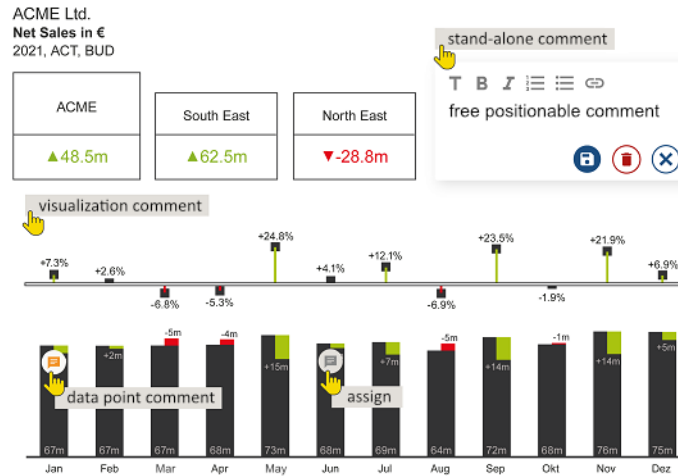


User Manual	2
Introduction	3
Installation	4
Custom Widget	5
comments backend server	6
Installation Docker Compose	7
Installation Kubernetes	8
Active Directory User	9
Definition of terms	11
Database structure comments	12
Context	13
graphomate property sheet GPS	14
Data Tab	15
Labels Tab	16
Scaling Tab	18
Input Output Tab	19
Scripton Documentation	20
Admin UI	22
Comments	23
Contexts	24
Users	25
User Roles	26
User Groups	27
Authorizations	28
Keycloak	30
Runtime interactivity	31
Know Issues	32
Changelog	33
Archive	34

User Manual for the graphomate comments 2023.2.0 for SAP Analytics Cloud (SAC)



PDF Download

Version 2023.2.0 – Stand Juli 2023

<https://www.graphomate.com>

graphomate .ll



Table of contents

- [Introduction](#)
- [Installation](#)
- [Context](#)
- [graphomate property sheet GPS](#)
- [Scripton Documentation](#)
- [Admin UI](#)
- [Runtime interactivity](#)
- [Know Issues](#)
- [Changelog](#)
- [Archive](#)

Introduction

Commenting on business data

Good visualizations help to understand data better and to identify patterns or trends.

With our **graphomate suite** and the **graphomate tiles** for various BI tools, we have given our customers to visualize data in an appropriate and meaningful way and to define a visualization standard that complies to the International Business Communications Standard (IBCS).

By creating a "visual context", decision-makers can better understand the business situation.

Another aspect is the enrichment of reports and dashboards with **textual content**. One would like to be able to add explanatory text to data constellations that require explanation.

Our **graphomate comments** provide this possibility on different levels:

1. Comments as independent text objects

A freely positionable stand-alone custom widget provides space to enter explanatory text for a specific data constellation.

Comments are stored with the creator and date of creation and are displayed when the data constellation is recalled. For example: "The EBIT for the whole organisation decreased in the month of May 2022 because an extraordinary depreciation was booked due to a fire in a warehouse".

2. Data point comments

Data point comments explain a specific data value in a graphomate suite component. The presence of a comment is indicated by a small comment icon on the chart element or in the table cell: "Revenues in the month of May 2022 are 20% higher due to the invoicing of the major project 'Airport'."

The presence of a comment is indicated by a small comment icon on the chart element or in the table cell:



3. Comments on a visualization

Comments can also be saved as titles in a visualization component of the graphomate suite.

Not a single data point, but a recognisable trend or pattern should be described in the title of the component. These comments are displayed either as full text or in the form of an icon.

Comments are always entered via an **identical input field** that opens on request.

Important: A prerequisite for entering comments is the definition of a so-called **context**.

This context describes the space spanned by dimensions and key figures as well as other restrictions for the graphomate comments.

Within this space, all characteristics of the selected dimensions and key figures can be commented freely.

The graphomate comments will also offer the possibility to inform users about a comment by e-mail.

graphomate comments allows you to map content-related permissions on a contextual level, and we are working on a workflow concept - for example, for approving comments.

Installation

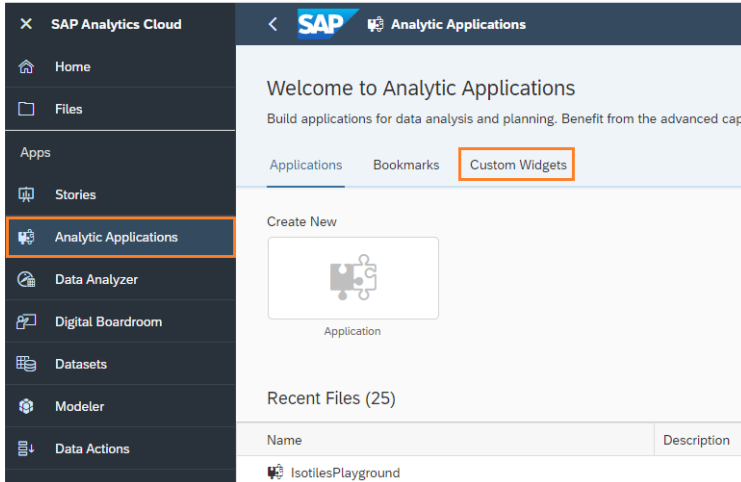
Custom Widget

In SAP Analytics Cloud, graphomate extensions are installed as custom widgets.

Please note, you must have administrator rights for the installation.

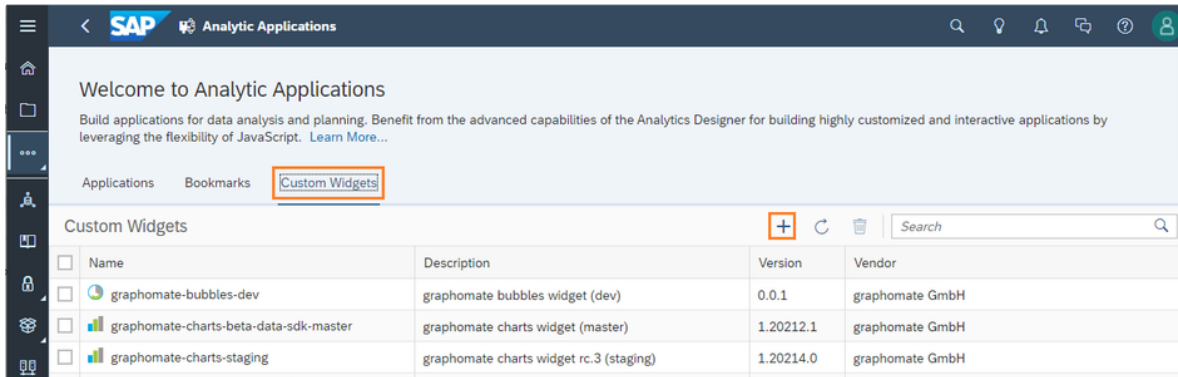
1. Step

Navigate to the sidebar and select the "Analytic Applications" entry



2. Step

Now switch to the "Custom Widgets" tab. Via the '+'-symbol you can add the graphomate JSON files provided by us to install the graphomate widgets.



The graphomate extensions can now be used as 'custom widgets' in Analytic Applications and Stories.

comments backend server

Installation bundle

There are two different installation bundles. One is for installation in a pure Docker environment via docker-compose. The other is intended for installation in an environment with Kubernetes, e.g. Open Shift.

Important:

The **admin user** created during installation is also used as a service user. This means that it must not be locked and if its password is changed in Keycloak, the password must then be adjusted in the Docker or Kubernetes configuration.

- [Installation Docker Compose](#)
- [Installation Kubernetes](#)

Installation Docker Compose

Our installation bundle for the comment server consists of several files:

.env

In this file, environment variables are set that define under which SSL port the application should run. It also defines the names of the files that contain the SSL certificate and SSL key you have provided for your domain.

***_password.txt**

Two files ending with "_password.txt" define which passwords (initial) are to be used for the database and the global administrator user.

config.properties

This is where the settings for our comment server are defined. Currently it is used to configure the SMTP mail server used to send notification emails when a user is mentioned in a comment.

init-commenting-db.sql

The initialisation of the Postgres server with a new database and two schemas is defined here. Nothing is to be adjusted here by the user.

nginx.conf

The configuration file contains the definition of how the NGINX reverse proxy should forward the requests against the commenting backend to the individual Docker containers. Usually nothing needs to be adjusted by the user here.

docker-compose.yaml

The definition of which containers must be created for the commenting backend, which files may be used by the host system and how the internal network is structured does not usually need to be adapted.

commenting.war

This file contains the logic for the commenting backend and is automatically installed in the Tomcat container in Docker.

Installation process

1. Install Docker as described here: [🌐 Install Docker Engine](#)
2. Unpack the installation bundle and copy the files from the installation bundle to the Docker host system in any location.
3. You insert your SSL certificate with the file name "ssl_certificate.crt" and the corresponding private key with the file name "ssl_certificate.pem" in the same location as in step 1. If the paths or file names are different, the ".env" file must be adapted.
4. If you want to use a port other than "3333" for the server, adjust the port in the ".env" file.
5. In "commenting_db_password.txt" file you define the password for the user administration database and the comment repository. This is transmitted internally to the corresponding containers.
6. In the file "keycloak_admin_password.txt" you set password for the comment backend administrator. The administrator is authorised to read, write, edit and delete comments and to create new users. You will need this password to log in to our components and the admin interface for the first time. The username here is "admin". Newly created users can then be assigned the roles "viewer", "editor" or "admin".
7. When the configuration is complete, you can run "docker-compose up" from the command line in the installation bundle folder. This will launch all the required components.

Installation Kubernetes

Our installation bundle for the comment server on Kubernetes consists of several files:

In the following, we assume that you know how to use Dockerfiles and .yaml configuration files in Kubernetes.

Dockerfile*

The three Dockerfiles contain the instructions on how to create the three containers needed for commenting. The containers created in this way must then be pushed into your docker registry in order to use them in Kubernetes.

Files comments-keycloak-theme, commenting.jar, nginx.conf, keycloak-docker-entrypoint.sh

These files are used by the Dockerfiles to create the individual containers. A modification is usually not necessary here.

-kubernetes-.yaml

The five YAML files contain the configurations for Kubernetes to be able to start our service. These serve as orientation and adjustments may be necessary depending on your Kubernetes configuration.

- The **0-kubernetes-ns.yaml** defines the namespace in which the graphomate comments entities are created in Kubernetes.
- The **1-kubernetes-role.yaml** defines the roles necessary to allow graphomate services to read the configurations.
- The **2-kubernetes-config.yaml** file defines various configuration parameters for our service. This includes the connection details for the databases where the comments and users are stored. These can also be stored in the same database. This requires two schemas in PostgreSQL. So far, we have also been able to successfully test MSSQL on Azure without schemas. The configuration for this is somewhat more complex than for PostgreSQL and can be explained by us if required. The definition for the mail server is also done here. This is used to notify a user mentioned in a comment by mail.
- In the **3-kubernetes-secrets.yaml**, access data for the databases, the mail server and the initial administrator are maintained. These are Base64 encoded as usual for Kubernetes.
- In **4-kubernetes-deployment.yaml**, the three containers needed for our service are configured. In our example, these are all united in a pod and use temporary in-memory volumes. Of course, you are free to use other, for example persistent, volumes and to distribute the containers in different pods for better load balancing. However, in the case of the keycloak container, this requires a relatively complex configuration and an adaptation of the nginx.conf for the Nginx container, which we will not go into here.

Installation process

The three Docker containers have to be created with the Dockerfiles we deliver and pushed into your Docker registry used by Kubernetes.

After the configuration parameters and secrets have been adjusted, they should be applied in Kubernetes. Apart from creating a database and the schemas, our service takes care of the rest of the configuration of the database itself once the service is started for the first time.

In front of port 80 of the Nginx container, for example, you should set an ingress controller that takes over the SSL termination and exposes the port to the outside secured with SSL. Further ports on our service are not necessary, as the Nginx container acts here as a reverse proxy.

Once you have successfully installed the service, there are **three things** you can do to check that it is working:

- At the root URL to the service, a JSON should appear with information about the service and the installed version.
- `<Server URL>/auth` should open the user interface of Keycloak, where you can log in with the initial admin user data by clicking on "Administration Console". Here you can also connect other ident providers to our service. For example, Active Directory via SAML and external OAuth providers. For more information, please consult the Keycloak documentation.
- Finally, please check whether our user interface for the backend appears under `<server URL>/commenting/administration/`. Here you can, with a valid user (e.g. the admin again), view existing comments, edit them and also create new ones. You have the option of creating new users and authorising them to use context combinations.

Active Directory User

To connect users to our *comments backend* that are managed via Microsoft Active Directory, you need to use Active Directory Federation Services (AD FS) and SAML v2.0. If you have already set up AD FS on your server, please follow the instructions here to connect your AD to comments: [Configure MS ADFS as a brokered Identity Provider in Keycloak](#)

The necessary administration of Keycloak can be reached at: https://<SERVER_URL>/auth

Step 6 under "SETUP IDENTITY PROVIDER IN KEYCLOAK" is optional and in our case offers the possibility to map existing attributes in SAML to the three roles defined in the comments. In our case, the last field should contain either "viewer", "editor" or "admin" instead of "manager".

If you have followed the instructions, there is now another option when logging into *comments*:

A screenshot of a web form titled "Sign in to your account". It contains two input fields: "Username or email" and "Password". Below these is a blue "Sign In" button. Underneath the button is the text "Or sign in with" followed by a button labeled "ActiveDirectory".

If you now select the lower option, you can log in (if you are not already logged in) with a user from the Active Directory. When you log in for the first time, a form asks for Given-name and Surname of the user to be imported. To prevent this, additional mappers are necessary. The configuration required for this in the AD FS console looks as follows:

Edit Rule - email, subject, firstname, lastname

You can configure this rule to send the values of LDAP attributes as claims. Select an attribute store from which to extract LDAP attributes. Specify how the attributes will map to the outgoing claim types that will be issued from the rule.

Claim rule name:

Rule template: Send LDAP Attributes as Claims

Attribute store:

Mapping of LDAP attributes to outgoing claim types:

	LDAP Attribute (Select or type to add more)	Outgoing Claim Type (Select or type to add more)
▶	E-Mail-Addresses	E-Mail-Adresse
	SAM-Account-Name	UPN
	Given-Name	Angebener Name
	Surname	Nachname
*		

In Keycloak, two more mappers need to be added to the SAML Identity Provider:

ID:

Name:

Sync Mode Override:

Mapper Type:

Attribute Name:

Friendly Name:

User Attribute Name:

ID:

Name:

Sync Mode Override:

Mapper Type:

Attribute Name:

Friendly Name:

User Attribute Name:

After this configuration, no further data entry is necessary after an Active Directory user logs in.

Automatic login without manual selection of the new identity provider

After this configuration, a user must manually select the new identity provider from Active Directory when logging in. To skip this step, please follow the "default identity provider" step in the following instructions: [Server Administration Guide](#)

Definition of terms

Docker

Docker is an open source project designed to run software in an isolated environment called a container. This means that software running inside a container shares the basic functionality of the actual operating system, but is otherwise isolated from the rest of the operating system and other containers in terms of the file system and memory. This offers the advantage that software that is executed within a Docker container is always executed in a firmly defined environment with firmly defined and installed dependencies.

The Docker host, which is necessary to run Docker containers, can be installed on any Linux, Windows or Mac OSX server. However, since a Linux "intermediate layer" is always necessary, a Linux server is recommended. More information on installing Docker can be found here: [🐳 Install Docker Engine](#)

Docker Container

Docker offers a variety of predefined containers for specific application purposes. These containers are often maintained by the developers of a software themselves. For example, there is an official container image of Postgres, which runs the database of the same name. To configure a Docker container, it exposes a predefined set of settings that can be passed to the container from the outside.

Docker Compose

The description language "Docker Compose" can be used to configure how an application consisting of several Docker containers is structured. Here, for example, it is defined which containers including their configuration must be executed for the entire application, how the containers communicate with each other in a virtual network, which files from the host system should be available in the container and which passwords should be used.

Database structure comments

comment

- id: UUID;
- createdAt: Timestamp;
- changedAt: Timestamp;
- createdBy: string;
- lastModifiedBy: string;
- text: Lob;
- dashboardId: string;
- dashboardName: string;
- hostId: string;
- hostName: string;
- contexts: Foreign Key Context[];

context

- id: UUID;
- createdAt: Timestamp;
- changedAt: Timestamp;
- createdBy: string;
- lastModifiedBy: string;
- type: "DataContext" | "EnvironmentContext";
- key: string;
- value: string;
- keyText: string <optional for type DataContext>;
- valueText: string <optional for type DataContext>;

Example table rows

comments

	id	created_by	last_modified_by	changed_at	created_at	dashboard_id	dashboard_name	host_id	host_name	text	plain_text	html_text	encode	encode
1	178051cc6744-4366-9f5e-4577a6071...	admin	admin	2022-11-21 13:42:37.827812	2022-11-21 13:42:37.827812	[null]	[null]	[null]	[null]	18046	18045	[null]	*locks*	test

contexts

	type	id	created_by	last_modified_by	changed_at	created_at	key	value	key_text	value_text	comment_id
1	DataContext	1d79ad5c-df3f-4204-8cd3...	admin	admin	2022-11-21 13:42:37.961738	2022-11-21 13:42:37.845847	genre	Action	[null]	[null]	178051cc6744-4366...
2	DataContext	a7ec342f-78c2-4fe0-9b16...	admin	admin	2022-11-21 13:42:37.967597	2022-11-21 13:42:37.840345	Measures	imdrating	[null]	[null]	178051cc6744-4366...
3	DataContext	dc2c5940c4a-48d4-b1a...	admin	admin	2022-11-21 13:42:37.969246	2022-11-21 13:42:37.850286	name	Result	[null]	[null]	178051cc6744-4366...

Context

In order to be able to assign a comment, it is stored with a freely definable set of so-called contexts. These contexts allow the comment to be associated with a data constellation, a filter constellation, a specific visualization, a specific dashboard or any other restriction. For example, that the current comment applies to the "year 2021" in the "country Germany".

A context is always described by a *key*, e.g. a dimension, and a *value*, e.g. a member. In the examples above, the keys would be 'year' or 'country' and the values would be '2021' or 'Germany'. Contexts can be defined **statically** in the GPS or **dynamically** using the scripting language in BI environments that allow scripting, such as SAP Analytics Cloud (SAC). If you use **data point comments** in a visualization, the contexts of the comments are automatically derived from the dimensions members used in the visualization. For a **title comment** that describes the visualization as a whole, an environment context is automatically created that refers to a randomly generated and persisted ID of the visualization.

A hierarchy is implicitly formed from the set of contexts for a comment, which is mapped in our independent comment component under "more specific comment(s)". For example, if only one context is specified as the context combination to be displayed, all comments containing **at least** that context will be displayed here. Assuming that only the name of the dashboard is specified as the context combination, more specific comments about each year will also be displayed here.

Context Types

Environment Context

The "Environment Context" is intended to limit the comment space to properties that are not dependent on the data. For example, the comment area could be restricted to the current dashboard. An Environment Context is also defined by a combination of *key* and *value*. In our example, the *key* could be something like "Dashboard Name" and the *value* could be "Sales Dashboard".

Data Context

The "Data Context" is used to restrict the comment space to a certain data constellation. In addition to the properties *key* and *value*, it also contains the optional properties *keyText* and *valueText*. These are used to maintain the display name from the data source (if available) for the dimension and member. This makes the display of contexts associated with a comment easier to understand. Dashboards and data layers can be very diverse and dynamic. Therefore, they are usually read and maintained directly in scripting environments. For example, the data context could be set so that the *key* 'Year' is defined with the *value* '2021'.

Context setting types

Static Contexts

The "Static Contexts" are defined in the GPS and thus apply independently of filter states defined by scripting, for example, throughout the entire runtime. A static context could be used, for example, to limit the comment space to the current dashboard. The type "Environment Context" should be selected for this.

Dynamic Contexts

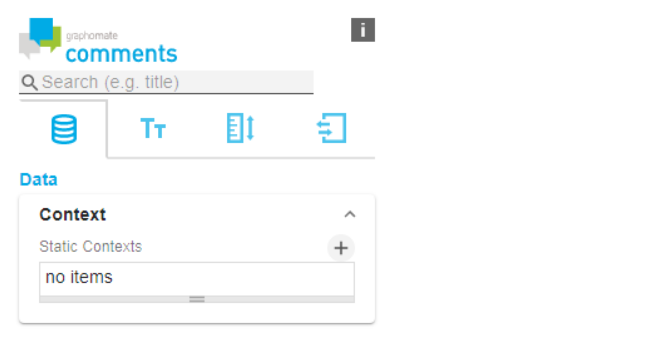
The "Dynamic Contexts" are set in environments with scripting, there based on the current state. These are usually used to dynamically pass on the current filter state on the data to the comments. In this way, a maintained comment is assigned to exactly this data constellation.


For example, we can connect a dropdown with a selection for the current year. When the dropdown is used, it filters the data source at the same time and adds a data context. This data context then consists of the *key* "Year" and the *value* that was selected in the dropdown. Static contexts are of course retained.

Automatic Contexts

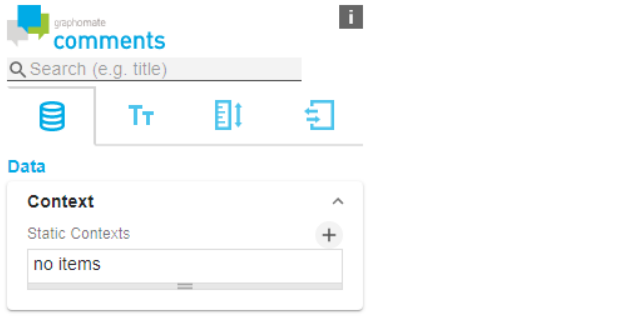
Automatic contexts are created when using our visualizations with comments. The data hierarchy associated with the data point is automatically assigned to the comment. The visualization comment is automatically assigned the visualization ID.

graphomate property sheet GPS

<p>Context</p> <p>Static Contexts</p> <p>Static contexts associated with the comment, such as a dashboard name or dashboard page, can be maintained here.</p>	
---	--

 Error loading excerpt from "CO:en GPS Tab2 Input Output (comments)".

Data Tab

<p>Context</p> <p>Static Contexts</p> <p>Static contexts associated with the comment, such as a dashboard name or dashboard page, can be maintained here.</p>	 <p>The screenshot displays the 'comments' data tab in a software application. At the top, there is a search bar with the placeholder text 'Search (e.g. title)'. Below the search bar is a toolbar containing four icons: a database icon, a text icon, a list icon, and a table icon. The main content area is titled 'Data' and contains a 'Context' section. Under 'Context', there is a 'Static Contexts' list with a '+' icon to add items. The list currently shows 'no items'.</p>
---	--

Labels Tab

Information

Show Contexts

If this option is enabled, the contexts for which the current comment is valid are displayed above the comment.

Show Authoring Information

By activating this option, a text will be displayed below the comment showing who created or last edited it. It also shows when this was done.

Show More Comments

If this option is enabled, more specific comments will be displayed below the 'main' comment.

More Comments Expandable

The more specific comments are all displayed directly in the standard. As this can be a very large number of comments, it is possible with this option for them to be 'collapsed' and only displayed with a further click.

Show Menu Edge

Shows a blue corner at the top right, which opens a menu. The menu shows the user currently logged in, with an option to log out.

Font Family

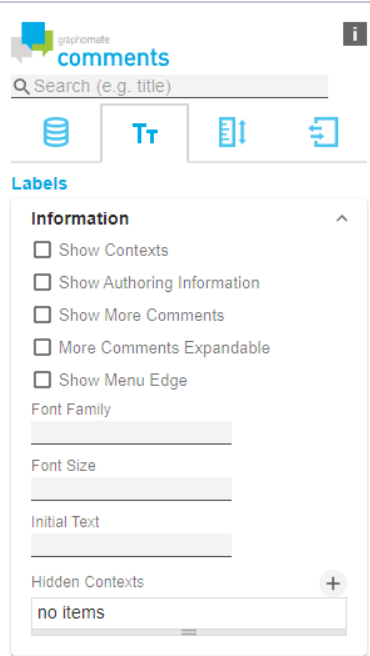
Here you can define the font to be used for the comment text and also the buttons in the comments.

Initial Text

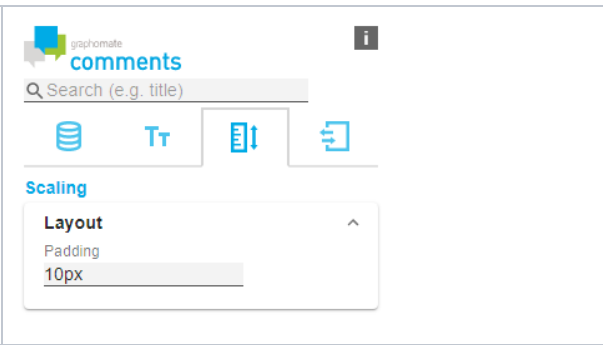
If no comment is available, a note text indicates this condition. This can be modified here. This function can also be used to create a basic structure for new comments as a template. HTML can be used here for formatting.

Hidden Contexts

As not all contexts are set dynamically and therefore remain the same in every state of a dashboard, it can be useful to hide them. By defining a hidden context, it is possible to specify that a context is used to classify a comment, but is not displayed in the UI. All options are optional. For example, if you define a hidden context of type "Environment Context", all contexts of this type will be hidden. All contexts with a particular key can also be hidden in this way.



Scaling Tab

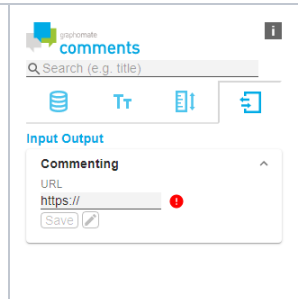
<p>Scaling</p> <p>Padding</p> <p>The padding defines how much space the component should leave from the outer edge.</p>	 <p>The screenshot shows a design tool interface for a component named 'comments'. At the top, there is a search bar with the placeholder text 'Search (e.g. title)'. Below the search bar are four icons: a database icon, a text icon, a list icon, and a document icon. The 'Scaling' tab is selected and expanded, showing a 'Layout' section with a 'Padding' property set to '10px'.</p>
---	--

Input Output Tab

Server

Comments Server URL

This URL points to the *graphomate comments backend* and is used for storing comments and user administration. Once this URL has been maintained, it is usually not necessary to maintain it again in each instance of our comments and dashboard. When you click on "save", the system checks whether the server is responding correctly. If this is the case, it is symbolised by a green "tick". At the same time, a link to the backend administration interface will appear.



Scription Documentation

graphomate comments integrate seamlessly into the SAP Analytics Cloud. Scripting is also supported in the Application Designer of the SAP Analytics Cloud.

The properties shown in the property sheet can be retrieved and set by scripting from the component. There are also methods that make it easy to set contexts for comments:

createDataContext(key: string, value: string): DynamicContext;

A data context can be created via this method, which can then be set via the **setDynamicContexts** method, which is passed an array of contexts. A data context usually contains a combination of dimension and member. For example, the *key* is "year" and the *value* is "2021".

createDataContextWithText(key: string, value: string, keyText: string, valueText, string): DynamicContext;

If your data source makes a difference between IDs for dimensions and members and their display text, this method can be used in the same way as `createDataContext`.

In addition, it is possible to set the display texts. For example, here the key is "CAL_YEAR" and the keyText is "Year".

createEnvironmentContext(key: string, value: string): DynamicContext;

An environment context can be used to limit the comment space to, for example, a specific environment or dashboard. Here, the key could be something like "Dashboard Name" and the value could be "Global Sales 2021".

addOrReplaceDataContext(key: string, value: string): void;

This method works in the same way as `createDataContext`. However, it adds or changes the context defined here directly to the widget. The path through `setDynamicContexts` is therefore not necessary.

addOrReplaceDataContextWithText(key: string, value: string, keyText: string, valueText: string): void;

The function parameters behave like those of the `createDataContextWithText` method. As in `addOrReplaceDataContext`, however, the context is also added or changed directly.

addOrReplaceEnvironmentContext(key: string, value: string): void;

The parameters work in the same way as `createEnvironmentContext`. However, they are added or modified directly.

removeDataContext(key: string): void;

Removes the data context with the *key* from the context space of the comment widget.

removeEnvironmentContext(key: string): void;

Removes the environment context with the *key* from the context space of the comment widget.

setSelectedData(selectedData: SelectionContext[]): void; (ab Version 1.20224.1)

The data selected by click in a graphomate visualization can thus be transferred to the comments widget to create data contexts.

setDynamicContexts(dynamicContexts: DynamicContext[]): void; (ab Version 1.20231.0)

Contexts created via the **create*Context** methods can be used here to reset the entire dynamic contexts. The previous status is overwritten in the process.

addOrReplaceContexts(contexts: DynamicContext[]): void; (ab Version 1.20231.0)

Setting several contexts at the same time leads to inconsistencies when using **addOrReplace*Context**. This method provides a remedy because it enables several contexts created via **create*Context** to be transferred to the widget in parallel.

removeContexts(contexts: DynamicContext[]): void; (ab Version 1.20231.0)

To remove several contexts at the same time, this method should be used to pass contexts created with **create*Context** to this method instead of individual calls to **remove*Context**. These are then removed from the state of the widget. In the **create*Context** method, the *key* or the *value* can also be set as an empty string. Then all contexts to which only the defined *key* or *value* applies are removed.

Examples

To set a data context depending on the selection of a value in a drop-down field, the following code can be adjusted:

```
1 var countryMember = Country_Dropdown.getSelectedKey();
2 if (countryMember === "None") {
3     Chart_1.getDataSource().setDimensionFilter("Country_Region", []);
4     graphomate_comments_1.removeDataContext("Country_Region");
5 } else {
6     Chart_1.getDataSource().setDimensionFilter("Country_Region", countryMember);
7     graphomate_comments_1.addOrReplaceDataContext("Country_Region", countryMember);
8 }
```

The following code sets the information about a data point selected in a graphomate chart as contexts for the comments widget:

```
1 graphomate_comments_1.setSelectedData(graphomate_charts_1.getSelectedData()); // ab Version 1.20224.1
```

Multiple contexts can be changed or added at the same time as follows

```
1 var regionContext = graphomate_comments_1.createDataContext("region", "North");
2 var countryContext = graphomate_comments_1.createDataContext("country", "Germany");
3 graphomate_comments_1.addOrReplaceContexts([regionContext, countryContext]);
```

To remove the contexts added above at the same time, you can do the following

```
1 var regionContext = graphomate_comments_1.createDataContext("region", "North");
2 var countryContext = graphomate_comments_1.createDataContext("country", "Germany");
3 graphomate_comments_1.removeContexts([regionContext, countryContext]);
```

If the two contexts above are to be removed, regardless of what value they are set to, empty strings can be used:

```
1 var regionContext = graphomate_comments_1.createDataContext("region", "");
2 var countryContext = graphomate_comments_1.createDataContext("country", "");
3 graphomate_comments_1.addOrReplaceContexts([regionContext, countryContext]);
```

To reset all the dynamic contexts that have been added and to remove the ones that were previously set, proceed as follows:

```
1 // first, we add a context for the year just for clarification
2 graphomate_comments_1.addOrReplaceDataContext("year", "2023");
3 // then we set dynamic contexts for region and country
4 var regionContext = graphomate_comments_1.createDataContext("region", "North");
5 var countryContext = graphomate_comments_1.createDataContext("country", "Germany");
6 // by using setDynamicContexts instead of addOrReplaceContexts, the context for the year is no longer set.
7 graphomate_comments_1.setDynamicContexts([regionContext, countryContext]);
```

Admin UI

The Admin UI is a web interface that is delivered with the graphomate comments server backend. This can be reached via the URL of the server with the addition "/commenting/administration/". Alternatively, there is a button with the label "ADMIN" below the input field for the server URL in the component, which also provides access to this interface.

It serves both as an overview and export facility for the existing comments, as well as for carrying out administrative tasks relating to user administration and authorization.

A user without the role "admin" only sees the item "Comments" and thus has the possibility of an overview of the comments open for him.

However, if the role "admin" is assigned to him, the additional menu items "Contexts", "Users", "User Roles", "User Groups" and "Authorizations" appear.

An explanation of the individual sub-items follows on the next sub-pages.

Comments

Every user, even those without the user role "admin", has the possibility here to obtain an overview of already existing comments. Only those comments are displayed that the user is authorised to view.

By clicking on a column header, the comments can be sorted according to this property.

Menu bar

The following options are available in the menu bar under the heading "Comments":

1. **Columns:** The visibility of table columns can be restricted and extended, here. In addition to the standard columns, the comment ID and the time and user of the last change can be displayed.
2. **Filters:** The list of comments displayed can be restricted here. Currently available is a restriction to 'author', 'editor', 'creation time' and 'modification time'.
3. **Export:** There are two export options. Print the current page or export to CSV. The CSV export includes all comments and table columns, including those not displayed.
4. **Refresh:** Clicking this button will load new data from the server.
5. **Add:** This allows you to create comments "manually". As contexts must also be defined manually, this feature is more for testing purposes than for productive use.

Tabellenspalten

1. **ID:** Not visible in the standard, the ID of the comment is shown here. This is a randomly generated UUID.
2. **Creation:** The time the comment was written.
3. **Changed:** Not visible in the standard, the time of the last change.
4. **Text:** The text of the comment without formatting.
5. **Creator:** The username of the author of the comment.
6. **Changed:** Not visible in the standard, the name of the user who made the last change.
7. **Contexts:** The list of contexts associated with the comment.
8. **Actions:** Buttons for editing and deleting a comment.

Edit

After clicking the Edit button, the comment editor familiar from the components opens. Click on the floppy disk icon to save the changes.

Page selection

At the bottom right is a menu that allows you to set the number of comments displayed per page and the current page.


Contexts

Users with the admin role will see an overview of existing contexts. These can only be deleted if the context has no comment assigned to it.

Contexts are created automatically when comments are created. However, if you want to use contexts in advance, you can create them here by clicking on "ADD". This is useful to create authorizations based on contexts before comments exist.

Users

Users already created or defined by importing from an external user administration are shown here.

A user can be edited by clicking on the edit pencil. Text fields can be edited by clicking on the text. For Roles and Groups, it is possible to remove Roles already added by pressing the  or to add new Roles by using the Add Dropdown.

Click the (+) Add button to create a new user. The Username and Initial Password fields are mandatory. The initial password must be changed by the user at the first login. An email address is required to use the automatic email notification feature when a user is mentioned in a comment. Roles and Groups are required if permissions are based on them.

Clicking on the "(+) JSON" button opens a window in which multiple users can be created together. The JSON format required is shown in the example below:

```
1  [
2    {
3      "username": "Meier",
4      "password": "secretPassword"
5    },
6    {
7      "username": "Michaelsen",
8      "password": "nobodyKnows",
9      "firstName": "Bernd",
10     "lastName": "Michaelsen",
11     "email": "bernd.michaelsen@example.com",
12     "roles": ["viewer", "editor"],
13     "groups": ["CEO"]
14   }
15 ]
```

User Roles

User Roles are shown here and can be added by clicking on "(+) ADD" button. All that is required is a name of your choice.

These User Roles can then be assigned to users on the Users page.

These can be used in authorizations to assign rights to the users of this role.

User Groups

User Groups are shown here.

Click the (+) ADD button to add a new group with a name of your choice.

These groups can be assigned to users on the Users page.

Groups can be used in authorizations to assign rights to users in a group. To assign entire user roles to a group, you currently need to use the Keycloak UI itself.

Authorizations

Under the item "Authorizations" in the backend admin UI, you can define which users, user roles and user groups are authorized to comment on a context combination. These settings are only available to users with the Admin role.

Initial configuration

When graphomate comments is shipped, three user roles are automatically created: "admin", "editor" and "viewer".

The corresponding rules are already predefined in the "Authorizations" section. The first rule prohibits all users from interacting with the comments, so the following rules represent a so-called *whitelist*.

The second rule defines that users with the role "admin" or "editor" are allowed to see, create, edit and delete all comments.


The last rule defined is that users with the role "viewer" are only allowed to see all comments.

Of course, you can adapt these configurations as you wish for your scenarios.

Add authorization

A new authorization can be added by clicking on the + ADD button in the menu bar above the list of existing authorizations. It should be mentioned in advance that if no options are selected for a property of the authorization, the authorization applies to all characteristics of this sub-setting.

A special case here is the setting of users, user roles and user groups. Only if all three are left blank does the rule apply to all users.

An authorization consists of several sub-settings that can be configured when adding a new authorization and when editing an existing one. When creating an authorization, several items can be selected individually by selecting their checkboxes or collectively by selecting the upper checkbox next to the title. Clicking on the arrows will either assign them to the authorization (they will appear on the right hand side) or deselect them (they will appear on the left hand side). In the edit view of an existing authorisation, selected sub-permissions can be removed by clicking on the  in the right-hand margin, and new ones can be added using the drop-down box. In both cases, a search is available.

Authorization properties

Name

The freely definable name of the authorization is used, for example, to summarise the functionality.

Type

The selection between "Allow" and "Deny" defines whether this authorization rule allows or prohibits the selected actions for a context combination. This allows rules to be defined by combination according to the scheme "everything except" and "exclusively for".

Actions

The actions that can be performed in relation to comments are divided into four rights:

1. "view": Existing comments can be read by authorised users.
2. "create": Allows users to create new comments.
3. "edit": Comments that already exist can be edited.
4. "delete": Comments are allowed to be deleted.

If no actions are selected, this release applies to all actions. The difference to selecting all actions is that without a selection, actions that could be added in later versions are also directly released.

Contexts

This area defines for which context combination the authorisation is to be applied. The representation begins with an "env" for EnvironmentContext or a "dat" for DataContext. This is followed by the key of the context before the "=" and the *value* after it.

Contexts that are selected here also apply in principle to more specific context combinations. This means that as soon as a comment contains at least the selected contexts, the rule applies. For example, if a user is authorized for the context combination "Region=Central" and "Country=Germany", he or she can access comments for the context combination Central+Germany and "Year=2021". However, access is not allowed for a comment that applies, for example, to Central+Austria.

If no contexts are selected, the rule applies to all comments.

Users

Individual users for whom authorization applies can be selected here.

If no user, user role or user group is selected, this rule applies to all users.

Roles

Authorization can be assigned here to all users in one or more user roles.

Groups

Just like the "Roles", entire user groups can be selected here.

Order

In the column "Order" on the overview page of the authorizations, the order of the rules can be changed by clicking on the arrows. The rules are processed sequentially, starting with the lowest index. This means that if, as in the initial configuration, a prohibition is initially defined for everything, this prohibition can be weakened again by individual rules with a higher index. The initial configuration written in textual form thus reads:

Initially, prohibit all users from interacting with comments. However, allow "view", "create", "edit" and "delete" for users with the role "admin" or "editor". In the last step, allow viewing of comments for users with the role "viewer" to view any comments.

Another hypothetical example: If at the end of the list of authorizations with the highest order there was a rule that prohibited everything, all previous authorizations would have no effect.

Keycloak

If "/auth" is appended to the server URL, the Keycloak UI will open, where all user administration settings can be made. This includes settings that can be made in the rest of the admin UI as a clearer shortcut.

There is a link to the 'Administration Console' and 'Documentation' on the home page. Clicking on 'Administration Console' will prompt the user to authenticate if they are not already logged in. A user with the 'admin' role will have full rights here.

The most important settings are briefly described below. For full documentation, please refer to the [Keycloak documentation](#).

Realm Settings / Master / Login

Features such as remembering a user after a browser restart and the Forgot Password function can be enabled here. For the latter, it is necessary to configure an SMTP host under **Email**.

Roles

Roles can be created here in the same way as in the Admin UI. **Default Roles** allows you to define which default roles will be assigned to a new user. For example, new users can always be allowed to see all comments by automatically assigning the role "viewer".

Identity Providers

External user administrations such as an Active Directory via SAML or OpenID Connect can be connected here. Roles and groups can be mapped here from the external system to internal roles and groups.

User Federation

This menu item allows you to import users from an external system. For example, you can connect to an Active Directory via LDAP. The difference with Identity Providers is that users are not just imported on first login, but are available in the user list. Roles and groups can also be mapped from the external system to internal roles and groups.

Authentication / Password Policy

Define requirements for strong passwords.

Groups

Groups can be created and edited here in a similar way to the Group settings in the Admin UI. It is also possible to assign entire user roles to a group under **Role Mappings**.

In this way, you unite a collection of roles, which then jointly receive authorisation via the Authorizations. Under **Default Groups**, it can be defined that new users are automatically assigned to groups.

Users

Users can be created and edited here, similar to the user settings in the Admin UI. Under **Credentials**, the password of a user can be reset. The "Temporary" switch ensures that the user has to change this password again at the next login.

Runtime interactivity

The graphomate comments can be extensively edited at runtime. If you have already maintained the URL to the comments backend in our GPS, a field for the initial login will appear when the comments widget is loaded. If you are already logged in, this window will not appear. There you can login as the initial administrator or, if you are already created users, as any user in our backend. If the login was successful, a note appears if you have not yet set a context for the comment that a context must first be defined. A comment must always be written in a context. The definition of the context can be done in the GPS and by scripting.

If a comment does not exist for the current context, a message will be displayed. Clicking on the 'Create' button opens an editor for writing a comment. The same applies if a comment already exists. Clicking on the "Edit" button opens the editor again and the comment can be edited. In addition to formatting options for the text, the editor offers the possibility of embedding hyperlinks in the text in its menu bar.

If another user is mentioned in the text with the character "@" followed by their username, they will receive an email about the comment if the email configuration in the backend is correct. The available users with matching names are suggested as soon as at least two letters are written with the "@" character. When the editor is open, there are buttons below the text input field to save, delete or cancel the comment.

The blue "corner" in the top right corner opens a menu when clicked. It shows the current user and allows you to log out.

Below the current comment you can see if there are more specific comments about the current context. A comment is more specific if it contains further restricting contexts in addition to the currently defined set of contexts. For example, in a comment widget that contains the context 2017 but no month, under more specific comments, those that were maintained for a specific month in 2021 are displayed.

By clicking on the reference to more specific comments, the list of additional comments opens. These are sorted according to how many additional contexts they contain. The additional contexts that apply to the comment are shown under the respective comment. Clicking on an additional context adds it to the filter of the current context set. If the current contexts are displayed (configuration in the GPS), this filter can be removed again by clicking on the additional context in the list.

Know Issues

At the moment there are no Known Issues

Changelog

Version 2023.2.X

Frontend

- New editor options "Title" and "Hyperlink"
- Enable and disable features based on authorizations
- "Are your sure?" dialog for deleting comments
- Scrolling fixed
- Base font size configurable
- UI improvements

Backend

- Database scheme normalized to unique contexts
- Routes secured on backend side based on authorizations by contexts
- Visible comments filtered by authorizations by contexts
- Definition of user roles in Admin UI
- Definition of user groups in Admin UI
- Assingment of user groups to user in Admin UI
- Definition of "Authorizations by Contexts" in AdminUI
- User JSON import in Admin UI
- Backend UI improvements

Version 2023.1.X

- UI improvements
- Performance improvements
- Configurable visibility of contexts
- More specific comments display configurable
- Font family configurable
- Initial text for unset comment configurable
- User edge hidable
- Editor text formatting as title
- Editor text with embedded hyperlink
- Compatibility with data point and visualisation comments improved

Backend

- CSV Export with better formatting and more information
- Filter von Author and Date fixed
- Pagination showing strange result lists fixed

