FABRIC UPGRADE PROCEDURE TO V6.4.7

This document describes:

- How to upgrade Fabric to the present version: from the latest **V6.3** to **V6.4.7**:
- How to reimplement the modified product features.

Notes:

- This document does not cover the Fabric server topology changes, such as addition of nodes, DCs, change of replication factor or 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 test must be performed upon the completion of the upgrade procedure, such as performing few GET commands and doing other checks per the sanity procedure defined in your project.

SOFTWARE UPGRADE PROCEDURE

Preliminary Step

Download the latest release of Fabric V6.4 package and copy it to the server(s):

Studio 6.4.6
Server 6.4.6

Stop Fabric

Do these steps in the specified order:

1. If your project has iidFinder:
   - Stop 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 check the lag on those 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 doing the following steps per each LU:
     a. Run MIGRATE command on all distinct IIDs.
     b. Check the results to decide how to proceed with the failed entities messages.
     c. Clean the Delta table.
2. Stop Fabric on all nodes.

FABRIC UPGRADE PROCEDURE

Open the Package

Do the following steps:

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

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

2. 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 (if relevant)

When upgrading to a version which is not immediately after the version you are using, run all the upgrade scripts of all versions in between. For example, when upgrading from the latest V6.2 to V6.4.2, run all the upgrade scripts in the following order: 6.3 -> 6.4.

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

Upgrade scripts are included in the package only when the version has schema related changes. If there are no such changes, the upgrade script will not exist.

This step is not relevant for the upgrade from the latest V6.3 to the latest V6.4.

Verify Upgrade Success

Use the following command to verify that the upgrade was successful:

```
k2fabric -version
```

The result will display the Fabric package number as follows:

```
Tag fabric-6.4.2_47 at revision = 9078ce82cffc548feb671bf1fb49258bc02bb535
```

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

Re-Start Fabric

Do these steps in the specified order:

1. Backup your project configuration files: config.ini and iifConfig.ini.
2. Re-start Fabric on all nodes.
3. Deploy the project.
4. If your project has iidFinder, re-start iidFinder on all nodes.
5. Compare the new configuration files with your project to verify if there are changes which should be applied to the project configuration.

Note that the above steps may vary per your project's runbook. For example, you might first re-start iidFinder on one node only, verify the success and then proceed to re-start it on all other nodes.

IMPLEMENTATION CHANGES

To Version 6.4.0

- **iidFinder** Partitioned Delta mechanism was added to enhance Delta handling performance. The details about this change, the configuration changes and implementation recommendations can be found in the **Farbic_iidFinder_Release_Notes_V6.4** document.
  https://support.k2view.com/Academy_6.4/Release_Notes_And_Upgrade/V6.4/Fabric_Release_Notes_V6.4.0_iidFinder.pdf.html

- **MaskingSequence** Actor's properties were updated as follows:
  - RedisInterface property was removed.
  - SequenceInterface property was added, which enables using DB for sequence handling.
  - initialValue property can now be an Inner flow.
  - useEnvironment property was added, which indicates whether to separate a masked value per the environment.
  - Lazy sequence support was added.

  If the Actor is already in use in your project, adapt it to the new behavior.
  https://support.k2view.com/Academy_6.4/articles/19_Broadway/actors/07_masking_and_sequence_actors.html

- **Subscribe** Actor has been changed and now subscribes to all partitions when **Partition** property value is set to -1.

- **Broadway Transaction** - the management of Commit-per-Iteration has been changed. A non-transactional Stage should be added at the end of the loop, so that the commit will be performed per each iteration.
  https://support.k2view.com/Academy_6.4/articles/19_Broadway/23_transactions.html

- **Broadway Iterations** – when the flow is split into branches and the loop starts before the split, the loop is now effective per the whole level and includes all branches. In the previous version a loop could be set per branch.
  In addition, when iterating over two iterators of different size in parallel, Broadway now returns null when one result set is finished but the other one still has values.
  If Broadway iterations are in use in your project, adapt them to this new behavior.
  https://support.k2view.com/Academy_6.4/articles/19_Broadway/21_iterations.html

- **Regular/Computed Field** column type was removed from the LU table schema.
  If you were using Computed fields in your project, modify the LU to carry out the previous functionality.

- Implementation functions changes:
  - The **ludbFunctions** function was replaced by **invokeFunction**:
    public Object invokeFunction(String name, Execution execution, Object... args);
  - The new **getDeltasStream** function is introduced under the iidFinder API in the UserCode and it replaces **getDelta** that will be deprecated in the future version.

To Version 6.4.1

- **Data Catalog Interface type** – the **Transactional** connection setting is now set to **false** by default. Using the transactional interface synchronizes the Catalog Write executions in parallel, which will be used in multi-Fabric clusters with a single catalog DB instance.
  If the Data Catalog interface is already in use in your project, check and update this setting if needed.
  https://support.k2view.com/Academy_6.4/articles/33_data_catalog/02a_data_catalog_interface.html

- The **Search** command has been enhanced to support the following date formats by default:
  - yyyy-MM-dd
  - yyyy-MM-dd HH:mm:ss
  - yyyy-MM-dd HH:mm:ss.SSS

  This change applies to newly defined date columns only. A full reindex is required for existing values.

- **iidFinder data type change** – instead of returning **double** data type for numbers in message, iidFinder now returns them as **long**.
  Example:
  - Before this version: dataChange.getValues().get("BAN") returned **double** data type although BAN is defined as **long** in the source DB).
  - Now: dataChange.getValues().get("BAN") returns **long** data type since this is how BAN is defined in the source DB.

  If you are using this method and dealing with long data types, adapt it to the new behavior.

- In the **Web Services**, the **Boolean** class default is null, and primitive is false. Prior to this version, the default for Boolean class was false. If your project includes Web Services which are using Boolean arguments, adapt it to the new behavior.

To Version 6.4.2

- N/A

To Version 6.4.3

- Elasticsearch now requires an interface name 'search' (in small letters).
  If you are using the Elasticsearch, you need to adjust the interface name.

To Version 6.4.4

- N/A

To Version 6.4.5

- iidFinder process was enhanced to allow it to consume the next several Kafka batches to avoid getting stuck on a message that takes a long time to process. The number of upcoming batches to be consumed (ie, how many are "several batches") is controlled by a new parameter added to iifConfig.ini called MAX_ACTIVE_MESSAGE with a default value 2500. This parameter can be fine-tuned according to the needs of the project. Specifically,

you should take into account memory consumption and performance impact vs. the utility of this feature. iidFinder can still read several kafka batches in parallel and is still controlled by the KAFKA_POLL_RECORDS parameter. It is recommended to set MAX_ACTIVE_MESSAGE as KAFKA_POLL_RECORDS multiplied by ~25-50.

To Version 6.4.6

- Added the following parameters to the [search engine] section of the config.ini file:
  - BULK_PROCESSOR_MAX_CONCURRENT_WORKERS: maximum number of concurrent threads to process the bulk actions. Default value is 5.
  - BULK_PROCESSOR_MAX_ACTIONS: Bulk size. The maximum number of actions (requests) in one bulk. Default value is 1000.

To Version 6.4.7

N/A