服务器与客户端协议

服务器与客户端之间采用websocket交互协议。数据格式采用：命令字+**#**+Json序列化后的字符串。

1. 登录
   1. 用户登录

发送时机：客户端登录服务器

命令方向：客户端->服务器

命令字：LOG

数据结构：List<string>

* 1. 登录结果

发送时机：客户端登录服务器

命令方向：服务器->客户端

命令字：LOG

数据结构：Success/Wrong/Already，如果登录成功返回用户类型

例如： *LOG#* *Wrong*

*LOG#Success#Admin*

*LOG#Success#Control*

* 1. 向客户端返回对应用户的设备组和成员

发送时机：客户端登录服务器成功

命令方向：服务器->客户端

命令字：GroupExt

数据结构：List<GroupData>

struct GroupData

{

string groupid;//0为键权电话，T为中继线路，B为广播电话组

string extid;

}

* 1. 向客户端返回对应用户的中继电话

发送时机：客户端登录服务器成功

命令方向：服务器->客户端

命令字：GroupTrunk

数据结构：List<GroupTrunk>

struct GroupTrunk

{

string trunkid;

string name;

string bindingnumber

}

* 1. 向客户端返回对应用户的广播电话

发送时机：客户端登录服务器成功

命令方向：服务器->客户端

命令字：GroupBroadcast

数据结构：List<GroupBroadcast>

struct GroupBroadcast

{

string name;

List<> bmemberlist

{

string callno;

string name;

}

}

1. 控制命令

CMD

* 1. 呼叫命令

发送时机：点击呼叫按钮

命令方向：客户端->服务器

命令字：CMD#Call

数据结构：

struct call

{

string fromid;

string toid;

}

* 1. 呼叫外线命令

发送时机：点击呼叫按钮

命令方向：客户端->服务器

命令字：CMD#CallOut

数据结构：

struct call

{

string fromid;

string toid;

string trunkid;

}

* 1. 来电转接命令

发送时机：点击呼叫按钮

命令方向：客户端->服务器

命令字：CMD#Visitor

数据结构：

struct call

{

string fromid;

string toid;

}

* 1. 语音菜单呼叫分机命令

发送时机：点击呼叫按钮

命令方向：客户端->服务器

命令字：CMD#MenuToExt

数据结构：

struct call

{

string fromid;//语音菜单号

string toid;//分机号

}

* 1. 语音菜单设置命令

发送时机：点击呼叫按钮

命令方向：客户端->服务器

命令字：CMD#SetMenu

数据结构：

struct

{

string menuid;//语音菜单号

string voicefile;//分机号

}

* 1. 分机组设置命令

发送时机：

命令方向：客户端->服务器

命令字：CMD#AssignGroup

数据结构：

struct

{

string distribution;//呼叫分配规则sequential(顺选)、group（群振）、circular（轮选）

List<> devlist

{

string ext; //分机号,需为同一软交换的有效电话

}

}

* 1. 强插命令

发送时机：点击强插按钮

命令方向：客户端->服务器

命令字：CMD#Bargein

数据结构：

struct call

{

string fromid;

string toid;

}

* 1. 强拆命令

发送时机：点击强拆按钮

命令方向：客户端->服务器

命令字：CMD#Clear

数据结构：string extid

例如：*CMD#Clear#204*

* 1. 监听命令

发送时机：点击监听按钮

命令方向：客户端->服务器

命令字：CMD#Monitor

数据结构：

struct call

{

string fromid;

string toid;

}

* 1. 夜服开启

发送时机：点击强拆按钮

命令方向：客户端->服务器

命令字：CMD#NightServiceOn

数据结构：

struct call

{

string fromid;

string toid;

}

* 1. 夜服关闭

发送时机：点击强拆按钮

命令方向：客户端->服务器

命令字：CMD#NightServiceOff

数据结构：

struct call

{

string fromid;

string toid;

}

* 1. 呼叫保持

发送时机：点击强拆按钮

命令方向：客户端->服务器

命令字：CMD#Hold

数据结构：string extid

* 1. 取消呼叫保持

发送时机：点击强拆按钮

命令方向：客户端->服务器

命令字：CMD#Unhold

数据结构：string extid

* 1. 查询通话记录

发送时机：服务器主报

命令方向：服务器->客户端

命令字：CMD#GETCDR#ALL

数据结构：

应答

命令字：CMD#GETCDR

数据结构：

public struct list<>cdrlist

{

public string Cdrid;//话单id

public string callid;//通话的相对唯一标识

public string type;//通话类型 IN(打入)/OU(打出)/FI(呼叫转移入)/FW(呼叫转 移出)/LO(内部通话)/CB(双向外呼)

public string TimeStart;//呼叫起始时间，即发送或收到呼叫请求的时间

public string TimeEnd;//呼叫结束时间，即通话的一方挂断的时间

public string CPN;//主叫号码

public string CDPN;//被叫号码

public string Duration;//通话时长，值为 0 说明未接通。

}

* 1. 状态查询

发送时机：客户端发起

命令方向：客户端->服务器

命令字：CMD#GETSTATE

数据结构：string extid

1. 状态上报

STATE

* 1. 分机忙

发送时机：服务器主报

命令方向：服务器->客户端

命令字：STATE#BUSY

数据结构：string extid

* 1. 分机空闲

发送时机：服务器主报

命令方向：服务器->客户端

命令字：STATE#IDLE

数据结构：string extid

* 1. 分机上线

发送时机：服务器主报

命令方向：服务器->客户端

命令字：STATE#ONLINE

数据结构：string extid

* 1. 分机离线

发送时机：服务器主报

命令方向：服务器->客户端

命令字：STATE#OFFLINE

数据结构：string extid

* 1. 振铃事件

发送时机：服务器主报

命令方向：服务器->客户端

命令字：STATE#RING

数据结构：

struct call

{

string fromid;

string toid;

}

* 1. 回铃事件

发送时机：服务器主报

命令方向：服务器->客户端

命令字：STATE#ALERT

数据结构：

struct call

{

string fromid;

string toid;

}

* 1. 呼叫应答事件

发送时机：服务器主报

命令方向：服务器->客户端

命令字：STATE#ANSWER

数据结构：

struct call

{

string fromid;

string toid;

}

* 1. 呼叫被应答事件

发送时机：服务器主报

命令方向：服务器->客户端

命令字：STATE#ANSWERED

数据结构：

struct call

{

string fromid;

string toid;

}

* 1. 通话结束事件

发送时机：服务器主报

命令方向：服务器->客户端

命令字：STATE#BYE

数据结构：

struct call

{

string fromid;

string toid;

}

* 1. 呼叫失败事件

发送时机：服务器主报

命令方向：服务器->客户端

命令字：STATE#FAILED

数据结构：

struct call

{

string fromid;

string toid;

string reason;

}

* 1. 呼入事件

发送时机：服务器主报

命令方向：服务器->客户端

命令字：STATE#INVITE

数据结构：

struct callsession

{

string trunkid;

string visitorid;

string fromnumber;

string tonumber;

string called;

}

1. 管理命令

MAN

* 1. 添加软交换设备

发送时机：服务器主报

命令方向：服务器->客户端

命令字：MAN#ADDSW

数据结构：

struct XX

{

string sequence;

string index;

string name;

string ip;

string port;

string type;

string username;

string password;

}

应答

命令字：MAN#ADDSW

数据结构：

struct XX

{

string sequence;

string index;

string result;//成功：Success 失败：Fail

string reason;

}

* 1. 删除软交换设备

发送时机：服务器主报

命令方向：服务器->客户端

命令字：MAN#DELSW

数据结构：

struct XX

{

string sequence;

string index;

}

应答

命令字：MAN#DELSW

数据结构：

struct XX

{

string sequence;

string index;

string result;

string reason;

}

* 1. 修改软交换设备

发送时机：服务器主报

命令方向：服务器->客户端

命令字：MAN#EDITSW

数据结构：

struct XX

{

string sequence;

string index;

string name;

string ip;

string port;

string type;

string username;

string password;

}

应答

命令字：MAN#EDITSW

数据结构：

struct XX

{

string sequence;

string index;

string result;

string reason;

}

* 1. 查询软交换设备列表

发送时机：服务器主报

命令方向：服务器->客户端

命令字：MAN#QUERYSW

数据结构：

struct XX

{

string sequence;

string index;//没用

}

应答

命令字：MAN#QUERYSW

数据结构：

struct XX

{

string sequence;

list< > switchlist

{

string index;

string name;

string ip;

string port;

string type;

string username;

string password;

}

}

* 1. 查询软交换下属设备列表

发送时机：服务器主报

命令方向：服务器->客户端

命令字：MAN#QUERYALLDEV

数据结构：

struct XX

{

string sequence;

string index;

}

应答

命令字：MAN#QUERYALLDEV

数据结构：

struct XX

{

string sequence;

list< >devlist

{

string callno;

string type;

string name;

string level;

string description;

}

}

* 1. 获取软交换下属设备列表

发送时机：服务器主报

命令方向：服务器->客户端

命令字：MAN#GETALLDEV

数据结构：

struct XX

{

string sequence;

string index;

}

应答

命令字：MAN#GETALLDEV

数据结构：

struct XX

{

string sequence;

list<> devlist

{

string callno;

string type;

string name;

string level;

string description;

}

}

* 1. 获取注册设备列表

发送时机：服务器主报

命令方向：服务器->客户端

命令字：MAN#GETALLREGISTERDEV

数据结构：

struct XX

{

string sequence;

}

应答

命令字：MAN#GETALLREGISTERDEV

数据结构：

struct XX

{

string sequence;

list<> devlist

{

string callno;

string type;

string name;

string level;

string description;

}

}

* 1. 修改软交换下属设备列表

发送时机：服务器主报

命令方向：服务器->客户端

命令字：MAN#EDITALLDEV

数据结构：

struct XX

{

string sequence;

string index;

list< > devlist

{

string callno;

string type;

string name;

string level;

string description;

}

}

应答

命令字：MAN#EDITALLDEV

数据结构：

struct XX

{

string sequence;

string index;

string result;

string reason;

}

* 1. 添加调度员

发送时机：服务器主报

命令方向：服务器->客户端

命令字：MAN#ADDUSER

数据结构：

struct XX

{

string sequence;

string name;

string password;

string privilege;

string description;

string status;

string role;

string desk;

}

应答

命令字：MAN# ADDUSER

数据结构：

struct XX

{

string sequence;

string result;

string reason;

}

* 1. 查询调度员

发送时机：服务器主报

命令方向：服务器->客户端

命令字：MAN#GETUSER

数据结构：

Struct

{

string sequence;

}

应答

命令字：MAN# GETUSER

数据结构：

Struct

{

string sequence;

list<>userlist

{

string name;

string password;

string privilege;

string description;

string status;

string role;

string desk;

}

}

* 1. 删除调度员

发送时机：服务器主报

命令方向：服务器->客户端

命令字：MAN#DELUSER

数据结构：

struct XX

{

string sequence;

string name;

}

应答

命令字：MAN# DELUSER

数据结构：

struct XX

{

string sequence;

string result;

string reason;

}

* 1. 修改调度员

发送时机：服务器主报

命令方向：服务器->客户端

命令字：MAN#EDITUSER

数据结构：

struct XX

{

string sequence;

string name;

string password;

string privilege;

string description;

string status;

string role

string desk;

}

应答

命令字：MAN# EDITUSER

数据结构：

struct XX

{

string sequence;

string result;

string reason;

}

* 1. 添加/修改调度键盘

发送时机：服务器主报

命令方向：服务器->客户端

命令字：MAN#ADDKEYBOARD

数据结构：

struct XX

{

string sequence;

string index;

string name;

string mac;

string ip;

List< > grouplist

{

string index;

string groupname;

string column;

string description;

List< > memberlist

{

string callno;

string type;

string name;

string level;

string description;

}

}

List< > hotlinelist

{

string callno;

string type;

string name;

string level;

string description;

}

List<>broadcastlist

{

string index;

string name;

List<>bmemberlist

{

string callno;

string name;

}

}

List<>trunklist

{

string trunkid;

string name;

string bindingnumber;

}

}

应答

命令字：MAN# ADDKEYBOARD

数据结构：

struct XX

{

string sequence;

string index;

string result;

string reason;

}

* 1. 查询所有调度键盘

发送时机：服务器主报

命令方向：服务器->客户端

命令字：MAN#GETALLKEYBOARD

数据结构：

struct XX

{

string sequence;

}

应答

命令字：MAN# GETALLKEYBOARD

数据结构：

struct XX

{

string sequence;

string index;

string name;

string mac;

string ip;

List< > grouplist

{

string index;

string groupname;

string column;

string description;

List< > memberlist

{

string callno;

string type;

string name;

string level;

string description;

}

}

List< > hotlinelist

{

string callno;

string type;

string name;

string level;

string description;

}

List<>broadcastlist

{

string index;

string name;

List<>bmemberlist

{

string callno;

string name;

}

}

List<>trunklist

{

string trunkid;

string name;

string bindingnumber;

}

}

* 1. 删除调度键盘

发送时机：服务器主报

命令方向：服务器->客户端

命令字：MAN#DELKEYBOARD

数据结构：

{

string sequence;

string index;

}

应答

命令字：MAN# DELKEYBOARD

数据结构：

struct XX

{

string sequence;

string result;

string reason;

}

* 1. 获取电话本

发送时机：服务器主报

命令方向：服务器->客户端

命令字：MAN#GETPHONEBOOK

数据结构：

{

string sequence;

}

应答

命令字：MAN# GETPHONEBOOK

数据结构：

struct XX

{

string sequence;

List<>departmentlist

{

string department;

List<>memberlist

{

string callno;

string name;

}

}

}

* 1. 修改电话本

发送时机：服务器主报

命令方向：服务器->客户端

命令字：MAN#EDITPHONEBOOK

数据结构：

{

string sequence;

List<>departmentlist

{

string department;

List<>memberlist

{

string callno;

string name;

}

}

}

应答

命令字：MAN# EDITPHONEBOOK

数据结构：

struct XX

{

string sequence;

string result;

string reason;

}