if the detection was caused by a file inside an archive or in a MIME email attachment. The file name is URL-encoded to make it possible to report file names that contain non-ASCII characters.

Example:

```
X-FSecure-Infected-Filename: “infected file %E5.exe”
```

X-FSecure-Versions

This header returns version information about the antivirus engines and associated components. The string is not designed to be machine-readable, and the format may change at any time.

Example:

```
X-FSecure-Versions: F-Secure Corporation Hydra/5.15 build
154/2017-01-26_01 F-Secure Corporation Aquarius/1.0 build
8/2017-01-26_11 F-Secure Corporation FMLib/4.66.50.16 build
e70eac0/2016-11-15_01 fsicapd/1.1.0-7f7ae6a
```

X-Attribute: <string list>

This header contains a comma-separated list of categories matching the reputation for the provided URI. The URI can be specified either with the `X-Meta-URI` header or - when it is omitted - through the `Host` header and path in encapsulated request headers. See **URL categories** for a list of possible values.

If Atlant is allowed to call Security Cloud, the response may also contain the following ICAP headers:

X-Location: <string>

If this header is not present, all analysis actions are completed and the result is final. Otherwise, the analysis is incomplete and clients can query for an updated result by sending a REQMOD or RESPMOD request for the resource specified by this header. If scanning a local file, the client should ensure that the local file exists until either the task is complete or the client no longer cares about the result. Failure to access the local file may cause the ICAP request to fail with a `500` error.

X-Retry-After: <seconds>

If present, this header specifies the number of seconds that the client should wait before polling for an updated result.

B.4 URL categories

The following categories are valid in the context of the `X-Forbidden-URI-Categories` and `X-Attribute` headers.

- abortion
- adserving
- adult
- alcohol_and_tobacco
- anonymizers
- auctions
- banking
- blogs
- chat
- dating
- disturbing
- drugs
- entertainment
- file_sharing
- forum
- gambling
- games
- hate
- illegal
- warez
- job_search
- paymentservice
- search_engines
- shopping
- social_networking

- software_download
- spam
- streaming_media
- tracking_cookie
- tracking_domain
- tracking_script
- tracking_object
- travel
- violence
- weapons
- webmail

B.5 ICAP over TLS

Atlant supports securing ICAP communication with TLS (Transport Layer Security). This is a non-standard extension to the ICAP protocol, also known as secure ICAP or ICAPS.

To enable ICAP over TLS:

1. Configure TLS certificates as described in Set up HTTPS. The same certificate and key are used for ICAPS.
2. Configure an ICAPS endpoint by adding it to the `scanning icaps_endpoints` array.

Important: ICAPS depends on the TLS settings (`tls certificate` and `tls key`). If TLS is not configured, ICAPS endpoints cannot accept connections.

Example:

```
/opt/f-secure/atlant/atlant/bin/atlantctl set tls '{"certificate": "/root/cert.pem", "key":
"/root/key.pem"}'
/opt/f-secure/atlant/atlant/bin/atlantctl add scanning icaps_endpoints '{"address": "0.0.0.0",
"port": 11344}'
```

Appendix C

ICAP detection template

You can customize the template of the block page that is sent to the client in an ICAP response when content is blocked.

The template contains the message that is displayed when the ICAP service detects something, for example, a virus detection or a forbidden URL category.

To modify the template, edit the `/etc/opt/f-secure/atlant/atlant/templates/fsicap_infected.html` file.

Note: You need system administrator rights to edit the template.

You can use the following variables in the detection message:

{{scan_result}}	Virus or detection name
{{footer}}	Product version and database timestamp

The changes take effect immediately after editing the template.

Appendix D

Cloud services

The product connects to various services over the network, for example to check for updates to malware definitions.

The product may need to access any subdomain within the `fsapi.com` domain. Ensure it can connect to all addresses under `*.fsapi.com`.

Note: More detailed information about network addresses can be found in the WithSecure Community page.