FABRIC V8.0.0 RELEASE NOTES

These Release Notes describe the new features in Fabric release V8.0.0 and list bugs that have been fixed since the V7.2.2 release.

Certification of this Fabric release is based on:

- Cassandra version 4.1.3
- SQLite version 3.44.1.0
- OpenJDK Runtime Environment 21.0.1
- Confluent Kafka version 7.2.1
- Neo4j 5.12.0 - enterprise
- Elasticsearch - 8.5.3
- AWS OpenSearch – 1.3.4
- PostgreSQL 15.4

MAIN FEATURES AND IMPROVEMENTS

1. Business Entity on PostgreSQL

- Fabric can now use PostgreSQL as a Logical Unit's storage layer, where each business entity's instance is saved in separate rows in PostgreSQL. This functionality should be used when the main use case is driven by cross-instance queries: for reporting, dashboards, or data analytics systems.

  https://support.k2view.com/Academy/articles/32_LU_storage/04_business_entity_on_pg.html

2. Fabric Catalog

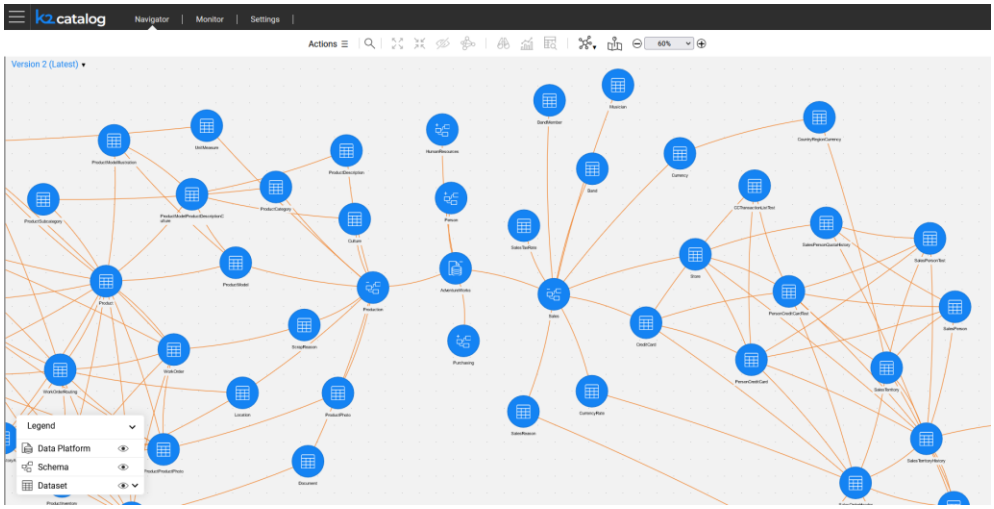
- **Complex fields support** – the Catalog now supports parsing of text fields with a complex structure (JSON or XML). It is done by a new **Complex Field Parser** plugin which uses the data snapshot taken from the source to parse the complex structures. Once the complex field is parsed, **Class** nodes are created. They are linked using a new relation type – **definedBy**. The complex structures then undergo auto-profiling by using the same profiling rules as all other Catalog fields.

  https://support.k2view.com/Academy/articles/39_fabric_catalog/plugins/01_complex_field.html

- The **plugins.discovery productization** – this configuration file is now a part of the product's resources and it is located in the `/fabric/resources/discovery` folder. When you need to override the product definitions on a project level (e.g. setting an exclusion list or disabling a plugin), the file should be copied to the Project tree under the `Implementation/SharedObjects/Interfaces/Discovery/` folder. Otherwise, there is no need to keep the file as part of your project.

  https://support.k2view.com/Academy/articles/39_fabric_catalog/04_plugin_framework.html

- Major performance improvement has been made to allow working with large schemas (over 2,000 nodes and relations) in the Catalog Application UI.
    - The performance improvement has been achieved by moving to a **cloud-like layout of the Catalog diagram** whenever all the schemas that are currently explored in the Catalog App have over 100 relations.
    - This cloud-like layout can help visualize the datasets and the relations between them much faster than the original **layered** layout. Note that you cannot switch back to the layered layout once the Catalog opens the cloud-like layout, thus preventing the client from getting stuck due to high number of nodes and/or relations.



- **Catalog Advanced Masking Settings** pop-up window enables setting up additional parameters for the masking:
    - **Masking indicators** (such as 'Use Environment'). These indicators can be used during task executions in TDM. They override the default values of the Masking Actor – on the Classification level.
    - **Formatter** name and parameters. The formatter can be an actor (e.g. **SimpleMaskingFormat** Actor described further) or an inner flow.

  https://support.k2view.com/Academy/articles/39_fabric_catalog/10_catalog_settings.html

3. Broadway

YAML Format

- Broadway flows are now saved in a **YAML format** (instead of JSON), along with some semantic changes, to improve the flow's readability and help users compare between 2 flow versions, when either committing the updates to Git or merging 2 files or file versions. Note that Broadway actors remain in a JSON format.
- The flow structure in a YAML format is as follows:

```
stages:
    <stage name>:
        actors:
            <actor name>:
                <various attributes - parent, condition, etc>
                in:
                    <actor's input param name>
                        external: <external name, applicable when param is external>
                        link: <actor/port name connected to this input param>
                out:
                    <actor's output param name>
                    schema: '#ref' <when object, the fields are described in 'schemas' section>
    split: '--------------------' <indication of split between stages of the same level>
schemas:
    <actor name>.out.<output name>:
        <list of properties and their data types>
```

  - The **stages** part of the file describes the sequence of stages in a flow, including the actors in each stage and input/output parameters of each actor. The input parameters that have a link display the link details. The input/output parameters also indicate whether they are defined as external.
  - The **schemas** part of the file describes the output parameters that are defined as an object.

- The existing flows are converted into YAML upon the first save of the flow. When the flow can't be converted into YAML due to conversion issues, it remains in JSON format.
- To leverage this feature for the **existing** flows, first save and commit the flow without any changes (so the only change would be the new YAML format). Then make your changes and compare the flow with the latest YAML version.
- Refer to the Fabric_Upgrade_Procedure_To_V8.0 document for more details on how to disable the feature.

General

- You can now define **Elseif** conditions in a flow as follows:
  - When a Stage is split into several stages on the same dependency level, you can define the **Elseif** logic by adding a Stage Condition and marking the same stage as else. The conditions of all stages of the same level will continue to be evaluated top -> down.

  https://support.k2view.com/Academy/articles/19_Broadway/19_broadway_flow_stages.html
- You can now define the **retry mechanism** for an actor in a flow as follows:
  - Retry can be enabled by adding an error handler to a stage. Any actor or an inner flow can be used for this purpose. Retry is then performed on each actor of this stage: if an actor fails and the error handler is triggered, the ***retry*** key word (instead of ***true*** or ***false***) is returned in order to continue. When using the ErrorHandler Actor, the retry mechanism is already built-in, as explained below.

  https://support.k2view.com/Academy/articles/19_Broadway/24_error_handling.html

Modified Actors

- **ErrorHandler** Actor has been enhanced to support the Retry mechanism setup. It now includes the retry attributes - **number of retries** and **interval duration** (in msec). The retry can be defined for either all exception types or only selected ones, if needed.
- **ErrorFields** Actor has been enhanced with a new output field - **attempt**, that indicates the execution attempt of the actor. This parameter can be used when creating an inner flow for the retry.

  https://support.k2view.com/Academy/articles/19_Broadway/actors/06_error_handling_actors.html
- **Masking** Actor has been enhanced with a new optional parameter - **formatter**. It can be set with either an actor or a flow, which can format the input value while **preserving the original format in the masked value**.

  https://support.k2view.com/Academy/articles/26_fabric_security/06_data_masking.html

New Actors

- **BroadwayLanguageConvert** Actor – for translating a Broadway flow file between YAML and JSON and vice versa.
- **YamlParser** Actor – for analyzing an input stream in a YAML format and returning the objects found in the stream.
- **YamlStringify** Actor – for converting any given object or primitive value to a YAML-format document.
- **SimpleMaskingFormat** Actor – for performing the formatting of the input value. The actor can work in 2 modes:
  - **Normalize** – normalizes the original value by removing all characters defined in an exclusion list (**formatDeny** param) and are not in an inclusion list (**formatAllow param**) from the original value.
  - **Format** – adds the formatting characters to a normalized value for building an output that has the same format as the input's original value.

  https://support.k2view.com/Academy/articles/19_Broadway/actors/07_masking_and_sequence_actors.html

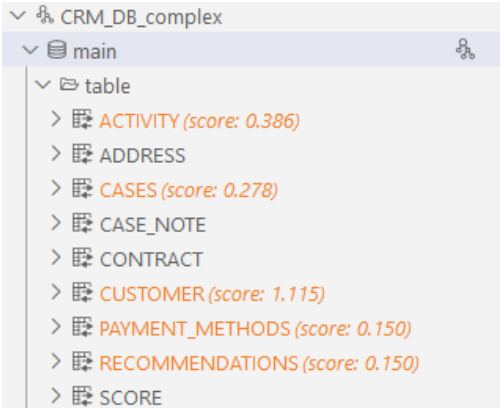
4. Hybrid Cloud Support

- Fabric supports multi–DC architecture when there is only an **https** connection between the DCs. For example: DC1 is on-prem, DC2 is on cloud, and in addition Fabric has access to the source database from one of the DCs only (e.g., on-prem). In order to support this architecture, the following features were introduced:
  - Several Fabric nodes can have direct access to the System DB, while the rest of the nodes use the direct-access System DB nodes as proxy. This capability is controlled by using ENABLE_SYSTEM_DB_PROXY parameter in the config.ini file and setting SYSTEM_DB_TYPE=remote_fabric:POSTGRESQL.

- While executing the Fabric commands **deploy** and **set**, a job is created with other DC as affinity to distribute the command on all other nodes in the cluster.
- While executing the Fabric commands **migrate**, **ps**, **node_status**, etc., the command runs on the local DC nodes only. It can be controlled by setting DISTRIBUTE_LOCAL_DC_ONLY parameter in the config.ini file.

5. Web Studio

- **DB Interface Explorer** has a new indication as to whether the interface data is retrieved from the data source or from the Catalog (neo4j). In addition, it is now possible to get a root table recommendation per interface and schema - whose data is retrieved from the Catalog. The recommended tables (including the recommendation score) are retrieved by clicking the icon on the schema level and are colored orange.



- **Schema reconciliation** – Tables Reconciliation allows you to analyze changes made to the connected source data platform tables upon which your project's tables are based on. It provides a valuable solution for 2 primary scenarios, throughout the project lifecycle:
  - Data Product Business Entity Model changes: During the initial setup of project tables from source data platforms, implementors may decide to select only those essential for the business entity model of the data product. Subsequent requirements may necessitate the incorporation of columns previously omitted. The Tables Reconciliation feature facilitates the identification of these columns for this prospective addition.
  - Source Data Platforms changes: Over the course of a project, changes may occur in the source data platform tables linked to it columns could be added or removed. Tables Reconciliation can be used to spot these differences between the source and project tables, enabling informed decision-making within the Table Editor.

  https://support.k2view.com/Academy/articles/06_LU_tables/07_reconciliation.html

6. K2exchange

- The K2exchange is a store-like platform for publishing and consuming Fabric's modules that are not part of the product and that are hence considered as extensions. Such extension modules can be Broadway actors and flows, connectors, templates, Java code and lib, LU web-app, and SQL queries.
- Web Studio allows project implementors to explore the extensions list, install and embed them into their project with a click.
- Additionally, Web Studio provides tools to create, pack, and publish extensions to the K2exchange store.

7. Miscellaneous

- The **JDK version** has been upgraded from 17 to **21**. The Fabric Studio now builds the code using **Java 17** (instead of 8). If you wish to use the new features of Java 21 in Studio, perform the following:
  - Update the **JavacArgs** from 17 to **21** in the **k2FabricStudio.exe.config** file under the **Studio** folder.
- **Fabric and project dependencies** – Fabric's dependencies are now isolated from the project's dependencies by default. This isolation allows a higher flexibility for the project to use the 3rd party JARs version rather than the version used by Fabric.
  - In the k2FabricStudio.exe.config, 2 new entries to set the compilation classpath
  - It is possible to disable the isolation and return to 7.2.x behavior.
  - Refer to Fabric_Upgrade_Procedure_To_V8.0 document for more details on how to upgrade your project to 8.0 and on how to disable the isolation feature.
- **In-memory LU storage** – Fabric storage mechanism has been enhanced with the ability to save the LUI data in the Fabric memory instead of saving it to SQLite files. To utilize this feature, the new parameter CACHE_TYPE in the config.ini file should be set to MEMORY_NO_CACHE (the default is FILES_CACHE). The advantage of keeping the LU Storage in memory is to prevent the waiting time caused by contention, when 2 Fabric sessions are simultaneously trying to access the same LUI.
- **Tomcat** has been upgraded from 8.5 to 10.

RESOLVED ISSUES

- Ticket #36762 – The redundant parameter READ_ONLY_AUTHENTICATOR in config.ini. The problem has been resolved.
- Ticket #37206 – Disappearing input params in Db Actors. The problem has been resolved.
- Ticket #37363 – When using Parallel Execution, the flow moves to the next stage before completing all actors in a stage. The problem has been resolved.