# **DKB Documentation**

**DKB** team

# **CONTENTS:**

1	pyDKB package	1
	1.1 Quickstart guide	1
	1.2 Subpackages	3
	Stages           2.1         Stage 055	<b>25</b> 25
3	Indices and tables	31
Рy	ython Module Index	33
In	dex	35

**CHAPTER** 

ONE

# PyDKB package

Common library for Data Knowledge Base Dataflow stages development.

Dataflow ETL process (extract-transform-load) for populating internal DKB storages and keeping them up to date

Dataflow stage Logical step of ETL process, implemented as standalone executable program (worker)

Dataflow stages are standalone programs, but can be combined into a pipeline by means of Kafka-based supervising program. For details about program compatibility with the supervisor please check documentation for the Metadata Integration Topology Management System (MInT MS) workers<sup>1</sup>. Worker program can be written in any language; pyDKB is intended to simplify this process for Python.

**Warning:** There are three types of stages corresponding three types of ETL operations: *source connector* (data extraction), *processor* (transformation) and *sink connector* (load to internal DKB storage). Currently pyDKB library can be used only for *processor* stages, but in future versions *connector* stages will also be supported.

# 1.1 Quickstart guide

To create simple processor stage application first decide input and output data format. In following examples we will work with data in JSON format (for the full list of supported formats check *pyDKB.dataflow.communication.messages module* section of this documentation).

Now let's start writing example processor welcome.py and implement message handler – functional part of the stage (operations to be performed on data flow units):

```
from pyDKB.dataflow.communication.messages import JSONMessage

def my_process(stage, message):
    """ Single message processing. """
    input_data = message.content()
    name = input_data.get('name')
    if name:
        out_data = {'message': "Welcome, %s!" % name}
        out_message = JSONMessage(out_data)
        stage.output(out_message)
    return True
```

Function must take two arguments: stage (stage context object) and message (input message, which should be transformed by our stage). Message is a smallest data unit in the data flow running through the processor, and every message is to be processed independently of previous or following ones. message.content() and

<sup>1</sup> WIP

JSONMessage (out\_data) statements are used to decode/encode message to/from Python dict object. Message, passed to the function, is taken from the input data flow; to write new message(s) to the output data flow, stage.output(out\_message) is used. It can be used as many times as many output messages were generated (or once with the list of messages). In our example, messages without key 'name' will produce no output messages, so stage.output() will not be called at all. In terms of data flow it means that the input message is filtered out and will not reach the *sink connector*.

Boolean return value of my\_process() indicates if the processing was successful or not. If processing failed (False is returned), produced output messages will be dropped to avoid loading sketchy information into the DKB storages.

Now as we have processing logic implemented, we need to turn it into fully functional application. Add following lines to welcome.py:

First we create stage object: stage = JSONProcessorStage(); then indicate that input and output message format is JSON: stage.set\_{input,output}\_message\_type (messageType.JSON) (for full list of message types check pyDKB.dataflow.communication.messages module section of this documentation); then set stage processing function to our function my\_process(), parse command line arguments and configure stage instance according to it (stage.configure(sys.argv[1:])). Now we ready to start the stage execution.

Easy, right?

It's time to run our application. Create input data sample input.ndjson with following lines:

```
{"name": "James", "city": "New York"}
{"user": "Jonathan", "role": "support"}
{"name": "John Smith"}
```

and type:

```
$ python welcome.py --dest s input.ndjson
{"message": "Welcome, James!"}
{"message": "Welcome, John Smith!"}
```

--dest s indicates that output destination is (s)tdout (default destination is file). For full information about modes in which the stage application can be used, run python welcome.py -h.

That's it, your first application is ready to be integrated into an ETL process as data processing node. For details about ETL process creation check *MInT Supervisor*<sup>2</sup> documentation.

 $<sup>^{2}</sup>$  WIP

# 1.2.1 pyDKB.common package

Common modules.

#### **Submodules**

# pyDKB.common.LoggableObject module

```
pyDKB.common.LoggableObject
```

```
class pyDKB.common.LoggableObject
Bases: object
```

Common ancestor for all classes that need 'log' method.

```
classmethod log(message, level=3)
```

Output log message with given log level.

#### **Parameters**

- message (str) message to output
- level (pyDKB.common.types.logLevel member) log level of the message

### pyDKB.common.Type module

Abstract class for type definitions.

#### **Example**

```
class pyDKB.common.Type.Type(*args)
    Bases: object
```

Abstract class for type definitions.

Member names (str) are passed to the constructor as positional arguments.

```
add(name)
```

Add member.

**Parameters** name (str) – name of the member to be added

#### hasMember (val)

Check if the member exists (by value).

**Parameters val** (*int*) – member to be checked

Returns True/False

Return type bool

#### member (name)

Check if the member exists (by name).

**Parameters** name (str) – name to be checked

Returns member value or False if member does not exist

Return type int, bool

#### memberName (val)

Return string name of the member.

**Parameters val** (*int*) – member to retrieve name for

Returns member name of False if member does not exist

Return type str, bool

### pyDKB.common.custom\_readline module

Implementation of "readline"-like functionality for custom separator.

Todo: make import of fcntl (or of this module) optional to avoid errors when library is used under Windows.

```
pyDKB.common.custom_readline.custom_readline(f, newline)
```

Read lines with custom line separator.

Construct generator with readline-like functionality: with every call of next () method it will read data from f untill the newline separator is found; then yields what was read.

Warning: the last line can be incomplete, if the input data flow is interrupted in the middle of data writing.

To check if iteration is not over without reading next value, one may *send(True)* to the generator: it will return *True* if there is another message to yield or raise *StopIteration* if nothing left.

#### **Parameters**

- **f** (file) readable file object
- **newline** (str) **delimeter** to be used instead of  $\n$

**Returns** iterable object

Return type generator

#### Todo:

- make last "line" handling more strict: no newline == no line;
- rethink function name (as "line" is actually a "message");

5

• move functionality to pyDKB.dataflow.communication submodule)

# pyDKB.common.exceptions module

```
Definition of common modules exceptions
```

```
exception pyDKB.common.exceptions.HDFSException
Bases: exceptions.RuntimeError
Base Exception for HDFS module.
```

### pyDKB.common.hdfs module

```
Utils to interact with HDFS.
pyDKB.common.hdfs.File(fname)
     Get and open temporary local copy of HDFS file
     Return value: open file object (TemporaryFile).
pyDKB.common.hdfs.basename(p)
     Return file name without path.
pyDKB.common.hdfs.check_stderr(proc, timeout=None, max_lines=1)
     Wait till the end of the subprocess and send its STDERR to STDERR.
     Output only MAX_LINES of the STDERR to the current STDERR; if MAX_LINES == None, output all the
     STDERR.
     Return value is the subprocess' return code.
pyDKB.common.hdfs.dirname(p)
     Return dirname without filename.
pyDKB.common.hdfs.getfile(fname)
     Download file from HDFS.
     Return value: file name (without directory)
pyDKB.common.hdfs.join(p, *args)
     Join given paths.
pyDKB.common.hdfs.listdir(dirname, mode='a')
     List files and/or subdirectories of HDFS directory.
     Parameters: dirname – directory to list mode – 'a': list all objects
              'f': list files 'd': list subdirectories
pyDKB.common.hdfs.makedirs(dirname)
     Try to create directory (with parents).
pyDKB.common.hdfs.movefile (fname, dest)
     Move local file to HDFS.
pyDKB.common.hdfs.putfile(fname, dest)
     Upload file to HDFS.
```

<sup>1</sup> https://github.com/PanDAWMS/dkb/pull/129

### pyDKB.common.json\_utils module

Utils to work with JSON (dict) objects.

```
pyDKB.common.json_utils.nestedKeys(key)
```

Transform STRING with nested keys into LIST.

#### **Parameters:**

**STRING key – dot-separated list of nested keys.** If a key contains dot itself, the key must be put between quotation marks.

```
pyDKB.common.json_utils.valueByKey(json_data, key)
```

Return value by a chain (list) of nested keys.

Parameters: DICT json\_data – to search in STRING key – dot-separated list of nested keys

### pyDKB.common.misc module

pyDKB.common.misc

Miscellanious utility functions.

```
pyDKB.common.misc.log(message, level=3, *args)
```

Output log message with given log level.

In case of multiline messages or list of messages only first line (message) is prepended with provided prefixes and timestamp; in all the next lines (messages) they are replaced with special prefix '(==)', representing that these lines belong to the same log record.

Empty lines and lines containing only whitespace symbols are ignored.

### **Parameters**

- message (object) message to output (string, list of strings or any other object)
- level (pyDKB.common.types.logLevel member) log level of the message
- \*args additional prefixes (will be output between log level prefix and message body)

## pyDKB.common.types module

pyDKB.common.types

Definitions of types used across all the library modules.

# 1.2.2 pyDKB.dataflow package

Dataflow organization utils.

# **Subpackages**

### pyDKB.dataflow.communication package

pyDKB.dataflow.communication

```
pyDKB.dataflow.communication.Message (msg_type) Return class XXXMessage, where XXX is the passed type.
```

### **Subpackages**

Consumer submodule init file.

#### pyDKB.dataflow.communication.consumer package

```
class pyDKB.dataflow.communication.consumer.ConsumerBuilder(config={})
    Bases: object
    Constructor for Consumer instance.
    build(config={})
        Return constructed consumer.
    consumerClass = None
    setSource(source)
        Set data source for the consumer.
    setType(Type)
        Set message type for the consumer.
```

#### **Submodules**

### pyDKB.dataflow.communication.consumer.Consumer module

Return message class.

```
pyDKB.dataflow.communication.consumer.Consumer
class pyDKB.dataflow.communication.consumer.Consumer (config={})
     Bases: pyDKB.common.LoggableObject.LoggableObject
     Data consumer implementation.
     close()
         Close opened data stream and data source.
     config = None
     get_message()
          Get new message from current source.
          Return values: Message object False (failed to parse message) None (all input sources are empty)
     get_source_info()
         Return current source info.
     get_stream()
         Get input stream linked to the current source.
          Return value: InputStream None (no sources left to read from)
     init_stream()
         Init input stream.
     message_class()
```

```
message_type = None
     next()
          Return new Message, read from input stream.
     reconfigure (config={})
          (Re)initialize consumer with stage config arguments.
     reset stream()
          Reset input stream to the current source.
     set_message_type(Type)
          Set input message type.
     stream is readable()
          Check if input data stream is readable.
              Returns True - stream is initialized and not empty, False - stream is empty, None - stream is
                 not initialized
              Return type bool, NoneType
exception pyDKB.dataflow.communication.consumer.Consumer.ConsumerException
     Bases: pyDKB.dataflow.exceptions.DataflowException
     Dataflow Consumer exception.
pyDKB.dataflow.communication.consumer.FileConsumer module
pyDKB.dataflow.communication.consumer.FileConsumer
Data consumer implementation for common (static) files.
TODO: think about:

    updatable files

        • pipes (better, from the point of StreamConsumer)
        • round-robin (for updatable sources)
class pyDKB.dataflow.communication.consumer.FileConsumer.FileConsumer(config={/})
     Bases: pyDKB.dataflow.communication.consumer.Consumer.Consumer
     Data consumer implementation for HDFS data source.
     current file = None
     get source()
          Get nearest non-empty source (current or next).
     get_source_info()
          Return current source info.
     init_sources()
          Initialize sources iterator if not initialized yet.
     next_source()
          Reset $current_file to the next non-empty file.
          Return value: File descriptor of the new $current file None (no files left)
```

```
(Re)initialize consumer with Stage configuration.
     source_is_readable()
         Check if current source is readable.
             Returns None – no source, False – source is empty / fully read, True – source is defined and is
                 not empty
             Return type bool, NoneType
pvDKB.dataflow.communication.consumer.HDFSConsumer module
pyDKB.dataflow.communication.consumer.HDFSConsumer
class pyDKB.dataflow.communication.consumer.HDFSConsumer(config={})
     Bases: pyDKB.dataflow.communication.consumer.FileConsumer.FileConsumer
     Data consumer implementation for HDFS data source.
     reconfigure (config={})
         Configure HDFS Consumer according to the config parameters.
pyDKB.dataflow.communication.consumer.StreamConsumer module
pyDKB.dataflow.communication.consumer.StreamConsumer
Data consumer implementation for a single stream.
TODO: think about multiple streams (like a number of named pipes, etc). Prehaps, even merge this class with
     FileConsumer.
class pyDKB.dataflow.communication.consumer.StreamConsumer.StreamConsumer(config={/})
     Bases: pyDKB.dataflow.communication.consumer.Consumer.Consumer
     Data consumer implementation for Stream data source.
     fd = None
     get_source()
         Get Stream file descriptor.
     get_source_info()
         Return current source info.
     next source()
         Return None.
         As currenty we believe that there is only one input stream
     reconfigure (config={})
```

# pyDKB.dataflow.communication.producer package

(Re)configure Stream consumer.

Producer submodule init file.

reconfigure (config={})

```
class pyDKB.dataflow.communication.producer.ProducerBuilder(config={/})
     Bases: object
     Constructor for Producer instance.
     build(config={})
         Return constructed producer.
     message_type = None
     producerClass = None
     setDest (dest)
          Set data destination for the producer.
     setSourceInfoMethod(src_info)
          Set method to get current source info.
     setType(Type)
         Set message type for the producer.
     src info = None
Submodules
pyDKB.dataflow.communication.producer.FileProducer module
pyDKB.dataflow.communication.producer.FileProducer
Data producer implementation for common (static) files.
TODO: think about:
        • pipes (better, from the point of StreamProducer)
        • multiple parallel dests
class pyDKB.dataflow.communication.producer.FileProducer.fileProducer(config={/})
     Bases: pyDKB.dataflow.communication.producer.Producer.Producer
     Data producer implementation for local file data dest.
          Close opened files and remove temporary one.
     close_file()
         Close current file.
     config_dir(config={})
          Configure output directory.
     current_file = None
     default_dir()
         Get default directory name.
     dirname (dirname=None)
          Set/get preferable directory name.
     ensure_dir()
          Ensure that current directory for output files exists.
```

```
file info()
          Return output file metadata (name, directory, ...).
     get_dest()
          Get destination file descriptor.
     get_dest_info()
          Get current destination info.
     get dir()
          Get current directory for output files.
     get_filename()
          Return filename, corresponding the source, or timestamp-based.
     get_source_info()
          Set current data source, if any.
     reconfigure (config={})
          (Re)configure producer according to the config hash.
     reset file()
          Resets current file according to the current source info.
          Metadata include:
                • fd – open file descriptor
                • name – file name
                • dir – directory name
                • local_path – local path to the file
     set_default_dir()
          Set default directory name.
     subdir (base_dir, sub_dir=")
          Construct full path for $subdir of $base_dir.
pyDKB.dataflow.communication.producer.HDFSProducer module
pyDKB.dataflow.communication.producer.HDFSProducer
Data producer implementation for common (static) files in HDFS.
TODO: think about:
        • pipes (better, from the point of StreamProducer)
        • multiple parallel dests
class pyDKB.dataflow.communication.producer.HDFSProducer(config={/})
     Bases: pyDKB.dataflow.communication.producer.FileProducer.FileProducer
     Data producer implementation for HDFS data dest.
     close_file()
          Close current file and move it to HDFS.
     config dir(config={})
          Configure output directory.
```

```
ensure dir()
          Ensure that current directory for output files exists.
     file_info()
          Return output file metadata (name, directory, ...).
     set default dir()
          Set default directory name.
     subdir(base dir, sub dir=")
          Construct full path for $sub_dir of $base_dir.
pyDKB.dataflow.communication.producer.Producer module
pyDKB.dataflow.communication.producer.Producer
class pyDKB.dataflow.communication.producer.Producer.Producer(config={})
     Bases: pyDKB.common.LoggableObject.LoggableObject
     Data producer implementation.
     close()
          Close opened data stream and data dest.
     config = None
     drop()
          Drop buffered messages.
     eop()
          Write EOP marker to the current dest.
     flush()
          Flush buffered messages to the current dest.
     get_dest()
          Return current destination.
     get_dest_info()
          Return current dest info.
     get_stream(actualize=True)
          Get output stream linked to the current dest.
          If $actualize parameter set to True, will try to reset current stream destination; else will use last known
          destination or None.
     init stream()
          Init output stream (without real destination).
     message_class()
          Return message class.
     message_type = None
     reconfigure (config={})
          (Re)initialize producer with stage config arguments.
     reset_stream()
          Reset input stream to the current dest.
     \mathtt{set\_message\_type}\ (\mathit{Type}\ )
          Set input message type.
```

```
write (msg)
         Put new message to the current dest (buffer).
exception pyDKB.dataflow.communication.producer.Producer.ProducerException
     Bases: pyDKB.dataflow.exceptions.DataflowException
     Dataflow Producer exception.
pyDKB.dataflow.communication.producer.StreamProducer module
py DKB. data flow. communication. producer. Stream Producer\\
Data producer implementation for a single stream.
TODO: think about multiple streams (like a number of named pipes, etc). Prehaps, even merge this class with
     FileProducer.
class pyDKB.dataflow.communication.producer.StreamProducer.StreamProducer(config={/})
     Bases: pyDKB.dataflow.communication.producer.Producer.Producer
     Data producer implementation for Stream data dest.
     fd = None
     get dest()
         Get Stream file descriptor.
     get_dest_info()
         Return current dest info.
     reconfigure (config={})
         (Re)configure Stream producer.
pyDKB.dataflow.communication.stream package
pyDKB.dataflow.communication.stream
class pyDKB.dataflow.communication.stream.StreamBuilder(fd, config={})
     Bases: object
     Constructor for Stream object.
     build(config={})
         Create instance of Stream.
     message_type = None
     setStream(stream)
         Set stream type: 'input' or 'output'.
     setType (Type)
         Set message type for the Stream.
     streamClass = None
class pyDKB.dataflow.communication.stream.Stream(fd=None, config=\{\})
     Bases: pyDKB.common.LoggableObject.LoggableObject
     Abstract class for input/output streams.
     EOM = None
```

```
close()
          Close open file descriptors etc.
     configure (config)
          Stream configuration.
     get fd()
          Return open file descriptor or raise exception.
     message_type()
          Get type of the messages in the stream.
     reset (fd, close=True)
          Reset file descriptor in operation.
              Parameters fd – open file descriptor TODO: IOBase objects
              Returns previous file descriptor (or None)
     set_message_type (msg_type)
          Set type of the messages in the stream.
class pyDKB.dataflow.communication.stream.InputStream(fd=None, config={})
     Bases: pyDKB.dataflow.communication.stream.Stream.Stream
     Implementation of the input stream.
     get message()
          Get next message from the input stream.
          Return values: Message object False (failed to parse message) None (no messages left)
     is_readable()
          Check if current input stream is readable.
              Returns None – not initialized, False – empty, True – not empty
              Return type bool, NoneType
     next()
          Get next message from the input stream.
     parse_message (message)
          Verify and parse input message.
          Retrun value: Message object False (failed to parse)
     reset (fd, close=True, force=False)
          Reset current stream with new file descriptor.
          Overrides parent method to reset __iterator property.
class pyDKB.dataflow.communication.stream.OutputStream(fd=None, config={}))
     Bases: pyDKB.dataflow.communication.stream.Stream.Stream
     Implementation of the output stream.
     configure (config={})
          Configure instance.
     drop()
          Drop buffer without sending messages anywhere.
          Signalize Supervisor about end of process.
```

```
flush()
          Flush buffer to the output stream.
     msg_buffer = []
     write (message)
          Add message to the buffer.
Submodules
pyDKB.dataflow.communication.stream.InputStream module
pyDKB.dataflow.communication.stream.InputStream
class pyDKB.dataflow.communication.stream.InputStream.InputStream (fd=None,
                                                                                     config=\{\})
     Bases: pyDKB.dataflow.communication.stream.Stream.Stream
     Implementation of the input stream.
     get_message()
          Get next message from the input stream.
          Return values: Message object False (failed to parse message) None (no messages left)
     is readable()
          Check if current input stream is readable.
              Returns None – not initialized, False – empty, True – not empty
              Return type bool, NoneType
     next()
          Get next message from the input stream.
     parse_message (message)
          Verify and parse input message.
          Retrun value: Message object False (failed to parse)
     reset (fd, close=True, force=False)
          Reset current stream with new file descriptor.
          Overrides parent method to reset __iterator property.
pyDKB.dataflow.communication.stream.OutputStream module
pyDKB.dataflow.communication.stream.OutputStream
class pyDKB.dataflow.communication.stream.OutputStream.OutputStream(fd=None,
                                                                                       con-
                                                                                       fig=\{\})
     Bases: pyDKB.dataflow.communication.stream.Stream.Stream
     Implementation of the output stream.
     configure (config={})
          Configure instance.
     drop()
          Drop buffer without sending messages anywhere.
```

```
eop()
          Signalize Supervisor about end of process.
     flush()
         Flush buffer to the output stream.
     msg_buffer = []
     write (message)
          Add message to the buffer.
pyDKB.dataflow.communication.stream.Stream module
py DKB. data flow. commuication. stream. Stream \\
class pyDKB.dataflow.communication.stream.Stream(fd=None, config={})
     Bases: pyDKB.common.LoggableObject.LoggableObject
     Abstract class for input/output streams.
     EOM = None
     close()
         Close open file descriptors etc.
     configure (config)
         Stream configuration.
     get fd()
         Return open file descriptor or raise exception.
     message_type()
         Get type of the messages in the stream.
     reset (fd, close=True)
          Reset file descriptor in operation.
             Parameters fd – open file descriptor TODO: IOBase objects
             Returns previous file descriptor (or None)
     set_message_type (msg_type)
          Set type of the messages in the stream.
pyDKB.dataflow.communication.stream.exceptions module
pyDKB.dataflow.communication.stream.exceptions
exception pyDKB.dataflow.communication.stream.exceptions.StreamException
     Bases: pyDKB.dataflow.exceptions.DataflowException
     Exception for Stream operations.
Submodules
pyDKB.dataflow.communication.messages module
pyDKB.dataflow.communication.messages
```

```
Definition of abstract message class and specific message classes
class pyDKB.dataflow.communication.messages.AbstractMessage(message=None)
     Bases: object
     Abstract message
     content()
          Return message content.
     decode (code)
          Decode original from CODE to TYPE-specific format.
          Raises ValueError
     decoded = None
     encode (code)
          Encode original message from TYPE-specific format to CODE.
          Raises ValueError
     encoded = None
     classmethod extension()
          Return file extension corresponding to this message type.
     getOriginal()
          Return original message.
     incompl = None
     incomplete(status=None)
          Set message incomplete marker and/or get previous/current value.
              Parameters status (bool, NoneType) – new status (if not passed, current status is re-
                 turned)
              Returns incomplete marker status (previous value, if reset)
              Return type bool
     msg_type = None
     native_types = []
     classmethod typeName()
          Return message type name as string.
exception pyDKB.dataflow.communication.messages.DecodeUnknownType (code, cls)
     Bases: exceptions.NotImplementedError
     Exception to be thrown when message type is not decodable.
\textbf{exception} \hspace{0.1cm} \texttt{pyDKB.dataflow.communication.messages.} \textbf{EncodeUnknownType} \hspace{0.1cm} (code, cls) \\
     Bases: exceptions.NotImplementedError
     Exception to be thrown when message type is not encodable.
class pyDKB.dataflow.communication.messages.JSONMessage(message=None)
     Bases: pyDKB.dataflow.communication.messages.AbstractMessage
     Message in JSON format.
     decode(code=1)
          Decode original data as JSON.
```

```
encode (code=1)
          Encode JSON as CODE.
     incompl_key = '_incomplete'
     incomplete(status=None)
          Set message incomplete marker and/or get previous/current value.
          For JSON messages the marker is implemented as additional field: "incomplete".
             Parameters status (bool, NoneType) - new status (if not passed, current status is re-
                 turned)
             Returns incomplete marker status (previous value, if reset)
             Return type bool
     msg\_type = 2
     native_types = [<type 'dict'>]
pyDKB.dataflow.communication.messages.Message (msg_type)
     Return class XXXMessage, where XXX is the passed type.
class pyDKB.dataflow.communication.messages.TTLMessage(message=None)
     Bases: pyDKB.dataflow.communication.messages.AbstractMessage
     Messages in TTL format
     Single message = single TTL statement
     decode(code=1)
          Decode original data as TTL.
          Currently takes text as it is. TODO: check some formal matter to confirm the string is TTL.
     encode(code=1)
         Encode TTL as CODE.
     msg\_type = 3
     native_types = [<type 'str'>, <type 'unicode'>]
pyDKB.dataflow.stage package
Stage submodule init file.
class pyDKB.dataflow.stage.ProcessorStage (description='DKB Dataflow data processing
                                                     stage.')
     Bases: pyDKB.dataflow.stage.AbstractStage.AbstractStage
     Abstract class to implement Processor stages
     Processor stage – is a stage for data processing/transformation.
     Class/instance variable description:
        • communication.consumer.Consumer instance __input
        • Generator object for output file descriptor OR file descriptor (for (s)tream mode)
              __output
        • List of objects to be "stopped" __stoppable
```

```
clear buffer()
     Drop buffered output messages.
configure (args=None)
     Configure stage according to the config parameters.
     If $args specified, arguments will be parsed anew.
defaultArguments()
     Default parser configuration.
flush_buffer()
     Flush message buffer to the output.
forward()
     Send EOPMarker to the output stream.
get_source_info()
     Get information about current source.
input()
     Generator for input messages.
     Returns iterable object. Every iteration returns single input message to be processed.
input_message_class()
     Get input message class.
output (message)
     Put the (list of) message(s) to the output buffer.
output_message_class()
     Get output message class.
static process(stage, input_message)
     Transform input_message -> output_message.
     To be implemented individually for every stage. Takes the stage as first argument to allow calling output()
         from inside the function.
     Return value: True – processing successfully finished False – processing failed (skip the input message)
run()
     Run process() for every input() message.
set_input_message_type (Type=None)
     Set input message type.
set_output_message_type(Type=None)
     Set output message class.
```

Skip mode is turned on with command line parameter *-skip*; in this mode stage is expected to skip its "semantic" part of the message transformation (like changes in data fields format, definition of additional data fields, etc.) but perform actions required to keep the dataflow seamless.

static skip\_process(stage, input\_message)

Process input\_message in "skip" processing mode.

The simpliest way to achieve this is to send the input message to the output without changes (marking it as "incomplete"), and this is the default implementation of the method.

In some cases it may be necessary to re-implement it (just like process()) to keep the dataflow unbroken.

```
NOTE: the output messages in "skip" mode MUST be marked as "incomplete".
              Parameters input_message (pyDKB.messages.AbstractMessage) - message to
                  process
              Returns True/False (success/failure)
              Return type bool
     stop()
          Finalize all the processes and prepare to exit.
Submodules
pyDKB.dataflow.stage.AbstractStage module
Definition of an abstract class for Dataflow Stages.
class pyDKB.dataflow.stage.AbstractStage.AbstractStage (description='DKB Dataflow
                                                                         stage')
     Bases: pyDKB.common.LoggableObject.LoggableObject
     Class/instance variable description: * Argument parser (argparse.ArgumentParser)
          __parser

    Parsed arguments (argparse.Namespace) ARGS

        • Stage config parser (ConfigParser.SafeConfigParser) __config

    Stage custom config (defaultdict(defaultdict(str))) CONFIG

     add_argument (*args, **kwargs)
          Add specific (not common) arguments.
     args_error (message)
          Output USAGE, error message and exit with code 2.
     config_error (message='Failed to read config file:')
          Output error message and exit with code 3.
     defaultArguments()
          Config argument parser with parameters common for all stages.
     log_configuration()
          Log stage configuration.
     output_error (message=None, exc_info=None)
          Output traceback of the passed (or last) error with message.
     parse_args (args)
          Parse arguments and set dependant arguments if needed.
          Exits in case of error with code: 2 – failed to parse arguments 3 – failed to read config file
     print_usage (fd=<open file '<stderr>', mode 'w'>)
          Print usage message.
```

read\_config()

Read stage custom config file.

```
Returns (TruelFalse)

run ()
Run the stage.

set_error (err_type, err_val, err_trace)
Set object _err variable from the last error info.

stop ()
Stop running processes and output error information.
```

### pyDKB.dataflow.stage.ProcessorStage module

Definition of an abstract class for Dataflow Data Processing Stages.

**USAGE:** ProcessorStage [<options>] [<input files>]

#### **OPTIONS:**

```
-s, --source
                       {flslh} - where to get data from: local (f)iles, (s)tdin, (h)dfs
                       DIR - base directory for relative input file names (for local and
-i, --input-dir
                       HDFS sources). If <input files> not specified, all files from the
                       directory will be taken as the input.
-d, --dest
                       {flslh} - where to send data to: local (f)iles, (s)tdout, (h)dfs
-o, --output-dir
                       DIR - base directory for output files (for local and HDFS
--hdfs
                         • equivalent to "-source h -dest h"
-m, --mode
                       MODE - MODE: (f)ile = -source f
                           -dest f (can be
                             rewritten with 's' or 'h')
                       (s)tream = -source s (can be
                             rewritten with 'h')
                           -dest s
                       (m)apreduce = -source s (can be
                             rewritten with 'h')
                           -dest s
```

Bases: pyDKB.dataflow.stage.AbstractStage.AbstractStage

Abstract class to implement Processor stages

 $Processor\ stage-is\ a\ stage\ for\ data\ processing/transformation.$ 

Class/instance variable description:

- communication.consumer.Consumer instance \_\_input
- Generator object for output file descriptor OR file descriptor (for (s)tream mode)

\_\_output

# • List of objects to be "stopped" \_\_stoppable clear buffer() Drop buffered output messages. configure (args=None) Configure stage according to the config parameters. If \$args specified, arguments will be parsed anew. defaultArguments() Default parser configuration. flush\_buffer() Flush message buffer to the output. forward() Send EOPMarker to the output stream. get\_source\_info() Get information about current source. input() Generator for input messages. Returns iterable object. Every iteration returns single input message to be processed. input message class() Get input message class. output (message) Put the (list of) message(s) to the output buffer. output\_message\_class() Get output message class. static process(stage, input\_message) Transform input\_message -> output\_message. To be implemented individually for every stage. Takes the stage as first argument to allow calling output() from inside the function. Return value: True – processing successfully finished False – processing failed (skip the input message) run() Run process() for every input() message. set\_input\_message\_type (Type=None) Set input message type. set\_output\_message\_type (Type=None) Set output message class. static skip\_process(stage, input\_message) Process input\_message in "skip" processing mode.

Skip mode is turned on with command line parameter *–skip*; in this mode stage is expected to skip its "semantic" part of the message transformation (like changes in data fields format, definition of additional data fields, etc) but perform actions required to keep the dataflow seamless.

The simpliest way to achieve this is to send the input message to the output without changes (marking it as "incomplete"), and this is the default implementation of the method.

In some cases it may be necessary to re-implement it (just like *process()*) to keep the dataflow unbroken.

NOTE: the output messages in "skip" mode MUST be marked as "incomplete".

```
Parameters input_message (pyDKB.messages.AbstractMessage) - message to process

Returns True/False (success/failure)

Return type bool

stop()

Finalize all the processes and prepare to exit.
```

#### **Submodules**

#### pyDKB.dataflow.cds module

Extended CDSInvenioConnector allowing us to login via Kerberos

```
class pyDKB.dataflow.cds.CDSInvenioConnector(*args)
    Bases: invenio_client.contrib.cds.CDSInvenioConnector
    CDSInvenioConnector which closes the browser in most cases.
    delete(restore_handlers=True)
    handlers = False
    kill(signum, frame)
        Run del and propagate signal.
    orig_handlers = {}

class pyDKB.dataflow.cds.KerberizedCDSInvenioConnector(login='user', word='password')
```

Represents same CDSInvenioConnector, but this one is aware about SPNEGO: Simple and Protected GSSAPI Negotiation Mechanism

### pyDKB.dataflow.dkbID module

```
Utils to generate unique yet meaningful identifier for DKB objects.
```

Bases: pyDKB.dataflow.cds.CDSInvenioConnector

```
pyDKB.dataflow.dkbID.dkbID (json_data, data_type)
Return unique identifier for object of TYPE based on DATA.
```

### pyDKB.dataflow.exceptions module

Definition of DKB Dataflow exceptions

```
exception pyDKB.dataflow.exceptions.DataflowException
Bases: exceptions.Exception
Base Exception for Dataflow modules.
```

# pyDKB.dataflow.types module

Type definitions for library objects.

**CHAPTER** 

**TWO** 

# **Stages**

# 2.1 Stage 055

Stage for converting JSON documents (output of stage 015) into TTL documents (input for stage 060). Initial JSON document should have the following structure:

Some functions accept specific parts of this JSON - for example, if 'data' variable contains the initial JSON then "'CDS' part of the initial JSON" means "data.get('CDS')".

Resulting TTL file has the following structure:

```
PAPER a atlas:Paper .
PAPER atlas:hasGLANCE_ID ___ .
PAPER atlas:hasShortTitle ___ .
PAPER atlas:hasFullTitle ___ .
PAPER atlas:hasRefCode ___ .
PAPER atlas:hasCreationDate
PAPER atlas:hasCDSReportNumber ___ .
PAPER atlas:hasCDSInternal ___ .
PAPER atlas:hasCDS_ID __ .
PAPER atlas:hasAbstract ___ .
PAPER atlas:hasArXivCode ___ .
PAPER atlas:hasFullTitle ___ .
PAPER atlas:hasDOI ___ .
PAPER atlas:hasKeyword ___ .
JOURNAL_ISSUE a atlas:JournalIssue .
JOURNAL_ISSUE atlas:hasTitle ___ .
JOURNAL_ISSUE atlas:hasVolume __ .
```

```
JOURNAL_ISSUE atlas:hasYear __ .

JOURNAL_ISSUE atlas:containsPublication> PAPER .

SUPPORTING_DOCUMENT a atlas:SupportingDocument .

SUPPORTING_DOCUMENT atlas:hasGLANCE_ID __ .

SUPPORTING_DOCUMENT atlas:hasLabel __ .

SUPPORTING_DOCUMENT atlas:hasURL __ .

SUPPORTING_DOCUMENT atlas:hasCreationDate __ .

SUPPORTING_DOCUMENT atlas:hasCDSInternal __ .

SUPPORTING_DOCUMENT atlas:hasCDS_ID __ .

SUPPORTING_DOCUMENT atlas:hasAbstract __ .

SUPPORTING_DOCUMENT atlas:hasKeyword __ .

PAPER atlas:isBasedOn SUPPORTING_DOCUMENT .
```

#### **TODO:** This module doesn't convert authors metadata. This task is still under consideration.

055\_documents2TTL.documents2ttl.abstract\_extraction(data) Extract abstract from JSON.

Parameters data (dict) - 'CDS' part of the initial JSON

Returns abstract or None if it was not found

Return type str or NoneType

055\_documents2TTL.documents2ttl.arxiv\_extraction(data) Extract arXiv code from JSON.

Parameters data (dict) - 'CDS' part of the initial JSON

**Returns** arXiv code or None if it was not found

Return type str or NoneType

055\_documents2TTL.documents2ttl.cds\_id\_extraction(data) Extract CDS id from JSON.

Parameters data (dict) - 'CDS' part of the initial JSON

Returns CDS id or None if it was not found

**Return type** int or NoneType

055\_documents2tTL.documents2ttl.cds\_internal\_extraction(data) Extract CDS internal report number parameter from JSON.

Parameters data (dict) - 'CDS' part of the initial JSON

Returns CDS internal report number or None if it was not found

**Return type** unicode or NoneType

055\_documents2TTL.documents2ttl.cds\_parameter\_extraction(param\_name, json\_data) Extract CDS parameter value from JSON.

## **Parameters**

- $param_name(str)$  name of the parameter, defined in \*\_CDS\_ATTRS dict
- json\_data 'CDS' part of the initial JSON

**Returns** parameter value or None if it was not found

**Return type** int, str, NoneType

055\_documents2TTL.documents2ttl.creation\_date\_extraction(*data*) Extract creation date from JSON.

26 Chapter 2. Stages

Parameters data (dict) - 'CDS' part of the initial JSON

Returns creation date or None if it was not found

Return type str or NoneType

055\_documents2TTL.documents2ttl.define\_globals(args)

Define global variables for further usage in other functions.

Global variables GRAPH and ONTOLOGY are defined, their values are received from the command line arguments via argparse.

Parameters args (argparse. Namespace) - stage arguments

055\_documents2TTL.documents2ttl.document\_cds (data, doc\_iri, cds\_attrs)
Convert CDS metadata from JSON to TTL.

#### **Parameters**

- data (dict) 'CDS' part of the initial JSON
- doc\_iri (str) document IRI for current graph
- cds\_attrs(list)-PAPER\_CDS\_ATTRS|NOTE\_CDS\_ATTRS

**Returns** TTL string with CDS metadata

Return type str

055\_documents2TTL.documents2ttl.document\_glance(data, doc\_iri, glance\_attrs)
Convert GLANCE metadata from JSON to TTL.

#### **Parameters**

- data (dict) 'GLANCE' part of the initial JSON
- doc\_iri (str) document IRI for current graph
- $\bullet \ {\tt glance\_attrs} \ ({\tt list}) {\tt PAPER\_GLANCE\_ATTRS} \ | \ {\tt NOTE\_GLANCE\_ATTRS} \\$

Returns TTL string with GLANCE metadata

Return type str

055\_documents2TTL.documents2ttl.document\_links(data)

Construct TTL sentences to link paper to its supporting documents.

The result looks as following: PAPER atlas:isBasedOn SUPPORTING\_DOCUMENT

Parameters data (dict) - initial JSON

**Returns** TTL string with links

**Return type** str

055\_documents2TTL.documents2ttl.doi2ttl(doi,doc\_iri)

Convert DOI parameter from JSON to TTL.

#### **Parameters**

- doi(str, unicode or list) 'doi' part of the initial JSON
- doc\_iri (str) document IRI for current graph

**Returns** TTL string with DOI

Return type str

2.1. Stage 055 27

```
055 documents2TTL.documents2ttl.fix list values(list vals)
     Apply fix_string to each item in a list.
          Parameters list_vals (list) – list with strings to be fixed
          Returns list with fixed strings
          Return type list
055_documents2TTL.documents2ttl.fix_string(wrong_string)
     Fix escape sequences in a string.
          Parameters wrong_string (object) – string to be fixed, or any non-string object.
          Returns fixed string, or unchanged non-string object
          Return type object
055_documents2TTL.documents2ttl.generate_journal_id(journal_dict)
     Generate a journal issue ID based on title, volume and year.
          Parameters journal_dict (dict) – journal parameters
          Returns journal ID
          Return type str
055_documents2TTL.documents2ttl.get_document_iri(doc_id)
     Construct an IRI for a document.
          Parameters doc_id (str) - document id
          Returns IRI
          Return type str
055_documents2TTL.documents2ttl.glance_parameter_extraction(param_name,
                                                                            json_data)
     Extract a GLANCE parameter value from JSON.
         Parameters
               • param_name (str) – name of the parameter
               • json_data (dict) - 'GLANCE' part of the initial JSON
          Returns parameter value or None if it was not found
          Return type str, unicode, NoneType
055_documents2TTL.documents2ttl.keywords2ttl(keywords, doc_iri)
     Convert keywords from JSON to TTL.
          Parameters
               • keywords (dict or list of dicts) - 'keywords' part of the initial JSON
               • doc_iri (str) – document IRI for current graph
          Returns TTL string with keywords
          Return type str
055 documents2TTL.documents2ttl.main(argv)
     Parse command line arguments and run the stage.
          Parameters argv (list) - arguments
```

28 Chapter 2. Stages

055\_documents2TTL.documents2ttl.process(stage, msg)

Process a message. Convert the message's contents from JSON to TTL.

Implementation of AbstractProcessorStage.process() for hooking the stage into DKB workflow. Output message containing the TTL result is generated in this function.

#### **Parameters**

- stage (pyDKB.dataflow.stage.ProcessorStage) stage instance
- msg (pyDKB.dataflow.Message) input message with initial JSON

#### Returns True

### Return type bool

055\_documents2TTL.documents2ttl.process\_journals(data, doc\_iri) Convert journal data from JSON to TTL.

#### **Parameters**

- data (list, dict) 'CDS' part of the initial JSON
- **doc\_iri** (str) document IRI for current graph

Returns TTL string with journal issue with connection to paper

#### Return type str

055\_documents2TTL.documents2ttl.report\_number\_extraction(*data*) Extract report number from JSON.

**Parameters data** (dict) - 'CDS' part of the initial JSON

Returns report number or None if it was not found

Return type unicode or NoneType

055\_documents2TTL.documents2ttl.title\_extraction(data) Extracting title from JSON.

Parameters data (dict) - 'CDS' part of the initial JSON

Returns title or None if it was not found

Return type str or NoneType

2.1. Stage 055 29

30 Chapter 2. Stages

# **CHAPTER**

# **THREE**

# Indices and tables

- genindex
- modindex
- search

# **PYTHON MODULE INDEX**

```
0
                                         pyDKB.dataflow.communication.stream.InputStream,
055_documents2TTL.documents2ttl,25
                                         pyDKB.dataflow.communication.stream.OutputStream,
р
                                         pyDKB.dataflow.communication.stream.Stream,
pyDKB, 1
                                                16
pyDKB.common, 3
                                         pyDKB.dataflow.dkbID.23
pyDKB.common.custom_readline, 4
                                         pyDKB.dataflow.exceptions, 23
pyDKB.common.exceptions,5
                                         pyDKB.dataflow.stage, 18
pyDKB.common.hdfs,5
                                         pyDKB.dataflow.stage.AbstractStage, 20
pyDKB.common.json_utils,6
                                         pyDKB.dataflow.stage.ProcessorStage, 21
pyDKB.common.LoggableObject, 3
                                         pyDKB.dataflow.types, 24
pyDKB.common.misc, 6
pyDKB.common.Type, 3
pyDKB.common.types,6
pyDKB.dataflow, 6
pyDKB.dataflow.cds, 23
pyDKB.dataflow.communication, 6
pyDKB.dataflow.communication.consumer,
pyDKB.dataflow.communication.consumer.Consumer,
pyDKB.dataflow.communication.consumer.FileConsumer,
pyDKB.dataflow.communication.consumer.HDFSConsumer,
pyDKB.dataflow.communication.consumer.StreamConsumer,
pyDKB.dataflow.communication.messages,
pyDKB.dataflow.communication.producer,
pyDKB.dataflow.communication.producer.FileProducer,
pyDKB.dataflow.communication.producer.HDFSProducer,
pyDKB.dataflow.communication.producer.Producer,
pyDKB.dataflow.communication.producer.StreamProducer,
pyDKB.dataflow.communication.stream, 13
pyDKB.dataflow.communication.stream.exceptions,
       16
```

34 Python Module Index

# **INDEX**

0   1	
Symbols	close() (pyDKB.dataflow.communication.consumer.Consumer.Consumer
055_documents2TTL.documents2ttl (module), 25	method), 7
Λ	close() (pyDKB.dataflow.communication.producer.FileProducer.FileProduc
A	method), 10
abstract_extraction() (in module 055_docu-	close() (pyDKB.dataflow.communication.producer.Producer.Producer method), 12
ments2TTL.documents2ttl), 26	close() (pyDKB.dataflow.communication.stream.Stream
AbstractMessage (class in py-	method), 13
DKB.dataflow.communication.messages),	close() (pyDKB.dataflow.communication.stream.Stream
17	method), 16
AbstractStage (class in py-	close_file() (pyDKB.dataflow.communication.producer.FileProducer.FilePro
DKB.dataflow.stage.AbstractStage), 20	method) 10
add() (pyDKB.common.Type.Type method), 3	close_file() (pyDKB.dataflow.communication.producer.HDFSProducer.HDl method), 11
method), 20	method), 11
args_error() (pvDKB.dataflow.stage.AbstractStage.Abstrac	config (pyDKB.dataflow.communication.consumer.Consumer.Consumer attribute), 7
method), 20	
arxiv_extraction() (in module 055_docu-	config (pyDKB.dataflow.communication.producer.Producer.Producer
ments2TTL.documents2ttl), 26	attribute), 12
D	config_dir() (pyDKB.dataflow.communication.producer.FileP
В	config_dir() (pyDKB.dataflow.communication.producer.HDFSProducer.HD
basename() (in module pyDKB.common.hdfs), 5	1 1 14
build()  (pyDKB. data flow. communication. consumer.	method), 11 nerBuilder config_error() (pyDKB.dataflow.stage.AbstractStage.AbstractStage
method), 7	method), 20
build() (pyDKB.dataflow.communication.producer.Produce	method), 20 rBuilder configure() (pyDKB.dataflow.communication.stream.OutputStream
method), 13	ilder configure() (pyDKB.dataflow.communication.stream.OutputStream.OutputS
method), 13	method), 15
C	configure() (pyDKB.dataflow.communication.stream.Stream
cds_id_extraction() (in module 055_docu-	method), 14
ments2TTL.documents2ttl), 26	configure() (pyDKB.dataflow.communication.stream.Stream.Stream method), 16
cds_internal_extraction() (in module 055_docu-	configure() (pyDKB.dataflow.stage.ProcessorStage
ments2TTL.documents2ttl), 26	method), 19
cds_parameter_extraction() (in module 055_docu-	configure() (pyDKB.dataflow.stage.ProcessorStage.ProcessorStage
ments2TTL.documents2ttl), 26	method), 22
CDSInvenioConnector (class in pyDKB.dataflow.cds), 23	Consumer (class in py-
check_stderr() (in module pyDKB.common.hdfs), 5	DKB.dataflow.communication.consumer.Consumer),
clear_buffer() (pyDKB.dataflow.stage.ProcessorStage	7
method), 18	ConsumerBuilder (class in py-
clear_buffer() (pyDKB.dataflow.stage.ProcessorStage.Proc	
method), 22	7

```
consumerClass (pyDKB.dataflow.communication.consumer.drops)\()\( \text{tream}\) (pyDKB.dataflow.communication.stream.OutputStream
                                                      attribute), 7
                                                                                                                                                                                                                                                                                                                                                                                                  method), 14
ConsumerException, 8
                                                                                                                                                                                                                                                                                                                                           drop() (pyDKB.dataflow.communication.stream.OutputStream.OutputStream.
content() (pyDKB.dataflow.communication.messages.AbstractMessagmethod), 15
                                                        method), 17
                                                                                                                                                                                                                                                                                                                                            Ε
creation date extraction()
                                                                                                                                                                  (in module
                                                                                                                                                                                                                                                                  055 docu-
                                                       ments2TTL.documents2ttl), 26
                                                                                                                                                                                                                                                                                                                                            encode() (pyDKB.dataflow.communication.messages.AbstractMessage
current_file (pyDKB.dataflow.communication.consumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileConsumer.FileC
                                                        attribute), 8
                                                                                                                                                                                                                                                                                                                                            encode() (pyDKB.dataflow.communication.messages.JSONMessage
current_file (pyDKB.dataflow.communication.producer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileP
                                                       attribute), 10
                                                                                                                                                                                                                                                                                                                                            encode() (pyDKB.dataflow.communication.messages.TTLMessage
custom_readline()
                                                                                                                                                                                                                  module
                                                                                                                                                  (in
                                                                                                                                                                                                                                                                                                           ру-
                                                                                                                                                                                                                                                                                                                                                                                                   method), 18
                                                      DKB.common.custom readline), 4
                                                                                                                                                                                                                                                                                                                                            encoded (pyDKB.dataflow.communication.messages.AbstractMessage
                                                                                                                                                                                                                                                                                                                                                                                                   attribute), 17
D
                                                                                                                                                                                                                                                                                                                                           EncodeUnknownType, 17
                                                                                                                                                                                                                                                                                                                                           ensure dir() (pyDKB.dataflow.communication.producer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileP
DataflowException, 23
decode() (pyDKB.dataflow.communication.messages.AbstractMessagenethod), 10
                                                                                                                                                                                                                                                                                                                                           ensure dir() (pyDKB.dataflow.communication.producer.HDFSProducer.HD
                                                        method), 17
decode() (pyDKB.dataflow.communication.messages.JSONMessage method), 11
                                                       method), 17
                                                                                                                                                                                                                                                                                                                                            EOM (pyDKB.dataflow.communication.stream.Stream
decode() (pyDKB.dataflow.communication.messages.TTLMessage attribute), 13
                                                        method), 18
                                                                                                                                                                                                                                                                                                                                            EOM (pyDKB.dataflow.communication.stream.Stream
decoded (pyDKB.dataflow.communication.messages.AbstractMessageattribute), 16
                                                       attribute), 17
                                                                                                                                                                                                                                                                                                                                           eop() (pyDKB.dataflow.communication.producer.Producer.Producer
DecodeUnknownType, 17
                                                                                                                                                                                                                                                                                                                                                                                                   method), 12
default_dir() (pyDKB.dataflow.communication.producer.File Druguer.File Druguer.File
                                                       method), 10
                                                                                                                                                                                                                                                                                                                                                                                                   method), 14
                                                                                                                                                                                                                                                                                                                                           eop() (pyDKB.dataflow.communication.stream.OutputStream.OutputStream
defaultArguments()
                                                                                                                                                                                                                                                                                                       (py-
                                                                                                                                                                                                                                                                                                                                                                                                   method), 15
                                                      DKB.dataflow.stage.AbstractStage.AbstractStage
                                                        method), 20
                                                                                                                                                                                                                                                                                                                                           extension() (pyDKB.dataflow.communication.messages.AbstractMessage
defaultArguments()
                                                                                                                                                                                                                                                                                                                                                                                                    class method), 17
                                                                                                                                                                                                                                                                                                       (py-
                                                        DKB.dataflow.stage.ProcessorStage method),
defaultArguments()
                                                                                                                                                                                                                                                                                                      (py-
                                                                                                                                                                                                                                                                                                                                           fd (pyDKB.dataflow.communication.consumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamConsumer.StreamCon
                                                       DKB.dataflow.stage.ProcessorStage.ProcessorStage
                                                                                                                                                                                                                                                                                                                                                                                                    attribute), 9
                                                        method), 22
                                                                                                                                                                                                                                                                                                                                            fd (pyDKB.dataflow.communication.producer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamProducer.StreamPro
define_globals()
                                                                                                                                                                                                                                                                  055_docu-
                                                                                                                               (in
                                                                                                                                                                                    module
                                                                                                                                                                                                                                                                                                                                                                                                   attribute), 13
                                                       ments2TTL.documents2ttl), 27
                                                                                                                                                                                                                                                                                                                                           File() (in module pyDKB.common.hdfs), 5
                                                                     (pyDKB.dataflow.cds.CDSInvenioConnector
delete()
                                                                                                                                                                                                                                                                                                                                           file info() (pyDKB.dataflow.communication.producer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FilePr
                                                       method), 23
                                                                                                                                                                                                                                                                                                                                                                                                  method), 10
dirname() (in module pyDKB.common.hdfs), 5
                                                                                                                                                                                                                                                                                                                                           file_info() (pyDKB.dataflow.communication.producer.HDFSProducer.HDF
dirname() (pyDKB.dataflow.communication.producer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProducer.FileProd
                                                       method), 10
                                                                                                                                                                                                                                                                                                                                           FileConsumer
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      (class
dkbID() (in module pyDKB.dataflow.dkbID), 23
                                                                                                                                                                                                                                                                                                                                                                                                    DKB.dataflow.communication.consumer.FileConsumer),
document cds()
                                                                                                                             (in
                                                                                                                                                                                  module
                                                                                                                                                                                                                                                                  055 docu-
                                                       ments2TTL.documents2ttl), 27
                                                                                                                                                                                                                                                                                                                                           FileProducer
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                (class
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               in
                                                                                                                                                                                         module
document glance()
                                                                                                                                          (in
                                                                                                                                                                                                                                                                  055 docu-
                                                                                                                                                                                                                                                                                                                                                                                                   DKB.dataflow.communication.producer.FileProducer),
                                                       ments2TTL.documents2ttl), 27
document links()
                                                                                                                                    (in
                                                                                                                                                                                      module
                                                                                                                                                                                                                                                                  055 docu-
                                                                                                                                                                                                                                                                                                                                           fix_list_values()
                                                                                                                                                                                                                                                                                                                                                                                                                                                                           (in
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                module
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             055 docu-
                                                       ments2TTL.documents2ttl), 27
                                                                                                                                                                                                                                                                                                                                                                                                   ments2TTL.documents2ttl), 27
doi2ttl()
                                                                                                                                                                    module
                                                                                                                                                                                                                                                                  055 docu-
                                                                                                                                                                                                                                                                                                                                           fix_string()
                                                                                                                                                                                                                                                                                                                                                                                                                                                         (in
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     module
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             055_docu-
                                                        ments2TTL.documents2ttl), 27
                                                                                                                                                                                                                                                                                                                                                                                                   ments2TTL.documents2ttl), 28
drop()\ (pyDKB. data flow. communication. producer. Pr
                                                       method), 12
                                                                                                                                                                                                                                                                                                                                                                                                   method), 12
```

```
flush() (pyDKB.dataflow.communication.stream.OutputStreamt source info() (pyDKB.dataflow.communication.producer.FileProducer.
                        method), 14
                                                                                                                                                                       method), 11
flush() (pyDKB.dataflow.communication.stream.OutputStreamt.QoutputStimeRon) (pyDKB.dataflow.stage.ProcessorStage
                        method), 16
                                                                                                                                                                       method), 19
flush buffer()
                                           (pyDKB.dataflow.stage.ProcessorStage get_source_info() (pyDKB.dataflow.stage.ProcessorStage.ProcessorStage
                        method), 19
                                                                                                                                                                       method), 22
flush buffer() (pyDKB.dataflow.stage.ProcessorStage.ProcessorStage.ProcessorStage.DKB.dataflow.communication.consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Co
                        method), 22
                                                                                                                                                                       method), 7
forward()
                                           (pyDKB.dataflow.stage.ProcessorStage get stream() (pyDKB.dataflow.communication.producer.Producer.Producer
                        method), 19
                                                                                                                                                                       method), 12
forward() (pyDKB.dataflow.stage.ProcessorStage.ProcessorStage() (in module pyDKB.common.hdfs), 5
                        method), 22
                                                                                                                                              getOriginal() (pyDKB.dataflow.communication.messages.AbstractMessage
                                                                                                                                                                       method), 17
G
                                                                                                                                               glance_parameter_extraction() (in module 055_docu-
                                                                                                                                                                       ments2TTL.documents2ttl), 28
generate_journal_id()
                                                                                  module
                                                                                                               055_docu-
                                                               (in
                        ments2TTL.documents2ttl), 28
get_dest() (pyDKB.dataflow.communication.producer.FileProducer.FileProducer
                                                                                                                                              handlers (pyDKB.dataflow.cds.CDSInvenioConnector at-
                        method), 11
get_dest() (pyDKB.dataflow.communication.producer.Producer.Produceribute), 23
                                                                                                                                              hasMember() (pyDKB.common.Type.Type method), 3
                        method), 12
get_dest() (pyDKB.dataflow.communication.producer.Stream Producer(class
                                                                                                                                                                       DKB.dataflow.communication.consumer.HDFSConsumer),
                        method), 13
get dest info() (pyDKB.dataflow.communication.producer.FileProducer.FileProducer
                        method), 11
                                                                                                                                              HDFSException, 5
get_dest_info() (pyDKB.dataflow.communication.producer.PHDE6SProductercer
                                                                                                                                                                                                            (class
                                                                                                                                                                                                                                                  in
                                                                                                                                                                                                                                                                                ру-
                        method), 12
                                                                                                                                                                       DKB.dataflow.communication.producer.HDFSProducer),
get_dest_info() (pyDKB.dataflow.communication.producer.StreamProducer.StreamProducer
                        method), 13
get_dir() (pyDKB.dataflow.communication.producer.FileProducer.FileProducer
                        method), 11
                                                                                                                                              incompl (pyDKB.dataflow.communication.messages.AbstractMessage
get_document_iri()
                                                                                module
                                                                                                               055_docu-
                                                                                                                                                                       attribute), 17
                                                            (in
                        ments2TTL.documents2ttl), 28
                                                                                                                                              incompl key (pyDKB.dataflow.communication.messages.JSONMessage
get_fd() (pyDKB.dataflow.communication.stream.Stream
                                                                                                                                                                       attribute), 18
                        method), 14
                                                                                                                                              incomplete() (pyDKB.dataflow.communication.messages.AbstractMessage
get_fd() (pyDKB.dataflow.communication.stream.Stream
                                                                                                                                                                       method), 17
                        method), 16
                                                                                                                                              incomplete() (pyDKB.dataflow.communication.messages.JSONMessage
get_filename() (pyDKB.dataflow.communication.producer.FileProducen&theBroducer
                        method), 11
                                                                                                                                              init_sources() (pyDKB.dataflow.communication.consumer.FileConsumer.Fi
get message() (pyDKB.dataflow.communication.consumer.Consumer.floethoodness
                        method), 7
                                                                                                                                              init_stream() (pyDKB.dataflow.communication.consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Consumer.Cons
get_message() (pyDKB.dataflow.communication.stream.InputStream_method), 7
                        method), 14
                                                                                                                                              init_stream() (pyDKB.dataflow.communication.producer.Producer.Producer
get_message() (pyDKB.dataflow.communication.stream.InputStream.ImputStream.ImputStream.InputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.ImputStream.Imp
                        method), 15
                                                                                                                                              input() (pyDKB.dataflow.stage.ProcessorStage method),
get_source() (pyDKB.dataflow.communication.consumer.FileConsumerPfileConsumer
                        method), 8
                                                                                                                                              input() (pyDKB.dataflow.stage.ProcessorStage.ProcessorStage
get_source() (pyDKB.dataflow.communication.consumer.StreamConsumethshipeathConsumer
                        method), 9
                                                                                                                                              input message class()
                                                                                                                                                                                                                                                                              (py-
get_source_info() (pyDKB.dataflow.communication.consumer.Consumer&Bahstaflow.stage.ProcessorStage method),
                        method), 7
                                                                                                                                                                       19
get_source_info() (pyDKB.dataflow.communication.consumerptitle@consumer
                                                                                                                                                                                                                                                                              (py-
                        method), 8
                                                                                                                                                                       DKB.dataflow.stage.ProcessorStage.ProcessorStage
get_source_info() (pyDKB.dataflow.communication.consumer.Stream Getsut)et2StreamConsumer
                        method), 9
```

InputStream DKB.dat	(class aflow.communication	in on.stream), 14		_	type (pyDKB.dataflow.communication.producer.ProducerBuilder attribute), $10$
InputStream DKB.dat	(class aflow.communication	in on.stream.Inpi			type (pyDKB.dataflow.communication.stream.StreamBuilder attribute), 13
15		г			type() (pyDKB.dataflow.communication.stream.Stream
	KB.dataflow.comm	unication.strea	ım.Inpı		
method),			1		type() (pyDKB.dataflow.communication.stream.Stream
is_readable() (pyD	KB.dataflow.comm	unication.strea	ım.Inpı		
method),	. 15				) (in module pyDKB.common.hdfs), 5 er (pyDKB.dataflow.communication.stream.OutputStream
J				_	attribute), 15
ioin() (in module p	yDKB.common.hdt	Fs). 5			er (pyDKB.dataflow.communication.stream.OutputStream.Output
JSONMessage	(class	in	ру-	_	attribute), 16
	aflow.communication	on.messages),	1.	msg_type	(pyDKB.dataflow.communication.messages.AbstractMessage
17					attribute), 17
K					(pyDKB.dataflow.communication.messages.JSONMessage attribute), 18
KerberizedCDSInv	venioConnector	class in	py-		(pyDKB.dataflow.communication.messages.TTLMessage
DKB.dat	aflow.cds), 23	•	1,		attribute), 18
keywords2ttl() ments2T	(in modul TL.documents2ttl),	_	docu-	N	
kill() (pyDK method),	XB.dataflow.cds.CD	SInvenioConn	ector		bes (pyDKB.dataflow.communication.messages.AbstractMessage attribute), 17
L					bes (pyDKB.dataflow.communication.messages.JSONMessage attribute), 18
	pyDKB.common.h yDKB.common.mis				pes (pyDKB.dataflow.communication.messages.TTLMessage attribute), 18
	nmon.LoggableObje		hiect		vs() (in module pyDKB.common.json_utils), 6
class met	thod), 3	ct.LoggableO		next() (py	DKB.dataflow.communication.consumer.Consumer.Consumer
log_configuration()		.0. 41 .	(py-		method), 8
		tStage.Abstra	ctStage		DKB.dataflow.communication.stream.InputStream method), 14
method), LoggableObject	(class	in	py-		DKB.dataflow.communication.stream.InputStream.InputStream
	nmon.LoggableObj		ру-		method), 15
M	illion.Loggableobj	cct), 3		next_sour	ce() (pyDKB.dataflow.communication.consumer.FileConsumer.Fi method), 8
	055_documents2T	TI doguments	·2++1\		ce() (pyDKB.dataflow.communication.consumer.StreamConsume
28			szui),		method), 9
	lule pyDKB.commo .common.Type.Typ			0	
	yDKB.common.Typ lule pyDKB.dataflo				llers (pyDKB.dataflow.cds.CDSInvenioConnector attribute), 23
6	iule pyDKB.uatano	w.communica	11011),		pyDKB.dataflow.stage.ProcessorStage method),
Message()	`	dule	py-		19
DKB.dat 18	aflow.communication	on.messages),			pyDKB.dataflow.stage.ProcessorStage.ProcessorStage method), 22
message_class() (p	yDKB.dataflow.con	nmunication.c	onsum	eactpus <u>u</u> ca	eor Compschile B. dataflow.stage. Abstract Stage. Abstract Stage
method),	7				method), 20
	yDKB.dataflow.com	nmunication.p	roduce	_	= = = = = = = = = = = = = = = = = = = =
method),					DKB.dataflow.stage.ProcessorStage method),
	OKB.dataflow.comm	nunication.con	sumer.		
attribute)		• ,•		-	essage_class() (py-
message_type (pyL attribute)		iunication.pro	aucer.F		<b>DKiBed</b> ataflow.stage.ProcessorStage.ProcessorStage method), 22
auribute	1, 1 <i>L</i>				memou), ZZ

OutputStream DKB.dataflov	(class v.communication	in n.stream), 14	ру-	pyDKB.dataflow.communication.consumer.HDFSConsumer (module), 9
OutputStream	(class	in	nv-	pyDKB.dataflow.communication.consumer.StreamConsumer
-	w.communication			
15				pyDKB.dataflow.communication.messages (module), 16
				pyDKB.dataflow.communication.producer (module), 9
Р				pyDKB.dataflow.communication.producer.FileProducer
parse_args() (pyDKB.d	ataflow stage Ah	stractStage A	hetrac	± • • • • • • • • • • • • • • • • • • •
method), 20				pyDKB.dataflow.communication.producer.HDFSProducer
	B.dataflow.com	munication.s	tream.	InputStreammodule), 11
method), 14				pyDKB.dataflow.communication.producer.Producer
	B.dataflow.com	munication.s	tream.	InputStream InputStream
method), 15				pyDKB.dataflow.communication.producer.StreamProducer
	dataflow.stage.A	bstractStage.	Abstra	actStage (module), 13
method), 20			_	pyDKB.dataflow.communication.stream (module), 13
process() (in	module		locu-	pyDKB.dataflow.communication.stream.exceptions
	locuments2ttl), 2			(module), 16
process() (pyDKB.dat method), 19	aflow.stage.Proc	essorStage s	static	pyDKB.dataflow.communication.stream.InputStream (module), 15
	flow stage Proces	ssorStage Pro	ocessor	RENIDEKB.dataflow.communication.stream.OutputStream
static method		ssorbinge.i re		(module), 15
process_journals()	(in modul	e 055 d	locu-	pyDKB.dataflow.communication.stream.Stream (mod-
	locuments2ttl), 2		iocu	ule), 16
ProcessorStage (class in				pyDKB.dataflow.dkbID (module), 23
ProcessorStage (class in	(class	in	nv-	pyDKB.dataflow.exceptions (module), 23
_	w.stage.Processo		РУ	pyDKB.dataflow.stage (module), 18
	-	in	ру-	
			oduce Py	rpyDKB.dataflow.stage.ProcessorStage (module), 21
12	w.communication	i.producei.i i	oduce	pyDKB.dataflow.types (module), 24
ProducerBuilder	(class	in	ру-	p) 2 12 (datation) pas (modulo), 2
	w.communication		РЈ	R
9	v.communication	nproducer),		read_config() (pyDKB.dataflow.stage.AbstractStage.AbstractStage
producerClass (pyDKB	dataflow commi	inication pro	ducer	ProducerRuilderad) 20
attribute), 10	.datanow.commi	ameuron.pro	aucei.	reconfigure() (pyDKB.dataflow.communication.consumer.Cons
ProducerException, 13				
putfile() (in module pyI	OKB common h	ife) 5		method), 8
pyDKB (module), 1	JKD.Common.no	115), 5		reconfigure() (pyDKB.dataflow.communication.consumer.FileConsumer.File
pyDKB (module), 1	ula) 3			method), 8
pyDKB.common.custor		hule) 4		reconfigure() (pyDKB.dataflow.communication.consumer.HDFSConsumer.
pyDKB.common.excep				method), 9
pyDKB.common.hdfs (		3		reconfigure() (pyDKB.dataflow.communication.consumer.StreamConsumer
= -				method), 9
pyDKB.common.json_i		v1a) 2		reconfigure() (pyDKB.dataflow.communication.producer.FileProducer.FileI
pyDKB.common.Logga	•	uie), 3		method), 11
pyDKB.common.misc				reconfigure() (pyDKB.dataflow.communication.producer.Producer.Producer
pyDKB.common.Type				method), 12
pyDKB.common.types				reconfigure() (pyDKB.dataflow.communication.producer.StreamProducer.S
pyDKB.dataflow (modu				method), 13
pyDKB.dataflow.cds (n		1-) 6		report_number_extraction() (in module 055_docu-
pyDKB.dataflow.comm			. 7	ments2TTL.documents2ttl), 29
pyDKB.dataflow.comm				reset() (pyDKB.dataflow.communication.stream.InputStream
pyDKB.dataflow.communication.consumer.Consumer				method), 14
(module), 7	umication	man Eil-Ca		reset() (pyDKB.dataflow.communication.stream.InputStream.InputStream
pyDKB.dataflow.comm (module), 8	iumcanon.consu	mer.rnecons	umer	method), 15

reset()	(pyDKB.dataflow.communication.stream.Stream method), 14		n() (pyDKB.dataflow.communication.stream.StreamBuilder method), 13
reset() (	pyDKB.dataflow.communication.stream.Stream.St method), 16	• • •	(pyDKB.dataflow.communication.consumer.ConsumerBuilder method), 7
reset_fil	e() (pyDKB.dataflow.communication.producer.File	ePsretaTiyquer(JF	[hehbitBcdataflow.communication.producer.ProducerBuilder
	method), 11		method), 10
reset_st	ream() (pyDKB.dataflow.communication.consume method), 8		(Control of the Control of the Contr
reset_st	ream() (pyDKB.dataflow.communication.producer method), 12		https://www.stage.processorStage static method), 19
run() (p	yDKB.dataflow.stage.AbstractStage.AbstractStage method), 21		cess() (pyDKB.dataflow.stage.ProcessorStage.ProcessorStage static method), 22
run() (p	yDKB.dataflow.stage.ProcessorStage method), 19	source_is	_readable() (py-
run() (p	yDKB.dataflow.stage.ProcessorStage.ProcessorSta method), 22	-	DKB.dataflow.communication.consumer.FileCons
	method), 22		pyDKB.dataflow.communication.producer.ProducerBuilder
S		~~~	attribute), 10
set defa	ult_dir() (pyDKB.dataflow.communication.produc	eistad(b(b)	
set_dere	method), 11	CI.I HOI KUU	method), 21
set_defa	nult_dir() (pyDKB.dataflow.communication.produc method), 12	er:14DFSPP	WARE HISTORY SINGE Processor Stage method),
set_erro	r() (pyDKB.dataflow.stage.AbstractStage.Abstract method), 21	Statep() (py	DKB.dataflow.stage.ProcessorStage.ProcessorStage method), 23
set inn	it_message_type() (py-		lass in pyDKB.dataflow.communication.stream),
sct_mpt	DKB.dataflow.stage.ProcessorStage method),	(1	13
	19	Stream (c	lass in pyDKB.dataflow.communication.stream.Stream),
set inni	it_message_type() (py-	`	16
set_mpt	DKB.dataflow.stage.ProcessorStage.ProcessorSt	astream_is	s_readable() (py-
	method), 22	8-	DKB.dataflow.communication.consumer.Consumer.Consumer
set_mes	sage_type() (py-		method), 8
	DKB.dataflow.communication.consumer.Consur	n <b>StComB</b> i	
	method), 8		DKB.dataflow.communication.stream), 13
set_mes	sage_type() (py-		ass (pyDKB.dataflow.communication.stream.StreamBuilder
	DKB.dataflow.communication.producer.Produce	r.Producer	attribute), 13
	method), 12	StreamCo	`
set_mes	sage_type() (py-		DKB.dataflow.communication.consumer.StreamConsumer),
	DKB.dataflow.communication.stream.Stream	C. F	9
	method), 14		sception, 16
set_mes	sage_type() (py-	StreamPr	oducer (class in py- DKB.dataflow.communication.producer.StreamProducer),
	DKB.dataflow.communication.stream.Stream.St	ream	13
44	method), 16	subdir() (	pyDKB.dataflow.communication.producer.FileProducer.FileProdu
set_out	out_message_type() (py- DKB.dataflow.stage.ProcessorStage method),		method), 11
	19		pyDKB.dataflow.communication.producer.HDFSProducer.HDFSF
set outr	out_message_type() (py-		method), 12
set_out	DKB.dataflow.stage.ProcessorStage.ProcessorSt		,,
	method), 22	"°T	
setDest(	() (pyDKB.dataflow.communication.producer.Prod	ucteitlBuekter	action() (in module 055_docu-
5012 051(	method), 10	<u></u>	ments2TTL.documents2ttl), 29
setSourc	ce() (pyDKB.dataflow.communication.consumer.C	onETihMeBu	**
	method), 7		DKB.dataflow.communication.messages),
setSour	ceInfoMethod() (py-		18
	DKB.dataflow.communication.producer.Produce	er <b>Bypde</b> cla	ss in pyDKB.common.Type), 3
	method), 10		

typeName() (pyDKB.dataflow.communication.messages.AbstractMessage class method), 17

# ٧

valueByKey() (in module pyDKB.common.json\_utils), 6

# W

write() (pyDKB.dataflow.communication.producer.Producer.Producer method), 12

write() (pyDKB.dataflow.communication.stream.OutputStream method), 15

 $write () \ (pyDKB. data flow. communication. stream. Output Stre$