FABRIC V7.2.0 RELEASE NOTES

These Release Notes describe the new features in Fabric release V7.2.0 and list bugs that have been fixed since the V7.1.1 release.

Certification of this Fabric release is based on:

- Cassandra version 4.1.3
- SQLite version 3.42
- Open JDK version jdk-17.0.3.1
- Kafka version 3.2.0
- Confluent Kafka version 7.2.1
- OrientDB - 3.1.3-tp3 and 3.2.4-tp3
- Neo4j 5.9.0-enterprise
- Elasticsearch - 7.16+ and 8.5.3
- AWS OpenSearch - 1.3.4
- PostgreSQL 15.4 (Debian 15.4-1.pgdg120+1)

MAIN FEATURES AND IMPROVEMENTS

1. Fabric Catalog (MVP)

Introducing Fabric's Catalog, a solution that provides an insight into the Fabric interfaces, starting with the RDBMS interface types in the MVP version.

- The Catalog’s discovery process includes the following steps:
    - Auto-discovery of the data source's elements (schemas, tables, fields) and the existing foreign key relations between them, while modeling the data source in the *neo4j* GraphDB.
    - Enrichment of the data model by creating the relations between the data source elements when the foreign keys don’t exist.
    - Auto-profiling of the data model elements by both the metadata (field name) and data (field value) to identify PII information and classify it based on pre-defined categories. The auto-profiling process uses a set of predefined rules that can be modified on a project level.
- The discovery results are presented in a new Catalog application (part of the Fabric Web Framework), which allows to:
    - Display the Catalog data model as a tree and navigate through its elements.
    - View the Catalog elements’ properties and manually edit them, if needed.
    - Support multiple Catalog versions and version comparisons.
    - Search any node by either keywords or additional advanced search parameters.

  - View and modify the profiling rules.
  - Create the Catalog artifacts based on a selected version.
- The DB Interface Explorer tab in the Web Studio presents the Discovery schema from neo4j rather than the data source schema. As a result, the Logical Unit creation is now based on the discovered and enriched data model.
- Masking can now be based on the Catalog's profiling results.
- The Catalog data model is exposed via the Fabric REST APIs.
- The Catalog solution is available in the Fabric Web Studio only.

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

2. Project Versioning

As project implementation is an ongoing process, ensuring clarity of the deployed code on the server becomes essential. Fabric provides a robust solution through tagging of project versions, allowing easy identification of the deployed version/s on the server.

The version creation process is done along with GIT tagging capability and is structured into 3 key steps across the project lifecycle:

- Version Tagging: Tag a version efficiently within the Studio.
- Fabric Object Deployment: Deploy Fabric objects, that have been appropriately tagged with a version, to the server.
- Version Assessment: Gain valuable insights into the deployed project objects' versions through the Fabric console or terminal.

https://support.k2view.com/Academy/articles/16_deploy_fabric/04_project_versioning.html

3. Support SQLite and Postgres as Fabric System DB

Until this release, only Cassandra could serve as Fabric System DB (to store history of jobs execution, etc.). Starting V7.2, it is possible to use either Cassandra, SQLite or PostgreSQL. Using SQLite as Fabric System DB enables to create a Fabric environment without installing Cassandra, so that SQLite can serve as a System DB and, for example, S3 or GCS as a Fabric storage.

The CQL command is now executed on the System DB rather than on Cassandra.

https://support.k2view.com/Academy/articles/02_fabric_architecture/06_cassandra_keyspaces_for_fabric.html

Check the Upgrade document for more details regarding the configuration changes:

https://support.k2view.com/Academy/Release_Notes_And_Upgrade/V7.2/Fabric_Upgrade_Procedure_To_V7.2.pdf.html

Important note: iidFinder solution doesn't support any System DB type other than Cassandra.

4. MDB Export / Import

Fabric now supports exporting or importing the MicroDB data from SQLite to another DB type (PostgreSQL). This solution allows to back up Fabric data, share it with others or import data from external data sources into Fabric.

The following Fabric commands have been introduced for this purpose:

- MDB_IMPORT with IID
- MDB_EXPORT with or without IID

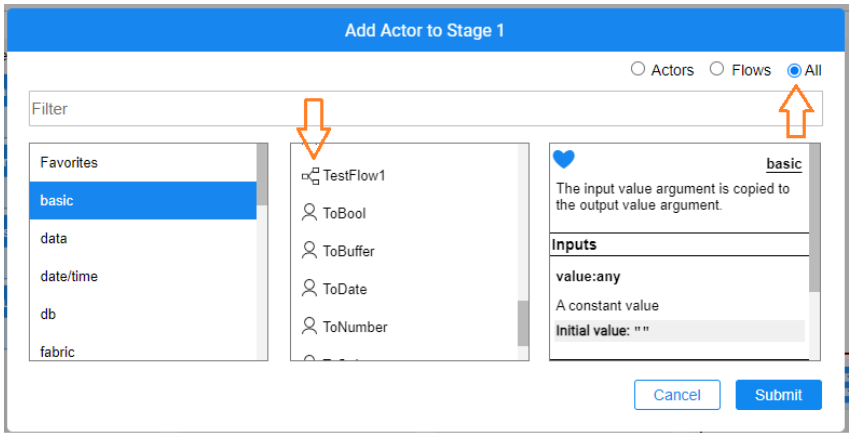
Both import and export commands have an optional parameter, EXCLUDED_TABLES, to specify a list of tables to be excluded from the import/export process.

https://support.k2view.com/Academy/articles/02_fabric_architecture/04_fabric_commands.html#mdb-export--import

5. Broadway

General

- The **Actor/Flow selection** pop-up window has been enhanced with an 'All' radio button that enables viewing both actors and flows in the same list. Each element has an indication (icon) that allows distinguishing between actors and flows.



- A new `com.k2view.regex` **plugin** has been added to the list of Broadway editors located in the `Server/fabric/staticWeb/editors` folder. It allows the inclusion of a predefined list of frequently used regular expressions to any Actor, e.g., the **Regex** and **RandomRegexGenerator** Actors (more details about the modified actors are found below). 2 separate lists are defined as part of this plugin:
  - A list of regular expressions for **parsing** - the list includes the expressions for parsing an input value, such as Email, US phone number and US SSN.
  - A list of regular expressions for **generation** - the list includes the expressions for generating a new value, such as US EIN, US ITIN, US phone number and Email.
  - Both lists' expressions can be edited on the project level.

Modified Actors

- **Regex** and **RandomRegexGenerator** Actors include a regex input argument. The editor of this parameter has been updated to `com.k2view.regex` described earlier in this document.
    - The **regex** argument has become an editable drop-down list. Once the expression is selected, it can be edited in the Actor's input field.
    - The **Regex** Actor retrieves the list of the expressions for parsing. The **RandomRegexGenerator** Actor retrieves the list of the expressions for generation (by setting the editor's input argument isGenerating = true).

New Actors

- The following 3 new actors have been introduced to apply the masking mechanism based on the Catalog's artifacts: **CatalogMaskingMapper, CatalogMaskingRecord** and **CatalogMaskingField**.
    - The **CatalogMaskingMapper** receives a data set, which maps the data on the fly, and does not load the entire data set to memory. The actor internally iterates on each entry and invokes the **CatalogMaskingRecord**. The actor returns a data set with the same structure as it received it.
    - The **CatalogMaskingRecord** receives a record, internally splits it into the pairs of key and value and invokes the **CatalogMaskingField** for each pair. The actor returns an object with the same structure as it received it.
    - The **CatalogMaskingField** checks if the field should be masked – based on the classification in the Fabric Catalog. If yes, it masks the input value using the **Masking** Actor and the generator (based on prior configuration).
    - For more information, refer to the Catalog documentation.
- The following 2 new actors have been introduced to generate synthetic data based on the Catalog's artifacts: **CatalogGeneratorRecord** and **CatalogGeneratorField**.
- The following actors have been introduced in the **data** category:
    - **Filter** – to filter a collection of values based on JavaScript or a Broadway flow and return the filtered list. The actor filters the data on the fly and does not load the entire list to memory.
    - **Limit** - to limit the number of elements read from an input collection. The actor evaluates the data on the fly and does not load the entire list to memory.
    - **Mapper** – to transform values in a collection of values, using JavaScript and/or Broadway flow logic. The actor maps the data on the fly and does not load the entire data set to memory.
    - **Peek** – to traverse values in a collection, and call JavaScript and/or Broadway flow logic for each entry. This is like **Mapper** but does not change the value (the result of the JavaScript /flow is ignored). The actor traverses the data on the fly and does not load the entire data set to memory.

6. Miscellaneous

- The .NET Studio has moved to a new encryption mechanism, AES 256, for all passwords in the project (used in the interfaces and environments). This encryption upgrade is done automatically upon opening of the project. Check the Upgrade document for more details:

https://support.k2view.com/Academy/Release_Notes_And_Upgrade/V7.2/Fabric_Upgrade_Procedure_To_V7.2.pdf.html

- Query Builder's default number of max rows in the .NET Studio was changed to 100. Additionally, a new QueryBuilderMaxRowsToFetch property was added to the Studio app config file (the default is 100).
- The default PubSub type has been changed to MEMORY.
- The existing **BATCH_DETAILS** command has been enhanced as follows:
    - The **LIMIT** parameter can now be set to -1, which means unlimited. It will then override the default setting of 10,000.
- The existing **DELETE INSTANCE** command has been enhanced with the new parameter mdbFinder = true (default) / false.
    - When set to true (default), delete the data from iid_info table.
    - When set to false, skip the deletion.

https://support.k2view.com/Academy/articles/02_fabric_architecture/04_fabric_commands.html

- The **jdbc-tester.sh** utility has been added to test the use of a JDBC driver and its compliance with APIs used by Fabric.
- The default timeout setting for deploy, drop, cancel batch and delete instance commands - **DISTRIBUTE_COMMANDS_TIMEOUT_MINUTES** - has been increased to 60 min.
- The Smart Proxy mechanism can now be defined per specific Web Service. It can be applied by setting the Web Service annotations *smartProxy*, *serverRedirect* and *affinity*.

https://support.k2view.com/Academy/articles/15_web_services_and_graphit/15_Fabric_smart_proxy.html

RESOLVED ISSUES

- Ticket 35222 – When using an Inner Flow, it loses schema definition. The problem has been resolved.
- Ticket 35605 – Default settings in iifconfig.ini. The problem has been resolved.
- Broadway - when an input parameter is removed from an LU function, it isn't refreshed in the actor that uses the LU function. The problem has been resolved.
- Resolved 3rd party libraries security vulnerabilities findings.