



DOC 4.4.0

Migration Guide

December 11th, 2024

Copyright © 2012-2024 DecisionBrain S.A.S. All rights reserved.

All specifications and information regarding the products in this document are subject to change without notice and should not be construed as a commitment by DecisionBrain. DecisionBrain assumes no responsibility or liability for any mistakes or inaccuracies that may appear in this document. All statements and recommendations in this document are believed to be accurate but are presented without warranty. Users must take full responsibility for their application of any product.

DOC 4.4.0 Migration Guide

- Application Changes..... 9**
 - General Application Changes..... 9**
 - New Beta Features..... 9
 - Access Control..... 10**
 - Improved Security for the Web Client Login..... 10
 - Views & Dashboards..... 10**
 - New Configurable Buttons for the Widget Toolbar..... 10
 - New Dashboard Filters..... 11
 - New Filter Bar..... 11
 - New Filtering Scope..... 12
 - Improved Sidenav Icons..... 12
 - Workspaces & Scenarios..... 13**
 - New Scenario Types for the CDM..... 13
 - Improved Display
for the CDM..... 13
 - Improved Scenario import..... 14
- Data Changes..... 16**
 - Model..... 16**
 - New Composite Data Model Feature..... 16
 - JDL..... 17**
 - New JDL Syntax for the Composite Data Model..... 17
 - GraphQL..... 18**
 - Improved GraphQL Introspection..... 18

Built-in Import/Export	18
Updated Import for the CDM.....	18
Data Integration Framework	19
Deprecated DBPF and CSV collectors.....	19
New XCSV File Format.....	19
Data Service	19
Removed Data Service Classes.....	19
Deprecated Methods in the Data Service REST API.....	20
Updated SchemaChecker API for the CDM.....	20
Scenario Service	21
Removed Property in the Scenario Service GraphQL API.....	21
Deprecated Property in the Scenario Service GraphQL API.....	21
Deprecated Excel Import/Export Endpoints in the Scenario Service REST API.....	21
Updated Scenario Service REST API.....	21
Execution Service	22
New Dependency.....	22
Updated JSON Serialization Limit for the Execution Service.....	22
Backend Service	23
Updated JSON Serialization Limit for the Backend Service.....	23
DBOS Changes	24
Workers (Java)	24
Updated Scenario Data Format for Java Workers.....	24
Workers (Python)	27
Deprecated CSV Collector for Python Workers.....	27
New Business Key in XCSV Format for Python Workers.....	27
New Debug Configuration for Python Workers.....	27

- Dev Changes..... 28**
 - REST API..... 29**
 - Removed REST API Elements..... 29
 - Deprecated "GET" for "scenariosIdsAndNames"..... 29
 - Updated REST API Elements..... 29
 - 3rd-Party Components..... 30**
 - Improved Jackson JSON Serialization Capabilities..... 30
 - Updated NPM Dependencies..... 31
 - Updated Java dependencies..... 33
 - Updated Infra Dependencies..... 34
 - Updated Jackson Dependencies..... 34
 - Build..... 35**
 - Deprecated Old DSL of Code Replicate Plugin..... 35
 - New Angular Post Installation Script..... 35
 - New Option for Beta Features in the Application Generator..... 35
 - Improved Test Fixtures Modules..... 35
 - Updated Default Size Limit for JSON Objects..... 35
 - Updated Docker Compose Files..... 36
 - Updated Gradle Files..... 36
 - Updated Code Replicate Plugin..... 37
 - Updated Packaging Dependencies..... 38
 - Updated Helm Chart..... 38
 - Security..... 38**
 - Improved Security for Trivy CVEs..... 38
 - Deployment..... 38**
 - New Deployment Files for Jupyter Notebook..... 38
 - New Java Memory Usage Limit..... 38

Gene Online	39
New Gene Online Beta Feature.....	39
Python	39
New Python Module for Jupyter.....	39
Updated Python Requirement.....	39
Jupyter	40
New JupyterLab Integration.....	40
New Jupyter Notebook Sample for Scenario Import into a Pandas Data Frame.....	40
CPLEX	40
Updated CPLEX Integration.....	40
Documentation Chatbot	41
New Documentation Chatbot Beta Feature.....	41
Documentation	41
Updated Documentation.....	41
Scripted Tasks Changes	42
Definition	42
New Task for the ChatGPT Feature.....	42
Updated Tasks for Scenario Characteristics.....	42
Updated File Formats in Scenario Processing Method.....	42
Updated Tasks for Scenario Creation.....	42
Updated Parameter in Scenario Creation Method.....	43
Routines	43
Deprecated File Formats in Routines.....	43
New File Format in Routines.....	44
Updated DBDomCollector API in CDMs.....	44
Updated GeneIssue in CDMs.....	44

- UI Changes..... 45**
 - General UI Changes..... 45**
 - Deprecated Navigation Button Widget..... 45
 - Application Preferences..... 46**
 - New Parameter for Beta Features in Application Preferences..... 46
 - New Option for Application Controllers in Application Preferences..... 46
 - Updated Scenario Import Parameter in Application Preferences..... 46
 - Extensibility..... 46**
 - Deprecated Method in the Application Controller API..... 46
 - New Status Bar API..... 47
 - Updated Application Controller API..... 47
 - Tables..... 49**
 - Removed Deprecated Elements..... 49
 - Updated AG Grid Import..... 49
 - Updated Libraries..... 49
 - Updated Gene Table Component..... 50
 - Data Grid/Explorer Widgets..... 51**
 - New Index Bar Localization..... 51
 - New Advanced Filters (EXPERIMENTAL)..... 51
 - Improved Date Filter in the Data Grid/Explorer Widgets..... 52
 - Improved Display for Booleans in the Data Grid/Explorer Widgets..... 52
 - Workspace and Scenario Widgets..... 52**
 - Updated Scenario List Widget..... 52
 - Button Widget..... 53**
 - New Button Widget..... 53
 - New Button Handler..... 53

Gantt Chart Widget	54
Deprecated Methods in the Gantt Chart Widget Controller.....	54
New Resource Grouping for the Gantt Chart Widget.....	54
New Resource and Group Identification for the Gantt Chart Widget.....	54
New Context Menu for Gantt Chart Widget.....	54
New Data Selection for the Gantt Chart Widget.....	54
Improved Event Loading for the Gantt Chart Widget.....	54
Improved Display Options for the Gantt Chart Widget.....	55
Improved Event Labels for the Gantt Chart Widget Controller.....	55
Updated Date Parsing in the Gantt Chart Widget Controller.....	55
Filter Widget	56
New Filtering Scope.....	56
Updated Method in the Filter Widget Controller.....	56
Updated Parameter in the Filter Widget Controller.....	56
Map Widget	57
Improved Map Widget Controller.....	57
Pivot Table Widget	58
Updated Pivot Table Items.....	58
Improved Pivot Table Widget.....	58
Code Editor Widget	59
New Code Editor Widget.....	59
New Code Editor Task Input.....	59
New Code Editor ChatGPT Task.....	59
Rule Script Editor Widget	61
New Rule Script Editor Widget.....	61

-
- Each section is structured as follows: Removed, Deprecated, New, Improved, and Updated.
 - REST API updates are listed in Section **Dev Changes > REST API**.
 - Dependency updates are listed in Section **Dev Changes > 3rd-Party Components**.
 - Future removals are listed in Section **Deprecated Features and APIs scheduled for Removal**.
 - For more details, please refer to the [DOC Documentation and Release Notes](#).

Application Changes

This part details all changes related to:

- *general features on application element management, including events and hooks in Section **General Application Changes**.*
 - *permissions and API keys, including the UI in Section **Access Control**.*
 - *the lifecycle, non-UI features, or APIs of views and dashboards in Section **Views & Dashboards**.*
 - *the workspaces and scenarios lifecycle, features, or APIs in Section **Workspaces & Scenarios**.*
-

General Application Changes

This section details all changes related to general features of the application element management, including events and hooks.

New Beta Features

For the first time, DOC introduces beta features. They need to be generated during the build phase and can be enabled from the Application Preferences.

They are described in the following Sections:

- **Dev Changes > Python.**
- **UI Changes > Code Editor.**
- **UI Changes > Rule Script Editor.**

For more details on how to enable beta features, please refer to Section **Dev Changes > Build** and Section **UI Changes > Application Preferences**.

Access Control

This section details all changes related to permissions and API Keys, including the UI.

Improved Security for the Web Client Login

The password policy now requires a password with a minimal length of 8 with letters, numbers, and special chars. The username is excluded from possible passwords.

In addition, user accounts are locked for 1 minute after 5 failed login attempts.

Improved Scenario Permissions

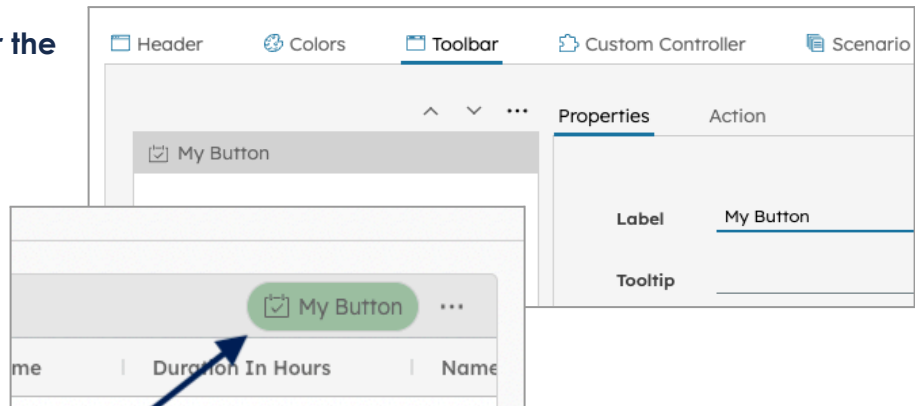
When a user creates a scenario link or a CDM scenario referencing another scenario, a permission rule giving access to the referenced scenario is now created. This allows users to revoke access to a specific scenario referenced through scenario links or other CDM scenarios.

Views & Dashboards

This section details all changes related to the lifecycle, the non-UI features, or the APIs of views and dashboards.

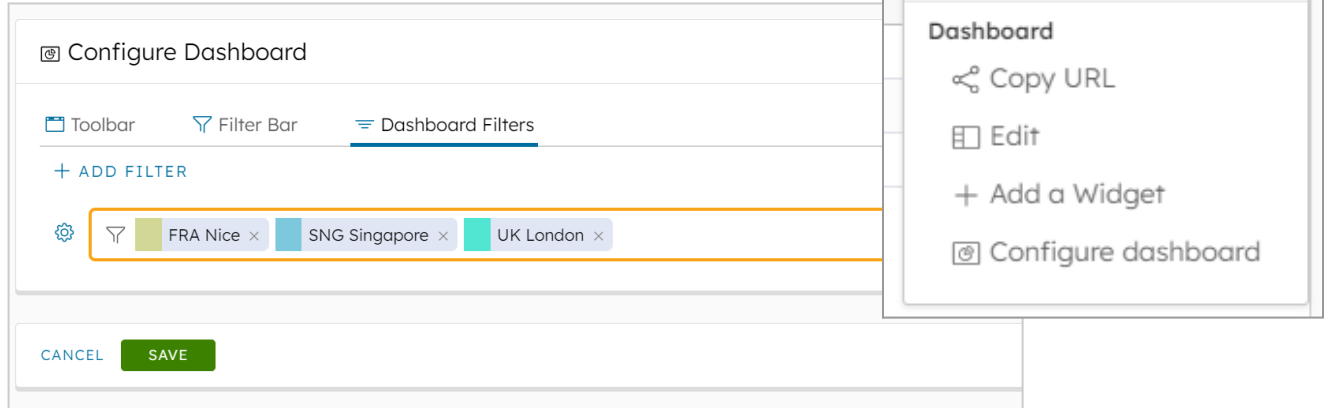
New Configurable Buttons for the Widget Toolbar

Every widget configurator now displays a new *Toolbar* tab. It allows adding action buttons to the widget toolbar.



New Dashboard Filters

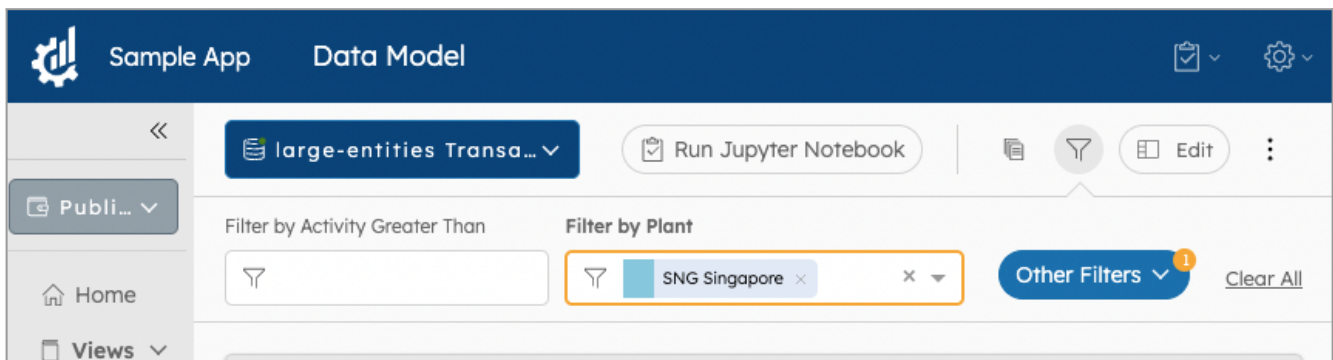
The option Configure Toolbar, available on custom dashboards, has evolved into Configure Dashboard. It introduces a new Dashboard Filters tab that allows setting permanent filters specific to the dashboard.



New Filter Bar

A new Filter Bar can now be displayed on all custom dashboards. It provides users with additional dynamic filtering on top of any permanent filters already set on the dashboard or on the widgets.

When using a Filter Bar, filters from other dashboards now show up in the Other Filters dropdown instead of in the Context Selection dropdown which now switches to a toggle button.

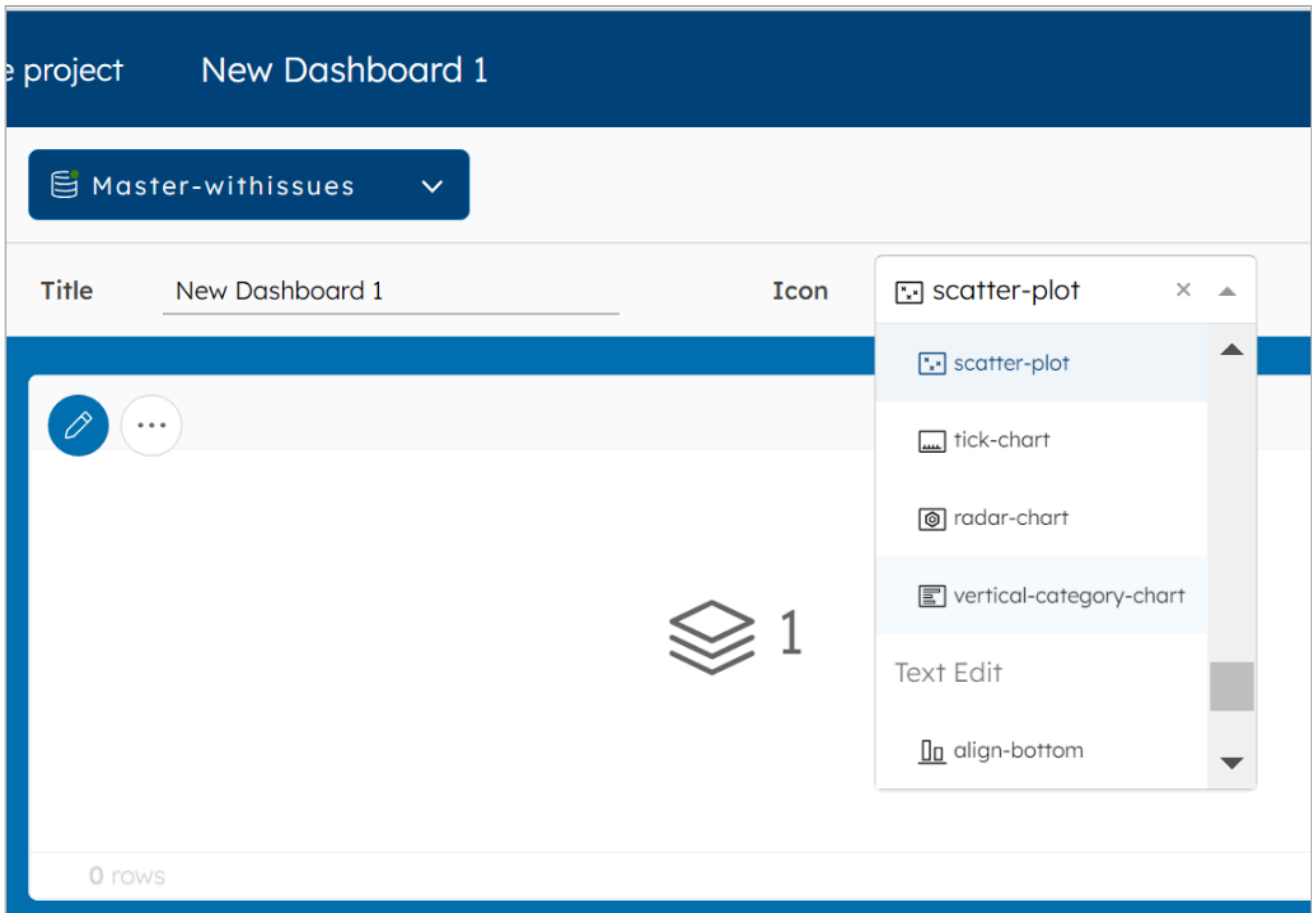


New Filtering Scope

In the Application Preferences, the new *FILTER_SCOPE* parameter is set by default to *GLOBAL*. This means that the Filter widget and new Filter Bar apply to all views and dashboards across the application. It can be set to *VIEW* to limit the filtering to the dashboard or view of the element.

Improved Sidenav Icons

When editing the layout of a custom view or dashboard, users can now configure an *Icon* in addition to the *Title* that is displayed in the Sidenav.

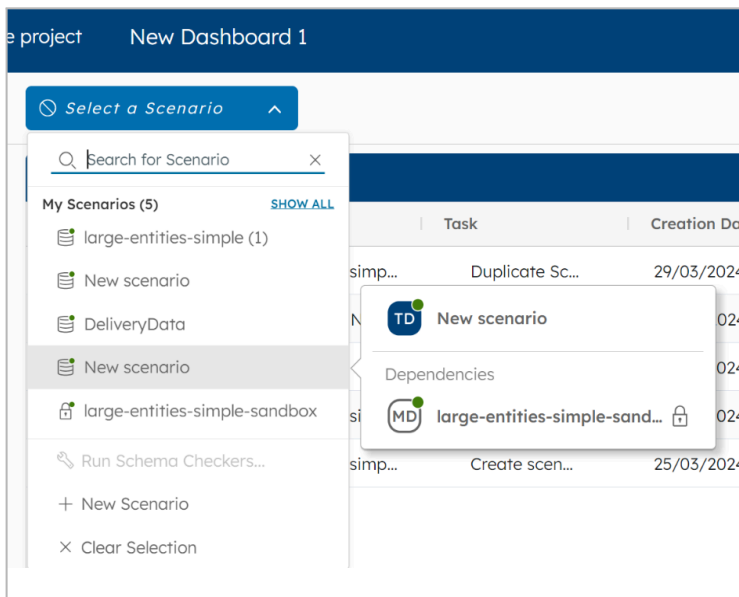


Workspaces & Scenarios

This section details all changes related to the lifecycle, the features, or the APIs of workspaces and scenarios.

New Scenario Types for the CDM

In a Composite Data Model, scenarios are associated with a type and can reference or be referenced by other scenarios.



Improved Display for the CDM

Users can now display the type and references of a scenario when hovering over the scenario, in the Scenario Selector.

In the Scenario List widget, users can also display a column for the scenario type. It can also be found, along with its references, using the option "Scenario details" in the Actions column, or when hovering over the scenario.

The tooltip displays the same information when hovering over a scenario in the Job List widget.

Name	Creation Date	Modification Date	Author	Data Status	Latest Job	Type	Actions
Public Workspace (...)							...
large-entities-...	25/03/2024 1...	25/03/2024 1...	gene_admin	Valid	Create scenar...	DebugData	...
large-entities-...	25/03/2024 1...	25/03/2024 1...	gene_admin	Valid	Create scenar...	MasterData	...
New scenario	25/03/2024 1...	25/03/2024 1...	gene_admin	Valid	Create an em...	Transactional...	...
DeliveryData	0...	0...	gene_admin	Valid	Create an em...	DeliveryData	...
New scen	0...	0...	gene_admin	Valid	Create an em...	PlanData	...
Trash (1)							...

Improved Scenario import

When using a composite data model, as described above, users must now specify the type of scenario during its import and, if need be, reference other scenarios.

If a scenario to reference is unavailable, the option "Create a new scenario from the same file" can be used to create one from the imported file.

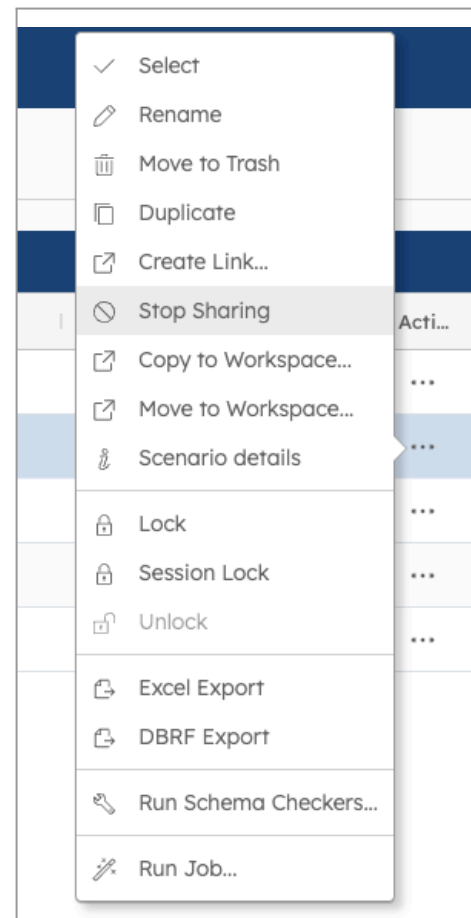
Users can also now select a scenario link as a reference scenario when adding a CDM scenario.

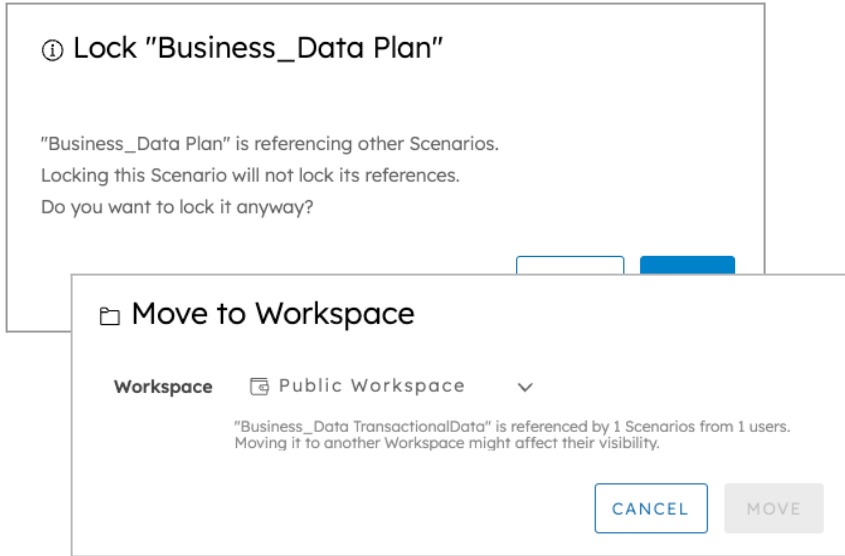
Improved Scenario Permissions

The permissions evaluation has changed. When a user creates a scenario link or a CDM scenario referencing another scenario, a permission rule giving access to the referenced scenario is now created. This allows users to revoke access to a specific scenario referenced through scenario links or other CDM scenarios.

Note that users can only display the shared scenario when they have access to the scenario's workspace or when they have access to the created scenario link. For an application in a previous version of DOC with scenarios referenced by one or more scenario links, once the application is migrated to DOC 4.4.0, these scenarios are automatically migrated and associated with a permission rule giving access to the scenario. Also, The new permission rule will override any existing one.

Suppose the scenario is shared with everyone through a scenario link or a CDM scenario. In that case, the web client now displays an indicator on the scenario icon and the option Stop Sharing in the Action menu.





Improved Lock Mechanism for the CDM

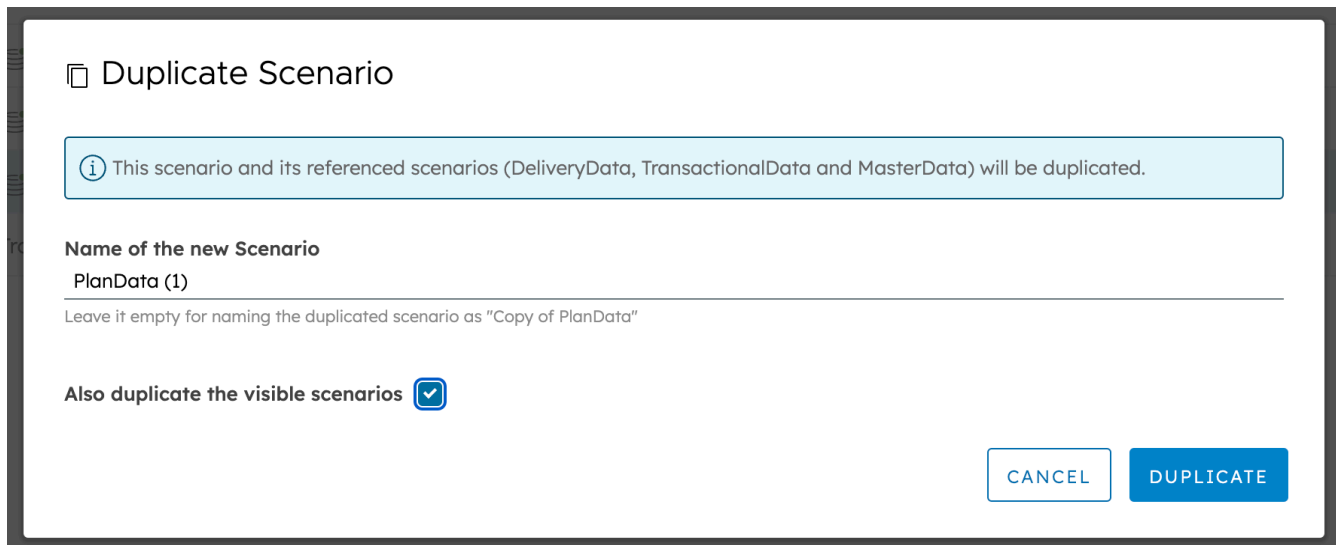
To avoid breaking data continuity, moving or locking a scenario with references from or to other scenarios now triggers a warning.

Note that, duplicating a scenario only duplicates the data it contains and its references. It does not copy the data in the referenced scenarios.

Improved Scenario Duplication and Deletion

Users can now duplicate a scenario and the referenced scenarios they have access to.

In a Composite Data Model application, it is possible to delete multiple scenarios, regardless of their hierarchy. All the scenarios referencing a deleted scenario are now flagged as "corrupted".



Data Changes

This part details all changes related to:

- the data model in Section **Model**.
- the language definition and parsing in Section **JDL**.
- the GraphQL interface elements or extensions in Section **GraphQL**.
- the import or export mechanism provided by the product in Section **Built-in Import/Export**.
- the Data Integration core API in Section **Data Integration Framework**.
- the Data Service API in Section **Data Service**.
- the Scenario Service API in Section **Scenario Service**.
- the Execution Service API in Section **Execution Service**.
- the Backend Service API in Section **Backend Service**.

Model

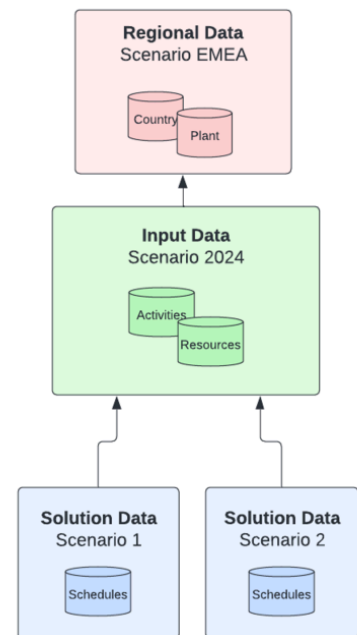
This section details all changes related to the data model.

New Composite Data Model Feature

DOC 4.4.0 introduces the Composite Data Model (CDM), which offers a new level of abstraction to the business data model definition.

This feature allows sharing data between scenarios, which avoids duplicates and improves the application performance by reducing resource usage.

The CDM helps solution designers define different scenario types, each one defining a part of the application data model.



In this configuration, each scenario of the application:

- has a scenario type,
- contains only the data defined in the tables corresponding to its scenario type, and
- reference scenarios a specific type as expressed in the data model.

For example, an application may have three scenario types defined:

- *Regional Data*, which contains global shared data;
- *Input Data*, which contains optimization inputs on the one hand and references *Regional Data* on the other; and
- *Solution Data*, which contains optimization results and references *Input Data*.

This way, any dashboard displaying data for a *Solution Data* scenario can also display data from the referenced *Input Data* and *Regional Data* scenarios.

JDL

This section details all changes related to the language definition and parsing.

New JDL Syntax for the Composite Data Model

The `JdlApplication` class and the associated ones have been extended to reflect the Composite Data Model features. In particular, when you build a `JdlApplication` instance, you must provide a list of `JdlScenarioTypes`; if your JDL is still in the style of the 4.1.0 release, that is, without scenario types, then you can pass a singleton list of `JdlScenarioType.default()`. Similarly, instances of `JdlEntity` must be given a scenario type name (the name of the default scenario type is ""). Finally, if you are using the `JdlApplicationLoader` class, its current `readJdl` methods have been replaced with one that takes a `JdlStreamResolver`.

GraphQL

This section details all changes related to the GraphQL interface elements or extensions.

Improved GraphQL Introspection

As it should not be activated in production, GraphQL introspection feature is now disabled by default for the Data Service and Scenario Service. To enable the introspection in development, use the following Spring property:

```
Unset
spring:
  graphql:
    schema:
      introspection:
        enabled: true
```

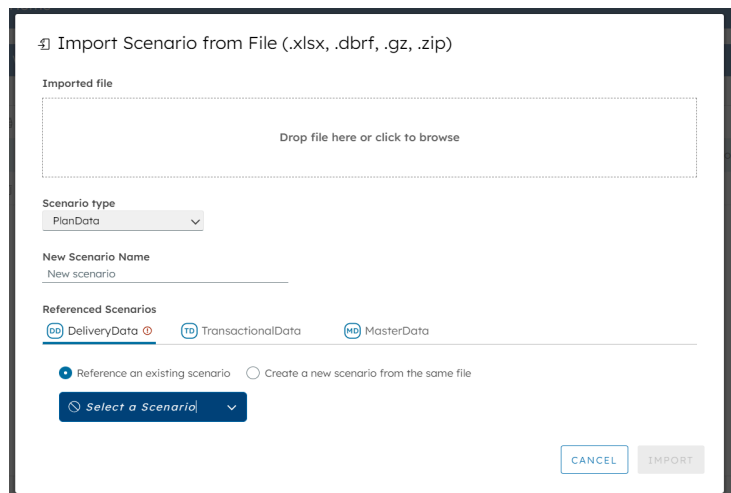
Built-in Import/Export

This section details all changes related to the import or export mechanism provided by the product.

Updated Import for the CDM

When using a composite data model, as described above, users must now specify the type of scenario during its import and, if need be, reference other scenarios.

If a scenario to reference is unavailable, the option "Create a new scenario from the same file" can be used to create one from the imported file.



Data Integration Framework

This section details all changes related to the Data Integration core API.

Deprecated DBPF and CSV collectors

CSV collectors and the DBPF file format are now deprecated in favor of the XCSV format in scripted tasks and DOMCollector APIs.

New XCSV File Format

The new XCSV format to export or import scenario files now contains both the `internalId` (as CSV collectors) and the business key (as in DBRF files). An XCSV file has the `.xcsv` extension. It is a ZIP file that contains `.csv` files. In short, it is a CSV collector to which the business key fields have been added. In addition, the XCSV format now replaces the CSV format in scripted tasks.

The new endpoint `/data/xcsv-export` or the `XCSVExportTask` task can now be used to export a scenario to XCSV.

Data Service

This section details all changes related to the Data Service API.

Removed Data Service Classes

The following classes have been deprecated:

- `ReflexionGetService`. Instead, use `DatabaseGetService`.
- `ReflexionInsertService`. Instead, use `DatabaseInsertService`.
- `ReflexionUpdateService`. Instead, use `DatabaseUpdateService`.

Deprecated Methods in the Data Service REST API

The following methods of the data-service-base library have been deprecated:

- In `BatchCollectorService`, method `saveItems(String scenarioId, List<T> entityDTOs)` has been deprecated in favor of `saveEntities(String scenarioId, List<T> entityDTOs)`.
- In `ScenarioUpdateService`, method `saveItems(String, List<DataServerEntityDTO> items, SchemaCheckersRunOptions)` has been deprecated in favor of `saveEntities(String, List<DataServerEntityDTO> entities, SchemaCheckersRunOptions)`.

Updated SchemaChecker API for the CDM

The interface `SchemaChecker` has been updated to support the Composite Data Model (CDM) feature. This implies the following things:

- Update the method signatures of your `SchemaChecker`, if you implemented any.
- Adapt your `SchemaChecker` to support the CDM.

The following methods have been updated:

```
Unset
public interface SchemaChecker<T extends SchemaCheck> {
    List<T> createSubjects(ScenarioReferenceGraphAware scenario);
    List<String> findInvalidInternalIds(T checker, ScenarioReferenceGraphAware
scenario);
}
```

The parameter `ScenarioReferenceGraphAware scenario` has been added.

This object contains all the CDM. it represents the scenario for which the checker is called and contains all the CDM information of the scenario.

In general, you will need to have JDL representation to adapt the implementation of your checker– a bean of type `JdlApplication` is available in the Spring context.

Now, for instance, to get the list of entities contained by the scenario, you can use `scenario.getProperEntities(jdlApplication)`.

Scenario Service

This section details all changes related to the Scenario Service API.

Removed Property in the Scenario Service GraphQL API

The property `scenarioReferenceGraph` used by the Scenario Service GraphQL API has been removed from the definition of object Path. The REST API is still exposing this property.

Deprecated Property in the Scenario Service GraphQL API

The Scenario Service GraphQL API was returning the `Path { scenarioReferenyceGraph }` property, which is incompatible with GraphQL structures. This property will be removed in the next version. It is recommended to use the corresponding REST APIs to access this property.

Deprecated Excel Import/Export Endpoints in the Scenario Service REST API

The following endpoints have been deprecated:

- `/data/simple-excel-export`, replaced by `/data/excel-export`.
- `/data/simple-excel-import`, replaced by `/data/excel-import`.

The deprecated parameter `sortColumns` has also been removed from the Excel Export API.

Updated Scenario Service REST API

`PathEventDTO` model has been updated, its field `modificationDate` has been renamed to `lastModificationDate`.

Execution Service

This section details all changes related to the Scenario Service API.

New Dependency

In the `execution-service` module, the `build.gradle` file now declares a new dependency to the `:gene-model:gene-model-desc` module.

```
Unset
dependencies {
    // Execution service application
    implementation
    "com.decisionbrain.gene:execution-service-application:${versions.decisionbrain.dbgene
}"
    // Extension
    implementation project(":extensions:execution-service-extension")

    // Description of the application data model
    implementation project(":gene-model:gene-model-desc")

    // =====
    // = There should not be any additional dependencies here. =
    // = Please customize the extension instead.               =
    // =====
}
```

Updated JSON Serialization Limit for the Execution Service

To handle large scenarios, the Execution Service limit for JSON serialized objects can now be set to 250MB in `extensions/execution-service-extension/src/main/resources/application.yml`:

```
Unset
# Configure the maximum size of a JSON payload. Used to send and receive
input/outputs to/from a routine.
gene:
  object-mapper:
    stream-max-string-length: 250000000 # 250MB
```

For more details, please refer to Section **3rd Party Components**.

Backend Service

This section details all changes related to the Scenario Service API.

Updated JSON Serialization Limit for the Backend Service

To handle large scenarios, the Backend Service limit for JSON serialized objects can now be set to **250MB** in `extensions/backend-service-extension/src/main/resources/application.yml`:

```
Unset
# Configure the maximum size of a JSON payload. Used to send and receive
input/outputs to/from a routine.
gene:
  object-mapper:
    stream-max-string-length: 250000000 # 250MB
```

For more details, please refer to Section **3rd Party Components**.

DBOS Changes

This part details all changes related to:

- the DBOS Java task definition and worker library in Section **Workers (Java)**.
 - the DBOS Python task definition, worker library, and data model mapping in Section **Workers (Python)**.
-

Workers (Java)

This section details all changes related to the DBOS Java task definition and worker library.

Updated Scenario Data Format for Java Workers

- `ScenarioDataExpression`, which formerly relied on the CSV format, now uses XCSV as the default file format for exchanges.
- `TheDBDomCollector#loadSnapshot(InputStream, DbDomCollectorSerializerFormat)` has been deprecated in favor of `DBDomCollector#loadSnapshot(InputStream)`, which handles the default file format automatically.

Consequently, using the legacy `loadSnapshot(...)` in worker code for tasks is only possible by explicitly setting a `ScenarioDataExpression` format through the `withFormat(...)` method.

Otherwise, the loaded collector will be empty of data.

Solution 1: Update the worker code

Update the worker code so it no longer uses the deprecated `loadSnapshot()` method but the new method to load the snapshot instead.

Unset

//Before 4.4.0

```
private CapacityPlanning extractInputCollector(ExecutionContext executionContext) {
    DbDomCollectorFactory<CapacityPlanning> fact = new CapacityPlanningFactoryImpl();
    return executionContext.getInputData("inputCollector").load(inputStream -> {
        CapacityPlanning coll = fact.createCollector();
        coll.loadSnapshot(inputStream, DbDomCollectorSerializerFormat.CSV);
        return coll;
    });
}
```

For example, the code above can be edited as follows:

Unset

//In 4.4.0 and above

```
private CapacityPlanning extractInputCollector(ExecutionContext executionContext) {
    DbDomCollectorFactory<CapacityPlanning> fact = new CapacityPlanningFactoryImpl();

    return executionContext.getInputData("inputCollector").load(inputStream -> {
        CapacityPlanning coll = fact.createCollector();
        coll.loadSnapshot(inputStream);
        return coll;
    });
}
```

Solution 2: Update the task definition

Update the task definition to make sure the `ScenarioDataExpression` explicitly uses CSV.

```
Unset
//Before 4.4.0

task.getScript()
    .addStatement(AskInputStatement.ofVariable(scenario, true,
JobInputType.scenarioId(WRITABLE)))
    .addStatement(ExecuteOptimizationServerTaskStatement
        .forTaskId(StringExpression.of("EngineTask"))
        .withInput(INPUT_COLLECTOR, ScenarioDataExpression.of(scenario))
        .withOutputScenario(OUTPUT_COLLECTOR, scenario)
    );
```

For example, the code above can be edited as follows:

```
Unset
//In 4.4.0 and above

task.getScript()
    .addStatement(AskInputStatement.ofVariable(scenario, true,
JobInputType.scenarioId(WRITABLE)))
    .addStatement(ExecuteOptimizationServerTaskStatement
        .forTaskId(StringExpression.of("EngineTask"))
        .withInput(INPUT_COLLECTOR,
ScenarioDataExpression.of(scenario).withFormat(ScenarioDataFormat.CSV))
        .withOutputScenario(OUTPUT_COLLECTOR, scenario)
    );
```

Workers (Python)

This section details all changes related to the DBOS Python task definition, worker library, and data model mapping.

Deprecated CSV Collector for Python Workers

The CSV collector, which is a `.zip` archive of `.csv` files, is now deprecated and has been replaced by the XCSV format.

New Business Key in XCSV Format for Python Workers

With the introduction of the XCSV format, the data frames loaded in the worker now contain the business key fields of each entity in addition to their `internalId`. In a composite data model application, one must pay particular attention when saving data frames with consistent business keys across multiple scenario types.

In addition, when saving a Python collector to XCSV, the business keys are saved, too.

New Debug Configuration for Python Workers

Developers can use IntelliJ to debug a running Python worker. To do so in IntelliJ, one must:

1. Add the *Run configuration* of type *Gradle*.
2. Add the *Run configuration* of type *Python Debug Server*.
3. Start the *Run configuration* of type *Python Engine Worker* in *debug* mode.
4. Start the *Run configuration* of type *Python Debug Server*.

Developers can then add debug breakpoints to the Python code.

Dev Changes

This part details all changes related to:

- DOC REST API in Section **REST API**,
except for the **Data Integration Framework**, **Data Service**, and **Scenario Service** API.
 - DOC external dependencies and libraries, such as Spring, Angular, or Keycloak
in Section **3rd-Party Components**.
 - DOC scaffolding and Gradle scripts in Section **Build**.
 - DOC security issues in Section **Security**.
 - DOC integration to Docker or Helm in Section **Deployment**.
 - DOC integration with Python, except for **Workers (Python)** and **Routines** in Section **Python**.
 - DOC integration with Jupyter in Section **Jupyter**.
 - DOC integration with CPLEX in Section **CPLEX**.
 - DOC and DBOS documentation in Section **Documentation**.
-

REST API

This section details all changes related to DOC REST API, except for the Data Integration Core, Data Service, and Scenario Service API.

Removed REST API Elements

- In class `GeneContextService`, methods `setScenarioIds()` and `addScenarioId()` have been deprecated and removed. Instead, use `setScenarioSelection()` and `addToScenarioSelection()`, respectively.
- The type and constant `GeneScenarioEventType` have been deprecated and removed. Instead, use `ScenarioNotificationType`.
- In class `GeneSettingsService`, methods `registerDefaultSettings()` and `resetSettings()` have been deprecated and removed. Instead, use `registerDefaultApplicationSettings()` and `resetApplicationSettings()`, respectively.
- In interface `GeneWidgetHeaderConfiguration`, member `showMenu` has been deprecated and removed. Instead, use `GeneMenuItemsProvider`.
- In interface `GeneModalDialogButton`, the member `shortcut` and its associated type `GeneDialogButtonShortcut` have been deprecated and removed.
- In class `ExecuteOptimizationServerTaskStatement`, the variant of method `withOutputScenario()` that takes a format as argument has been deprecated and removed. Instead, use the other variant of this method as only the CSV format is supported.
- In type `JobInputType`, constant `NUMERIC` and method `numeric()` have been deprecated and removed. Instead, use `REAL` and `real()`, respectively.

Deprecated “GET” for “scenariosIdsAndNames”

For the `/api/scenario/scenariosIdsAndNames` endpoint, the `GET` method is now deprecated. Use the `POST` method instead.

Updated REST API Elements

The type `ScenarioDTO` has been renamed into `ScenarioCreationRequestDTO`.

3rd-Party Components

This section details all changes related to DOC external dependencies and libraries, such as Spring, Angular, or Keycloak.

Improved Jackson JSON Serialization Capabilities

The `jackson-core` library in version ≥ 2.15 introduced a configurable size limit for the serialized JSON object with a default value of `20MB`. This change in the `jackson-core` library used by DOC produced issues in some applications while transferring large scenarios between the Execution Service and Backend Service.

An application property now allows configuring the `jackson-core` size limit. This setting can be overridden and increased for both services when dealing with large scenarios.

Therefore, the default value for both the Execution and Backend Services can now be set to `250MB`.

`extensions/execution-service-extension/src/main/resources/application.yml` and `extensions/backend-service-extension/src/main/resources/application.yml` need to be configured with the same value.

```
Unset
# Configure the maximum size of a JSON payload. Used to send and receive
input/outputs to/from a routine.
gene:
  object-mapper:
    stream-max-string-length: 250000000 # 250MB
```

For more details, please refer to the [Jackson Core Github Page](#).

Updated NPM Dependencies

```
Unset
// Updated NPM Dependencies

@angular/animations: 18.2.10
@angular/cdk: 18.2.2
@angular/cli: 18.2.11
@angular/common: 18.2.10
@angular/compiler: 18.2.10
@angular/compiler-cli: 18.2.10
@angular/core: 18.2.10
@angular/forms: 18.2.10
@angular/google-maps: 17.2.1
@angular/language-service: 18.2.10
@angular/localize: 18.2.10
@angular/platform-browser: 18.2.10
@angular/platform-browser-dynamic: 18.2.10
@angular/router: 18.2.10
@angular-devkit/build-angular: 18.2.11
@fullcalendar/angular: 6.1.15
@fullcalendar/core: 6.1.15
@fullcalendar/daygrid: 6.1.10
@fullcalendar/interaction: 6.1.10
@fullcalendar/list: 6.1.10
@fullcalendar/timegrid: 6.1.10

ag-grid-angular -> 32.3.1
ag-grid-community -> 32.3.1
ag-grid-enterprise -> 32.3.1
echarts: 5.4.3
ngx-echarts: 17.1.0
// New NPM Dependencies

@cds/angular: 6.10.0
@cds/core: 6.10.0
@clr/angular: 17.0.1
@clr/ui: 16.3.7
```

```
@danielmoncada/angular-datetime-picker: 18.1.0
@ng-select/ng-select: 13.7.0

ag-grid: 31.3.2
angular-gridster2: 18.0.1
daypilot-pro-angular: 2024.3.6155
keycloak-angular: 16.0.1
keycloak-js: 24.0.2
monaco-editor: 0.49.0
ngx-monaco-editor-v2: 18.0.1
ngx-quill: 26.0.8
quill: 2.0.2
rimraf: 6.0.1
typescript: 5.5.4
zone.js: 0.14.10




// Removed NPM Dependencies

@asymmetrik/ngx-leaflet
@asymmetrik/ngx-leaflet-markercluster
```

Typescript has been updated from 5.2 to 5.5, implying notable changes, such as:

- [Regular Expression Syntax Checking](#)
- [Support for New ECMAScript Set Methods](#) .

The list of changes can be found at:

-  [Announcing TypeScript 5.3 - TypeScript](#)
-  [Announcing TypeScript 5.4 - TypeScript](#)
-  [Announcing TypeScript 5.5 - TypeScript](#)

Updated Java dependencies

Unset

```
// New Java Dependencies
```

```
drools-engine: 8.44.0.Final
```

```
drools-mvel: 8.44.0.Final
```

```
JGraphT: 1.5.2
```

```
mongock: 5.4.4
```

```
// Updated Java Dependencies
```

```
Apache commons compress: 1.26.1
```

```
Apache commons text: 1.12.0
```

```
Apache POI: 5.2.5
```

```
com.decisionbrain.db-keycloak: 23.0.4
```

```
Gradle: 7.6.4
```

```
Jackson: 2.17.2
```

```
Keycloak: 26.0.5
```

```
Log4J: 2.23.1
```

```
org.openapitools:openapi-generator-gradle-plugin: 7.9.0
```

```
RabbitMQ: 4.0.2
```

```
Spring Boot: 3.3.5
```

```
Spring Cloud: 2023.0.3
```

```
Spring framework: 6.1.14
```

For more details on the Spring Boot update to 3.3.5, please refer to the [Spring 3.3.5 Github Page](#).

Updated Infra Dependencies

```
Unset
// Updated Other Dependencies
Node: 20.17

// New Other Dependencies
JupyterLab: 4.2.4
Python: 3.11.9
ipywidgets: 8.1.3
```

Updated Jackson Dependencies

The libraries `object-mapper-base` that provide a common configuration for Jackson mappings do not include the Kotlin configuration by default. However, it still can be added at the project customization level. Jackson mappings are used for REST payload serialization and deserialization. A constraint has to be added to ensure a consistent version of Jackson libraries across the application.

Edit the file `gradle/templates/spring-bom.gradle` and add the following content in the `configurations.configureEach` block.

```
Unset
configurations {
    configureEach {
        // ... Existing configuration ...
        resolutionStrategy.eachDependency { DependencyResolveDetails details ->
            if (details.requested.group.startsWith('com.fasterxml.jackson')) {
                details.useVersion '2.14.3'
                details.because '25/03/2024: Force Jackson to the version provided by
Spring Boot BOM'
            }
        }
    }
}
}
```

Build

This section details all changes related to DOC Scaffolding and Gradle scripts.

Deprecated Old DSL of Code Replicate Plugin

The content of the block `codeUpdates { }` should be migrated to `codeReplicas { }`.

New Angular Post Installation Script

The `.angular` folder is now automatically deleted after installing Angular dependencies.

This ensures that Angular does not serve stale cached packages after the migration, which could cause issues that are difficult to debug.

New Option for Beta Features in the Application Generator

The `generator.sh` script now accepts a new option to enable beta features in the scaffolded project.

To do so, the `generator.sh` script can be called:

- with `--experimentalFeatures`,
- with `--experimentalFeatures=yes`. Or
- without any form of the `--experimentalFeatures` flag, but with any form of the `--dbInternal` flag.

If one calls the script with no flag or with `--experimentalFeatures=no`, the scaffolded project will not contain beta features.

Improved Test Fixtures Modules

Spring BOM has been added to the `test-fixtures` modules.

Updated Default Size Limit for JSON Objects

Jackson property `streamMaxStringLength` is now configurable and set to `250MB` by default.

Updated Docker Compose Files

The following changes have been brought to the Docker Compose files:

- The Docker network has been renamed to `shared-network` in the `docker-compose.yml` files, the created network will be named using `COMPOSE_PROJECT_NAME` as before.
- Some environment variables have been added to the `.env` file.
- Some environment variables have been changed (`KC_HOSTNAME_*`) or added (`MASTER_JWT_KEY`, `MALLOC_ARENA_MAX`, `SERVICES_DOCUMENTATIONCHATBOT_*`).
- The `JWT` secret key has been placed in the variable `MASTER_JWT_KEY` of the `.env` file.
- [beta feature] The folder where Jupyterlab notebooks are stored has been placed in the variable `JUPYTERLAB_NOTEBOOKS_DIR` of the `.env` file.

Updated Gradle Files

The following changes have been brought to the Gradle files:

- The Helm repository URL has been moved into the property `HELM_PUSH_REPOSITORY` of the `gradle.properties` file.
- The raw repository URL has been moved into the property `RAW_PUSH_REPOSITORY` of the `gradle.properties` file.
- In the `python-engine-worker` module, the file `build.gradle` has been updated. Some of its content has been moved to the new file `python-worker.gradle`.
- In addition, the files `python-library.gradle` and `python-root.gradle` have changed.
- The file `local-cplex.gradle`, which eases the CPLEX integration from a local CPLEX installation into a Java library, has been updated. In addition, the file `local-cplex-worker.gradle` was added to integrate CPLEX into the module `engine-worker`. For more details, please refer to the [DOC Documentation](#).
- Some parts of the `codeReplicas` block have been removed from the main `build.gradle` file.

Updated Code Replicate Plugin

The plugin `com.decisionbrain.code-replicate-plugin` has been updated to version 0.9.8.

Consequently:

- It has been renamed `com.decisionbrain.gradle.code-replicate-plugin`.
- It uses a new DSL (Domain Specific Language) but supports backward compatibility.

The old DSL is still available with the `block codeUpdates {}` and the new with the `codeReplicas {}`.

The scaffolded `build.gradle` files have been migrated but it is necessary to also migrate the block `codeReplicas {}` if it has been customized,

Here is a comparison of the old and new DSL:

Unset

//Old DSL

```
codeUpdates {
    globalVersion(globalValue)

    update("file.txt") {
        regex("foo(.*?)bar")
    }

    update("setup.py") {

pythonVersion('version="%s"')
    }
}
```

Unset

//New DSL

```
codeReplicas {
    withValue(globalValue)

    inFile("file.txt") {
        atLocation(regex("foo(.*?)bar"))
    }

    inFile("setup.py") {
        withValuePythonVersion()

atLocation(regexPythonVersion('version="%s" '))
    }
}
```

For more details, please refer to the [Code Replicate Plugin Gitlab Page](#).

Updated Packaging Dependencies

In the file `package.json`, the web client dependencies are now sorted alphabetically. This facilitates migration from previous versions as it makes comparison between `package.json` files easier to read.

Updated Helm Chart

The scaffolded Helm chart now allows users to pass arguments to the `mongodb` command line.

In addition, the Helm chart parameter `InitialRAMPercentage` is now removed from the scaffolded file `/deployment/helm/src/values.yaml`.

Security

This section details all changes related to any DOC security issues.

Improved Security for Trivy CVEs

Trivy CVEs are now fixed using RabbitMQ 4.0.2.

Deployment

This section details all changes related to DOC integration to Docker and Helm.

New Deployment Files for Jupyter Notebook

Deployment files for Docker and Helm have been added for Jupyter.

New Java Memory Usage Limit

Java native memory usage is now limited using the `MALLOC_ARENA_MAX` parameter, with negligible to no loss of performance.

Gene Online

This section lists all changes related to Gene Online.

New Gene Online Beta Feature

Gene Online Beta 4.4.0 is now available. It allows you to customize an easy-to-deploy application.

For now, it can only be used through DecisionBrain's Deploy Manager (easily deployable from "Start an Official Demo" → "GENE-ONLINE:4.4.0").

Python

*This section details all changes related to the Python integration to DOC, except for **Workers (Python)** and **Routines**.*

New Python Module for Jupyter

A Python module called `processing/jupyter-notebook` has been added.

It invokes a JupyterLab interface (formerly Jupyter Notebook) which allows processing scenario data and updating scenarios accordingly. Developers can locally run the JupyterLab interface from the Docker Compose stack.



Updated Python Requirement

For Python development, DOC now requires Python 3.12.x.

Jupyter

This section details all changes related to JupyterLab.

New JupyterLab Integration

The new integration of JupyterLab into DOC as a beta feature eases online development with Python, Java, and CPLEX.

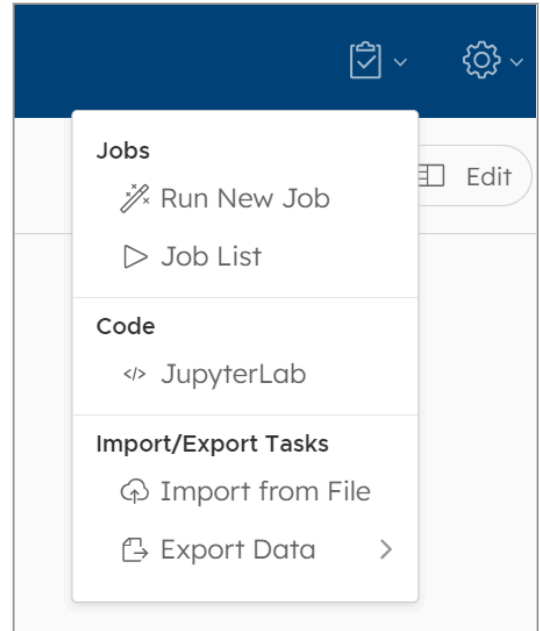
Users are provided with an option in the Topbar Tasks menu and the following sample notebooks in a base Docker image:

- CPLEX for Python
- CPLEX for Java
- CPO for Python
- CPO for Java

In addition, the Helm chart has been improved to allow deploying the JupyterLab service.

New Jupyter Notebook Sample for Scenario Import into a Pandas Data Frame

The application now provides users with a Jupyter Notebook sample to import scenario data into a Pandas data frame.



CPLEX

This section details all changes related to CPLEX.

Updated CPLEX Integration

The CPLEX integration has been improved. For more details, please refer to the [DOC Documentation](#).

Documentation Chatbot

This section lists all changes related to the Documentation Chatbot.

New Documentation Chatbot Beta Feature

The Documentation Chatbot now allows asking questions on the application features based on the Jira tickets, source code, and selected specification documents of the project.

Usage instructions can be found in the file `EXPERIMENTAL_FEATURES.md` generated for applications with the experimental feature enabled.

To activate this chatbot, follow the instructions in `documentation-chatbot/build.gradle`. Then, the build chain will scan the data sources (Jira, Gitlab, and/or Google Drive) to create vector stores and inject them into the Docker image for the chatbot.

From the application web client, users can open the chatbot via a command in the Topbar Tasks menu. To display this menu, the application preferences `EXPERIMENTAL_FEATURES` and `TASK_MENU_SHOW_DOCUMENTATION_CHATBOT` must both be set to `true`.

Also, an OpenAI API key is required. It can be set as a Spring property in the `application.yml` file of the Backend Service extension, or provided at deployment time in the `app/.env` file for Docker Compose deployment, or in the values file of the deployment configuration when deploying with Kubernetes. If no key is provided, it can be indicated using the Chatbot from the web client.

Documentation

This section details all changes related to DOC and DBOS documentation.

Updated Documentation

Documentation has been refactored.

Also, details on all widget configurators and Prometheus endpoints are now available.

In addition, the documentation now explains how to prevent scenarios from being moved to the Lost and Found workspace.

Scripted Tasks Changes

This part details all changes related to:

- the task definition and script language in Section **Definition**.
 - routines, in Java or Python, in Section **Routines**.
-

Definition

This section details all changes related to the task definition and script language.

New Task for the ChatGPT Feature

A new task allows communicating with ChatGPT and can be used by the new Code Editor widget.

Updated Tasks for Scenario Characteristics

Scenario creation tasks now also allow setting scenario characteristics.

Updated File Formats in Scenario Processing Method

In the method `ScenarioDataExpression`, the CSV and DBRF file formats have been marked as deprecated. XCSV is now the new default file format.

Updated Tasks for Scenario Creation

Scenario creation is now available in all scripted tasks and can be automated. Also, the sample task *Create an empty scenario* has been improved for CDM applications.

Updated Parameter in Scenario Creation Method

In the method `ScenarioCreationExpression.of`, the first parameter has been changed to an expression that evaluates to `Types.SCENARIO_CREATION_PARAMETERS`.

There are two ways to obtain such value within a scripted task, either by building it with `ScenarioCreationParametersExpression`, or from a job input as follows:

Unset

```
VariableAccessExpression scenarioCreationParameters =  
VariableAccessExpression.of(SCENARIO_CREATION_PARAMETERS)  
  
task.getScript()  
  
.addStatement(AskInputStatement.of(scenarioCreationParameters.getVariableName()  
, true, JobInputType.SCENARIO_CREATION_PARAMETERS, "Parameters to create  
the Scenario"));  
  
// Then use the expression scenarioCreationParameters whenever you need
```

Routines

This section details all changes related to routines in Python or Java.

Deprecated File Formats in Routines

In routines, the CSV and DBRF file formats have been marked as deprecated

New File Format in Routines

In routines, XCSV is now the new default file format.

Updated DBDomCollector API in CDMs

When using workers and routines in a composite data model, whether in Java or Python, any `DBDomCollector` of a scenario including referenced scenario data must be loaded using the default CSV serialization format. Note that referenced scenario data cannot be modified.

Updated GeneIssue in CDMs

When creating a `GeneIssue` in a composite data model for an `InstanceClass`, the `GeneIssue` must be saved on the scenario owning the `InstanceClass` (ie: the `InstanceClass` must be a proper entity of the scenario).

If the `GeneIssue` is not associated with the right scenario, the UI might not display it correctly.

UI Changes

This part details all changes related to:

- *all widgets, regardless of their type, or non-widget elements of the UI, such as the sidebar, menus, scenario picker, or views and dashboards, except for **Access Control** elements, such as permissions and API keys in Section **General UI Changes**.*
- *the Application Preferences, not its **Configuration** in Section **Application Preferences**.*
- *the custom renderers and controllers in Section **Extensibility**.*
- *all table-based widgets, regardless of their type, such as the Data Explorer, Data Grid, Job List, Scenario List, Workspace List, Issue List, and Issue Details widgets in Section **Tables**,*
- *specific UI widgets in Section **Data Grid/Explorer Widgets, Workspace and Scenario Widgets, Button Widget, Gantt Chart Widget, Filter Widget, Map Widget, Pivot Table Widget, Code Editor Widget, or Rule Script Editor Widget**— if a change impacts more than one widget, it is listed in the most relevant section and referred to in the others.*

General UI Changes

*This section details all changes related to all widgets, regardless of their type, or non-widget elements of the UI, such as the sidebar, menus, scenario selector, or views and dashboards, except for the **Access Control** elements, such as permissions and API keys.*

Deprecated Navigation Button Widget

The Navigation Button widget is deprecated and can no longer be added to a dashboard or view, as its role can be fulfilled using the new Button widget.



Application Preferences

*This section details all changes related to the Application Preferences, not the **Application Configuration**.*

New Parameter for Beta Features in Application Preferences

A new `EXPERIMENTAL_FEATURE` parameter is set to `true` by default in the Application Preferences.

New Option for Application Controllers in Application Preferences

The application controller mechanism has been improved. Application controllers are no longer set by code but by configuration through the Application Preferences.

Updated Scenario Import Parameter in Application Preferences

The parameter `DEFAULT_EXCEL_IMPORT_TASK_ID` has been removed in favor of `DEFAULT_SCENARIO_IMPORT_TASK_ID`.

Requesting to import an Excel file from the web client now launches a job from the task defined using the `DEFAULT_SCENARIO_IMPORT_TASK_ID` parameter.

Extensibility

This section details all changes related to the custom renderers and controllers.

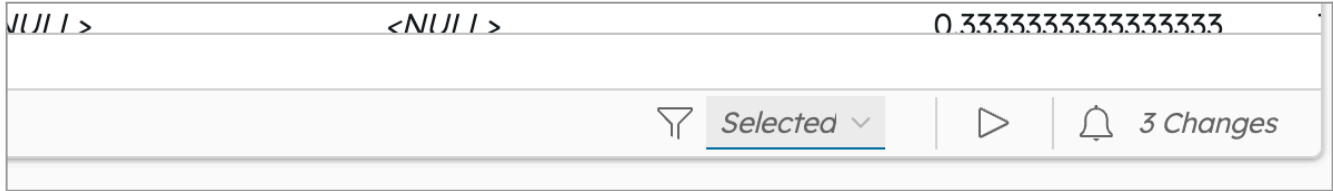
Deprecated Method in the Application Controller API

The method `setCustomApplicationController(..)` from the `GeneApplicationService` has been deprecated and replaced by `registerApplicationController(..)`.

New Status Bar API

The Status Bar API, which allows adding a status bar at the bottom of some widgets, is now available.

The `SampleStatusBarController` sample is available by default.



Updated Application Controller API

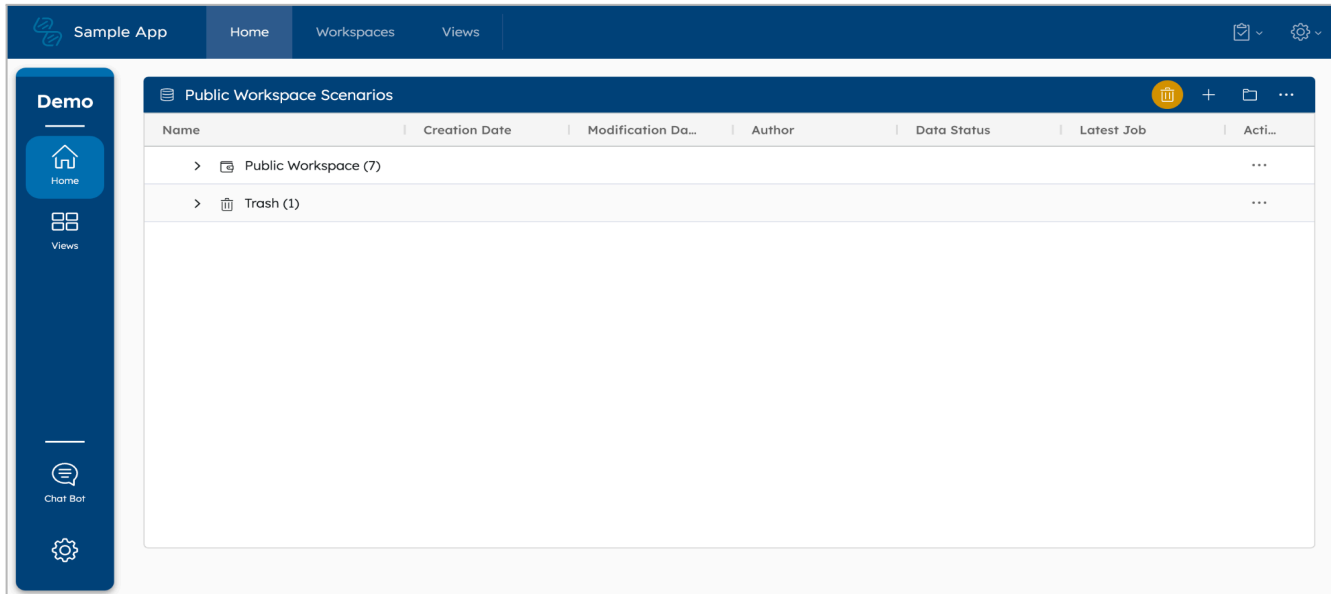
The application controller mechanism has been improved. Application controllers are no longer set by code but by configuration through the Application Preferences.

Also, developers can now register multiple application controllers through the API.

```
Unset
// Register a custom GeneApplicationController.
// @param controllerName the name of the application controller
// as it will be displayed in the UI
// @param controllerFactoryFn the factory function
// that will create the controller instance
registerApplicationController(controllerName: string,
                             controllerFactoryFn: (injector:Injector)
=> GeneApplicationController);
```

The application controller API has also been improved to let developers provide the Sidebar and the Topbar with custom components to replace the default ones.

In the folder `web/modules/sample-controllers/application`, some code samples illustrate how to



use this API.

Unset

```

export interface GeneApplicationConfiguration {
  // ...
  header?: {
    // When provided, the component will replace
    // the default Gene component for the header bar
    component?: Type<any>;
    // ...
  },
  sidenav?: {
    // When provided, the component will replace
    // the default Gene component for the sidebar
    component?: Type<any>;
  }
}

```


Tables

This section details the changes related to all table-based widgets, regardless of their type, such as the Data Explorer, Data Grid, Job List, Scenario List, Workspace List, Issue List, and Issue Details widgets.

Removed Deprecated Elements

Class `GeneScenarioListParams` has been removed.

Instead, use `GeneScenarioListConfiguration` if needed.

Property `comparisonState` has been removed from the class `GeneTableWidgetState`.

Updated AG Grid Import

After the AG Grid upgrade to 31.3.2, some imports may have to be updated, for example, in the custom controller code.

This applies to most imports where a path was needed, such as:

```
Unset
import { ValueGetterParams } from
'ag-grid-community/dist/lib/entities/colDef';
```

As with most other `ag-grid` imports, this can now be simplified as follows:

```
Unset
import { ValueGetterParams } from 'ag-grid-community';
```

Updated Libraries

The `gene-ag-grid-utils.ts` that exports methods `setGeneContext`, `getGeneContext`, `getGeneTranslateService`, and `setGeneTranslateService` has been moved to the library `@gene/widget-core`. If you use these methods in your Web UI code, you should update the imports.

Updated Gene Table Component

Some breaking changes related to the AG Grid v31 upgrade should be noted:

- `gridOptions` are now set to read-only during table initialization. Setting a property on the `gridOptions` object may not be detected by the grid. Instead, in any custom code that alters the grid configuration, use `api.setGridOption(property, value)`.
 - The function `getRowId` can only be set once. Updating after the `ag-grid` component has been initialized is no longer needed.
 - For the reasons described above, function `GeneTableController.customizeGridOption` is now called when the widget configuration and/or custom controller is assigned to the widget. This should not require changes to the custom controller code. It was previously called upon the initialization of the `ag-grid` component.
 - `columnApi` has been deprecated in AG Grid. Functions of `columnApi` have been moved to `gridApi` and the calls in the custom code should be changed accordingly
 - If changes to the `gridOptions` are made only through `GeneTableController.customizeGridOption`, there should be no change required on the custom code unless it requires the grid to be initialized. If the custom code somehow mutates the `gridOptions` after it is set to the `ag-grid` component, the corresponding code should be replaced by a call to `api.setGridOption`
-

Data Grid/Explorer Widgets

This section details all changes related to the Data Grid and Data Explorer widget.

New Index Bar Localization

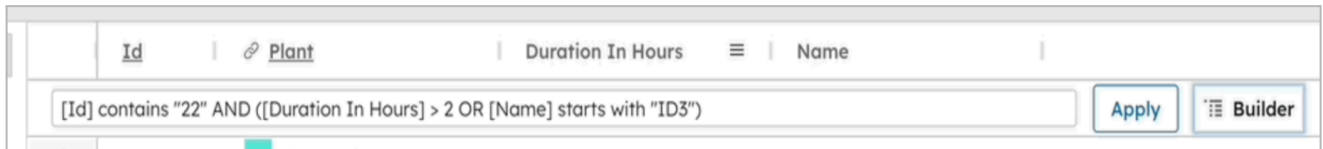
It is now possible to localize the message "row X of Y" at the bottom of the Data Grid widget, which gives the index of the row selected.



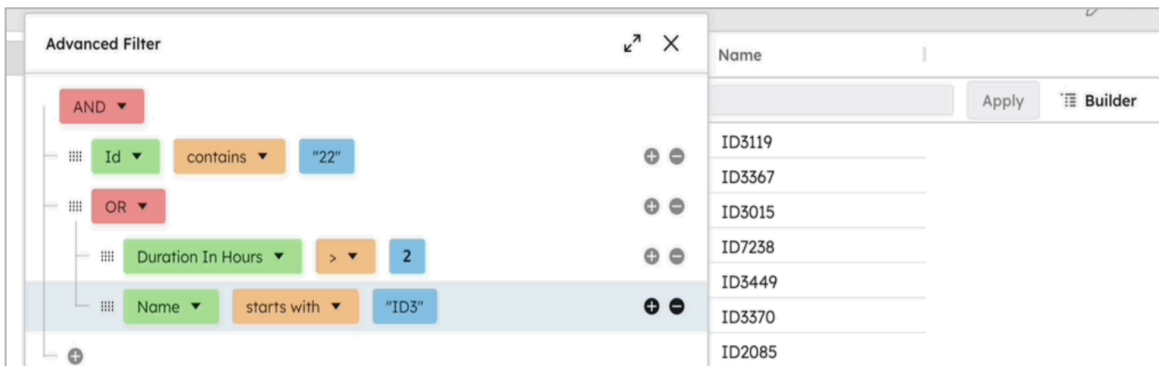
New Advanced Filters (EXPERIMENTAL)

In the tab *Advanced* of the Data Explorer and Data Grid widgets configurator, an option now allows to *Enable advanced Filter*.

When this option is enabled, the Data Grid has a filter field where users can type complex expressions.



The user can edit the filter expression by typing the expression in the filter field or using the graphical filter builder.



Several limitations of the advanced filters are to be considered in 4.4.0:

- For *Duration* and *LocalTime* entities, advanced filters are not available.
- For *LocalDateTime* and *Instant* entities, advanced filters are not available and, in a simple filter, the value will be the selected date with a time of 00:00.

Improved Date Filter in the Data Grid/Explorer Widgets

In the Data Grid/Explorer widgets, **DateTime** columns can now be filtered only by **Date**.

Improved Display for Booleans in the Data Grid/Explorer Widgets

In the Data Grid/Explorer widgets, undefined Boolean values are now shown as `<NULL>`.

Workspace and Scenario Widgets

This section details all changes related to the Scenario List and Scenario Timeline widget.

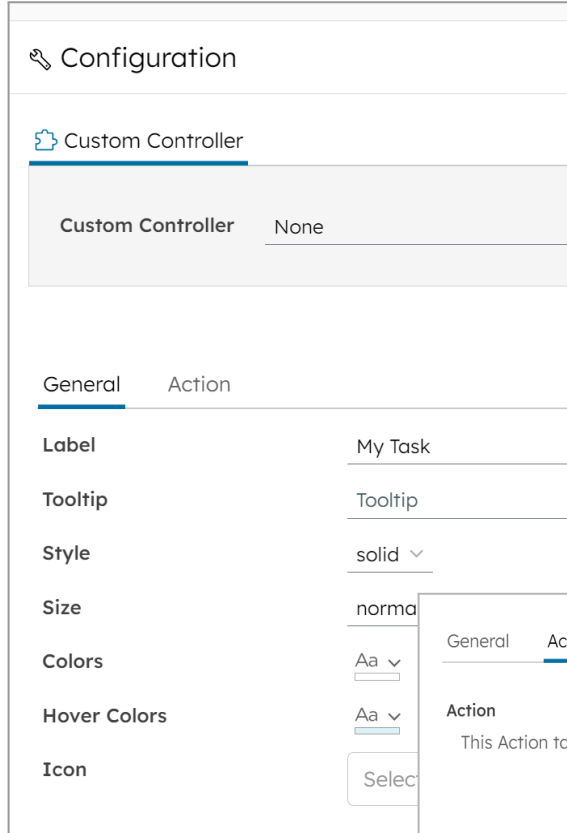
Updated Scenario List Widget

Custom Actions now use the new Action API.

- `GeneScenarioListConfiguration#customTasks` is now deprecated in favor of `GeneScenarioListConfiguration#GeneScenarioListConfiguration`.
 - `GeneScenarioListCustomActionTask` is now deprecated in favor of `GeneScenarioListCustomAction`.
-

Button Widget

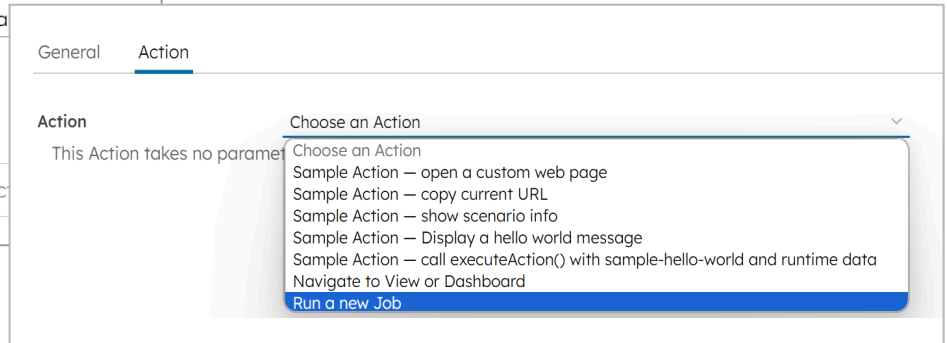
This section details all changes related to the Navigation Button widget.



New Button Widget

The Navigation Button and New Job Button widgets are now deprecated and make way for a new widget called Button. This new Button widget is easily customizable and relies on the Action API introduced in 4.1.0.

Note that, even if they are not available as new widgets to create, existing Navigation Button and New Job Button widgets still work as expected.



New Button Handler

`GeneNewJobButtonController` has a new method `actionHandler` to handle clicks on the button.

```
Unset
/**
 * Implementing this method allows to override the click behavior of the button.
 * @param event The native pointer event
 * @param context The GeneContext when the click occurred
 * @param action The action configuration of the button
 */
actionHandler?: (event: PointerEvent, context: GeneContext, action: GeneAction<any>)
=> void;
```

Gantt Chart Widget

This section details all changes related to the Gantt Chart widget.

Deprecated Methods in the Gantt Chart Widget Controller

- The methods `getResourceQueryName()` and `getEventsQueryName()` in `DbGanttBuilder` methods are not used internally by the `DbGanttBuilder` and may be removed later.
- Deprecated methods `loadResources` and `loadEvents` are not used in `GanttController` and will be removed in a later version.

New Resource Grouping for the Gantt Chart Widget

The Gantt Chart widget now allows grouping resources.

New Resource and Group Identification for the Gantt Chart Widget

- The Gantt Chart widget now allows identifying resources and groups by `internalId`.

New Context Menu for Gantt Chart Widget

The Gantt Chart widget now allows setting a custom context menu.

New Data Selection for the Gantt Chart Widget

Just like for the Data Grid/Explorer widgets, it is now possible to select events and resources from the widget, on the *None / Select / Highlight* basis.

Improved Event Loading for the Gantt Chart Widget

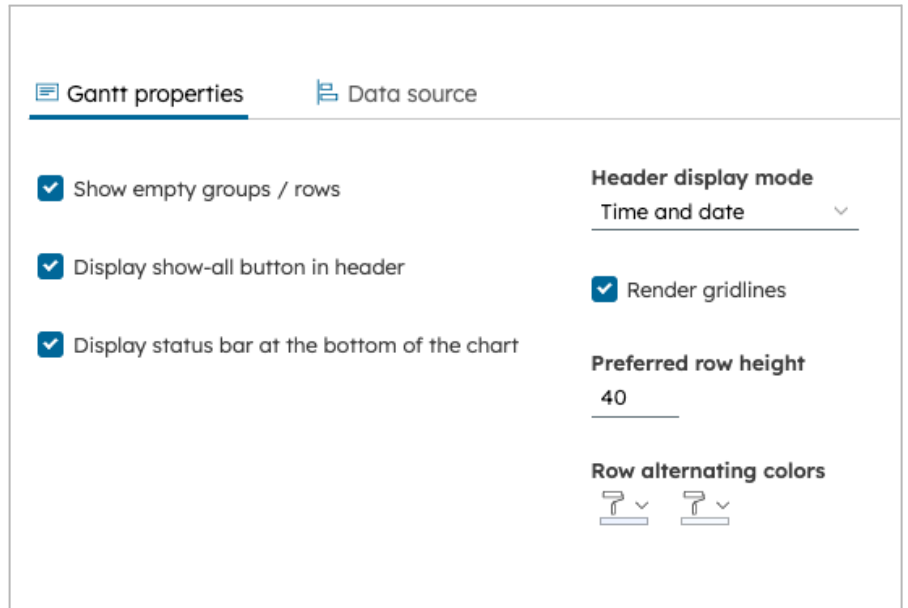
The Gantt Chart widget now allows loading events dynamically.

Improved Display Options for the Gantt Chart Widget

New options for row height and alternating colors are now available.

Also, the Gantt Chart widget now uses the same Time axis options as the Chart widget.

Finally, the zoom level and scroll position are now saved after leaving the view or dashboard.



Improved Event Labels for the Gantt Chart Widget Controller

Using a custom controller, it is now possible to set a renderer for the Event label.

Updated Date Parsing in the Gantt Chart Widget Controller

A bug related to parsing `LocalDateTime` fields was fixed in `TimeScaleHeaderRenderer`.

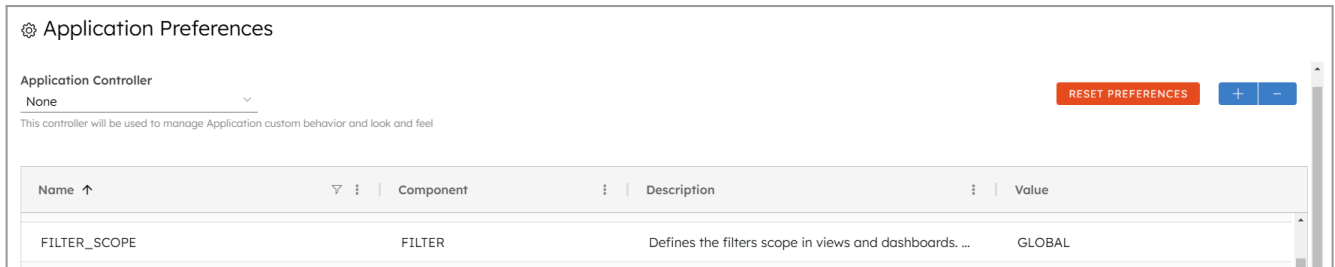
When using custom code to provide data to the Gantt chart, dates must be parsed with `parseGeneDate`.

Filter Widget

This section details all changes related to the Filter widget.

New Filtering Scope

In the Application Preferences, the new `FILTER_SCOPE` parameter is set by default to `GLOBAL`.



This means that the Filter widget and new Filter Bar apply to all views and dashboards across the application.

It can be set to `VIEW` to limit the filtering to the dashboard or view of the element.

Updated Method in the Filter Widget Controller

`GeneEntityFilter.fromSelection` is deprecated in favor of `GeneEntityFilter.fromConfig`.

To know if a `GeneEntityFilter.fromConfig` corresponds to a `GeneWidgetFilter`, you can use `getFromConfigComparator` from `@gene/components`.

Updated Parameter in the Filter Widget Controller

For the `buildFilters` function in `@gene/components`, the parameter `contextSelectionKey: string` has been replaced by `filterIdentifier: GeneFilterIdentifier`.

A `GeneFilterIdentifier` consists of a `typeName` and the paths leading to that `typeName`.

Map Widget

This section details all changes related to the KPI widget.



Improved Map Widget Controller

In the Map widget, the following custom controller method now allows overriding the marker series tooltip.

Unset

```
getMarkerPopupHtmlContent = (markerInfo: MarkerInfo): string => {  
    return `  
  
    This <i>${markerInfo.tag}</i> is located at  
  
    <ul>  
        <li><b>Latitude</b>: ${markerInfo.position.latitude}</li>  
        <li><b>Longitude</b>: ${markerInfo.position.longitude}</li>  
    </ul>  
  
    `
```

Pivot Table Widget

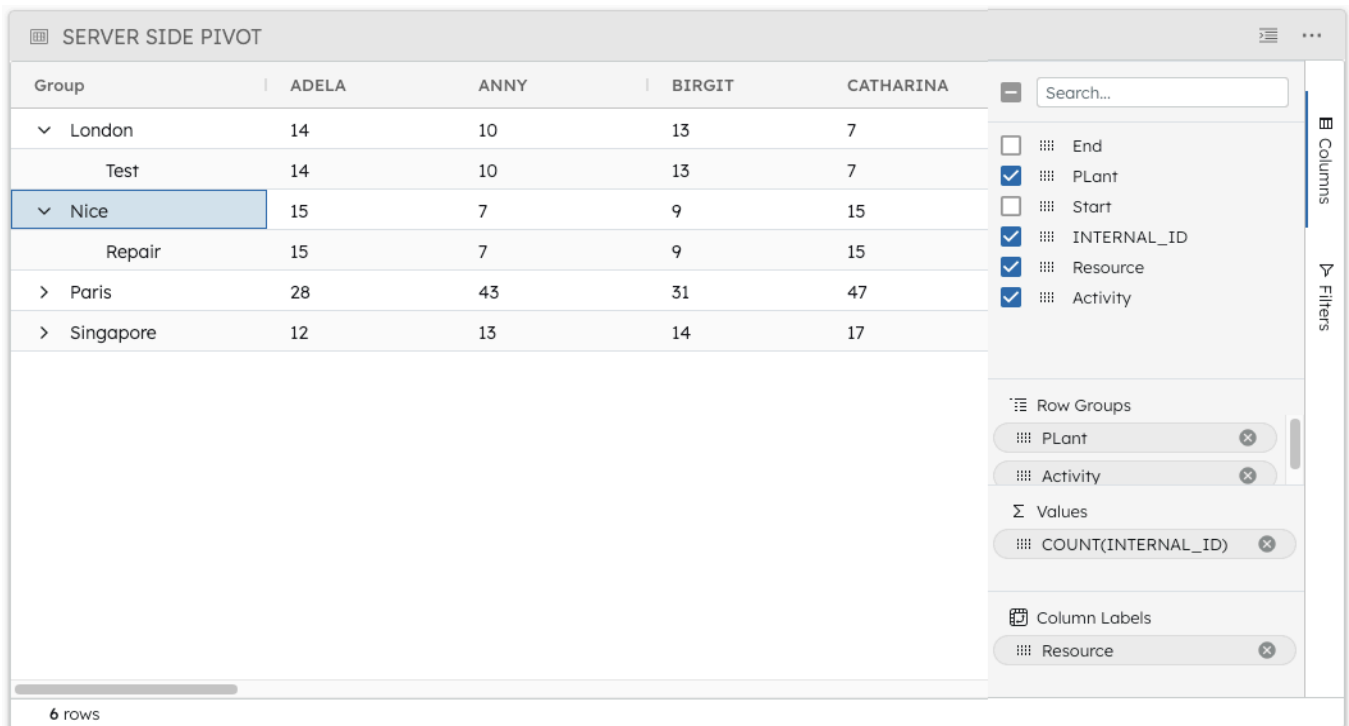
This section details all changes related to the Pivot Table widget.

Updated Pivot Table Items

Pivot Table (Experimental) is now renamed to Pivot Table and Pivot Table - Legacy is no longer available in the widget dropdown list.

Improved Pivot Table Widget

DOC 4.4.0 introduces a new version of the Pivot Table widget..



This new version allows using a server-side data source.

It leverages the AG Grid enterprise Pivot feature and can handle large scenarios.

Code Editor Widget

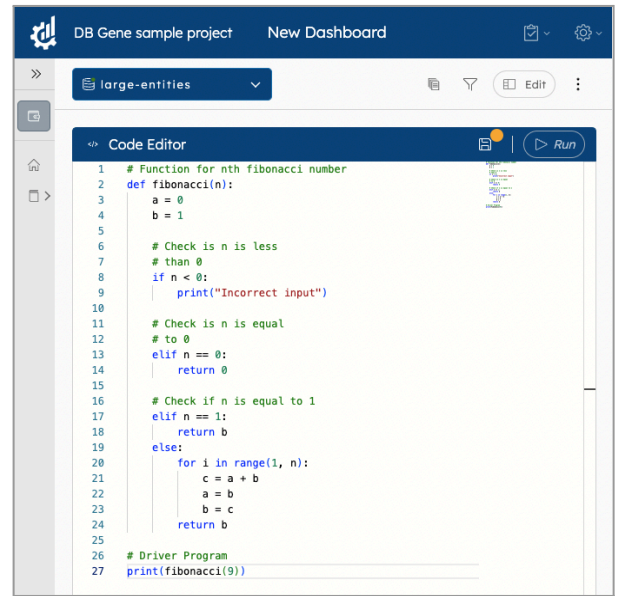
This section details all changes related to the Code Editor widget.

New Code Editor Widget

The new Code Editor widget, based on the Monaco widget from the VS Code project, allows users to:

- Edit and save free text with syntax highlighting capabilities in many languages.
- Use the text as input for a button in the widget toolbar to trigger an action such as running a task.
- Discuss with ChatGPT.

The content of a Code Editor widget is read from and saved as a parameter in a row of the **GeneParameter** entity of the current scenario. The **name** of the parameter can be edited in the widget configurator.



New Code Editor Task Input

Users can configure a button in the widget toolbar to trigger an action, such as running a task or any user-defined routine created beforehand.

Default tasks, allowing to 'Execute a Ruleset on a Scenario' for example, use the ID of the current scenario which can be edited to indicate a predefined scenario.

New Code Editor ChatGPT Task

The Code Editor widget can be configured with a button that launches a new default task **ProcessChatGptConversationTask** which turns the widget into a ChatGPT discussion terminal.

Markers » and – are used to differentiate the question and the answer, with question-answer pairs separated by lines of dashes.

Questions can be added below the last dashed line and sent to ChatGPT by clicking the button.

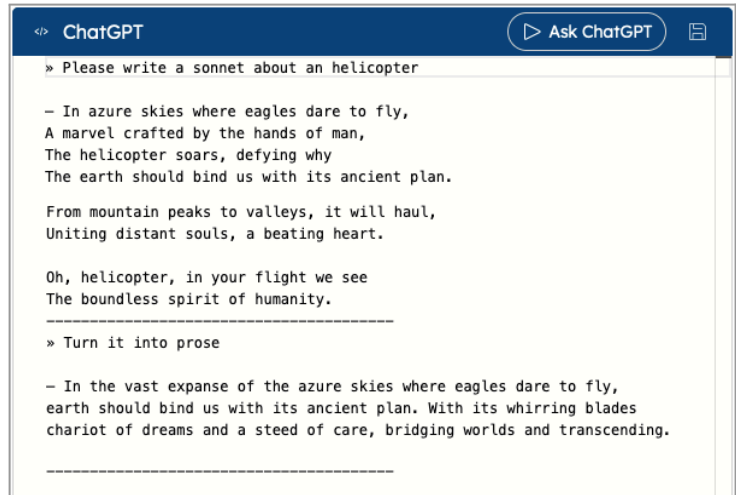
The text in the editor represents the conversation history which can be referenced in subsequent questions or cleared by hand before entering a new question.

A pre-prompt can also be provided as instructions for the GPT model. To achieve this, if the parameter `conversation` is associated with the Code Editor containing the questions and answers, the pre-prompt should be stored in the parameter `conversation-instructions`.

ProcessChatGptConversationTask

requires the following inputs:

- The current scenario;
- The name of the parameter associated with the Code Editor widget;
- An optional API key for OpenAI (useful if none is provided in the Spring properties);
- An optional model name (to override the default);
- An optional model temperature (to override the default).



The defaults for the LLM model and its temperature are configured as Spring properties in the `application.yml` file of the backend service extension. An OpenAI API key is required. This can be specified in one of the following ways:

- As a Spring property in the `application.yml` file of the backend service extension (not the preferred option);
- Locally, when running from an IDE or through Docker Compose, as an environment variable;
- In the `deployment/docker/app/.env` file when deploying with Docker Compose;
- In the values file of the deployment configuration when deploying with Kubernetes.

The cost of each execution of the `ProcessChatGptConversationTask` is calculated and included as an output of the task execution. This cost is determined based on official pricing and is configured via Spring properties in the `application.yml` file of the backend service extension.

More advanced use cases may involve programmatically computing the content of the conversation and instruction parameters, instead of relying on the Code Editor widget.

Rule Script Editor Widget

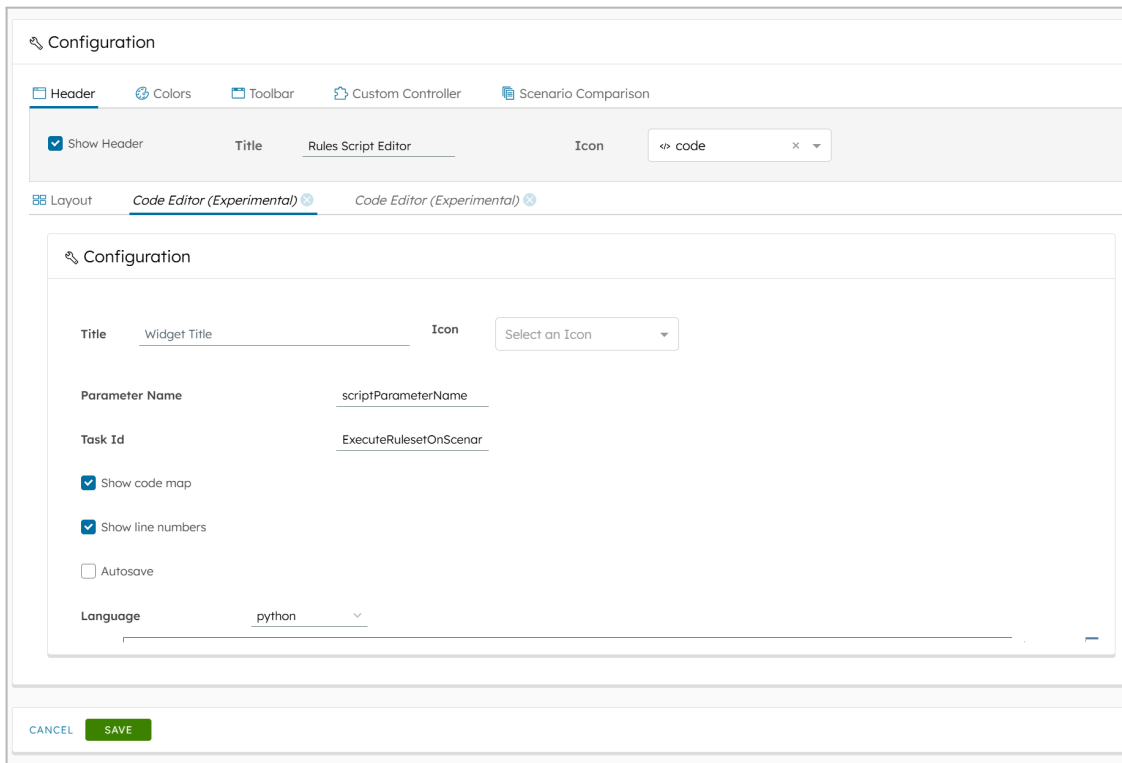
This section details all changes related to the Code Editor widget.

New Rule Script Editor Widget

The new Rule Script Editor widget allows executing operations, coded as Drools rules¹, on scenario data.

It consists of two Code Editor widgets, each configured by default with a toolbar button to run the built-in `ExecuteRulesetOnScenario` routine that executes a ruleset on a scenario. This routine takes two inputs:

- a `ScenarioData` input with the data of the scenario on which to run the rules, and
- a `string`, which can either be:
 - the name of a `GeneParameter` in the scenario above, from which the rules will be retrieved, or
 - the text of the rules themselves.



¹ For more details on the syntax of the Drools Rule Language (DRL), which is used with the Java-based Drools Rules Engine, please refer to the [Drools Documentation](#).

A Drools ruleset can be configured as follows:

```

Java
```drools
rule "Name of the rule, between quotes"
when
 $optionalVariable: ClassName(field == value, otherField < someValue)
 ...
then
 // some Java-like code using the $variables defined in the 'when' part
end
...

Example 1: Detecting and reporting issues
The rule engine tries to match each condition in the 'when' part with an entity instance in the scenario. For each successful match, the 'then' part is executed. An example of a (admittedly silly) rule that adds an issue if two employees have the same first name is:
```drools
rule "First names of employees should be distinct"
when
    $empl1: Employee($first1: firstName)
    $empl2: Employee(this != $empl1, firstName == $first1)
then
    helper.addIssue($empl2, "Employee has same first name as employee #%" + $first1.getId());
end
...

```

As this example shows:

- There can be several conditions, matching the same or different entity types.
- Variables can be bound to the entity instance matched by a condition, or to (the value of) a field.
- In the **when** part, one can use the field names (i.e. `firstName` as opposed to `getFirstName()`).
- In the **then** part, one must use getters.
- String comparison is made with **equals** even if written with `==`.
- A **helper** global variable is available with utility methods.

Any change that a rule performs is saved in the scenario.

The following example shows how to output logs, for instance, the number of GeneIssue in the scenario:

```
Java
```drools
rule "Count issues"
when
 accumulate(GeneIssue(); $n: count())
then
 logger.info("There are {} issues on the current scenario.", $n);
end
```
```

Two global variables are predefined. These variables can be used in the rules whenever useful.

- The `helper` global variable is bound to an instance of the `RulesCollectorHelper` class, which is scaffolded in the Backend Service extensions module. It natively contains a few methods to create issues (instances of `GeneIssue`) and add them to the collector. It is possible to add more utility methods that could be useful to the project at hand. These methods can then be called in the `when` or `then` parts of the rule. Note that methods with a side effect should not be called in the `when` part.
- The `dataset` global variable is bound to the collector that reflects the scenario.

Invoking this routine with the name of a `GeneParameter` typically corresponds to the use case where the rules are edited in a Code Editor widget.

The routine first retrieves and compiles the ruleset. It then creates the helper instance, sets the globals, and inserts all the entities of the scenarios into the rule engine working memory. Finally, it triggers the execution of the rules and marks the scenario as modified.

The routine exits with a code that describes how things went:

- `0`: The rules were successfully compiled and executed.
- `1`: The ruleset could not be retrieved.
- `2`: The ruleset could not be compiled.
- `3`: Executing the ruleset raised an error.
- `4`: Some other error occurred in the process.

All compilation and execution messages are added to the log, and emitted as an output of the routine, named `executionLogs`. If the ruleset was retrieved from a `GeneParameter`, the messages are also stored in another `GeneParameter` that has the same name, plus the `-logs` suffix.

The built-in `ExecuteRulesetOnScenarioTask` task is meant to be called from the Run button of the Code Editor widget. It takes as inputs:

- In an input named `scenario`, the ID of the scenario on which to execute the rules.
- In an input named `scriptParameterName`, the name of the `GeneParameter` that contains the text of the rules.

The task invokes the above routine, emits the log as a task output named alike, and exits in Alerting status if the routine exit code is not zero.

One way to extend the built-in behavior is to invoke the built-in routine in one of your tasks, to include it in a custom workflow. Note that the built-in routine compiles the ruleset on each invocation. Note also that it inserts the whole content of the scenario into the working memory of the rule engine. Depending on your use case, these elements may harm performance or not.

Another way to extend the built-in behavior is to extend the `helper` class.
