Unica Journey V12.1.8 Tuning Guide

Contents

Chapter 1. About tuning Unica Journey for best performance

An installation of Unica Journey consists of several components including third-party tools (such as web application servers, databases, and load balancers) and components such as Unica Platform. All of these components have several properties, features, and settings you can configure to improve performance.

Unica Journey itself has several configuration properties which you can use to tune your installation for best performance.

Defining 'best performance' is difficult. Every environment, every implementation has different requirements.

Unica Journey runtime performance can be affected by many factors, including hardware configuration, network configuration, and Unica Journey configuration. The following guidelines and recommendations can have different results in your environment.

Journey Master Configuration Properties

List of engine properties for configuring Journey Master

1. spring.kafka.retries.config = number of time Kafka Producer Client should try to submit data to broker in case of failure
2. connections.max.idle.ms=-1, specified the idle time to keep Producers and Consumer Alive by default is 540 seconds, changing it to -1 to keep the connections alive.
3. *.topic=DATA_CLEAN--specifies name of topic from where data is read by service(DO NOT CHANGE)
4. *.topic.group.id=data-clean-service-consumer-group , -group to which consumer of this service belongs, this is unique for each service (DO NOT CHANGE)
5. *.topic.max.poll.records = number of message service reads ,each time it communicates with broker
6. *.topic.partitions = number of partition per topic (recommended:5)
7. *.topic.replications = Ref :Configuration for replication of Kafka
8. *.topic.min.threshold = values used for scaling service, based on number of messages polled, number of IO operations performed (default:2), should always be less than max threshold
9. *.topic.max.threshold = values used for scaling service, based on number of messages polled, number of IO operations performed (default:10), should always be greater than min threshold

   (Service threshold ping should be between min and max threshold, if ping is below min then service is considered for scaling-in (Instance is stopped), if threshold ping is more than max then service is considered for scale-out (Instance is increased)
10. *.topic.max.instance = (Number of max Instance this service can have , when threshold value increases, value of this parameter should always be equal to value of topic.partitions (recommended:5)
11. *.topic.default.instance = default number of instance for this service (recommended:1)
12. *.topic.sleep.time = sleep time(in milli seconds) between processing of two batches (recommended:500)

13. *.topic.max.isalive.threshold = threshold value(in milli seconds) to compare heartbeat of each service instance, if threshold value increase this assigned value , instance is restarted (recommended:200)
14. journey_control.topic=JOURNEY_CONTROL, used for communication between web/engine/platform (DO NOT CHANGE)
15. journey.engine.monitoring.topic=JOURNEY_ENGINE_MONITORING, used for monitoring service (DO NOT CHANGE)
16. journey.engine.errors.topic=JOURNEY_ENGINE_ERRORS -all engine errors are posted to this topic (DO NOT CHANGE)
17. pause_audience_exec.topic.max.isalive.threshold = 18000 (in seconds). This value is set based on the following data - `Total audiences: 5 million | Audiences paused: 1 million`. This parameter needs to be set to higher values if audience is higher than 5 million.
18. pause_audience_exec.topic.max.poll.interval = 18000000 (in milliseconds). This value is set based on the following data - `Total audiences: 5 million | Audiences paused: 1 million`. This parameter needs to be set to higher values if audience is higher than 5 million.
19. hip.request.topic=OUTGOING_MESSAGES, topic used for communicating with HIP service for sending (Email, SMS) (DO NOT CHANGE)
20. journey.kafka.consumer.fetch.min.bytes=2097152 and journey.kafka.consumer.max.partition.fetch.bytes=2097152 increasing value for these parameter may give you performance improvements based on the OS and network socket buffer size.
21. decision-split.retry.max-attempts = 3 and decision-split.retry.back-off.delay-ms = 2000 [Fixed delay (in milliseconds) between two retrial attempts for decision split query execution] This parameter needs to be set to higher values in case An Error - Transaction system exception occurs in the logs and JDBC query fails.
22. Oracle DB may need to tune for redo log size based on the Audience data inserted in the Journey tables.

Pause Audience

In the `journey_master_config.properties` file of journey Engine which resides under <install>/Journeys/Engine/, ensure that thresold and poll.interval values are very high (in terms of hours) for **PAUSE_AUDIENCE_CALC_COUNT** and **PAUSE_AUDIENCE_EXEC** entries . Since pause rule when run might result in a time-consuming query (it being analytical), the time allotted to kafka consumer for pause rule processing activity might be insufficient. Depending on customer's use case, these values need to be increased further:

```
pause_audience_calc_count.topic=PAUSE_AUDIENCE_CALC_COUNT

pause_audience_calc_count.topic.group.id=exit-user-consumer-group

pause_audience_calc_count.topic.max.poll.records=5

pause_audience_calc_count.topic.partitions = 5

pause_audience_calc_count.topic.replications = 1

pause_audience_calc_count.topic.min.threshold = 4

pause_audience_calc_count.topic.max.threshold = 10

pause_audience_calc_count.topic.max.instance = 5

pause_audience_calc_count.topic.default.instance = 1

pause_audience_calc_count.topic.sleep.time = 50

pause_audience_calc_count.topic.max.isalive.threshold = 3600
pause_audience_calc_count.topic.max.poll.interval= 3600000
pause_audience_exec.topic=PAUSE_AUDIENCE_EXEC
pause_audience_exec.topic.group.id=pause-audience-exec-consumer-group
pause_audience_exec.topic.max.poll.records=5
pause_audience_exec.topic.partitions = 5
pause_audience_exec.topic.replications = 1
pause_audience_exec.topic.min.threshold = 4
pause_audience_exec.topic.max.threshold = 10
pause_audience_exec.topic.max.instance = 5
pause_audience_exec.topic.default.instance = 1
pause_audience_exec.topic.sleep.time = 50
pause_audience_exec.topic.max.isalive.threshold = 18000
pause_audience_exec.topic.max.poll.interval = 18000000
```

*All Properties are part of journey_master_config.properties file

Table 1. Configuration for replication of Kafka

Number of Kafka Broker	Replication Factor
1	1
2	1
3	2
5	3
7	5

Journey Engine Application Properties

List of engine application properties

spring.ignite.ipFinder.List - This property is used to set pre-configured list of IP addresses specified By default, this IP finder is not shared, which means that all grid nodes have to be configured with the same list of IP addresses. For example (Server A and Server B are in cluster then for:

1. Server A – (ServerA_IP:port,ServerB_IP:Port)
2. Server B - – (ServerB_IP:port,ServerA_IP:Port)

spring.ignite.defaultDataRegion.max.size - This property is used for memory allocation to Ignite; the total size should not be less than 10 MB (Recommended 2GB on 16GB RAM in Linux)/(1GB on 16GB RAM in Windows).

journey.audience.next.state.on.data.error - This property is to decide whether audiences will move to 'no flow' or 'error state'. Values can be 'error' or 'no'

engine.logging.cron - This property is used to set run time of logging interaction scheduler job

journey_configuration.topic - specifies name of topic from where data is read by service(DO NOT CHNAGE)

journey_configuration.topic.group.id Group to which consumer of this service belongs, this is unique for each service (DO NOT CHNAGE)

journey_configuration.short_sleep sleep time(in milli seconds) between processing of two batches

restRequest.topic.max.poll.interval - This parameter need to set to higher values restRequest.topic.max.poll.interval = <3600000> in case commit failed exception is seen in the logs for a specific service then following parameters for that service need to set higher values:

- topic.max.isalive.threshold = 3600 [Sec]
- topic.max.poll.interval = 3600000 [Milli Seconds]

web.client.connections.pendingAcquireTimeout=90 - Specifies the maximum time after which a pending acquire must complete at the TCP level else the TimeoutException will be thrown (The value is in Second)

spring.jpa.show-sql - This property is used for troubleshooting issues and when user wants to check the hibernate generated sql queries.

journey.datasource.maxpool.size - This property specifies maximum datasource pool size.

journey.datasource.minIdle.size - This property specifies minimum datasource pool size.

journey.datasource.connectionTimeout - This property specifies connection timeout (in milli seconds).

journey.datasource.transactionTimeout - This property specifies transaction timeout (in milli seconds).

journey.kafka.producer.linger.ms - This property in milli seconds is used to introduce a delay in sending the messages so that we can increase the chances of messages being sent together in a batch. Default value is zero. This property helps to increase the throughput, compression, and efficiency of the producer.

journey.kafka.producer.max.batch.size - This property specifies the maximum number of bytes that will be included in one batch, default is 16KB.

journey.kafka.consumer.fetch.min.bytes - This property specifies the minimum amount of data the server should return for 1 fetch request. In case the specified data is not available yet then the request will wait for that much data to accumulate before answering the request.

journey.kafka.consumer.max.partition.fetch.bytes - This property specifies the maximum amount of data per-partition the server will return.

journey.kafka.producer.compression.type - This property specifies the compression type. Possible options includes none, gzip, snappy, lz4, zstd.

spring.task.scheduling.pool.size - This property specifies scheduler thread pool size. Thread pool size must be increased to match the number of scheduled tasks.

archival.audience.cron - This property is used to set run time of audience archival scheduler job.

Journey Engine Service Thread Configuration

Journey engine is using all thread pool configs from service_config.properties

1. **Synchronous thread pool configuration**- This thread pool is managing all journey engine service threads.
    a. **sync.thread.pool.max.size=500** (ThreadPoolExecutor's maximum pool size).
    b. **sync.thread.pool.core.size=200** (ThreadPoolExecutor's core pool size).
    c. **sync.thread.pool.queue.capacity=10** (Capacity for the ThreadPoolExecutor's BlockingQueue).
2. **Asynchronous thread pool configuration:** This thread pool is managing all operation which can be done asynchronously. e.g.: Storing discarded data & reporting data into database, delay touchpoint processing.
    a. **async.thread.pool.max.size=20** (ThreadPoolExecutor's maximum pool size).
    b. **async.thread.pool.core.size=5** (ThreadPoolExecutor's core pool size).
    c. **async.thread.pool.queue.capacity=200000** (Capacity for the ThreadPoolExecutor's).
3. **Data batching:** Large audience data will be processed in chunk(Batch). Size of batch can be configured as below:
    a. **implicit.service.databatch.timeout = 15** (Waiting time in sec for batch. If batch size is not full till 15 sec, then batch will timeout and whatever records in batch will be processed).
    b. **implicit.service.databatch.batchSize = 100** (Max size of batch).
4. **journey.audience.batch.size** = 500 (Number of audience fetched from database and passed in batches of 500 to first node of journey for processing).
5. **batch.data.import.batch.size=5** (Number of audience records passed in batches of 5 when entrysource is of file type). Max value for this property is 5.
6. **engine.monitoring.time=900000** (Time period in ms after which engine will monitor the components required for engine i.e. Database, kafka and Ignite).
7. **hip.batch.size = 100** (Number of audience is passed to HIP for processing in batches of 2).

Audience Ingestion Hints

The following section explains the use of audience ingestion hints for optimizing audience entries into Journeys configured with audience deduplication setting. If the Journey is configured for audience deduplication, Journey engine needs to look up every incoming audience record to decide if it is a new audience or an existing one. As per the lookup result, it either inserts a new audience record or performs deduplication on existing record. This approach has an overhead of consulting the audience database for every single incoming message. Consequently, this slows down audience ingestion process, which leads to delayed Journey processing. With the intention to optimize audience deduplication process, Journey engine supports certain hints, which can be supplied along with the audience message.

Note: Following hints are ignored for the Journeys that have not been configured with deduplication setting. Furthermore, these hints are supported only for Kafka & REST entry sources.

Generally, the audience message for REST & Kafka entry sources would look like below JSON:

```
{
    "entrySourceCode": "ES-00000001",
    "entrySourceType": "KAFKA",
    "data": [
        {
            "email": "email1@domain.com",
            "mobile": "1122334455",
            "birthdate": "09 01 1985",
        },
        {
            "email": "email2@domain.com",
            "mobile": "5544332211",
            "birthdate": "01 09 1995",
        },
        {
            "email": "email3@domain.com",
            "mobile": "5566778899",
            "birthdate": "05 11 2000",
        }
    ]
}
```

This JSON can now carry an additional hint to convey existence of given audience(s). Based on the hint, Journey engine can skip the audience existence check, and directly go for new audience insertion or deduplication updates. Accordingly, above message can carry single hint common to all the audiences(3) included in the message.

For example,

```
{
    "entrySourceCode": "ES-00000001",
    "entrySourceType": "KAFKA",
    "__hints":{
        "action": "U" // I -> Insert, U -> Update
    },
    "data": [
        {
            "email": "email1@domain.com",
            "mobile": "1122334455",
            "birthdate": "09 01 1985",
        },
        {
            "email": "email2@domain.com",
            "mobile": "5544332211",
            "birthdate": "01 09 1995",
        },
        {
            "email": "email3@domain.com",
            "mobile": "5566778899",
            "birthdate": "05 11 2000",
        }
    ]
}
```

The ***action*** property inside the ***__hints*** object conveys the expected action for the audiences. The ***action*** property can take one the following values, and their impact is as mentioned below:

action property value	Impact on audience ingestion
I	Audience(s) will be considered completely new and will be blindly inserted into the audience database. **Caveat** – System will not check if the audiences are truly new or not. Hence, any erroneous hint may lead to the redundant record(s) of same audience if it is sent with "I" hint more than once.
U	Audience(s) will be deemed pre-existing, and system will update the existing records with newly received changes. **Caveat** – System will not check if there is really any existing audiences based on the deduplication fields. If no such audiences are present, then the updates will have no impact. System will not convert updates into new audience insertions.
null or missing	If the property value is set to null and/or if the hint itself is missing, then Journey engine will consult audience database to see if incoming audiences already exist or not. Accordingly, it will perform either audience insertion

action property value	Impact on audience ingestion
	or modification. In a nutshell, it will continue to run its normal course.

Like the common audience hint, above JSON message can alternatively carry separate hint for each audience. For example:

```
{
    "entrySourceCode": "ES-00000001",
    "entrySourceType": "KAFKA",
    "data": [
        {
            "__hints":{
                "action": "U" // I -> Insert, U -> Update
            },
            "email": "email1@domain.com",
            "mobile": "1122334455",
            "birthdate": "09 01 1985",
        },
        {
            "__hints":{
                "action": "I" // I -> Insert, U -> Update
            },
            "email": "email2@domain.com",
            "mobile": "5544332211",
            "birthdate": "01 09 1995",
        },
        {
            "email": "email3@domain.com",
            "mobile": "5566778899",
            "birthdate": "05 11 2000",
        }
    ]
}
```

The impact of this approach is exactly similar to the common hint approach explained earlier. For the above example, first audience will be modified with the new details, second will be inserted as a new audience, whereas for the third audience, Journey engine will first check its existence & work accordingly.

Additionally, combination of common & separate hints is also supported. For example,

```
{
    "entrySourceCode": "ES-00000001",
    "entrySourceType": "KAFKA",
    "__hints":{
        "action": "I" // I -> Insert, U -> Update
    },
    "data": [
        {
            "__hints":{
                "action": "U" // I -> Insert, U -> Update
            },
            "email": "email1@domain.com",
            "mobile": "1122334455",
            "birthdate": "09 01 1985",
        },
        {
            "email": "email2@domain.com",
            "mobile": "5544332211",
            "birthdate": "01 09 1995",
        },
        {
            "email": "email3@domain.com",
            "mobile": "5566778899",
            "birthdate": "05 11 2000",
        }
    ]
}
```

For the above example, common hint (I) will be applicable to the second & third audience records. Thereby, second & third audiences will be treated as new audiences & shall be inserted into the database. First audience however carries an overridden hint, accordingly to which Journey engine will perform deduplication update for it.