OX2OX Migration Framework Scheduler Technical Documentation for 2.1.0-rev20

2023-04-17

Copyright notice

©2023 by OX Software GmbH. All rights reserved. Open-Xchange and the Open-Xchange logo are trademarks or registered trademarks of OX Software GmbH. All other company and/or product names may be trademarks or registered trademarks of their owners. Information contained in this document is subject to change without notice.

Contents

1 General Information

1.1 Warnings

Warning
Custom configuration or template files are potentially not updated automatically. After the update, please always check for files with a **.dpkg-new** or **.rpmnew** suffix and merge the changes manually. Configuration file changes are listed in their own respective section below but don't include changes to template files. For details about all the configuration files and templates shipped as part of this delivery, please read the relevant section of each package.

1.2 Delivery Comment

This delivery was requested with following comment:

> *OMF Scheduler 2.1.0 Maintenance Delivery 20*

1.3 Install Package Repository

This delivery is part of a restricted software repository:

https://software.open-xchange.com/components/omf-scheduler/stable/2.1.0/RHEL7
https://software.open-xchange.com/components/omf-scheduler/stable/2.1.0/DebianStretch
https://software.open-xchange.com/components/omf-scheduler/stable/2.1.0/DebianBuster
https://software.open-xchange.com/components/omf-scheduler/stable/2.1.0/DebianBullseye

1.4 Build Dependencies

This delivery was build and tested with following dependencies:

```
RedHat:rhel-7,Debian:Stretch,Debian:Buster,
Debian:Bullseye
```

2 Shipped Version

2.1 Package open-xchange-omf-orchestrator

OMF Orchestrator CLI to interoperate with the OX2OX Migration Framework.
Version: 2.1.0-20
Type: Other

2.1.1 Installation

Install on nodes with package installer **apt-get** or **yum**:

```
<package installer> install open-xchange-omf-orchestrator
```

2.1.2 Configuration

For details, please see appendix A
/opt/open-xchange/omf/orchestrator/etc/omf-orchestrator.yml (page 5)

2.2 Package open-xchange-omf-scheduler

OMF Scheduler OX2OX Migration Framework Scheduler.
Version: 2.1.0-20
Type: Other

2.2.1 Installation

Install on nodes with package installer **apt-get** or **yum**:

```
<package installer> install open-xchange-omf-scheduler
```

2.2.2 Configuration

For details, please see appendix A
/opt/open-xchange/omf/scheduler/etc/omf-scheduler.yml (page 10)

A Configuration Files

File 1 /opt/open-xchange/omf/orchestrator/etc/omf-orchestrator.yml

```
micronaut:
  application:
    name: omf
  http:
    services:
      # The OMF Scheduler HTTP REST API service configuration
      scheduler-admin-source:
        url: "${omf.scheduler.url}/omf/scheduler/admin/source/"
        read-timeout: ${omf.http.read-timeout}
        connect-timeout: ${omf.http.connect-timeout}
        ssl:
          enabled: ${omf.http.ssl.enabled}
          trust-store:
            path: ${omf.http.ssl.trust-store.path}
            password: ${omf.http.ssl.trust-store.password}
            type: ${omf.http.ssl.trust-store.type}

      scheduler-admin-target:
        url: "${omf.scheduler.url}/omf/scheduler/admin/target/"
        read-timeout: ${omf.http.read-timeout}
        connect-timeout: ${omf.http.connect-timeout}
        ssl:
          enabled: ${omf.http.ssl.enabled}
          trust-store:
            path: ${omf.http.ssl.trust-store.path}
            password: ${omf.http.ssl.trust-store.password}
            type: ${omf.http.ssl.trust-store.type}

      # The OMF Scheduler Migration HTTP REST API service configuration
      scheduler-migration:
        url: "${omf.scheduler.url}/omf/scheduler/migration/"
        read-timeout: ${omf.http.read-timeout}
        connect-timeout: ${omf.http.connect-timeout}
        ssl:
          enabled: ${omf.http.ssl.enabled}
          trust-store:
            path: ${omf.http.ssl.trust-store.path}
            password: ${omf.http.ssl.trust-store.password}
            type: ${omf.http.ssl.trust-store.type}

      scheduler-batch:
        url: "${omf.scheduler.url}/omf/scheduler/batch/"
        read-timeout: ${omf.http.read-timeout}
```

```
        connect-timeout: ${omf.http.connect-timeout}
        ssl:
          enabled: ${omf.http.ssl.enabled}
          trust-store:
            path: ${omf.http.ssl.trust-store.path}
            password: ${omf.http.ssl.trust-store.password}
            type: ${omf.http.ssl.trust-store.type}

      # The OMF Scheduler Migration HTTP REST API service configuration
      scheduler-userinfo:
        url: "${omf.scheduler.url}/omf/scheduler/user/"
        read-timeout: ${omf.http.read-timeout}
        connect-timeout: ${omf.http.connect-timeout}
        ssl:
          enabled: ${omf.http.ssl.enabled}
          trust-store:
            path: ${omf.http.ssl.trust-store.path}
            password: ${omf.http.ssl.trust-store.password}
            type: ${omf.http.ssl.trust-store.type}

      # The OMF Scheduler Monitoring HTTP REST API service configuration
      scheduler-monitoring:
        url: "${omf.scheduler.url}/ws/omf/scheduler/workers/monitor/"
        read-timeout: ${omf.http.read-timeout}
        connect-timeout: ${omf.http.connect-timeout}
        ssl:
          enabled: ${omf.http.ssl.enabled}
          trust-store:
            path: ${omf.http.ssl.trust-store.path}
            password: ${omf.http.ssl.trust-store.password}
            type: ${omf.http.ssl.trust-store.type}

omf:
  http:
    read-timeout: 30s
    connect-timeout: 10s
    ssl:
      enabled: true
      # If the scheduler does not have a valid public certificate
      # (e.g. uses a self-signed certificate), then its certificate can be
      # configured here.
      trust-store:
        path: file:/opt/open-xchange/omf/certs/scheduler.p12
        password: secret
        type: PKCS12
  readonly: false
  shell:
    start.dir:
    config:
      user.dir: ${user.dir}/.omf/config
      app.dir: /opt/open-xchange/omf/lib/scripts
  scheduler:
    # Credentials for the scheduler
    # On multi-user systems, specifying the password in a configuration file
    # with proper file system permissions is preferred to specifying it on
    # the command line, since the command line is visible to all local users.
    #
    # Example:
    # username: admin
    # password: secret

    # Location of the scheduler. Only the protocol and host name need to be
    # specified.
    url: "https://localhost:8443"
  ui:
    color: true
    color.theme: DARK
    unicode: true
    expandIds: false
    tz: UTC
    showTz: false
    showAgo: false
```

```
    prettyJson: false
    highlightJson: false
    shell:
        prettyJson: true
        highlightJson: true
        fancyPrompt: true
        rightHandPrompt: true
    history.file: ${user.dir}/.omf_history

logger:
  levels:
    # change this to TRACE to see a detailed log of the HTTP traffic between the
    # Orchestrator and the Scheduler
    io.micronaut.http.client: INFO
```

File 2 /opt/open-xchange/omf/scheduler/etc/omf-scheduler.yml

```
# https://docs.micronaut.io/latest/guide/config.html#configurationProperties
---
micronaut:
  # SSL configuration
  # Required for production environments.
  # See https://docs.micronaut.io/latest/guide/index.html#https for details.
  ssl:
    enabled: true
    port: 8443
    key-store:
      path: file:/opt/open-xchange/omf/certs/keystore.p12
      type: PKCS12
      password: secret
  server:
    dual-protocol: true
    port: 8080

  http:
    services:
      # The omf-source service is use to collect health status and metrics from the Source
      # OMF nodes. The HTTP client can be configured (ex: ssl) here by referencing
      # https://docs.micronaut.io/latest/guide/configurationreference.html#io.micronaut.
          http.client.ServiceHttpClientConfiguration
      # and the subsequent sections related to micronaut.http.services.*
      omf-source:
        connect-timeout: 30s
        read-timeout: 120s
        # Example SSL configuration in case a source uses a private certificate
        # ssl:
        #   trust-store:
        #     path: file:/opt/open-xchange/omf/certs/source.p12
        #     type: PKCS12
        #     password: secret
      omf-target:
        connect-timeout: 30s
        read-timeout: 120s
  application:
    name: omf-scheduler
  # Configure security including basic auth: https://micronaut-projects.github.io/
      micronaut-security/latest/guide/#basicAuth
  # Must be set to true or the Source Controller is not secure
  security:
    enabled: true
    # Change the security of the open api views to anonymous so that they can be viewed
        without credentials
    intercept-url-map:
      - pattern: /swagger/**
        access:
          - isAnonymous()
      - pattern: /swagger-ui/**
        access:
```

```
          - isAnonymous()
      - pattern: /rapidoc/**
        access:
          - isAnonymous()
      - pattern: /redoc/**
        access:
          - isAnonymous()
  # https://docs.micronaut.io/latest/guide/index.html#_configuring_caches
  #caches:
    #example:
      #charset: UTF-8
      #expire-after-access: 1h
  metrics:
    enabled: true
    export:
      # Creates an endpoint like http://host/prometheus - uses basic auth from
      # credentials under scheduler.http.admin
      prometheus:
        enabled: true
        step: PT1M
        descriptions: true
  router:
    # Adds api versioning: https://docs.micronaut.io/latest/guide/index.html#apiVersioning
    versioning:
      enabled: true
      parameter:
        enabled: true
        names: 'v'
      header:
        enabled: true
        names: 'X-API-VERSION'
    # Allows the openapi views to be seen
    static-resources:
      swagger:
        paths: classpath:META-INF/swagger
        mapping: /swagger/**
      redoc:
        paths: classpath:META-INF/swagger/views/redoc
        mapping: /redoc/**
      rapidoc:
        paths: classpath:META-INF/swagger/views/rapidoc
        mapping: /rapidoc/**
      swagger-ui:
        paths: classpath:META-INF/swagger/views/swagger-ui
        mapping: /swagger-ui/**
---
scheduler:
  # Must be unique for each Scheduler instance.
  # Be very careful when changing this value, as it is also used to determine the transactional ID for writing
  # batches into Kafka topics for each Scheduler node.
  id: 'scheduler-0'
  hostname: ''
  window:
    # If more than this many batches are created as part of a new Window, don't return the list of
    # Batch IDs as part of the Window creation result DTO, as they would just be too many to display
    # in the first place (also affects the debug logging):
    batch.id.threshold: 50
    # If more than this many context IDs are assigned as part of a new Window, don't return the list of
    # context IDs as part of the Window creation result DTO, as they would just be too many to display
    # in the first place:
    context.id.threshold: 50
  batch:
    presync:
      # Max number of contexts in a batch
      size: 10
      # Strategy to use when creating batches.
      # Current supported strategies:
```

```
      #    - fill-first: create batches up to the batch size then create the next batch
      #    - fill-equal: create batches of equal size
      strategy: fill-equal
    cutover:
      size: 10
      strategy: fill-equal
    preprovisioning:
      size: 10
      strategy: fill-equal
  kafka:
    enabled: true
    wait: false
    queues:
      batch: "omf-batch"
      response: "omf-response"
    resize:
      batch: true
      response: true
    record.header.enhance: true
    topic:
      list:
        timeout: -1s
      describe:
        timeout: -1s
        retry:
          attempts: 3
          wait: 3s
      create:
        # whether topics should be created whan a source is created or synced (true) or
        # whether we make use of auto-creation instead (old behaviour prior to 2.1.0-6):
        enabled: true
        timeout: -1s
        partitions: 2
        replication.factor: 0
        config:
          retention.ms: 432000000
    rest:
      lastNextBatches: 10
  workers:
    allow:
      # allow Worker shutdown via the REST API
      shutdown: false
      # allow overriding the Sources Workers subscribe to via the REST API
      changeSources: false
  http:
    users:
       # Users defined here can have roles OMF_ADMIN or OMF_USER. Users without a role
           automatically have role OMF_USER
       # assigned. Users with the role OMF_ADMIN have access to every REST API method.
           OMF_USER role is restricted to
       # a subset of the REST API.
       admin:
         password: secret
         role: OMF_ADMIN
       omfuser:
         password: secret
         role: OMF_USER
    workers:
      # a list of valid tokens for workers to use
      # can generate some with `pwgen -n1 64 1`
      - 'bei9Ohchie8nai5em5asee9wohz6uu0ahshaigh0bia1isi4liKi0iwo8bu2niey'
      - 'eeW5moi6eleik0ziw7ivaen3phoi6oolae9aht2ox9uY0ebiVaht0gashoof1rai'
      - 'oogheePhaeB5iezairu6ongee8Ee6faePashi9thietahG0bieghiixeivahroco'
    admin.controller.path: /omf/scheduler/admin
    migration.controller.path: /omf/scheduler/migration
    user.controller.path: /omf/scheduler/user
    batch.controller.path: /omf/scheduler/batch
  metrics:
    migration:
      enabled: true
    leadership:
      enabled: true
```

```
    batchresponse:
      enabled: true
      percentiles: true
    schedulers:
      enabled: true
    source:
      enabled: true
    target:
      enabled: true
    monitor:
      windows:
        enabled: true
        interval: 5m
        delay: 30s
      batches:
        enabled: true
        interval: 5m
        delay: 30s
      sources:
        enabled: true
        interval: 5m
        delay: 30s
      targets:
        enabled: true
        interval: 5m
        delay: 30s
      contextmappings:
        enabled: true
        interval: 5m
        delay: 30s
      usermappings:
        enabled: true
        interval: 60m
        delay: 5m
      migrationevents:
        enabled: true
        interval: 5m
        delay: 30s
    workers:
      enabled: true
      idle.since: [5m, 10m, 30m]
  orphan-check:
    context.batch.size: 50
  database:
    migration:
      allowRead: false
      allowWrite: false
    scheduler:
      allowRead: false
      allowWrite: false
      useSkipLocked: false
  batchSkipList:
    reap:
      windows: true
      batches: true
---
jackson:
  bean-introspection-module: true
  serialization:
    indent-output: true
    writeDatesAsTimestamps: false
---
datasources:
  # Used to persist scheduling data
  scheduler:
    # url should use createDatabaseIfNotExist=true if the database will not
    # already exist: https://dev.mysql.com/doc/connector-j/8.0/en/connector-j-reference-
        configuration-properties.html
    url: jdbc:mysql://localhost:3306/scheduler?createDatabaseIfNotExist=true
    username: root
    password: my-secret-pw
    driverClassName: org.mariadb.jdbc.Driver
```

```
    dialect: MYSQL
    maximumPoolSize: 10
    maxLifetime: 180000
  # Used to create/drop databases for sources. This is not really the "default" data source
  # but we need to use default because of bug https://github.com/micronaut-projects/micronaut-data/issues/598s
  default:
    url: jdbc:mysql://localhost:3306/
    username: root
    password: my-secret-pw
    driverClassName: org.mariadb.jdbc.Driver
    dialect: MYSQL
    maximumPoolSize: 5
    maxLifetime: 180000
---
endpoints:
  loggers:
    enabled: true
    sensitive: true
  health:
    discovery-client:
      enabled: false
    sources:
      enabled: false
    targets:
      enabled: false
  liquibase:
    # fails with missing transition, might be fixed in later Micronaut releases
    enabled: false
  info:
    enabled: true
    sensitive: true
    sourceCodeOrigin:
      enabled: true
      location: file:/opt/open-xchange/omf/scheduler/share/SourceCodeOrigin.txt
---
zookeeper:
  server: zookeeper:2181
  blockUntilConnected: true
  maxConnectedWaitTime: 30s
  sessionTimeout: 1m
  connectionTimeout: 15s
  maxCloseWait: 15s
  waitForShutdownTimeout: 15s
  connectionRetry:
    baseSleepTime: 5s
    maxSleepTime: 30s
    maxRetries: 50
---
kafka:
  bootstrap:
    servers: kafka-1:9092, kafka-2:9092, kafka-3:9092
  producers:
    batch-producer:
      enable.idempotence: true
      # This enables transactions for the Batch Producer
      # The value must be unique per application, but should
      # not change for the same app after a crash, etc.
      # Note that if you only want to allow a single Scheduler instance to be capable of
      # writing Batches into the Kafka topics, then change this to be the same value across
      # all Scheduler instances as Kafka will fence them (see PRODUCER_FENCED).
      # But if you want all Scheduler instances to be able to write batches into Kafka topics,
      # use a unique value for each Scheduler instance:
      transactional.id: ${scheduler.id}
  #consumers:
    #response-consumer:
---
mail:
  # whether to send emails
```

```
  enabled: false
  window:
    # whether to send emails when a Window succeeds:
    success: false
    # whether to send emails when a Window fails:
    failure: true
  # mandatory, must be set to be able to send emails and it
  # must be a valid email address in the form localpart@domain, or sending will fail:
  from: ${scheduler.id}@example.com
  # whom to send those mails to (can be a comma separated list):
  to:
  cc:
  bcc:
  # text to include in the subject line, wrapped in []:
  subject.id:
  smtp:
    host: localhost
    port: 25
    # leave empty for no authentication:
    username:
    password:
    # SMTP, SMTPS or SMTP_TLS (SMTP with mandatory StartTLS):
    transport: SMTP
    # whether to allow SMTP without StartTLS:
    smtp.plain: true
    tls:
      # whether to trust all SMTP server keys
      trustall: false
      # whether to verify SMTP server keys
      verify: true
---
liquibase:
  datasources:
    scheduler:
      change-log: 'classpath:liquibase/scheduler/liquibase-changelog.xml'
---
logger:
  levels:
    ROOT: INFO
    com.openxchange: INFO
    omf: INFO
    omf.scheduler.admin.AuthenticationProviderUserPassword: WARN
    org.apache.kafka.clients.consumer.ConsumerConfig: WARN
    omf.scheduler.security: INFO
---
```