**Data Receiving module:**

This module is able to read data from the various data sources including.

**1. CSV**: Path to the CSV file and delimiter type.

**2. TSV**: Similar to CSV but uses a tab character (\t) as the delimiter.

**3. JSON**: Path to the JSON file with encoding.

**4. XML**: Path to the XML file with encoding, root and row elements specified for processing.

**5. S3**: Configuration for reading from an S3 bucket, including AWS credentials and region.

**6. MongoDB**: Connection information for MongoDB, including host, port, database, collection, and credentials.

**7. Spark**: Spark application configuration, including the master URL, app name, JAR file path, and specific Spark configurations.

**Configuration file:**

{

"production": {

"execution\_mode": {

"sequential": {

"on\_failure": "continue"

},

"parallel": {

"max\_threads": 5,

"on\_failure": "retry"

},

"specific\_order": {

"order": [

"csv",

"json",

"mongodb",

"s3",

"tsv",

"xml",

"spark"

],

"on\_failure": "skip"

}

},

"data\_sources": [

{

"type": "csv",

"path": "/path/to/file.csv",

"delimiter": ",",

"encoding": "utf-8"

},

{

"type": "tsv",

"path": "/path/to/file.tsv",

"delimiter": "\t",

"encoding": "utf-8"

},

{

"type": "json",

"path": "/path/to/file.json",

"encoding": "utf-8"

},

{

"type": "xml",

"path": "/path/to/file.xml",

"encoding": "utf-8",

"root\_element": "root",

"row\_element": "row"

},

{

"type": "s3",

"bucket": "my-bucket-name",

"prefix": "folder/subfolder/",

"aws\_access\_key\_id": "YOUR\_AWS\_ACCESS\_KEY\_ID",

"aws\_secret\_access\_key": "YOUR\_AWS\_SECRET\_ACCESS\_KEY",

"region": "us-west-2"

},

{

"type": "mongodb",

"host": "localhost",

"port": 27017,

"database": "my\_db",

"collection": "my\_collection",

"username": "user",

"password": "pass"

},

{

"type": "spark",

"master": "spark://localhost:7077",

"app\_name": "MySparkApp",

"jar": "/path/to/spark-app.jar",

"config": {

"spark.executor.memory": "4g",

"spark.driver.memory": "2g"

}

}

]

},

"development": {

"execution\_mode": {

"sequential": {

"on\_failure": "continue"

},

"parallel": {

"max\_threads": 2,

"on\_failure": "skip"

},

"specific\_order": {

"order": [

"csv",

"json",

"s3",

"mongodb"

],

"on\_failure": "continue"

}

},

"data\_sources": [

{

"type": "csv",

"path": "/dev/path/to/file.csv",

"delimiter": ",",

"encoding": "utf-8"

},

{

"type": "json",

"path": "/dev/path/to/file.json",

"encoding": "utf-8"

},

{

"type": "s3",

"bucket": "dev-bucket-name",

"prefix": "folder/dev/",

"aws\_access\_key\_id": "DEV\_AWS\_ACCESS\_KEY\_ID",

"aws\_secret\_access\_key": "DEV\_AWS\_SECRET\_ACCESS\_KEY",

"region": "us-west-2"

},

{

"type": "mongodb",

"host": "localhost",

"port": 27017,

"database": "dev\_db",

"collection": "dev\_collection",

"username": "dev\_user",

"password": "dev\_pass"

}

]

},

"test": {

"execution\_mode": {

"sequential": {

"on\_failure": "skip"

},

"parallel": {

"max\_threads": 3,

"on\_failure": "retry"

},

"specific\_order": {

"order": [

"csv",

"mongodb",

"s3"

],

"on\_failure": "continue"

}

},

"data\_sources": [

{

"type": "csv",

"path": "/test/path/to/file.csv",

"delimiter": ",",

"encoding": "utf-8"

},

{

"type": "mongodb",

"host": "localhost",

"port": 27017,

"database": "test\_db",

"collection": "test\_collection",

"username": "test\_user",

"password": "test\_pass"

},

{

"type": "s3",

"bucket": "test-bucket-name",

"prefix": "folder/test/",

"aws\_access\_key\_id": "TEST\_AWS\_ACCESS\_KEY\_ID",

"aws\_secret\_access\_key": "TEST\_AWS\_SECRET\_ACCESS\_KEY",

"region": "us-east-1"

}

]

},

"support": {

"execution\_mode": {

"sequential": {

"on\_failure": "skip"

},

"parallel": {

"max\_threads": 1,

"on\_failure": "skip"

},

"specific\_order": {

"order": [

"tsv",

"xml"

],

"on\_failure": "skip"

}

},

"data\_sources": [

{

"type": "tsv",

"path": "/support/path/to/file.tsv",

"delimiter": "\t",

"encoding": "utf-8"

},

{

"type": "xml",

"path": "/support/path/to/file.xml",

"encoding": "utf-8",

"root\_element": "root",

"row\_element": "row"

}

]

}

}

### Explanation of New Modes:

1. **"production"**: For live or production environments, with full configurations and more robust handling.
2. **"development"**: Used for development purposes with different configurations, such as a reduced number of threads and development-specific paths and credentials.
3. **"test"**: Designed for testing, with configurations for a small set of data sources and handling failures with retries.
4. **"support"**: Intended for support or troubleshooting environments with minimal configuration and simplified execution modes.

Each mode allows you to tailor the configuration for different environments, making it easy to switch between production, development, testing, and support scenarios while maintaining a consistent structure.

* **"start": false** is added to each mode section (production, development, test, and support).
* This parameter can be used as a flag to indicate whether the mode should be active or inactive.

### Explanation of the specific\_order Configuration:

* **"specific\_order"**: Defines the exact order in which the data sources should be processed.
  + **"order"**: A list of data source types in the specific order that they should be processed. For example, it will first process csv, then json, and so on.
  + **"on\_failure"**: Specifies the action to take when a failure occurs in a data source. In this case, "skip" means that if an error occurs with a specific source, it will skip to the next source in the order.

### Example Use Case:

* **Specific Order Execution**: If "execution\_mode" is set to "specific\_order", the data sources will be processed in the defined order (csv, json, mongodb, etc.), regardless of their position in the data\_sources list. If any source fails, it will be skipped and the next source will be processed as per the order.

This configuration allows for greater control over the processing order of data sources, enabling specific workflows where the sequence of operations matters.

### Use Case:

**"execution\_mode"**: A new top-level section containing configuration for both sequential and parallel processing.

* + **sequential**: Includes an "on\_failure" option that dictates what happens when a failure occurs (e.g., "continue" on failure).
  + **parallel**: Includes configuration for parallel processing, such as "max\_threads" (maximum number of threads to use for parallel execution) and "on\_failure" behavior (e.g., "retry" on failure).
* **Sequential Execution**: If "execution\_mode" is set to "sequential", the data sources will be processed one by one. If a failure occurs in any data source, it will continue to the next source based on the "on\_failure": "continue" setting.
* **Parallel Execution**: If "execution\_mode" is set to "parallel", data sources will be processed concurrently. The "max\_threads" setting can control how many sources can be processed at the same time. In case of a failure, the "on\_failure": "retry" option allows for retrying the failed source.

This configuration gives flexibility to switch between sequential and parallel data processing depending on the use case.