s8.capital | Акционерное общество «С8 Капитал» | Волгоградский пр-т, д. 43, корп.3, Москва, 109316 | Тел.: +7 (495) 204-88-88 Факс: +7 (495) 204-88-87

Программное обеспечение «Меркурий»

Руководство для установки и эксплуатации программного обеспечения

Оглавление

1. Инфраструктура для ПО «Меркурий»

1. Хост с сервисом MariaDB (для хранения данных об играх баркодах, пачках, операциям с баркодами и пачками, терминалах и разрешениях для терминалов - далее db_mercury) версия MariaDB 10.4.14
2. Хост с сервисом MariaDB (для хранения транзакционной информации - далее db_trns) версия MariaDB 10.4.14

3. Хост с сервисом MariaDB (для хранения репликационной информации - далее db_rep) версия MariaDB 10.4.14
4. Хост с сервисом MariaDB (для передачи репликационной информации - далее db_bus) версия MariaDB 10.4.14
5. Хост с сервисом MariaDB (для хранения авторизационной информации для агентов - далее localdb0) версия MariaDB 10.4.14
6. Система оркестрации Kubernetes (для запуска модулей службы) версия Kubernetes v1.20.15
   версия Ingress https://github.com/kubernetes/ingress-nginx/releases/tag/ingress-nginx-2.11.1
7. Система хранения образов Harbor (registry)
8. Хост с установленной ОС Centos 7.0 (или аналогичной, поддерживающей rpm пакеты) - для модуля загрузки информации о баркодах и пачках в db_mercury
9. Хост с установленным брокером сообщений
10. Внешняя система с хранением данных о терминалах на базе СУБД Oracle (модуль mercury-sync)
11. Внешняя система идентификации (модуль transaction-manager при выплате свыше 15000 рублей)
12. Внешняя система экспорта репликационных данных на базе СУБД Oracle (или иная система вычитки транзакционных данных)
13. Внешняя система учета финансового баланса терминалов (модуль fin-server)

2. Описание модулей ПО «Меркурий»

Модуль esb_mercury_reload

Осуществляет корректирующие выгрузки на основании созданных задач**.**

Конфигурация:

```
daemon:
 package_size: 256
 thread_count: 1
 timeout_milliseconds: 6000
 repeat_reset_timeout_seconds: 256 # ()
 queue_task_timeout_seconds: 256 # ()
 limit_interval: 1000
 level: 32
```

```
db_mercury:
 db_host: "db_mercury_host"
 db_name: "db_mercury"
 db_user: "mercury_usr"
 db_pass: "$ENV_databasePasswordMercuryUsr"
 db_port: 3306

db_bus:
 db_host: "db_bus_host"
 db_name: "db_bus"
 db_user: "bus_writer"
 db_pass: "$ENV_databasePasswordBusWriter"
 db_port: 3306

log:
 level: "trace"
 output: "console"
 backtrace_buffer_size: 20
```

Модуль fcgi_mercury

Точка доступа к API mercury.xsd для внешних агентов интеграции.

Конфигурация:

```
[DEBUG]
LEVEL = 99 // VERY_FULL_DEBUG - for network profiling

[LOG]
LEVEL = "trace"
OUTPUT = "console"
BACKTRACE_BUFFER_SIZE = 20

[SCHEMA]
SCHEMA_PATH = "/var/www/webchannel/xsd/mercury.xsd"

[MODULE]
ID_KERN = 12
ID_PROV = 9002
CHECK_SIGNATURE_FLAG = 0
USE_METRICS=0

[metrics]
mhost= "prometheus-pushgateway"
mport=9091
mnamespace="tfi-mercury"

[SERVER]
```

```
REMOTE_ADDRESS = "tfi-transaction-manager-svc"
REMOTE_PORT = 7777
DEFAULT_TIMEOUT_RECEIVE = 20
DEFAULT_TIMEOUT_CONNECT = 5 //    (, )
DEFAULT_TIMEOUT_SEND = 20 //                (, )

[AGENTS]

TRUST_TERMINAL_COUNT = 1
TRUST_TERMINALS[0] = 2000000001

NUMBER = 0

[GAME_IDS]
NUMBER = 0
GATE_GAME_ID [0] = 30510
AGENT_GAME_ID [0] = 510

[ERRORS]
NUMBER = 0
DB_ERROR_VAL [0] = 0
DB_ERROR_DESC [0] = " "
PROTO_SIGN_VAL [0] = 0
```

Модуль fcgi_mercury_pos

Точка доступа к API mercury.xsd для терминалов Wave, S3.

Конфигурация как **fcgi_mercury**

Модуль fcgi_mercury_signature

Точка доступа к API mercury.xsd для внешних агентов интеграции.

Конфигурация как **fcgi_mercury**

Модуль mercury_sync

Обеспечивает одностороннюю синхронизацию данных в направлении от СУБД Oracle в db_mercury.

Конфигурация:

```
daemon:
 sql_error_limit: 10
 sync_interval_ms: 5000
 max_trans_duration_ms: 1000000
 debug: 99

db_mercury:
 host: "db_mercury_host"
 name: db_mercury
 user: mercury_dwh
 pass: "$ENV_databasePasswordMercuryDwh"

oracle:
 host: "$ENV_databaseHostOracle"
 name: DWHDEV
 user: "USR_GATE"
 pass: "$ENV_databasePasswordFromOracle"

metrics:
 mhost: 0.0.0.0
 mport: 3000

log:
 level: "trace"
 output: "console"
 backtrace_buffer_size: 64
```

Модуль mercuryos

Модуль фиксации изменений в db_mercury

Конфигурация:

```
log:
 level: trace
 output: "console"
 backtrace_buffer_size: 20

daemon:
 worker_count: 2
 level: 50
 reset_queue_timeout: 100
 life_task_timeout: 60

server:
 host: 0.0.0.0
 port: 7782
```

```
metrics:
 mhost: 0.0.0.0
 mport: 3000

database:
 tracer_on: true
 host: "db_mercury_host"
 name: db_mercury
 user: mercury_usr
 pass: "$ENV_databasePasswordMercuryUsr"
```

Модуль mercuryos_compensate

Модуль компенсации фиксации изменений в db_mercury в случае ошибки

Конфигурация:

```
daemon:
 level: 30

 receive_timeout_msec: 1000
 request_queue:
  name: "Mercuryos.Request"
  max_redelivery_count: 5
  worker_count: 1
 reply_queue: "MERCURY.COMPENSATION_RESPONSE"

message_broker:
 url: "failover://(tcp:/message_broker_host)?
startupMaxReconnectAttempts=2&maxReconnectAttempts=2&maxReconnectDelay=5000&tim
eout=3000"
 user: "compensator"
 pass: "$ENV_brokerPasswordCompensator"

db_mercury:
 db_host: "db_mercury_host"
 db_name: db_mercury
 db_user: "mercuryos_compensate"
 db_pass: "$ENV_databasePasswordMercuryosCompensate"

log:
 level: "trace"
 output: "console"
 backtrace_buffer_size: 20

metrics:
 mhost: 0.0.0.0
```

```
 mport: 3000

archiving:
 cmps_arch_interval: 3600

cleaning:
 cmps_clear_interval: 3600
```

Модуль transaction_compensator

Модуль компенсации фиксации изменений в db_trns в случае ошибки.

Конфигурация:

```
log:
 level: "trace"
 output: "console"
 backtrace_buffer_size: 20

metrics:
 mhost: 0.0.0.0
 mport: 3000

daemon:
 level: 30
 task_creator_workers: 1
 task_result_checker_workers: 1

task_creator:
 request_queue: "MERCURY.COMPENSATION_REQUEST"
 max_redelivery_count: 5
 task_check_sleep_ms: 10000

task_result_handler:
 reply_queue: "MERCURY.COMPENSATION_RESPONSE"
 receive_timeout_ms: 1000

kernel_info: # select id, name from kernel; MERC-56
 mercuryos:
 - 3
 pos:
 - 6
 ortax:
 - 9
 bonus:
 - 10
```

```
message_broker:
 url: "failover:///(tcp:/message_broker_host)?
startupMaxReconnectAttempts=2&maxReconnectAttempts=2&maxReconnectDelay=5000&tim
eout=3000"
 user: "compensate_processor"
 pass: "$ENV_brokerPasswordCompensateProcessor"

db_trns:
 host: "db_trns_host"
 name: "db_trns"
 user: "trns_user"
 pass: "$ENV_databasePasswordTrnsUser"
 min_connections: 1
 max_connections: 2
 tracer: false
"urlSrv": "http://vps.website.ru/api",
"urlRng": "http://vps.website.ru/api",
"showDevActions": false,
"showTestActions": false
```

Модуль transaction_fixer

Модуль осуществляет поиск транзакций, обработка которых завершилась модулем transaction_manager, но результат не зафиксирован.

Конфигурация:

```
log:
 level: "trace"
 output: "console"
 backtrace_buffer_size: 20
daemon:
 worker_count: 1
 level: 90
 reset_queue_timeout: 100
 life_task_timeout: 60
 commit_timeout: 10

database:
 host: "db_trns_host"
 port: 3306
 name: "db_trns"
 user: "trns_user"
 pass: "$ENV_databasePasswordTrnsUser"
```

Модуль transaction_manager

Модуль обеспечивает транзакционность обработки запросов на основании планов исполнения, формирует задания компенсаций.

Конфигурация:

```
log:
 level: "trace"
 output: "console"
 backtrace_buffer_size: 20

metrics:
 mhost: 0.0.0.0
 mport: 3000

handler:
 providers:
 - 9001
 - 9002
 - 9003

kernels:
 io_pool_size: 16
 connect_timeout_sec: 5
 read_timeout_sec: 5
 write_timeout_sec: 5
 connection_count: 4

server:
 io_pool_size: 64
 host: "0.0.0.0"
 port: 7777
 connection_limit: 300
 watch_rate: 6000

db_trns:
 host: "db_trns_host"
 name: "db_trns"
 user: "trns_user"
 pass: "$ENV_databasePasswordTrnsUser"
 min_connections: 8
 max_connections: 32
 tracer: false

localdb0:
 host: "localdb0_host"
 name: "localdb0"
 user: "galaxy_user"
 pass: "$ENV_databasePasswordGalaxyUser"
 min_connections: 8
```

```
 max_connections: 16
 tracer: false

ortax:
 url: "https://external_identification_system/fprov/fcgi_ortax"
 connect_timeout_sec: 5
 request_timeout_sec: 3
 basic_auth_user: "test_user"
 basic_auth_pass: "test_pass"
 ca_cert_path: "/etc/orglot/ssl/ca-bundle.trust.crt"
 client_cert_path: "/etc/orglot/ssl/trn-ortax-client.pem"
 client_key_path: "/etc/orglot/ssl/trn-ortax-client.key.pem"
 client_key_pass: "$ENV_ortaxClientKeyPass"
 terminal_id: 12345
 kernel_id: 9
 default_tax_limit: 1500000
 default_ident_limit: 1500000
 detailed_taxes_flag: false
 default_invalidation_rate: 12345677
```

Модуль trns_dwh_export

Модуль обеспечивает репликацию данных из буферной базы db_rep.

Конфигурация:

```
daemon:
 reset_queue_timeout: 1000 # ( )
 comment: ResetQueueWorker
 queue_task_timeout: 5000 #
 check_interval: 1000 # ( )
 worker_count: 1 # 2 for full test
 pack_size: 1000 #
 level: 256

db_rep:
 host: "db_bus_host"
 name: db_rep
 user: rep_ora
 pass: "$ENV_databasePasswordRepOra"

db_dwh:
 host: "$ENV_databaseHostOracle"
 name: DWHDEV
 user: GTW_LOGS
 pass: "$ENV_databasePasswordToOracle"

 schema: ""
```

```
log:
 level: "trace"
 output: "console"
 backtrace_buffer_size: 20
```

Модуль trns_esb_cntrl

Модуль обеспечивает формирование еженедельных отчетов по транзакциям за неделю.

Конфигурация:

```
log:
 level: "trace"
 output: "console"
 backtrace_buffer_size: 20

daemon:
 level: 29
 start_wday: 1
 start_time: "10:00:00"
 trns_count_limit: 100

db_rep:
 host: "db_rep_host"
 name: "db_rep"
 user: "rep_esb_writer"
 pass: "$ENV_databasePasswordRepEsbWriter"

 tracer: false
db_bus:
 host: "db_bus_host"
 name: "db_bus"
 user: "bus_writer"
 pass: "$ENV_databasePasswordBusWriter"
 tracer: false
```

Модуль trns_esb_corrector

Модуль обеспечивает корректирующие выгрузки на основании созданных задач.

Конфигурация:

```
daemon:
 package_size: 1
 thread_count: 1
 timeout_milliseconds: 6000
```

```
 limit_interval: 20001
 level: 99
 min_trns_id: 2000000000000

db_rep:
 db_host: "db_rep_host"
 db_name: "db_rep"
 db_user: "rep_user"
 db_pass: "$ENV_databasePasswordRepUser"

db_bus:
 db_host: "db_bus_host"
 db_name: "db_bus"
 db_user: "bus_writer"
 db_pass: "$ENV_databasePasswordBusWriter"

log:
 level: "trace"
 output: "console"
 backtrace_buffer_size: 20
```

Модуль trns_ifd_export

Модуль обеспечивает репликацию данных из буферной базы db_rep.

Конфигурация:

```
daemon:
reset_queue_timeout: 1000
queue_task_timeout: 5000
check_interval: 1000
worker_count: 1
pack_size: 1000
level: 20

db_rep:
host: "db_bus_host"
name: "db_rep"
user: "rep_ora"
pass: "$ENV_databasePasswordRepOra"

db_bus:
host: "db_bus_host"
name: "db_bus"
user: "bus_writer"
pass: "$ENV_databasePasswordBusWriter"

log:
```

```
level: "trace"
output: "console"
backtrace_buffer_size: 20
```

Модуль trns-parser

Модуль обеспечивает репликацию данных из операционной базы db_trns в db_rep , создает задания на репликацию для модулей trns_dwh_export , trns_ifd_export.

Конфигурация:

```
log:
 level: "trace"
 output: "console"
 backtrace_buffer_size: 64

daemon:
 level: 99
 record_pack_size: 10
 trn_check_interval: 2000
 reset_timeout: 20
 queue_interval: 5
 worker_count: 1

db_trns:
 host: "db_trns_host"
 name: "db_trns"
 user: "trns_parser"
 pass: "$ENV_databasePasswordTrnsParser"

db_rep:
 host: "db_bus_host"
 name: "db_rep"
 user: "rep_writer"
 pass: "$ENV_databasePasswordRepWriter"

metrics:
 mhost: 0.0.0.0
 mport: 3000
```

Модуль trns-cleaner

Модуль обеспечивает очистку db_trns.

Конфигурация:

```
log:
 level: "debug"
 output: "console"
 backtrace_buffer_size: 20

db_trns:
 db_host: "db_trns_host"
 db_name: "db_trns"
 db_user: "db_trns_cleaner_user"
 db_pass: "$ENV_databasePasswordTrnsCleaner"

cleaning_settings:
 week_day: 7
 time: 23:00:00
 package_size: 100000
 workers_count: 1
 clean_period: 30
```

Ingress

Сущность в Kubernetes обеспечивающая маршрутизацию внешнего HTTP и HTTPS трафика кластера Kubernetes среди модулей fcgi-mercury* внутри кластера.

3. Установка ПО «Меркурий»

1. Для **каждого** из модулей необходимо осуществить перенос артефактов в локальный registry

```
$ sudo docker image load --input module_name.tar.gz
$ sudo docker image tag original-registry.example.org/module_name:release-1.777 new-registry.example.com/module_name:release-1.777
$ sudo docker push new-registry.example.com/module_name
```

2. Создание namespace

```
$ kubectl create namespace mercury
```

3. Создание секрета для доступа к registry

```
$ kubectl create secret generic regcred --from-file=.dockerconfigjson=/tmp/config.json --type=kubernetes.io/dockerconfigjson --kubeconfig ~/.kube/kubeconfig -namespace mercury
$ cat /tmp/config.json
{"auths":{"new-registry.example.com":{"username":"registry_user","password":"password","auth":"cmVnaXN0cnlfdXNlcjpwYXNzd29yZAo="}}}
```

4. Для каждого из модулей необходимо осуществить установку с помощью утилиты helm, подставив актуальные значения в конфигурационные файлы.

$ helm upgrade --install module_name module_name -f module_name/values.yaml --kubeconfig ~/.kube/kubeconfig --namespace mercury

Также установка допустима любым иным, валидным для системы оркестрации kubernetes способом.

4. Выключение/включение модулей

При необходимости можно выключать не все модули как в примере, а помодульно. Для этого надо вместо аргумента --all указать имя требуемого модуля.

```
#
kubectl scale --replicas=0 deployment --all --kubeconfig ~/.kube/kubeconfig --namespace mercury
#
kubectl scale --replicas=1 deployment --all --kubeconfig ~/.kube/kubeconfig --namespace mercury
```