FABRIC UPGRADE PROCEDURE TO V8.0.0

This document describes:

- How to upgrade Fabric from the latest **V7.2.2** to the present **V8.0.0** version.
- How to re-implement the modified product features.

Notes:

- This document does not cover the Fabric server topology changes, such as nodes addition, DCs, change of replication factor and consistency level.
- It is a must to perform the Fabric upgrade procedure in the testing environments prior to applying it on your production deployment.
- Sanity tests must be performed upon the completion of the upgrade procedure, such as performing a few GET commands and conducting other checks per the sanity procedure defined in your project.

SOFTWARE UPGRADE PROCEDURE

Preliminary Step

Download the latest release of Fabric V8.0.0 package and copy it to the server(s).

Stop Fabric

Take the following steps in the specified order:

1. If your project has an iidFinder:
   - Stop the iidFinder on all nodes.
   - Wait until Kafka lags are zero in the relevant Kafka topics:
     - Delta_cluster_<LU_name> topics (e.g. deltaGroup)
     - DeltaPriority_cluster_<LU_name> topics (e.g. IDfinderGroupId_Cluster)
   - Run the following command to verify that the lag on the above topics is zero:

     ```
     /opt/apps/k2view/apps/kafka/bin/kafka-consumer-groups --bootstrap-server <internal Kafka server IP>
     --group <Kafka interface group ID> --describe
     ```

   - Investigate the remaining messages in the Delta tables and clean them, by taking the following steps per each LU:
     - Run MIGRATE command on all distinct IIDs.
     - Check the results in order to decide how to proceed with the failed entity messages.
     - Clean the Delta table.
2. Stop Fabric on all nodes.

Open the Package

Perform the following steps:

- Rename the Fabric directory as shown, type the specific Fabric version in the indicated location:

  ```
  cp -r config config_$(k2fabric -version |awk '{print $2}'|head -n1)
  mv fabric $(k2fabric -version |awk '{print $2}'|head -n1)
  ```

- Extract (un-TAR) the Fabric directories from the upgrade package (extract only the directories) as shown. The specific file name will depend on the specific Fabric version:

```
tar -zxvf k2fabric-server-fabric-<package_name>.tar.gz fabric
```

Run Upgrade Scripts (where applicable)

When upgrading to a version whose number is not consecutive to the version you are using, run all the upgrade scripts of all versions in between. For example, when upgrading from the latest V6.4 to V7.0, run all the upgrade scripts in the following order: 6.4 -> 6.5 -> 7.0.
The versions that have the upgrade scripts between the latest 6.4 and 7.1 are: 6.5.8, 7.0, 7.1.
The upgrade scripts should run from one Fabric node only. After running the scripts, verify that all the changes have been applied.

This step is not relevant for the upgrade procedure from the latest **V7.2.2** to the **V8.0.0** version.

Verify Upgrade Success

Use the following command to verify that the upgrade procedure has been completed successfully:

```
k2fabric -version
```

The result will display the Fabric package number, for example:

```
Tag fabric- 7.0.0_188 at revision = 3fc12ed1c6839614a9fd27e2894828bcd160988f
```

If there are local files on the server (such as local JARs), their names will be displayed here.

Configuration Changes

Prior to performing the changes, backup your project configuration files, such as config.ini, etc.
Please read the below configuration changes summary per each version.

To Version 8.0.0

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 in a version different from the version used by Fabric. Follow these steps to perform the upgrade correctly:

1. Upon opening the project in Studio V8.0, you might receive *ClassNotFoundException* compilation errors. Find (for example, via Google) what JARs contain the problematic Java classes.
2. There are two options to resolve the exceptions in the project:
   - By copying these JARs to the project's **lib** folder – to use the 3rd party JARs in a version different from the Fabric version.
   - Alternatively, by updating the Server and Studio configuration parameters as described in the table below – to use the same 3rd party JARs as used by Fabric.

Fabric Component	Configuration Update

Fabric Server	**PACKAGE_NAMES_CLASS_LOADING_FILTER** in **config.ini** should be updated with the problematic class package name.
Desktop (.NET) Studio	The **k2FabricStudio.exe.config** file under the Studio folder: • The **FabricFoldersClasspath** key should be updated with the JAR name. • In addition, the **FabricLibJarsToClasspath** key can be updated with specific JAR names.
Web Studio	The **referencedLibraries** in the Workspace **settings.json** should be updated with the JAR name.

   - Note that only the 3rd party JARs (and not Fabric JARs) should be copied to the project's **lib** folder. If a Fabric JAR is missing – add it to the Studio settings (as per the above table).

3. It is recommended to move all External JARs to the lib folder prior to deployment.
4. Once the compilation errors are resolved, deploy the project locally to verify that no more JARs are missing.

Note that additional steps are described in the Start Fabric section of this document. Make sure to follow these steps in order to successfully complete the upgrade process.

If you wish to **disable** the Fabric and project dependencies isolation and return to V7.2.x behavior, perform the following:

Fabric Component	Configuration Update
Fabric Server	**PACKAGE_NAMES_CLASS_LOADING_FILTER** in **config.ini** should be set to empty.
Desktop (.NET) Studio	The **k2FabricStudio.exe.config** file under the Studio folder: • The **FabricFoldersClasspath** key should be updated by adding the fabric\lib\fabric* value.
Web Studio	The **referencedLibraries** in the Workspace **settings.json** should be updated by adding /opt/apps/fabric/workspace/fabric/lib/fabric/*

Saving Broadway flow in a YAML format

Broadway flows are now saved in a **YAML format** (instead of JSON) by default. The existing flows are converted into YAML upon the first save of the flow. The flow format is controlled by the hidden parameter **BROADWAY_FILE_FORMAT** in the **[broadway]** section of the **config.ini**. The parameter's valid values are:

- **yamlTry** (default) – for saving a flow in a YAML format, first checking if the file after a "double conversion" (JSON > YAML > JSON) is identical to the original file. If not, the file remains in the JSON format and a warning with the issue is written to the log.

- **yaml** – for saving a flow in a YAML format.
- **json** – for turning off the saving of flows in a YAML format and keep them in JSON.

Fabric Keystore upgrade

The keystore has been upgraded from JCEKS to PKCS12. In case your project has been using the keystore, the following script must be executed:

```
/fabric/scripts/JCEKS-to-PKCS12.sh
```

Implementation Changes

Please see the below implementation changes summary for each version.

To Version 8.0.0

N/A

Start Fabric

Take the following steps in the specified order:

1. Prior to starting Fabric, set **PACKAGE_NAMES_CLASS_LOADING_FILTER** in **config.ini** to be empty, on all nodes.
2. Start the first Fabric node and wait until it has been successfully completed.
3. Start all other nodes.
4. Deploy the project.
5. Return to **config.ini** file and update the **PACKAGE_NAMES_CLASS_LOADING_FILTER** to the value taken from the local config.ini, on all nodes.
6. Re-start the first node again and then re-start all other nodes.
7. If your project has an iidFinder, re-start the iidFinder on all nodes.

Note that the above steps may vary per your project's runbook. For example, you may first restart the iidFinder on a single node only, verify that the process has been successfully completed and then proceed to restarting the iidFinder on all other nodes.