**数据字典设计**

[1. 数据字典 4](#_Toc516044327)

[1.1 OMC 数据字典 4](#_Toc516044328)

[1.1.1 omc 配置数据表omc\_omcconfig 4](#_Toc516044329)

[1.1.2 网元配置表omc\_neconfig 8](#_Toc516044330)

[1.1.3告警配置表omc\_alarminfo 8](#_Toc516044331)

[1.1.4告警数据表omc\_alarmrec 9](#_Toc516044332)

[1.1.5 性能测量数据表omc\_measurekpi 10](#_Toc516044333)

[1.1.6 告警过滤数据表omc\_alarmfilter / 告警抑制数据表omc\_alarmrestrain 11](#_Toc516044334)

[1.1.7 系统操作日志omc\_logrec 11](#_Toc516044335)

[1.1.8 性能数据omc\_performancedata 12](#_Toc516044336)

[1.1.9 用户数据omc\_user 12](#_Toc516044337)

[1.1.10 心跳告警配置omc\_heartbeat 12](#_Toc516044338)

[1.1.11 越限告警配置omc\_vallimit 13](#_Toc516044339)

[1.1.12 范围告警配置omc\_scopelimit 13](#_Toc516044340)

[1.1.13 表定义表omc\_propertydef 13](#_Toc516044341)

[1.2 HA 数据字典 15](#_Toc516044342)

[1.2.1结点上服务进程配置ha\_appconfig 15](#_Toc516044343)

[1.3 VXD 16](#_Toc516044344)

[1.3.1 VXD虚拟终端遥信配置表 omc\_vxd\_rt\_yx 16](#_Toc516044345)

[1.3.2 VXD虚拟终端遥测配置表 omc\_vxd\_rt\_yc 16](#_Toc516044346)

[1.3.3 VXD虚拟终端遥文配置表 omc\_vxd\_rt\_yw 17](#_Toc516044347)

[1.4 RT 17](#_Toc516044348)

[1.4.1 三遥信息订阅表 omc\_rtsubscribe 17](#_Toc516044349)

[1.5 用户权限 17](#_Toc516044350)

[1.5.1 用户表 user 17](#_Toc516044351)

[1.5.2 用户组表 group 18](#_Toc516044352)

[1.5.3 用户用户组关联表 user\_group 18](#_Toc516044353)

[1.5.4 权限表 authority 18](#_Toc516044354)

[1.5.5 用户组权限关联表 group\_authority 18](#_Toc516044355)

[2. 配置文件 19](#_Toc516044356)

[2.1 全系统配置文件 19](#_Toc516044357)

[2.2 RT 配置文件 19](#_Toc516044358)

[2.2.1 表模式文件 19](#_Toc516044359)

[2.2.2 列模式文件 20](#_Toc516044360)

[2.3 数据库中的初始配置 22](#_Toc516044361)

[2.3.1 缺省网元配置 22](#_Toc516044362)

[2.3.2 缺省告警配置 22](#_Toc516044363)

[2.3.3缺省系统设置 24](#_Toc516044364)

[2.3.4缺省进程配置 26](#_Toc516044365)

[2.3.5 缺省属性配置 27](#_Toc516044366)

[3.通信包 30](#_Toc516044367)

[3.1 通用描述 30](#_Toc516044368)

[3.1.1请求包通用描述 30](#_Toc516044369)

[3.1.2 退出包 31](#_Toc516044370)

[3.2 OMC 协议包 31](#_Toc516044371)

[3.2.1命令码 31](#_Toc516044372)

[3.2.2登录 / 心跳 31](#_Toc516044373)

[3.2.3数据更新 32](#_Toc516044374)

[3.2.4告警设置 / 告警恢复 / 告警确认 / 告警取消确认 / 告警消除 32](#_Toc516044375)

[3.2.5 登出 33](#_Toc516044376)

[3.3 HA 协议包 33](#_Toc516044377)

[3.3.1 命令码 33](#_Toc516044378)

[3.3.2获取 OS 状态 33](#_Toc516044379)

[3.3.3获取 App 状态 34](#_Toc516044380)

[3.4 RT 协议包 34](#_Toc516044381)

[3.4.1命令码 34](#_Toc516044382)

[3.4.2 数据包新增参考代码 36](#_Toc516044383)

[3.4.3 数据包更新参考代码 36](#_Toc516044384)

[3.5 DA 协议包 37](#_Toc516044385)

[3.5.1 命令码 37](#_Toc516044386)

[3.5.2时间点数据获取 38](#_Toc516044387)

[3.5.3时间段数据获取 38](#_Toc516044388)

# 1. 数据字典

## 1.1 OMC 数据字典

以下数据字典的说明，只针对了数据表、数据字段，以及数据字段的含义说明。因为不同数据库采用了不完全相同的表定义，具体定义参看 \*.sql 脚本；具体的代码编写可以以Omc\_mysql.sql为数据样本，代码编写过程中，担当应注意不同类型的数据库的兼容问题。目前中心数据库要求同时支持 Oracle / Mysql。

### 1.1.1 omc 配置数据表omc\_omcconfig

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| ItemNo | 整型 | 配置项 ID，从1开始顺序增加 |
| ItemValue | 字符串 | 配置项对应的值 |

详细说明：变量头文件　basic/Bas\_OmcCfg.h

|  |  |  |
| --- | --- | --- |
| **配置序号** | **配置项说明** | **配置值** |
| 1 | 通道控制端口号，omc客户端连接omc服务器所使用的端口  omc-server和 omc-client 之间的通信端口； | 6687 |
| 2 | App 连接omc服务器所使用的端口，用于传送告警  数据端口：omc-server和业务模块之间的通信端口，用于业务通知 / 获取告警状态；该端口对业务 app 透明，因为业务 app 通过OmcApi.h 中NotifyAlarm函数接口来设置/恢复/确认告警；  通过OmcApi.h 中GetAlarmInfo函数接口来获取告警状态； | 6688 |
| 3 | Omc 服务器地址 | 类似192.168.1.1，目前只能手动修改 |
| 4 | 历史告警存放天数，告警天数，允许保存多少天的告警 | 365 |
| 5 | 历史告警存放条数，告警容量，允许保存多少万条告警；（单位：万条，例如2万条，该值为2）； | 100000 |
| 6 | 历史日志存放天数，日志天数，允许保存多少天的告警； | 365 |
| 7 | 历史日志存放条数，日志容量，允许保存多少万条日志了； | 100000 |
| 8 | 历史性能数据存放天数，性能天数，允许保存多少天的性能数据； | 365 |
| 9 | 历史性能数据存放条数，性能容量，允许保存多少条性能数据； | 100000 |
| 4~9 | 以上门限数据为0，表示该项不设限；但是3个类型的数据，在保留天数 / 保留条数中必然具备其中一个条件，由客户端限制； |  |
| 10 | 自动备份的方式  0：表示无自动备份  1：按天备份  2：按周备份 | 1 |
| 11 | 自动备份的内容  空：表示全备份  数据表字符串：表示备份目标表  待自动备份的表名称，逗号分割，**头尾皆无分隔符，例如： AlarmRec, omcconfig,alarmconfig**  **为空，且有自动备份的情况下，表示全备份** | 空 |
| 12 | 自动备份的介质，自动备份的路径，例如：E:/Backup/ （linux 格式存储的路径字符串，末尾需要/，一定为 omc-server 的本地目录） | D:\\ics4000\\Backup\\ |
| 13 | 自动备份的时间：  按天备份的情况下，形式为 0xcc，其中 0xcc表示小时，取值 0x00~0x17；  按周备份的情况下，形式为 0xaabb ，其中0xaa表示周几，取值0x00~0x07；0xbb表示小时，取值 0x00~0x17； | 0 |
| 14 | 时差报警阀值，单位：秒 | 180 |
| 15 | 系统cpu报警阀值，单位：% | 75 |
| 16 | 系统mem报警阀值，单位：% | 75 |
| 17 | 进程cpu报警阀值，单位：% | 75 |
| 18 | 进程mem报警阀值，单位：M | 4096 |
| 19 | 服务器结点性能采集的时间间隔，单位：秒 | 600 |
| 20 | DB监视采集的时间间隔，单位：秒 | 600 |
| 101 | 外部日志报警阀值，单位：M | 1048576 |
| 102 | 外部日志报警阀值，单位：天 | 365 |
| 103 | 单一 app log 大小，单位：M | 1024 |
|  |  |  |
| 0x0100 | rt proxy 监听端口 | 6689 |
| 0x0101 | rt proxy 监听地址 | 127.0.0.1 |
| 0x0102 | Rt 同步本地端口 | 5661 |
| 0x0103 | Rt 同步本地地址 | 127.0.0.1 |
| 0x0104 | Rt同步远端端口 | 9102 |
| 0x0105 | Rt同步远端地址 | 127.0.0.1 |
| 0x0106 | Rt同步sourceID | 2305 |
| 0x0107 | Rt-proxy -> Rt-Bus（协议栈）端口 | 6695 |
| 0x0108 | Rt-proxy -> Rt-Bus（协议栈）地址 | 127.0.0.1 |
| 0x0109 | Rt-proxy -> Rt-Bus（协议栈）Source | 6695 |
| 0x010a | RtBus 本地udp端口 | 6696 |
| 0x010b | RT 切片保存的时间间隔（单位：秒） | 10 |
| 0x010c | RT 切片保存的总时间（单位：天） | 180（３个月） |
| 0x010d | RT 切片保存的总大小（单位：G） | 10 |
| 0x010e | DA 监听端口 | 6697 |
| 0x010f | DA 线程池个数 | 5 |
|  |  |  |
| 0x0200 | HA监听端口，允许连接客户端获取系统/服务运行数据 | 6690 |

**以下表格涉及设备/服务类型定义，告警类型的定义，通用定义如下：**

0x31 表示系统  
0x51 表示服务器结点  
-- 0x61 表示运行服务（服务器上的运行进程）

形式为 0xxxx  
0x02xx 表示服务的类型，告警播报的业务源  
0x7xxx 表示设备的类型，告警播报的业务源

业务网元定义：

0x0201 表示重车衡业务

0x0202 表示轻车衡业务

0x0203 表示采样机（皮带）

0x0204 表示汽车采样业务

0x0205 表示火车采样业务

0x0206 表示原煤传输业务

0x0207 表示制样机业务

0x0208 表示下料仓业务

0x0209 表示预备仓业务

0x020a 表示天平业务

0x020b 表示破碎机业务

0x0210 表示存样柜业务

0x0211 表示化验业务

0x0212 表示火车计量业务

0x0213 表示入炉采样业务

0x0214 表示火车识别业务

设备网元定义：

-- 0x71 运输线

0x7101 火车出入厂识别装置

0x7102 火车衡器

0x7103 火车采机识别装置

0x7104 火车翻车识别装置

0x7105 火车轻车识别装置

0x7106 火车翻车机

0x7107 火车采样机

0x7108 汽车无人值守装置

0x7109 汽车厂前无人值守装置

0x710a 汽车入厂无人值守装置

0x710b 汽车重车无人值守装置

0x710c 汽车采样无人值守装置

0x710d 汽车煤场无人值守装置

0x710e 汽车轻车无人值守装置

0x710f 汽车出厂无人值守装置

0x7110 汽车采样机

0x7111 船运采样机

-- 0x72 燃煤线

0x7201 入厂采样机

0x7202 入炉采样机

0x7203 输煤程控

0x7204 皮带秤

0x7205 斗轮机

0x7206 斗轮机定位装置

0x7207 煤场测温装置

-- 0x73 样品线

0x7301 原煤集样装置

0x7302 原煤封样装置

0x7303 原煤缓存装置

0x7304 原煤转运车

0x7305 原煤动力控制柜

0x7306 原煤卸料装置

0x7307 全自动制样机

0x7308 存样柜

0x7309 样瓶动力控制柜

0x730a 取样终端

0x730b 钥匙管理机

0x730c 在线全水仪

0x730d 弃料回收装置

0x730e 弃样车

0x730f 样瓶清洗装置

0x7310 原煤上料装置

-- 0x75 化验室

0x7501 温湿度装置

0x7502 电子天平

0x7503 全水仪

0x7504 工分仪

0x7505 量热仪

0x7506 测硫仪

0x7507 测氢仪

-- 0x76 网络综合

0x7601 SIS

0x7602 存储阵列

0x7603 广播控制器

0x7604 拼接控制器

0x7605 LED控制器

0x7606 门禁控制器

0x7607 录像机

0x7608 解码器

0x7609 信号灯

0x760a 摄像头

0x760b 门禁系统

0x760c RF读卡器

### 1.1.2 网元配置表omc\_neconfig

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| NeNo | 整型 | 网元ID，形式为 0xaabbbb，其中0xaa定义方式同设备类型；  0xbbbb 表示当前设备类型下具体的设备，例如重车衡有3个，那么分别编号为 0x010001 / 0x010002 / 0x010003； |
| NeAlias | 字符串 | 网元中文描述名 |
| NeType | 整型 | 定义方式同设备/服务类型定义 0xaa； |
| AccessType | 整型 | 1: 被动连接 2: Ethernet ...... |
| EthernetIP | 字符串 | 设备代理的连接地址 |
| EthernetPort | 整型 | 设备代理的连接端口 |
| EthernetType | 整型 | 1: tcp 2:udp |

### 1.1.3告警配置表omc\_alarminfo

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| AlarmType | 整型 | 告警类型，形式为 0xaa??bbbb  其中0xaa定义方式同设备类型；  0xbbbb 表示当前设备类型下具体的告警种类，例如原煤传送的入口端超差报警 0x02070001 ，原煤传送的出口端超差报警0x02070002；  例如系统告警:  0x310001: 心跳告警  0x310002: 限值告警  0x310003: 范围告警  0x510001: os time  0x510002: os cpu  0x510003: os mem  0x510004: app cpu  0x510005: app mem  0x510006: app abend  0x510007: alarm data overflow  0x510008: log data overflow  0x510009: performance data overflow |
| AlarmName | 字符串 | 配置项对应的值 |
| AlarmLevel | 整型 | 取值范围1~5，分别告示从重到轻的5种告警级别：  严重告警、主要告警、次要告警、警告告警、消除告警 |
| AlarmClass | 整型 | 取值范围1～7，表示告警所属的分类：  处理错误、业务质量、链路通常（通信异常）、运行告警、软件告警、硬件告警、环境告警 |
| IsEvent | 整型 | 0 表示非事件型告警  1 表示事件型告警  事件型告警是指产生即恢复的告警，例如 HA 上报的进程异常告警，因为 HA 在检测到服务程序异常退出的同时，就会重新启动服务程序，也就是告警被检测到的时候，就已经消除了，对于这类告警，只有在客户端当前页面没有被刷新的情况下才需要显示，一旦刷新，则不再显示； |

### 1.1.4告警数据表omc\_alarmrec

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| AlarmNo | 整型 | 告警 ID，顺序增长，为标识每个不同的 Alarm |
| NeID | 整型 | 同 neconfig 中 NeID 的定义 |
| AlarmType | 整型 | 同 alarminfo 中 AlarmType 定义 |
| ModuleNo | 整型  模块号  结点号 | 对应设备报警的情况，该项表示设备号；  对于服务报警的情况，该项表示结点号，即服务运行所在的服务器结点； |
| CardNo | 整型  板卡号  业务编码 | 对应设备报警的情况，该项表示板卡号；  对于服务报警的情况，0xaabbbb，其中0xaa表示业务编码，同上定义；  0xbbbb 表示某个具体的业务，例如制样的业务在同一个机器上可以运行3个，那么分别对应编号是：0x070001 0x070002 0x070003 |
| AlarmValue | 整型  端口号  批次号 | 对应设备报警的情况，该项表示故障的端口号；  对于服务报警的情况，表示批次号 / 操作票号等同一个业务中的序号； |
| AlarmTime | 字符串 | 告警时间，检测到告警时候的时间；1970以来的秒数； |
| ResumeTime | 字符串 | 告警恢复时间，如果告警已经恢复，那么将产生恢复时间，未恢复的告警，恢复时间为0；1970以来的秒数； |
| RepaireMark | 字符串 | 告警备注 |
| Status | 整型 | 告警状态：  0：活动告警  1：已恢复告警 |
| Confirm | 整型 | 0：告警未确认  1：告警已确认 |
| ConfirmTime | 字符串 | 告警的确认时间，告警确认的时间； |
| ConfirmUser | 字符串 | 告警的确认用户，操作告警确认的这个用户的用户名； |
| Eliminate | 整型 | 0：告警未清除  1：告警已清除 |
| EliminateTime | 字符串 | 告警的消除时间，告警确认的时间； |
| EliminateUser | 字符串 | 告警的消除用户，操作告警确认的这个用户的用户名； |

备注：告警初次产生，Confirm / Eliminate相关各项均为0，只有用户使用告警确认功能时候，Confirm / ConfirmTime / ConfirmUser才会填值；

对于系统告警：

0x01: 心跳告警, ModuleNo 代表点号

0x02: 限值告警, ModuleNo 代表点号

0x03: 范围告警, ModuleNo 代表点号

### 1.1.5 性能测量数据表omc\_measurekpi

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| ID | 字符串 | 配置项 ID，从1开始顺序增加 |
| NeID | 整型 | 网元ID，同 neconfig 中 NeID |
| Time | 字符串 | 性能数据上报的时间 |
| Cpu | 整型 |  |
| Mem | 整型 |  |

### 1.1.6 告警过滤数据表omc\_alarmfilter / 告警抑制数据表omc\_alarmrestrain

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| ID | 整型 | 配置项 ID，从1开始顺序增加 |
| NeID | 整型 | 网元ID，同 neconfig 中 NeID |
| AlarmType | 整型 | 告警类型 |
| AlarmStartTime | 字符串 | 屏蔽告警的起始时间，距离 1970.1.1 的秒数； |
| AlarmEndTime | 字符串 | 屏蔽告警的结束时间，距离 1970.1.1 的秒数； |
| AlarmLevel | 整型 | 告警级别 |
| AlarmClass | 整型 | 告警类型 |
| Enabled | 整型 | 该条件是否生效，0 表示不生效，1表示生效 |

### 1.1.7 系统操作日志omc\_logrec

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| LogNo | 整型 | ID，从1开始顺序增加 |
| UserId | 字符串 | 产生操作的实际用户 |
| CustomerIP | 字符串 | 产生操作的操作台 IP |
| CustomerPort | 整型 | 产生操作的操作台端口号 |
| OperationNo | 整型 | 操作类型 |
| RecTime | 字符串 | 记录时间，距离 1970.1.1 的秒数； |
| Details | 字符串 | 操作秒数 |

备注：该表为各个具体运行的业务、服务填写，例如用户删除告警操作，会进行日志记录，便于后期追索。

### 1.1.8 性能数据omc\_performancedata

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| ID | 整型 | ID，从1开始顺序增加 |
| NeID | 整型 | 设备ID，或者服务ID |
| Node | 整型 | 产生操作的操作台 IP |
| RecTime | 字符串 | 产生时间，距离 1970.1.1 的秒数； |
| Type | 整型 | 类型  1：服务器cpu测量数据  2：服务器mem测量数据  3：告警数据容量监视  4：日志数据容量监视  5：性能测量数据容量监视 |
| Value | 字符串 | 测量数值 |

### 1.1.9 用户数据omc\_user

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| UserName | 字符串 | 用户名 |
| Password | 字符串 | 密码 |
| Level | 整型 | 待实现：  用户等级，目前只定义0级用户和1级用户  类似服务结点上进程配置等页面，只允许0级用户进入；  0级用户只1个；  后期 omc 服务器增加协议，多个0级用户登录，通知前者踢出信息； |
| Details | 字符串 | 详细信息 |

### 1.1.10 心跳告警配置omc\_heartbeat

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| SignalNo | 整型 | 遥信号 |
| LiveTime | 整型 | 生存时间，秒为单位 |

### 1.1.11 越限告警配置omc\_vallimit

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| SignalNo | 整型 | 遥信号 |
| AlarmType | 整型 | 告警类型，不同的遥信号产生告警，告警名称和级别未必相同，允许相同 |
| LimitVal | 整型 | 限值，不正常的遥信值 |
| InScope | 整型 | 限值判断的类型，缺省为>=  RT\_DATASPY\_LG 1 // >  RT\_DATASPY\_LE 2 // >=  RT\_DATASPY\_SM 3 // ==  RT\_DATASPY\_SE 4 // <=  RT\_DATASPY\_EQ 5 // == |

### 1.1.12 范围告警配置omc\_scopelimit

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| SignalNo | 整型 | 遥测号 |
| AlarmType | 整型 | 告警类型，不同的遥测号产生告警，告警名称和级别未必相同，允许相同 |
| MinVal | 整型 | 下限值 |
| MaxVal | 整型 | 上限值 |
| InScope | 整型 | 1: 表示范围内告警　　0:表示不在范围内告警 |
| BlindVal | 整型 | 死区: 相当于该项测量值的误差允许范围 |

### 1.1.13 表定义表omc\_propertydef

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| PropID | 整型 | 唯一索引号 |
| NeType | 整型 | 设备类型/服务器类型，目前约定如下：  0x31 表示系统  0x51 表示服务器结点  0x61 表示运行服务  0x0201 表示重车衡业务  0x0202 表示轻车衡业务  0x0203 表示采样机（皮带）  0x0204 表示汽车采样业务  0x0205 表示火车采样业务  0x0206 表示原煤传输业务  0x0207 表示制样机业务  0x0208 表示下料仓业务  0x0209 表示预备仓业务  0x020a 表示天平业务  0x020b 表示破碎机业务  0x0210 表示存样柜业务  0x0211 表示化验业务 |
| PropName | 字符串 | 服务进程名称 |
| ColumnName | 字符串 | 服务进程数据库字段名 |
| PropDesc | 字符串 | 服务进程描述 |
| PropType | 整型 | 属性类型  1: 整型  2: 浮点数  3: 枚举类型  4: 字符串  5: 文本 |
| Required | 整型 | 是否必选项 |
| Visible | 整型 | 是否可见 |
| ReadOnly | 整型 | 是否只读 |
| MaxLength | 整型 | 最大长度 |
| MinLength | 整型 | 最小长度，为0表示无限值 |
| DefaultValue | 字符串 | 缺省值 |
| RegularExpression | 字符串 | 正则表达式，保留 |
| ForeignKey | 字符串 | 外部关键字，保留 |
| IsKey | 整型 | 是否为关键字 |
| Version | 整型 | 0~0xff 系统保留  1：系统必须，对应表不能新增、修改、删除任何操作  2：系统必须，对应表允许新增属性，不允许修改删除已有系统项  用户新增属性只能使用 0x100~ 的版本号 |
| FormType | 表格类型 | 1：横表  2：纵表 |
| ValueScopes | 字符串 | 取值范围，由,号分割各个取值的字符串，如：1,3,5 |

## 1.2 HA 数据字典

### 1.2.1结点上服务进程配置ha\_appconfig

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| AppID | 整型 | 进程ID |
| NodeID | 整型 | 服务器结点 |
| OrderNo | 整型 | 顺序号 |
| AppName | 字符串 | 进程名 |
| AppPath | 字符串 | 进程路径 |
| AppParam | 字符串 | 进程启动参数 |
| CanNotRunning | 整型 | 守护方式 0 保留, 1 正常守护,2不许运行, 3 不守护  --默认 0 |
| Supervise | 整型 | 监视方式 0 不监视 1 保活 2 性能监视 3 全监视 |
| WorkPath | 字符串 | 工作路径  --默认空表示没要求 |
| AllOver | 整型 | 此程序一旦over就必须全部应用程序重新启动  --默认0 |
| OperationalCfg | 整型 | 1: 长期运行  2: 启动一次，随 ha 软狗启动，只启动一次  3: 定周期运行  4: 手工启动  --默认1 |
| OperationalTm | 字符串 | 如果进程是定周期运行的情况，这里值类似1200,1300  12表示12点  00表示0分  多个分段表示进程启动时点，周期运行的进程允许一天之内多次启动 |
| StaySleep | 整型 | 停留多长时间才启动下一个  是应用程序首次启动等待多长时间再启动下一个  --默认0，为启动进程后不等待 |
| Heartbeat Port | 整型 | 是网络监听端口号列表，监听  --默认为空，不启动网络判断 |
| Heartbeat Timeout | 整型 | 是网络超时，没有收到数据的超时  --默认为"60000" |
| HeartbeatData | 整型 | 是网络心跳报文  --默认为"a55aa55a" |
| WithWin | 整型 | 进程窗口模式 0 无窗口 1 控制台  --默认0 |
| CpuLimit | 整型 | 该机器结点上该服务进程的cpu限值，单位为：%  --默认0，表示无特殊约定 |
| MemLimit | 整型 | 该机器结点上该服务进程的mem限值，单位为：M  --默认0，表示无特殊约定 |
| AppAlias | 字符串 | 进程的描述名 |
| RunMode | 整型 | 运行周期 1: 长期运行 2: 启动一次 3: 定周期运行 |
| AppType | 整型 | 程序类型 0 保留 1 后台服务 2 工具 3 客户端 4 服务模块 |

## 1.3 VXD

### 1.3.1 VXD虚拟终端遥信配置表 omc\_vxd\_rt\_yx

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| ID | 整型 | 唯一索引号 |
| AppID | 整型 | 应用ID |
| TerminalID | 整型 | 终端ID |
| RTUrl | 字符 | 实时数据(RTURL) |

### 1.3.2 VXD虚拟终端遥测配置表 omc\_vxd\_rt\_yc

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| ID | 整型 | 唯一索引号 |
| AppID | 整型 | 应用ID |
| TerminalID | 整型 | 终端ID |
| RTUrl | 字符 | 实时数据(RTURL) |

### 1.3.3 VXD虚拟终端遥文配置表 omc\_vxd\_rt\_yw

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| ID | 整型 | 唯一索引号 |
| AppID | 整型 | 应用ID |
| TerminalID | 整型 | 终端ID |
| RTUrl | 字符 | 实时数据(RTURL) |

## 1.4 RT

### 1.4.1 三遥信息订阅表 omc\_rtsubscribe

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| ID | 整型 | 唯一索引号 |
| AppID | 整型 | App ID |
| IP | 整型 | App 的本地ip地址 |
| Port | 整型 | App 的本地端口 |
| TableName | 字符 | 订阅的表名 |
| StartID | 整型 | 订阅的起始ID |
| EndID | 整型 | 订阅的终止ID ，StartID EndID都为0，表示全订阅 |

## 1.5 用户权限

### 1.5.1 用户表 user

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| ID | 整型 | 用户ID |
| UserName | 字符串 | 账号 |
| PassWord | 字符串 | 密码 |
| Name | 字符串 | 用户名（用于页面显示） |

### 1.5.2 用户组表 group

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| ID | 整型 | 用户组ID |
| PID | 整型 | 父用户组ID |
| GroupName | 字符串 | 用户组名 |

### 1.5.3 用户用户组关联表 user\_group

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| ID | 整型 | 唯一索引号 |
| UID | 整型 | 用户ID |
| GID | 整型 | 用户组ID |

### 1.5.4 权限表 authority

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| ID | 整型 | 唯一索引号 |
| MenuID | 整型 | 菜单ID |
| MenuPID | 整型 | 父菜单ID |
| Name | 字符串 | 菜单名称 |
| MenuURL | 字符串 | 菜单URL |
| ElementID | 整型 | 页面元素ID |

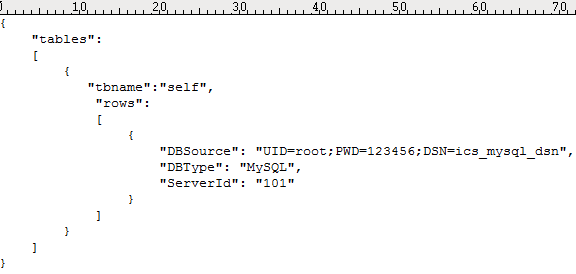
### 1.5.5 用户组权限关联表 group\_authority

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段类型** | **字段说明** |
| ID | 整型 | 唯一索引号 |
| GID | 整型 | 用户组ID |
| AID | 整型 | 权限ID |
| Visible | 整型 | 是否可见 |

# 2. 配置文件

## 2.1 全系统配置文件

ICS4000的根目录下有文件SvrGlobal.json。



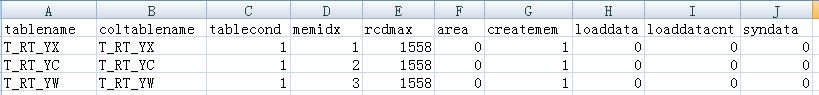
|  |  |
| --- | --- |
| **字段名** | **字段说明** |
| DBSource | 当前数据库的链接信息 |
| DBType | 当前数据库类型 |
| ServerId | **当前服务器结点的编号，要求所有的服务器，这个编码不重复** |

## 2.2 RT 配置文件

实时库的配置分为表模式配置和列模式配置，表模式配置文件中，定义了当前实时库需要哪些数据表，以及这些数据表内存顺序号，最大允许的记录条数等信息；列模式配置文件中，定义了当前实时表中的字段的相关信息，例如字段名称、字段类型、字段长度等等；

### 2.2.1 表模式文件

根据Rttable.csv，由CsvFileToDefFile将 excel 表内容转换为紧凑的 def 二进制格式，更紧凑，适应于大量表定义时候，减少文件读取所需内存空间。

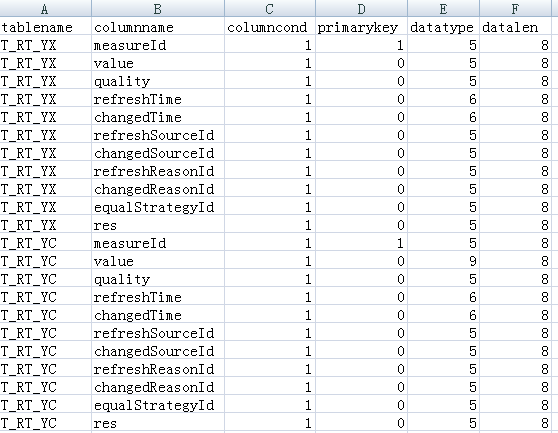


其中，各项定义如下：

|  |  |
| --- | --- |
| **字段名** | **字段说明** |
| **tablename** | 表名称，示例中定义了6个表； |
| **tablecond** | 表的有效性，通常该项都为1；只在虚表（中间临时中转表定义，该项可为0，不常用） |
| **memidx** | 在单分区服务器中，该表所使用的内存索引值，各表之间不能重复 |
| **rcdmax** | 当前表的最大记录数（行数） |
| **area** | 保留字段 |
| **createmem** | 初始化是否创建内存，如果是单服务器运行，该项全部设置为1；如果是分区运行，该项设置为0 |
| **loaddata** | 是否加载初始数据，保留项 |
| **loaddatacnt** | 加载初始数据个数，保留项 |
| **syndata** | 是否向数据库进行数据同步，保留项 |

### 2.2.2 列模式文件

根据Rtcolumn.csv，由CsvFileToDefFile将 excel 表内容转换为紧凑的 def 二进制格式，更紧凑，适应于大量列定义时候，减少文件读取所需内存空间。



其中，各项定义如下：

|  |  |
| --- | --- |
| **字段名** | **字段说明** |
| **tablename** | 列对应的表的名称，为匹配使用 |
| **columnname** | 列名称 |
| **columncond** | 列是否有效，保留项 |
| **primarykey** | 当前列是否为关键字，目前的实时库只支持 |
| **datatype** | 数据类型  RDB\_UCHAR = 0,  RDB\_CHAR = 1,  RDB\_USHORT = 2,  RDB\_SHORT = 3,  RDB\_UINT = 4,  RDB\_INT = 5,  RDB\_ULONG = 6,  RDB\_LONG = 7,  RDB\_FLOAT = 8,  RDB\_DOUBLE = 9,  RDB\_STRING = 10, |
| **datalen** | 数据长度 |

## 2.3 数据库中的初始配置

### 2.3.1 缺省网元配置

网元大部分由omc客户端页面上进行维护，有一类为系统初始必须具备的网元，不允许删除、修改：

1. NeNo=1& NeType=0x31的网元，为系统网元，不允许删除、修改；

2. NeNo=0x510001，代表中心服务器，不允许删除、修改；

3. 0x610001，代表ha传递进程异常信息等；

4. NeNo=0x0201,0x0202,~0x02ff为业务网元，不允许删除、修改；

5.NeNo=0x040001,0x040002~0x04ffff为业务网元，随系统安装自动创建；

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **网元ID** | **网元名称** | **网元类型** | **连接方式** | **连接地址** |
| **1** | 系统 | 0x31 | 被动 | 空 |
| **0x510001** | 中心服务器 | 0x51 | tcp | Ip地址 |
| **0x0201~02ff** | 【[详细定义转至1.1.1](#业务网元定义)】 | 0x0201 | 被动 | 空 |
|  |  |  |  |  |

### 2.3.2 缺省告警配置

数据表omc\_alarminfo中定义了某种类型的告警，它的名称、告警级别、分类等信息。由于我们的系统中有些告警属于系统告警，不能由 OMC 客户端后期添加，所以作为基础数据在建立数据库的时候就已经建立；

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **告警类型** | **告警名称** | **告警级别** | **告警分类** | **事件型** |
| **0x310001** | 心跳告警 | 3 | 4 | 0 |
| **0x310002** | 越限告警 | 3 | 4 | 0 |
| **0x310003** | 范围告警 | 3 | 4 | 0 |
| **0x510001** | 时钟告警 | 1 | 7 | 0 |
| **0x510002** | 服务器cpu告警 | 1 | 7 | 0 |
| **0x510003** | 服务器内存告警 | 1 | 7 | 0 |
| **0x510004** | 服务进程cpu告警 | 1 | 7 | 0 |
| **0x510005** | 服务进程内存告警 | 1 | 7 | 0 |
| **0x510006** | 历史告警数据越限 | 1 | 4 | 0 |
| **0x510007** | 历史日志数据越限 | 1 | 4 | 0 |
| **0x510008** | 历史性能数据越限 | 1 | 4 | 0 |
| **0x060001** | 差超告警－前端质量差超 | 2 | 2 | 0 |
| **0x060002** | 差超告警－后端质量差超 | 2 | 2 | 0 |

**备注**

告警级别定义如下，具体参见数据字典文档：

1: 严重告警

2：主要告警

3：次要告警

4：警告告警

5：消除告警

告警分类定义如下，具体参见数据字典文档：

1：处理错误

2：业务质量

3：链路通常（通信异常）

4：运行告警

5：软件告警

6：硬件告警

7：环境告警

事件型定义如下，具体参见数据字典文档：

0：当前告警非事件型告警；

1：当前告警是事件型告警；某些告警的产生本身就带有告警恢复的含义，例如操作日志容量越限告警，产生的时候，系统本身就已经转移了日志，降低日志在容量门限以下，这类告警产生即恢复，视为事件型告警。

### 2.3.3缺省系统设置

　　数据表 omc\_omconfig 中包含整个系统的基础配置，是系统运行的基础。该表以键值对的形式表达ics基础配置；

|  |  |  |
| --- | --- | --- |
| **配置序号** | **配置项说明** | **配置值** |
| 1 | 通道控制端口号，omc客户端连接omc服务器所使用的端口 | 6687 |
| 2 | App 连接omc服务器所使用的端口，用于传送告警 | 6688 |
| 3 | Omc 服务器地址 | 类似192.168.1.1，目前只能手动修改 |
| 4 | 历史告警存放天数 | 365 |
| 5 | 历史告警存放条数 | 100000 |
| 6 | 历史日志存放天数 | 365 |
| 7 | 历史日志存放条数 | 100000 |
| 8 | 历史性能数据存放天数 | 365 |
| 9 | 历史性能数据存放条数 | 100000 |
| 10 | 自动备份的方式  0：表示无自动备份  1：按天备份  2：按周备份 | 1 |
| 11 | 自动备份的内容  空：表示全备份  数据表字符串：表示备份目标表 | 空 |
| 12 | 自动备份的介质 | D:\\ics4000\\Backup\\ |
| 13 | 自动备份的时间 | 0 |
| 14 | 时差报警阀值，单位：秒 | 180 |
| 15 | 系统cpu报警阀值，单位：% | 75 |
| 16 | 系统mem报警阀值，单位：% | 75 |
| 17 | 进程cpu报警阀值，单位：% | 75 |
| 18 | 进程mem报警阀值，单位：M | 4096 |
| 19 | 服务器结点性能采集的时间间隔，单位：秒 | 600 |
| 20 | DB容量采集的时间间隔，单位：秒 | 600 |
| 21 | omc 北向端口 | 6691 |
| 22 | omc 北向地址 | 无 |
| 23 | omc 南向端口 | 6692 |
| 24 | omc 南向地址 | 无 |
| 25 | DB User，中心数据库用户 | root |
| 26 | DB Password，中心数据库密码 | 123456 |
| 27 | DB Source，中心数据库 | ics4000 |
| 28 | cics 数据库source | UID=ICS;PWD=ICS;DSN=LuaHost\_ICS\_dsn; |
| 29 | cics 数据库type | Oracle |
| 40 | cics 中接口数据库 source | UID=ICS;PWD=ICS;DSN=LuaHost\_ICS\_dsn; |
| 41 | cics 中接口数据库type | Oracle |
| 30 | LogDisables | 缺省关闭的log通道，缺省为空 |
| 31 | LogMeasureChanged |  |
| 32 | MonsbMapName | ygct\_ics\_measures\_monsb |
| 33 | MonsbCount | 20480 |
| 34 | YcaddMapName | ygct\_ics\_measures\_ycadd |
| 35 | YcaddCount | 20480 |
| 36 | StrawMapName | ygct\_ics\_measures\_straw |
| 37 | StrawCount | 5120 |
| 38 | FileInfoMapName | ygct\_ics\_fileinfo |
| 39 | FileInfoCount | 1024 |
|  |  |  |
|  |  |  |
|  | | |
| 101 | 外部日志报警阀值，单位：M | 1048576 |
| 102 | 外部日志报警阀值，单位：天 | 365 |
| 103 | 单一 app log 大小，单位：M | 1024 |
|  | | |
| 0x0100 | rt proxy 监听端口 | 6689 |
| 0x0101 | rt proxy 监听地址 | 127.0.0.1 |
| 0x0102 | Rt 同步本地端口 | 5661 |
| 0x0103 | Rt 同步本地地址 | 127.0.0.1 |
| 0x0104 | Rt同步远端端口 | 9102 |
| 0x0105 | Rt同步远端地址 | 127.0.0.1 |
| 0x0106 | Rt同步sourceID | 2305 |
| 0x0107 | Rt2Stack RT到协议栈本地端口 | 6695 |
| 0x0108 | Rt2Stack RT到协议栈本地地址 | 127.0.0.1 |
| 0x0109 | Rt2Stack RT到协议栈SourceID | 6695 |
|  | | |
| 0x0200 | HA监听端口，允许连接客户端获取系统/服务运行数据 | 6690 |
| 0x0201 | HA启动等级 | 0 |
| 0x0202 ~  0x02ff | HA 缺省启动进程，ha\_appconfig 的缺省值 | 缺省启动进程sql语句 |

### 2.3.4缺省进程配置

　　数据表ha\_appconfig中缺省包含了中心服务器启动时候必须的进程。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **进程ID** | **进程名称** | **服务器结点** | **延迟启动** | **运行模式** |
| 101 | Rt服务器 --- RtdbProxy | 0x510001 | 12秒 | 1: 常驻运行 |
| 102 | Ics Rt数据同步服务 --- IcsRtDataSync | 0x510001 | 12秒 | 1: 常驻运行 |
| 103 | Omc服务器 --- OmcServer | 0x510001 | 12秒 | 1: 常驻运行 |
| 104 | Ics Rt数据监视服务 --- IcsRtDataSpy | 0x510001 | 12秒 | 1: 常驻运行 |
| 105 | Rt数据查看工具 --- RtDataShow | 0x510001 | 0秒 | 4: 手工运行 |
| 106 | Omc测试工具 --- OmcTestApi | 0x510001 | 0秒 | 4: 手工运行 |
| 107 | Rt模拟数据测试工具 --- RtProxyTestApi | 0x510001 | 0秒 | 4: 手工运行 |

### 2.3.5 缺省属性配置

数据表omc\_propertydef中缺省包含了服务器结点/进程服务/设备添加属性时，必须具备的属性。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **属性**  **编号** | **对应**  **设备**  **类型** | **属性名称** | **数据表**  **列名称** | **属性**  **描述** | **属性**  **类型** | **必填** | **可见** | **只读** | **最大**  **长度** | **最小**  **长度** | **缺省值** | **表达式** | **外键** | **选值范围** | **版**  **本**  **号** | **表格**  **类型** |
| 1 | 0x31 | ID | ID | NULL | 1 | 1 | 1 | 1 | 11 | 1 | NULL | auto | NULL | NULL | 1 | 1 |
| 2 | 0x31 | IP地址 | IP | NULL | 4 | 1 | 1 | 0 | 15 | 7 | 127.0.0.1 | NULL | NULL | NULL | 1 | 1 |
| 3 | 0x61 | 进程ID | AppID | NULL | 1 | 1 | 1 | 1 | 11 | 0 | NULL | auto | NULL | NULL | 1 | 1 |
| 4 | 0x61 | 运行结点 | NodeID | NULL | 1 | 1 | 0 | 0 | 11 | 0 | NULL | NULL | property\_0x31,ID | NULL | 1 | 1 |
| 5 | 0x61 | 运行次序 | OrderNo | NULL | 1 | 1 | 1 | 0 | 11 | 0 | NULL | unique | NULL | NULL | 1 | 1 |
| 6 | 0x61 | 运行路径 | AppPath | NULL | 1 | 1 | 1 | 0 | 512 | 0 | NULL | NULL | NULL | NULL | 1 | 1 |
| 7 | 0x61 | 运行参数 | AppParam | NULL | 1 | 1 | 1 | 0 | 512 | 0 | NULL | NULL | NULL | NULL | 1 | 1 |
| 8 | 0x61 | 禁止运行 | CanNotRunning | NULL | 3 | 1 | 1 | 0 | 1 | 1 | 0 | NULL | NULL | 1:是,0:否 | 1 | 1 |
| 9 | 0x61 | 工作路径 | WorkPath | NULL | 4 | 1 | 1 | 0 | 512 | 0 | NULL | NULL | NULL | NULL | 1 | 1 |
| 10 | 0x61 | 重启服务 | AllOver | NULL | 3 | 1 | 1 | 0 | 11 | 0 | 0 | NULL | NULL | 1:是,0:否 | 1 | 1 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 11 | 0x61 | 运行方式 | OperationalCfg | NULL | 3 | 1 | 1 | 0 | 1 | 1 | 4 | NULL | NULL | 1:长期运行,2:启动一次,3:定周期运行,4:手工启动 | 1 | 1 |
| 12 | 0x61 | 启动周期 | OperationalTm | NULL | 1 | 1 | 1 | 0 | 512 | 0 | NULL | NULL | NULL | NULL | 1 | 1 |
| 13 | 0x61 | 运行时延 | StaySleep | NULL | 1 | 1 | 1 | 0 | 11 | 1 | 0 | NULL | NULL | NULL | 1 | 1 |
| 14 | 0x61 | 心跳端口 | HeartbeatPort | NULL | 1 | 1 | 1 | 0 | 11 | 1 | NULL | NULL | NULL | NULL | 1 | 1 |
| 15 | 0x61 | 心跳超时 | HeartbeatTimeout | NULL | 1 | 1 | 1 | 0 | 11 | 1 | 30000 | NULL | NULL | NULL | 1 | 1 |
| 16 | 0x61 | 心跳数据 | HeartbeatData | NULL | 4 | 1 | 1 | 0 | 512 | 0 | NULL | NULL | NULL | NULL | 1 | 1 |
| 17 | 0x61 | 窗口运行 | WithWin | NULL | 3 | 1 | 1 | 0 | 1 | 1 | NULL | NULL | NULL | 1:是,0:否 | 1 | 1 |
| 18 | 0x61 | cpu阀值 | CpuLimit | NULL | 1 | 1 | 1 | 0 | 3 | 1 | NULL | NULL | NULL | NULL | 1 | 1 |
| 19 | 0x61 | mem阀值 | MemLimit | NULL | 1 | 1 | 1 | 0 | 3 | 1 | NULL | NULL | NULL | NULL | 1 | 1 |
| 20 | 0x01 | ID | ID | NULL | 1 | 1 | 1 | 1 | 11 | 1 | NULL | auto | NULL | NULL | 2 | 1 |
| 21 | 0x01 | 名称 | Name | NULL | 4 | 1 | 1 | 0 | 512 | 6 | NULL | NULL | NULL | NULL | 2 | 1 |
| 22 | 0x01 | 设备型号 | Type | NULL | 4 | 1 | 1 | 0 | 512 | 6 | NULL | NULL | NULL | NULL | 2 | 1 |
| ......  ~58 | 0x02~0x0b 0x10 0x11  按照如上设备类型重复，每种设备都具备ID/名称/设备型号等3个基本属性。 | | | | | | | | | | | | | | | |

# 3.通信包

## 3.1 通用描述

### 3.1.1请求包通用描述

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **长度** | **描述** |
| Start | BYTE | 1 | 起始位 0xe9 |
| CmdSeq | BYTE | 4 | 命令序列号，低位在前 |
| Command | BYTE | 4 | 命令序列号，低位在前 |
| ReqLen | BYTE | 4 | 包长度，特指下面 BUF 的长度 |
| CmdBuf | BYTE | 2048之内 | 包内容，具体长度由ReqLen指定 |
| Crc | BYTE | 1 | 循环校验码，普通包，该项不必做成、不必检测  特定需要检测crc的包中特别说明，例如防网络攻击所使用的心跳包 |
| End | BYTE | 1 | 终止位 0xea |

整个包内容部分，遇到>= 0xe8 的进行转译，避免解包错误，转译方式将原BYTE拆分为2个BYTE，第一个为 0xe8，第二个为原字节-0xe8。

例如包的实体内容为：

0xe9 0x00 0x00 0x00 0xec 0x00 0xea

转译后为

0xe9 0x00 0x00 0x00 0xe8 0x04 0x00 0xea

e9

02000000**0**

**8000000**

200000000

c10002011000000e81703000000000000848e615a00000000000000000000000034ea  
e9020000001c000000**040000002c010000**2bea

### 3.1.2 退出包

|  |  |
| --- | --- |
| **命令码** | **命名解释** |
| 发起包：0x01  响应包：无 | 推出连接 |

## 3.2 OMC 协议包

### 3.2.1命令码

|  |  |
| --- | --- |
| **命令码** | **命名解释** |
| 发起包：0x12  响应包：0x22 | 登录 / 心跳 |
| 发起包：0x13  响应包：0x23 | 数据更新 |
| 发起包：0x14  响应包：0x24 | 告警设置 / 告警恢复 / 告警确认 / 告警取消确认 / 告警消除 |
| 发起包：0x15  响应包：0x25 | 根据告警地址信息获取告警信息 |
| 发起包：0x16  响应包：0x26 | 根据告警编号获取告警信息 |
| 发起包：0x17  响应包：0x27 | 退出登录 |

### 3.2.2登录 / 心跳

omc-client -> omc-server

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **长度** | **描述** |
| User | 字符串 | 32 | 用户 |
| Password | 字符串 | 32 | 密码 |
| PhysicalNo | 字符串 | 48 | 当前机器硬件编码 |

omc-server -> omc-client

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **长度** | **描述** |
| Rst | 整型 | 4 | 成功/失败标识  0: 表示成功  非0：表示失败 |

### 3.2.3数据更新

omc-client -> omc-server

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **长度** | **描述** |
| User | 字符串 | 32 | 用户 |
| Kind | 整型 | 4 | 更新类型  0 //所有数据  1 //网元数据  2 //用户数据  3 //配置数据  4 //告警配置  5 //告警抑制配置 |

omc-server -> omc-client

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **长度** | **描述** |
| Rst | 整型 | 4 | 成功/失败标识  0: 表示成功  非0：表示失败 |

### 3.2.4告警设置 / 告警恢复 / 告警确认 / 告警取消确认 / 告警消除

app -> omc-server

omc-client -> omc-server

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **长度** | **描述** |
| AlarmNo | 整型 | 4 | 告警编号 |
| Action | 整型 | 4 | 动作  OMC\_CONFIRM\_ALARM 1 //告警确认  OMC\_INVOKE\_CONFIRM\_ALARM 2 //取消确认  OMC\_ERASE\_ALARM 3 //告警清除  OMC\_SET\_ALARM 4 //告警同步  OMC\_CLEAR\_ALARM 5 //告警恢复 |
| User | 字符串 | 32 | 操作用户 |
| NeID | 整型 | 4 | 网元ID / 服务 ID |
| AlarmType | 整型 | 4 | 告警类型 |
| ModuleNo | 整型 | 4 | 模块号 |
| CardNo | 整型 | 4 | 板卡号 |
| PortNo | 整型 | 4 | 端口号 |

omc-server -> omc-client

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **长度** | **描述** |
| AlarmNo | 整型 | 4 | 告警编号，为0表示出现错误 |

### 3.2.5 登出

omc-client -> omc-server

无包内容

omc-server -> omc-client

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **长度** | **描述** |
| Rst | 整型 | 4 | 成功/失败标识  0: 表示成功  非0：表示失败 |

## 3.3 HA 协议包

### 3.3.1 命令码

|  |  |
| --- | --- |
| **命令码** | **命名解释** |
| 发起包：0x31  响应包：0x41 | 获取 OS 状态 |
| 发起包：0x32  响应包：0x42 | 获取 App 状态 |

### 3.3.2获取 OS 状态

发起包无具体包内容

ha->omc-server

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **长度** | **描述** |
| Rst | 整型 | 4 | 成功/失败标识  0: 表示成功  非0：表示失败 |
| cpu | 整型 | 4 | Cpu 的% 表达 |
| mem | 整型 | 4 | Mem的% 表达 |
| time | 字符串 | 20 | 距离1970 的秒数，字符串 |

### 3.3.3获取 App 状态

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **长度** | **描述** |
| AppNo | 整型 | 4 | App编号 |

响应包

ha-> omc-server

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **长度** | **描述** |
| Rst | BYTE | 4 | 成功/失败标识  0: 表示成功  非0：表示失败 |
| cpu | 整型 | 4 | Cpu 的% 表达 |
| mem | 整型 | 4 | Mem K为单位 |
| livetime | 字符串 | 20 | 距离1970 的秒数，字符串 |

## 3.4 RT 协议包

### 3.4.1命令码

|  |  |
| --- | --- |
| **命令码** | **命名解释** |
| 发起包：23  响应包：43 | 心跳 |
| 发起包：1  响应包：21 | 打开表 RTDB\_REQ\_OPEN\_TABLE |
| 发起包：17  响应包：37 | 保存表 RTDB\_REQ\_SAVE\_TABLE |
| 发起包：18  响应包：38 | 关闭表 RTDB\_REQ\_CLOSE\_TABLE |
| 发起包：2  响应包：22 | 查询表记录数 RTDB\_REQ\_RCDCNT |
| 发起包：3  响应包：23 | 查询表记录长度 RTDB\_REQ\_RCDLEN |
| 发起包：4  响应包：24 | 增加记录 RTDB\_REQ\_ADDRCD |
| 发起包：5  响应包：25 | 根据关键值删除记录  RTDB\_REQ\_DELRCD\_BY\_KEY |
| 发起包：6  响应包：26 | 根据索引删除记录  RTDB\_REQ\_DELRCD\_BY\_IDX |
| 发起包：7  响应包：27 | 删除全部记录 RTDB\_REQ\_DELALL |
| 发起包：8  响应包：28 | 根据关键值更新记录  RTDB\_REQ\_UPDRCD\_BY\_KEY |
| 发起包：9  响应包：29 | 根据索引更新记录  RTDB\_REQ\_UPDRCD\_BY\_IDX |
| 发起包：10  响应包：30 | 根据关键值获取记录  RTDB\_REQ\_GETRCD\_BY\_KEY |
| 发起包：11  响应包：31 | 根据索引获取记录  RTDB\_REQ\_GETRCD\_BY\_IDX |
| 发起包：12  响应包：32 | 获取全部记录 RTDB\_REQ\_GET\_RCD\_SEG |
| 发起包：26  响应包：33,34 | 获取全部记录，分包发送出来  RTDB\_REQ\_GET\_ALL\_RCD |
| 发起包：14  响应包：35 | 订阅 RTDB\_REQ\_SUBSCRIBE |
| 发起包：16  响应包：36 | 同步记录（多条） |
| 发起包：19  响应包：39 | 获取列定义  RTDB\_REQ\_GET\_COLUMN\_INFO |
| 发起包：20  响应包：40 | 根据查询条件获取记录  RTDB\_REQ\_GETRCD\_BY\_COND |
| 发起包：21  响应包：41 | 根据查询条件更新记录  RTDB\_REQ\_UPDRCD\_BY\_COND |
| 发起包：22  响应包：42 | 根据查询条件删除记录  RTDB\_REQ\_DELRCD\_BY\_COND |
| 发起包：25  响应包：无 | 登录 |
| 发起包：27  响应包：47 | 更新多条记录  RTDB\_REQ\_UPD\_RCD\_LIST |

上述协议中只有红色标注项为外部协议

### 3.4.2 数据包新增参考代码

数据库存在对应记录，新增操作变为更新

基础函数见 Bas\_Package.h Bas\_Package.cpp

TPackageReq tReq;

tReq.m\_nStart = PACKAGE\_REQ\_START; //0xe9

tReq.m\_nEnd = PACKAGE\_REQ\_END; //0xea

tReq.m\_nCmdSeq = 1;

tReq.m\_nCommand = RTDB\_REQ\_ADDRCD;

tReq.m\_nReqLen = RTDB\_MAX\_TABLE\_NAME + nRcdLen; // RTDB\_MAX\_TABLE\_NAME=64

BYTE pCmdBuf[PACKAGE\_SIMPLE\_REQ\_LEN]; // PACKAGE\_SIMPLE\_REQ\_LEN=409够用

memset(pCmdBuf, 0, PACKAGE\_SIMPLE\_REQ\_LEN);

memcpy(pCmdBuf, sTableName.c\_str(), sTableName.length());

int nIdx = RTDB\_MAX\_TABLE\_NAME;

memcpy(&(pCmdBuf[nIdx]), pRcdBuf, nRcdLen);

nIdx += nRcdLen;

tReq.m\_pCmdBuf = pCmdBuf; // pCmdBuf为新增数据的内容二进制表达

int nCmdLen = 0;

BYTE pBuf[PACKAGE\_MAX\_REQ\_LEN];

memset(pBuf, 0, PACKAGE\_MAX\_REQ\_LEN);

EncodeRequest(&tReq, pBuf, nCmdLen); //最后压包，见paqckage

### 3.4.3 数据包更新参考代码

数据库无记录，更新操作变为新增

基础函数见 Bas\_Package.h Bas\_Package.cpp

TPackageReq tReq;

tReq.m\_nStart = PACKAGE\_REQ\_START;

tReq.m\_nEnd = PACKAGE\_REQ\_END;

tReq.m\_nCmdSeq = 1;

tReq.m\_nCommand = RTDB\_REQ\_UPDRCD\_BY\_KEY;

tReq.m\_nReqLen = RTDB\_MAX\_TABLE\_NAME + RTDB\_MAX\_KEY + nRcdLen;

BYTE pCmdBuf[PACKAGE\_SIMPLE\_REQ\_LEN];

memset(pCmdBuf, 0, PACKAGE\_SIMPLE\_REQ\_LEN);

memcpy(pCmdBuf, sTableName.c\_str(), sTableName.length()); **sTableName="T\_RT\_YX"**

int nIdx = RTDB\_MAX\_TABLE\_NAME;

memcpy(&(pCmdBuf[nIdx]), pKeyValue, RTDB\_MAX\_KEY);

nIdx += RTDB\_MAX\_KEY;

memcpy(&(pCmdBuf[nIdx]), pRcdBuf, nRcdLen);

nIdx += nRcdLen;

tReq.m\_pCmdBuf = pCmdBuf;

int nCmdLen = 0;

BYTE pBuf[PACKAGE\_MAX\_REQ\_LEN];

memset(pBuf, 0, PACKAGE\_MAX\_REQ\_LEN);

EncodeRequest(&tReq, pBuf, nCmdLen);

**3.4.4 数据包删除参考代码**

TPackageReq tReq;

tReq.m\_nStart = PACKAGE\_REQ\_START;

tReq.m\_nEnd = PACKAGE\_REQ\_END;

tReq.m\_nCmdSeq = 1;

tReq.m\_nCommand = RTDB\_REQ\_DELRCD\_BY\_KEY;

tReq.m\_nReqLen = RTDB\_MAX\_TABLE\_NAME + RTDB\_MAX\_KEY;

BYTE pCmdBuf[PACKAGE\_SIMPLE\_REQ\_LEN];

memset(pCmdBuf, 0, PACKAGE\_SIMPLE\_REQ\_LEN);

memcpy(pCmdBuf, sTableName.c\_str(), sTableName.length());

int nIdx = RTDB\_MAX\_TABLE\_NAME;

memcpy(&(pCmdBuf[nIdx]), pKeyVal, RTDB\_MAX\_KEY);

nIdx += RTDB\_MAX\_KEY;

tReq.m\_pCmdBuf = pCmdBuf;

int nCmdLen = 0;

BYTE pBuf[PACKAGE\_MAX\_REQ\_LEN];

memset(pBuf, 0, PACKAGE\_MAX\_REQ\_LEN);

EncodeRequest(&tReq, pBuf, nCmdLen);

## 3.5 DA 协议包

### 3.5.1 命令码

|  |  |
| --- | --- |
| **命令码** | **命名解释** |
| 发起包：0x01  临时响应包：0x31  最终响应包：0x61 | 获取时间点上各个YX/YC/YW的值 |
| 发起包：0x02  临时响应包：0x32  最终响应包：0x62 | 获取时间段上各个YX/YC/YW的值 |

### 3.5.2时间点数据获取

client -> da-server

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **长度** | **描述** |
| StartTm | 整型 | 8 | 需要获取数值的时间点，毫秒数  如：1527181331349 |
| Key | 整型 | 8 | 待获取点号个数 |
| KeyLen | 整型 | 8 | 待获取点号的长度，不允许获取过程中包含key长度不一的多个数值，目前YX/YC/YW的key 长度一致，可以一次获取 |
| KeyList |  | Key\*KeyLen | Key 列表 |

da-server -> client

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **长度** | **描述** |
| Key | 整型 | 8 | 点号 |
| Value | 二进制 | 8~128 | 根据点号的类型，判断后续值的长度 |
| ...... |  |  | Key-value 重复多次 |

### 3.5.3时间段数据获取

client -> da-server

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **长度** | **描述** |
| StartTm | 整型 | 8 | 需要获取数值的起始时间，毫秒数  如：1527181331349 |
| EndTm | 整型 | 8 | 需要获取数值的终止时间，毫秒数  如：1527181331349 |
| Interval | 整型 | 8 | 需要获取数值的时间间隔，毫秒数  如：10000 10秒间隔 |
| Key | 整型 | 8 | 待获取点号个数 |
| KeyLen | 整型 | 8 | 待获取点号的长度，不允许获取过程中包含key长度不一的多个数值，目前YX/YC/YW的key 长度一致，可以一次获取 |
| KeyList |  | Key\*KeyLen | Key 列表 |

da-server -> client

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **长度** | **描述** |
| Count | 整型 | 8 | 每个点号对应的值个数，通过StartTm,EndTm,Interval 可直接计算出来 |
| Key | 整型 | 8 | 点号 |
| Value(\*) | 二进制 | 可变 | 根据点号的类型，判断后续单个值的长度  这里的长度是单个值长度×Count |
| KV对 |  |  | Count之后所有的 key-values 符合1个key对应 Count 个 value：Key×(KeyLen+Count×ValueLen) |