DSS-API

目录

一、 dssDAOHelper 插件开发文档

1. 保存数据

名称: /api/db_update

说明:
保存数据到数据库

提交参数:

```
{
  "datasource":"main",      // 数据库 ID
  "list":
  [
    {    // 更新表
     "__updateTable": "test_fieldtype",    // 更新表名
     "__keyFields": "FKey",                    // 删除
     " __fieldINfo":
      {
          "FFieldName":{
              "type":"int"          // 字段类型，详细见备注(1)
           }
      },
     "list": [        // 更新数据记录
          {
              "FIntValue": {"newValue": 9},
              "FKey": { "value": "{659B308C-820D-412B-91DA-6993141E5CCC}" },
              "__type": "update"     // 更新
          },
          {
              "FNumeric": { "newValue": 6},
              "FIntValue": { "newValue": 3},
              "FName": { "newValue": "010" },
              "FKey":   { "newValue": "{E28B57FE-DED3-4523-97DD-1B2F568D1BAE}"},
              "__type": "insert"    // 新增
         },
         {
             "fkey": { "value": "{679B308C-820D-412B-91DA-6993141E5CCC}"},
             "__type": "delete"     // 删除
         }
      ]
    },
    {    // 执行 sql 段
```

```
        "id":"main",          //   id
        "sql":
         {
              "file":"order.select.sql",                    // sql 文件
              "rep_params":                                  //  替换参数(替换脚本中的宏)
               {
                   "$rep_condi$":" and FKey='001'"
               }
         }
    }
  ]
}
```

响应

```
{
    "__result":0,   //  成功返回  0，  失败-1
    "__msg":"错误信息",      //   异常信息(失败时)
    "__tracesql":"update xxx"      // tracesql = 1 (配置文件中配置)
}
```

备注:

(1), 字段类型(__fieldINfo 可以不指定), 默认按照字符串类型进行

"binary", blob 字段值应该是 16 进制字符串(二进制数据转 16 字符串)

"int", 整数类型, "float":浮点型

2. 查询数据

名称: /api/db_query

说明:

查询数据

提交参数:

```
{
  "datasource":"main",       //  数据库 ID
  "response":"json", // json:json 打包方式, cds.xml:clientdataset 的 xmldata 数据
  "list":
  [
    {
        "id":"main",          //  查询 id
        "sql":
         {
              "file":"order.execute.sql",                    // sql 文件
              "page":               //  分页参数
```

```
                {
                    "index":1,   // 获取第几页数据
                    "size":10,   // 每页记录数
                },
                "rep_params":                          // 替换参数(替换脚本中的宏)
                {
                    "$rep_condi$":" and FKey='001'"
                }
            }
        },
        {
            "id":"detail",          // 查询 id
            "sql":
            {
                "file":"order.detail.select.sql",              // sql 文件
                "rep_params":                                  // 替换参数(替换脚本中的宏)
                {
                    "$rep_condi$":" where 1=0"
                }
            }
        },
        {
            "id":"goods",          // 查询 id
            "sql":"select * from bas_goods"     // 直接指定要执行的 sql(allow_client_sql=1, datasource.config.ini 配置)
        }
    ]
}
```

响应

```
{
    __result:0,                          // 成功返回 0， 失败-1
    __msg:"错误信息",                  //  异常信息(失败时)
    "__tracesql":"select xxx"          // tracesql = 1 (配置文件中配置)
    "main":"reponse_data",             // 返回数据,key 值根据查询 id 命名, 值数据根据 reponse 打包
    "main_recordcount":100,            // 返回记录的总数(可以根据记录总数计算出页数(page.index=0)
    "detail":"response_data"           // 返回数据,key 值根据查询 id 命名, 值数据根据 reponse 打包
}
```

返回数据格式说明:

json 格式说明:

```
{
   "list":
   [
       {"FKey":"xxx1","FCode":"001","FName":"001"},
```

```
        {"FKey":"xxx2","FCode":"002","FName":"002"},
    ]
}
```

cds.xml 格式说明:

<XML...>

可以直接赋值给 CDS.XmlData 属性

本说明只对有分页参数提交时才有效，

分页目前只适合 mysql 数据库

分页的脚本需要按照规定格式编写

page.index=0，将返回总记录数(["id"]_recordcount)

分页 sql 格式说明：

```
select    [selectlist]
            bx.BoxName, b.BoxPortID, b.DeviceName
            [/selectlist]
    from wn_device b
        left join wn_box bx on b.BoxID = bx.BoxID
    [countIgnore]
    order by bx.BoxName
    [/countIgnore]
    [page]
```

[selectlist][/selectlist] 区域在进行 count 统计记录时会进行替换 成 count(1) as RecordCount

[countIgnore][/countIgnore] 区域在进行 count 统计记录时会被替换成空字符串

[page] 分页语句 limit 0, 10

3. 执行语句

名称: /api/db_exec

说明:

执行一些 sql 语句(开启事务执行，如果有出现异常将全部回滚)

提交参数:

```
{
    "datasource":"main",       // 数据库 ID
    "list":
    [
        {
            "id":"main",          // 查询 id
            "sql":
             {
```

```
            "file":"order.select.sql",                    // sql 文件
            "rep_params":                                  // 替换参数(替换脚本中的宏)
             {
                 "$rep_condi$":" and FKey='001'"
             }
        }
    },
    {
        "id":"detail",          // 查询 id
        "sql":
         {
            "file":"order.detail.select.sql",             // sql 文件
            "rep_params":                                  // 替换参数(替换脚本中的宏)
             {
                 "$rep_condi$":" where 1=0"
             }
         }
    },
    {
        "id":"exec_param",          // 查询 id
        "sql":
         {
            "script":"insert into abc(a,b,c) values(:a, :b, c)",          // 直接 sql 脚本
            "param_use_type":"param",        // 参数使用方式 param, rep
            "params":                                      // 替换参数(替换脚本中的宏)
             {
                 "a":"1",
                 "b":{value:"base64", type:"base64.blob"},
                 "c":2
             }
         }
    },
    {
        "id":"exec_stroreProc1",          // id
        "storeProc":
         {
            "name":"Get_Data",            // 直接 sql 脚本
            "params":                                      // 替换参数(替换脚本中的宏)
             {
                 "a":"1",
                 "b":{value:"base64", type:"base64.blob"},
                 "out_param1":{output:1},
                 "c":2
             }
```

```
        }
    },

    {
        "id":"goods",          // 查询 id
        "sql":"select * from bas_goods"     // 直接指定要执行的 sql(allow_client_sql=1, datasource.config.ini 配置)
    }
  ]
}
```

响应

```
{
    __result:0,                          // 成功返回 0， 失败-1
    __msg:"错误信息",                    //  异常信息(失败时)
    "__tracesql":"select xxx"           // tracesql = 1 (配置文件中配置)
    "main":1,                           //"main", 为 list 中每一节的 id 执行语句的影响记录条数
    "detail":0
    " exec_stroreProc1":                // 存储过程返回的参数
     {
          "out_param1":{"value":"001"}
     }
}
```

二、 认证监控插件开发文档

客户端通过首次认证获取一个 APICode，在后面的请求中将该 apicode 放到请求头(HTTP 请求头)
如下

```
GET /api/dssDAOHelper/Query HTTP/1.0
apicode:xxxxxx
```

DSS 将根据 apicode 和请求的 api 接口进行认证是否可以访问，如果可以访问则执行 api 接口插件，否则返回错误信息

1. 认证获取 apicode(交互 API)

名称: /api/dssAuthentication/shake

提交参数:

```
{
    "authentication":"xxxxx"      // base64("user:pass")
}
```

响应

{
"apicode":"xxxxx", // 响应 apicode
}

备注: 提交认证信息，认证成功返回 apicode，请在以后的请求头中携带 apicode

2. 配置用户

名称: dssAuthentication.AddUser
参数 user : 用户名 password : 密码 level : 级别
level 范围, 0 – 10 假如，级别为 5，该用户拥有 0-5 级别的 API 访问权限

3. 添加监控 API

名称: dssAuthentication.AddAPI
参数 level 级别 apiname API 名称

4. 获取 apicode

名称: dssAuthentication.CheckUser
参数 authentciation, // base64(user:pass) 认证成功返回 apicode，否则返回空字符串

5. 检测访问合法性

名称: dssAuthentication.CheckAPI
参数 apicode apicode，认证时产生的 apicode apiname API 名称 返回值(整数): 0: 允许访问,

-1: APICode 失效, -2: 访问权限不足 API 的级别为 0 直接返回 0

三、 多帐套插件

1. 获取帐套

名称: /api/account

提交参数:
{
cmdIndex:1001,
}

响应
{
配置的帐套 Json 信息
}

2. 用户登陆

名称: /api/account

提交参数:
{
cmdIndex:1002,
ServiceTypeID:"PC-ERP",
TypeID:"ERP",
UserKey:"001",
UserCode:"001",
UserName:"张三",
AccountOnlyCode:"1001"
}

响应
{
"__result":-1,
"msg":"失败原因"
}

说明: ServiceTypeID: // 服务程序类型 ID PC-Main(电脑的主程序), PC-POS(电脑的 POS 后台), PC-SELL(电脑 POS 收银)

Phone-Main(手机主程序) Pad-Main(平板) TypeID: ERP(ERP 服务), POS(POS 收银) "__result": 0，成功，其他值为失败 "msg"，失败时的错误消息,

3. 用户退出

名称: /api/account

提交参数:
{
cmdIndex:1003
}

响应
{
}

4. 帐套心跳

名称: /api/account

提交参数:
{
cmdIndex:1010
}

响应
{
__result : 1,　　// 1 成功, 2:当前连接未登陆
}

说明: 服务端会检测心跳， 30 秒任何数据会主动踢掉用户

5. 在线列表

名称: /api/account

提交参数:
{
cmdIndex:1020
}

响应

```
{
 OnLineUserList:
 [
   {
     "UserKey":"99247F16-4015-42E6-86DB-49C72CB917A4", //用户 Key
     "UserCode":"a1001",                                    //用户编号
     "UserName":"李齐春",                    //用户名称
     "list":   // 服务程序列表
      {
          "PC-Main":    // 服务程序类型 ID
          {
              "accountOnlyCode":10001,              //帐套唯一码
              "status": 1,    // 在线
          },
          "Phone-POS": // 服务程序类型 ID
          {
              "accountOnlyCode":10001,              //帐套唯一码
              "status": 1,    // 在线
          }
       }
   },
   {
     "UserKey":"99247E16-4015-42E6-86DB-49C72CB917A4", //用户 Key
     "UserCode":"a1002",                                    //用户编号
     "UserName":"张三",                    //用户名称
     "list":
      {
          "PC-Main":
          {
              "accountOnlyCode":10001,              //帐套唯一码
              "status": 1,    // 在线
          },
          "Phone-POS":
          {
              "accountOnlyCode":10001,              //帐套唯一码
              "status": 1,    // 在线
          }
       }
   },
 ]
}
```

6. 帐套服务器信息

名称: /api/account.info
提交: GET

响应:
{
"online":{ // 服务在线信息
"PC-ERP":{
"online":1, // 当前在线数量
"auth":1 // 授权数量
},
"ERP":{
"online":1,
"auth":10
}
}
}

说明: online 节点为帐套在线信息

7. 授权限制配置

配置参数

{
"Auth-Online": // 在线用户授权配置
{
"PC-ERP":10, // 允许登陆 10 个用户
"Phone-ERP":5, // 允许登陆 5 个,
"ERP":10 // ERP 允许登陆 10 个 混合模式(TypeID 查找授权信息)
}
}

说明: 可以通过 ServiceTypeID 查找，也可以通过 TypeID 查找 授权信息 应该从加密狗读取， 网络读取

四、 文件服务插件

1. 下载文件

名称: /api/downfiles
说明: 服务器下载文件(支持迅雷等下载工具下载)
提交参数: GET: http://127.0.0.1:8081/api/downfiles?filename=A001.rar
说明, filename: 需要下载的文件名, A001.rar 是服务端 files 下面的文件

五、 DMS 消息中心

1. 消息结构

- 消息头, UTF-8 BYTES

```
content-length:113<CR><LF>
content-type:json<CR><LF>
<CR><LF>
```

- 消息体

```
[BODY(113Byte)]
```

本协议中 json 字符串传递时使用 utf-8 编码

2. 认证获取 apicode

请求:

```
{
    "__msgid": 1001,
    "clientid":"001",
    "authorization": "aHVnb2JlbjpodWdvYmVuMTIz"        // base64(user:password)
}
```

响应:

```
{
    "__msgid": 1001
    "apicode":"xxxxx",
    "__result": 0,                    // 成功返回 0， 失败-1
    "__msg":"错误信息",               //  异常信息(失败时)
}
```

注意:

这里返回的 apicode 在后面的请求中需要传输到服务端

3. 订阅消息

请求:

```
{
    "__msgid": 1002,
    “apicode":"xxxxx",
    "subscribe":"news"                // 订阅的消息 id
}
```

响应:

```
{
    "__msgid": 1002
    "__result": 0,                    // 成功返回 0， 失败-1
    "__msg":"错误信息",               //  异常信息(失败时)
    "subscribe":"news"                // 订阅的消息 id

}
```

说明:

4. 订阅通道消息

请求:

```
{
    "__msgid": 1003,
    “apicode":"xxxxx",
    "channel":"x001"                    // 订阅的消息通道 id
}
```

响应:

```
{
    "__msgid": 1003
    "__result": 0,                      //  成功返回 0， 失败-1
    "__msg":"错误信息",                 //  异常信息(失败时)
    " channel":" x001"                  //  订阅的消息通道
}
```

通道消息只能被订阅一次(不允许多个连接同时订阅),
通道推送的消息需要响应后才能会被删除。

5. 推送消息

```
{
    "__msgid":"1100",
    "apicode":"xxxxx",
    "type":"news",
    "msg":"要发送的信息"
}
```

响应:

```
{
    "__msgid": 1100
    "__result": 0,                      //  成功返回 0， 失败-1
    "count":5,          // 尝试推送的客户端数量
    "__msg":"错误信息",                 //  异常信息(失败时)
}
```

说明:
type: 推送的消息类型, 与订阅消息的 subscribe 对应

6. 推送通道消息

请求:

```
{
    "__msgid": 1101,
    "apicode":"xxxxx",
    "id":"xxx-xxx-xxx",        //消息 id
    "channel":"x001",          //消息通道
    "msg":"要推送的消息"
}
```

响应:

```
{
    "__msgid": 1101
    "__result": 0,                    //   成功返回 0，  失败-1
    "__msg":"错误信息",               //   异常信息(失败时)
    "id":"xxxxx"                      //  信息 id
}
```

7. 接收到推送消息(客户端)

响应:

```
{
    "__msgid":"1200",
    "type":"news",
    "userid":"001",                   //  发送的用户 id
    "msg":"发送的信息"
}
```

8. 接收到推送通道消息(客户端)

响应:

```
{
    "__msgid":"1201",
    "id":"xxx-xxx-xxx",           //  消息 id
    "channel":"x001"              //  消息通道
    "userid":"001",               //  发送的用户 id
    "msg":"发送的信息"
}
```

9. 接收到推送系统消息(客户端)

```
响应:
{
    "__msgid":"1201",
    "msg":"你的帐号已经在其他地方登陆!"
}
```

10.主动请求通道消息

```
请求:
{
    "__msgid": 1301,
    "apicode":"xxxxx",
    "id":"xxx-xxx-xxx",        //消息 id
    "channel":"x001"           //消息通道
}
```

```
响应:
{
    "__msgid": 1301
    "__result": 0,                  //   成功返回 0，  失败-1
    "__msg":"错误信息",             //   异常信息(失败时)
    "list":
     [
        {
          "id":"xxxxx",                  // 信息 id
          "msg":"消息",
        },
        {
          "id":"xxxxx",                  // 信息 id
          "msg":"消息",
        }
     ]
}
```

11.标志消息已经接收完成 (客户端)

```
请求:
{
    "__msgid": 1302,
    "apicode":"xxxxx",
```

"channel":"x001", "id":"xxxxx" // 信息 id }

响应
{
"__msgid": 1302,
"__result": 0, // 成功返回 0， 失败-1
"__msg":"错误信息", // 异常信息(失败时)
"channel":"x001",
"id":"xxxxx" // 信息 id
}

12.心跳(客户端发起)

请求:
{
"__msgid": 1400
}

响应
{
"__msgid": 1400,
"__result": 0, // 0 正常，其他值不正常
}

备注: 心跳是客户端为了维持与服务端建立长连接，定时发送一个小的消息包，来检测网络是否畅通