DB Gene 4.4.0 Migration Guide

December 11th, 2024

DB Gene 4.4.0 Migration Guide

- *Each section is structured as follows:* Removed, Deprecated, New, Improved, *and* Updated.
- *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* DB Gene Documentation and Release Notes.

Data Changes

This part lists all changes related to:

- *the Scenario Service API in Section **Scenario Service**.*
- *the Execution Service API in Section **Execution Service**.*
- *the Backend Service API in Section **Backend Service**.*

Scenario Service

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

Removed Property from 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.

Execution Service

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

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 lists 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 as follows, 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***.

Dev Changes

This part lists all changes related to the:

- *DB Gene external dependencies and libraries, such as Spring, Angular, or Keycloak in Section **3rd-Party Components**.*
- *DB Gene Scaffolding and Gradle scripts in Section **Build**.*
- *DB Gene security issues in Section **Security**.*
- *DB Gene integration with Docker or Helm in Section **Deployment**.*
- *DB Gene integration with Python, except for **Workers (Python)** and **Routines** in Section **Python**.*
- *DB Gene integration with Jupyter in Section **Jupyter**.*

3rd-Party Components

This section lists all changes related to DB Gene 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 DB Gene 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 Angular Dependencies

```
Unset
// ------ Updated Angular Dependencies


ag-grid-angular -> 32.3.1
ag-grid-community -> 32.3.1
ag-grid-enterprise -> 32.3.1
@angular/animations -> 18.2.10
@angular/common -> 18.2.10
@angular/compiler -> 18.2.10
@angular/core -> 18.2.10
@angular/forms -> 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
@angular/cli -> 18.2.11
@angular/compiler-cli -> 18.2.10
@angular/language-service -> 18.2.10


// ------ New Angular Dependencies


rimraf 6.0.1
```

Updated Java dependencies

```
Unset
// ------ Updated Java Dependencies


Spring Boot -> 3.3.5
Spring Cloud -> 2023.0.3
Spring framework -> 6.1.14
Keycloak 24.0.2 -> 26.0.5
RabbitMQ 3.13.0 -> 4.0.2
Jackson -> 2.17.2
Log4J -> 2.23.1


org.openapitools:openapi-generator-gradle-plugin: 6.3.0 -> 7.9.0
```

For more details on the Spring Boot update to 3.3.5, please refer to the Spring 3.3.5 Github Page.

Build

This section lists all changes related to DB Gene Scaffolding and Gradle scripts.

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.

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.
- Some parts of the `codeReplicas` block have been removed from the main `build.gradle` file.

Security

This section lists all changes related to any DB Gene security issues.

Updated Security for Trivy CVEs

Trivy CVEs are now fixed using RabbitMQ 4.0.2.

Deployment

This section lists all changes related to DB Gene integration to Docker and Helm.

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 lists all changes related to the Python integration to DB Gene, except for **Workers (Python)** and **Routines**.*

Updated Python Requirement

For Python development, DB Gene now requires Python 3.12.x.

Jupyter

This section lists all changes related to JupyterLab.

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.

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.

UI Changes

This part lists all changes related to:

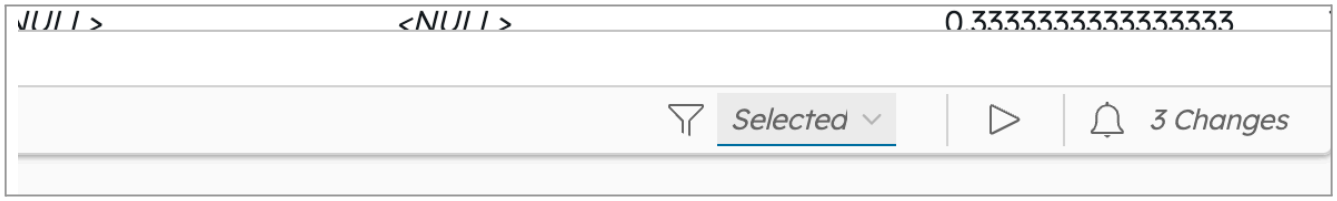
- *the custom renderers and controllers in Section **Extensibility**.*
- *specific UI widgets in Section **Data Grid/Explorer Widgets**, **Gantt Chart Widget**, **Filter Widget**, **Pivot Table Widget**, or **Code Editor Widget** — if a change impacts more than one widget, it is listed in the most relevant section and referred to in the others.*

Extensibility

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

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.



Data Grid/Explorer Widgets

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

Improved Date Filter in the Data Grid/Explorer Widgets

In the Data Grid/Explorer widgets, `DateTimecolumns` 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>*.

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 may be removed later.

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`.

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.

Improved Event Loading for the Gantt Chart Widget

The Gantt Chart widget now allows loading events dynamically.

Filter Widget

This section lists all changes related to the Filter widget.

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`.

Pivot Table Widget

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

Updated Pivot Table Items

Pivot Table (Experimental) is now renamed to *Pivot Table*.

Pivot Table - Legacy is no longer available in the widget dropdown list.

Code Editor Widget

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

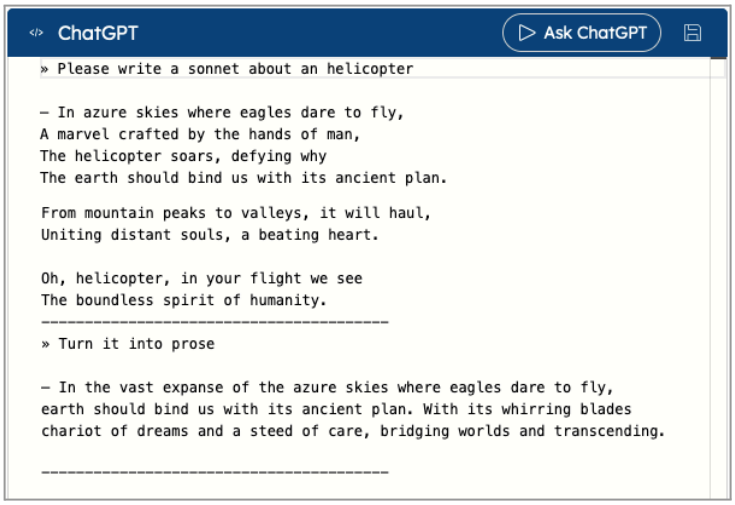
New Code Editor ChatGPT Task

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.
