Migration Guide
DB Gene 4.0.3-fp3

November 6th, 2023

DOC 4.0.3-fp3 Migration Guide

Application Changes

General

New Scenario Selector and Toolbar Visibility Options

Access Control

New Role “API_KEY_ADMIN”

Views & Dashboards

New Widget Deletion Message

New Widget Duplication Option

New Custom Home View

Workspaces & Scenarios

New Scenario List Widget Disabled “Actions”

New Scenario List Widget Custom Actions Configuration

Data Changes

Built-in Import/Export

New Automatic Import Scenario Name

New Automatic Export Scenario File Name

Data Integration Framework

New API Collector Loading

DBOS Changes

DBOS Master

New "resources" and "ttlSecondsAfterFinished" Configuration

Dev Changes

3rd-party Components and Libraries

Updated Gene UI Libraries

Updated Angular Dependencies

Updated AG Grid Dependencies

Updated Platform Library Dependencies

Updated Python Dependencies

New SpringBoot 3 GraphQL Dependency

Build

Updated Python and Docker Gradle Scripts

Updated Generator Gradle Plugin

Python

Updated Python Tasks

New Python DOM Collector API

New Per-Project Python Gradle Configuration

New Python IntelliJ Configuration

REST API

Updated Platform API

Updated Open API

Scripted Tasks Changes

Definition

Removed Execution Service Scripted Task Methods

Updated Contract of “Expression.evaluate()” Method

New Exit Script

Routines

New Routine Interruption Protocol

UI Changes

General

New Widget Configurator Tab “Filters”

New Luxon Value Formatter

Look & Feel

Updated Scenario List Widget "Empty Trash" Warning Message

Updated Application Management View "Revert" Warning Message

Updated Conflict Editor Information Message

Scenario Comparison

New Widget Always-On Scenario Comparison Mode

Tables

Updated Data Grid and Data Explorer Widgets Column Filter Indication

Updated Data Grid and Data Explorer Widgets Column Filter Labels

Updated Issue Details Group Widget Display

New Issue List Widget Warning Message

Charts

- Updated Chart Widget Layout Computation
- Updated Chart Widget Custom Controller “CategoryFormatter” Option
- New Chart Widget Custom Controller “GeneFieldDescriptor” Option
- New Chart Widget Custom Controller Grouping and Display Configuration
- New Chart Widget Custom Controller "Instant" and "LocalDateTime" Series Joint Display
- New Chart Widget Numeric and Temporal Fields Category Formats
- New Chart Widget "Split by" Option in Comparison Mode for "Radar" Charts
- New Chart Widget "Time axis" Option “Automatic” Parameter
- New Chart Widget "Time axis" Option Alignment on Months
- New Chart Widget "Time axis" Option Alignment on Weeks
- New Chart Widget "Parse as ISO" Option

Gantt

- Updated Gantt Chart Widget Custom Controller Tooltip Configuration
- New Gantt Chart Widget Tooltip Content Options
- New Gantt Chart Widget Tooltip Formatting Options
- New Gantt Chart Widget "Start" and "End" Time Nested Fields
- New Gantt Chart Widget Rendering Customization
- New Gantt Chart Widget Multi-Category “Color by” Option
- New Gantt Chart Widget Custom Controller “loadData” Method

KPI

- Updated KPI Widget "Conditional Formatting"
- Updated KPI Widget “Comparison Conditional Formatting”
- Updated KPI Widget CSS Class
- Updated KPI Widget Custom Controller
- New KPI Widget “Custom Query” Data Source
- New KPI Widget "Duration" Value Format
- New KPI Widget "Thousands (K)" and “Millions (M)" Value Formats

Filter

- Updated Filter Widget Search Field

Job

- New Job List Widget "Job duration" Column
- New Job Details Widget Output Size Display
- New Custom Controller Class in New Job Button Widget

- ★ For more details, please refer to the DOC 4.0.3 FP3 Documentation.
- ★ The order of the sections is based on the DOC 4.0.3 FP3 Release Notes Changelog.
- ★ The order of the subsections is as follows: *Removed*, *Deprecated*, *Updated*, and *New*.
- ★ Changes made to the DOC API are listed in Section REST API.
- ★ Changes made to the DOC dependencies are listed in Section 3rd-party Components.

Application Changes

General

New Scenario Selector and Toolbar Visibility Options

Users can now configure the visibility of the Scenario Selector and the Toolbar for any view or dashboard.

Access Control

New Role "API_KEY_ADMIN"

Users can now access the API Keys Management view to manage their API keys using the role `API_KEY_ADMIN`.

Previously, the role `PERMISSIONS_ADMIN` was required to list or delete the API Keys from every user. Now the same features can be achieved with the new role `API_KEY_ADMIN`.

The Keycloak provisioning has been updated accordingly and every member of the group `GENE_ADMINS` also has the role `API_KEY_ADMIN`.

Views & Dashboards

New Widget Deletion Message

Users are now prompted with a confirmation message before deleting widgets.

New Widget Duplication Option

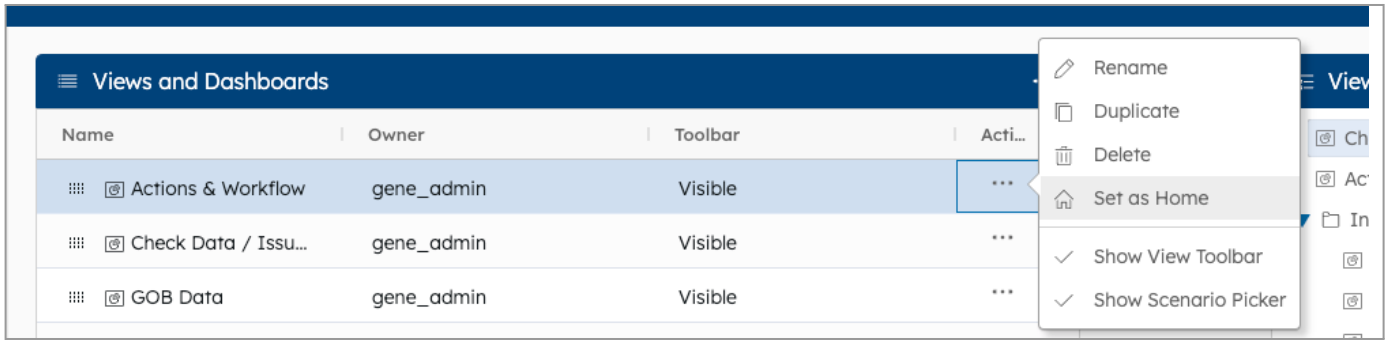
Users can now duplicate widgets and edit their layout in the process.

New Custom Home View

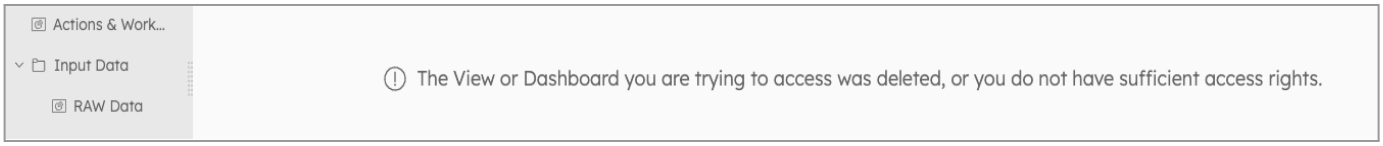
Starting with this version, any dashboard can be used as the Home View. The previous `HomeModule` of the web module has been removed from the scaffolded code.

The first time you open the web application, a check is performed to make sure a Home View is set. If that is not the case, a new one is created and set as the Home View. This setting is stored in the Application Preferences under `HOME_VIEW_ID`.

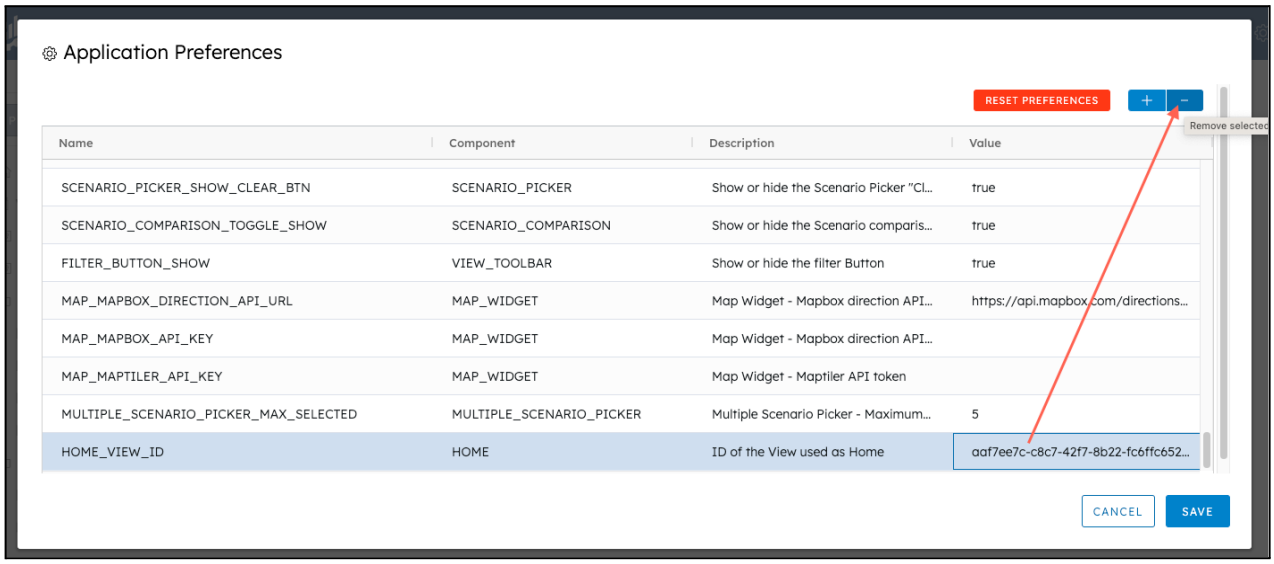
If you did not migrate to version 4.0.3-fp3 and still have the legacy `HomeModule`, a warning will be displayed in the browser console. Later on, you can decide to use any dashboard as Home View from the Views and Dashboards management view and configure the visibility of the toolbar and the scenario picker.



Note: *If you are importing an old version of an* `app-config.json` *that does not contain application preferences, you may have issues accessing the Home View.*



To fix this, you need to delete the ***`HOME_VIEW_ID`*** *setting through the Application Preferences. This will create a new Home View, after which, you will be able to export the application configuration file with the latest changes.*



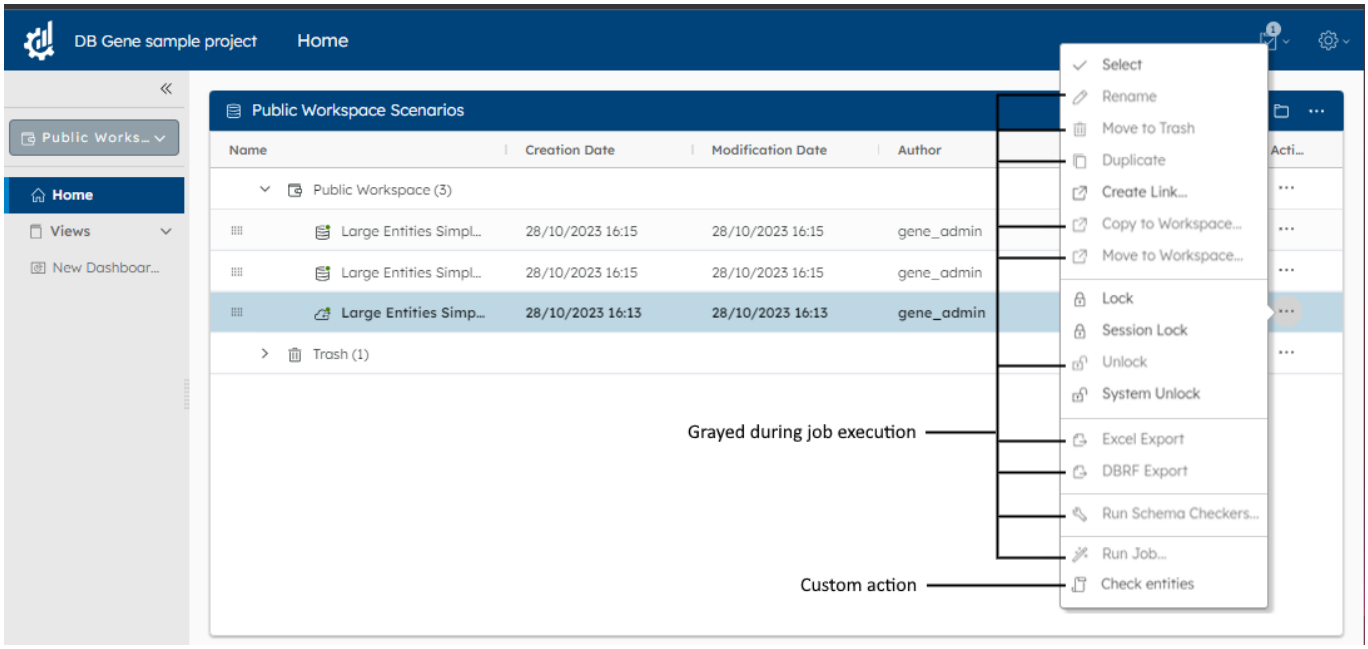
Workspaces & Scenarios

New Scenario List Widget Disabled "Actions"

In the Scenario List widget, actions that cannot be run in parallel on a scenario are now disabled and grayed out in the *Actions* menu when a task is performed on the selected scenario.

New Scenario List Widget Custom Actions Configuration

The Scenario List widget configurator now displays the tab *Custom Actions*.
From this tab, users can set custom tasks to display in the *Actions* menu of each scenario in the widget list. Custom Actions can also be set via custom controllers.



Data Changes

Built-in Import/Export

New Automatic Import Scenario Name

The name of an imported scenario is now based more accurately on the file name.

New Automatic Export Scenario File Name

The exported file name is now based on the scenario name.

Data Integration Framework

New API Collector Loading

Collectors are now loaded using the data service internal API.

DBOS Changes

DBOS Master

New "resources" and "ttlSecondsAfterFinished" Configuration

The DBOS worker-on-demand chart now allows `resources` and `ttlSecondsAfterFinished` configuration. Also, the Kubernetes Job that triggers the worker registration has been renamed from `wod-register-request` to `wod-registration-trigger`.

Dev Changes

3rd-party Components and Libraries

Updated Gene UI Libraries

The following classes, interfaces and types were moved from `@gene/task` to `@gene/component`:

```
class BasePickerModalComponent
class GeneBooleanInputModalComponent
class GeneBooleanInputModalModule
class GeneEntitiesInputModalComponent
class GeneEntitiesInputModalModule
class GeneNumericInputModalComponent
class GeneNumericInputModalModule
class GeneScenarioInputModalComponent
class GeneScenarioInputModalModule
class GeneTemporalInputModalComponent
class GeneTemporalInputModalModule
class GeneWorkspaceInputModalComponent
class GeneWorkspaceInputModalModule
class GeneEditionModalsModule
class GeneEditionModalService
class EmptyInputModalResponseImpl
class ValueInputModalResponseImpl
class CurrentInputModalResponseImpl
class GeneTaskInputTableComponent
class GeneTaskInputTableModule
class InputDefaultValueRendererComponent
class InputDefaultValueRendererModule
class InputEditModeRendererComponent
class InputEditModeRendererModule
class InputTypeRendererComponent
class InputTypeRendererModule

interface EmptyInputModalResponse
interface ValueInputModalResponse
interface CurrentInputModalResponse
interface GeneTaskInputRowData
interface InputConfigurationChangedAwareContext

type DateTimeMode
type InputModalResponse
```

Updated Angular Dependencies

DB Gene 4.0.3-fp3 now uses Angular 16.2.8. It was formerly version 15.2.0.
Please review the Angular 16 Migration Guide as Angular 16 introduces breaking changes.
For instance, how components are declared in the Angular modules needs to be updated.
Also, the `entryComponents` section needs to be removed. Here is an example below:

```
@NgModule({
  declarations: [SampleWidgetParametersComponent],
 // entryComponents: [SampleWidgetParametersComponent], <-- this line needs to be removed
with angular 16

    imports : [
        CommonModule,
        ClrIconModule,
        ClrStackViewModule,
        ClrDatagridModule
    ]
})

export class SampleWidgetParametersModule {
    constructor(customWidgetFactory: GeneCustomWidgetFactoryService) {
        customWidgetFactory.registerWidget(
            SampleWidgetParametersComponent.MANIFEST,
            SampleWidgetParametersComponent
        );
    }
}
```

Note: *The Angular compilation cache needs to be deleted.*
You can type the following command from the root folder of your project to achieve this goal:

```
(cd web && rm -rf .angular)
```

Please find below the Angular-related dependencies updated in 4.0.3-fp3:

```
// ------ Removed Angular Dependencies
@circlon/angular-tree-component

// ------ Updated Angular Dependencies
@angular/animations: 15.2.0  -> 16.2.8
@angular/cdk: 15.2.0  -> 16.2.7
@angular/common: 15.2.0  -> 16.2.8
@angular/compiler: 15.2.0  -> 16.2.8
@angular/core: 15.2.0  -> 16.2.8
@angular/forms: 15.2.0  -> 16.2.8
@angular/google-maps: 15.2.0  -> 16.2.7
@angular/localize: 15.2.0  -> 16.2.8
@angular/platform-browser: 15.2.0  -> 16.2.8
@angular/platform-browser-dynamic: 15.2.0  -> 16.2.8
@angular/router: 15.2.0  -> 16.2.8
angular-gridster2: 15.0.3  -> 16.0.0
uuid: 8.3.2   -> 9.0.0
@ng-select/ng-select: 10.0.3  -> 11.0.0
ngx-echarts: 4.2.2   -> 16.0.0
@danielmoncada/angular-datetime-picker: 15.0.2  -> 16.0.1
zone.js: 0.11.4  -> 0.13.1
core-js: 3.21.1  -> 3.31.0
html-duration-picker: 2.3.4  -> 2.4.0
@fullcalendar/angular: 6.1.4   -> 6.1.8
@fullcalendar/core: 6.1.4   -> 6.1.8
@fullcalendar/daygrid: 6.1.4   -> 6.1.8
@fullcalendar/interaction: 6.1.4   -> 6.1.8
@fullcalendar/list: 6.1.4   -> 6.1.8
@fullcalendar/timegrid: 6.1.4   -> 6.1.8
@ngx-translate/core: 14.0.0  -> 15.0.0
@ngx-translate/http-loader: 7.0.0   -> 8.0.0
@clr/angular: 15.0.0  -> 15.5.1
@clr/ui: 15.0.0  -> 15.5.1
@asymmetrik/ngx-leaflet: 15.0.1  -> 16.0.1
@asymmetrik/ngx-leaflet-markercluster: 15.0.0  -> 16.0.0
keycloak-angular: 13.0.0  -> 14.0.0
@angular-devkit/build-angular: 15.2.0  -> 16.2.5
@angular/cli: 15.2.0  -> 16.2.5
@angular/compiler-cli: 15.2.0  -> 16.2.8
@angular/language-service: 15.2.0  -> 16.2.8
ngx-color-picker: 12.0.0  -> 15.0.0
typescript: 4.8.4   -> 5.1.3

// ------ New Angular Dependencies
mobx@4.14.1
```

Updated Routing

Starting with Angular 14, the `CanActivate` interface implemented by all guards has been deprecated and replaced by functional guards. As a consequence, the `app-routing.config.ts` has been updated as follows:

```
Import
 { Routes } from '@angular/router';
import { GeneKeycloakAuthGuard } from '@gene/core';
import { GeneNoWorkspaceGuard } from "@gene/layout";
import { inject } from "@angular/core";

export const CUSTOM_ROUTES: Routes = [

 {
   path : 'home',

   loadChildren : () => import('@app/modules/home/home.module').then(m =>
m.HomeModule),

   canActivate : [
       (route,state) => inject(GeneKeycloakAuthGuard).canActivate(route, state),
       (_, state) => inject(GeneNoWorkspaceGuard).canActivate(state)
     ]
 },

 {
     path: '**',
     redirectTo: '/home'
 }

];
```

Removed "GeneDynamicWidgetComponent" Class

`GeneDynamicWidgetComponent` is no longer injected with a `ComponentFactoryResolver` as it was no longer necessary to create dynamic components. If you were extending this class, please update your constructor `super()` call.

Updated Typescript Target

The Typescript compilation target is now set to `es2022`. Since Angular 15, this has been the minimum compilation target possible.
This change will remove the following warning that the Angular CLI was giving during compilation:

```
TypeScript compiler options "target" and "useDefineForClassFields" are set to
"ES2022" and "false" respectively by the Angular CLI. To control ECMA version
and features use the Browserslist configuration. For more information, see
https://angular.io/guide/build#configuring-browser-compatibility

 NOTE: You can set the "target" to "ES2022" in the project's tsconfig to remove
 this warning.
```

Removed “ngcc” Angular Compatibility Compiler

Due to the removal of the Angular Compatibility Compiler (`ngcc`) in v16, projects on v16 and later no longer support View Engine libraries.
If you rely on libraries not compatible with Ivy, you should update them to a version compatible with Ivy. Contact the library author if there is no compatible version available, or look for alternatives.

New “GeneBaseDataWidget” “onResize()” Handler

The base method `onResize()` now takes an optional new parameter and can now be overridden in all widgets extending `GeneBaseDataWidget`.
The method will be called each time the browser window gets resized, or when the widget is being resized because of a layout change (eg: sidebar collapsed/expanded).

```
/**
* Notify the widget that it has been resized.
* @param event can be of type DOM UIEvent when the browser triggers a window resize,
* of type GeneUIEvent when being triggered from an applicative event (eg: sidebar
* hidden/showed) or undefined when not applicable
**/
  onResize(event?: GeneResizeEvent);
```

Removed “angular-tree-component” Package

`@circlon/angular-tree-component` package is no longer compatible with Angular 16, so the dependency was removed.

The code for `angular-tree-component` now lives in `@gene/component`.

If you were using it, you can safely replace your imports from `@circlon/angular-tree-component` to `@gene/component`.

New “ngx-echarts” Module

With the update to `ngx-echarts`, the following module needs to be added in the import of the AppModule for charts to work:

```
@NgModule({

    imports: [
        /** ... */
        NgxEchartsModule.forRoot({
            echarts: () => import('echarts')
        }),
    ],
    /** ... */

)}
export class AppModule implements DoBootstrap {
```

Updated AG Grid Dependencies

The Platform now relies on `AG Grid 30.2.0`.

```
ag-grid-angular: "29.3.5" -> "30.2.0"
ag-grid-community: "29.3.5" -> "30.2.0"
ag-grid-enterprise: "29.3.5" -> "30.2.0"
```

Updated Platform Library Dependencies

The Platform now relies on `platform-common-lib 1.0.3`.

Updated Python Dependencies

- `setuptools` => `build`.
- `python` minimal version `3.12`.
- `pandas` minimal version `2.0.0`.

New SpringBoot 3 GraphQL Dependency

The Platform Data Service now relies on SpringBoot 3 GraphQL Integration.
Also, the Data Service configuration has been renamed from `graphql.parser.max-tokens` to `services.data.graphql.parser.max-tokens`.

Build

Updated Python and Docker Gradle Scripts

Updated Python Gradle Script

Some adjustments have been made to the `python-library.gradle` file to ensure the Python tasks are re-run when necessary.

You need to replace the `gradle/templates/python-library.gradle` in your project with this new version (see below).

In this Gradle script, an important change has been made: all Python source files have been moved into a dedicated subfolder named `python`.

For each Gradle module that uses `python-library.gradle`, the following files have changed:

- `src` and `tests` folders: only moved, no changes inside.
- `requirements.txt` file: the path inside has changed.
- `package-equirements.txt` and `setup.py` files: only moved, no changes inside.

```
import java.nio.file.Files
apply plugin: 'com.decisionbrain.code-replicate-plugin'

// ----------------------------------------------------------------------------------------
// Configuration of Python Gradle plug-in: add pip modules

ext.PYTHON_DIR = "python"

ext.PYTHON_REQUIREMENTS = "${PYTHON_DIR}/requirements.txt"
ext.PYTHON_PACKAGE_REQUIREMENTS = "${PYTHON_DIR}/package-requirements.txt"

pythonVirtualenv {
    pip {
        requirements file(PYTHON_PACKAGE_REQUIREMENTS)
        editable file(PYTHON_DIR)
    }
}

// ----------------------------------------------------------------------------------------
// Implement the 'base' tasks: assemble, check

ext.PYTHON_SRC_DIR = "${PYTHON_DIR}/src"
ext.PYTHON_TEST_DIR = "${PYTHON_DIR}/tests"
ext.PYTHON_DIST_DIR = "$buildDir/dist"
ext.PYTHON_COVERAGE_XML = "$buildDir/coverage-reports/coverage.xml"
ext.PYTHON_COVERAGE_SQLITE = "$buildDir/coverage-reports/coverage.sqlite"
ext.PYTHON_SETUP_PY_DIR = PYTHON_DIR
ext.PYTHON_SETUP_PY = "${PYTHON_SETUP_PY_DIR}/setup.py"

tasks.register('pythonAssemble', Exec).configure {
    pythonVirtualenv.configurePythonExecTask(it)
    inputs.dir(PYTHON_SRC_DIR)
    inputs.file(PYTHON_SETUP_PY)
    outputs.dir(PYTHON_DIST_DIR)
    outputs.cacheIf { true }
    args  '-m', 'build', '--wheel', '--outdir', PYTHON_DIST_DIR, PYTHON_SETUP_PY_DIR
}
tasks.assemble.dependsOn pythonAssemble
tasks.register('pythonTest', Exec).configure {
    pythonVirtualenv.configurePythonExecTask(it)
    inputs.dir(PYTHON_SRC_DIR)
    inputs.dir(PYTHON_TEST_DIR)
    outputs.file(PYTHON_COVERAGE_SQLITE)
    outputs.cacheIf { true }
    args  '-m', 'pytest', "--cov=$PYTHON_SRC_DIR", "--cov-report=xml:$PYTHON_COVERAGE_XML",
"$PYTHON_TEST_DIR"
    doLast {
```

```
        delete(PYTHON_COVERAGE_SQLITE)
        Files.move(
            file(".coverage").toPath(),
            file(PYTHON_COVERAGE_SQLITE).toPath()
        )
    }
}
tasks.named('check') {
    dependsOn pythonTest
}
// ------------------------------------------------------------------------------------------
// Sonar
apply plugin: 'org.sonarqube'
rootProject.tasks.named('sonarqube') {
    dependsOn check
}
sonarqube {
    properties {
        property 'sonar.sources', PYTHON_SRC_DIR
        property 'sonar.language', 'py'
        property 'sonar.python.coverage.reportPaths', PYTHON_COVERAGE_XML
    }
}
// ------------------------------------------------------------------------------------------
// Management of module version number and 'install_requires' in setup.py
// We do it afterEvaluate so that codeUpdates.globalVersion can be overridden
project.afterEvaluate {
    codeUpdates {
        update(project.file(PYTHON_SETUP_PY)) {
            pythonVersion('version="%s"')
        }
    }
}
rootProject.tasks.named('pythonInitVirtualenv') {
    dependsOn project.tasks.updateCodeCheck
}
```

Updated Docker Gradle Script

The Docker Compose project and network names (used for isolating Docker Compose applications) are now configurable in the Gradle build.

You can customize them by editing the root `build.gradle` and modifying the `projectName` and `networkName` variables.

You need to run the `./gradlew updateCode` command after editing these variables.

```
codeUpdates {
    // Ensure the docker compose project and network names
    String projectName = constants.app.name
    String networkName = constants.app.name
    update(files(   'deployment/docker/app/.env',
                    'deployment/docker/dbos/.env',
                    'deployment/docker/infra/.env',)) {
        regex('COMPOSE_PROJECT_NAME=(.*)', projectName)
        regex('COMPOSE_NETWORK_NAME=(.*)', networkName)
    }
    update(files(   'deployment/docker/app/docker-compose-wml-worker.yml',
                    'deployment/docker/app/docker-compose-workers.yml',
                    'deployment/docker/app/docker-compose.yml',
                    'deployment/docker/dbos/docker-compose.yml',
                    'deployment/docker/infra/docker-compose.yml')) {
        pattern(  ~"\\s*networks:\\s*(.+):", networkName)
    }
    // --
```

Updated Generator Gradle Plugin

The generator has been modified with the following changes:

- the generator follows the same version as the rest of the platform:
  the Maven artifact will be `com.decisionbrain.gene:dom-generator-plugin:4.0.3-fp3`
- the generator plugin follows the DecisionBrain convention for Gradle plugin naming:
  the name will be `com.decisionbrain.gradle.gene.model-generation`
- you can add in your "`inputConfig`" file any parameter that is usually passed as command line argument (`--forceGenerationDir`, for instance)
- you can pass as command line argument any parameter that is usually placed into your "`inputConfig`" file (`--inputFile`, for instance)
- you can have missing parameters among your "`inputConfig`" file and the command line arguments: the generator will ask for the value of any missing required parameters
- a new optional parameter `masterJwtKey` is available to configure the DBOS master's JWT key (it is recommended to use the existing one when migrating, otherwise it will be generated).

Python

Updated Python Tasks

Python tasks are no longer "up-to-date" when there are changes in the *virtualenv*.

New Python DOM Collector API

Python developers can now manipulate scenario data using the DOM collector API.

New Per-Project Python Gradle Configuration

It is now possible to configure which Python executable to specifically use for the current project and your local environment, via Gradle properties.

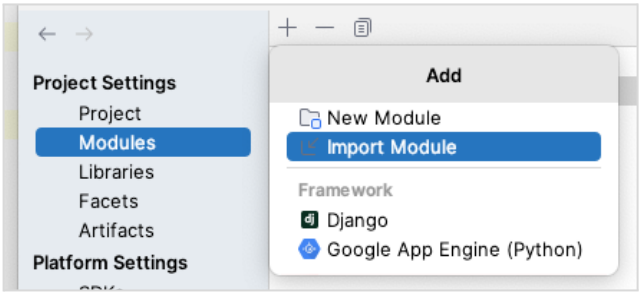
To do so, one can create a `gradle-local.properties` file in the root folder of the project, based on the example file `gradle-local.properties.sample`. This file is not committed (it is in the `.gitignore` file), so developers can configure the Python executable on their environment.

New Python IntelliJ Configuration

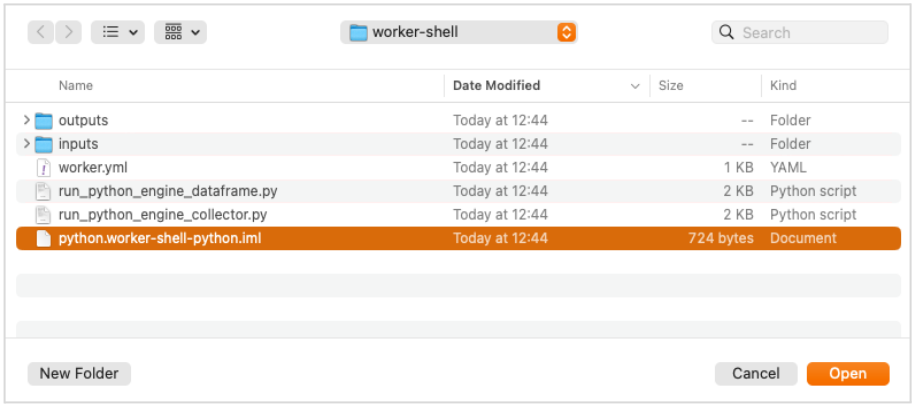
You can now easily configure your IntelliJ IDE to develop and debug Python code as follows:

1. From the command line, run `./gradlew pythonInitVirtualenv` if not yet done. Remember to do it if you update Python dependencies.
2. In the IntelliJ menu, go to File > Project Structure. A pop-up window opens.
   a. In the "*SDKs*" section, add the *virtualenv* generated by *gradle*:
      ★ Click on the button +.
      ★ Select "*Existing environment*" and provide the *Interpreter* from the directory `.gradle/python/virtualenv/bin/python` of the project root.

   b. In the Modules section, import the Python modules of the project by clicking the button +
   c. Selecting the following files using the previously configured *virtualenv* as SDK:



★`gene-model/gene-model-dom-python/python/python.gene-model-dom-python.iml`
★`processing/python-engine/python/python.python-engine.iml`
★`workers/python-engine-worker/worker-shell/python.worker-shell-python.iml`



You can now develop with IntelliJ, run tests with debug, navigate through the code, etc.

REST API

Updated Platform API

- `GeneTransactionalQuery` updated to correspond to the REST/GraphQL request parameter.
- Deprecated field `GeneTransactionalFieldValue#fieldValue` removed. Use `newValue` field instead.

Updated Open API

- `GeneTransactionalQuery` definition is updated. It is now aligned with the UI query.
- `GeneContextSelection` definition is updated. It is now aligned with the UI definition.
- `GeneTypeMetaModel` now contains a field named `singleton` that reflects the JDL file value `// DOM [single.row] : [true]`.
-

Scripted Tasks Changes

Definition

Removed Execution Service Scripted Task Methods

The following methods of `ScriptedTaskDescription` have been deprecated:

- `boolean getPersistSystemLog()`
- `void setPersistSystemLog(boolean)`

The log of a job, as displayed in the Log tab of the Job Details widget, no longer contains "system" logs. System logs, that is, the log of the execution service, used to be included if `setPersistSystemLog(true)` was called on the scripted task description. This method is now deprecated. System logs are still present in the log of the execution service.

Updated Contract of "Expression.evaluate()" Method

The contract of the `Expression.evaluate()` method has evolved. If you did not define a custom expression as a subclass of the Expression class in your project, you will not be affected.
The `Expression.evaluate()` method must now return a normalized value. This method returns a Java object, and this object must be normalized by calling `Type.getNormalizedValue()`. For example, if the evaluation of your expression returns a number, you should write:

```
@Override
public Object evaluate(ScriptExecutionContext executionContext) {
  ...
  var result = ...
  return NUMBER.getNormalizedValue(result);
}
```

Actually, for almost all types, the `getNormalizedValue()` method returns its argument unchanged. There are two important exceptions:

- For numbers, it converts ints to longs and floats to doubles. This is to avoid the fact that, for example, *ints* and *longs* are not comparable.
- For lists, it makes sure that the list is unmodifiable.

New Exit Script

In addition to its script, a scripted task can be given an exit script. The exit script is a sequence of statements. It cannot contain `AskInputStatements`. When executed, it inherits the job memory and outputs. Before executing the exit script, the job memory is augmented with variables (whose names start with `com.decisionbrain.gene`) to store the job exit status and the error, if any.

The exit script can basically do anything a regular script can do, including calling routines, adding outputs, or exiting with a different status. If the exit script does not include an `ExitTaskStatement`, the task exits with the status it had at the end of the main script. (For example, if the main script fails, the exit script is executed and the job ends in `FAILED`.)

The exit script is executed after the main script, whether the main script ends normally, or is interrupted by an exception or an `ExitTaskStatement`. The exit script is not executed if the job is aborted by the user.

Routines

New Routine Interruption Protocol

Routines are now interruptible.

UI Changes

General

New Widget Configurator Tab “Filters”

All the widgets that load data now allow displaying filtered data using the tab "Filters" in the configurator.

New Luxon Value Formatter

Users can now set a custom value format based on either ICU or Luxon.

Look & Feel

Updated Scenario List Widget "Empty Trash" Warning Message

The Scenario List widget *Empty Trash* warning message has now been rephrased.

Updated Application Management View "Revert" Warning Message

The Application Management view *Revert* warning message has now been rephrased.

Updated Conflict Editor Information Message

The Conflict Editor information message has now been rephrased.

Scenario Comparison

New Widget Always-On Scenario Comparison Mode

The Scenario Comparison Mode is now toggled on by default for all compatible widgets.

Tables

Updated Data Grid and Data Explorer Widgets Column Filter Indication

The Data Grid and Data Explorer widgets now display a more visible indication of an active filter on a column.

Updated Data Grid and Data Explorer Widgets Column Filter Labels

In the Data Grid and Data Explorer widgets, the column filter now displays *Before* and *After* for date fields instead of *Less than* and *Greater than*.

Updated Issue Details Group Widget Display

The Issue Details widget now displays all Issues of a Group selected in the related Issue List widget.

New Issue List Widget Warning Message

The Issue List widget now displays a warning if there are too many issues to display.

Charts

Updated Chart Widget Layout Computation

The Chart widget layout is now computed more precisely.

Updated Utility Method "toAggregationQuery"

The method `toAggregationQuery` that could be used to fetch data for a chart is no longer a static method of `GeneAbstractChartBuilder`, but is provided as a function in '`@gene/common-widget`'.

If you used it in a custom controller, change `GeneAbstractChartBuilder.toAggregationQuery` to `toAggregationQuery` and import `toAggregationQuery` instead of `GeneAbstractChartBuilder`.

Updated Chart Widget Custom Controller "CategoryFormatter" Option

The `CategoryFormatter` signature has changed and is now defined as the following:

```
(seriesConfig: GeneChartSeries, categoryField: GeneField, categoryValue: string) => string;
```

The first argument is the configuration of the series currently parsed by the chart builder, instead of the whole chart configuration. If the controller needs the whole configuration, it can retrieve it from the widget.

New Chart Widget Custom Controller "GeneFieldDescriptor" Option

A new interface `GeneFieldDescriptor` has been introduced to represent a field path with additional information. The category field for charts has been migrated from `GeneChartSeries.categoryByField (string)` to `GeneChartSeries.categoryField (GeneFieldDescriptor)`.

In a custom controller that implements methods that update the configuration such as `GeneChartController.updateConfiguration` or methods that read the configuration passed in the parameters (`GeneChartController.loadData`, `GeneChartController.getChartOptions`, etc.), all accesses to `GeneChartSeries.categoryByField` should be replaced with `GeneChartSeries.categoryField.fieldPath`.

New Chart Widget Custom Controller Grouping and Display Configuration

The Chart widget custom controller now allows setting grouping and display options for the time axis.

New Chart Widget Custom Controller "Instant" and "LocalDateTime" Series Joint Display

The Chart widget custom controller now allows displaying both `Instant` and `LocalDateTime` series simultaneously, by implementing `preprocessCategories` to align date and time.

New Chart Widget Numeric and Temporal Fields Category Formats

The Chart widget *Category format* now allows selecting numeric and temporal fields.

New Chart Widget "Split by" Option in Comparison Mode for "Radar" Charts

The Chart widget option *Split by* is now available in Comparison Mode for "Radar" charts.

New Chart Widget "Time axis" Option “Automatic” Parameter

The Chart widget option *Time axis* can now be set to *Automatic*.

New Chart Widget "Time axis" Option Alignment on Months

The Chart widget option *Time axis* now allows aligning data on months when all date fields fall on the same day of the month and all date-time fields also have the same time.

New Chart Widget "Time axis" Option Alignment on Weeks

The Chart widget option *Time axis* now allows aligning data on weeks when all date fields fall on the same day of the week and all date-time fields also have the same time.

New Chart Widget "Parse as ISO" Option

The Chart widget option *Time axis* now allows using the ISO date and time format for string categories with the option *Parse as ISO*.

Gantt

Updated Gantt Chart Widget Custom Controller Tooltip Configuration

Custom controllers now allow overriding the content of the Gantt Chart widget tooltip. The new version offers options to customize the tooltip contents. Consequently, if no change is made to an existing Gantt widget, the tooltip will no longer display the label field. In previous versions, it displayed ***Entity:*** *label*. It now only displays ***Entity***. This can be fixed by adding fields to the tooltip contents.

The interface for tooltip contents has changed. This means that existing custom Gantt controllers that implement the `loadEvents` method must be updated.

- In `GanttEvent`, `tooltipProperties` are no longer defined as `[string, string?][]`, but as `TooltipEntry[]`.
- The `TooltipEntry` interface contains the label, value, and optional `cssClass` properties used to display the tooltips.
- The `GanttController` also provides a new optional method to implement the `TooltipContentsProvider` interface. This interface will allow the controller to replace or complete the tooltip contents generated by the Gantt Chart.

New Gantt Chart Widget Tooltip Content Options

The Gantt Chart widget configurator now allows configuring information displayed in the tooltip.

New Gantt Chart Widget Tooltip Formatting Options

The Gantt Chart widget configurator now allows formatting numeric and temporal fields in the tooltip.

New Gantt Chart Widget "Start" and "End" Time Nested Fields

The Gantt Chart widget configurator now allows selecting the *Start* and *End* time fields by navigating in the dedicated dropdown menus.

New Gantt Chart Widget Rendering Customization

The Gantt Chart widget now allows rendering customization.

New Gantt Chart Widget Multi-Category “Color by” Option

The Gantt Chart widget now allows selecting multiple categories in the option *Event color by*. If two categories or more are selected and the widget configuration is saved, a dropdown menu *Color by* appears in the toolbar widget to easily switch between options.

New Gantt Chart Widget Custom Controller “loadData” Method

The custom Gantt chart controller interface now provides a new method:
`loadData?: (context: GeneContext) => Observable<GanttChartModel>`.
Custom controllers should now implement this method instead of `loadEvents` and `loadResources`.

KPI

Updated KPI Widget "Conditional Formatting"

The KPI widget configurator now allows setting value-based *Conditional Formatting*.

Updated KPI Widget “Comparison Conditional Formatting”

The KPI widget configurator now allows setting enhanced *Comparison Conditional Formatting*.

Updated KPI Widget CSS Class

The KPI widget CSS class `.gene-kpi-delta` becomes `.gene-kpi-comparison`.

Updated KPI Widget Custom Controller

The `GeneKpiWidgetConfiguration` has two new properties `format` and `refFormat` of type `GeneFormattingCondition[]` that hold the configuration of the conditional formatting.

The following properties in `GeneKpiWidgetConfiguration` are now deprecated: `positiveDeltaBackgroundColor`, `positiveDeltaColor`, `negativeDeltaBackgroundColor`, `negativeDeltaColor`. Use `format` and `refFormat` instead.

Also, the two following APIs of `GeneKpiController` have a new signature:

- `comparisonBackgroundColor?: (kpiValue: number, kpiDeltaValues: number[], kpiReferenceValues: number[]) => string | undefined;`
  Becomes: `comparisonBackgroundColor?: (kpiValue: string | number, kpiReferenceValues: string[] | number[]) => string | undefined;`
- `comparisonTextColor?: (kpiValue: number, kpiDeltaValues: number[], kpiReferenceValues: number[]) => string | undefined;`
  Becomes: `comparisonTextColor?: (kpiValue: string | number, kpiReferenceValues: string[] | number[]) => string | undefined;`

The missing parameter `kpiDeltaValues` can be computed from the other two parameters in the code of your custom controller.

New KPI Widget “Custom Query” Data Source

The KPI widget configurator now allows setting a data source using a *Custom query*.

New KPI Widget "Duration" Value Format

The KPI widget configurator now allows setting a *Duration* as *Value format*.

New KPI Widget "Thousands (K)” and “Millions (M)" Value Formats

The KPI widget configurator now allows setting a *Number* in thousands (K) and millions (M) as *Value format*.

New KPI Widget String Values Comparison Mode

The KPI widget now allows Scenario Comparison Mode for string values.

Filter

Updated Filter Widget Search Field

The Filter widget is now case-insensitive by default.

Job

New Job List Widget "Job duration" Column

The Job List widget now displays the column *Job duration*.

New Job Details Widget Output Size Display

The Job Details widget now displays the size of job output files next to their name in the Results tab.

New Custom Controller Class in New Job Button Widget

The New Job Button widget now accepts a custom controller. The custom controller can now manage the disabled state of the button, its label, as well as its behavior when users click on it. This can be performed by implementing the methods of its interface:

```
export class GeneNewJobButtonController extends
DefaultWidgetController<GeneNewJobButtonComponent> {

    /**
     * Implementing this method allows to control the disabled state of the button.
     * @param context$ The GeneContext observable
     */
    disabledState?: (context$: Observable<GeneContext>) => Observable<boolean>;

    /**
     * Implementing this method allows to control the label of the button.
     * @param context$ The GeneContext observable
     */
    label?: (context$: Observable<GeneContext>) => Observable<string>;

    /**
     * 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 taskConfig The task configuration of the button
     */
    clickHandler?: (event: PointerEvent, context: GeneContext, taskConfig: GeneTaskConfiguration)
=> void;
}
```