Một client muốn tham gia vào hệ thống phải kết nối với server thông qua địa chỉ và cổng, sau đó hệ thống sẽ xác thực thông tin user do client gửi lên. Mỗi lệnh gửi lên server phải tuân theo cấu trúc riêng trong giao thức. Trước mỗi lệnh phải có ID của lệnh đó hay gọi là <Tag> để đảm bảo không bị sai, vì server có thể trả về không theo thứ tự lệnh gửi lên nên trước mỗi kết quả trả về đều có ID của lệnh gửi lên tương ứng.

Server có thể tự động trả kết quả về cho client mà không cần có lệnh yêu cầu của client, khi này <Tag> trả về luôn luôn là 0. Đây là các thông tin mà server chủ động gửi về như admin gửi thông báo cho user, phiên bản mới được update,…

Thông tin trao đổi giữa client và server phải tuân theo một cấu trúc bắt buộc. Ta quy định các ký hiệu như sau:

* + - * <Tag>: ID của lệnh
      * <SP> : Dấu cách <space bar>
      * <ABC>: Mã lệnh
      * <xyz’s Length>: Độ dài phần xyz
      * <\r\n>: Dấu xuống dòng mới <phím ENTER>
      * [<xyz>]: Phần <xyz> có thể có hoặc không
* ***Lệnh chứng thực(LOGIN):***

**Client** : <Tag><SP><LOGIN><SP><Header’s Length><\r\n>[<Login struct>]

**Server**: <Tag><SP><LOGIN’s code><SP><Error Code><SP><Comment’s Length> <\r\n><Comment>

Trong đó trường Header’s length chỉ ra kích thước đoạn dữ liệu đằng sau.

Struct Login:

{

char Email[40];

char Pass[40];

};

* ***Lệnhcập nhật phiên bản(NEW\_VERSION):***

**Client** : <Tag><SP><NEW\_VERSION><SP><0><\r\n>

**Server**: <Tag><SP><NEW\_VERSION’s code><SP><Error Code><SP><Header’s Length><\r\n>[<NewVersion>]

* ***Lệnh tự lấy nội dung phiên bản phần mềm mới(AUTO\_UPDATE):***

**Client** : <Tag><SP><AUTO\_UPDATE><SP><0><\r\n>

**Server** :<Tag><SP><AUTO\_UPDATE’s code><SP><Error Code><SP><header’s Length><\r\n>[<NewVersion’s size><decode key>]<file content>

* ***Lệnh lấy thông tin các User trong phòng ban (GET\_USER\_DEP):***

**Client** : <Tag><GET\_USER\_DEPR><SP><0><\r\n>

**Server**: <Tag><GETUSER\_DEP’s Code><SP><Error Code><SP><Header’s length>\r\n[<Số cặp><List UserDep’s infor>]

Trong đó cấu trúc UserDep như sau:

Struct UserDep

{

Char Email[40];

Char DepID[40];

Int KeyUser;

Int chuc\_vu\_id;

Char ten\_chuc\_vu[50];

Int chuc\_nang;

}

* ***Lệnh lấy danh sách User(GET\_LISTUSER):***

**Client**:<Tag><GET\_LISTUSER><SP><0><\r\n>

**Server**:<Tag><GET\_ LISTUSER’s Code><SP><Error Code><SP><Header’s length>\r\n[<Số User><List User’s infor>]

Trong đó: <List User’s infor>:=<User 1>…<UserN>

struct User

{

char Email[40];

char Name[100];

};

* ***Lệnh lấy danh sách nhóm(GET\_LISTGROUP):***

**Client** : <Tag><GET\_LISTGROUP><SP><0><\r\n>

**Server**: <Tag><GET\_ LISTGROUP’s Code><SP><Error Code><SP><Header’s length>\r\n[<Số Group><List Group’s infor>]

Trong đó:<List Group’s infor>:=<Group 1>…<Group N>

Struct Group

{

char GroupName[40];

int GroupID;

Onwer: string

Private: int

};

* ***Lệnh danh sách các User trong các nhóm(GET\_MEM\_GROUP):***

**Client** : <Tag><GET\_MEMGROUP><SP><Header’s length><\r\n>[<GroupID>]

**Server**: <Tag><GET\_ MEMGROUP’s Code><SP><Error Code><SP><Header’s length>\r\n[<Số User><List User’s infor>]

Trong đó:

Struct User

{

char Email[40];

char Name[100];

Keyman

};

* ***Lệnh lấy danh sách các phòng ban(GET\_LISTDEPART):***

**Client** : <Tag><GET\_LISTDEPART><SP><0><\r\n>

**Server**: <Tag><GET\_ LISTDEPART’s Code><SP><Error Code><SP><Header’s length>\r\n[<Số phòng><Department List>]

Trong đó: <Department List>:=<Department 1>…<Department N>

struct Department {

char DeptID[40];

char DeptName[40];

};

* ***Lệnh lấy về vai trò của người tham gia(GET\_ROLE):***

**Client**: <Tag><GET\_ROLE><SP><0><\r\n>

**Server**: <Tag><GET\_ROLE’s code><sp><Error code><SP><Header’s length><\r\n>[<số vai trò có thể><Role list>]

Trong đó: *<Role list>:=<Role1><Role2>…*

Struct Role

{

int RoleID;

char RoleName[50];

int RolePermission;

int WarningType;

}

* ***Lệnh lấy về các tài nguyên(GET\_RESOURCE):***

**Client**:<Tag><GET\_RESOURCE><sp><0><\r\n>

**Server**:<Tag><GET\_RESOURCE’s code><sp><Error code><sp><Header’s length><\r\n>[<số lượng resource><Resource 1><Resource 2>…]

Trong đó:

Struc Resource

{

Char ResourceName[50];

Int ResourceID;

Int Quantity;

Char Description [5000];

Char Location[1000];

Int Status;

Int Blocking;

Int Bookable;

Int AutoAccept;

}

* ***Lệnh lấy danh sách các lịch(LIST\_CLD):***

**Client** : <Tag><SP><LIST\_CLD><SP><Header’s Length><\r\n>[<From><To><Số User><user 1> <user 2> ….<user n>]

**Server** : <Tag><SP><LIST\_CLD’s Code><SP><Error Code><SP><Header’s Length><\r\n>[<SoLich><Lich1><Lich2>…..<LichN>]

Trong đó:

*Lịch:=< CalendarID >< Version >*

*From, To*: Gửi lên dạng String chỉ sổ DWORD ->Ngày tháng

* ***Lệnh lấy nội dung một lịch(GET\_CLD):***

**Client** :<tag><sp><GET\_CLD><sp><header’s length><\r\n>[<số calendar><calendar\_id1><calendar\_id2>…]

**Server** :<tag><sp><GET\_CLD’s code><sp><error code><sp><header’s length><\r\n><số calendar><calendar 1’s content><calendar 2’content>…

Trong đó

Struct Calendar:

{

Int CalendarID;

Int Version;

ColeDatetime StartTime;

ColeDatetime EndTime;

ColeDatetime LastUpdateTime;

ColeDatetime NotifyBeforeStartTime;

Char\* CreatorEmail;

Char\* LatsUpdatePerson;

Char\* CalendarName;

Char\* Description;

Char\* Location;

Int RescureType;

Int Blocking;

Int Priority;

Int Private;

Int Status;

Int Notify;

Int NotifyUpdate;

Int iJoinerNumber;

Int iGroupNumber;

Int iDepartmentNumber;

Int iResourceNumber;

Int iAttachNumber;

Int iTagNumber;

Std::vector<User> lsUser;

Std::vector<Group> lsGroup;

Std::vector<Resource> lsResource

Std::vector<Attach> lsAttach;

Std::vector<Tag> lsTag;

}

Struct User

{

char Email[40];

int Role;

int Status;

char Note[1000]

}

Struct Group

{

int GroupID;

int Role;

int Status;

char Note[1000]

}

Struct Department

{

char DepID[40];

int Role;

int Status;

char Note[1000]

}

Struct Resource

{

Int ResourceID

Int Quantity

Int Status

Char\* Note

}

Struct attach

{

Int AttachID

Int AttachType

Char \* AttachName

Int Status

}

Struct Tag

{

Char\* Tag

}

* ***Lệnh thêm một lịch mới(ADD\_CLD):***

**Client** : <Tag><SP><ADD\_CLD><SP><Header’s length><\r\n>[<Ignor conflict byte><Nội dung lịch>]

**Server** : <Tag><ADD\_CLD’s Code><SP><Error Code><SP><Header’s length><\r\n>[<CalendarID><Số Lịch conflict><LichConflct1>…<LichConflctN> ]

Trong đó:

Nếu có conflict thì CalendarID = 0

Ignore conflict byte:

0: không cho phép conflict

1: bỏ qua conflict

Struct LichConflict

{

Int CalendarID;

Char CalendarName[1000];

COleDateTime StartTime;

ColeDateTime EndTime;

Int RescureType;

Int Blocking;

Int iUserNumber;

Int iResourceNumber;

Std::vector<User> lsUser;

Std::vector<Resource> lsResource;

}

Struct User

{

Char \* email;

Char \*Status;

}

Struct Resource

{

Int ResourceID;

Char \* Status;

}

* ***Lệnh thêm file attach(ADD\_CLD\_ATTACH)***

**Client** : <Tag><SP><ADD\_CLD\_ATTACH><SP><Header’s length><\r\n>[<CalenderID>< AttachType><AttachName><update ?><file size>]<file content>

**Server** : <Tag><ADD\_CLD\_ATTACH’s Code><SP><Error Code><SP><Header’s length><\r\n>[<AttachID><Error description>]

Trong đó:

File size: 4 byte chỉ kích thước của file

Attach type byte:

1:link

2:file

* ***Lệnh xóa file attach(DEL\_CLD\_ATTACH):***

**Client** : <Tag><SP><DEL\_CLD\_ATTACH><SP><Header’s length><\r\n>[<CalenderID><AttachID><update?>]

**Server**: <Tag><DEL\_CLD\_ATTACH’s Code><SP><Error Code><SP><Header’s length><\r\n><Error description>ư

* ***Lệnh lấy nội dung file attach(GET\_CLD\_ATTACH):***

**Client**: <Tag><SP><GET\_CLD\_ATTACH><SP><Header’s Length><\r\n>[<CalendarID><AttachID>]

**Server**: <Tag><GET\_CLD\_ATTACH’s Code><SP><Error Code><SP><Header’s length><\r\n>[<AttachID<Attach’s Size>]<Attach’s content>

* ***Lệnh cập nhật nội dung lịch(UPDATE\_CLD):***

Lệnh có hiệu lực khi mình là người tạo hoặc người sở hữu, bao gồm cả resource

**Client** : <Tag><SP><UPDATE\_CLD><SP><Header’s length><\r\n>[<CalendarID><ignore conflict byte><Calendar’s content>]

**Server**: <Tag><APDATE\_CLD’s Code><SP><Error Code><SP><Header’s length><\r\n>[<Số Lịch conflict><LichConflct1>…<LichConflctN> ]

* ***Lệnh trả lời khi nhận một lời họp(REPLY\_CLD):***

**Client** : <Tag><SP><REPLY\_CLD><SP><Header’s length><\r\n> [<CalendarID><respond byte><Note>]

**Server**: <Tag><REPLY\_CODE’s Code><SP><Error Code><SP><0><\r\n>

* ***Lệnh lấy danh sách các quyết định khi được mời họp(GET\_RESPONSE):***

**Client** : <Tag><SP><GET\_RESPONSE><SP><0><\r\n>

**Server**: <Tag><GET\_RESPONSE’s Code><SP><Error Code><SP><Header’s length><\r\n>[<so response><ResponseID1><ResponseName1><Priority1>….<ResponseIDN><ResponseNameN><PriorityN>]

* ***Lệnh xóa một lịch(DEL\_CLD):***

**Client** : <Tag><SP><DEL\_CLD><SP><Header’s length><\r\n>[<CalendarID>]

**Server**: <Tag><DEL\_CLD’s Code><SP><Error Code><SP><0><\r\n>

* ***Lệnh Notify do server gửi về(NOTIFY\_CALENDAR):***

**Server**: <0><NOTIFY\_CALENDAR’s Code><SP><NotifyID><SP> <Header’s length><\r\n>[<Notify’s content>]

Trong đó:

struct Notify

{

int TypeUpdate; //New, Update, Delete;

int CalendarID;

int GeneralID; // Tùy mục đích mà sử dụng

int TypeChange // Time, men, resource…. change

char \*Description;

};

Một số loại CalendarID:

enum TYPE\_UPDATE

{

CALENDAR\_UPDATE = 8,

CALENDAR\_NEW = 9,

CALENDAR\_REMOVE = 10,

TASK\_NEW = 0, // GeneralID = Notify

TASK\_UPDATE = 1,// GeneralID = Notify

TASK\_REMOVE = 2,// GeneralID = Notify

ATTACH\_TASK\_NEW = 4,// GeneralID = AttachID

ATTACH\_TASK\_REMOVE = 5,// GeneralID =

NEW\_CHAT = 6,// GeneralID = iInstructionID

UPDATE\_REPORT = 7,// GeneralID = ReportID

};