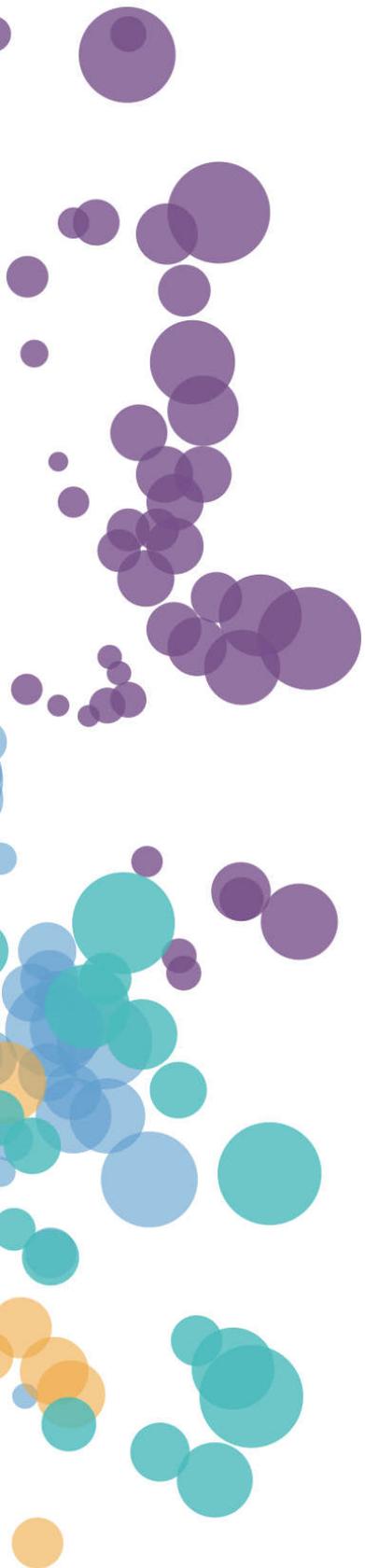




WHAT'S NEW AND RELEASE NOTES

Release: 2022.1



IN THIS RELEASE

NEW FEATURES AND IMPROVEMENTS	2
DATA PREPARATION.....	2
Dataset names when using other BI tools.....	2
Rename individual columns.....	3
Select schemas for data sources	4
Save your viewing preferences.....	5
Automatically resolve duplicate names for the imported datasets	6
Select private views to create TM1 cube view datasets.....	7
Double-click to select columns for data exploration	8
Create a dataset based on a TM1 MDX view.....	9
STORYBOARDS.....	10
Custom visualization widget.....	10
Python code widget.....	11
Schedule a storyboard subscription to run every 15 or 30 minutes	12
Send a storyboard subscription email on demand.....	13
Save your viewing preferences.....	14
Distinguish shared datasets in a visualization	15
Edit a storyboard without running the widgets	16
Receive notifications about new versions	17
INSTALLATION & CONFIGURATION	18
Configure email for notifications.....	18
Configure Data Engine's memory settings	19
REST API.....	20
OpenAPI specification & Swagger	20

NEW FEATURES AND IMPROVEMENTS

DATA PREPARATION

Dataset names when using other BI tools

Previously, when accessing DataClarity datasets from a third-party BI tool like Tableau or Power BI, you had all private and shared datasets displayed under a single public schema. To avoid possible name duplicates, all the datasets' names had an ID automatically appended. For example, "Sales Orders" was displayed as "sales_orders_1234". The approach has been changed to reflect the original names as displayed in Data Preparation. Thus, starting with this release, you will find datasets displayed under different schemas based on the dataset ownership, for example:

- george.becker.Sales Orders (a dataset created by your user George Becker)
- johnsmith.Sales Orders (a dataset with the same name that was shared by John Smith)

The screenshot shows the DataClarity interface. On the left, the 'Connections' panel shows a connection to '10.15.16.201' (PostgreSQL). Below it, the 'Database' is 'datasets'. The 'Table' list includes various datasets such as 'IBEX35_CONSOLIDATED_extract', 'Marketing Expenses', and 'Sales Orders'. The 'Navigator' window is open, showing a search for 'george.becker.Sales Orders'. The table view displays the following data:

	Sales per transaction	No of customers	Quantity	Unit Price	Discount
11	225	7	9	20	
5	561	5	43	28	
8	835	10	46	56	
14	130	13	14	4	
2	97	1	48	6	
7	292	15	39	4	
13	113	18	37	56	
15	480	8	34	881	
9	313	9	3	44	
15	813	14	47	44	
15	147	9	28	401	
9	1008	10	7	2550	
16	77	19	25	3	
7	394	6	48	9	
20	65	23	20	6	
10	89	9	33	180	

Rename individual columns

In addition to renaming all the columns in bulk on the **Choose tables to import** page, you can now rename each column individually when adding a new data source.

The screenshot displays the 'CHOOSE TABLES TO IMPORT' window, which is divided into two main sections: 'TABLES' on the left and 'SELECTED ITEMS' on the right. Both sections have a search bar at the top. The 'TABLES' section lists various categories like Customers, Locations, Products, and Measures. The 'SELECTED ITEMS' section shows a hierarchical view of these items, with 'Measures' expanded to show items like 'Items Per Transaction', 'Sales Per Transaction', and 'No Of Customers'. A blue box highlights the 'Items Per Transaction' item, and a mouse cursor is clicking on a 'Rename column' icon next to it. A 'CLEAR ALL COLUMNS' button is located at the bottom of the 'SELECTED ITEMS' section. A 'RENAME COLUMNS' dialog box is open in the foreground, showing a text input field with the text 'Items per Transaction' and 'CANCEL' and 'OK' buttons.

Select schemas for data sources

Starting with this release, you need to specify a database schema when adding a data connection to the following sources:

- Microsoft SQL Server
- PostgreSQL
- DB2
- Apache Derby
- Google Cloud SQL
- PostgreSQL

By selecting a schema, you narrow down the data for the data connection and improve query engine performance. After you enter server credentials, click **Load list** to view available schemas in the database and select one. If you edit a data connection created before this release, you will be prompted to select a schema.

ADD DATA CONNECTION [X]

Connection details Caching

Microsoft SQL Server
Microsoft SQL Server data source connection

Server: 10.15.16.888

Port: 1433

Database: database

Username: user

Password:

Schema: **dbo** [Load list]

Ask user for credentials: Off

Connection name: Microsoft SQL Server 10.15.16.888

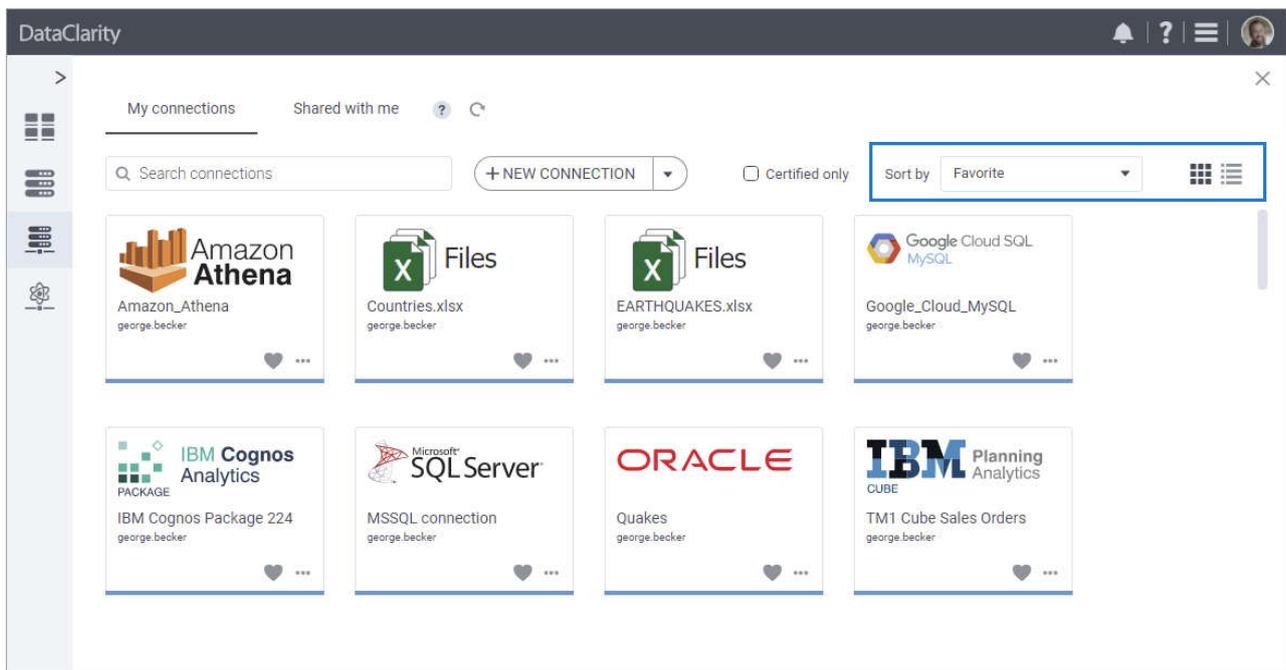
Test your connection
After you fill in the data connection details, test your connection. If something goes wrong, an error message will be displayed.

SAVE **TEST CONNECTION**

Save your viewing preferences

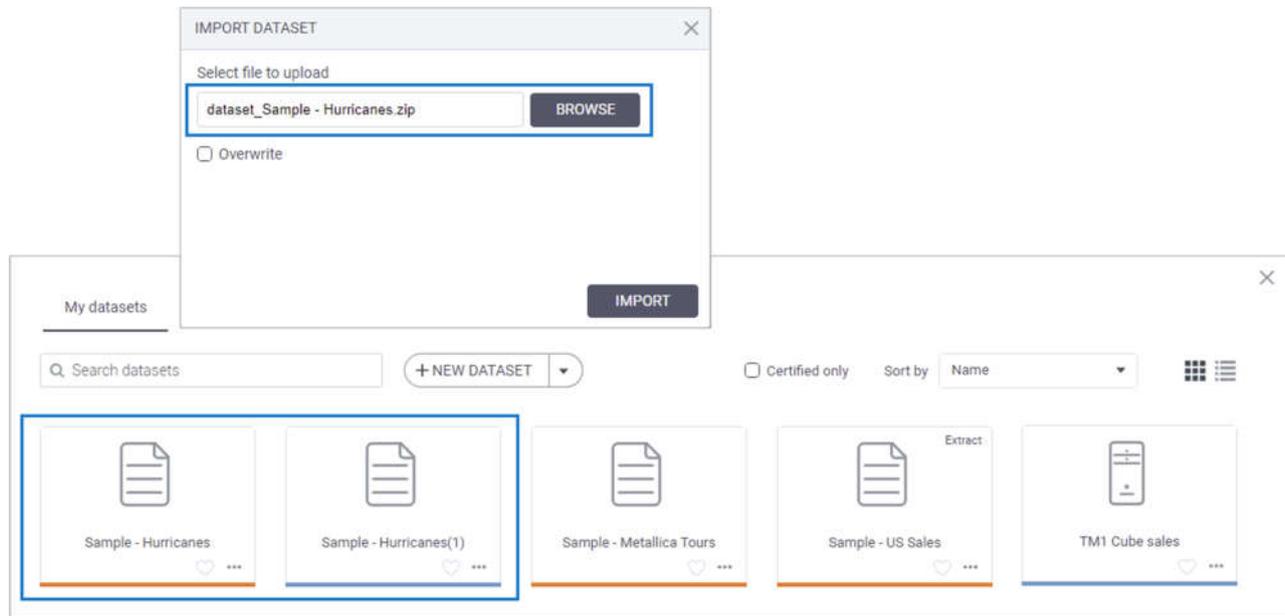
Previously, when you switched to **List view** or selected a different sorting option to view datasets, the selections were saved only within the user session. In other words, the view and sorting were restored to the default values with each subsequent login. The user experience has been improved by saving your viewing preferences using the browser's cookies. Additionally, the default sorting has been changed to **Last created** to have the most recent resources listed first.

The same improvements are applied to data connections and AI connections.



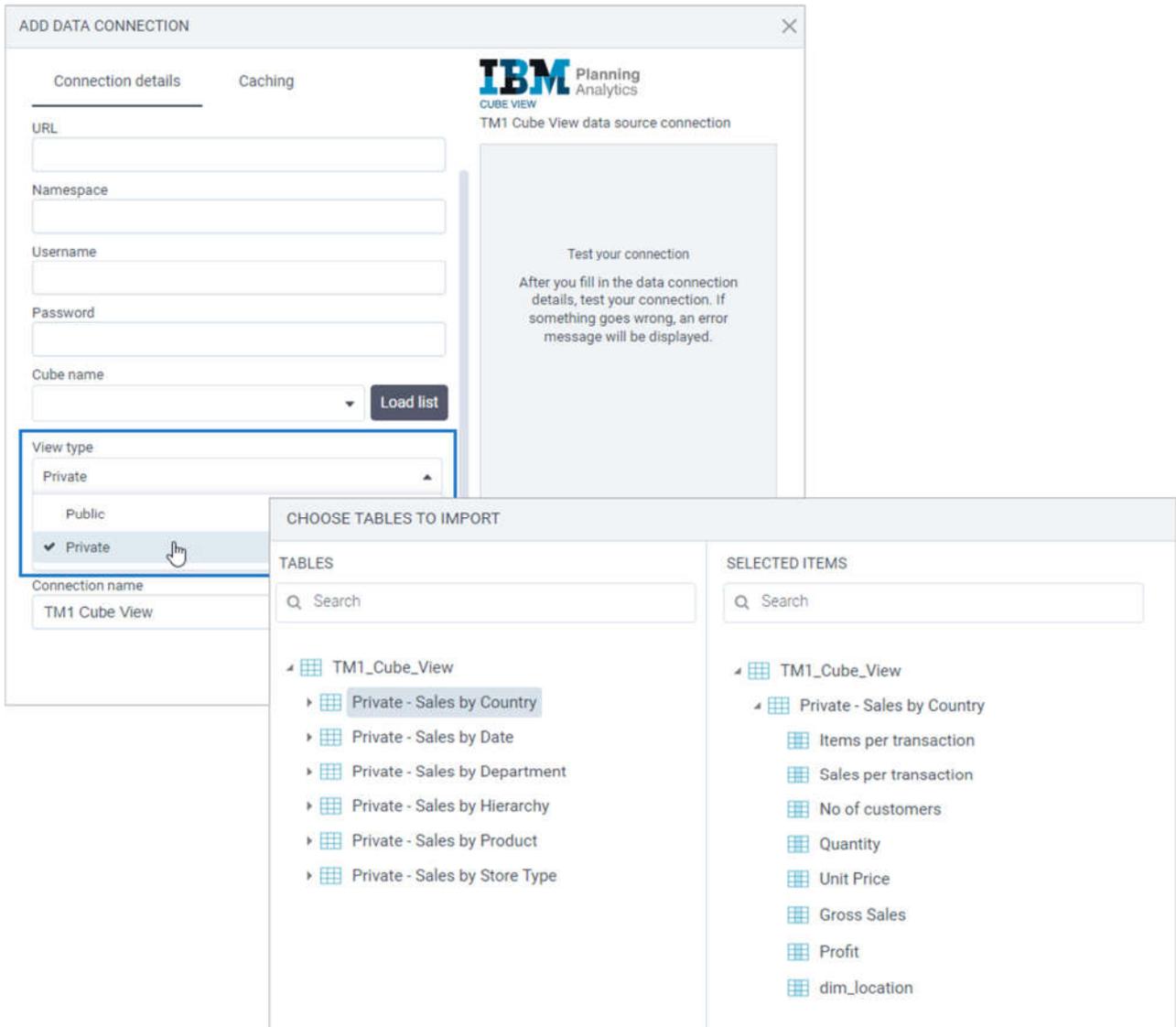
Automatically resolve duplicate names for the imported datasets

Now, if you are importing a dataset with the name that already exists in the **Datasets** pane, the name of the imported dataset will automatically include an index number in parentheses. For example, if you are importing the dataset named “Sales” and you already have a dataset with this name, it will be imported as “Sales(1).”



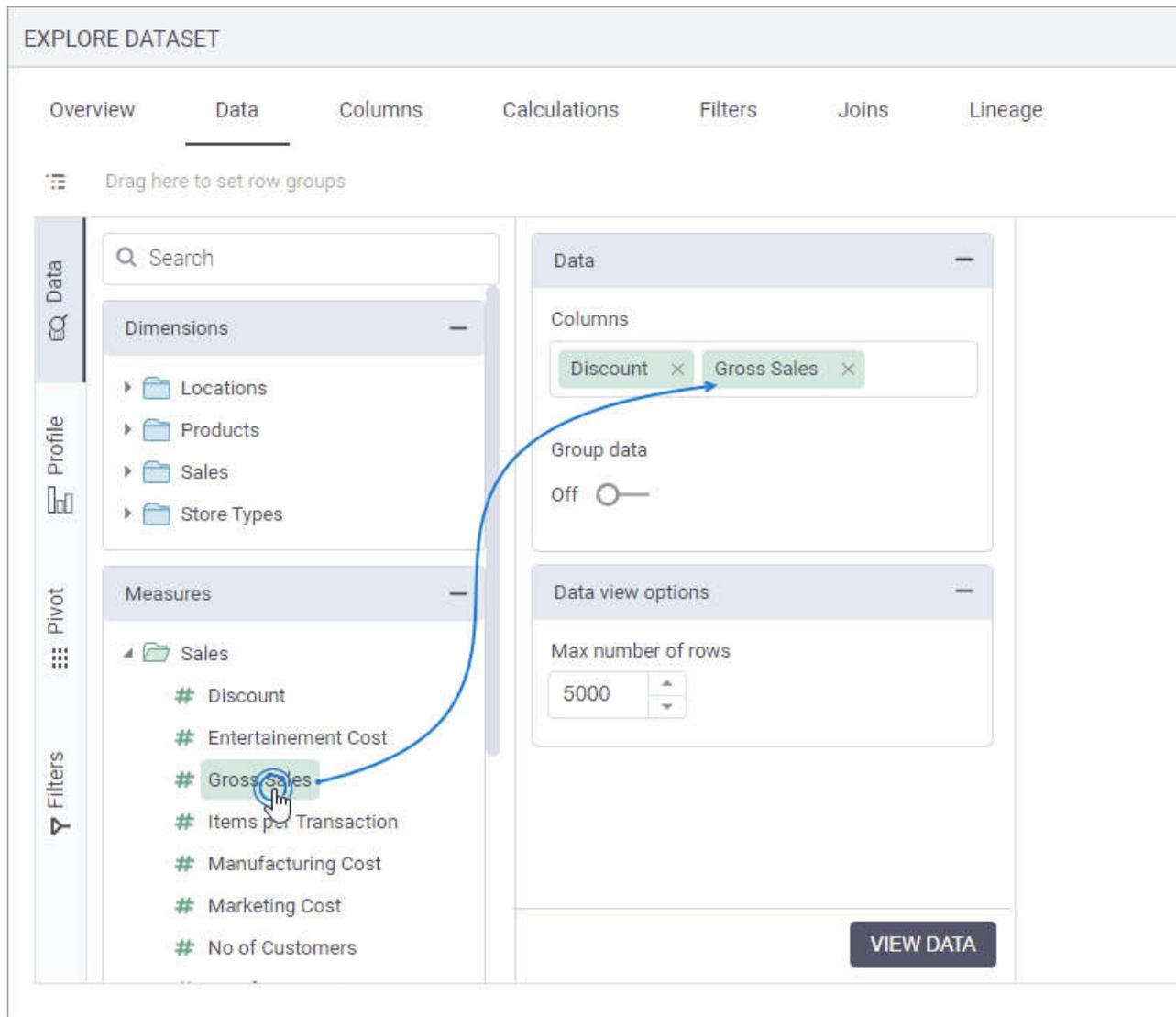
Select private views to create TM1 cube view datasets

Previously, after creating a data connection to a TM1 cube view, you could select a public view for a dataset. Now, you can create data connections with access to your private views. The new **View type** dropdown allows you to select the type of cube view: **Private** or **Public**. The **Public** option is selected by default.



Double-click to select columns for data exploration

Previously, to select columns for data exploration, you needed to drag them into the **Columns** field of the **Explore dataset** pane. Now, you can quickly add columns by double-clicking them under the **Dimensions** and **Measures** sections.



Create a dataset based on a TM1 MDX view

Previously, you could create datasets based on native TM1 cube views. The TM1 cube view driver has been extended to support MDX views (named MDX expressions stored inside the TM1 Server). Now, after creating an **IBM Planning Analytics / TM1 Cube View** connection, you can select MDX views for your datasets. Moreover, you can select attributes for MDX views the same way as you do for native TM1 views. The attributes are listed as columns that follow the "column_name @attribute_name" naming pattern.

The screenshot shows a 'TABLES' interface with a search bar and a tree view of TM1 MDX views. The tree structure is as follows:

- TM1_Cube_View
 - Sales by Store Type
 - Unit Price
 - Discount
 - Gross Sales
 - Manufacturing Cost
 - Profit
 - Planned Gross Sales
 - Planned Profit
 - Store Type
 - Store Type @Code
 - Store Type @Description
 - Store Type @Parent
 - Sales by Date
 - Sales by Location
 - Sales by Product Line

The items 'Store Type @Code', 'Store Type @Description', and 'Store Type @Parent' are highlighted with a blue box.

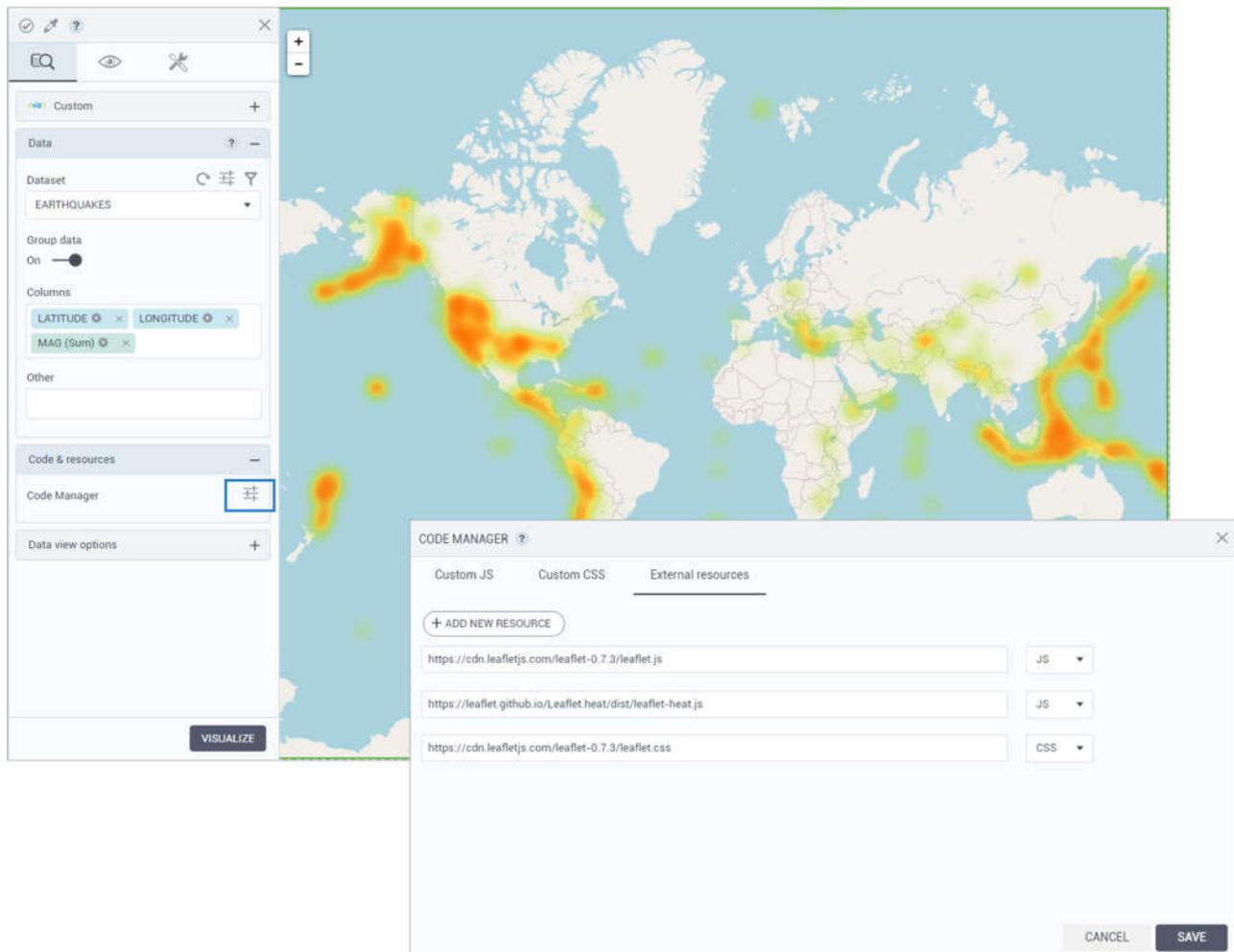
STORYBOARDS

Custom visualization widget

If you want to visualize your data in a specific way that is not possible with the visualizations included in Storyboards, you can build custom visualizations. The new **Custom viz** widget allows you to write custom JavaScript code by using popular third-party visualization libraries such as D3, amCharts, or Charts.js.

To create a custom visualization, select data columns that generate the query, and then write the code to process and visualize the data in the **Code Manager** using the following tabs:

- **Custom JS** – Add your custom JavaScript code.
- **Custom CSS** – Specify custom CSS code.
- **External resources** – Link the JavaScript and CSS resources from an external library.



The screenshot displays the DataClarity interface. On the left, a sidebar contains a 'Custom' widget configuration panel. The 'Data' section is set to 'EARTHQUAKES'. Under 'Columns', 'LATITUDE', 'LONGITUDE', and 'MAG (Sum)' are selected. The 'Code Manager' tab is active, showing a list of external resources. The main area shows a world map with a heatmap overlay, indicating earthquake activity. The 'Code Manager' dialog box is open, showing the 'External resources' tab with three entries:

Resource URL	Type
https://cdn.leafletjs.com/leaflet-0.7.3/leaflet.js	JS
https://leaflet.github.io/Leaflet-heat/dist/leaflet-heat.js	JS
https://cdn.leafletjs.com/leaflet-0.7.3/leaflet.css	CSS

Buttons for '+ ADD NEW RESOURCE', 'CANCEL', and 'SAVE' are visible in the dialog box.

Python code widget

Starting with this release, power users can execute Python code on a storyboard's page. The code is executed directly on the built-in Python server and allows using the results to feed other custom visualizations and, therefore, enhance the data analysis.

You can find the new widget in the **Widgets** pane, on the **Other widgets** tab under the **Web & Code** category. You can add your code by clicking **manage python code** on the **data** tab. To preview and verify the code results, click **Execute**. If you do not want to run the code when your storyboard is in Edit mode, turn on **Execute in View mode only**. This way, switching a storyboard to View mode will be a trigger to execute the code.

The screenshot displays the DataClarity interface with a Python code widget being configured. The main canvas shows a large Python logo. On the left, the 'Data' pane is open, showing the 'Python code' widget with the 'Execute in View mode only' toggle set to 'Off'. On the right, the 'WIDGETS' pane is open, showing the 'Web & Code' category with the 'Python' widget selected. A 'PYTHON CODE' dialog box is open in the foreground, containing the following code:

```
import datetime
import time
from datetime import datetime, timedelta
import os
import csv
import json
import requests

url = 'https://firms.modaps.eosdis.nasa.gov/api/area/csv/de5657f1c6c0c8d26e629bbebbaeeceb/MODIS_NRT/world/1'

with requests.Session() as s:
    download = s.get(url)

    decoded_content = download.content.decode('utf-8')

    cr = csv.reader(decoded_content.splitlines(), delimiter=',')
    my_list = list(cr)
    for row in my_list:
        print(row)
```

At the bottom of the dialog box, there are three buttons: 'EXECUTE', 'CANCEL', and 'SAVE'.

Schedule a storyboard subscription to run every 15 or 30 minutes

With a storyboard subscription, the subscribers receive emails with storyboard snapshots on a scheduled basis. Previously, you could schedule a subscription on a monthly, weekly, daily, and hourly basis. Starting with this release, you can schedule subscription emails to run every 15 or 30 minutes and select specific days of the week.

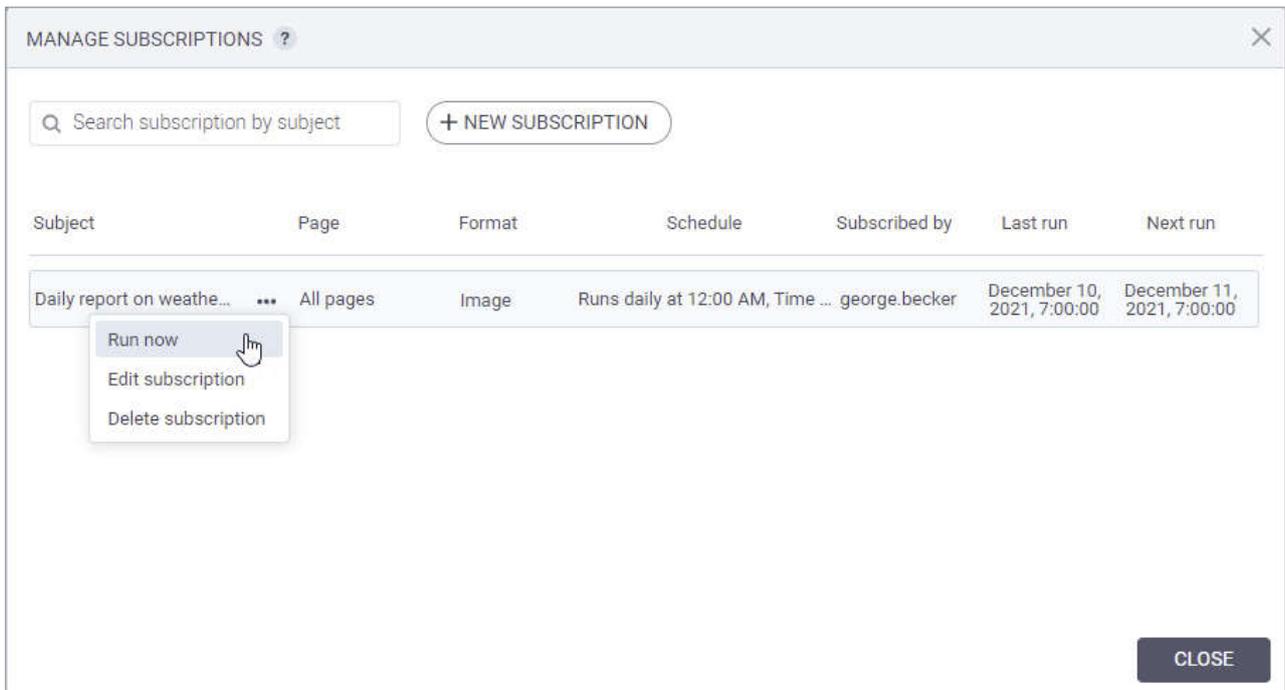
To set the new frequency, open the **Subscribe** dialog, click **Show advanced options**, and in the **Runs** dropdown, select **Minutes**.

The screenshot shows a 'SUBSCRIBE' dialog box with the following fields and options:

- Subject:** Weather conditions update
- Message:** Hi \${first.name} \${last.name},
Please find the latest report attached.
Best regards,
- Schedule preview:** Hide advanced options
Runs every 30 minutes on Monday, Tuesday, Wednesday, Thursday, Friday, Time zone (UTC-5:00) Eastern Time
- Runs:** A dropdown menu is open, showing 'Minutes' selected. The frequency is set to 'Every 30' minutes. A mouse cursor is pointing at the '30' option. Other options visible are '15'.
- Days:** Sunday, Monday, Wednesday, Thursday, Friday, Saturday
- Time zone:** (UTC-5:00) Eastern Time
- Buttons:** MANAGE SUBSCRIPTIONS and SUBSCRIBE

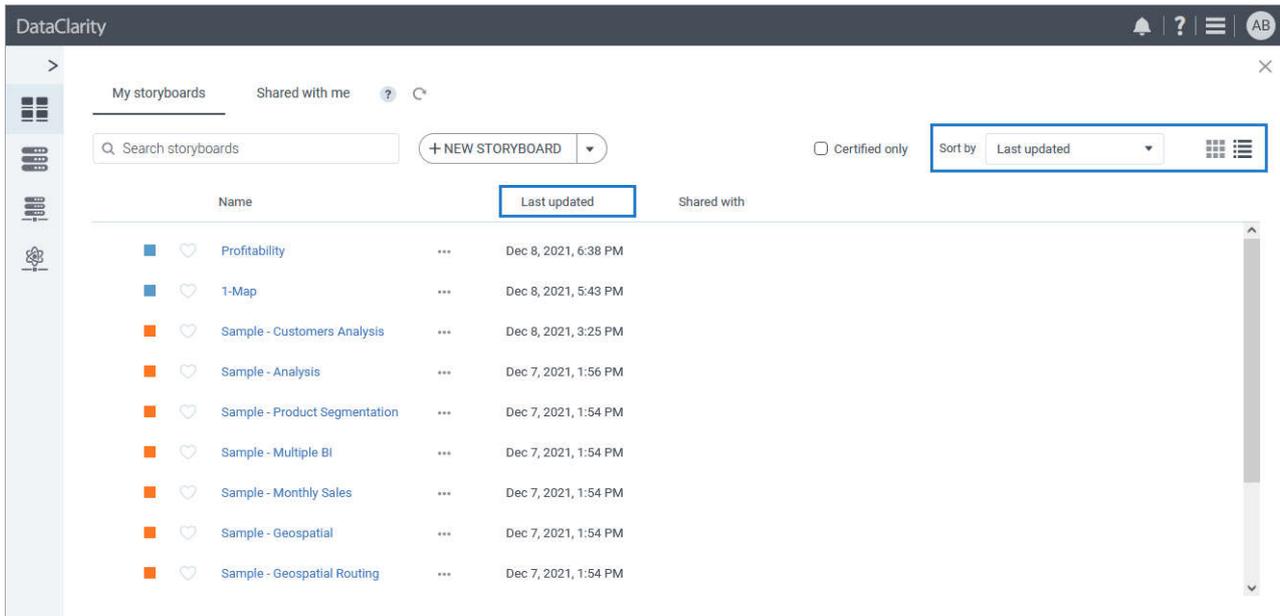
Send a storyboard subscription email on demand

Now, you can test the added subscription jobs by sending an email on demand. In the **Manage subscriptions** window, for a subscription, point to **More actions** and select the new menu option—**Run now**. After confirmation, the subscription email request is sent immediately. We recommend using this option only for testing purposes or in urgent situations.



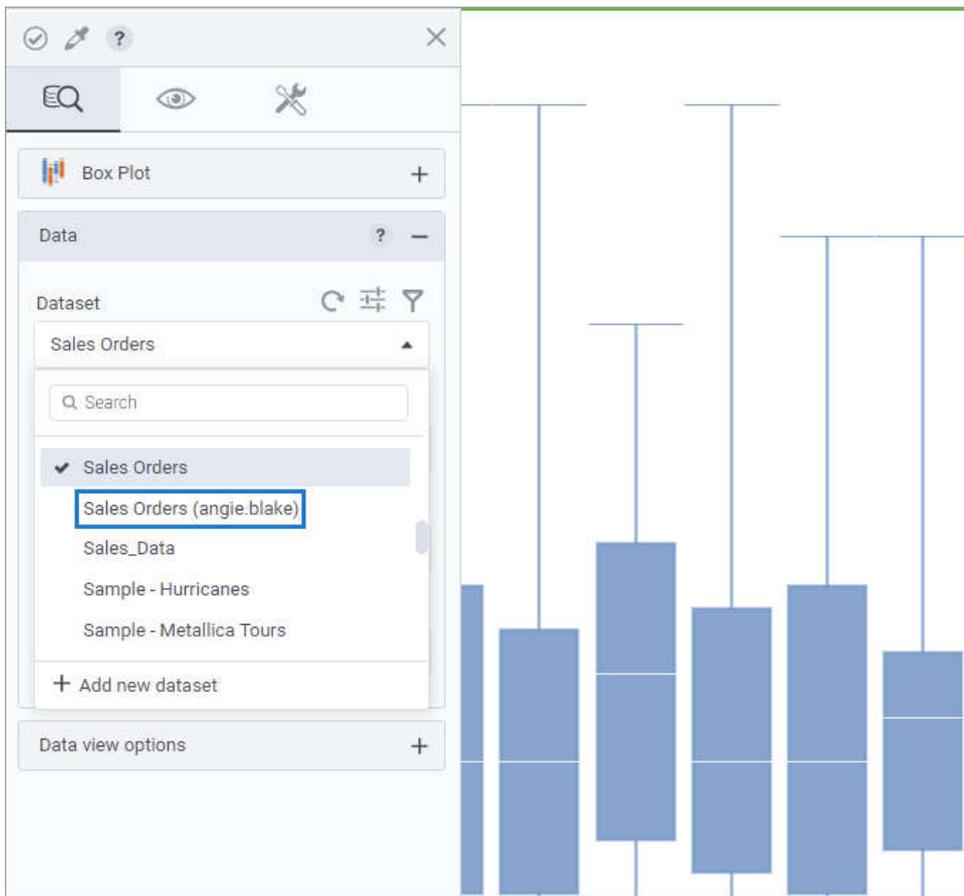
Save your viewing preferences

Previously, when you switched to **List view** or selected a different sorting option to view storyboards, the selections were saved only within the user session. In other words, the view and sorting were restored to the default values with each subsequent login. The user experience has been improved by saving your viewing preferences using the browser's cookies. Additionally, the default sorting has been changed to **Last created** to have the most recent storyboards listed first.



Distinguish shared datasets in a visualization

Previously, when selecting a dataset for visualization, you could not differentiate between your datasets and those shared with you. Now, if a dataset is shared with you, you can view its owner's username in parentheses next to the dataset name in the **Dataset** dropdown. For example, "Sales Orders (angie.blake)" is a dataset shared by the user Angie Blake.



Edit a storyboard without running the widgets

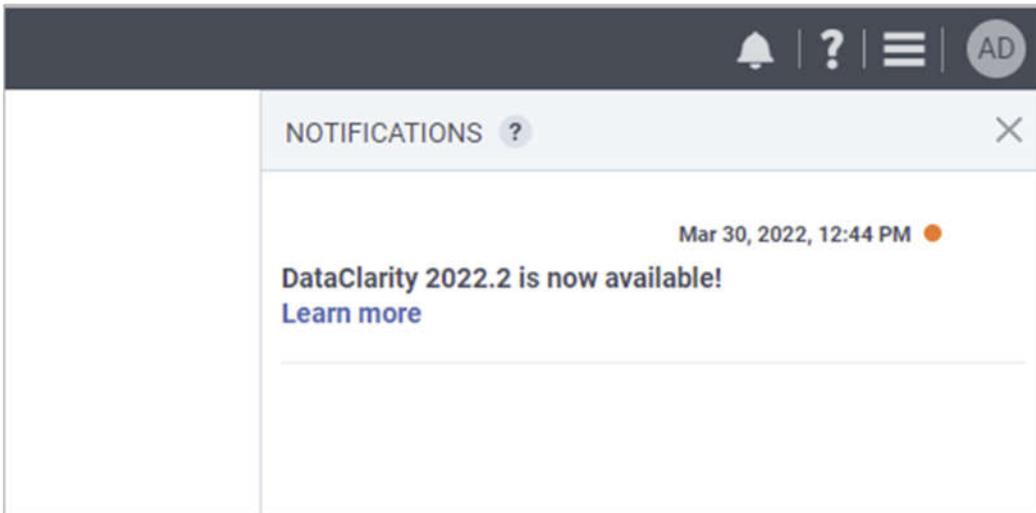
Previously, when opening a storyboard for editing, all its widgets ran automatically. As a result, for storyboards containing many widgets with complex data science calculations, any modification was time-consuming. The UX has been improved, and now, when you open a storyboard by selecting **More actions > Modify > Edit**, the visualizations do not run automatically. This way, you can quickly modify any widgets on a storyboard.

You can visualize all the widgets by switching a storyboard to View mode. You can still run each widget individually by clicking **Visualize** on the widget settings pane.



Receive notifications about new versions

Now you will receive notifications about each new version of the DataClarity Platform that is available for installation. The announcements appear in the **Notifications** pane and include a version number and the link to the *What's New and Release Notes* document. Additionally, you can specify an email for such notifications in **Configuration Manager > Notifications**, in the **Email for notifications** field.



INSTALLATION & CONFIGURATION

Configure email for notifications

You can now specify an email for receiving notifications about each new version of the DataClarity Platform that is available for you to install. The new **Email for notifications** field has been added in **Configuration Manager > Notifications**.

The screenshot shows the DataClarity Configuration Manager interface. On the left is a navigation sidebar with a search bar and a list of categories: TLS/SSL Certificates, Data Connectors, User Access, Configuration (expanded), Audit, Branding, Common, Data Preparation, Data Science, Data Server, and Notifications (highlighted). The main content area is titled 'Notifications' and contains two input fields: 'Notifications lifespan (in days) ?' with the value '30' and 'Email for notifications ?' which is highlighted with a blue border. Below these fields are 'SAVE' and 'RESET' buttons.

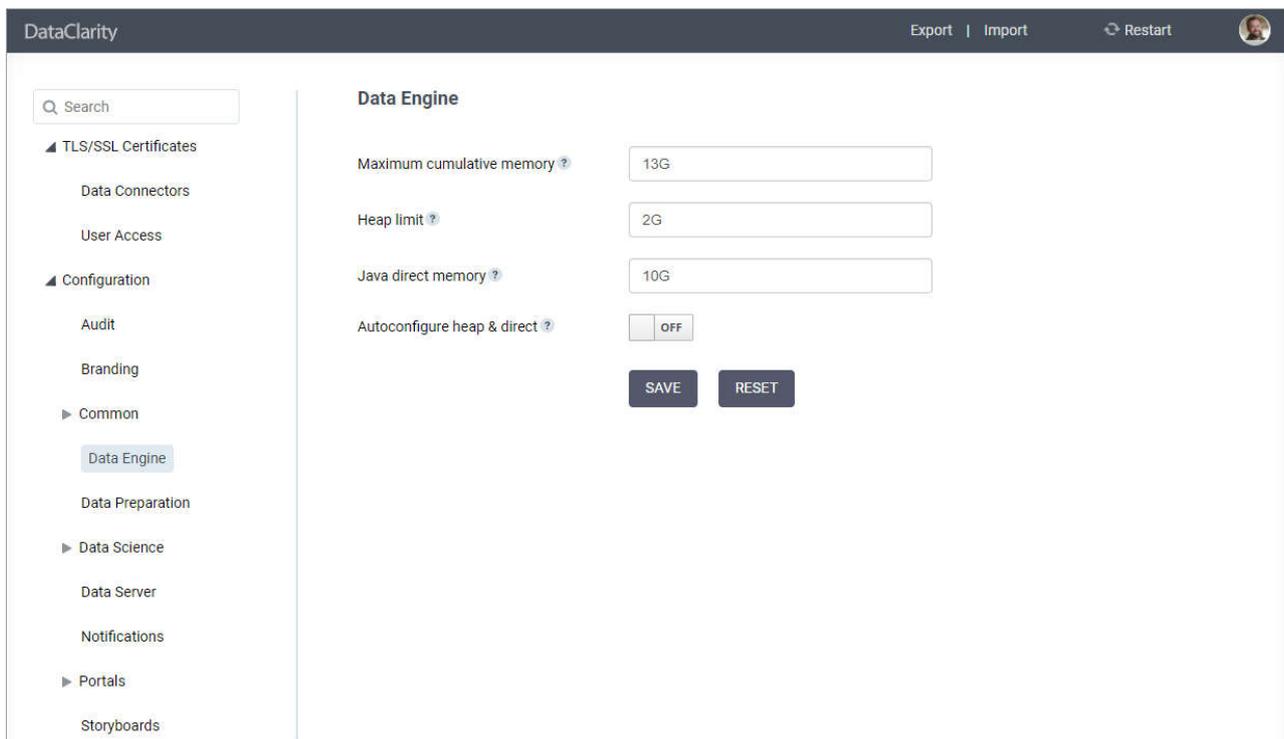
Configure Data Engine's memory settings

You can now control how much memory to allocate to Data Engine. This way, you can improve the data & query engine's speed and the platform server efficiency. In Configuration Manager, on the new **Data Engine** pane, you can find the following memory settings:

- **Maximum cumulative memory** – The maximum cumulative memory allocated to the Data Engine process during startup.
- **Heap limit** – The maximum theoretical JVM (Java virtual machine) heap limit.
- **Java direct memory** – Java direct memory allocated to query processing.
- **Autoconfigure heap & direct** – Choose how to define memory limits:
 - If **On**, Data Engine automatically determines the best allocation between heap and direct memory limits based on the specified **Maximum cumulative memory**. In this case, the values entered in the **Heap limit** and **Java direct memory** are ignored.
 - If **Off**, Data Engine uses the memory limits specified in the respective fields.

If you scale Data Engine to more pods, each pod will have the same memory limits. For example, if you have the max memory set to 16 GB, and you have two pods, then Data Engine uses 32 GB (16 x 2) as the limit.

For more information on how to allocate memory to Data Engine, refer to *Configuration Manager Help*.



REST API

OpenAPI specification & Swagger

DataClarity provides REST APIs to let you leverage, automate, or incorporate DataClarity Platform functions into your website or application. The DataClarity's API is based on REST principles and provides standard HTTP methods for getting, creating, updating, and deleting the platform's resources.

You can now benefit from the improved and restructured API reference documentation provided in Swagger, a fully interactive documentation tool that allows you to visualize and interact with API.

Swagger
Supported by SMARTBEAR

Select a definition **Data Preparation**

Data Preparation API

[/dp/swagger/DP_API_Spec.yaml](#)

API documentation to help you create and manage data connections, AI connections, and datasets.

[DataClarity Support Team - Website](#)
[Send email to DataClarity Support Team](#)
[Usage and SDK Samples](#)

Servers

{protocol}://{serverPath}/dp - Generated server url

Computed URL: `http://localhost:8080/dp`

Server variables

protocol **http**

serverPath `localhost:8080`

Authorize

- ai-connections** AI Connections (DCPY, RSERVICE, TABPY, MSR)
- audit** Audit
- certificates** Certificates
- common** Common Settings

Interact with API in Swagger

Each API request now includes a summary, description, and examples of a request body and response body where applicable. You can authenticate with the Bearer token and use the “Try it out” feature to experiment with the API before integrating it into your code.

The screenshot shows a Swagger UI interface for a POST endpoint: `/api/v1/datasources/{sourceId}/share`. The endpoint is titled "Share a data connection" and has a lock icon in the top right corner. Below the title, a description reads: "Share a data connection with the specified users or groups (recipients) with the View permission." A "Try it out" button is located in the top right of the parameters section.

The "Parameters" section is titled "Parameters" and contains a table with two columns: "Name" and "Description".

Name	Description
sourceId * required string (path)	Data connection ID to share
canShare boolean (query)	Recipients can reshare the data connection Default value : false

The "Request body" section is titled "Request body" and is marked as "required". It has a dropdown menu set to "application/json". Below this, a description reads: "A JSON object containing a list of recipients". There are links for "Example Value" and "Schema". The "Example Value" is displayed in a dark box with the following JSON:

```
[
  {
    "recipient": "angie.blake",
    "recipientType": "USER"
  },
  {
    "recipient": "/IT",
    "recipientType": "GROUP"
  }
]
```

OpenAPI

Swagger is generating the interactive API documentation based on the OpenAPI specification, namely the OpenAPI definition file version 3.0.3 in YAML format. You can download the file, view it in a text editor, or even import it in a Postman collection. The examples in the specification use sample data and credentials, so make sure you use your data for testing.

```
1 openapi: 3.0.3
2 info:
3   title: Data Preparation API
4   description: API documentation to help you create and manage data connections, AI connections, and datasets.
5   contact:
6     name: DataClarity Support Team
7     url: 'https://support.dataclaritycorp.com/hc/en-us'
8     email: 'customercare@dataclaritycorp.com'
9   version: ''
10 externalDocs:
11   url: '/dp/'
12   description: 'Usage and SDK Samples'
13 servers:
14   - url: '{protocol}://{serverPath}/dp'
15     description: Generated server url
16     variables:
17       protocol:
18         enum:
19           - 'http'
20           - 'https'
21         default: 'http'
22       serverPath:
23         # note! no enum here means it is an open value
24         default: 'localhost:8080'
25         description: this value is assigned by the service provider, in this example `prod.com`
26 tags:
27   - name: ai-connections
28     description: AI Connections (DCPY, RSERVICE, TABPY, MSR)
29   - name: audit
30     description: Audit
31   - name: certificates
32     description: Certificates
33   - name: common
34     description: Common Settings
35   - name: dataset-extract
36     description: Dataset Extracts
37   - name: datasets
38     description: Datasets
39   - name: datasource
40     description: Data Connections
41   - name: datasource-upload
42     description: File Upload
43   - name: demo-data
44     description: Sample Content
45   - name: preview
46     description: Data Preview
47   - name: tags
48     description: Tags
49   - name: user-setting
50     description: User Settings
51 paths:
52   '/api/v1/aiconnections':
53     get:
54       tags:
```

API references

API reference documentation per each component is provided with the Platform's installation, where "localhost" is the name or IP address that was configured for the Platform.

- **Data Preparation API** – <https://localhost/dp/swagger-ui/index.html>
- **Storyboards API** – <https://localhost/sb/swagger-ui/index.html>
- **Scheduler API** – <https://localhost/scheduler/swagger-ui/index.html>
- **Notification API** – <https://localhost/notification/swagger-ui/index.html>