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 21.0.3
- 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 Exchange

TBD - Eyal

3. Fabric Catalog

- **Complex fields support** – the Catalog now supports parsing of text fields that include complex structures (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 use relation type – **definedBy**. Then the complex structures undergo auto-profiling using the same profiling rules as all other Catalog fields.

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

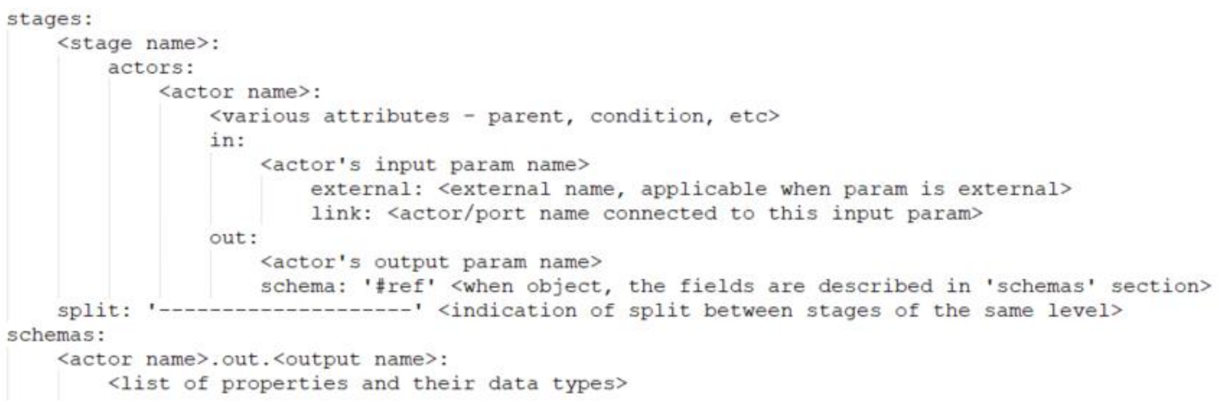
- The **plugins.discovery productization** – this configuration file is now a part of the product's resources and it is located in the `/fabric/resources/discovery` folder. When a project-level override needs to be done on the project level (e.g. setting an exclusion list or disabling a plugin), the file should be copied to the project tree under the `Implementation/SharedObjects/Interfaces/Discovery/` folder.
  https://support.k2view.com/Academy/articles/39_fabric_catalog/04_plugin_framework.html
- **Catalog Advanced Masking Settings** pop-up window enables setting up additional parameters for the masking:
  - Masking indicators (such as 'Use Environment')
  - Formatter name and parameters

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

4. Broadway

YAML Format

- Broadway flows are now saved in **YAML format**, to improve the flow readability and help users compare 2 flow versions, either when committing the updates to Git or when merging two files.
- The flow structure in YAML format is:



```
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, 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 if they are defined as external.
  - When the output parameter is an object, its schema is described in the **schemas** part of the file.
- 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 will be the new YAML format). Then make your changes and compare it with the latest YAML version.

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 as follows:
    - Retry can be enabled by adding an error handler to a stage. Any actor or an inner flow can be used for that purpose. Retry is then performed on each actor of this stage: if an actor fails and the error handler is triggered, it should return the ***retry*** key word (instead of ***true*** or ***false***) in order to continue.

  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 all exception types or only selected ones, if needed.
- **ErrorFields** Actor has been enhanced with a new output field: **attempt**, that indicates which execution attempt of the actor it is. 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 a formatter actor (or a flow), to **preserve the original format in the masked value** and to set the same masked values to all fields with the same normalized value.

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

New Actors

- **YamlParser** Actor – to analyze an input stream in YAML format and output the objects found in the stream.
- **YamlStringify** Actor – to convert any given object or primitive value to YAML document.
- **SimpleMaskingFormat** Actor – to perform the formatting of the input value. The actor can work in two modes:
    - **Normalize** – normalizes the original value by removing all characters defined in a backlist (**formatDeny** param) and are not in a whitelist (**formatAllow param**) from the original value.
    - **Format** – adds the formatting characters to 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

5. Stream Sync

- **Stream Sync** is a Fabric module, which enables Fabric proactive synchronization with source systems by processing only the changes in the source system, without the need to re-synchronize the entire instance on every change in the source.
- Stream Sync is applicable for **Business Entity over PostgreSQL (Fabric One)** only. It is configured via the config.ini file and it runs as a Fabric job.

  https://support.k2view.com/Academy/articles/40_stream_sync/README.html

6. 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, do 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 different versions of 3rd party JARs than those used by Fabric.
  - It is possible to disable the isolation and return to 7.2.x behavior.
  - For more details on how to upgrade your project to 8.0 and on how to disable the isolation feature, refer to Fabric_Upgrade_Procedure_To_V8.0 document.
- **Hybrid Cloud Support** – Fabric supports multi–DC architecture when there is only **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 System DB, while the rest nodes use the direct-access System DB nodes as proxy. This capability is controlled by using ENABLE_SYSTEM_DB_PROXY parameter in config.ini file and setting SYSTEM_DB_TYPE=remote_fabric:POSTGRESQL.
  - While executing 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 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 config.ini file.
- **Schema reconciliation** – TBD - Eyal
- **In memory faricdb** – TBD
- **Tomcat** has been upgraded from 8.5 to 10.

RESOLVED ISSUES

- Ticket xxx – xxx. The problem has been resolved.