DICOS API Gateway Manager Documentation

This a to-PDF-converted version of the original HTML/CSS Documentation

DICOS GmbH

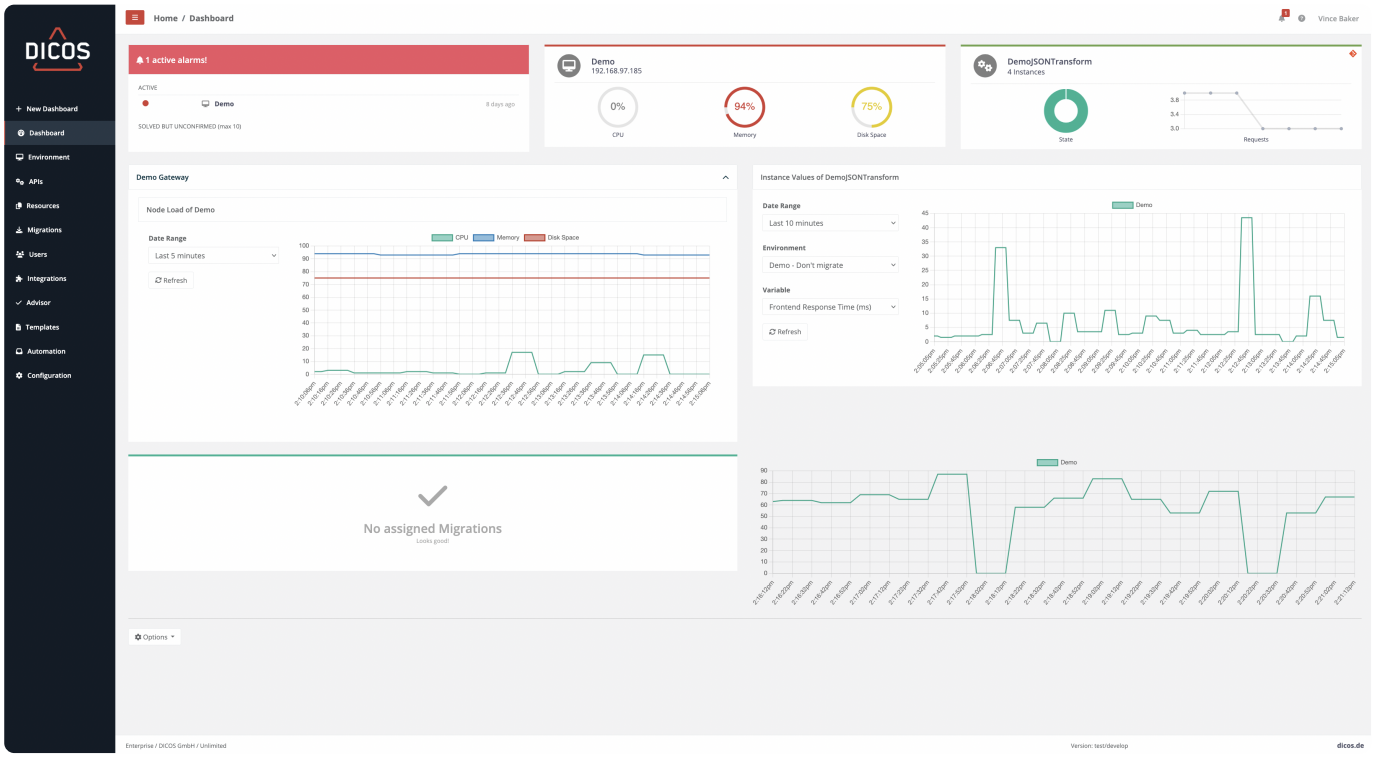
Table of contents

1. DICOS API Gateway Manager

API OPERATIONS AUTOMATION FOR BROADCOM LAYER7

DICOS API Gateway Manager is the market-leading solution for API Operations Automation for your Broadcom Layer7 environment, providing a complete set of features for fast and reliable API Deployment and Migration and comprehensive API Monitoring and Alarming.

API Gateway Manager is the missing link in Broadcom's Layer7 portfolio and allows you to use modern workflows including git and CI/CD support side by side with your API Gateway.



2. Documentation + Knowledge Base

2.1

2.2

- Installation
- Prepare the Gateway
- Upload your License
- First Run Wizard

2.3

- APIs, Folders and Instances
- Nodes, Clusters and Environments
- Migrations and Templates
- APIOps
- Storage Formats
- Dependencies

2.4

- Dashboard
- Environment
- APIs
- Resources
- Migrations
- Users
- Integrations
- Advisor
- Service Templates
- Automations
- Configuration

2.5

- Checking Gateway Manager's Status
- REST API
- Command Line Interface

2.6

- Work with git
- Automate migrations via API
- Deploy a API with Service Templates via API
- Connect an external Prometheus
- Work with the XML Deployer
- Use service variables
- Configure/Install ...
- How to Download Solution Kits and Resources

2.7

- Database related problems

- Access Container Logs
- Access Migration Artifacts
- Certificate Authentication Does Not Work
- Access Gateway Manager Logs
- Unable to upload client certificate to use in node config
- LDAP Login return 504 error message
- My CI/CD Pipeline Stopped Working After Update to 2023.2
- HTTPS Header Size
- Start tag expected, '<' not found
- Invalid Resource while migrating Security Zones

2.8

- Release Notes 2026.0.2-rc1
- Release Notes 2026.0.1
- Release Notes 2026.0
- Release Notes 2025.0.4
- Release Notes 2025.0.3
- Release Notes 2025.0.2
- Release Notes 2025.0.1
- Release Notes 2025.0
- Release Notes 2024.4
- Release Notes 2024.3.2
- Release Notes 2024.3.1
- Release Notes 2024.3
- Release Notes 2024.2
- Upgrade Guides
- Archived Versions

3. API and CLI Reference

3.1 API and CLI Reference

The DICOS API Gateway Manger can be fully integrated into any conceivable CI/CD pipeline. The articles below describe the first steps.

You can find more articles on the topic of API integration here.

Automate migrations via API

Deploy a API with Service Templates via API

3.2 Checking Gateway Manager's Status

If you want to check the Gateway Manager's health you can do so by accessing the alive and ready probes using a GET request to the API. Checking health does not require a valid access token.

3.3 Alive Probe

To check if the container is running you can use the alive probe:

```
https://<yourhostname>/api/v1/alive
```

3.4 Ready Probe

To check if the container is running and if the API Gateway Manager is ready to process incoming request you can use the ready probe:

```
https://<yourhostname>/api/v1/ready
```

3.5 Command Line Interface

The CLI allows you to perform the most common tasks directly from your terminal. It also allows you to interact with Docker-based gateways you run on you own machine that are not reachable from the central Gateway Manager due to firewall or routing issues.

To obtain the command line interface, please contact customer support.

3.6 REST API

The Gateway Manager provides a REST-based API that allows you to manage resources and start migrations. The Swagger (OpenAPI v3) documentation for our API is available directly from within your installation via:

```
http://<HOSTNAME>/api/v1/swagger
```

or

```
https://<HOSTNAME>/api/v1/swagger
```

3.6.1 Open API Documentation

Version: 2024.3 last updated: 10.10.2024

Thumbnail

Unknown Macro

3.7 Authentication

Every call to the API needs to present an access token in the X-TOKEN header of the request. To obtain an access token you need to call the /login endpoint using a valid username and password.

Only users stored in the Gateway Manager's own user table are allowed to perform API calls. SAML or oAuth authentications against the API are not available.

3.8 Test the API

You can test the API right from the SwaggerUI available using the address above. First call the /login endpoint in order to get an access token. Then click on the "Authorize" button in the top right and enter your token there. After doing this you can test all methods right from your browser.

image-20220610-062707.png

3.9 Example workflow

AAGM%20CI%20CD%20async%20.png

4. Concepts

4.1 Concepts

In these articles we will introduce you to the concepts underlying the DICOS API Gateway Manager. We will explain the basic principles and functionalities and how you can use it to optimize and automate your API-based processes.

- APIs, Folders and Instances
- Nodes, Clusters and Environments
- Migrations and Templates
- APIOps
- Storage Formats
- Dependencies

4.2 APIOps

APIOps is a methodology that applies the principles of DevOps to the development, provision and management of APIs. The aim is to improve the efficiency and reliability of the API lifecycle.

4.2.1 Automated Creation of APIs

Kickstart the development of new APIs while ensuring standardization. The DICOS API Gateway Manager automates the API creation process, saving you time and effort while maintaining consistency and quality.

Deploy a API with Service Templates via API

4.2.2 True Versioning

Version control is a central element of effective API management. With the DICOS API Gateway Manager you can version your Layer7 API using Git for better tansparency, collaboration, and traceability throughout the development and maintenance stages.

Work with git

4.2.3 Automated Delivery

Simplify and automate the release process of your APIs. This reduces the chances of human error and ensures that your APIs are consistently delivered on time.

Automate migrations via API

4.3 APIs, Folders and Instances

4.4 API

In DICOS API Gateway Manager, we are using the term API as being a Managed Service. Think of API as some kind of enhanced Layer7 service. An API can consist of one or more Layer7 services and is augmented with additional metadata. This metadata allows the Gateway Manager to perform additional tasks of which the Layer7 gateway itself is not capable.

You can monitor the basic metrics of each API like the number of requests or frontend and backend response times. Depending on the environment different alarming thresholds can be defined that trigger warnings and alarms at given values. You might want to define a rule, that alarms your team via Slack whenever the frontend response time raises above 100ms in production. For more information on alarming please consult the corresponding section of this documentation.

Another capability of APIs is the use of a repository. These repositories act as the single point of truth for an API. DICOS API Gateway Manager allows you to migrate a specific version stored in that repository to your gateways. You don’t need to manually import that version into some kind of integration gateway first. For more information on the possibilities of repository-based migrations or migrations in general please refer to the corresponding sections of this documentation.

Work with git

Please note that this functionality is only available in the Enterprise Edition of DICOS API Gateway Manager.

4.5 Folder

Folders first correspond to the folders from the gateway. They group the APIs and create a logical unit. If you register entire folders in the Gateway Manager, you also register all the underlying services.

4.6 Instance

When a folder or service is registered, it is created in the Gateway Manager and the instance of it is created at the same time. The instance is the deployed version of the API running on a gateway. An API cannot occur twice, but there is one instance for each gateway on which the API is present.

4.7 Dependencies

The dependency tree is built when a gateway's resources are imported into DAGM. There are three kinds of dependencies in DAGM (while there are only two in the gateway itself).

4.8 Regular Dependencies

Regular dependencies are the most basic form of a dependency. One resource requires another resource to be present to work correctly. This might be a service referencing a JDBC connection, a policy fragment or a certificate for example.

4.9 Dynamic Dependencies

Dynamic dependencies are resolved during runtime. Instead of referencing the dependency directly a cluster property is referenced that holds the name of the dependency. This way it is possible for library authors to write policies that can be configured in the customer's environment.

It is possible to reference either cluster properties (${gateway.NAME}) or variables within the policy itself (${NAME}).

DAGM is currently not able to resolve variables that are defined within a policy.

4.10 Missing Dependencies

Missing dependencies are either regular or dynamic. The target of the dependency is not available at the gateway. While migrating a service or policy with a missing dependency might work, running it however will fail during runtime.

4.11 Migrations and Templates

4.12 Migrations

Migrations within the DICOS API Gateway Manager resemble what you're accustomed to from the Layer7 gateway itself. They encompass the resources you intend to migrate from one gateway to another, along with most of their dependencies. However, the Gateway Manager enhances the migration experience by automatically addressing common pitfalls. This facilitates fast, reliable, and, most importantly, repeatable migrations without necessitating an in-depth understanding of the gateway's inner workings. As a result, your team can elevate its DevOps capabilities, ensuring that migration tasks aren't restricted to the most senior team member.

4.13 Templates

Service templates allow you to create starters for new services. Most services you will deploy on your gateway share boilerplate code that is almost the same throughout the board. Using the API Gateway Manager you can create templates that contain common fragments and have variables that can be replaced upon deploying new services based on this template. To make the template more versatile you can define blocks of assertion that are included or excluded based on choices presented to the user during service creation.

4.14 Nodes, Clusters and Environments

4.15 Nodes and Clusters

Within DICOS API Gateway Manager your Layer7 Gateways are referred to as nodes. As you are used to nodes can be grouped to clusters. As you will notice a cluster doesn’t have any connection details or credentials. This is due to the fact, that when you use API Gateway Manager’s monitoring functionality it will fetch metrics for every node in the cluster. Within the API Gateway Manager the cluster is only used to create derived metrics, i.e. the average response times or the sum of requests handled by the cluster in total. All operations and alarming thresholds are configured on the node level allowing you to configure those values differently even within a cluster. As this fine grained control doesn’t make a lot of sense when talking about migrations DICOS API Gateway Manager only shows the cluster a node belongs to during migrations. As migrating to or from any of the nodes in the cluster will yield the same results, there is no point in showing all of them in the selection boxes are in the resource explorer.

4.16 Environments

Nodes and clusters can be assigned to environments. The basic environments that are created by default in a fresh installation of API Gateway Manager are Development, Test and Production. However you can create as many environments as you like. For more information on environments see the corresponding section of the documentation.

4.17 Storage Formats

4.17.1 Storage Formats

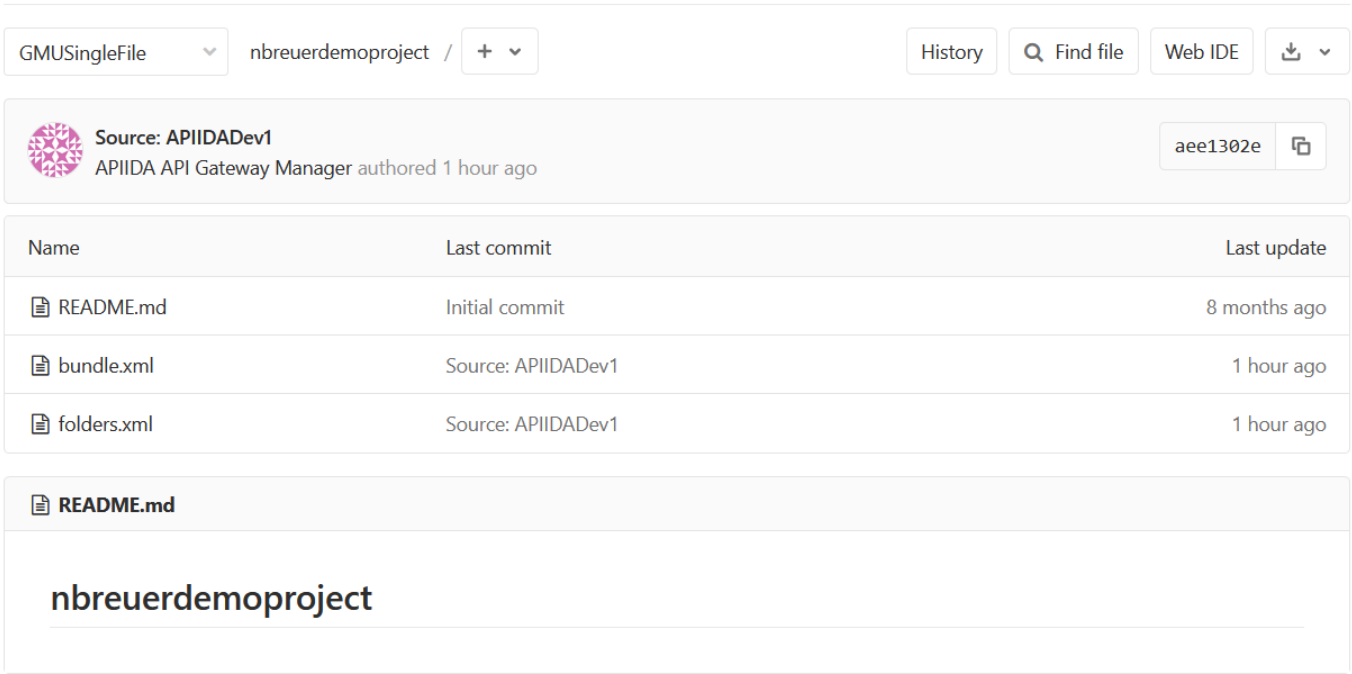
The Gateway Manager supports the storage of API definitions in different formats. The format defines how the service is stored and differs also in the way differences can be managed within the repository. There are two supported file formats: “GMU Single File” and “DICOS Enhanced”. The following article explains the differences between those formats.

Please note that GMU Single File is primarily meant to read already existing repositories. Its use is highly discouraged, as it is not well suited for git-based storage. This format contains timestamps and version numbers, that change with every push to the repository, making it hard to track what actually changed. DICOS’s Enhanced Format takes care of this by removing timestamps and version numbers before a push to a git repository.

DICOS’s Enhanced Format also splits up the actual policies from configuration. This eliminates nested XML structures. Everything that is not a policy is also stored conveniently in JSON, so that automated processes are much easier to implement. On top of that JSON is a much better fit for git, as it maintains its validity in automatic merges most of the time, while XML tends to break.

GMU Single File

The GMU Single File format relates to the storage format used by the Layer7 Tool “Gateway Migration Utility”. The storage format is shown in the following image:

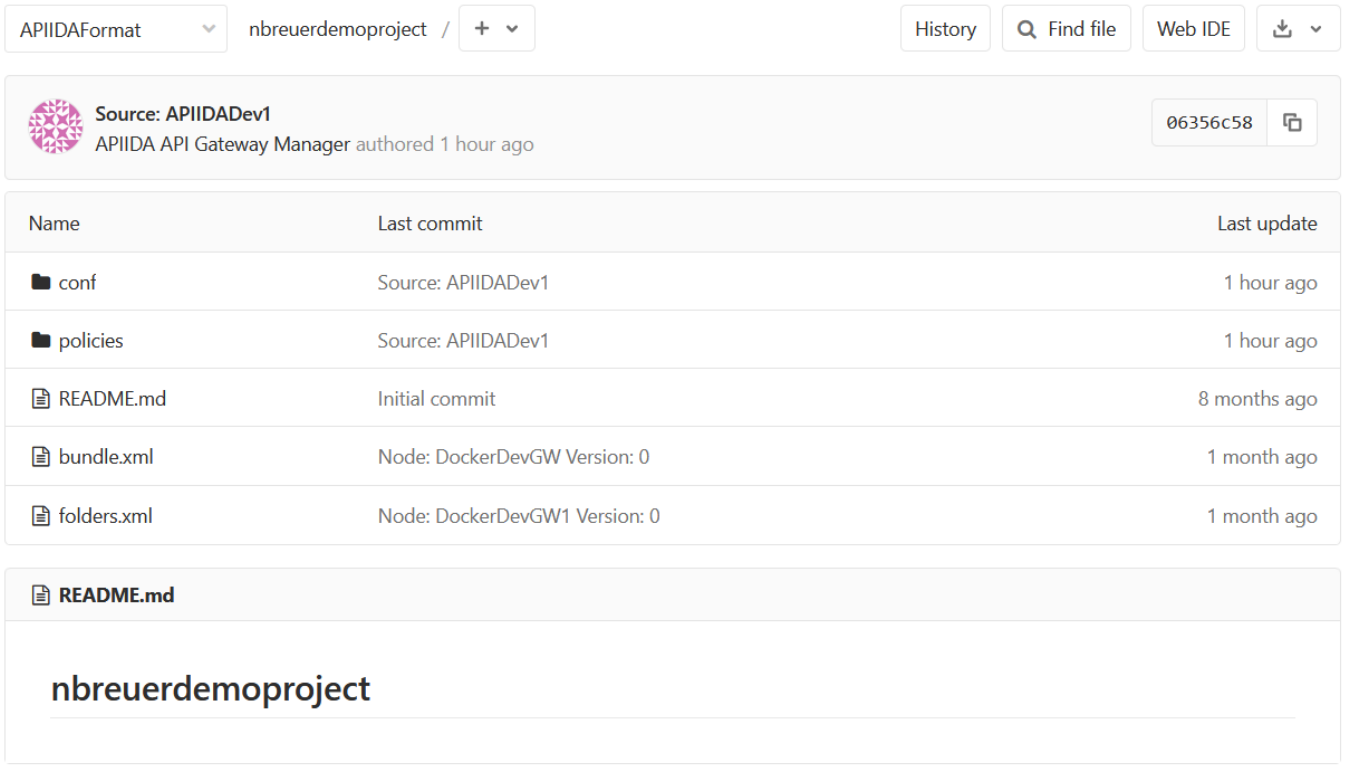


In this format, the service definitions are stored within a “Bundle.xml” file. For reference purposes, a seperate “folders.xml” file is added which stores the folder hirarchy of the service. An example for the content of the folders.xml file is shown below:

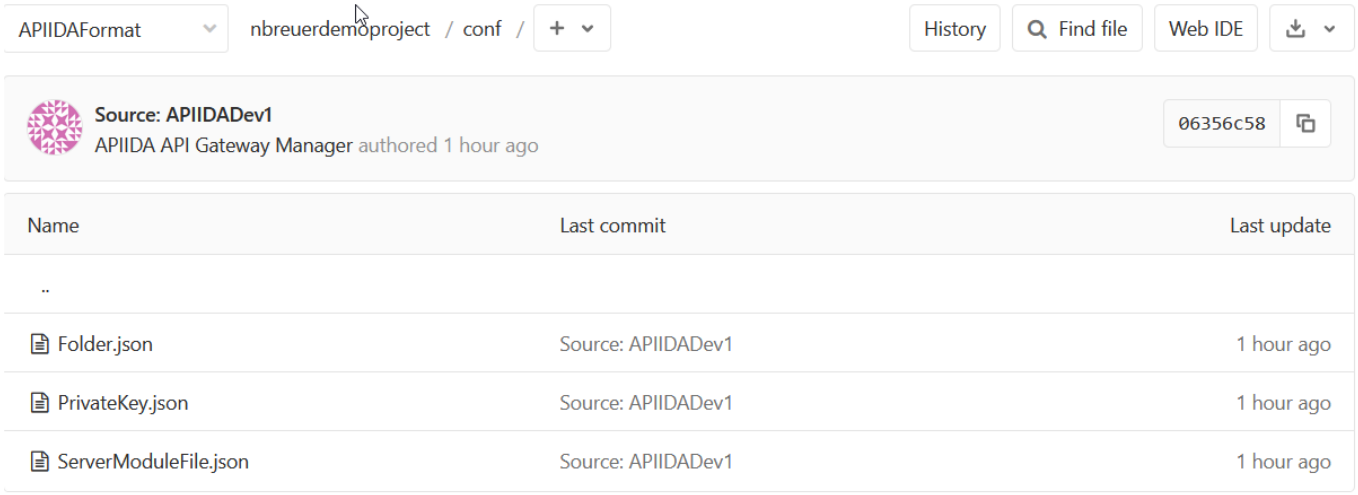
To get a more detailed description of the service definitions within the “bundle.xml”, please refer to the Restman Documentation on your Layer7 API Gateway.

DICOS Enhanced

The “DICOS Enhanced” format adds two folders to the described “GMU Single File” format above. A folder “policies” which contains one or more individual service definitions in XML as objects. Additionally, a “conf” folder is added which contains several configuration descriptions in JSON. The format is shown below:



In this example, the content of the conf folder shows three files:



These JSON files contain descriptions of several resources used within one or multiple policies. The definition of “resource” used here follows the official Layer7 definition (review the Restman documentation for a complete list of resources).

EXAMPLES:

“Folder.json” contains a JSON object which defines the folder of the solution:

```
{
    "folder": [
        {
            "folderId": "0000000000000000ffffffffffffec76",
            "name": "Demo",
            "id": "b56f10f3c04cd607eb080dce4b0426e5",
            "version": 2
        }
    ]
}
```

“PrivateKey.json” contains the definition of an encrypted private key and the corresponding certificate used within the services:

```
{
    "privatekey": [
        {
            "alias": "SSL",
            "keystoreId": "0000000000000000000000000000002",
            "certificateChain": {
                "certificateData": [
                    {
                        "issuerName": "CN%3D6d3095b61858",
                        "serialNumber": "3850365724295781246",
                        "subjectName": "CN%3D6d3095b61858",
                        "encoded": "MIIC8jCCAdqgAwIBAgIINW8\/
OeP9W34wDQYJKoZIhvcNAQEMBQAwFzEVMBMGA1UEAxMMNmQzMDk1YjYxODU4MB4XDTE5MDcwNDA5MjgzM1oXDTI5MDcwMTA5MjgzM1owFzEVMBMGA1UEAxMMNmQzMDk1YjYxODU4MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAozrf552zOGxS\/
ncF36U8DVuQL1RjDusyWkqVPPuS4dG9tBpvyMm7RzfphetbOMVvB1LxHXhmW7WdSV3qgHBTRPPe4Yp+7zyNtuHUAVQ5geNDKd\/OAbRK5KmVld5Fd6cldZkSoSjII+AOj\/WncNQh1NCVuezUfVtrZn6LW9enI+6lRgAOIae8tsoA5QHWGbwd25EKwl2JzI\/
3eGH1cUO1ZPnQuroWKsw+\/
klUXsJ+Mgb4ScdTP+32qKHyK7Dk4ROCYCXx5yF6PeuCoWOoMO6MO7KsPGps83ZRrStusJ7lY8IxfkiGZnt3nkkc9VkFmd78fWo21fs9tSIQy7YUE6GMuQIDAQABoOIwQDAdBgNVHQ4EFgQU13Uj1Z62ksi59h2S9ZaNyhv+Tn8wHwYDVR0jBBgwFoAU13Uj1Z62ksi59h2S9ZaNyhv+Tn8wDQYJKoZIhvcNAQEMBQADggEBAIK4iZf1RwkT9IsAD8siIyGj52c6TOP+xkAeyASlJXUA1ZoRl8UvUl6zK4qLaJtIgUyUGZOHFJdOR45I\/
NUcB6iQ1ZimReChao9vEA1wDZwJjkRXyrm+sxIhxdBUbBIlQ2o4teZOXadWyo6cxW9+iqf+YL+7ybahLmHvTcgQUPYmsPY8iH3NfvNBASgMjOW+Y61IBZifTYUSYTSnQZVKC2IMTVURZPv78LF4Yklx28VLpYoDcnEGumJ7+QWNhlGSDrjibD4ABDiTMDmEtwGZ
UT90G+MWaSZFOAOwjPPWK9BoCw3XjimHyFOTf89JJZkbiLvK7t2GS5CQiq\/q\/YWSubQ="
                    }
                ]
            },
            "properties": {
                "property": [
                    {
                        "key": "keyAlgorithm",
                        "stringValue": {
                            "value": "RSA"
                        }
                    },
                    {
                        "key": "keyData",
                        "stringValue": {
                            "value": "$L7C2$1,X7tYy\/bwua577cwHdHutiwl9K4RKA1YzTztvOysTR1c=$oWq\/Y+5gHUTrmyZK2kBT5Wz66E5YWtUugeXSfGoKOAOybSclyPT9FZdTICpw+
+xzNeYmQr+dMURJgLbjCqYktLi81HhksWzKBNeAgC75BOxS+fX0a4TRwFu+8736JZI+qPSAJcaJqtvHa69+234A418CW+1BqrII1DQ5jQwPbWOSTwCFEYLBUwHbPcVOq2aynUG3wHxUkgOJllF\/
iRLuq5Ph93ElB1etwpB8uCvUVhkXuhEy7pf7Od9y6QFNDbOnRmZkFd77idByaFL3USDqdm3\/cKlwgEPMmwFwTBwNYjlX5j\/oNZlT4fkoSvlG+qefvj35U9lGpCIm12O\/aihVVUAxUpMRKHpG\/
cxVjrT55i3L1mzSHnjXhC1FH7s0f9xuUAgpg+OD76ffljgFTymlXOu1UxsHXyv7KNHfTZa6eLHsGiZU5MtbLeHBhNFisz6+pVkSSFviKIGWeJsCn2oXHWGMH+pfjiAbTNHrmfKuKxJQrRAjDV9ijLzs1ulEJLp\/
mgcVviWKyJOfyxCt9iULPRr5axPIdQL5J6w518cE5xtE26bpqqseJaqHPqwQrGUZl6Q21+V1j1lfXF5Sx\/B7loBlMk3WESTO\/FyO663EB4\/WtmoD\/2A\/XcffOTTejG2iOOwxUgTbrDQ5XW4KlidOSYyTmxrpe\/
AlTrVxOF41o6KBaSz6XMkojr14xZkEoWDs1OUjrTUenXqE76NkIvZeBwkrsye26G2SZNy1bJwc4WhsmZbdPMjSyD+QBRAL52xpU8Z+tA\/
PlBH4kvizRgxDZlV47z1OeW86aHfvqh3Np78FFJXcEm6uey5ncVUlhcau9DIlPaWmPvofU5w1KLizgxcVeIR2h5cFPj8gCYIksWhDONHTd9W8InePnmjIw29Yhg2wBcDWdxEflWj9rucltH\/
```

```
WwGbNlpeC51SqVp8YCcrnt88Uj5WzOzOp9WwoKKWCcnud5nOuEJrViHthEHMCFybexOIkvnMr9qI33+9kusyQhs3jRkcJb+IyqsRqrBpf5opHjIIK1PxaBYsVm9eGwyyxPzxaZ3xc1OKEgOexu9h25Ccffw4zLOdvakYz5IC3lI7ZsTFZv9xro7S8w7amVwyr+M
ReORO6Q\/rOA7xqLq3ovWPfKoZNQ\/eL\/fzZ\/kA5z9Nl1ndwA7UPOKKycx9wlwoxxkwB9iPp3UGyE8QV3UMHKb4qGZeDYQmn4XJmsvsQX+tOIKexo\/wjHbVn4nKumer2INQ6GA\/Puj8AcNzI+pTcMYi\/
Uv932sSr8WkKrPPt2ZWwlWxzAP2YtD626fhAETqY6ZPxbQ+eoTJxoTxO6lbkjSiYzOONajv8nWNbLYnDGXFrzTEkMkyC2HqthZYx899WescyPN\/QrzC29biAw1\/iRjecS2ul\/1nfWOLo4M3LDaVXu4kaV1OGPkcgjlgakx+nbB7j2r1ivrbOl7\/
QhjUpdAAOlh1WMFVO93GdtrtrXRCbJXTVB2c62bqDYBs3bZyLdPtWKy7ofpgL6idMTebdhY+zVOcaRzRMKZuYUUhUnnc539oj6P4a1yKnOWKSsMSeiYLMt1\/c6XBvtM5djhkASaQqwez9Vx\/
rEeHdAvbAnjwAF1M8wrExSn3QZiYOVO8TagDpEjQMU0dQ9TiHSWYz7+OJJuiNuuRmkOOF4SB6OB1MuQS6PnoK2Lh1DWMQ4TUNfkyxlRAlGp2a9teMFGo=",
                        "bundleKey": "$L7C2$1bead,519aYRZKLTt71Mc96vNRtbvijxCHLgjuhNtM9pShwKE=$HWMVeY4zf2HYVgwVU1oGy18XcztMNsHPeme5KgIBgO8QIpuzD9uFfDjfeI4ysDGfhqvr1ri9Y+hchP2wXNt5veaWD\/
O9dkk6p6cgpEbN+k2HA4sglp1IFO5OYAMJQLmt",
                        "algorithm": "RSA"
                    }
                },
                {
                    "key": "specialPurposes",
                    "stringValue": {
                        "value": "Demo Key"
                    }
                }
            ]
        },
        "id": "00000000000000000000000000000002:SSL"
    }
]
}
```

4.17.2 DICOS Enhanced Format

When storing an API in a git repository the gateway's policies and configuration resources must be exported to files. API Gateway Manager supports two different formats:

- GMU Single File
- DICOS Enhanced Format

Please note that GMU Single File is primarily meant to read already existing repositories. Its use is highly discouraged, as it is not well suited for git-based storage. This format contains timestamps and version numbers, that change with every push to the repository, making it hard to track what actually changed. DICOS's Enhanced Format takes care of this by removing timestamps and version numbers before a push to a git repository.

DICOS's Enhanced Format also splits up the actual policies from configuration. This eliminates nested XML structures. Everything that is not a policy is also stored conveniently in JSON, so that automated processes are much easier to implement. On top of that JSON is a much better fit for git, as it maintains its validity in automatic merges most of the time, while XML tends to break.

5. Configuration and Usage

5.1 Configuration and Usage

This section describes how to configure and use the DICOS API Gateway Manager. You learn how to configure a Gateway Manager instance and how to prepare your Layer7 API Gateways.

The following terms are used throughout this section:

- "Gateway Manager" refers to DICOS API Gateway Manager.
- "Gateway" refers to the Layer7 API Gateway by Broadcom.
- "Policy Manager" refers to the Layer7 API Gateway Policy Manager.

5.1.1 Audience and Assumptions

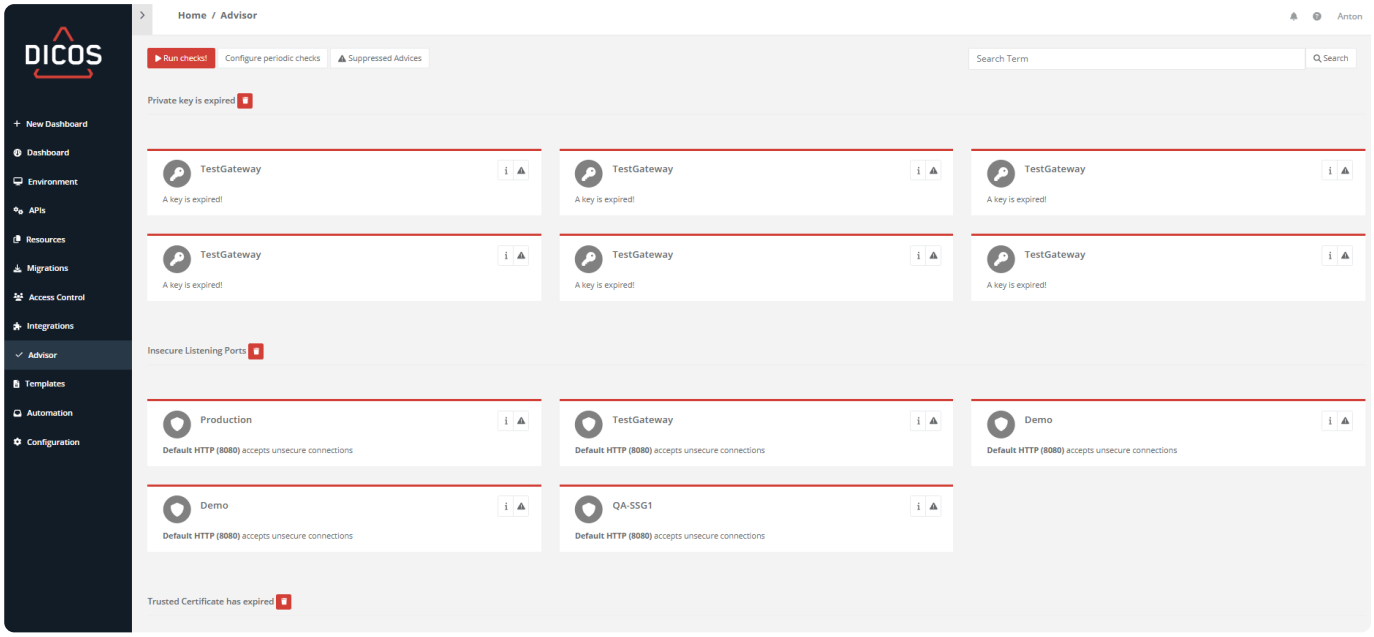
These instructions are intended for facility coordinators, security managers, and other IT infrastructure staff familiar with:

- The hardware and software infrastructure that will be used with the DICOS API Gateway Manager or Layer7 API Management products, such as firewalls, Layer 2 or Layer 3 switches, routers, Load Balancers, databases, identity management or access control systems, application servers, and more.
- Advanced TCP-IP and internet working concepts, web services, and user management knowledge.
- (For Docker-based installation) Advanced knowledge of the host operating system.

5.2 Advisor

The Advisor continuously monitors and analyses your Layer7 API Gateways and helps you stay secure and adhere to the latest best practices.

If you change the environment of a node, please run the checks again to renew any outdated advices.



5.3 Acting on Advices

All advices can be viewed through the “Advisor” link in the left-hand menu.

If you want to be proactively informed about new advices check out the automation documentation. You can configure a lot of automations that are executed whenever a advice is created. This might be everything from sending a simple email to the admin to more complex scenarios as creating a related Jira issue. Most integrations offer to bring over the actions, so that you can act on advices right from Jira for example.

5.3.1 Run Checks

To get a valid result please make sure that all imported resources are up to date. If not, you can ensure this in "Resources" -> "Reload Resources" -> "Reload Resources from all Gateways".

5.3.2 Search

To get a better overview, you can filter all advices by environments and nodes.

5.4 Suppressing Advices

If you think that a advice is a false positive, you can suppress it. This causes it to never be raised again until you end the suppression. To do this you just click the corresponding button in the action drop-down.

5.5 Types of Advices

5.5.1 Expiring Private Keys

Whenever the advisor detects, that one of a gateway's private keys is about to expire it will create a advice. Expiring private keys have the potential to cause serious issues in production when API consumers stop calling your APIs because they cannot establish a trusted and secure connection to your gateway.

5.5.2 Expiring Trusted Certificates

As private keys, expiring trusted certificates have the potential to bring down production as well. Be it because backend systems can no longer be called, or that authentication tokens cannot be signed.

5.5.3 Missing Resources Within the Same Environment

With the trend to decluster environments it is crucial that all the needed resources are present on all gateways of an environment. Whenever a new resource is detected that is missing on other gateways within the same environment, this recommendation is created. It is aware of ongoing migrations (when conducted via the API Gateway Manager), so that it does not create false positives during migration runs.

5.6 Differing Resources Within the Same Environment

Similar to the missing resources recommendation, this one checks the resources for their content. Changing information like version number or IDs are extracted before the comparison. This makes it easy to spot differences within your environment that lead to bugs or - even worse - different behaviour depending on on which gateway a request is processed. These are probably the bugs that are most hard to find, especially with a lot of gateways in a environment.

5.6.1 Insecure Listening Ports

This advice warn you about listen ports, that do not use SSL/TLS to establish a secure and encrypted connection. Only using secure TLS-enabled traffic is considered a best practice and conforms to the Zero Trust mindset.

5.7 Automations

Automations allow you to perform certain actions when a trigger happens. Beside the basic automations provided by the API Gateway Manager out of the box, integrations can add their own automations.

Automations can be accessed using different ways. All of them can be managed via the “Automations” link in the left-hand menu. Automations limited to a certain environment, a certain service or a certain gateways can additionally be managed right within the details of the corresponding resource.

5.8 Triggers

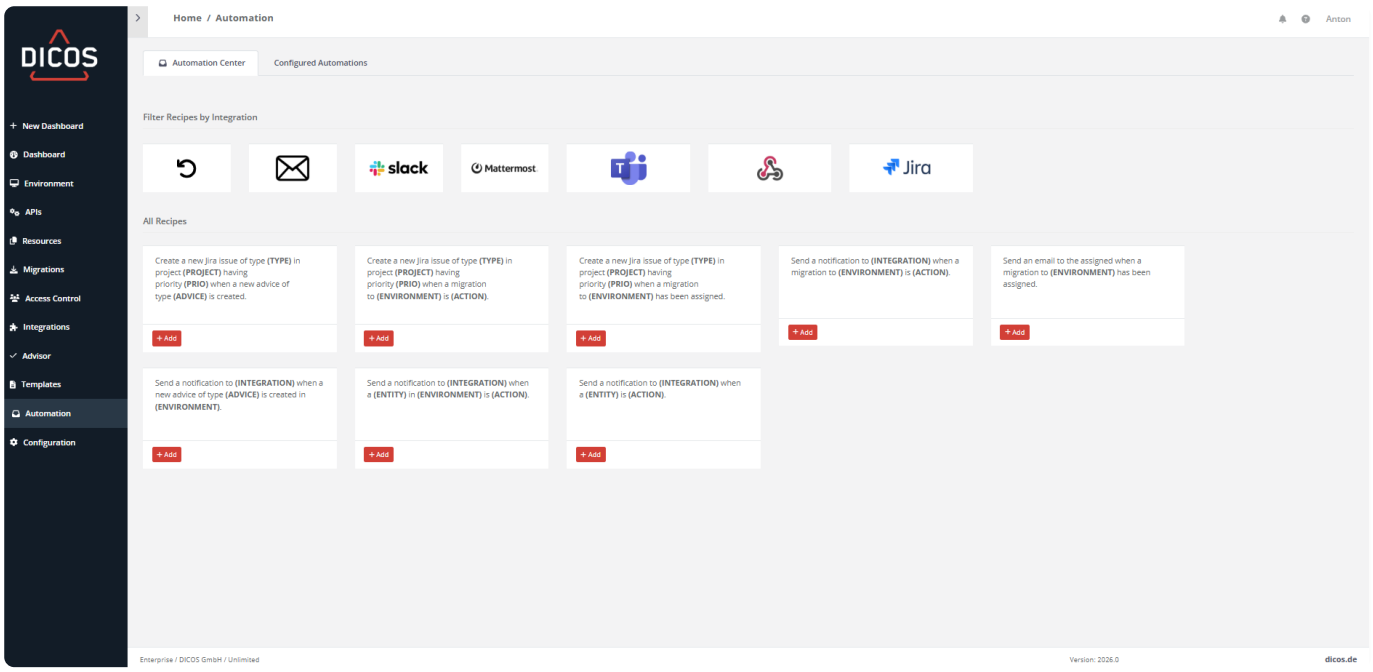
The following triggers are availabe to attach an automation to:

- Recommendation Created: This trigger is fired whenever the Advisor feature creates new recommandaten for you. See the Advisor documentation for a list of available recommendations. The trigger can be limited to a certain recommendation type, i.e. the automation is only started when a recommendation of a certain type is created.
- Migration Assigned: Start an automation (like creating a issue in Jira or send a reminder via Slack) whenever a migration is assigned to a user.
- Migration Started: This trigger is fired whenever a migration is started. The trigger can be limited to certain environments. It is fired in every case regardless if the migration was started using the UI, the API or the command line utiilty.
- Migration Completed: You can start automations whenever a migration completes. This trigger fires regardless if the migration was successful or not.
- Migration Successful: In contrast to the “Migration Completed” trigger this one only fires if the migration is successful.
- Migration Failed: Corresponding to the “Migration Successful” trigger this one only fires if a migration fails.
- Entity Created|Updated|Deleted: This trigger is released whenever an entity such as a user or environment is created, updated or deleted. The trigger can be limited to certain environments for entities such as clusters or nodes.

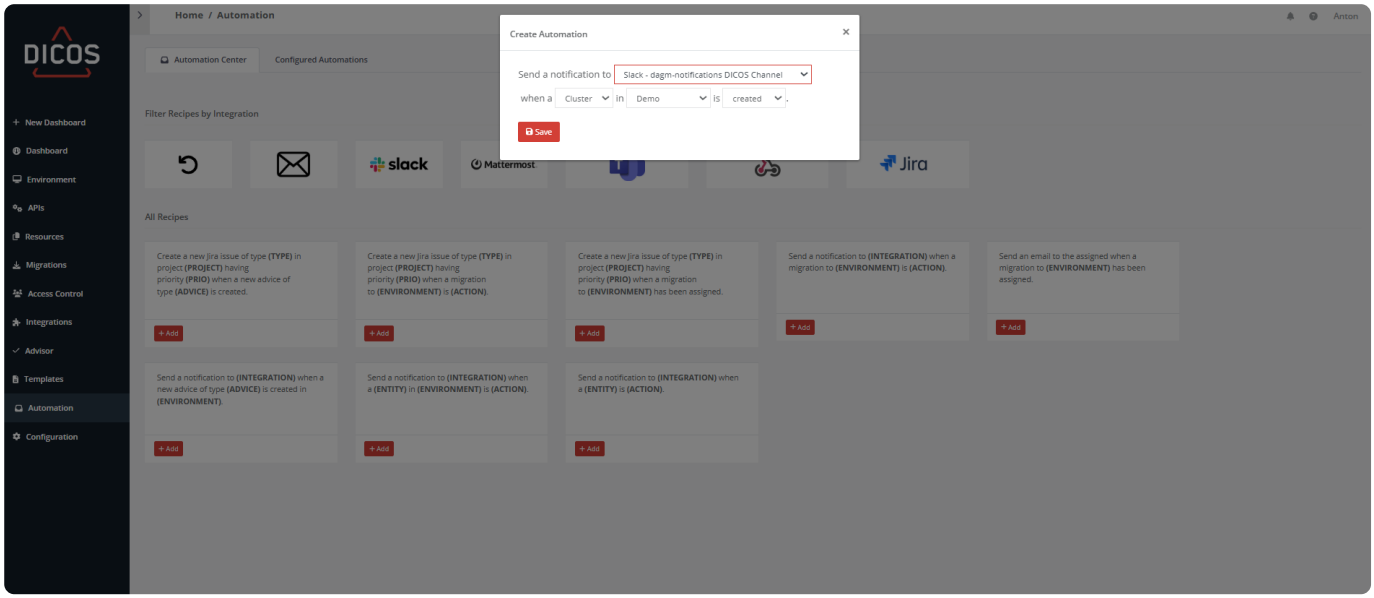
5.9 Recipes

Automations are based on recipes. Each recipes contains at least a trigger and a action that is about to be done.

5.10 Create Automation



If you have created a certain number of integrations, it may be useful to filter the recipes because the selection of (INTEGRATION) within the recipe is also filtered by the type of integration.



Example: Create a issue in Jira if a migration fails.

Besides the recipes the API Gateway Manager has out of the box, each integration can provide new recipes.

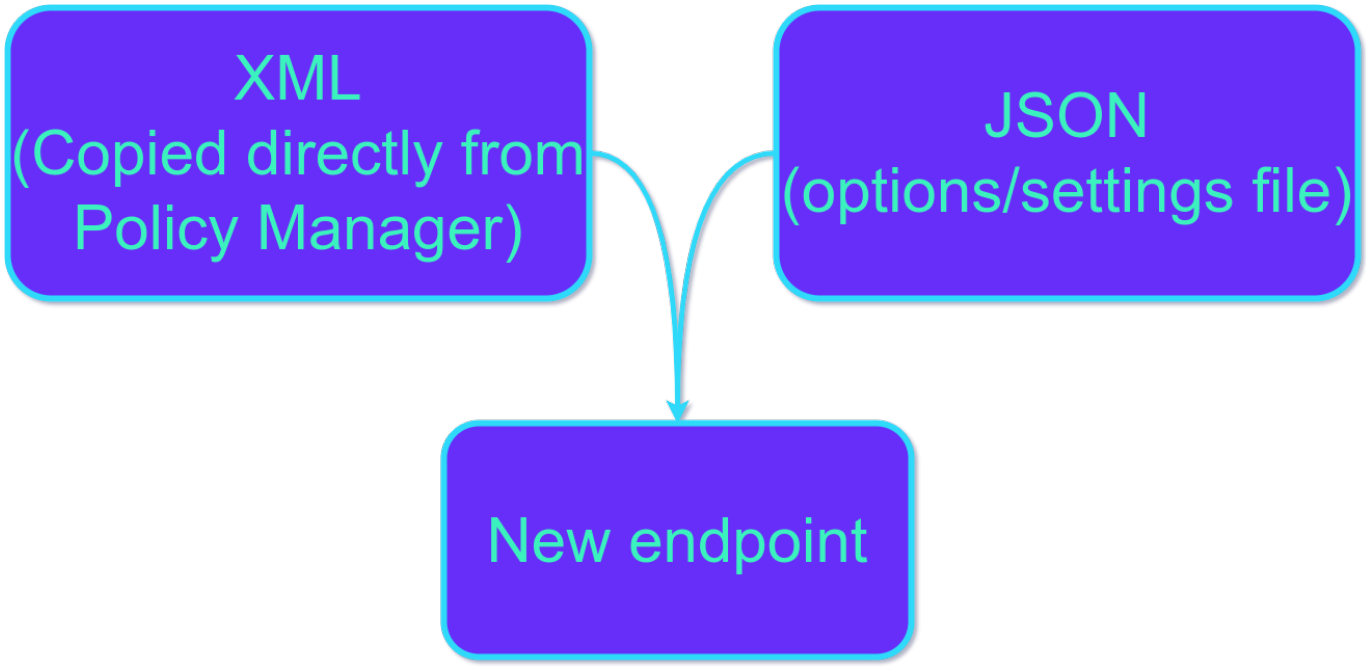
5.11 Service Templates

- Service templates provide a reusable foundation for new services. This reduces repetitive coding for common elements.
- Use the API Gateway Manager to create templates:
    - Include standard code blocks that are used across most services.
    - Add variables to be filled in during service deployment.
    - Define optional assertion blocks for flexible customization.

5.12 Template Parts

Templates have two key components:

- JSON file: Outlines the template structure, including all variables and conditional blocks.
- XML file: Holds the policy that will govern the new service.



5.13 Definition descriptions

5.13.1 XML Policy Definition

How to get the policy XML

Getting policy XML: Two easy methods:

- Policy Manager: Export the XML from an existing service.
- Copy & Paste: Select the assertions you want and directly paste the generated XML code into your editor/form.

5.13.2 JSON Definition

First, let's look at an example and then break it down:

```
{
  "$schema": "https://apiida.com/schemas/layer7-template",
  "$version": "1.0",
  "name": "Basic template example",
  "description": "This is a basic template to demo service templates",
  "blocks": [
    {
      "name": "useSSL",
      "description": "This API will only be available via SSL / https",
      "group": "Authentication"
    },
    {
      "name": "auth",
      "description": "Means of authentication",
      "options": [
        {
          "value": "basic",
          "text": "http Basic Authentication"
        },
        {
          "value": "cert",
          "text": "Client Certificate"
        }
      ],
      "group": "Authentication"
    },
    {
      "name": "validateBackendResults",
      "description": "Validate backend results against the Swagger-File"
    }
  ],
  "variables": [
    {
      "name": "comment",
      "description": "This value is inserted into the policy",
      "type": "string",
      "required": true
    },
    {
      "name": "date",
      "description": "Example for another datatype",
      "type": "date",
      "required": true,
      "group": "Authentication"
    },
    {
      "name": "someElement",
      "description": "Select something",
      "type": "select",
      "required": false,
      "options": [
        {
          "value": "A",
          "text": "This inserts A"
        },
        {
          "value": "B",
          "text": "This inserts B"
        }
      ]
    }
  ]
}
```

Variables, Blocks and Groups

Templates use variables, blocks, and groups for customization:

- Variables: Placeholders filled in by the user when creating a new service.
- Blocks: Sections of policy code included or excluded based on user choices.
- Groups: Organize related settings for a cleaner template structure. Within groups, variables and blocks maintain the order from your JSON definition.

Data Types

Type	Description
string	This datatype acts like a normal string. Please be advised that its inserted as it is. Which means that no base64 transformation happens. This datatype is not allowed to contain XML or HTML.
integer	A number without a fractional component (i.e. 5)
float	A number with a fractional component (i.e. 5.25)
date	A date value without time information
time	A time (formatted according to the format defined in the configuration)
datetime	A date value with time information
rawstring	In contrast to the string data type, this type allows any string, including XML or HTML.
select	A list of options. The value that is inserted and the text that is shown can be different from each other.

Modifiers

Modifiers define additional constraints for the entered values.

Modifier	Description
required	true / false. If a variable is declared to be required it will not be possible to create a new service if no value is entered in the variable.

5.13.3 XML Policy Definition with placeholders to work with JSON Definition

- Variables: Replace placeholders with double braces. Example: {{service_name}}
- Blocks: Embed definitions within XML comments for valid structure. Example: ``
- Select blocks: For dynamic content use this format: {{block_name=value}}. Example: `{{authentication_type=basic}}`

The format for a block is:

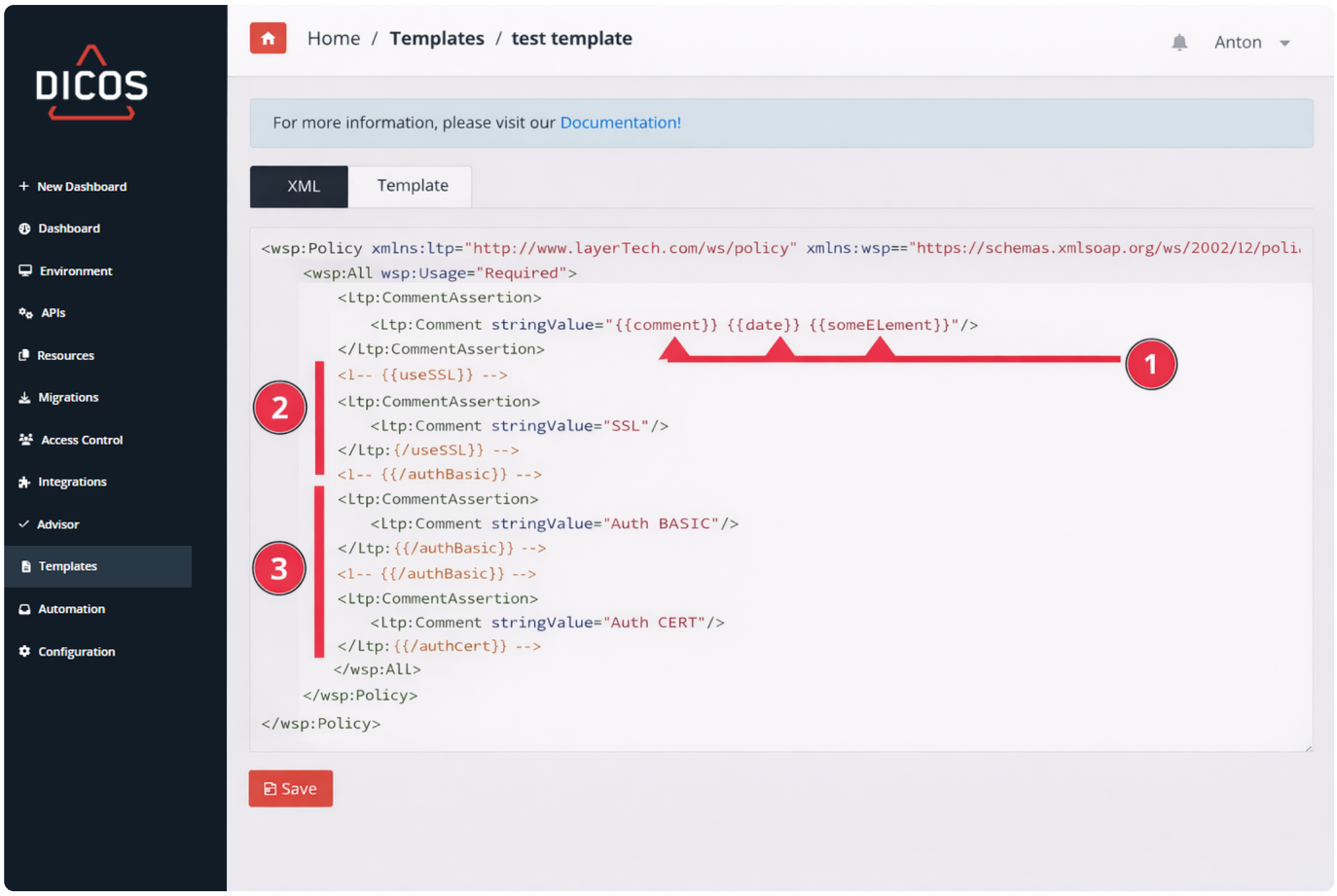
```
<!-- {{myBlock}} -->
<... logic to include if myBlock is selected ...>
<!-- {{/myBlock}} -->

<!-- {{myBlockA=choice1}} -->
<... logic to include if choice1 is selected ...>
<!-- {{/myBlockA}} -->
<!-- {{myBlockA=choice2}} -->
<... logic to include if choice2 is selected ...>
<!-- {{/myBlockA}} -->

<L7p:Comment stringValue="{{someElement}}"/>
```

5.14 Basic example:

5.14.1 XML Policy Definition

Get your policy XML by:

- Exporting from an existing service within the Policy Manager.
- Selecting specific assertions and using Copy & Paste.
- Example: We illustrate this with a basic policy containing four comments.



Example: Policy Manager Export

- The code below shows what the export looks like. This is your starting point.
- Important: You must edit this example to include placeholders and triggers (instructions provided 2 sections below).

```
<wsp:Policy xmlns:L7p="http://www.layer7tech.com/ws/policy" xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy">
    <wsp:All wsp:Usage="Required">
        <L7p:CommentAssertion>
            <L7p:Comment stringValue="some comment"/>
        </L7p:CommentAssertion>
        <L7p:CommentAssertion>
            <L7p:Comment stringValue="SSL"/>
        </L7p:CommentAssertion>
        <L7p:CommentAssertion>
            <L7p:Comment stringValue="Auth BASIC"/>
        </L7p:CommentAssertion>
        <L7p:CommentAssertion>
            <L7p:Comment stringValue="Auth CERT"/>
        </L7p:CommentAssertion>
    </wsp:All>
</wsp:Policy>
```

5.14.2 JSON Definition

Here is the example again, Copy and paste it under the Template tab:

```
{
  "$schema": "https://apiida.com/schemas/layer7-template",
  "$version": "1.0",
  "name": "Basic template example",
  "description": "This is a basic template to demo service templates",
  "blocks": [
    {
      "name": "useSSL",
      "description": "This API will only be available via SSL / https",
      "group": "Authentication"
    },
    {
      "name": "auth",
```

```
      "description": "Means of authentication",
      "options": [
        {
          "value": "basic",
          "text": "http Basic Authentication"
        },
        {
          "value": "cert",
          "text": "Client Certificate"
        }
      ],
      "group": "Authentication"
    },
    {
      "name": "validateBackendResults",
      "description": "Validate backend results against the Swagger-File"
    }
  ],
  "variables": [
    {
      "name": "comment",
      "description": "This value is inserted into the policy",
      "type": "string",
      "required": true
    },
    {
      "name": "date",
      "description": "Example for another datatype",
      "type": "date",
      "required": true,
      "group": "Authentication"
    },
    {
      "name": "someElement",
      "description": "Select something",
      "type": "select",
      "required": false,
      "options": [
        {
          "value": "A",
          "text": "This inserts A"
        },
        {
          "value": "B",
          "text": "This inserts B"
        }
      ]
    }
  ]
}
```

5.14.3 XML Policy Definition with placeholders to work with JSON Definition

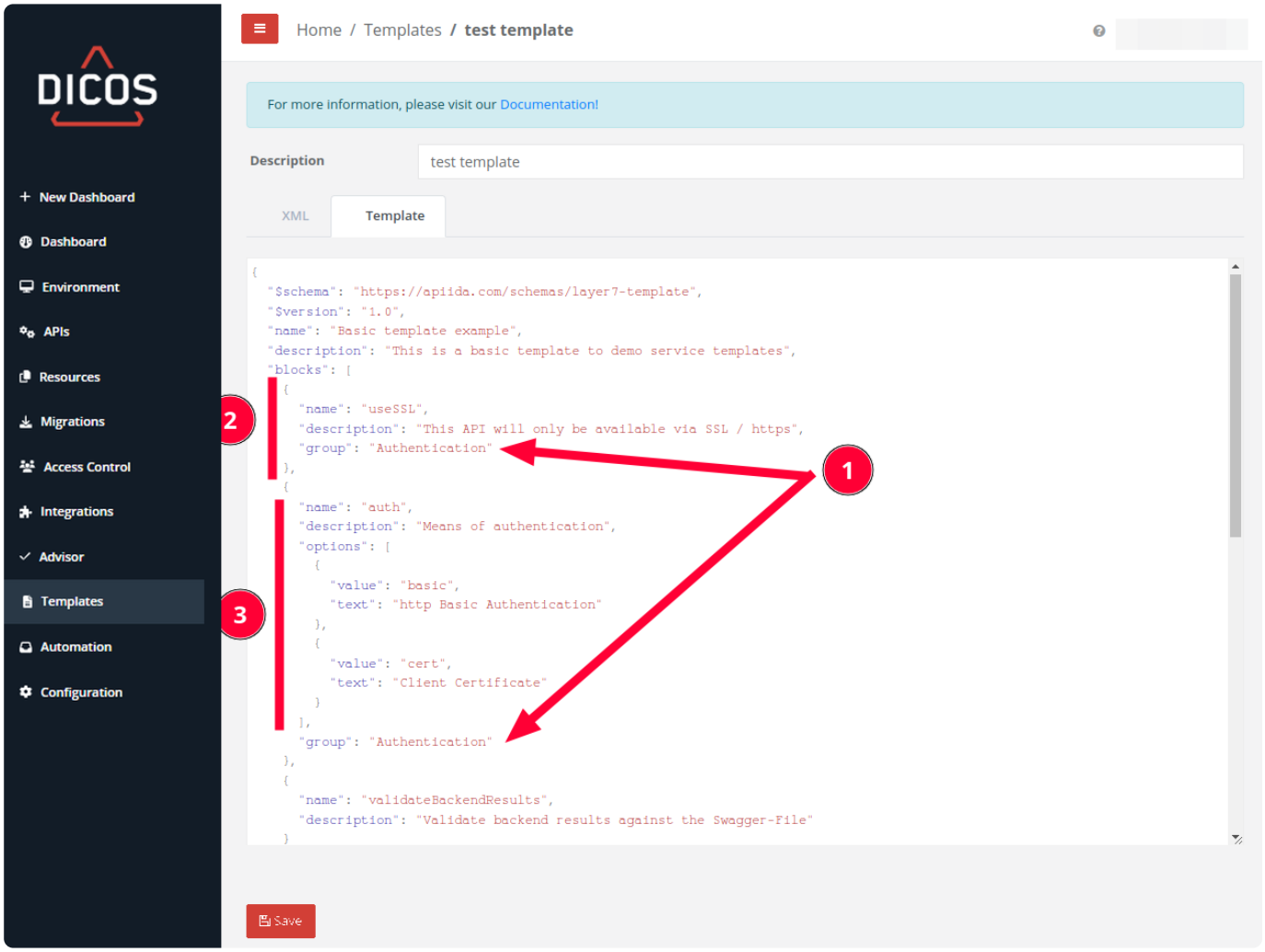
This is the edited XML that will work with the definition above Copy and paste it under the XML tab: :

```
<wsp:Policy xmlns:L7p="http://www.layer7tech.com/ws/policy" xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy">
    <wsp:All wsp:Usage="Required">
        <L7p:CommentAssertion>
            <L7p:Comment stringValue="{{comment}} {{date}} {{someElement}}"/>
        </L7p:CommentAssertion>

        <!-- {{useSSL}} -->
        <L7p:CommentAssertion>
            <L7p:Comment stringValue="SSL"/>
        </L7p:CommentAssertion>
        <!-- {{/useSSL}} -->

        <!-- {{auth=basic}} -->
        <L7p:CommentAssertion>
            <L7p:Comment stringValue="Auth BASIC"/>
        </L7p:CommentAssertion>
        <!-- {{/auth=basic}} -->

        <!-- {{auth=cert}} -->
        <L7p:CommentAssertion>
            <L7p:Comment stringValue="Auth CERT"/>
        </L7p:CommentAssertion>
        <!-- {{/auth=cert}} -->

    </wsp:All>
</wsp:Policy>
```

5.14.4 In DAGM

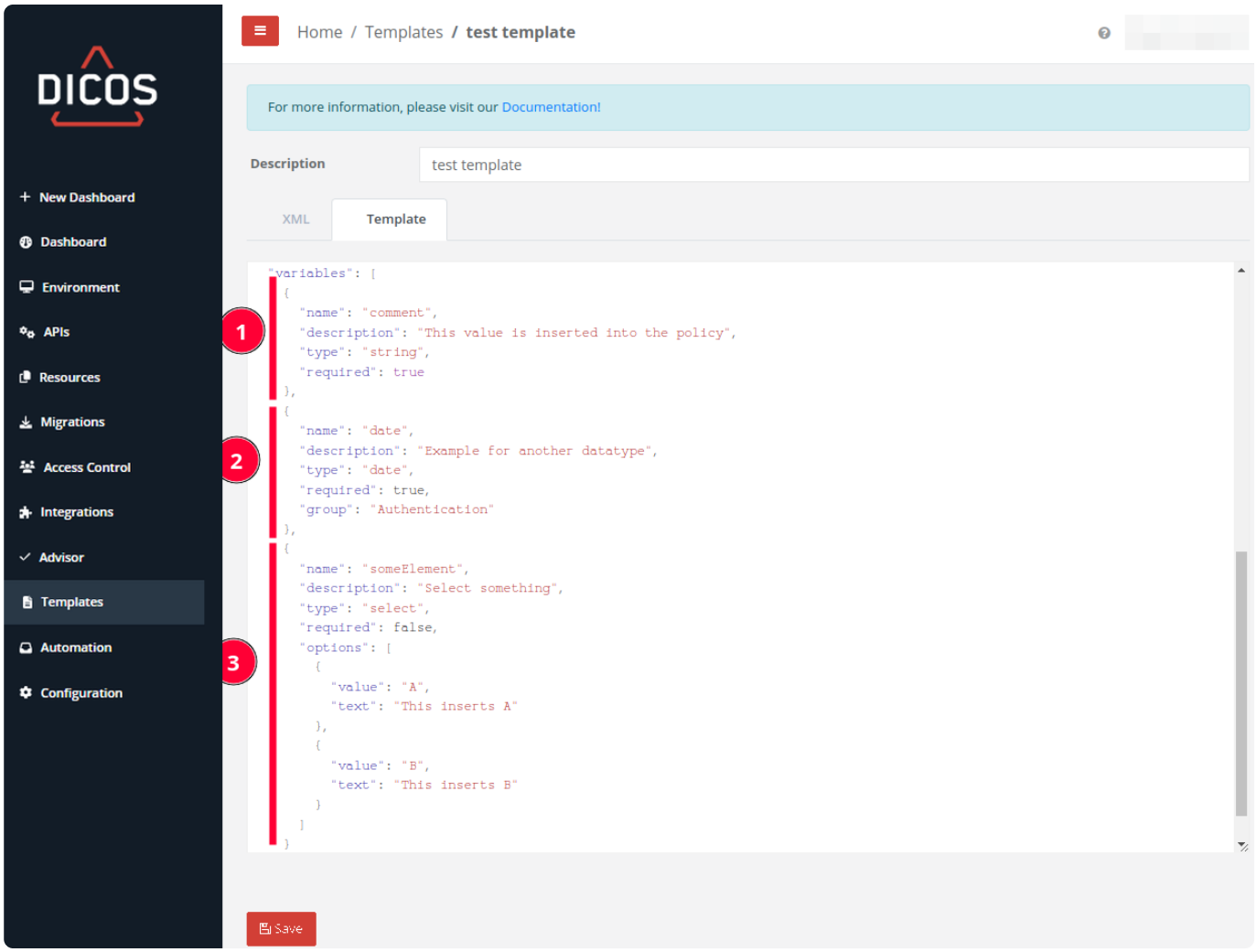
XML Template example

This is what it would look like in the Gateway Manager, navigate to Templates → New Templates and copy/paste the above logics.

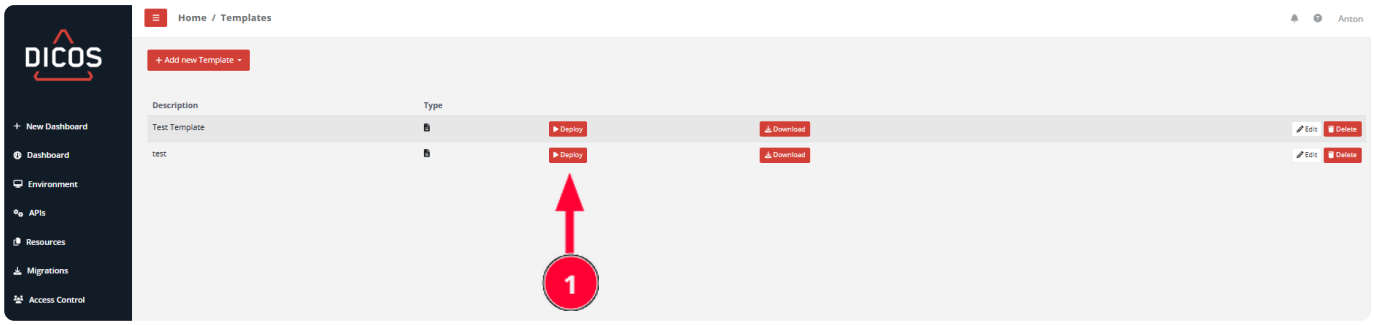


1. These are variables, those will be substituted with the values that you set for them in the template GUI
2. This enclosure makes sure that the inside XML block will be published in the logic on the Gateway only if the user will choose that variable in the GUI
3. This is an if-else case, were one can choose the logic to include from a dropdown.

JSON Template example



1. Determines the group, this is to group together functionalities in the GUI
2. This is the option that relates to point 2 in the section above (the XML)
3. This is the options list that relates point 3 in the section above (the XML)

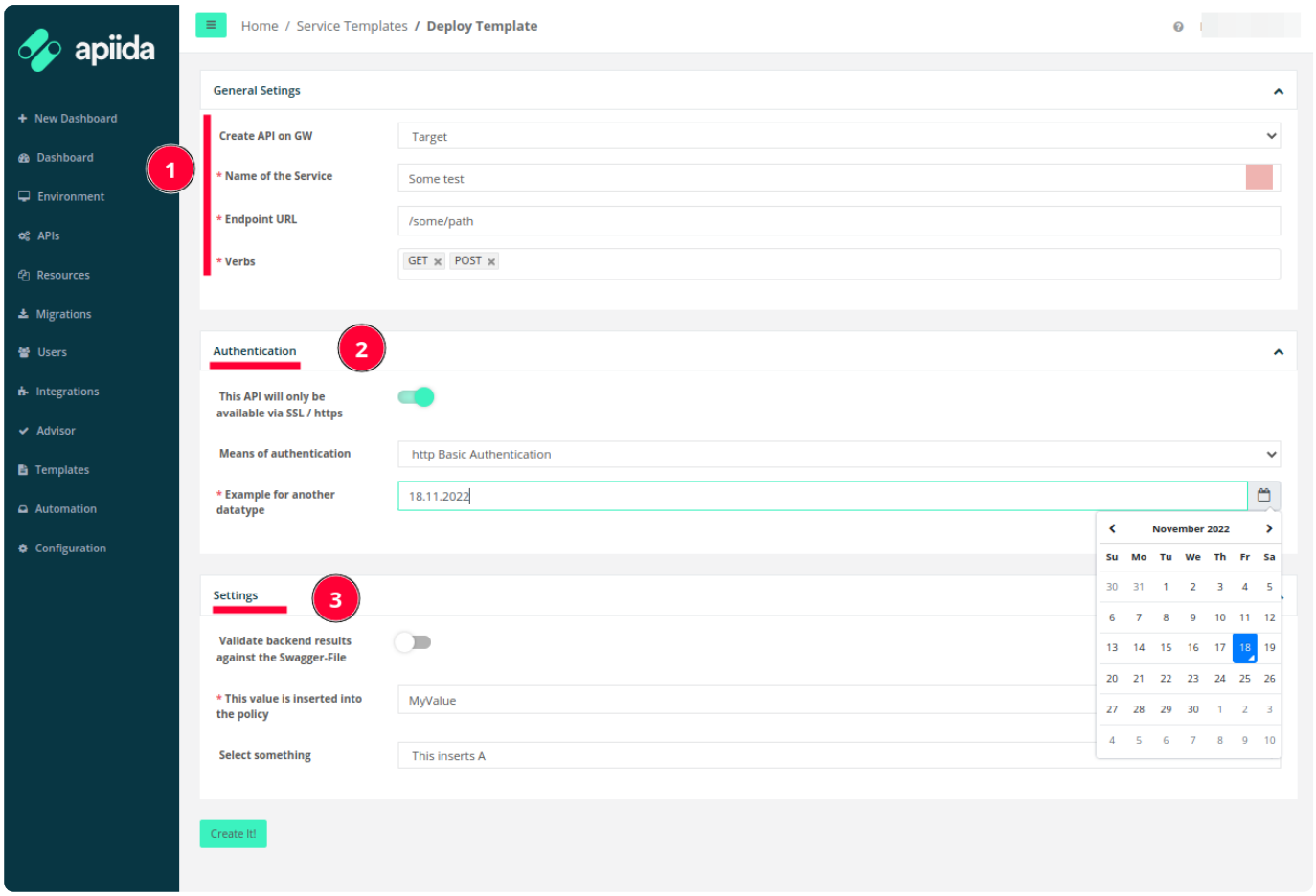


1. An example of how to request a variable input that is a string on the GUI, that will be then inserted in the placeholder described in step 1 in the 'XML Template example' section.
2. An example of how to request a variable input that is going to trigger a calendar selection on the GUI, that will be then inserted in the placeholder described in step 1 in the 'XML Template example' section.
   Note: a Group name has been defined, which means that the Date field will appear in the Authentication group.
3. An example of how to add a drop-down with given choices in the GUI, from which users need to choose allowing for set values and not open-ended as the steps here above, that will be then inserted in the placeholder described in step 1 in the 'XML Template example' section.

Result

Once saved here is the entry

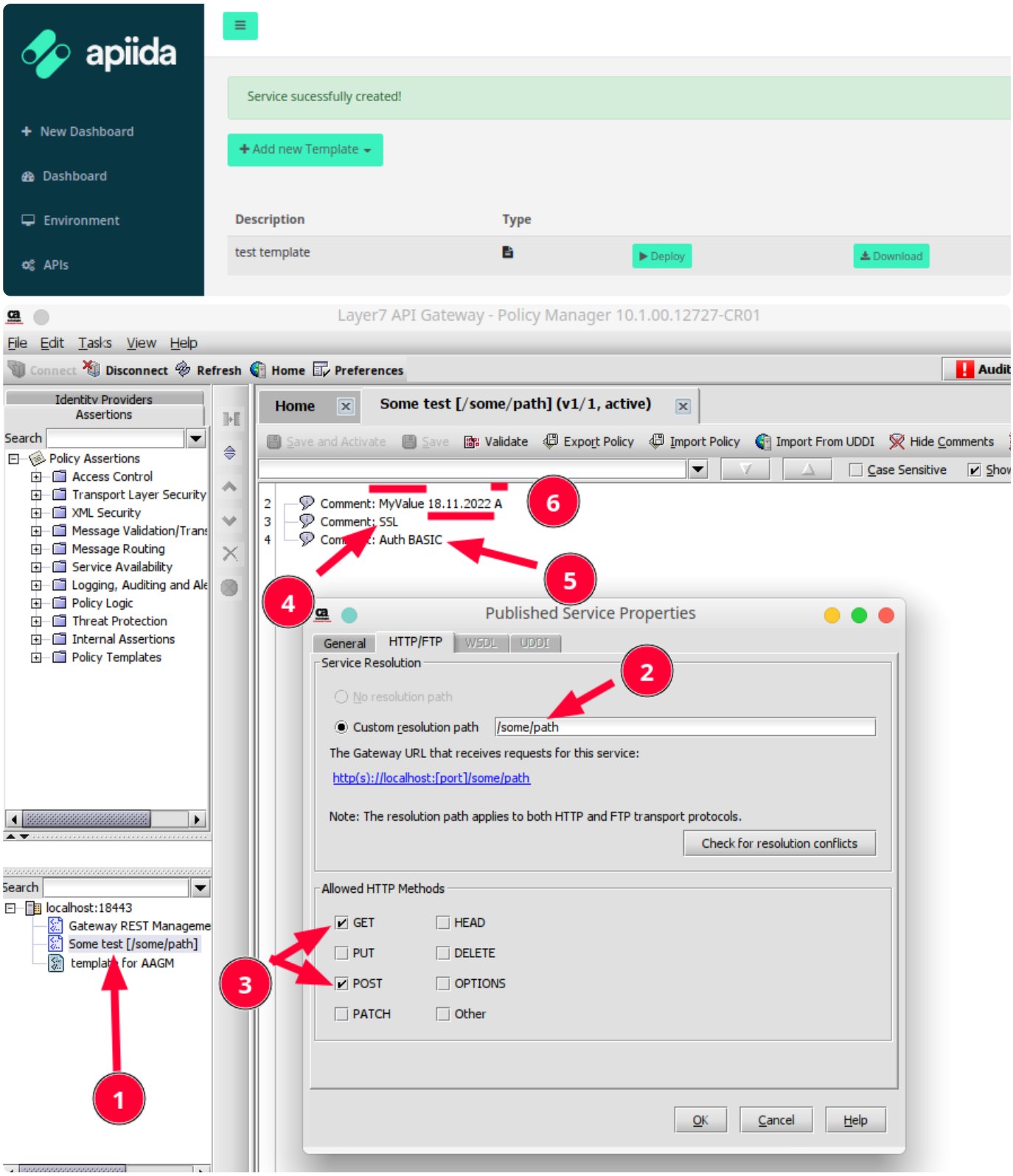
Clicking now on Deploy (1) you will get the following screen



1. Just some general information about the service to be deployed, this is not controlled by the JSON in the Template but is a standard form.
2. That is the group that was defined in the example above.
3. Any non-grouped components are listed in the Settings part, which is a sort-of Else grouping.

Push to Gateway

With the listed choices, the API is now pushed and here it is in the Gateway:



As mentioned, here is the service on the target:

1. The name is the chosen one.
2. The URL Path is also the right one.
3. The Methods (or Verbs) are also the ones chosen.
4. The SSL comment is shown as selected.
5. The Basic Auth is shown as selected from the drop-down.
6. The variables are all rewritten with the choices made in the GUI.

5.15 Users

The API Gateway Manager features an internal identity provider, allowing users within this system to be assigned to multiple permission groups. These groups enable access to specific functionalities within the API Gateway Manager. This approach, known as Role-Based Access Control (RBAC), is instrumental in granting users tailored access levels to the system, ensuring they have permissions that align with their needs and responsibilities. Importantly, the enforcement of these access controls is consistently applied both in the user interface and within the management API, providing a unified and secure approach to managing user access and permissions.

5.15.1 Create User

Internal Users and API Users

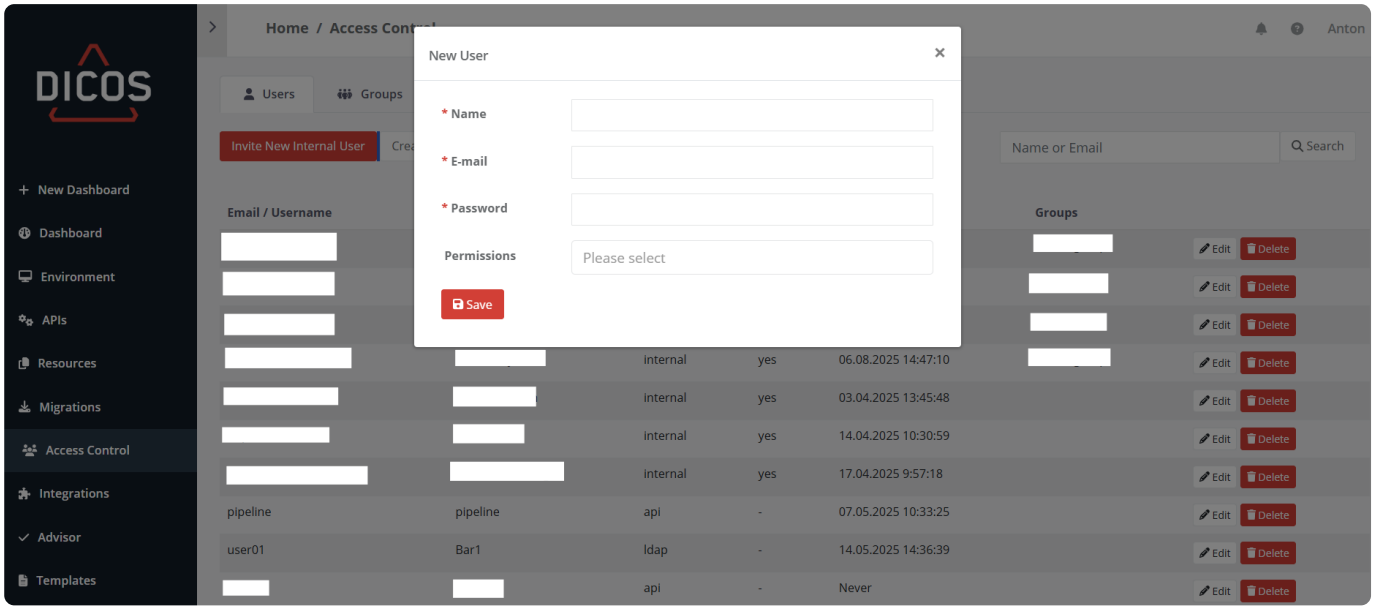
To create a user in the API Gateway Manager, first click Access Control in the left sidebar. In the Users toolbar, you will find three options:

- `Invite new Internal User`
- `Create New Internal User`
- `Create New API User`

`Invite New Internal User` sends an email to the specified address, providing a link for the user to set their own password at their convenience.

By selecting `Create New Internal User` you can create a user account and assign a password directly.

The `Create New API User` function is designed to establish a service account-type user, which uniquely doesn't require an email address for setup. This category of user is typically utilized for programmatic machine access through the Gateway Manager's REST API. Notably, these API users are not equipped with the capability to log in via the Web User Interface (UI), aligning with their intended use for backend, automated processes rather than direct user interaction.
This type of user can be assigned specific roles, and it's important to clearly define the environments to which it has access. This precise specification ensures appropriate security measures and operational efficiency, tailoring the user's capabilities to the necessary environments only.



Please note that you need to have a SMTP server configured in order to use the “Invite User” feature.

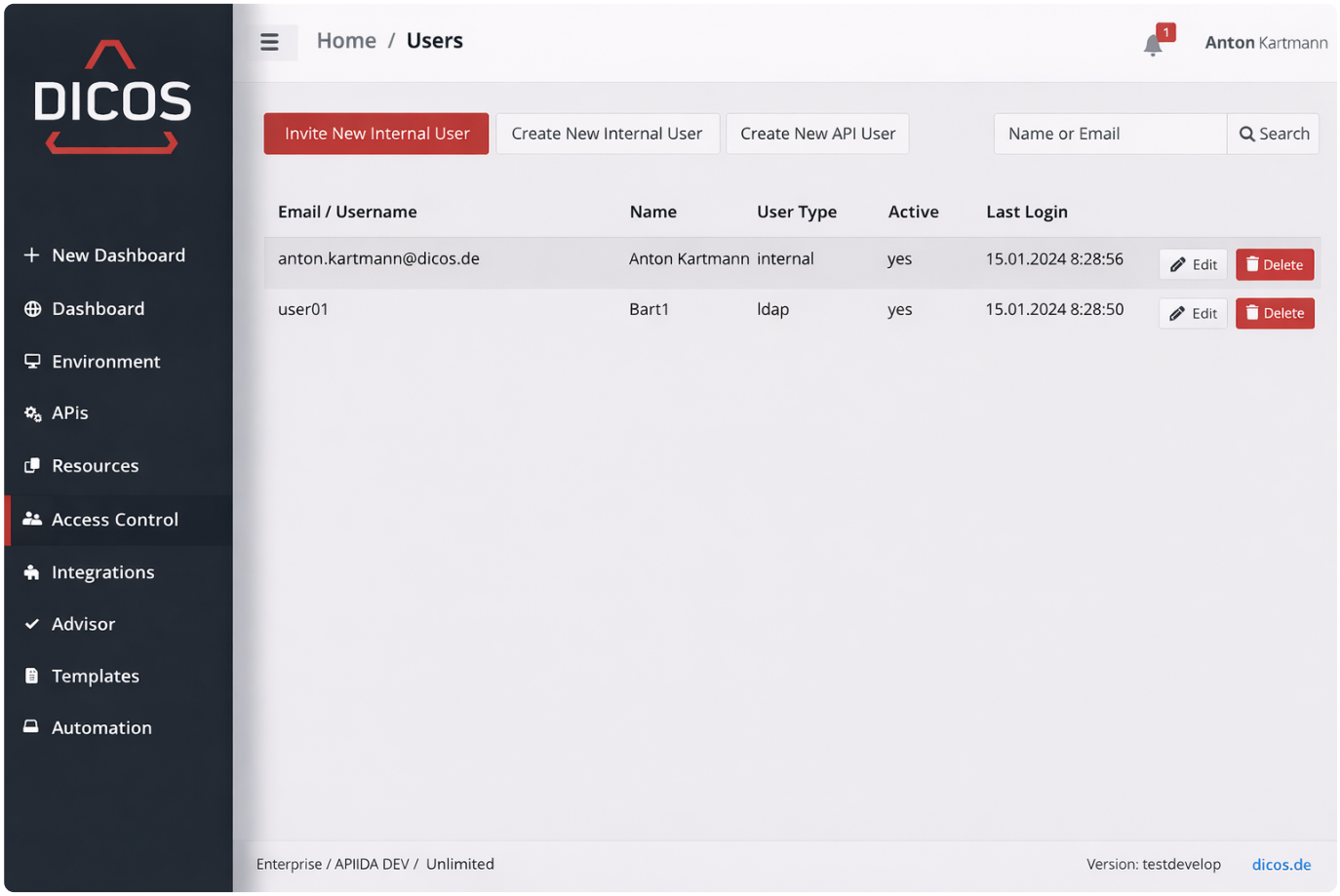
Both user creation wizards prompt you to select an initial permission group for the new user. Subsequently, you have the flexibility to add more permission groups or modify the original one through the user details view. This approach provides initial access control while allowing for adaptable permissions management as user roles evolve or require adjustments.

SAML and LDAP User

If LDAP or SAML authentication is set up within the Configuration → Authentication page, users can access the Gateway Manager using their standard LDAP or SAML credentials, eliminating the need to create user accounts in the Gateway Manager beforehand. However, for security purposes, these users are not automatically assigned any roles upon login. An administrator must explicitly grant the necessary permissions to each user after the user's initial login.

Typically, a user's LDAP username corresponds to the "User name attribute" or 'cn' (common name), not their email address.

Ensure not to create any user accounts in the User Interface (UI) before the individual has logged in using their LDAP credentials.

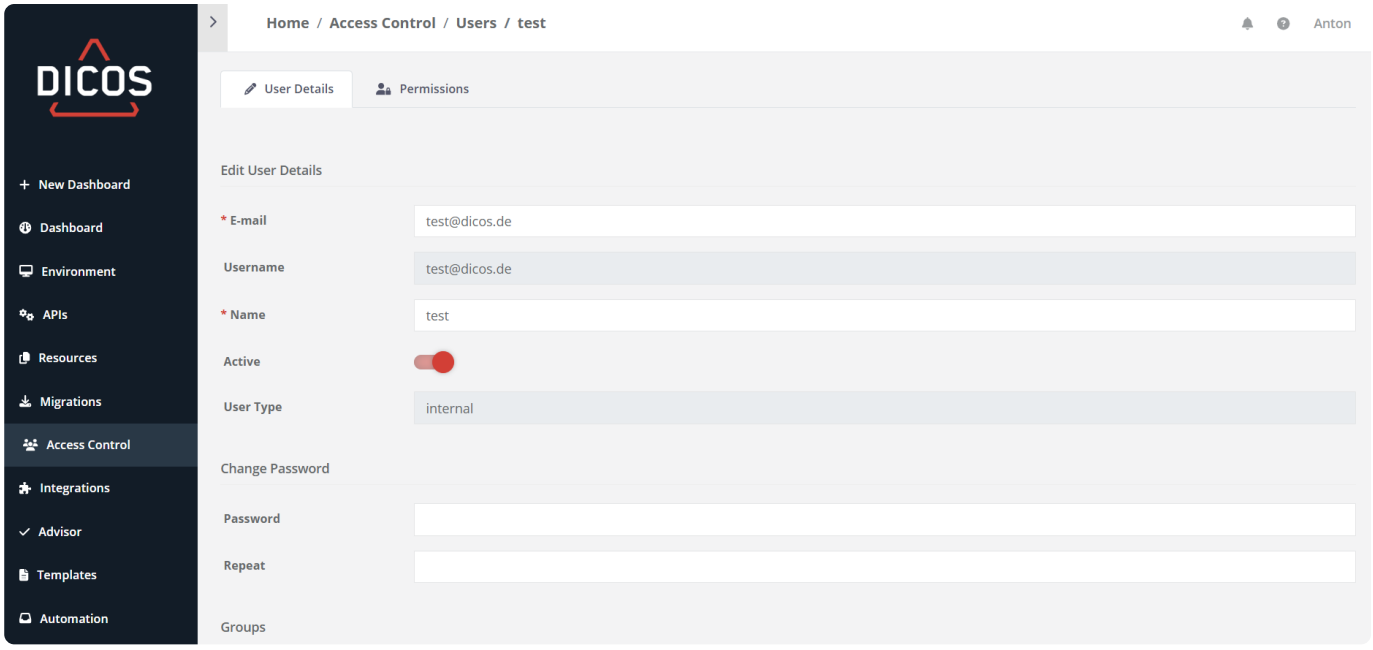


5.15.2 Permission groups (RBAC)

There are 8 different permission groups.

- Administrator
  - This permission group is like a superuser group. It has access to every part of the system. The first user that was created during the Installing process of API Gateway Manager has this permission group preset.
- Auditor
  - This permission group has access to all logs of API Gateway Manager. It can audit them to check for security issues, problems with migrations or any other problems.
- Manage Nodes
  - This permission group has access to all node and environment related functions. It can create new single nodes or cluster to the API Gateway Manager as well as edit those. It can also monitor all nodes on a dashboard to check for availability of the nodes and the status of their services.
- Manage APIs
  - This permission group can import and add new APIs as well as edit and configure them. It can add stage variables to APIs, configure alarms and configure the use of Git for APIs. This role also has access to libraries and solution kits.
- Manage Users
  - This permission group is in charge of all users registered in API Gateway Manager. Users with this group can edit other users to give them more or less permissions, they can edit there email address or password and they can deactivate unused user accounts.
- Operator
  - This permission group has access to dashboards that were made available and to errorlogs. This permission group is perfect for creating a technical user that is used on a beamer/projector or monitor so everyone can watch the real time stats
- Perform Migrations
  - This permission group is in charge of performing a migration. Users who can prepare migrations can assign people with this permission group to perform this migration.
- API Developer
  - This permission group is a subgroup of Manage Services. This permission group can only pull a service from git and prepare migrations as well as assign them to user with a Perform Migrations permission group.

5.15.3 Edit a User



Edit User Details

- The email is used as the user name for internal users. For other user types, it is taken from LDAP or SAML. Even if it is not the email.
- The user's name is displayed in the top right-hand corner and can be freely selected.
- Internal users can be locked out of the AAGM by setting the active flag to false.
- The user type can be freely changed at the bottom with "Change User Type".
    - When changing the user type to LDAP or SAML, the current password is removed and the user authenticates from now on only against LDAP or SAML.

Change Password

The password can only be set for internal and api users.

Permissions

The roles can be freely added to the user here. A user can also have several roles. For some roles, it must be specified which environments the user may access.

RBAC environmental settings

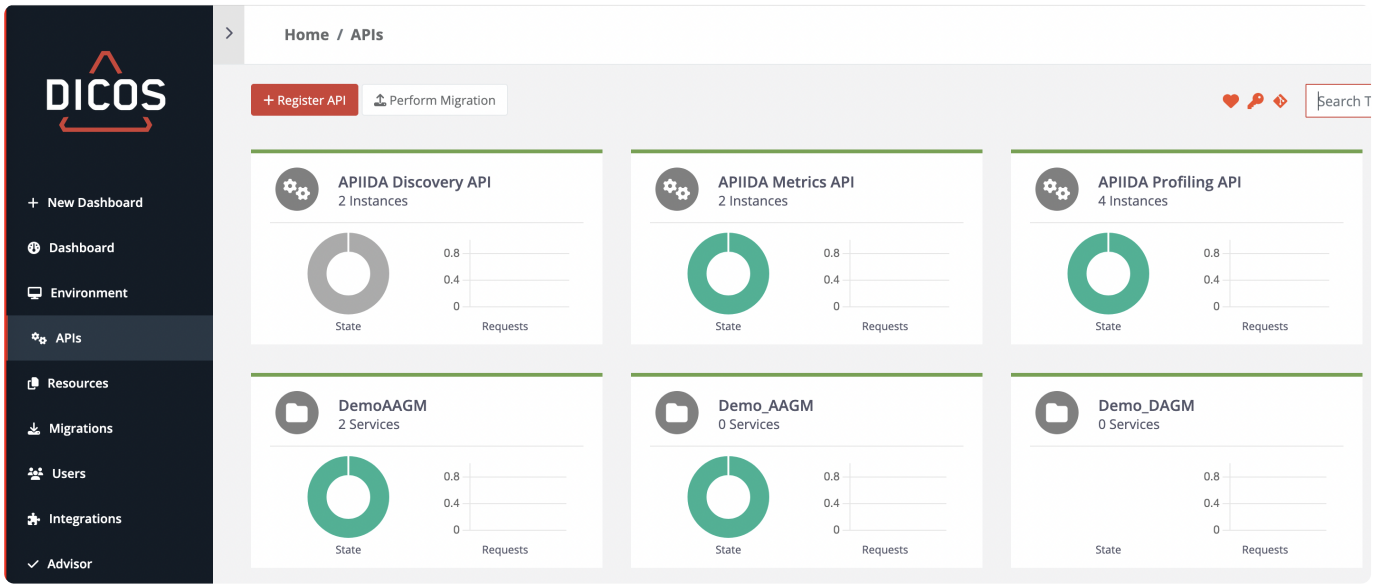
For users not assigned to the administrator permission group, it's essential to specify the environments they can access. Each environment has two distinct permissions that can be assigned: Read and Write. A user with Read permission can view every node and cluster within the specified environment, as well as their statuses. On the other hand, a user granted Write permission has the ability to perform actions corresponding to their assigned permission groups within that environment. This dual-level permission structure allows for both visibility (Read) and operational control (Write), based on the user's role and the permissions granted.

5.16 APIs

5.16.1 APIs

Clicking on the item APIs in the left hand menu, takes you to the list of all APIs and libraries. The list can be viewed with two different settings. You can either choose a list mode for a condensed table-like list or you can choose a more visual tile based grid, which includes the most important metrics.

APIs, Folders and Instances



The list only shows the APIs that have been imported into the Gateway Manager.

Each tile has an icon, to the right of it the corresponding name and the number of services/APIs/instances it contains and below it the merged metrics of all the elements it contains.

-  The folder icon represents a folder, a group of APIs that is viewed as a unit.
-  The gear icon represents an API that has been registered without its folder or it is located directly in the root folder.
-  A library is collection of resources.
-  Solution Kit

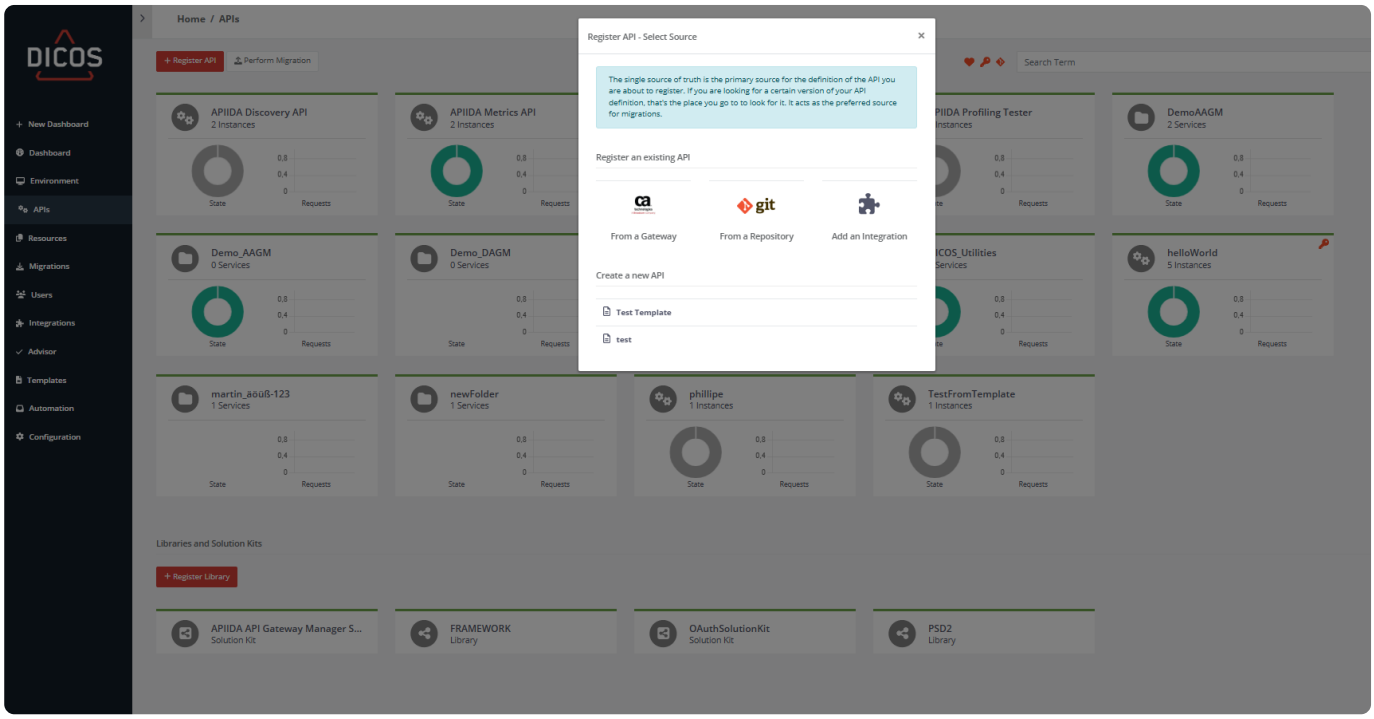
If you want to look at a complete list of APIs running on a gateway please use the resource explorer, available via the item ‘Resources’ in the left-hand menu.

Resources

Register API

Here you can import your API from your gateway, from git or from a template.

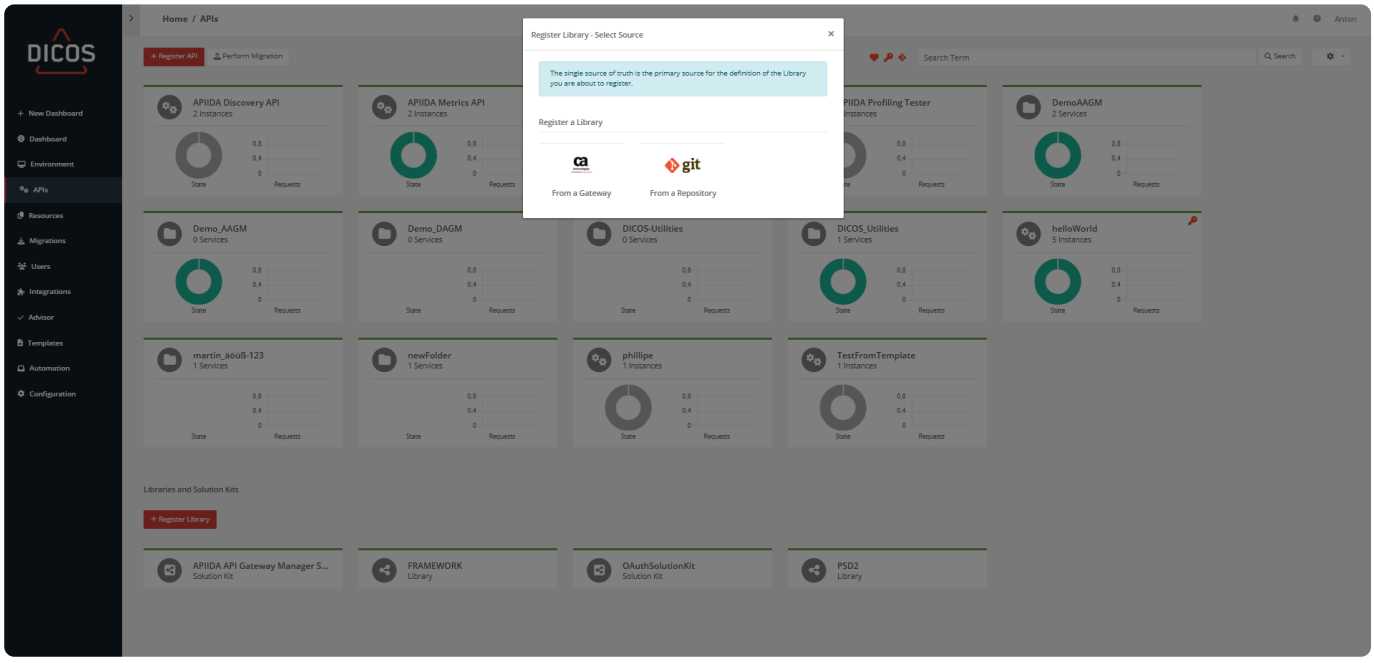
Work with git Service Templates



Register Library

Already existing libraries can be registered here from git. New ones can be created from the node.

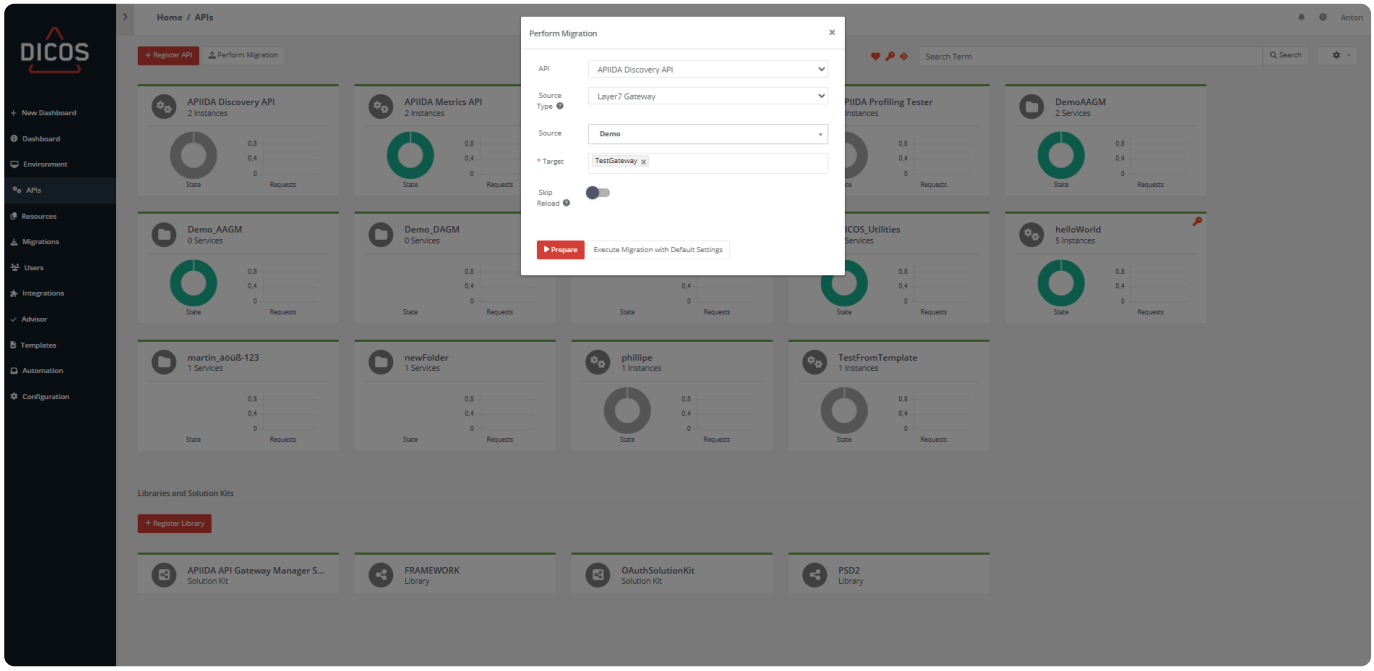
Libraries and Solution Kits



Perform Migration

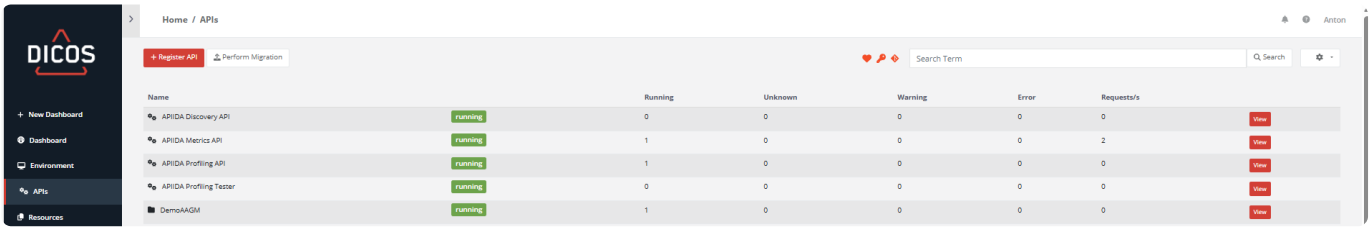
Right here in the overview you can also start a migration. Select source and targets and you are ready to go.

Perform a Migration



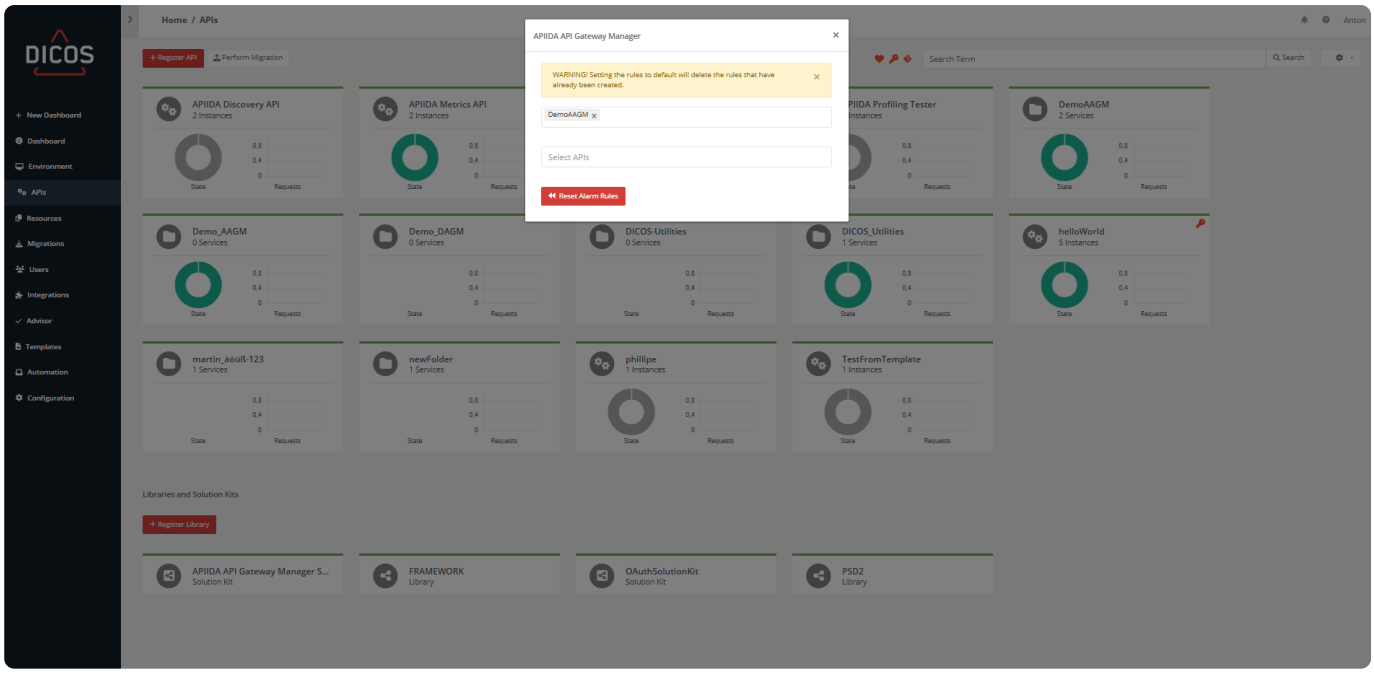
List view or Tile view

You can also view your APIs as a list. At the top right you can change the view with the gear wheel.



Reset Existing API

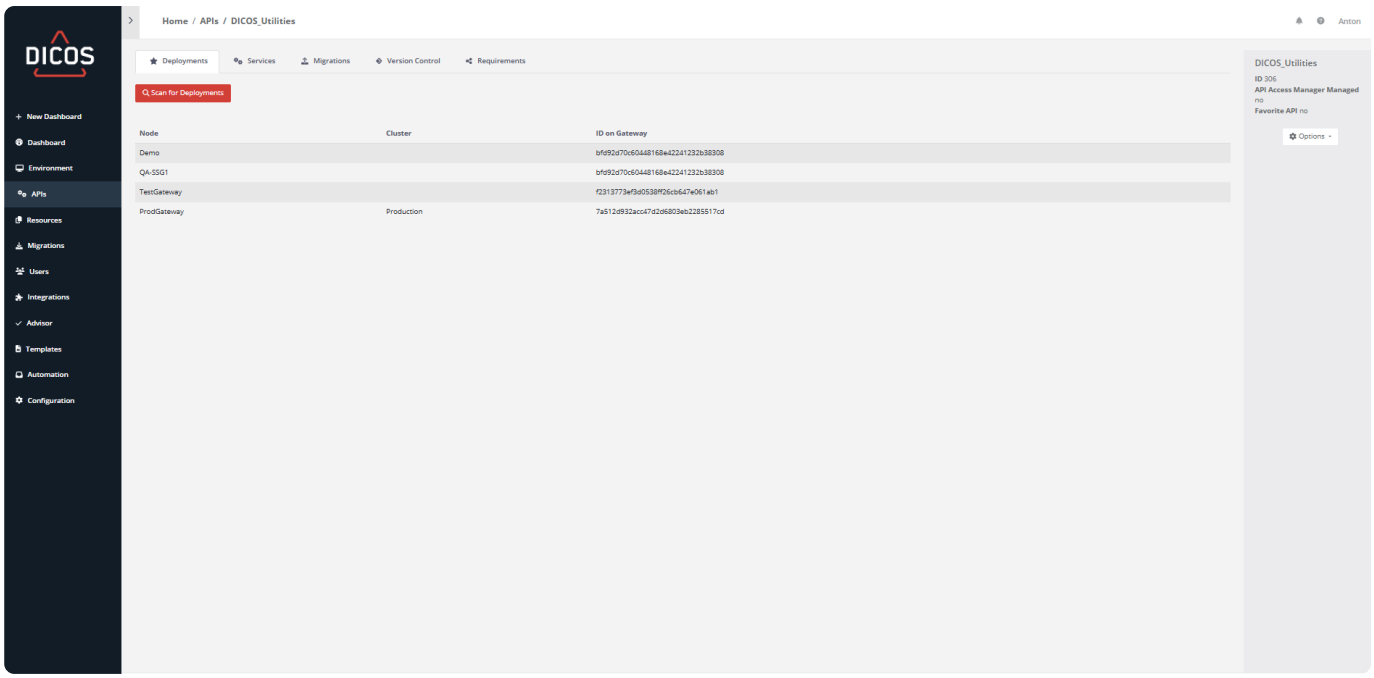
At the top right, you can reset the alerting of an API to its initial values. The initial values can be edited here:



Discover APIs with the DICOS API Access Manager Agent (Formerly Control Plane)

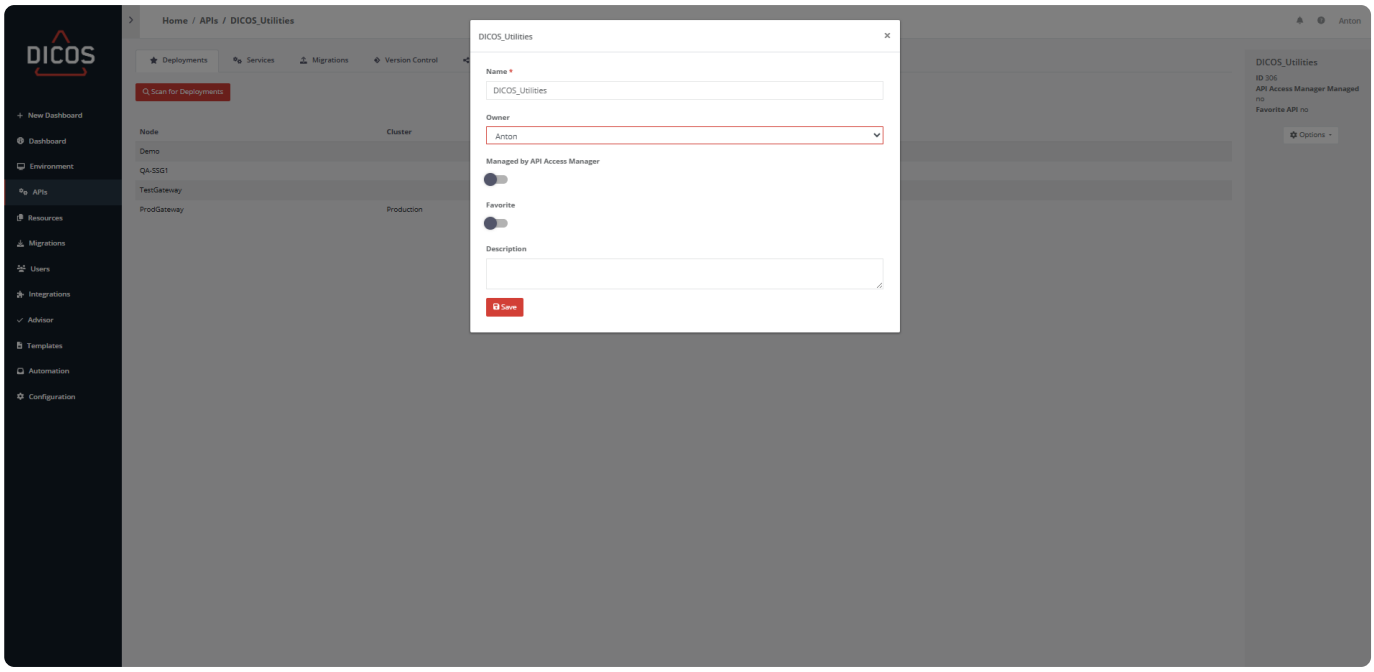
(When Gateway Manager is utilized for directing API Access Manager traffic.)

As of Agent version 11.1, each API must be marked as "API Access Manager Managed". This is possible from Gateway Manager version 2023.2.2.
To do this, the API must be registered in DAGM.



"Managed by DICOS API Access Manager" must be activated in the API options.

If you mark a folder as "managed", all APIs in it are also marked. But you can of course deactivate individual APIs again.

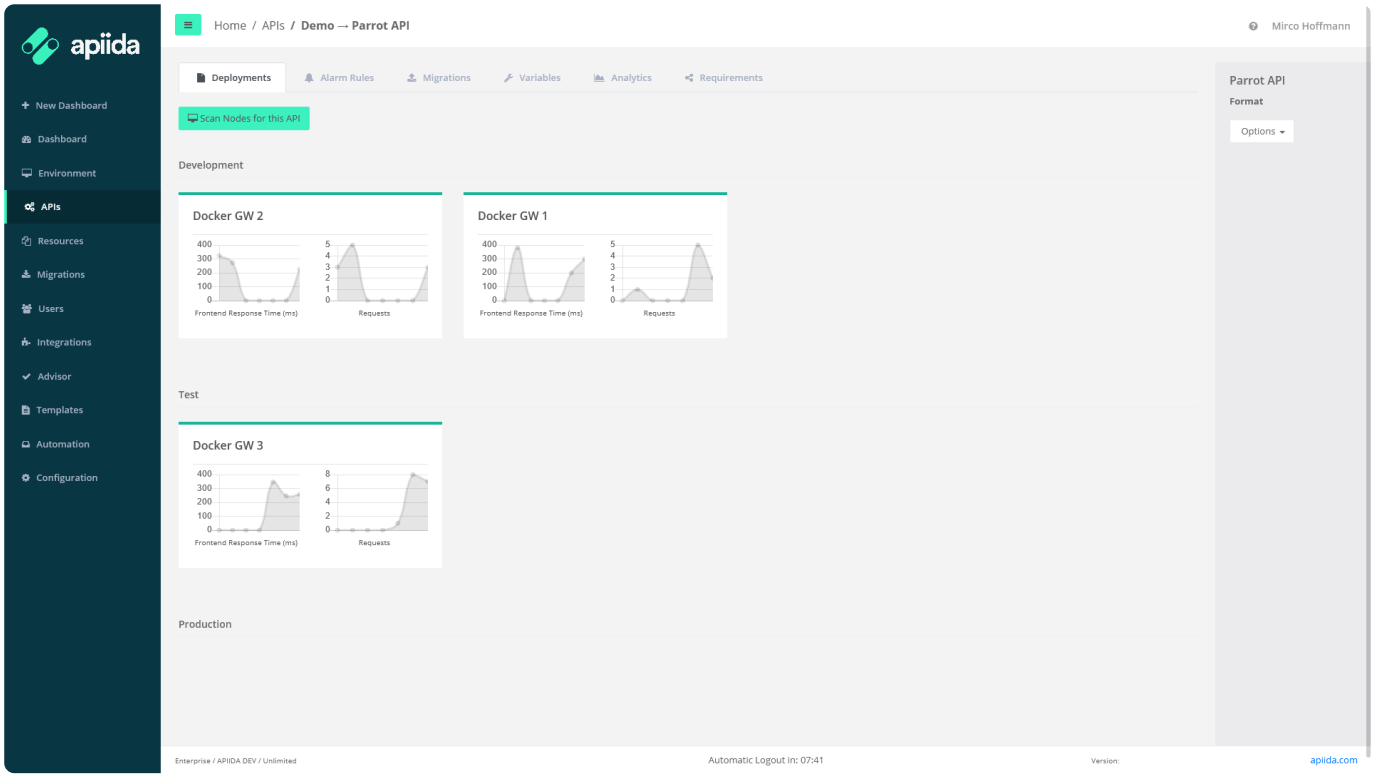


5.16.2 API

In the overview you will see either a folder with APIs or a single API depending on what you have selected in Discover APIs . Therefore, other options will be available to you in some cases. In this case, it is a single API.

Deployments

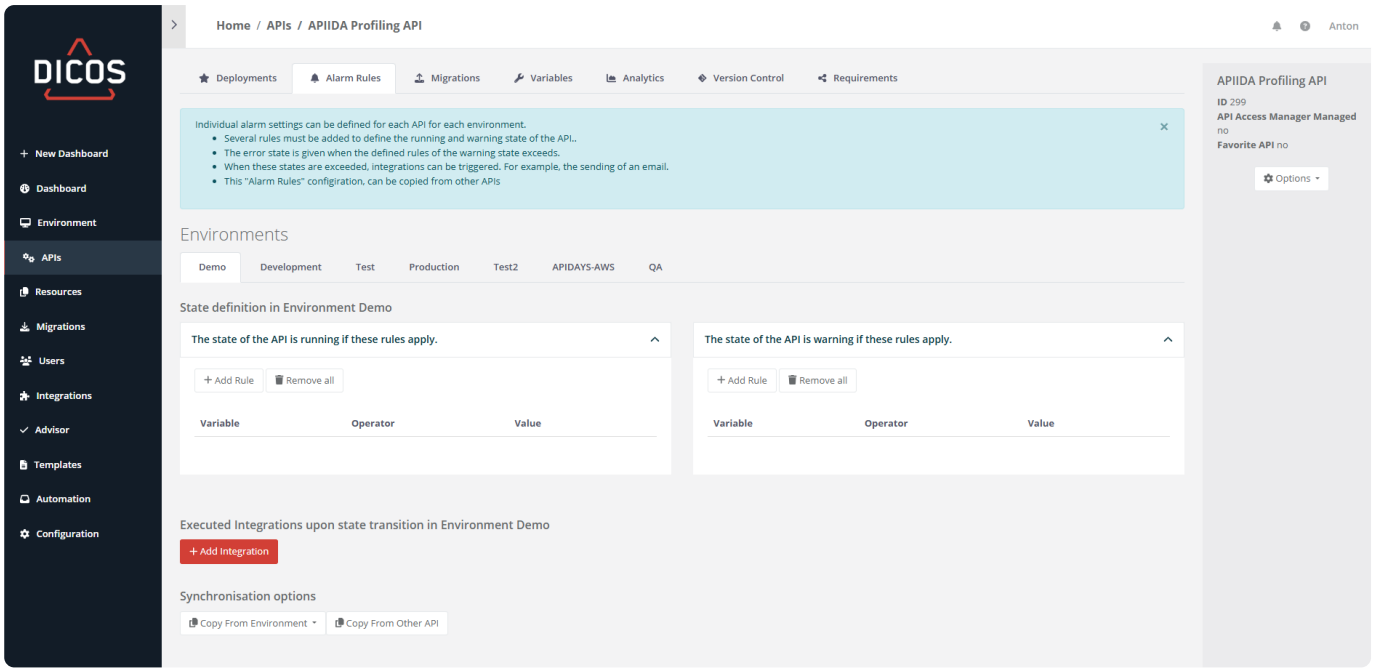
Here you can see all nodes on which your API is available.
With "Scan Nodes for this API" you can find further deployments on gateways that have been installed in DAGM. Clicking on a tile takes you to a detailed view of the metrics on this node.



Alarm Rules

You can edit the initial values here and reset to them for each API here.

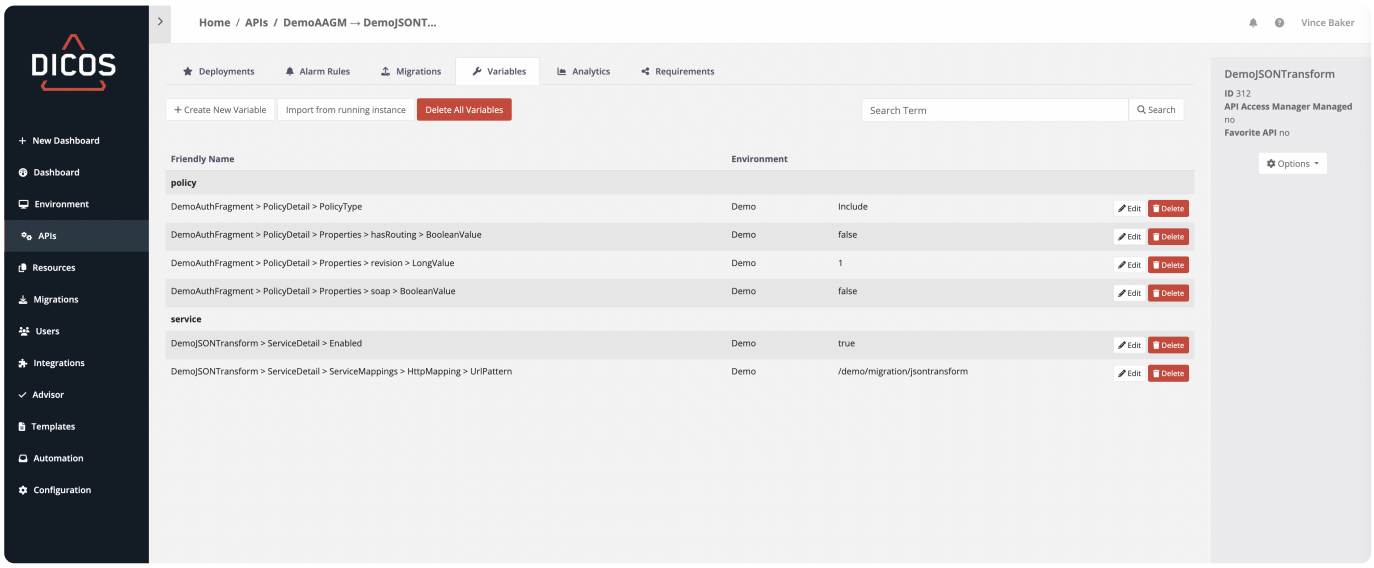
If you want to define certain rules for an API that do not apply to others, you can do this here in the API in the state definition.
The error state is given if the defined rules of the warning state are exceeded. If states are exceeded, all integrations that have been configured for this transition from running to warning are triggered. For example, sending an email.



Variables

Here you can create, edit and delete variables of your API. You can also import them from a running Instance.

Use service variables

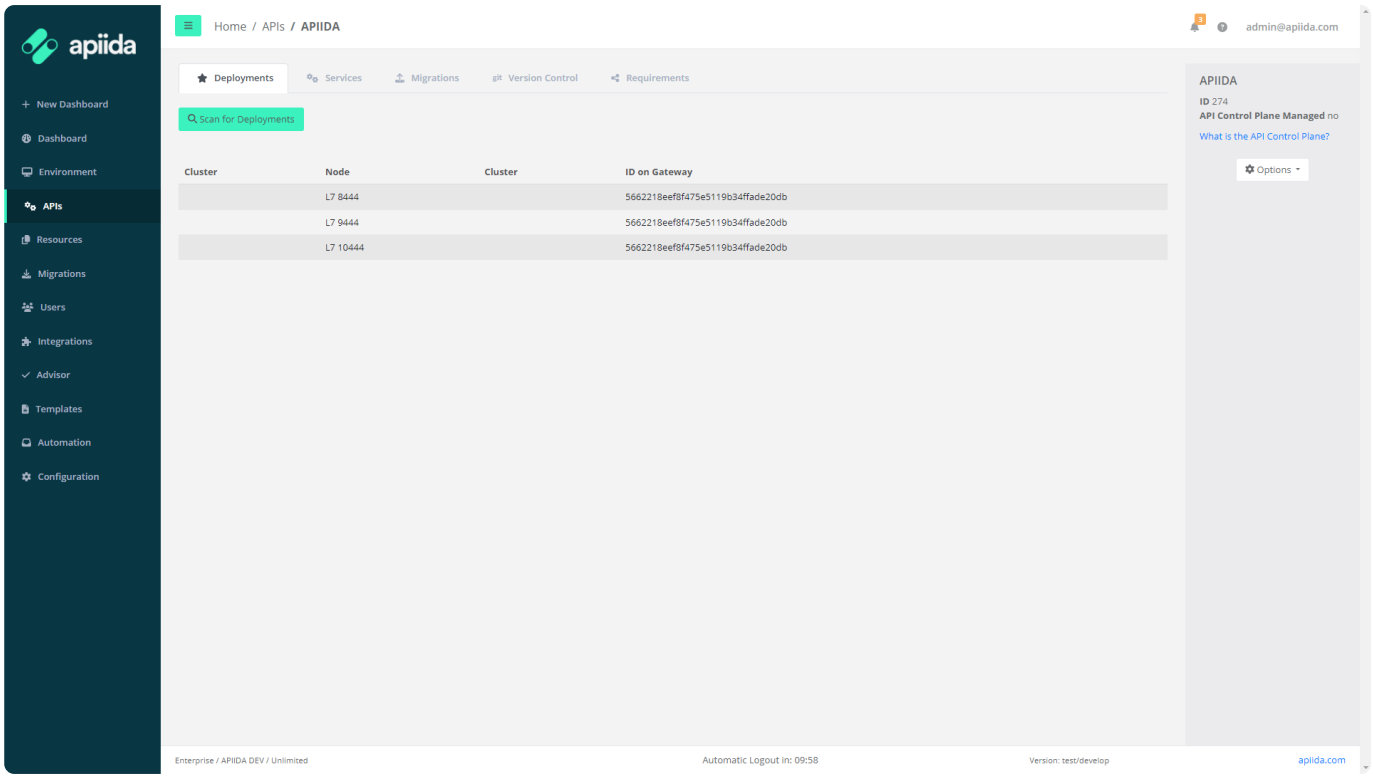


5.16.3 **Folder**

In the overview you will see either a folder with APIs or a single API depending on what you have selected in Discover APIs . Therefore, other options will be available to you in some cases. In this case it is the Demo folder with three APIs.

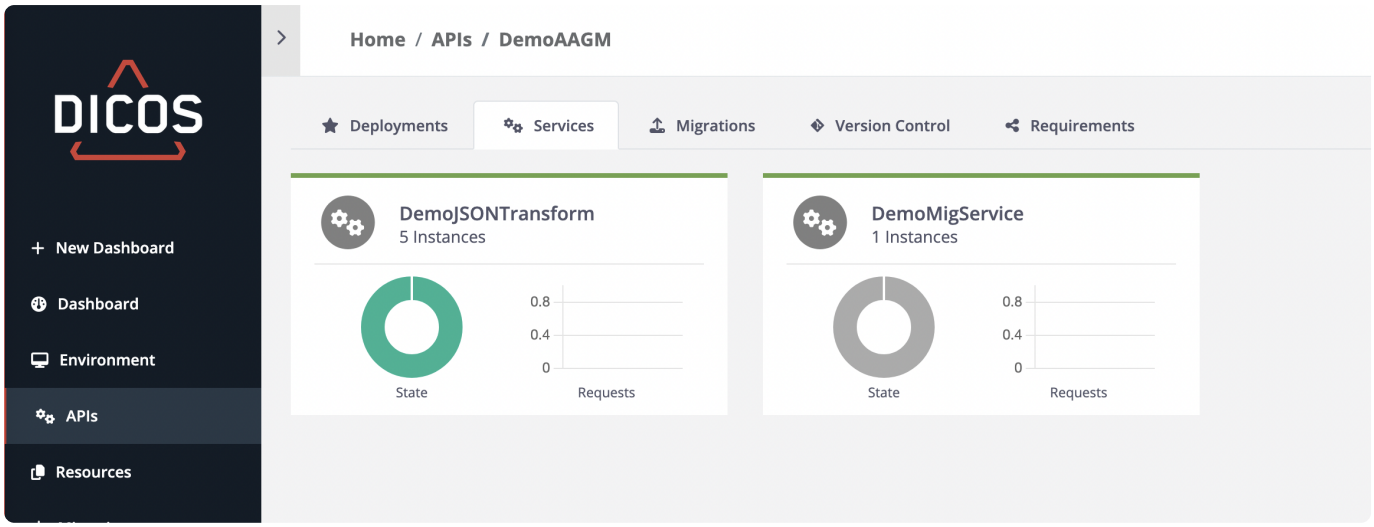
Deployments

All APIs are listed here for each deployment on each node. We also call these specific APIs running on a node an instance of the API. The "Scan for Deployments" button is used to scan the local clones of the nodes and update the list of instances. This happens automatically after each migration or reload of the resources.



APIs

All APIs of the currently selected folder are listed here. In the top left of each tile, you will see the symbol for the API, the name of the API and the number of instances to the right. The metrics of all instances are displayed together below. Click to access the details of this API.

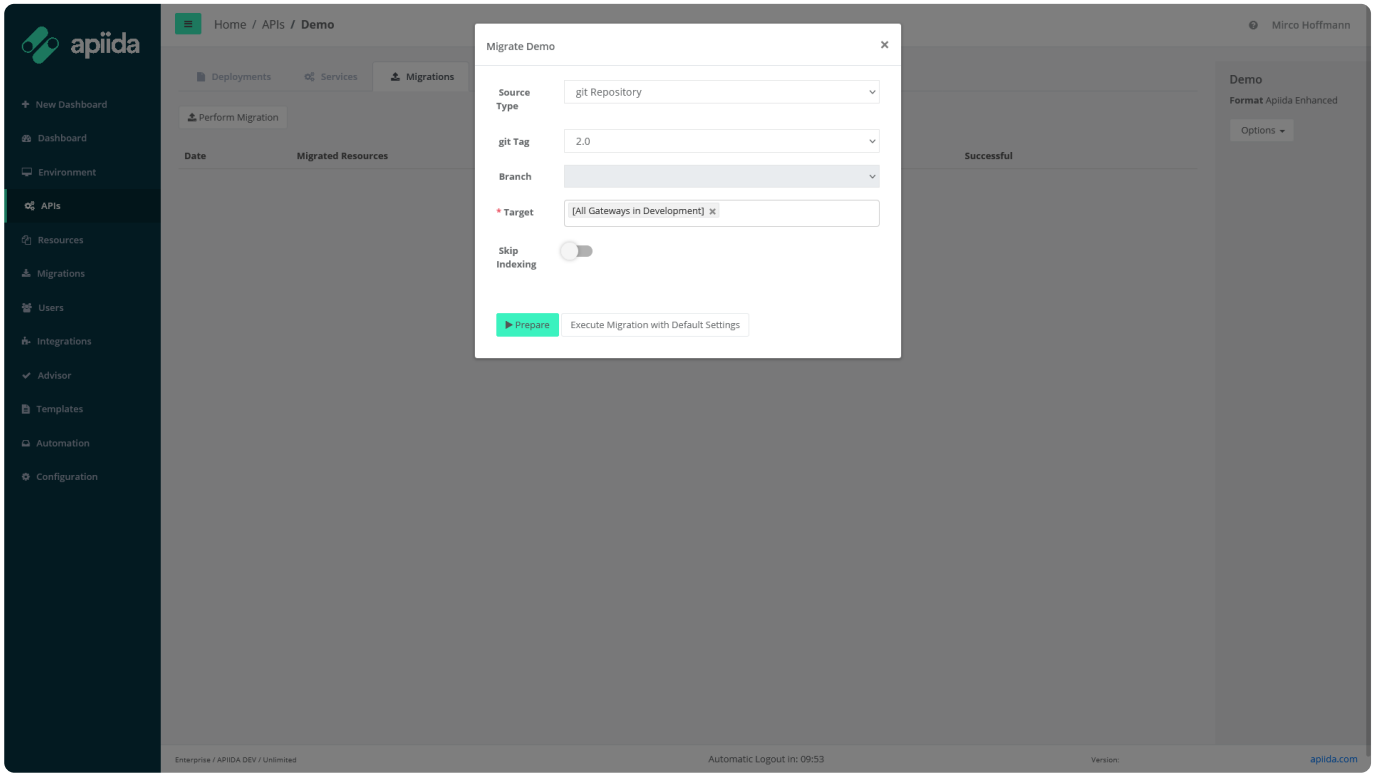


Migrations

Here you can migrate your APIs from git, from a gateway or even from a file to other gateways. You can also view the history of your past migrations

Migrations

Perform a Migration



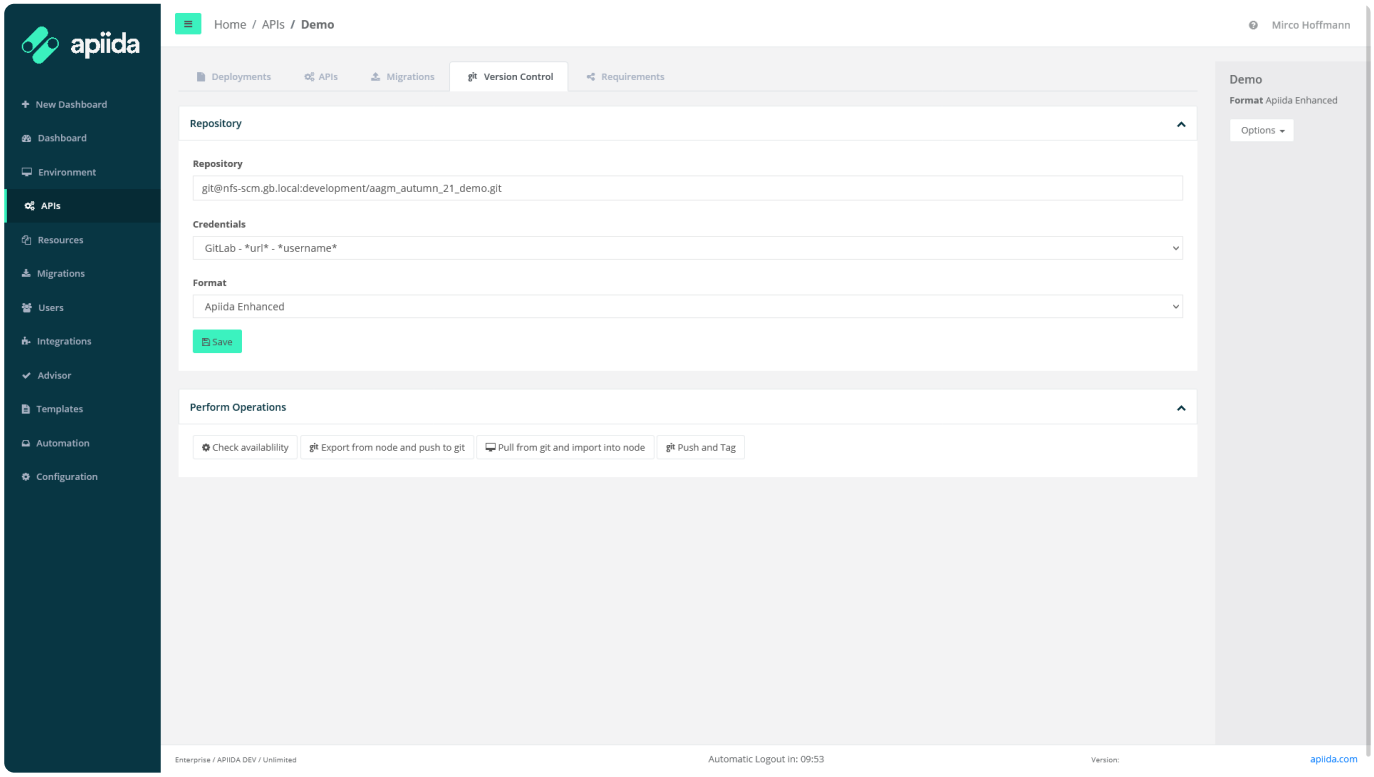
git Version Control

We recommend using git as the single point of truth of your APIs. Here you can check the connection to the git repository and send the source code of the entire folder with all APIs and dependencies to and from git and also version it.

We understand the folders as a thematic unit of the APIs, so it is not possible to additionally connect the APIs within the folder with git.

Work with git

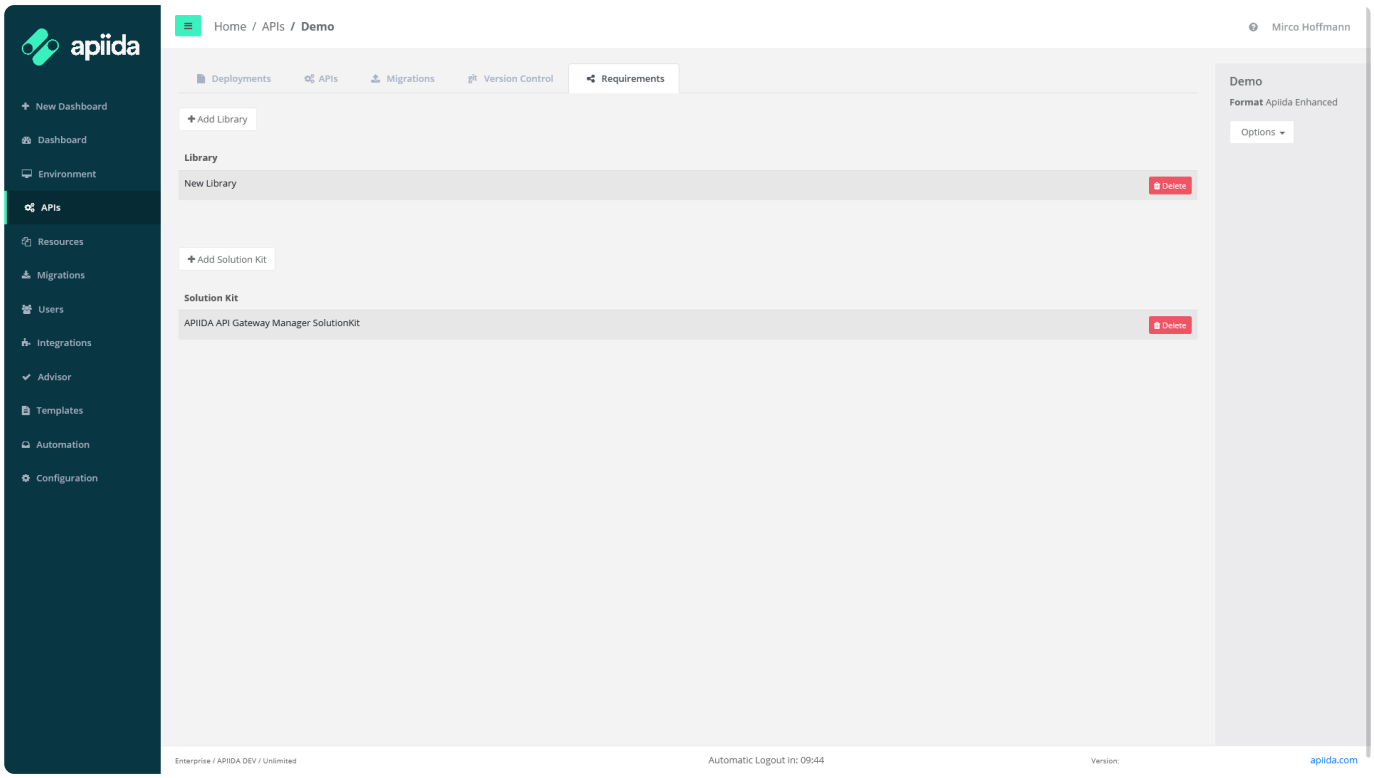
Storage Formats



Requirements

If your APIs require a solution kit or additional resources in the form of a library, you can note this here. During a migration, the system will then automatically check whether the resources and solution kits are available on the target.

Libraries and Solution Kits



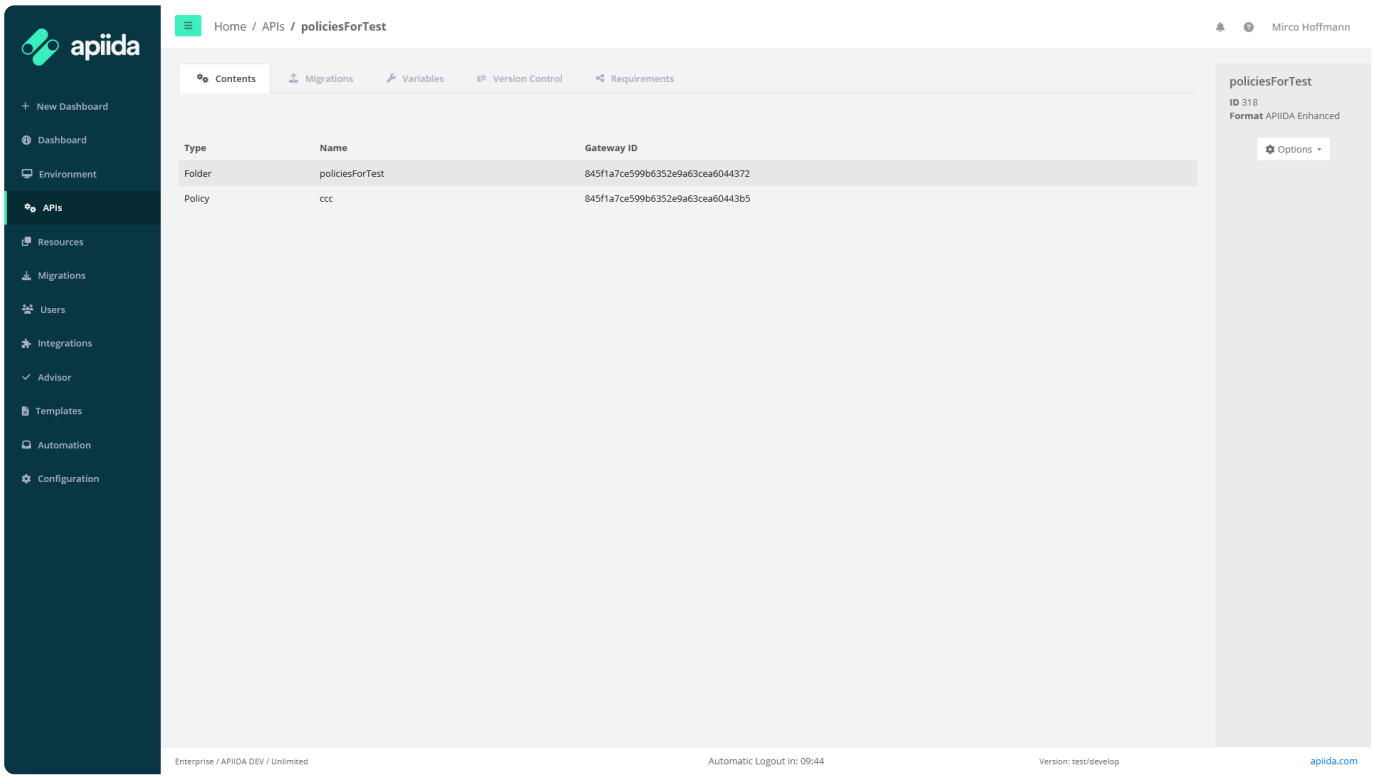
5.16.4 Libraries and Solution Kits

A library is collection of resources. Libraries or solution kits can be added to an API as a requirement. From this point on, when migrating this API, DAGM checks if the requirements are also present on the target.

Here you can see how to create a new library and add it to an API as a requirement.
Work with git

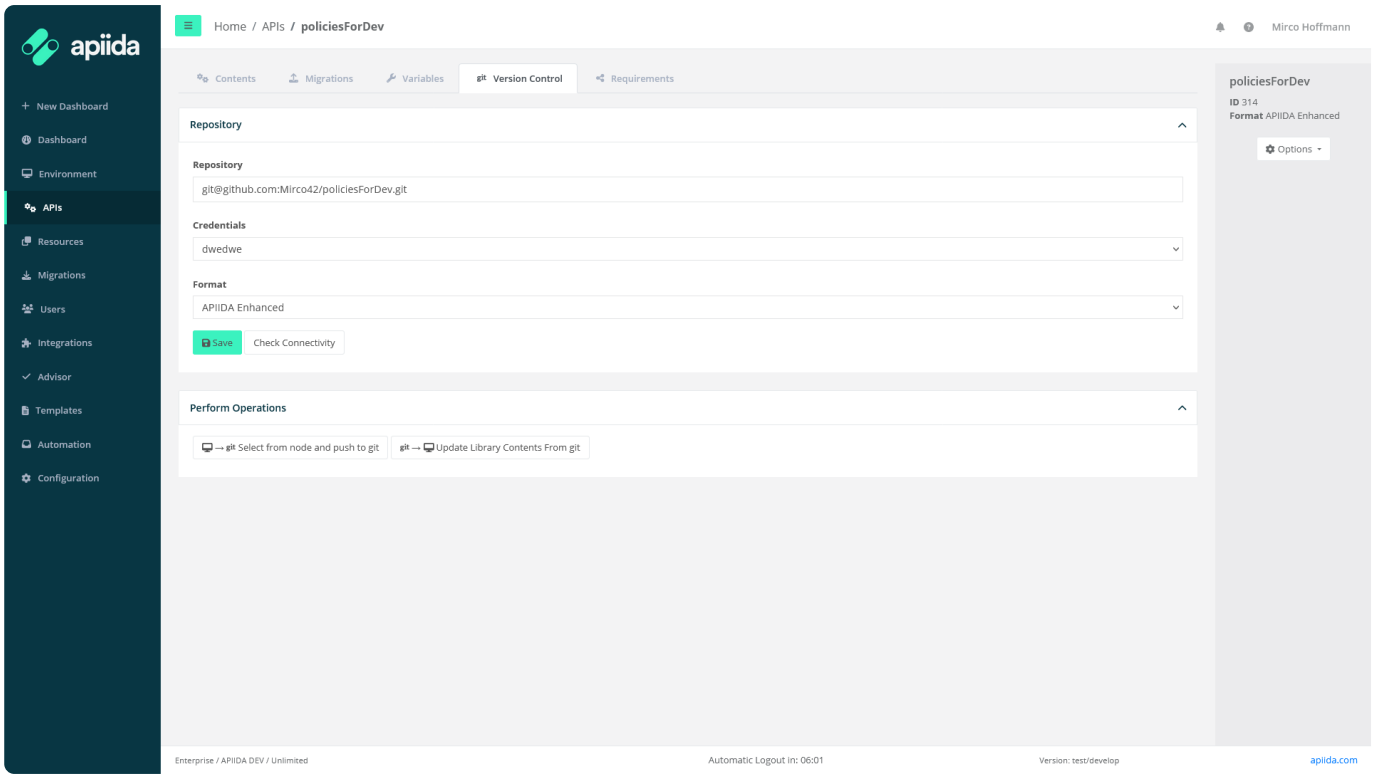
5.16.5 Contents

The contents of the library or solution kit are listed here.



5.16.6 Version Control (Library only)

Here the content can be updated from the git or also from the node.



Update an existing Library

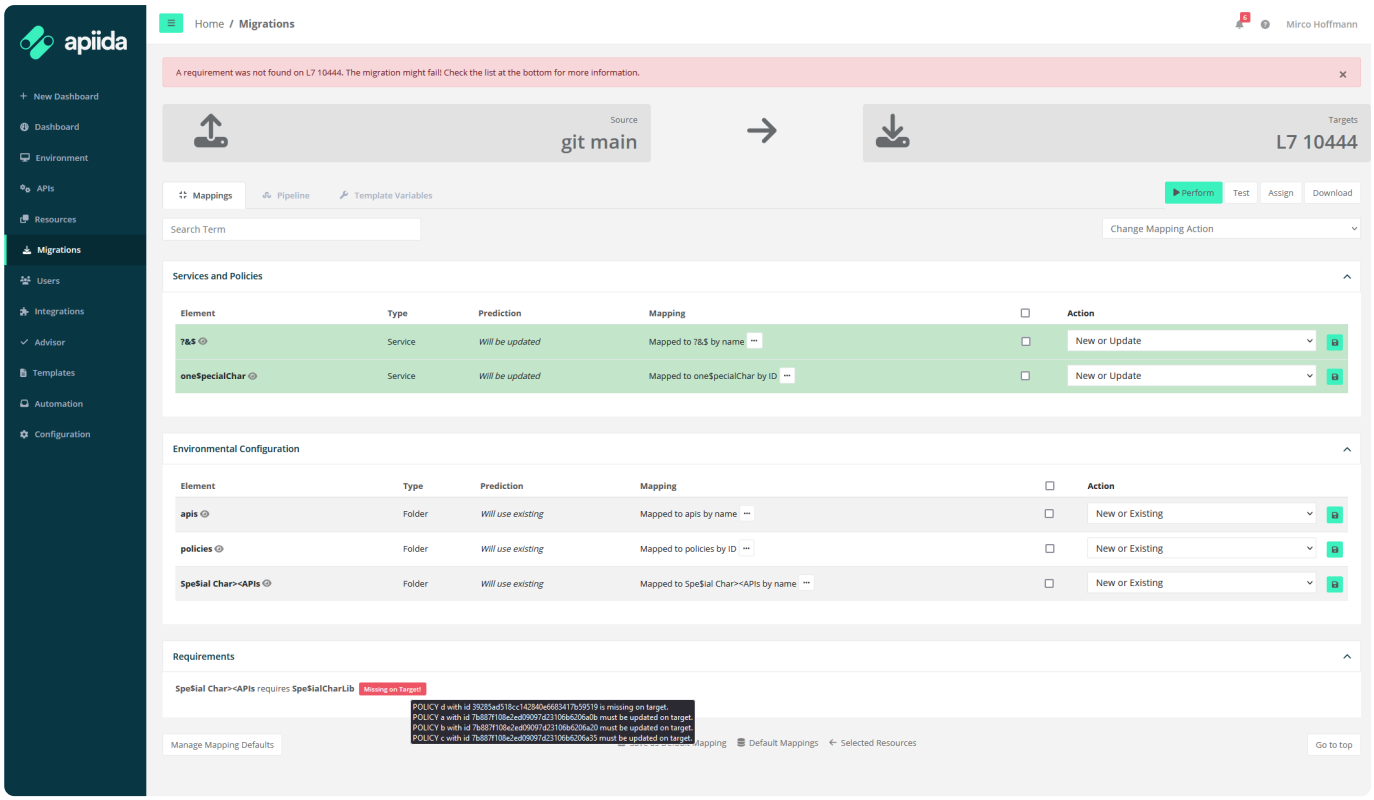
Select "Select from node and push to git" here. You will be redirected to the overview of the resources of the gateway from which the library was created.
You can also select a different gateway. But, some content may not be found and then you will have to select it manually. After you have made the needed changes to the content, click "Update Library".

If you only want to update the contents of the library itself, follow the steps described. But first you should update the resources of the corresponding node.
Resources -> Reload Resources

5.16.7 Migration, Variables, Requirements (Library only)

These functions behave as in Folder.
A library should be migrated like an API. When migrating an API that requires it, a check is made to see if it exists on the target.

It also checks if the library would need to be updated on the target.



5.17 Configuration

5.17.1 Configuration

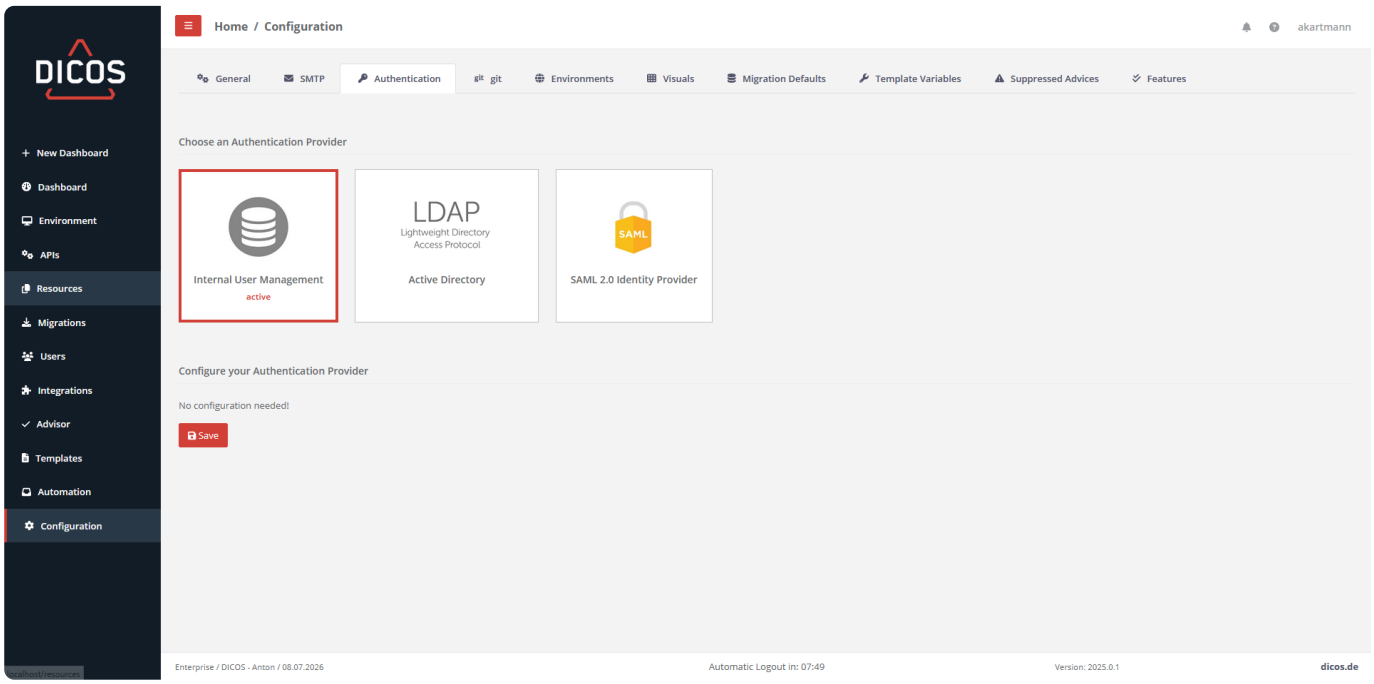
All the Gatway Manager setting options are collected here. This menu item is only visible to administrators. If you do not find what you are looking for here, have a look at the extended configuration articles.

Configure/Install ...

- General
- SMTP
- Authentication
- Git
- Environments
- Visuals
- Migration Defaults
- Template Variables
- Suppressed Advices
- Features

5.17.2 **Authentication**

DICOS API Gateway Manager supports three authentication methods.

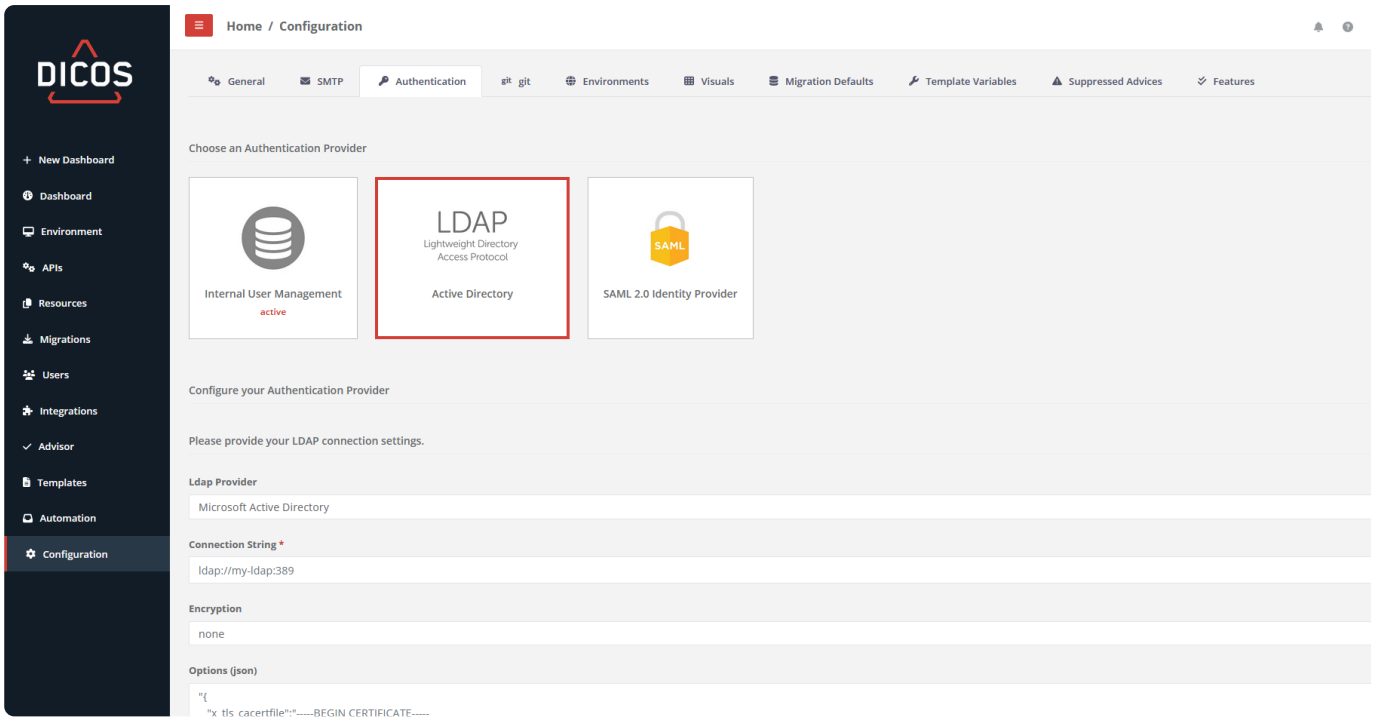


Internal User Management

You can manage the users under Users.

ActiveDirectory (LDAP)

When an Active Directory is connected and a user logs into AAGM, the user is pulled and created in AAGM.



- Ldap Provider Microsoft Active Directory or an openLDAP can be selected.
    - Microsoft Active Directory
    - openLDAP
- Connection String This consists of the protocol, the domain and the port.
    - ldaps://my-ldap:1234
- Encryption
    - SSL
    - TLS
    - none
- Options (json)
    - ```java
      {
          "x_tls_cacertfile":"-----BEGIN CERTIFICATE-----
                              MIIFazCCA1OgAwIBAgIUKJHXfgKThHc5dqU+vd2fm7IE+AswDQYJKoZIhvcNAQEL
                              ...
                              Your escaped json Ca certificate
                              ...
                              8uvO5DPtOK3oTA4PoobumJO2DcC+dZ51yK1XzE4ItFF4h7AutghKNe64m3GdrBE=
                              -----END CERTIFICATE-----"
      }
      ```
        - If the Active Directory is accessed via ldaps and has a self-signed certificate, the root certificate must be specified here. (escaped)
- Base DN
    - dc=example,dc=org
- User name attribute
    - cn
- Login domain
- Technical Read-Only User (DN)
    - cn=admin,dc=example,dc=org
- Password

SAML 2.0 Identity Provider

Support for SAML (Security Assertion Markup Language) gives users access to DICOS API Gateway Manager through an identity provider (IDP) of your choice, using two-factor authentication.

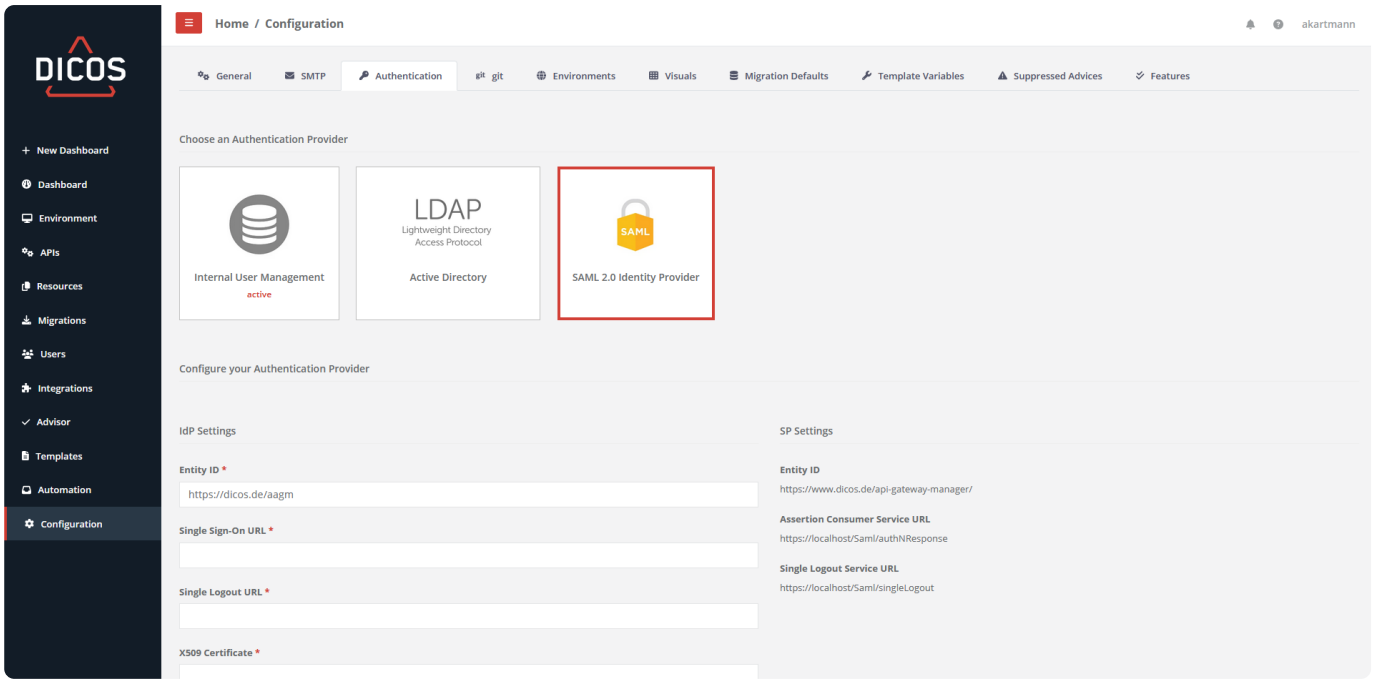
Note: Authentication using SAML and using your internal user credentials are enabled in parallel and the user can choose which method to use.

Enabling SAML authentication is done in two easy steps:

STEP 1: CONFIGURE DICOS API GATEWAY MANAGER

To configure DICOS API Gateway Manager for SAML authentication, go to Configuration > Authentication and select “SAML 2.0 Identity Provider”.

Configure the IDP settings according to your SAML identity provider, including its SAML signing certificate. You will typically find this information in the configuration or documentation of your identity provider software. To add the IDP’s SAML X.509 certificate, just open the .cer file in an editor and copy/paste the entire content of the file. In the example below, we are using DICOS Intelligent SSO as the identity provider:
      ```



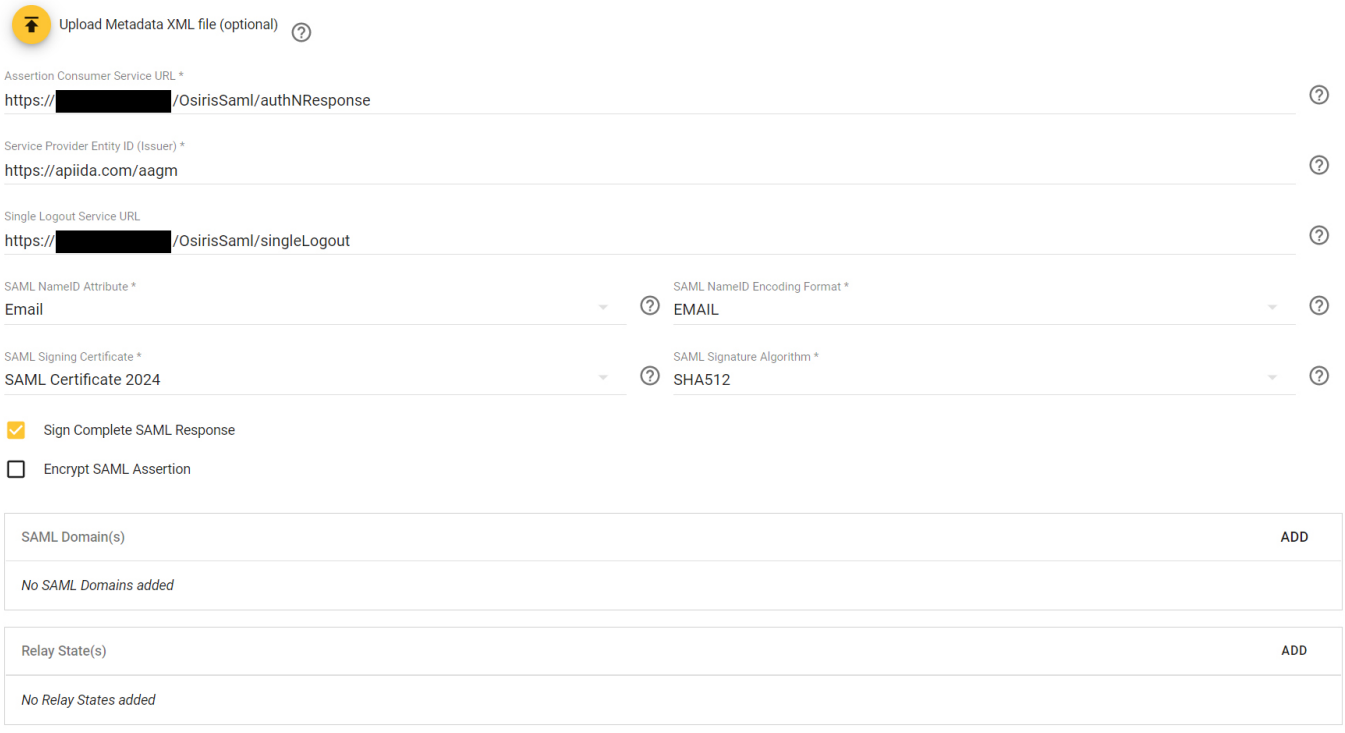
STEP 2: CONFIGURE THE SAML IDENTITY PROVIDER

To configure your SAML Identity Provider, you find all the required settings on the configuration page above, incl.:

- Service Provider Entity ID (Issuer)
- Assertion Consumer Service URL
- Single Logout Service URL

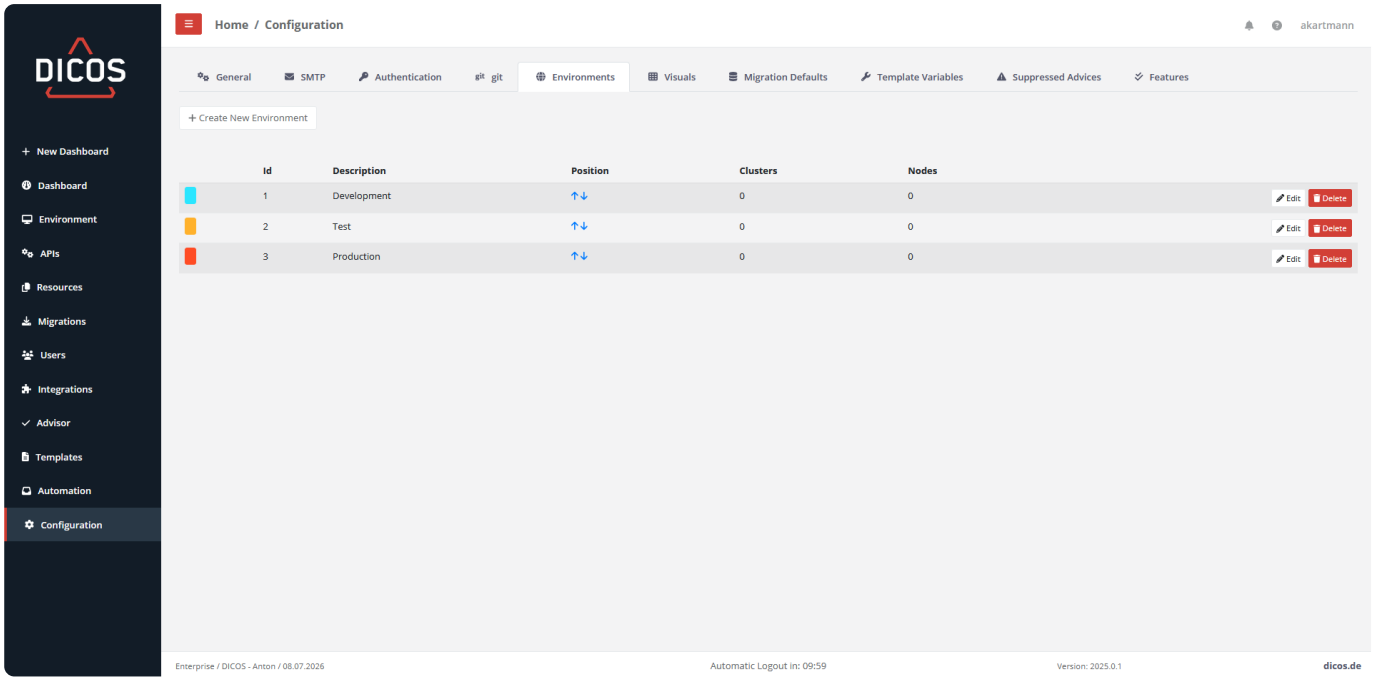
Select Email as the SAML nameID attribute and encoding format.

Here is an example of a configuration for DICOS Intelligent SSO:



5.17.3 Environments

Here you can see an overview of your environments. For a better overview, you can see an assigned colour and the name of the environment here. The ID is important for communication with the API. The sequence can also be customised here.



Edit an Environment

GENERAL

Name and Color

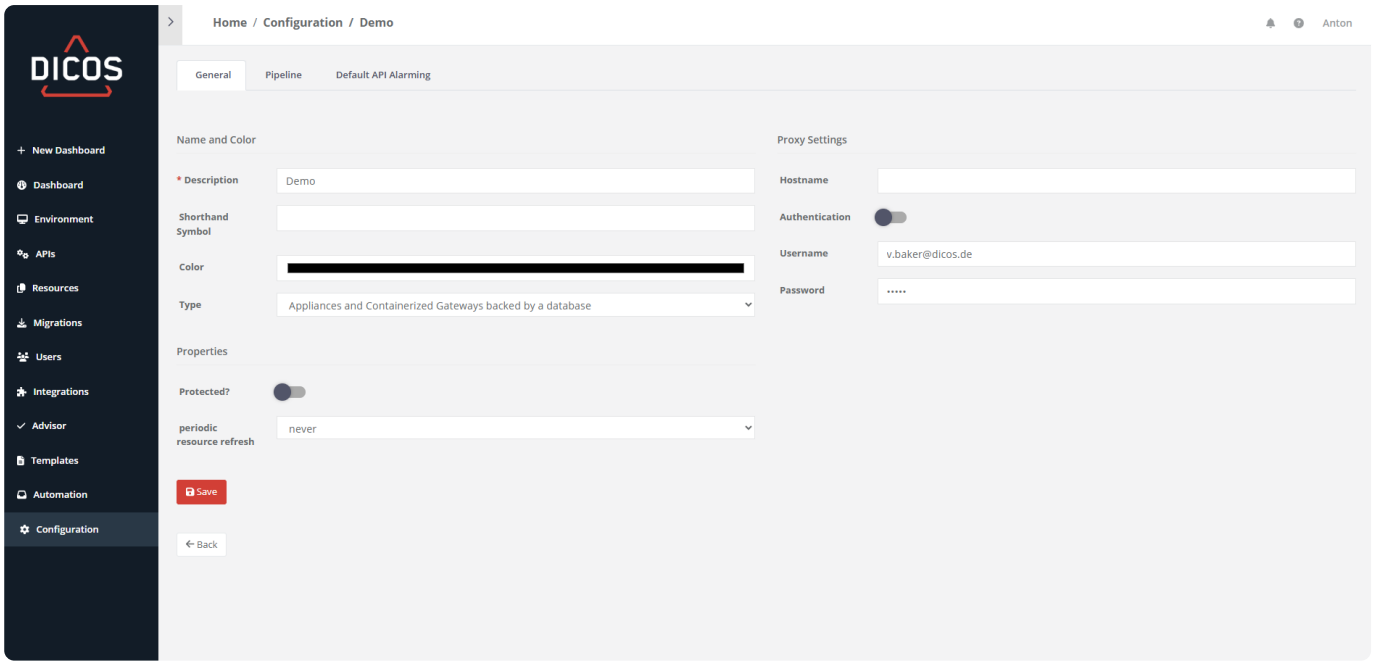
The description, i.e. the name, must be unique. You can freely choose the colour and a short symbol.

Properties

You can mark an environment as protected so that you have to enter your password again before performing a migration to this environment. You can set up periodic scans for all nodes in this environment. This keeps your nodes up to date and saves you having to index them before each migration.

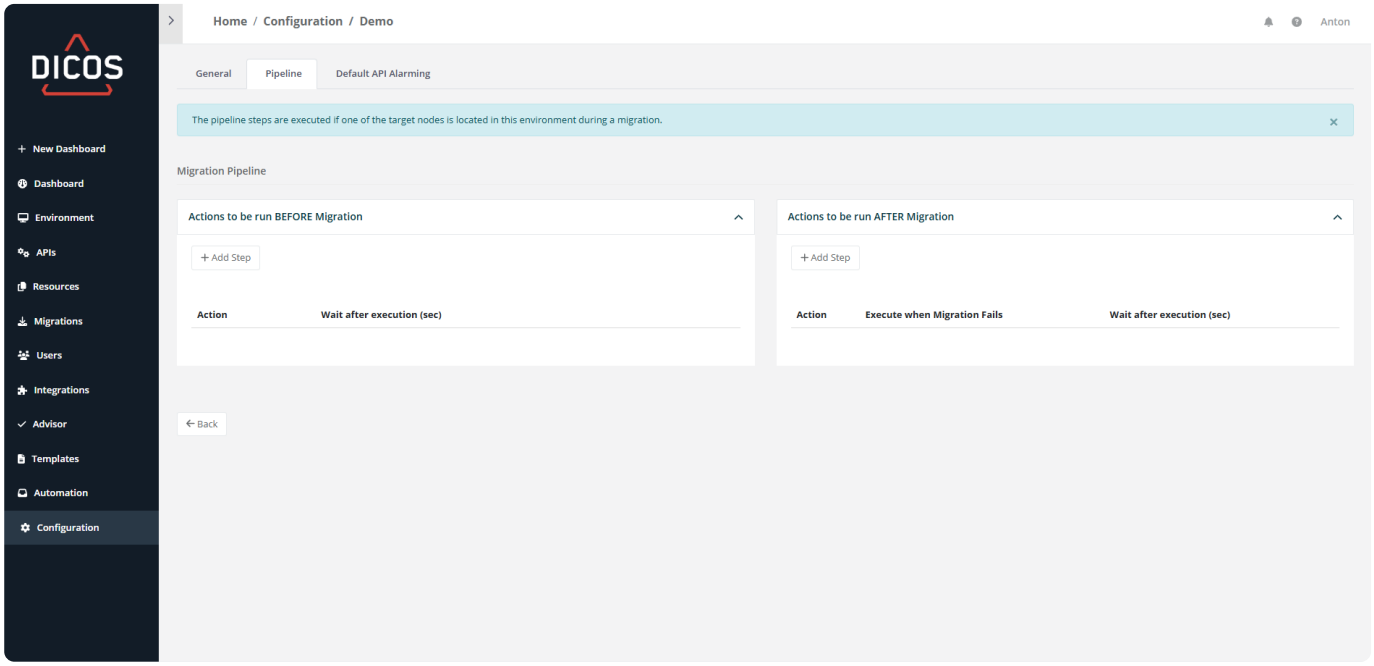
Proxy Settings

If a proxy is necessary for communication with the nodes in this environment, you can configure it here. The proxy is then automatically used for communication with Restman and Graphman.



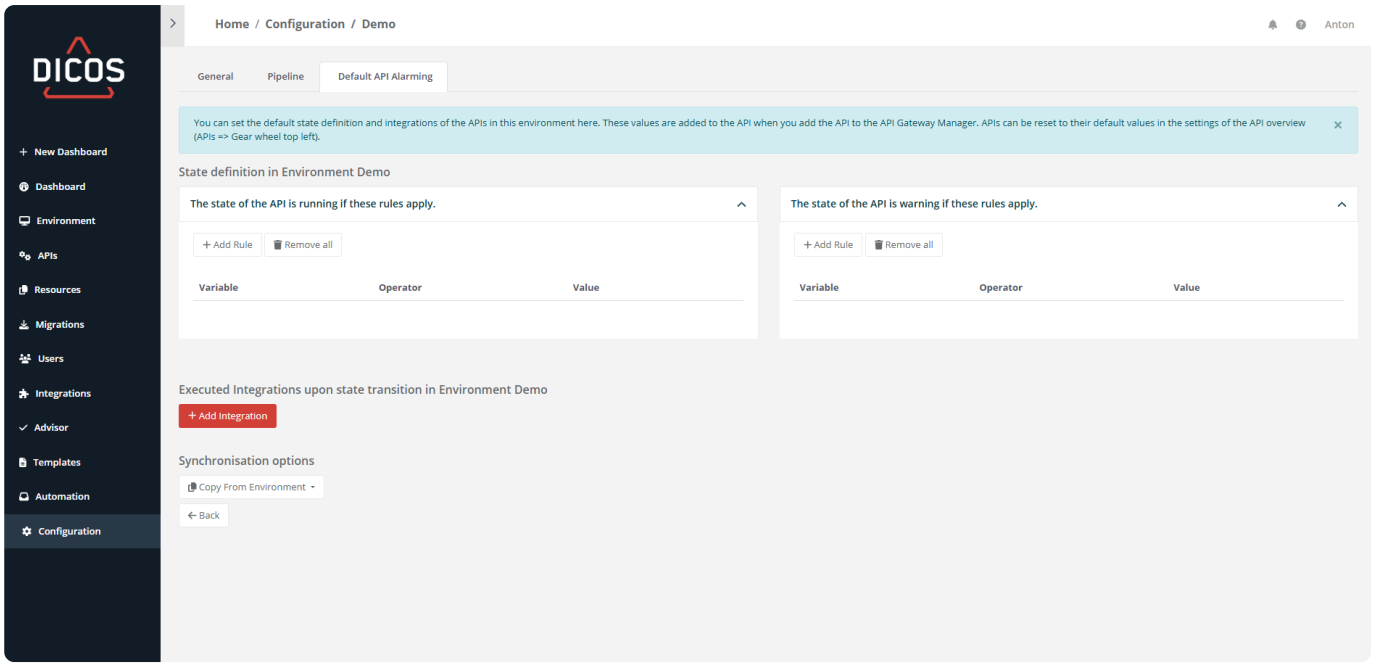
PIPELINE

Here you can define pipeline steps that are to be executed if the target of a migration is a node from this environment.

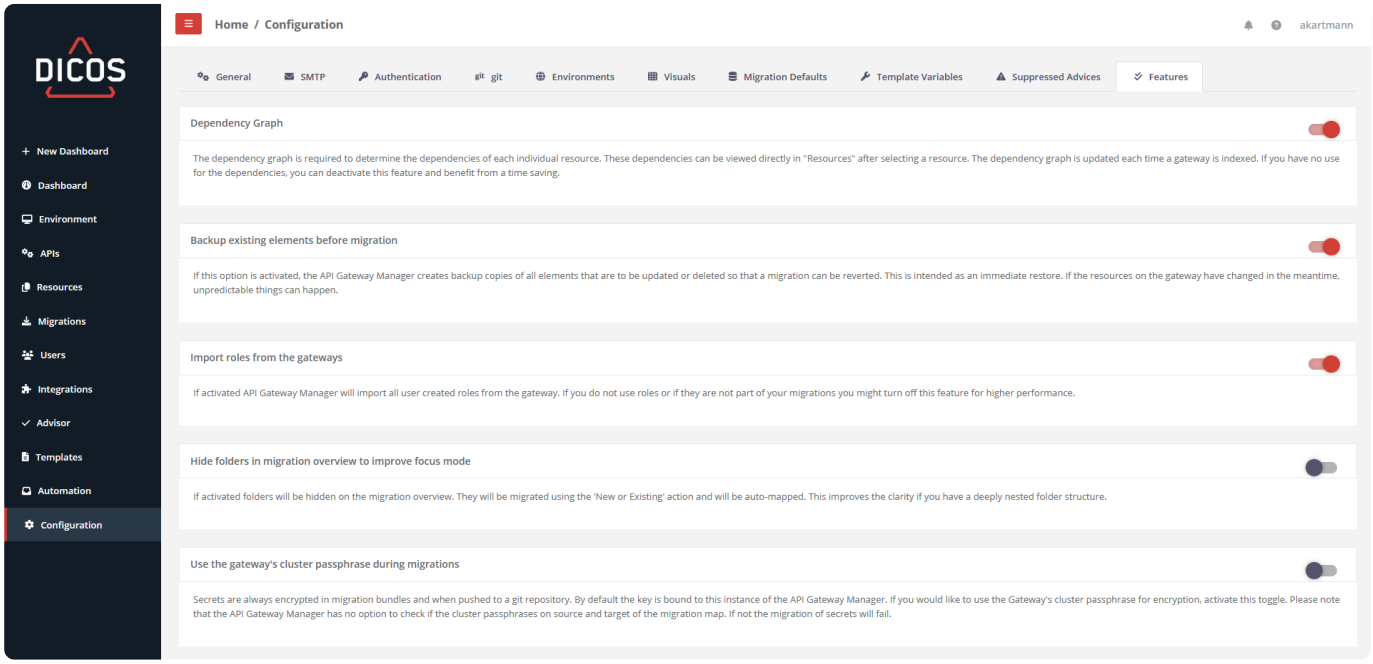


DEFAULT API ALARMING

Similar to nodes, you can define under which conditions an API is in which status. In addition, you can have a message sent automatically when an API changes to another status, for example from running to warning. You can edit these notification options in "Integrations". This is individual for each environment and all newly created APIs receive these initial Values.

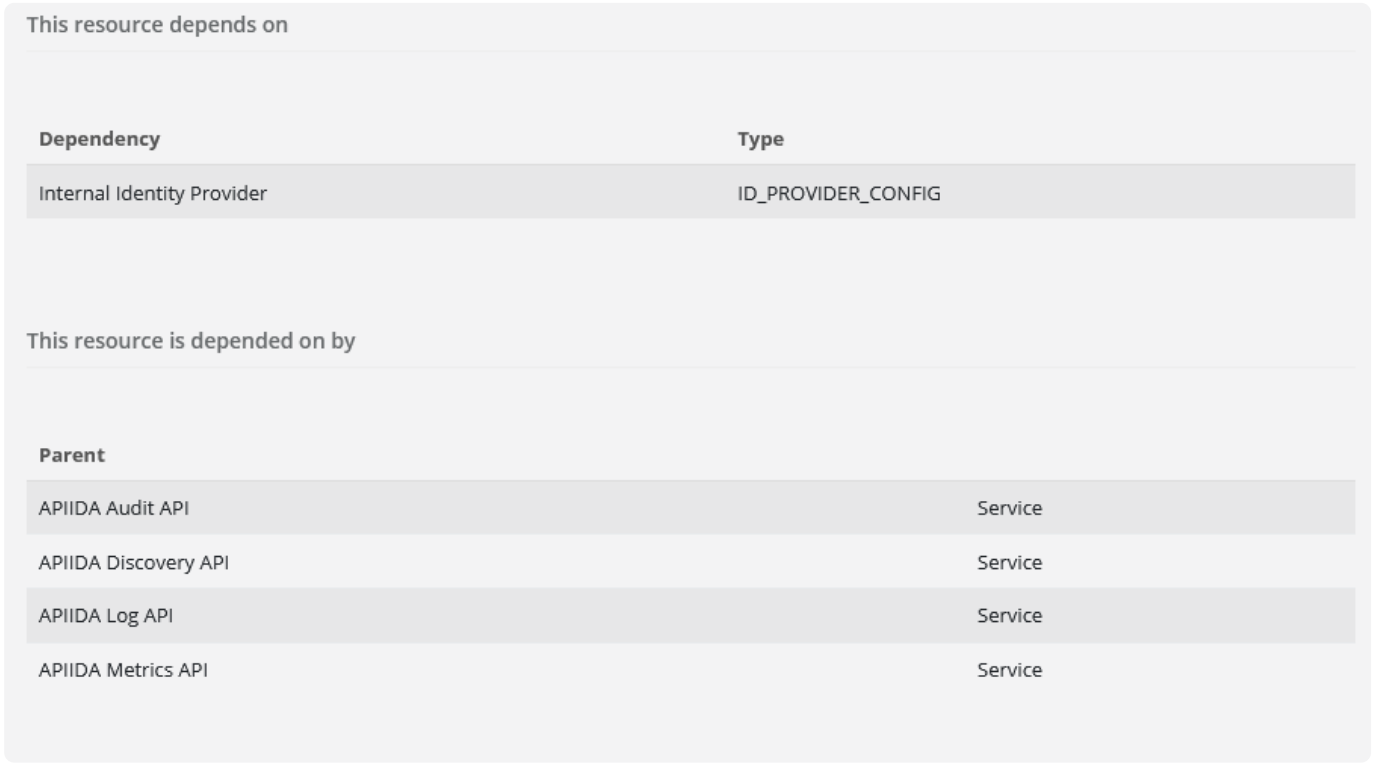


5.17.4 **Features**



5.17.5 **Dependency Graph**

The dependency graph is required to determine the dependencies of each individual resource. These dependencies can be viewed directly in "Resources" after selecting a resource. The dependency graph is updated each time a gateway is indexed.



If you have no use for the dependencies, you can deactivate this feature and benefit from a time saving.

5.17.6 Backup existing elements before migration

If this option is activated, the API Gateway Manager creates backup copies of all elements that are to be updated or deleted so that a migration can be reverted.

This is intended as an immediate restore. If the resources on the gateway have changed in the meantime, unpredictable things can happen.

Revert Migration

5.17.7 Import roles from the gateways

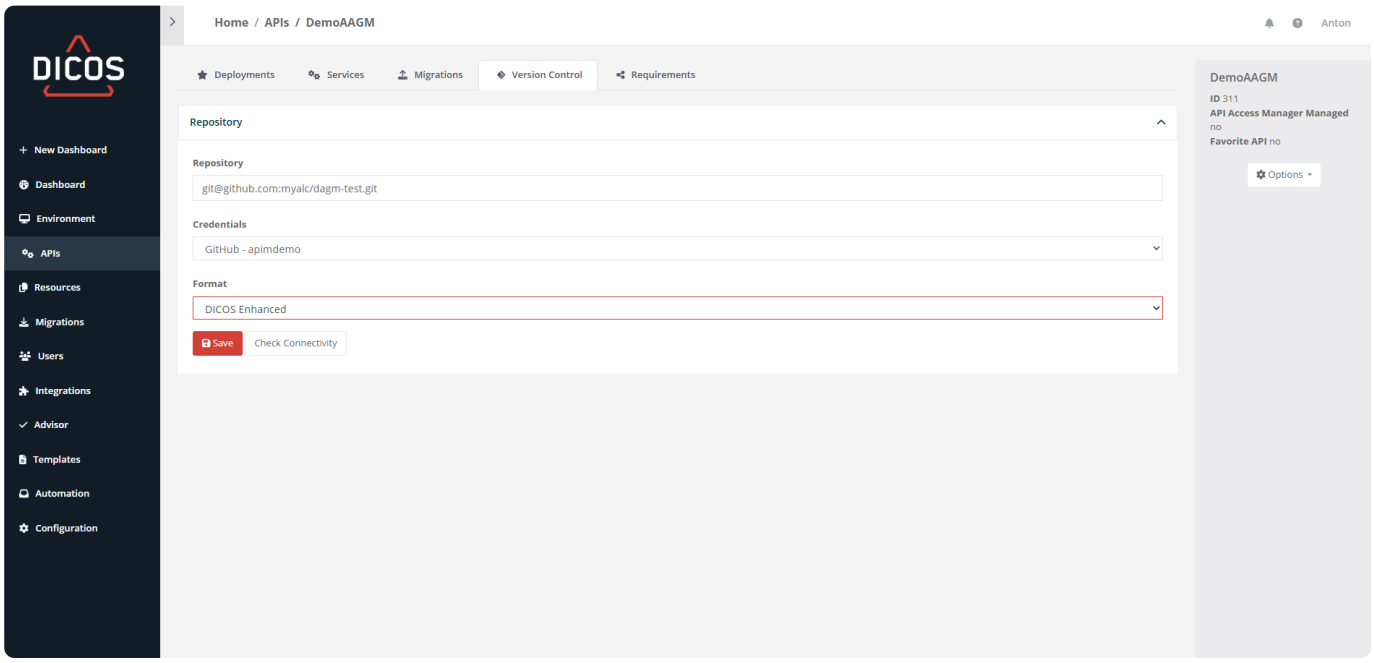
If activated API Gateway Manager will import all user created roles from the gateway. If you do not use roles or if they are not part of your migrations you might turn off this feature for higher performance.

5.17.8 Hide folders in migration overview to improve focus mode

If activated folders will be hidden on the migration overview. They will be migrated using the 'New or Existing' action and will be auto-mapped. This improves the clarity if you have a deeply nested folder structure.

5.17.9 Use the gateway's cluster passphrase during migrations

Secrets are always encrypted in migration bundles and when pushed to a git repository. By default the key is bound to this instance of the API Gateway Manager. If you would like to use the Gateway's cluster passphrase for encryption, activate this toggle. Please note that the API Gateway Manager has no option to check if the cluster passphrases on source and target of the migration map. If not the migration of secrets will fail.



This error message appears when the API repository was created by another AAGM instance. With the cluster passphrase this problem is fixed. Enable this feature in both AAGMs and recreate the repository.

5.17.10 Import Solution Kits and Their Contents

If activated API Gateway Manager will import Solution Kits and their contents. Turn off if you do not use requirements and the import takes too long.

5.17.11 Import Groups and Users

If activated API Gateway Manager will import groups and users that are associated with the identity providers found. When turning off you will loose the ability to migrate any such groups or users. Instead they will show as "Missing on source" during a migration.

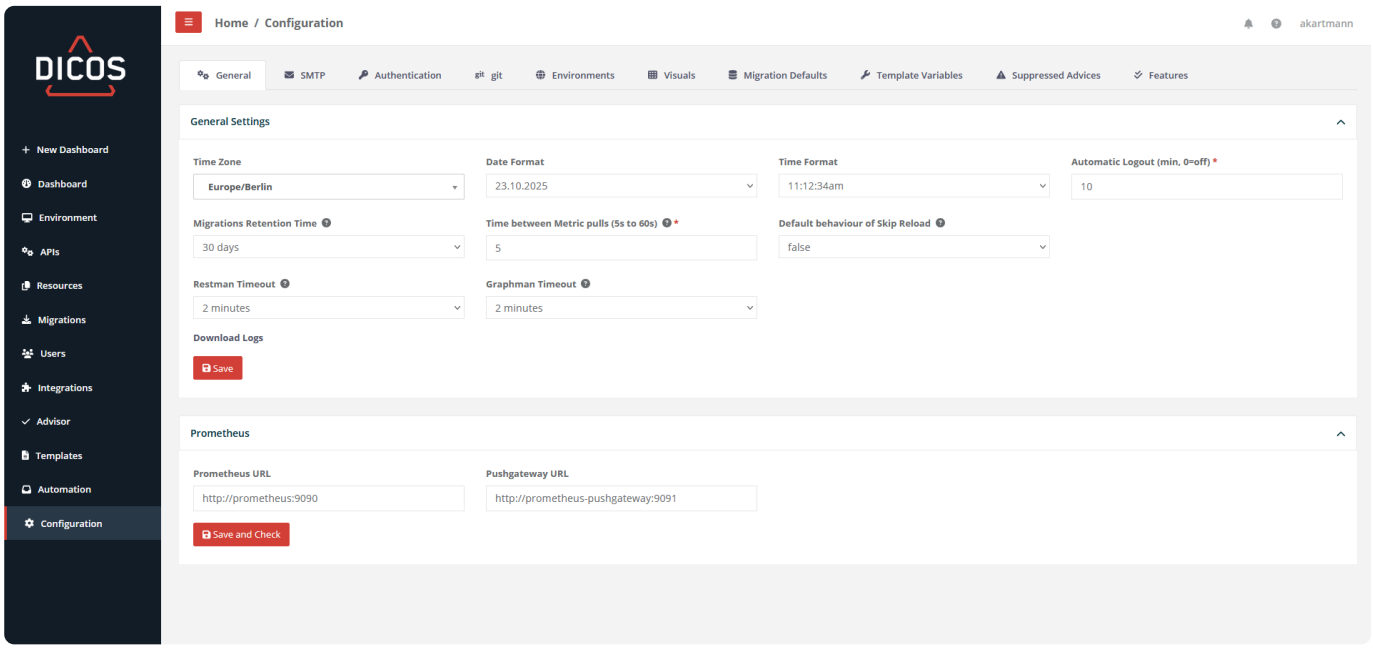
5.17.12 Use Graphman to communicate with the Gateways

Use Layer7's new Graphman interface for communicating with the gateway. As not all resources have been implemented by Broadcom, the Restman is used as a backup. Graphman dramatically reduces the time required to reload the resources of a node.

Reload Resources

5.17.13 General

The general settings allow you to configure some of the more fundamental properties of how DICOS API Gateway Manager behaves on a global level.



GENERAL SETTINGS

Setting	Description
Time Zone	The Gateway Manager saves all data / time values in UTC. This setting controls which time zone is used for display in the user interface. More elaborate settings concerning date and time (like the display formats) can be found in the configuration file.
Retention time	DICOS API Gateway Manager deletes old data regularly. These are metrics gathered from the gateways, alarms their confirmation and past migrations including their backups and logs. This setting allows you to specify the number of days after which that happens. Default is 30 days.
Automatic Logout	The number of minutes of inactivity after which the Gateway Manager logs a user out. The user is sent back to the login screen, requiring her to authenticate herself again in order to continue working in the system. Enter a 0 to disable the automatic logout.
Metrics time period	If you manage a large number of nodes in AAGM, it is advisable to increase the interval for scanning the metrics in order to reduce the load on the system.
Migrations Retention Time	Once the retention time has expired, migrations are completely removed. All files and logs are deleted.
Time between Metric pulls (5s to 60s)	If you only use the metrics as a basic overview and the accuracy and reliability of the metrics and alarms play a secondary role. You can increase the time period between pulls and thereby reduce the load on the Gateway Manager and your gateways.
Default behaviour of Skip Indexing	Before each migration, all required resources are retrieved again from the specified source and target. If the gateway is managed exclusively by this API Gateway Manager instance or you know that all resources are up-to-date, this step can be skipped.
Restman Timeout	Time until the request is cancelled by the Gateway Manager. For large migrations, it makes sense to increase this.
Graphman Timeout	Time until the request is cancelled by the Gateway Manager.

DOWNLOAD LOGS

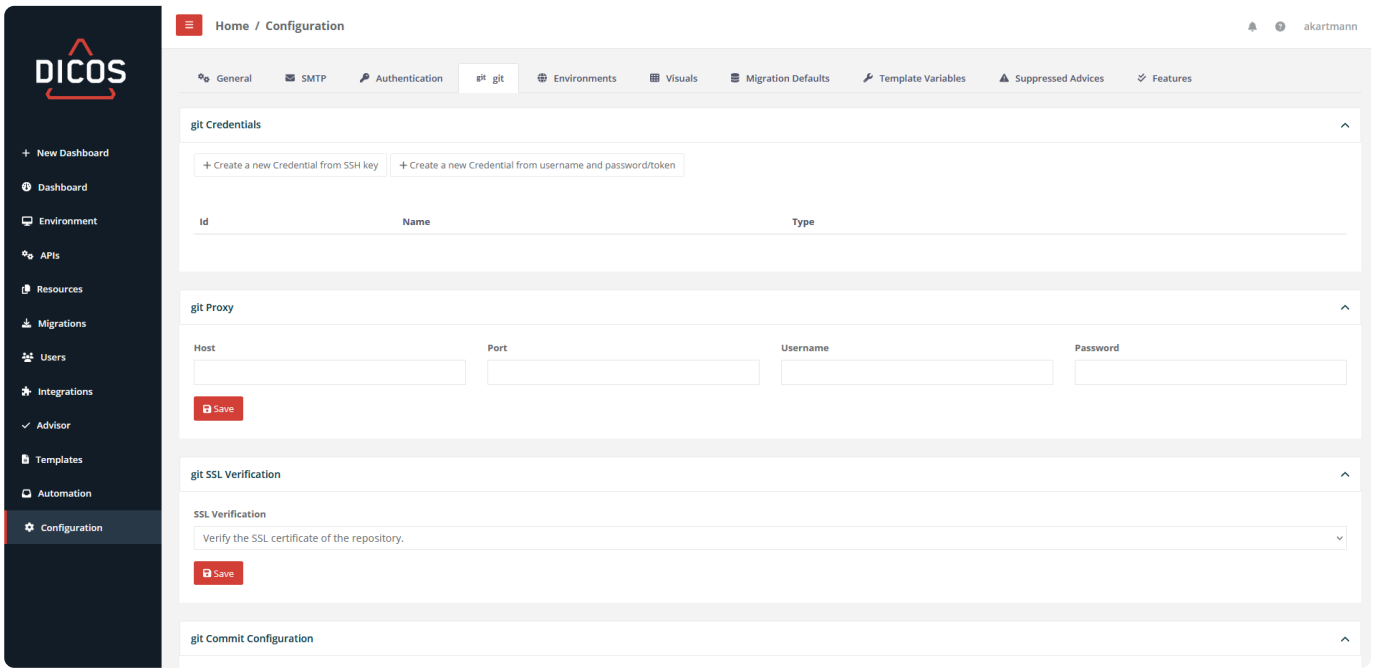
If you don’t have access to Gateway Manager’s persistent volume but need to look at the current log file you can use the button “Download Log” to download a copy of the current day’s log. Our customer support might ask you for this data to analyse an issues you have using the Gateway Manager.

PROMETHEUS

If you already have a Prometheus, you can also enter its URL to store the metrics of your APIs.

5.17.14 **Git**

Here you can make all settings relating to git.



GIT CREDENTIALS

Here you can create your git credentials either with your username and password or with an SSH key. The supported algorithms are: ed25519, ecdsa, dsa, rsa. You can also configure the key size.

The credentials are not only stored in the persistent volume but also encrypted in the data storage to prevent loss.

GIT PROXY

You may need to configure a proxy server if you're having trouble cloning or fetching from a remote repository or getting an error like `unable to access '...'` `Couldn't resolve host '...'`.

GIT SSL VERIFICATION

Verifying the SSL certificate of the repository is an important security measure to ensure that you are communicating with the actual Git server and not with a potentially malicious server.
You should only switch this off in secure and trusted networks.

GIT COMMIT CONFIGURATION

Here you can define the user name and email address of the user with which the gateway manager commits. Defaults are DICOS_API_Gateway_Manager and hello@apiida.com.

5.17.15 Migration Defaults

In order not to set the same mappings over and over again in each migration, you can also define standard mappings. These are set automatically when a new migration is created.

How are the actions of the mapping set?

The first/weakest definition is the bottom “Default Actions by Type”, all set actions would then be overwritten by what's in “Default Actions by Type and Regex” defined and these set actions would then be overwritten by what's in “Stored Manual Migration Mappings" defined.

However, directly selected elements are set to NewOrUpdate. These elements are marked with a green arrow in the migration overview.