FABRIC UPGRADE PROCEDURE TO V7.0

This document describes:

- How to upgrade Fabric to the present version: from the latest **V6.5.9** to **V7.0.**
- 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 V7.0 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:
     a. Delta_cluster_<LU_name> topics
     b. DeltaPriority_cluster_<LU_name> topics
   - 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 | awk '{if ($6>0) print $0}'
     ```
   - Investigate the remaining messages in the Delta tables and clean them, by taking the following steps per each LU:
     a. Run MIGRATE command on all distinct IIDs.
     b. Check the results in order to decide how to proceed with the failed entity messages.
     c. 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 apps
```

Adding **apps** after **fabric** installs the correct Java version and creates a soft link to Java.

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.

V6.5.8 and V7.0 are the only 2 versions that have upgrade scripts between the latest 6.4 and 7.0.

The upgrade scripts should run from one Fabric node only. After running the scripts, verify that all the changes have been applied.

Run the upgrade script provided with the package of V7.0: **upgrade.sh**. The script includes the following logical parts:

- **No CDC Publishers** - deletes the CDC publishers as they are no longer needed. Note that the script should be executed only when no CDC data is left to be published.
- **Hash non-hashed/encrypted tokens** – prepare the hash column for the API_token table in Cassandra. Once Fabric has started for the first time after upgrade to V7.0, it will re-encrypt and hash the tokens.

The script will prompt for required arguments. These parameters define the Fabric user's credentials in Cassandra and the Fabric connection details:

- Username
- Password
- Host
- Port
- (optional) Cluster ID

Note: The script must run once per Fabric cluster, and it will be effective for all nodes under this cluster.

Verify Upgrade Success

Use the following command to verify that the upgrade 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 see the below configuration changes summary per each version.

To Version 7.0

- Configuration changes in **jvm.options** and **jvm.iid_finder.options** due to upgrade to Java 17:
  - Some of the flags used by Fabric to startup the JVM have been changed and need to be updated in your project. Thus, copy the configuration files **jvm.options** and **jvm.iid_finder.options** from the `Server/fabric/config.template` folder to your project and merge your changes into the updated file (if you have any).
- Fabric Configuration for PubSub / CDC / Common DB for using the PubSub abstraction layer:
  - Update the **config.ini** as follows:

Section name	Action & Description
[default_pubsub]	New section, to be added from the template. Configure the section with the Kafka connection settings. When working in Studio, update as follows: TYPE = MEMORY
[common_area_pubsub]	New section, to be added from the template.
[cdc_data_publish]	Should be deleted from config.ini.
[cdc_data_publish_ssl]	Should be deleted from config.ini.
[cdc_data_consume]	Should be renamed to [search_loader_pubsub].
[cdc_data_consume_ssl]	Should be deleted from config.ini.
[common_area_kafka_producer]	Should be deleted from config.ini.
[common_area_kafka_consumer]	Should be deleted from config.ini.

  - If you have CDC and/or Common DB components in your project, note that they also use the Kafka settings defined in the **[default_pubsub]**.
    - If some of the CDC configuration should differ from the default, update the section **[cdc]** with the specific parameters.
    - If some of the Common configuration should differ from the default, update the section **[common_area_pubsub]** with the specific parameters.
  - Note that iidFinder is not included in the PubSub abstraction layer solution. Therefore the [common_area_kafka_producer] and [common_area_kafka_consumer] sections of the iifConfig.ini are still relevant and should not be removed.
- CDC Architecture Configuration:

- Most of the parameters in the **[cdc]** section were deprecated - except for the below - so either update the section manually or copy it from the template and merge it:
  - **CDC_PUBLISH_MODE**: If you were using the **IF_SETUP** publish mode, you must now set this parameter to either **ON** (Always try to publish) or **OFF** (Never publish), as the **IF_SETUP** was dropped from V7.0.
  - **CDC_CONSUMER_JOB_AFFINITY**: Merge the value if needed.
- Make sure to perform the configurations related to the PubSub abstraction layer as explained earlier in this document.

➢ Configuration of Hardened Testing Environments:

- If you run tests on a hardened environment that does not use a real self-signed certification (usually a non-production testing environment), update the **config.ini** (and iifConfig.ini) by adding **SSL_HOSTNAME_VALIDATION=false** to the **[default_session]** section.

➢ Security Configuration:

- Fabric Open APIs spec - https://<fabirc-endpoint>/api - are now accessible for logged-in users only. An unauthenticated call will fail.
- If it's required to work without the authentication (as before), add **OPENAPI_AUTH = false** in the config.ini. The parameter is hidden under the **[fabric]** section and is set to true by default.

Implementation Changes

Please see the below implementation changes summary for each version.

To Version 7.0

➢ Starting from Fabric 7.0, the Fabric product code is encrypted. Any project, which invokes the core product functions that are not described in the Fabric documentation, will have to modify the implementation accordingly.

➢ JDK Version Upgrade to Java 17:

- Open your project in the Fabric Studio 7.0 and verify that the project is compiled.
  - When using the import of **sun.misc.BASE64Encoder** in your project, replace it with the import of **java.util.Base64** and update your code accordingly, for example:

    String encoded = Base64.getEncoder().encodeToString(bBytes);

    byte[] decoded = Base64.getDecoder().decode(encoded);
  - When importing from JDK modules, verify that the modules were not deprecated. In case they were - use an alternative library.
  - When using external JARs in the project, verify that they can be upgraded to a higher version of Java.

➢ Fabric Configuration for PubSub/CDC/Common DB:

- To start using the new PubSub abstraction layer, create a new interface with a type **PubSub Configuration**.
- Update the config.ini file as explained in the Configuration Changes section of this document.
- If it is required to have several Kafka brokers or if you need to override some of the parameters, create additional section(s) with the configuration and add their names to the **PubSub Configuration** interface.
- Update the Pub/Sub actors in your existing flows by setting their **interface** argument to the new interface of **PubSub Configuration**. Additionally, perform one of the following:
  - Define the **GROUP_ID, TOPIC and PARTITIONS** parameters on each Pub/Sub actor.

    Or,
  - Create a custom section with these parameters' definition and add the section's name to the **PubSub Configuration** interface.
- You can continue using the existing Kafka (or JMS) interfaces. If you wish to use them rather than the new Abstraction layer:
  - Keep the existing Kafka (or JMS) interface and don't update the actors.
  - The section **[default_pubsub]** in the config.ini must still be configured as explained in the Configuration Changes section of this document, because CDC and Common will refer to it for getting the Kafka configuration details.

➢ CDC Architecture:

- Since the **CDC_TOPIC** has dropped from the solution, ensure that this topic is empty (all messages are consumed) before starting the upgrade to V7.0.

➢ Broadway result structure:

- The Broadway command now enables defining the format of the flow output by setting the new command parameter: RESULT_STRUCTURE=<ROW/COLUMN>. The default is set to COLUMN, which means that the outputs are returned in a separate column.
- Until V7.0, the Broadway flow outputs were returned as each output in a row, which is the ROW format.
- When upgrading to V7.0, perform one of the following:
  - Set **BROADWAY_COMMAND_RESULT_STRUCTURE = ROW** in the project's config.ini file (the parameter is hidden under the **[fabric]** section and is set to COLUMN by default).

    Or,
  - Add the parameter RESULT_STRUCTURE=ROW when running the **broadway** command from the User Code.

    Or,
  - Adjust your code in order to get the output from the new result structure.

- iidFinder:
  - All code references to **IidFinderUtils** class should be replaced with **IidFinderApi** class.
    - For example, the line *IidFinderUtils.getKeyspace()* should be replaced with *IidFinderApi.getKeyspace()*.
  - Since *com.k2view.cdbms.finder.api.IidFinderApi#getDeltas* API has been changed and now expects the LU proactive indicator, replace it with one of the following:
    - *com.k2view.cdbms.lut.UserCodeDelegate#getDeltas*
    - *com.k2view.cdbms.lut.UserCodeDelegate#getDeltasStream*
  - Access to all other iidFinder Util classes is not recommended and falls under the user's responsibility.
- GraphIt:
  - In order for the existing GraphIt files to support all the new features of V7.0 (such as setting the "Additional Permissions" and "Require Authentication" properties), move them under the LU's GraphIt folder and the predefined category.
  - This is optional. If not moved, the GraphIt files will continue working as before V7.0.
- CommonDB:
  - Starting from V7.0, the old snapshots are not deleted automatically. If the automatic delete is needed, set **DELETE_OLD_SNAPSHOTS_ON_SYNC = true** in the **[common_area_config]** section of the config.ini.

Re-Start Fabric

Take the following steps in the specified order:

1. Re-start the first Fabric node and wait until it has been completed successfully.
2. Restart all other nodes.
3. Deploy the project.
4. 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 re-start iidFinder on a single node only, verify the process has completed successfully and then proceed to re-starting it on all other nodes.