etlsdk Documentation

Release 1.0

Eigen

Dec 08, 2020

Contents:

1 **3**
- 1.1 plugin . . . 3
- 1.2 . . . 3

2 Demo **7**
- 2.1 1. Demo of plugin on pony . . . 7
- 2.2 2. Demo of plugin terminal . . . 7

3 etlsdk **9**
- 3.1 method_canonical . . . 9
- 3.2 etlsdk_args . . . 9
- 3.3 plugin_args: inputs / outputs . . . 10
- 3.4 plugin_args: args . . . 10

4 **13**
- 4.1 config . . . 13
- 4.2 . . . 14

5 etlsdk API **15**
- 5.1 . . . 15
- 5.2 api . . . 16
- 5.3 . . . 23
- 5.4 UT . . . 23
- 5.5 table . . . 25

6 **29**
- 6.1 . . . 29
- 6.2 plugin . . . 29
- 6.3 DQC . . . 31

etlsdk plugin

CHAPTER 1

1.1 plugin

inputs, outputs, args

1.2

oss itemshiveplugin

plugin

- dev120LDAP10.10.14.120

```
ssh username@10.10.14.120
```

- demodemodemo__init__.py

```
  mkdir demo
  cd demo/
  mkdir demo
  touch demo/__init__.py
```

- plugin.py

```
touch demo/plugin.py
```

- plugin.pyoss itemshiveplugindemo plugindemo

```
cp /usr/local/lib/python3.5/dist-packages/etlsdk/demo_plugins/oss2hive.py demo/plugin.
↪py
```

```
from etlsdk.lib.datasources.datasource_factory import DatasourceFactory
from pprint import pprint as print

```

(continues on next page)

(continued from previous page)

```
class Oss2Hive(object):

    def run_plugin(self, inputs, outputs, args):
        """
        :param inputs: dayu_iddayu_full_name dataframe
            inputs = {
                'OssItem': {'dayu_full_name': 'OSS_default:amazoncrawl:etlsdk_ut/
↪oss2hive_demo/',
                   'dayu_id': 2569,
                   'df': DataFrame[key: string, json: string],
                   'name': 'etlsdk_ut/oss2hive_demo/',
                   'partition': [{'tdate': '2019-01-01'}],
                   'table': <etlsdk.lib.handlers.table_handler.Table object>,
                   'type': 'oss'}
            }
            input --input OssItem:name=OSS_default:amazoncrawl:etlsdk_ut/oss2hive_
↪demo/
        :param outputs: inputs, df
            outputs = {
                'RawTable': {'dayu_full_name': 'Hive:etlsdk_test:raw_table',
                    'dayu_id': 1730,
                    'name': 'raw_table',
                    'partition': {'tdate': '2019-01-01'},
                    'table': <etlsdk.lib.handlers.table_handler.Table object>,
                    'type': 'hive'}
            }
            output RawTable:name=Hive:etlsdk_test:raw_table
        :param args: args
            args = {
              {'partition_date': '2019-01-01 00:00:00'}
            }
        :return:
        """

        print('==='*10+''+'==='*10)

        print('** inputs:')
        print(inputs) # inputinputs

        print('** outputs:')
        print(outputs) # outputoutputs

        print('** args:')
        print(args)# argsargs
        print('==='*25)

        dqc = {"rules": [{"rule": "count", "restrict": 1}]}
        oss_df = inputs['OssItem']['df']
        DatasourceFactory.write_dataframe(oss_df, outputs['RawTable'], dqc=dqc)
```

plugin

- plugin

```
python3 -m etlsdk.main demo.plugin.Oss2Hive.run_plugin --input OssItem:name=OSS_
↪default:amazoncrawl:etlsdk_ut/oss2hive_demo/  --output RawTable:name=Hive:etlsdk_
↪test:raw_table --partition 2019-01-01
```

- hue
  1. https://hue.aidigger.com/hue/
  2. `select * from etlsdk_test.raw_table where tdate="2019-01-01"`
  3.

etlsdk

CHAPTER 2

Demo

2.1 1. Demo of plugin on pony

2.1.1 demo

2.1.2 demo

pony

2.2 2. Demo of plugin terminal

dev120

2.2.1

step1:

```
python3 -m etlsdk.main etlsdk.demo_plugins.raw2parsed.Raw2ParsedPlugin.run   \
--input RawTable:name=Hive:etlsdk_test:raw_table \
--output ParsedTable:name=Hive:etlsdk_test:parsed_table \
--args extractor:tests.test_plugins.test_raw2parsed.raw2parsed_extractor.
↪AlmostHumanExtractor \
--partition "2019-01-01 00:00:00" \
--dependency tests:hdfs://user/hadoop/etlsdk_ut/tests.zip
```

step2:

```
1.  https://hue.aidigger.com/hue/
2.  'select * from etlsdk_test.parsed_table where tdate="2019-01-01"'
3.
```

2.2.2

```
1. 120
2.
    /usr/hdp/current/kafka-broker/bin/kafka-console-consumer.sh --bootstrap-
↪server=emr2-header-1.ipa.aidigger.com:6667 --topic etlsdk_test_etlsession_stream_
↪write --offset latest --partition 0
3. , 120
    python3 -m etlsdk.main etlsdk.demo_plugins.kafka.KafkaPlugin.run   \
      --input KafkaIN:name=Kafka::etlsdk_test_read_dataframe \
      --output KafkaOUT:name=Kafka::etlsdk_test_etlsession_stream_write \
      --partition "2019-01-01 00:00:00" \
      --isstreaming True
4. producer
  4.1
    /usr/hdp/current/kafka-broker/bin/kafka-console-producer.sh --broker-list emr2-
↪header-1.ipa.aidigger.com:6667  --topic etlsdk_test_read_dataframe

  4.2
    {"articleid":"articleid_test", "read_number":10, "tdate":"2019-01-02", "category":
↪'dingxiang', "content":"test" }

5. 2consumerconsume
```

CHAPTER 3

etlsdk

```
python3 -m etlsdk.main <method_canonical> [etlsdk_args] [plugin_args]
```

: dev120

: `python3 -m etlsdk.main --help`

3.1 method_canonical

plugin data_pipeline.plugins.WeixinProcessPlugin.joinimage ...

:

3.2 etlsdk_args

etlsdk

	/	
partition		**'%Y-%m-%d''%Y-%m-%d %H:%M:%S'** . **args.get("partition_date")ponypartition**
isstreaming		**streamingdefault: False**
driver_memory		default: 1g
executor_num		default: 1
executor_memory		default: 2g
spark_conf		default: None, `spark_conf_key:spark_conf_value` `spark.gevent.maxsize` gevent :20 dev120:100 spark.master local[2],cluster dev120local[2]. **cluster''spark.master'' "k8s://https://api.k8s.aipp.io:6443"**
dependency		default: None hdfspypi `module_name;module_url` module_name :eigen_config:https://pypi.aidigger.com/packages/EigenConfigLibrary-0.0.1.tar.gz
config		default: None json file:///config.jsonlocalhdfs:///config.jsonremote

3.3 plugin_args: inputs / outputs

/`$input_name:name=$dayu_full_name` or `$input_name:id=$dayu_iddayu_full_namedayu_id`

`input_name` plugin`inputs` key.

`dayu_id`table`ID` `dayu_full_name`table table `dayu_full_name`="Hive:default:douban_join_baike_movie_3"`dayu_id`=2529

outputs

3.4 plugin_args: args

`--args args_key:args_key_value`

*''args_key_value''(json, datetime)***''''''**

''args_key_value''dictconfig

:

1. `KafkaMaxFetchBytes` max.partition.fetch.bytes Kafka Topic: 20971520 : string
2. `KafkaMaxRequestBytes` max.request.size, Kafka Topic: 15728640 : string
3. `KafkaMaxOffsetsPerTrigger` maxOffsetsPerTrigger , Kafka TopicMicro-Batch: 50 string

4. `KafkaCheckpoint` checkpointLocation ,Kafka checkpointLocation `/tmp/spark/checkpoint/kafka/` : table.name+uuid.uuid1().hex dict exmaple: {“KafkaTopic1”: “KafkaTopic1_checkpointLocation_file_20201001”, “KafkaTopic2”: “KafkaTopic2_checkpointLocation_file_20201201”}
5. `offset_date` offset_date, Kafka Topic: None (latest) : datetime string exmaple: “2020-10-01 00:00:00”
6. `kafka.group.id` kafka.group.id string

CHAPTER 4

```
# run demo by json file:
python3 -m etlsdk.main data_pipeline.plugins.WeixinProcessPlugin.joinimage \
    --config hdfs://user/hadoop/etlsdk_ut/raw2parsed_demo.json

# run demo by terminal:
python3 -m etlsdk.main raw2parsed   \
    --input RawTable:name=Hive:etlsdk_test:raw_table \
    --output ParsedTable:id=1740 \
    --args extractor:tests.test_plugins.test_raw2parsed.raw2parsed_extractor.
↪AlmostHumanExtractor \
    --partition "2019-01-01 00:00:00" \
    --dependency tests:hdfs://user/hadoop/etlsdk_ut/tests.zip
```

`--input --output input,output input,output`

4.1 config

```
{
  "args": {
     "isstreaming": "False",
     "extractor": "tests.test_plugins.test_raw2parsed.raw2parsed_extractor.
↪AlmostHumanExtractor",
     "spark_conf": {
       "config": {
        "executor_num": "1",
        "advanced":{"spark.partition.num": "100"}
       },
       "dependency": {
           "tests": "hdfs://user/hadoop/etlsdk_ut/tests.zip"
       }
     }
  },
```

(continues on next page)

(continued from previous page)

```
  "input": [{"name": "RawTable", "type": "hive", "dayu_full_name": "Hive:etlsdk_
↪test:raw_table", "partition":[{"partition_date":"2019-01-01"}]}],
  "output": [{"name": "ParsedTable", "type": "hive", "dayu_full_name": "Hive:etlsdk_
↪test:parsed_table", "partition":{"partition_date":"2019-01-01"}}]
}
```

4.2

: etlsdk

4.2.1

```
d=start_date
while [ "$d" != end_date ]; do
  echo $d "start"
  <command>
  d=$(date -I -d "$d + 1 day")
done
```

4.2.2

```
d=2019-01-01
while [ "$d" != 2019-01-05 ]; do
  echo $d "start"
  python3 -m etlsdk.main etlsdk.demo_plugins.raw2parsed.Raw2ParsedPlugin.run   \
    --input RawTable:name=Hive:etlsdk_test:raw_table \
    --output ParsedTable:name=Hive:etlsdk_test:parsed_table \
    --args extractor:tests.test_plugins.test_raw2parsed.raw2parsed_extractor.
↪AlmostHumanExtractor \
    --partition $d \
    --dependency tests:hdfs://user/hadoop/etlsdk_ut/tests.zip
  d=$(date -I -d "$d + 1 day")
done
```

CHAPTER 5

etlsdk API

5.1

class etlsdk.lib.datasources.datasource_factory.**DatasourceFactory**

Datasourcedatasource: *OSSDataSource*, *HiveDataSource*, *ESDataSource*, *KafkaDataSource*

classmethod read_dataframe(*input*, ***kwargs*)

dataframe

Parameters

- **input** (*dict*) –

  key dayu_full_name (str) table‘‘. Examplehiveetlsdk_test.simple_table ‘‘ Hive:etlsdk_test:simple_table

  key partition(list of Partition, optional) tabledefault []: Partition: {$partition_column_name:$partition_column_value}

- ****kwargs** – read_dataframe

Example:

```
from etlsdk.lib.datasources.datasource_factory import DatasourceFactory

input = {
    "dayu_full_name": "Hive:etlsdk_test:parsed_table",
    "partition": [{"tdate": "2019-01-01"},{"tdate":"2019-01-02"}]
}
df = DatasourceFactory.read_dataframe(input)
```

Returns pyspark.sql.DataFrame

Raises

- DayuClientError –
- NotImplementedError –

classmethod write_dataframe(*df*, *output*, ***kwargs*)

dataframeoutput

Parameters

- **df** (*DataFrame*) – pyspark.sql.DataFrame
- **output** (*dict*) –

  key dayu_full_name (str) table‘‘. Examplehiveetlsdk_test.simple_table ‘‘ Hive:etlsdk_test:simple_table

  key partition (dict) table value type: {$partition_column_name:$partition_column_value}.
- ****kwargs** – write_dataframe

Example:

```
from etlsdk.lib.datasources.datasource_factory import DatasourceFactory

output = {
    "dayu_full_name": "Hive:parsed:dcrawl_parsed_weixin",
    "partition": [{"tdate": "2019-01-01"}]
    }
dqc_config = {"rules": [{'rule': 'count', 'restrict': 10}]}
df = DatasourceFactory.write_dataframe(output, dqc=dqc_config)
```

Raises

- DayuClientError –
- NotImplementedError –

5.2 api

5.2.1 HIVE

class etlsdk.lib.datasources.hive_datasource.**HiveDataSource**

read_dataframe(*input*)

dataframe

Parameters **input** (*dict*) – *TableHandler.normalize_table*

key table (Table) *Table*

key partition (list of partition, optional) tabledefault []: Partition: {$partition_column_name:$partition_column_value}

Example:

```
from etlsdk.lib.datasources.hive_datasource import HiveDataSource
from etlsdk.lib.handlers.table_handler import TableHandler

input = {
    "dayu_full_name": "Hive:etlsdk_test:parsed_table",
    "partition": [{"tdate": "2019-01-01"},{"tdate":"2019-01-02"}]
```

(continues on next page)

(continued from previous page)

```
}
hive_datasource = HiveDataSource()
df = hive_datasource.read_dataframe(TableHandler.normalize_table(input))
```

Returns pyspark.sql.DataFrame

Raises

- DayuClientError –
- NotImplementedError –

write_dataframe(*df*, *output*, *dqc=None*, *enable_dynamici=False*)

dataframeoutput

Parameters

- **df** (*DataFrame*) – pyspark.sql.DataFrame
- **output** (*dict*) – *TableHandler.normalize_table*

  key table (Table) *Table*

  key partition (dict) table value type: {$partition_column_name:$partition_column_value}.
- **dqc** (*dict, optional*) – dqc. default: {"rules": []} .
- **enable_dynamici** (*boolean, optional*) – default:False. hiveoutputpartitionoutput['partition'][]hive DynamicPartitions: https://cwiki.apache.org/Hive/dynamicpartitions

Example:

```
from etlsdk.lib.datasources.hive_datasource import HiveDataSource
from etlsdk.lib.handlers.table_handler import TableHandler

output = {
    "dayu_full_name": "eigen_rds_test:unittest:etlsdk_test",
    "partition": {"tdate": "2019-01-01"}
    }
dqc_config = {"rules": [{'rule': 'count', 'restrict': 10}]}

hive_datasource = HiveDataSource()
df = hive_datasource.write_dataframe(TableHandler.normalize_table(output),␣
↪dqc=dqc_config)
```

Raises NotImplementedError –

5.2.2 OSS

class etlsdk.lib.datasources.oss_datasource.**OSSDataSource**

read_dataframe(*input*, ***kwargs*)

dataframe oss textreturn Dataframe[key: string, json: string] oss jsonreturn Dataframe Dataframe schemaoss schema

Parameters **input** (*dict*) – *TableHandler.normalize_table*

key table (Table) *Table*

key partition (list of partition, optional) tabledefault []: Partition: {$partition_column_name:$partition_column_value}

Example:

```
from etlsdk.lib.datasources.oss_datasource import OSSDataSource
from etlsdk.lib.handlers.table_handler import TableHandler

input = {
    "dayu_full_name": "OSS_default:amazoncrawl:etlsdk_ut/test_ossutil/",
    "partition": [{"tdate": "2019-01-01"}]
}
oss_datasource = OSSDataSource()
df = oss_datasource.read_dataframe(TableHandler.normalize_table(input))
```

Returns pyspark.sql.DataFrame

Raises NotImplementedError –

write_dataframe(*df*, *output*, *unique_column='item_id'*, *dqc=None*)

dataframeoutput oss text Dataframe $unique_column: string, content: string oss json Dataframeschema

Parameters

- **df** (*DataFrame*) – pyspark.sql.DataFrame
- **output** (*dict*) – *TableHandler.normalize_table*

  key table (Table) *Table*

  key partition (dict) oss value type: {$partition_column_name:$partition_column_value}.
- **unique_column** (*str*) – dataframeoss keynamedefault: item_id
- **dqc** (*dict, optional*) – dqc. default: {"rules": []} .

Example:

```
from etlsdk.lib.datasources.oss_datasource import OSSDataSource
from etlsdk.lib.handlers.table_handler import TableHandler

output = {
    "dayu_full_name": "OSS_default:amazoncrawl:etlsdk_ut/test_ossutil/",
    "partition":{"tdate":"2019-01-01"}
    }
dqc_config = {"rules": [{'rule': 'count', 'restrict': 10}]}

oss_datasource = OSSDataSource()
df = oss_datasource.write_dataframe(df, TableHandler.normalize_table(output),
↪ unique_column="articleid", dqc=dqc_config)
```

5.2.3 ES

class etlsdk.lib.datasources.es_datasource.**ESDataSource**

write_dataframe(*df*, *output*, *unique_column='id'*, *json_cols=None*, *dqc=None*)
dataframeoutput

Parameters

- **df** (*DataFrame*) – pyspark.sql.DataFrame
- **output** (*dict*) – *TableHandler.normalize_table*

  key table (Table) *Table*
- **unique_column** (*str, optional*) – dataframees indexdefault: id.
- **json_cols** (*dict, optionla*) – dataframees indexlistdict
- **dqc** (*dict, optional*) – dqc. default: {"rules": []} .

Example:

```
from etlsdk.lib.datasources.es_datasource import ESDataSource
from etlsdk.lib.handlers.table_handler import TableHandler

output = {
    "dayu_full_name": "ES::etlsdk_test",
    }
dqc_config = {"rules": [{'rule': 'count', 'restrict': 10}]}

es_datasource = ESDataSource()
df = es_datasource.write_dataframe(TableHandler.normalize_table(output),␣
↪dqc=dqc_config)
```

5.2.4 KAFKA

class etlsdk.lib.datasources.kafka_datasource.**KafkaDataSource**

load_value(*df: <sphinx.ext.autodoc.importer._MockObject object at 0x7f04b6d20320>*, *table: etlsdk.lib.handlers.table_handler.Table*) → <sphinx.ext.autodoc.importer._MockObject object at 0x7f04b6d20320>

Parameters

- **df** – DataFrame
- **table** – Table

Returns

read_dataframe(*input*) → <sphinx.ext.autodoc.importer._MockObject object at 0x7f04b6d20320>
dataframe

Parameters **input** (*dict*) – *TableHandler.normalize_table*

key table (Table) *Table*

Example:

```
from etlsdk.lib.datasources.kafka_datasource import KafkaDataSource
from etlsdk.lib.handlers.table_handler import TableHandler

input = {
    "dayu_full_name": "Kafka::etlsdk_test_read_dataframe",
}
kafka_datasource = KafkaDataSource()
df = kafka_datasource.read_dataframe(TableHandler.normalize_table(input))
```

Returns pyspark.sql.DataFrame

Raises:

write_dataframe(*df: <sphinx.ext.autodoc.importer._MockObject object at 0x7f04b6d20320>*, *output*)

dataframeoutput

Parameters

- **df** (*DataFrame*) – pyspark.sql.DataFrame
- **output** (*dict*) – *TableHandler.normalize_table*

  key table (Table) *Table*

Example:

```
from etlsdk.lib.datasources.kafka_datasource import KafkaDataSource

output = {
    "dayu_full_name": "Kafka::etlsdk_test_etlsession_stream_write",
    }

kafka_datasource = KafkaDataSource()
df = kafka_datasource.write_dataframe(output, dqc=dqc_config)
```

Raises

- DayuClientError –
- RuntimeError – etlsession.isstreamingFalsedf

5.2.5 MYSQL

class etlsdk.lib.datasources.mysql_datasource.**MYSQLDataSource**

read_dataframe(*input*, *sql=None*)

dataframe

Parameters

- **input** (*dict*) – *TableHandler.normalize_table*

  key table (Table) *Table*

- **sql** (*str, optional*) – mysqlsql

Examples

mysql :

```
from etlsdk.lib.datasources.mysql_datasource import MYSQLDataSource
from etlsdk.lib.handlers.table_handler import TableHandler

table_info = {
    "type": "mysql",
    "name": table_name,
    "full_name": "",
    "database": db_name,
    "columns": [],
    "partitions": [],
    "storage": {
        "type": "mysql",
        "settings": {
            'host': mysql_host,
            'port': mysql_port, # int
            'user': mysql_username,
            "database": db_name,
            'password': mysql_password,
        }
    }
}
# table_info["primary_key"] = {'name': primary_key_column_name, 'type':␣
↪primary_column_type}
table_info["primary_key"] = {'name': "id", 'type': "int"}
mysql_input = {}
mysql_input["table"] = Table(table_info)

mysql_datasource = MYSQLDataSource()
df = mysql_datasource.read_dataframe(mysql_input)
```

mysql :

```
from etlsdk.lib.datasources.mysql_datasource import MYSQLDataSource
from etlsdk.lib.handlers.table_handler import TableHandler

mysql_datasource = MYSQLDataSource()
df = mysql_datasource.read_dataframe(TableHandler.normalize_table({"dayu_
↪full_name":"eigen_rds_test:unittest:etlsdk_test"}))
```

Returns pyspark.sql.DataFrame

Raises NotImplementedError –

Notes

mysql2G, mysqlprimary key. spark2G: https://issues.apache.org/jira/browse/SPARK-6235 sparkmysql: https://spark.apache.org/docs/latest/sql-data-sources-jdbc.html

write_dataframe(*df*, *output*, *dqc=None*, *auth_table_enable=True*)

dataframeoutput

Parameters

- **df** (*DataFrame*) – pyspark.sql.DataFrame
- **output** (*dict*) – *TableHandler.normalize_table*

  key table (Table) *Table*
- **dqc** (*dict, optional*) – dqc. default: {"rules": []} .

Examples

mysql :

```
from etlsdk.lib.datasources.mysql_datasource import MYSQLDataSource
from etlsdk.lib.handlers.table_handler import Table

table_info = {
    "type": "mysql",
    "name": table_name,
    "full_name": "",
    "database": db_name,
    "columns": [],
    "partitions": [],
    "storage": {
        "type": "mysql",
        "settings": {
            'host': mysql_host,
            'port': mysql_port, # int
            'user': mysql_username,
            "database": db_name,
            'password': mysql_password,
        }
    }
}
# table_info["primary_key"] = {'name': primary_key_column_name, 'type':␣
↪primary_column_type}
table_info["primary_key"] = {'name': "id", 'type': "int"}
mysql_output = {}
mysql_output["table"] = Table(table_info)

mysql_datasource = MYSQLDataSource()
mysql_datasource.write_dataframe(mysql_output, auth_table_enable=False) #␣
↪mysql``auth_table_enableFalse
```

mysql :

```
from etlsdk.lib.datasources.mysql_datasource import MYSQLDataSource
from etlsdk.lib.handlers.table_handler import TableHandler

mysql_output = {}
mysql_output["table"] = TableHandler.get_table({"dayu_full_name": "eigen_rds_
↪test:unittest:etlsdk_test"})

mysql_datasource = MYSQLDataSource()
mysql_datasource.write_dataframe(df, mysql_output)
```

5.3

class etlsdk.lib.session.**ETLSession**

classmethod get_instance()

ETLSessionpluginspark_session,inputs,outputs,args

Example:

```
from etlsdk.lib.session import ETLSession

etl_session = ETLSession.get_instance()
spark = etl_session.spark_session
df = spark.table('default.dcrawl_parsed_cars')

inputs = etl_session.inputs
>>print(etlsession.inputs)
{'OssItem': {'dayu_full_name': 'OSS_default:amazoncrawl:etlsdk_ut/oss2hive_
↪demo/',
            'dayu_id': 2569,
            'df': DataFrame[key: string, json: string],
            'name': 'etlsdk_ut/oss2hive_demo/',
            'partition': [{'tdate': '2019-01-01'}],
            'table': <etlsdk.lib.handlers.table_handler.Table object>,
            'type': 'oss'}}

>>print(etlsession.outputs)
{'RawTable': {'dayu_full_name': 'Hive:etlsdk_test:raw_table',
         'dayu_id': 1730,
         'name': 'raw_table',
         'partition': {'tdate': '2019-01-01'},
         'table': <etlsdk.lib.handlers.table_handler.Table object>,
         'type': 'hive'}}
```

5.4 UT

class etlsdk.tools.ut_utils.**ETLSessionUT**

pluginut

Example:

```
from etlsdk.tools.ut_utils import ETLSessionUT
from data_pipeline.plugins.oss2hive import OSS2HivePlugin

mock_datas = [{"key":str(num), "json":"json %d"%num, "tdate":"2019-01-01"} for␣
↪num in range(10)]
inputs = [
    {
        "name": "RawTable",
        "dayu_full_name":"Hive:etlsdk_test:raw_table",
        "partition":[{"partition_date":"2019-01-01"}],
        "mock_datas":mock_datas
        }
    ]
```

(continues on next page)

(continued from previous page)

```
outputs = [
    {
        "name": "ParsedTable",
        "dayu_full_name": "HIVE:etlsdk_test:parsed_table",
        "partition":{"partition_date":"2019-01-01"}
        }
    ]

args = {'test_args': 'test_args_value'}
etlsession = ETLSessionUT.create_etlsession(inputs, outputs, args)

>>print(etlsession.inputs)
{'RawTable': {'dayu_full_name': 'Hive:etlsdk_test:raw_table',
              'dayu_id': 1730,
              'df': DataFrame[key: string, json: string, tdate: string],
              'name': 'raw_table',
              'partition': [{'partition_date': '2019-01-01'}],
              'table': <etlsdk.lib.handlers.table_handler.Table object at␣
↪0x7f3100910a90>,
              'type': 'hive'}}
>>print(etlsession.outputs)
{'ParsedTable': {'dayu_full_name': 'Hive:etlsdk_test:parsed_table',
                 'dayu_id': 1740,
                 'name': 'parsed_table',
                 'partition': {'partition_date': '2019-01-01'},
                 'table': <etlsdk.lib.handlers.table_handler.Table object at␣
↪0x7f310090cf98>,
                 'type': 'hive'}}
>>print(etlsession.args)
{'test_args': 'test_args_value'}

oss2hive = OSS2HivePlugin()
oss2hive.run(etlsession.inputs, etlsession.outputs, etlsession.args)
writeout_results = etlsession.get_writeout_results()
```

classmethod create_etlsession(*inputs*, *outputs*, *args*)

ETLSession

Parameters

- **inputs** (*dict*) –

  key dayu_full_name (str) table‘‘. Examplehiveetlsdk_test.simple_table ‘‘ Hive:etlsdk_test:simple_table

  key partition (list of Partition, optional) tabledefault []: Partition: {$partition_column_name:$partition_column_value}

  key mock_datas (list of Row.asDict()) mock input df.

- **outputs** (*dict*) –

  key dayu_full_name (str) table‘‘. Examplehiveetlsdk_test.simple_table ‘‘ Hive:etlsdk_test:simple_table

  key partition (dict) table value type: {$partition_column_name:$partition_column_value}.

- **args** (*dict*) –

key isstreaming (str) ETLSession default:"False" ()

Returns *ETLSession*

classmethod get_sparksession()

Returns pyspark.sql.SparkSession [('spark.driver.memory', '2g'), ('spark.executor.cores', '2'), ('spark.executor.instances', '1'), ('spark.executor.memory', '2g')], local

classmethod get_writeout_results()
plugindf

Returns $df_datas}

Return type {$dayu_full_name

5.5 table

class etlsdk.lib.handlers.table_handler.**TableHandler**

classmethod get_table(*input*)
Table: MYSQL, OSS, HIVE, ES, KAFKA

Parameters **input** (*dict*) –

key dayu_full_name (str) '''' Example: hiveetlsdk_test.simple_table '''',Hive:etlsdk_test:simple_table

Returns *Table*

Raises

- DayuClientError –
- DayuServerError –

classmethod hybrid_to_hive(*table*)
Hybrid TableHive Table :param table: Table :return: Table

classmethod hybrid_to_kafka(*table*)
Hybrid TableKafka Table :param table: Table :return: Table

classmethod normalize_table(*table_info*)
:OSS, HIVE, ES, KAFKA

Parameters **table_info** (*dict*) –

key dayu_full_name (str) '''' Example: hiveetlsdk_test.simple_table '''',Hive:etlsdk_test:simple_table

key partition (dict, optional) table value type: {$partition_column_name:$partition_column_value}.

Example:

```
from etlsdk.lib.handlers.table_handler import TableHandler
output = {
    "dayu_full_name": "Hive:etlsdk_test:raw_table",
    "partition": {"tdate": "2019-01-01"}
    }
>> print(TableHandler.normalize_table(output))
```

(continues on next page)

(continued from previous page)

```
{
     'dayu_full_name': 'Hive:etlsdk_test:raw_table',
     'dayu_id': 1730,
     'name': 'raw_table',
     'partition': {'tdate': '2019-01-01'},
     'table': <etlsdk.lib.handlers.table_handler.Table object at␣
↪0x7f4fb0a7d9e8>,
     'type': 'hive'
     }
```

Returns normalize table.

Raises

- DayuClientError –
- DayuServerError –

class etlsdk.lib.handlers.table_handler.**Table**(*table_info*)

all_columns

return::

: list of column.
hive, mysql, oss, kafka. (esextras)
column: {"name":$column_name, "type":$column_type, "comment":$comment_type}

database_name
return

db_table_name
return db_name.table_name

extras

return::

OSS extras prefix_pattern, serializer, is_file
ES extras doc_type, source, mapping
HBASE extras prefix_pattern, default_column_family

full_name
return ""

id
return id

name
return

partition
return – list of partition. partition: {"name":$partition_name, "type":$partition_type, "comment":$comment_type}

primary_key
return . {"name":$primary_column_name, "type":$column_type}

schema

return::

: list of column.
hive, mysql, oss, kafka. (esextras)
column: {“name”:$column_name, “type”:$column_type, “comment”:$comment_type}

storage_settings
return

storage_type
return

CHAPTER 6

6.1

isstreaming: boolean type default: `False`

```
True: ``
False: ``
*: *
```

6.1.1 1. OSS

6.1.2 2. HIVE

6.1.3 3. ES

6.1.4 4. MYSQL

6.1.5 5. KAFKA

6.1.6 6. HBASE

6.2 plugin

configargsinput/outputpartition

6.2.1

```
python3 -m etlsdk.main <method_canonical> --config <config_file>
```

example:

```
python3 -m etlsdk.main etlsdk.demo_plugins.raw2parsed.Raw2ParsedPlugin.run  --config␣
↪hdfs://user/hadoop/etlsdk_ut/raw2parsed_demo.json
```

6.2.2 config

json, keys: (input, output, args)

:key input: type: list of input_element.

```
input_element: type:dict.  keys (`name`, `dayu_id`, `partition`) or (`name`, `dayu_
↪full_name`, `partition`)
          :key `name`: input value type: string
          :key `dayu_id`: table`id`. value type: int
          :key `dayu_full_name`: table``.  value type: str.
          :key `partition`: table  value type: list of {$partition_column_name:
↪$partition_column_value}.
              partitionpartition_date  partition_range
              partition_column: keys (`partition_date`), (`partition_range`)
                :key `partition_date`: date string  '%Y-%m-%d %H:%M:%S''%Y-%m-%d'
                :key `partition_range`: [$date_start, $date_end] partition_range.␣
↪date string  '%Y-%m-%d'
```

:key output: type: list of output_element.

```
output_element: type:dict. keys (`name`, `dayu_id`, `partition`) or (`name`, `dayu_
↪full_name`, `partition`)
          :key `name`: output value type: string
          :key `dayu_id`: table`id`. value type: int
          :key `dayu_full_name`: table``.  value type: str.
          :key `partition`: table  value type: {$partition_column_name:$partition_
↪column_value}.
              partitionpartition_range
              partition_column: keys (`partition_date`)
                :key `partition_date`: date string  '%Y-%m-%d %H:%M:%S'
```

:key args: type: dict.

```
    :key `spark_conf`: spark. type: dict.
          :key `config`: sparksparkpy files. type: dict.
                :key `executor_num`: spark.executor.instances. (e.g. 1, 4). default:␣
↪`1`. type: string.
                :key `executor_memory`: spark.executor.memory. (e.g. 3g, 2.5g).␣
↪default: `2g`. type: string.
                :key `advanced`: spark session application. format: {$spark_conf_key:
↪$spark_conf_value}:https://spark.apache.org/docs/2.3.1/configuration.html
↪#application-properties.(e.g. {"spark.executor.cores":"2"} ) : spark.driver.* . ␣
↪type: dict.
          :key `dependency`: sparkPythonPYTHONPATHtype:dict. format: {$module_import_
↪name: $module_url}.  module_url: pypihdfs. hdfsmodule setupzip. (e.g. {"data_
↪pipeline": "hdfs://user/hadoop/ETL/data_pipline.zip"})
```

(continues on next page)

(continued from previous page)

```
    :key `isstreaming`: spark default: `false` type: string.
    :key $user_args:
```

example:

```
{
    "args": {
        "isstreaming": "False",
        "extractor": "tests.test_plugins.test_raw2parsed.raw2parsed_extractor.
↪AlmostHumanExtractor",
        "spark_conf": {
            "config": {
                "executor_num": "1",
                "advanced": {
                    "spark.partition.num": "100"
                }
            },
            "dependency": {
                "tests": "hdfs://user/hadoop/etlsdk_ut/tests.zip"
            }
        }
    },
    "input": [
        {
            "name": "RawTable",
            "dayu_full_name": "Hive:etlsdk_test:raw_table",
            "partition": [
                {
                    "partition_date": "2019-01-01"
                }
            ]
        }
    ],
    "output": [
        {
            "name": "ParsedTable",
            "dayu_full_name": "Hive:etlsdk_test:parsed_table",
            "partition": {
                "partition_date": "2019-01-01"
            }
        }
    ]
}
```

6.3 DQC

6.3.1 dqc

```
{
    "rules": [
        {"rule": "count", "restrict": 10},
        {"column": "column1", "rule": "not_empty", "restrict": 10},
        {"column": "column2", "rule": "right_type", "type": "int", "restrict": 10},
```

(continues on next page)

(continued from previous page)

```
        {
            "column": "column3",
            "rules": [
                {"rule": "right_type", "type": "int", "restrict": 10},
                {"rule": "not_empty", "restrict": 10}
            ]
        }
    ],
    "not_report_time": [
        {"weekday":"Sat", "hour_start":0, "hour_end":23},
        {"weekday":"Sun", "hour_start":0, "hour_end":23}
    ]
}
```

rules: list. Dqc

not_report_time: list.

6.3.2 DQC Rule

Rule dict

- rule: stringRulecount, not_empty, right_type.
- restrict: int0
- column: stringRulecount

6.3.3 DQC Rule Type

6.3.4 DQC not_report_time

not_report_timedqcitemdict

- weekday, string, (Mon,Tue,Wed,Thu,Fri,Sat,Sun)
- hour_start, int, , default0
- hour_end, int, , default24

: [0:23)23 {“weekday”:”Sun”, “hour_start”:0, “hour_end”:23}

6.3.5

- DatasourceFactory.write_dataframe()dqc

oss2hive plugin 1

```
from etlsdk.lib.session import ETLSession
from etlsdk.lib.datasources.datasource_factory import DatasourceFactory

class Oss2HivePlugin(object):
```

(continues on next page)

(continued from previous page)

```
    dqc_configs = {
        "rules": [
            {"rule": "count", "restrict": 1},
        ],
        "not_report_time": [
            {"weekday":"Sat"},
            {"weekday":"Sun"}
        ]
    }

    def run(self, inputs, outputs, args):
        """
        ossBucket/xxItem/[tdate]/[hour]hiverawtable_db.rawtable_name
        :param inputs: OssItem:oss:Bucket.xxItem/, eg: OssItem:oss:amazoncrawl.
↪WeTaoContentDetailItem
        :param outputs: RawTable:hive:rawtable_db.rawtable_name, eg:␣
↪RawTable:hive:default.dcrawl_raw_wetao_content_summary
        :param args:
        :return:
        """
        oss_df = inputs['OssItem']['df']
        DatasourceFactory.write_dataframe(oss_df, outputs['RawTable'], dqc=self.dqc_
↪configs)
```

Index

A

all_columns (etlsdk.lib.handlers.table_handler.Table attribute), 26

C

create_etlsession() (etlsdk.tools.ut_utils.ETLSessionUT class method), 24

D

database_name (etlsdk.lib.handlers.table_handler.Table attribute), 26
DatasourceFactory (class in etlsdk.lib.datasources.datasource_factory), 15
db_table_name (etlsdk.lib.handlers.table_handler.Table attribute), 26

E

ESDataSource (class in etlsdk.lib.datasources.es_datasource), 19
ETLSession (class in etlsdk.lib.session), 23
ETLSessionUT (class in etlsdk.tools.ut_utils), 23
extras (etlsdk.lib.handlers.table_handler.Table attribute), 26

F

full_name (etlsdk.lib.handlers.table_handler.Table attribute), 26

G

get_instance() (etlsdk.lib.session.ETLSession class method), 23
get_sparksession() (etlsdk.tools.ut_utils.ETLSessionUT class method), 25
get_table() (etlsdk.lib.handlers.table_handler.TableHandler class method), 25
get_writeout_results() (etlsdk.tools.ut_utils.ETLSessionUT class method), 25

H

HiveDataSource (class in etlsdk.lib.datasources.hive_datasource), 16
hybrid_to_hive() (etlsdk.lib.handlers.table_handler.TableHandler class method), 25
hybrid_to_kafka() (etlsdk.lib.handlers.table_handler.TableHandler class method), 25

I

id (etlsdk.lib.handlers.table_handler.Table attribute), 26

K

KafkaDataSource (class in etlsdk.lib.datasources.kafka_datasource), 19

L

load_value() (etlsdk.lib.datasources.kafka_datasource.KafkaDataSource method), 19

M

MYSQLDataSource (class in etlsdk.lib.datasources.mysql_datasource), 20

N

name (etlsdk.lib.handlers.table_handler.Table attribute), 26
normalize_table() (etlsdk.lib.handlers.table_handler.TableHandler class method), 25

O

OSSDataSource (class in etlsdk.lib.datasources.oss_datasource), 17

P

partition (etlsdk.lib.handlers.table_handler.Table attribute), 26

primary_key (etlsdk.lib.handlers.table_handler.Table attribute), 26

R

read_dataframe() (etlsdk.lib.datasources.datasource_factory.DatasourceFactory class method), 15
read_dataframe() (etlsdk.lib.datasources.hive_datasource.HiveDataSource method), 16
read_dataframe() (etlsdk.lib.datasources.kafka_datasource.KafkaDataSource method), 19
read_dataframe() (etlsdk.lib.datasources.mysql_datasource.MYSQLDataSource method), 20
read_dataframe() (etlsdk.lib.datasources.oss_datasource.OSSDataSource method), 17

S

schema (etlsdk.lib.handlers.table_handler.Table attribute), 26
storage_settings (etlsdk.lib.handlers.table_handler.Table attribute), 27
storage_type (etlsdk.lib.handlers.table_handler.Table attribute), 27

T

Table (class in etlsdk.lib.handlers.table_handler), 26
TableHandler (class in etlsdk.lib.handlers.table_handler), 25

W

write_dataframe() (etlsdk.lib.datasources.datasource_factory.DatasourceFactory class method), 16
write_dataframe() (etlsdk.lib.datasources.es_datasource.ESDataSource method), 19
write_dataframe() (etlsdk.lib.datasources.hive_datasource.HiveDataSource method), 17
write_dataframe() (etlsdk.lib.datasources.kafka_datasource.KafkaDataSource method), 20
write_dataframe() (etlsdk.lib.datasources.mysql_datasource.MYSQLDataSource method), 21
write_dataframe() (etlsdk.lib.datasources.oss_datasource.OSSDataSource method), 18