技 术 文 件

技术文件名称：IKEMBX00 网络通讯规约

技术文件编号：

版　　本：　　0.4

拟　制 _____

审　核 _____

会　签 _____

_____

_____

_____

标准化 _____

批　准 _____

北京中科虹霸科技有限公司

版本变更记录

文件编号	版本号	拟制人/修改人	拟制/修改日期	更改理由	主要更改内容 （写要点即可）
	0.1	何颖醛	2013-11-28	无	创建文档
	0.2	何颖醛	2013-12-02	第一次讨论	增加第一次讨论结果及根据讨论结果的修改
	0.3	何颖醛	2014-01-08	第二次讨论	增加第二次讨论结果及根据讨论结果的修改
	0.4	何颖醛	2014-03-20	在实际编解码过程中，发现有需要改进的地方	集群识别请求中，识别参数和需要识别的数据混在一起，本次把这两个参数分开，用不同的请求实现
	0.5	何颖醛	2014-04-23	修订设备工作模式方案	修改特征等数据更新方式；删除联网注册、联网识别控制；增加服务器与设备的交互内容

注 1：每次更改归档文件（指归档发布数据库）时，需填写此表。

注 2：文件第一次归档时，“更改理由”、“主要更改内容”栏写“无”。

目录

1 引言

1.1 编写目的

定义 IKEMBX00 网络通讯的相关约定和活动。本规约规定了服务器端与客户端之间的网络数据传输协议，协议的原则基于此项目要实现的网络通讯功能。

1.2 适用范围

适用于 IKEMBX00 项目网络通讯数据需求开发和管理。

2 术语、定义与缩略语

2.1 术语、定义

术语、定义见表 1

表 1

术语/定义	说明
客户端	IKEMBX00 设备，总是作为客户端出现
服务器端	计算机上，与客户端进行网络通讯的程序
集群识别	IKEMBX00 设备作为采集设备，将采集到的虹膜或提取出的虹膜特征发到指定的集群服务器，完成识别功能。识别结果由集群服务器作为回应反馈给 IKEMBX00 设备。

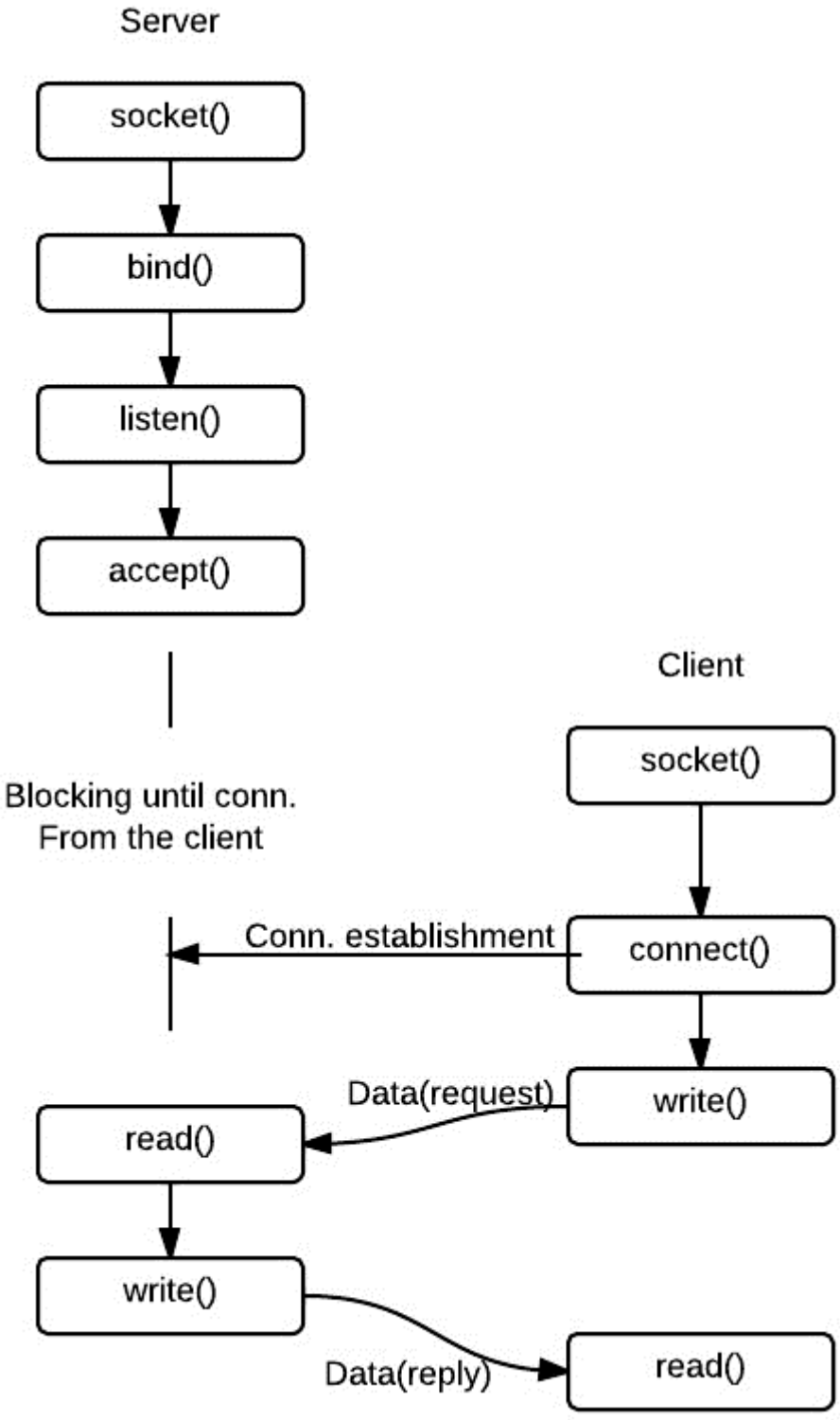
2.2 缩略语

缩略语见表 2。

表 2

缩略语	原文	中文含义
服务器	服务器端	计算机上，与客户端进行网络通讯的程序
UUID	通用惟一标识符 Universally Unique Identifier	128 比特的数字，用来惟一地标识因特网上的某些对象或者实体，UUID 由开放软件基金会(OSF)定义
GUID	全球唯一标识符 Globally Unique Identifier	微软对 UUID 这个标准的实现，UUID 还有其它各种实现。

3 基本 Socket 系统网络通信示意



图表 1 服务器端与客户端通信过程

4 IKEMBX00 网络通讯需求

以下需求摘自《IKEBMX00 系统需求书》

4.1 人员信息、虹膜特征、识别记录导入和导出

4.2 虹膜特征实时更新

4.3 兼容集群服务器

5 IKEMBX00 网络通讯规约

服务器端启动后，监听指定 socket 地址，等待接受请求。

客户端主动与服务器端连接。

连接成功后，双方通讯即是双向的，既可以是客户端向服务器端请求，也可以是服务器端向客户端请求。

5.1 规约基本格式

请求格式：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	(2 Bytes)	0xaaaa (2 Bytes)	传送的数据长度 (4 Bytes)	实际长度	(2 Bytes)

回应格式：

同步头	请求码	请求子码	实际数据长度	错误码	传送的数据	校验
0x5353 (2 Bytes)	(2 Bytes)	0xaaaa (2 Bytes)	（错误码+传送的数据长度） (4 Bytes)	(4Bytes)	实际长度	(2 Bytes)

请求格式和回应格式都可分为三部分：

头信息+数据+校验

三部分对应长度为：

10Bytes+变长+2Bytes

头信息数据结构如下：

```
struct NetCommHeadT
{
    char syncHead[2];       // 0x5353
    short commCode;         // 请求码或回应码，2 字节
    short commSubCode;      // 请求子码，2 字节，本次固定为 0xaaaa
    int32 dataLength;       // 后面实际传送的数据长度，不包括校验字段，4 字节
};
```

其中，

同步头 —— 两个字节，固定为 0x5353

请求码 —— 实际请求码或回应码，参见下面的定义

请求子码——可用于以后的需求扩充，或表示请求的版本号，本次固定为 0xaaaa

实际数据长度 —— 传送的实际数据的字节数。在回应结构中，实际数据长度包括错误码+传送的数据长度

错误码 —— 回应格式中标识本次请求是否成功执行，

0 成功执行

<0 执行结果有错误，错误原因在后面的“传送的数据”中以定长字符串方式说明，错误说明的字符串长度为 260Bytes

传送的数据 —— 实际传送数据

校验 —— 从同步头到传送数据所有数据的校验值。可选加和校验或 CRC 校验，在实际使用中确定用哪种；当数据字段中有图像或特征时，不执行真正的校验，在校验字段随便填写一个数据即可

5.2 规约基本原则

尽量使用 char* 的定长格式。

5.3 发送长度基本规定

网络发送数据，数据长度最小值尽量控制在 1.5K，最大值控制在 1M 左右。这样可以比较好地利用网络资源。

5.4 对齐的约定

使用没有字节间隙方式的对齐。

5.5 常用常量定义

```
const int32 g_nameLen        = 64;          // 名称长度，如姓名、部门名称
const int32 g_snLen          = 64;          // 序号长度，如工号、部门号
const int32 g_errorLen       = 260;         // 错误说明长度
const int32 g_memoLen        = 128;         // 补充说明字符串长度
const int32 g_timeLen        = 20;          // 时间字符串长度
                                            // 时间字符串示例：2013-01-01 00:00:00
const int32 g_cardNoLen      = 16;          // 卡号长度
const int32 g_idCardNoLen    = 20;          // 身份证号长度
const int32 g_versionLen     = 20;          // 版本号长度
const int32 g_enrollFeatureLen= 512;        // 注册特征长度
const int32 g_idenFeatureLen  = 1024;       // 识别特征长度
const int32 g_imageLen        = 307200;     // 虹膜图像纯数据长度，640*480
const int32 g_ipLen           = 16;         // IP 地址长度
```

5.6 常用枚举量

5.6.1 人员类型 EnumPersonType

```
enum EnumPersonType
{
    ordinaryPerson      =1,
    adminPerson         =2
};
```

5.6.2 性别 EnumSex

```
enum EnumSex
{
    male      = 1,
    female    = 2
};
```

5.6.3 眼睛标记 EnumEye

```
enum EnumEye
{
    both      = 0,
    left      = 1,
    right     = 2
};
```

5.6.4 集群服务器数据类型 EnumClusterMode

```
enum EnumClusterMode
{
    all       = 0,
    feature   = 1,
    image     = 2
};
```

5.6.5 识别搜索模式 EnumFindMode

```
enum EnumFindMode
{
    exhaustion = 0,          // 穷尽所有查找
```

```
    stopWhenFind = 1        //找到合适的即停止
};
```

5.6.6 设备工作模式 EnumWorkMode

```
enum EnumWorkMode
{
    online = 0,             // 联网工作模式
    offline=1               // 脱机工作模式
};
```

5.6.7 设备工作状态 EnumWorkStatus

```
enum EnumWorkStatus
{
    invalid=0,              // 无效
    operator=1,             // 操作员控制
    identify=2,             // 识别
    saving=3                // 省电
};
```

5.6.8 获取识别记录类型 EnumGetRecord

```
enum EnumGetRecord
{
    all=0,                  // 所有识别记录
    sended=1,               // 已发送的识别记录
    notyet=2,               // 未发送的识别记录
};
```

5.6.9 令牌类型 EnumTokenType

设备连接信息 DeviceConnInfoT 中包含有设备当前保留的令牌值，共 6 种，3 种在用，3 种保留。

```
enum EnumTokenType
{
    person = 0,             // 人员
    feature = 1,            // 特征
    photo   = 2,            // 照片
    reverved1=3,            // 保留
    reverved2=4             // 保留
    reverved3=5,            // 保留
};
```

5.7 常用数据格式

在结构定义中，为明确变量占用的字节数，以 int8 表示 8 位整数，int16 表示 16 位整数，以 int32 表示 32 位整数。

通讯中常用数据结构及存储格式定义如下：

5.7.1 名称定长字符串 NameString

名称字符串 NameString
长度= g_nameLen

如人名、部门名称等使用名称定长字符串类型。

5.7.2 序号定长字符串 SnString

序号字符串 SnString
长度= g_snLen

如人员工号、部门序号等使用序号定长字符串类型。

5.7.3 错误说明定长字符串 ErrorString

错误说明字符串 ErrorString
长度= g_errorLen

请求响应的错误说明使用错误说明定长字符串类型。

5.7.4 注册特征 EnrollBinary

注册特征 EnrollBinary
长度= g_enrollFeatureLen

5.7.5 识别特征 RecogBinary

识别特征 RecogBinary
长度= g_identifyFeatureLen

5.7.6 虹膜图像 IrisImageBinary

虹膜图像 IrisImageBinary
长度= g_imageLen

5.7.7 IP 字符串 IpString

Ip 字符串 IpString
长度= g_ipLen

5.7.8 版本字符串 VersionString

版本字符串 VersionString
长度= g_versionLen

5.7.9 时间字符串 TimeString

时间字符串 TimeString
长度= g_timeLen

5.7.10 变长字符串 VarString

字符串长度	实际字符串
4Byte	实际长度

5.7.11 变长二进制数据串 VarBinary

数据串长度	实际数据串
4Byte	实际长度

5.7.12 简要人员信息 PersonSimpleT

```
struct PersonSimpleT
{
    UUID        id;             // 人员 ID
    int32       hasFeature;     // 是否带有注册特征
    NameString  name;           // 人员姓名
    SnString    workSn;         // 工号
};
```

共 148Bytes

id	hasFeature	name	workSn
16Bytes UUID	4Bytes	64Bytes	64Bytes

相关请求 ACK_GET_PERSON_LIST。

5.7.13 人员 PersonBaseT

struct PersonBaseT

{

UUID id; // 人员 ID，UUID

int32 opToken; // 操作令牌，在服务器端每次操作（增、删、改，如果有的话）均有一个对应增长的 int 累计值作为操作令牌，用于与设备上同步各类数据，目前包括人员、特征和人员照片。此值在从服务器往设备发送数据时有效，在设备往服务器发送时无效，置为 0 即可

NameString name; // 人员姓名

SnString workSn; // 人员工号

EnumPersonType type; // 人员类型

EnumSex sex; // 性别

char idCardNumber[g_idCardNoLen]; // 身份证号

NameString departName; // 部门名称

char cardNumber[g_cardNoLen]; // 卡号

char memo[g_memoLen]; // 备注

};

共 384Bytes

id	opToken	name	workSn	type	sex	idCardNumber	departName	cardNumber	memo
16Bytes	4Bytes	NameString	SnString	4Bytes	4Bytes	g_idCardNoLen	NameString	g_cardNoLen	g_memoLen

相关请求 REQ_ADD_PERSON_INFO、REQ_ADD_ADMIN_INFO。

5.7.14 单幅人员照片 PersonPhotoT

struct PersonPhotoT

{

UUID personId; // 人员 ID

UUID photoId; // 照片 ID

int32 opToken; // 操作令牌，在服务器端每次操作（增、删、改，如果有的话）均有一个对应增长的 int 累计值作为操作令牌，用于与设备上同步各类数据，

目前包括人员、特征和人员照片。此值在从服务器往设备发送数据时有效，在设备往服务器发送时无效，置为 0 即可

```
    char        memo[g_memoLen];    // 备注
    int32       photoSize           // photo 的数据长度，以字节为单位
    VarBinary   photo;              // 照片，变长
};
```

数据结构变长，取决于照片大小。按照目前的要求，照片数据很有可能能是完整的 BMP 格式文件。

要求：人员照片数据小于 20KBytes。

相关请求 REQ_ADD_PERSON_PHOTO。

5.7.15 BMP 文件格式

5.7.15.1 BMP 文件头格式 BmpFileHeaderT

```
struct BmpFileHeaderT
{
    int16   type;           // 位图文件类型，’BM’
    int32   bfSize;         // 位图文件大小
    int16   bfReserved1;    // 保留字，必须为 0
    int16   bfReserved2;    // 保留字，必须为 0
    int32   bfOffBits;      // 位图数据的起始位置，以相对于文件头的偏移量表示
                            // 以字节为单位
};
```

共 14 字节

5.7.15.2 BMP 信息头格式 BmpInfoHeaderT

```
struct BmpInfoHeaderT
{
    int32   biSize;         // 本结构所占字节数
    int32   biWidth;        // 图像宽度
    int32   biHeight;       // 图像高度
```

```
    int16     biPlanes;          // 必须为 1
    int16     biBitCount;        // 图像中每个像素位数，在虹膜图像中，目前必须是 8
    int32     biCompression;     // 必须是 0
    int32     biImageSize;       // 位图大小，以字节为单位
    int32     biXPelsPerMeter;
    int32     biYPelsPerMeter;
    int32     biClrUsed;         // 图像使用的颜色表中的颜色数
    int32     biclrImportant;
};
```

共 40 字节。

5.7.15.3 颜色表项数据结构 RGBQuadT

颜色表用于说明图像中的颜色，每个表项是一个 RGBQuadT 结构，定义一种颜色的表现形式。对于虹膜图像来说，目前颜色表有 256 个表项，对应颜色数值从 0~255 的颜色表现形式。

```
struct RGBQuadT
{
    int8      rgbBlue;           // 蓝色的亮度（取值范围 0~255）
    int8      rgbGreen;          // 绿色的亮度（取值范围 0~255）
    int8      regRed;            // 红色的亮度（取值范围 0~255）
    int8      rgbReserved;       // 保留，必须为 1
};
```

共 4 字节。

5.7.15.4 BMP 信息格式 BmpInfoT

是 Bmp 信息头格式+所有用到的颜色表。对于虹膜图像来说，有 256 个颜色表，因此格式如下：

```
struct BmpInfoT
{
    BmpInfoHeaderT      bmiHeader;
    RGBQuadT            bmiColors[256];
};
```

共 1064 字节。

5.7.16 虹膜图像 IrisImageT

```
struct IrisImageT
{
    BmpFileHeaderT      bmiFileHeader;    // BMP 文件头
    BmpInfoT            bmiInfo;          // BMP 信息（信息头+颜色表）
    IrisImageBinary     irisImageRaw;     //  原始虹膜图像
};
```

共 308278 字节。

虹膜图像是符合 BMP 格式的图像数据。

5.7.17 单幅注册结果 IrisEnrollInfoT

单幅图像注册结果中各个值的意义请参考算法提供的虹膜图像结构信息。

```
struct IrisEnrollInfoT
{
    int32           pupilRow;
    int32           pupilCol;
    int32           pubilRadius;
    int32           irisRow;
    int32           irisCol;
    int32           irisRadius;
    int32           focusScore;
    int32           percetVisible;
    int32           spoofValue;
    int32           interlaceValue;
    int32           qualityLevel;
    int32           qualityScore;
    EnrollBinary    irisEnrollTemplate;     // 注册特征
    RecogBinary     irisRecogTemplate;      // 识别特征
```

```
    IrisImageT    irisImage;          // 图像数据都是固定带调色板的 BMP 格式数据
                                      // 图像本身大小仍为 640*480
};
```

数据定长，309862Bytes。

5.7.18 上传单幅虹膜注册数据 UploadIrisDataT

只有从设备到服务器才可能以上传虹膜注册数据的格式发送注册数据。

```
struct UploadIrisDataT
{
    SnString      deviceSn;     // 设备序列号，标识是在哪台设备上注册的，
                                // 应为唯一值，此值目前应为取到的设备 MAC 地址
    UUID          featureId;    // 16Bytes
    UUID          personId;     // 16Bytes
    EnumEye       eyeFlag;      // 标识是哪只眼睛
    char          enrollTime[g_timeLen]; // 注册时间
    IrisEnrollInfoT irisEnrollInfo; // 单幅注册结果数据
};
```

共 309982Bytes

deviceSn	featureId	personId	eyeFlag	enrollTime	irisImage
SnString	16Bytes UUID	16Bytes UUID	4Bytes	20Bytes	ImageBinary

相关请求 REQ_UPLOAD_PERSON_IRIS。

5.7.19 同步虹膜特征数据 SyncIrisDataT

用于服务器向设备更新虹膜特征。

服务器向设备更新虹膜特征的方式不再区分设备处于脱机工作模式，还是联网工作模式。设备收到特征后，通过查找 featureId 在本地数据库中是否存在来判断这个特征是本地数据库已经存储的，还是未存储的。

```
struct SyncIrisDataT
{
```

UUID featureId; // 16Bytes

UUID personId; // 16Bytes

int32 opToken; // 操作令牌，在服务器端每次操作（增、删、改，如果有的话）均有一个对应增长的 int 累计值作为操作令牌，用于与设备上同步各类数据，目前包括人员、特征和人员照片。此值在从服务器往设备发送数据时有效，在设备往服务器发送时无效，置为 0 即可

EnumEye eyeFlag; // 标识是哪只眼睛

EnrollBinary irisFeature; // 注册特征

};

共 552 字节。

featureId	personId	opToken	eyeFlag	irisFeature
16Bytes UUID	16Bytes UUID	4Bytes	4Bytes	EnrollBinary

相关请求 REQ_ADD_PERSON_IRIS。

5.7.20 识别记录 RecogRecordT

struct RecogRecordT

{

SnSting deviceSn; // 设备序列号，标识是在哪台设备上识别的，

// 应为唯一值，此值目前未确定

UUID featureId; // 特征 ID

UUID personId; // 人员 ID

int32 recogId; // 在设备本机数据库中的识别记录序列号，连续且自增长

EnumEye eyeFlag; // 标识是哪只眼睛

char idenTime[g_timeLen]; // 识别时间

};

共 120Bytes

deviceSn	featureId	personId	recogId	eyeFlag	idenTime
SnString	16Bytes UUID	16Bytes UUID	4Bytes	4Bytes	20Bytes

相关请求 REQ_UPLOAD_RECOG_RECORD，REQ_SEND_RECOG_RECORD，

ACK_GET_RECOG_RECORD。

5.7.21 集群识别相关数据结构

5.7.21.1 图像集群识别单条数据 ClusterImageItemT

struct ClusterImageItemT

{

int32 id; // 由发送方自定义的 ID 号，便于与识别结果对照

EnumEye eyeFlag; // 标识是哪只眼睛

IrisImageT irisImage; // 用于识别的虹膜图像，BMP 文件格式

};

数据定长，为 308286Bytes

id	eyeFlag	irisImage
4Bytes	4Bytes	308278Bytes

相关请求 REQ_CLUSTER_IMAGE_IDENTIFY。

5.7.21.2 特征集群识别单条数据 ClusterFeatureItemT

struct ClusterFeatureItemT

{

int32 id; // 由发送方自定义的 ID 号，便于与识别结果对照

EnumEye eyeFlag; // 标识是哪只眼睛

RecogBinary irisFeature; // 用于识别的虹膜特征

};

数据定长，为 1028Bytes

id	eyeFlag	RecogBinary
4Bytes	4Bytes	1024Bytes

相关请求 REQ_CLUSTER_FEATURE_IDENTIFY。

5.7.21.3 图像集群识别控制参数 ClusterImageIdenConfigT

struct ClusterImageIdenConfigT

{

```
    EnumFindMode    findMode;       // 识别结果查找模式，穷尽或找到即停
    int32           maxMatchNum;    // 最多匹配个数，findMode=0 时，
                                    // 返回的匹配结果最多发多少个
};
```

数据定长，为 8Bytes。

findMode	maxMatchNum
4Bytes	4Bytes

相关请求 REQ_CLUSTER_IMAGE_IDEN_CONFIG。

5.7.21.4 特征集群识别控制参数 ClusterFeatureIdenConfigT

```
struct ClusterFeatureIdenConfigT
{
    EnumFindMode    findMode;       // 识别结果查找模式，穷尽或找到即停
    int32       maxMatchNum;        // 最多匹配个数，findMode=0 时，
                                    // 返回的匹配结果最多发多少个
    char        version[g_versionLen]; // 客户端算法版本号
};
```

数据定长，为 28Bytes。

findMode	maxMatchNum	version
4Bytes	4Bytes	20Bytes

相关请求 REQ_CLUSTER_FEATURE_IDEN_CONFIG。

5.7.21.5 集群识别结果单条数据 ClusterIdenResultT

```
struct ClusterIdenResultT
{
    int32           id;             // 请求集群识别结构中的由发送方自定义的 ID 号
    UUID            personId;
    NameString      personName;
    SnString        workSn;
    int32           score;          // 匹配分数，单位 0.001
}
```

数据定长，152Bytes

id	personId	personName	workSn	score
4Bytes	16Bytes UUID	64Bytes	64Bytes	4Bytes

相关请求 ACK_CLUSTER_IMAGE_IDNENTIFY。

5.7.22 IP 地址设置 IpSettingT

```
struct IpSettingT
{
    IpString   ip;                  // IP
    IpString subnetMask;            // 子网掩码
    IpString gateWay;               // 网关
}
```

数据定长，48Bytes

ip	subnetMask	gateWay
IpString	IpString	IpString

相关请求 REQ_SRVCTRL_ADJUST_DEVIP。

5.7.23 侦听地址设置 ListenIpT

```
struct ListenIpT
{
    IpString   ip;                  // 侦听地址
    int32      port;                // 侦听端口
};
```

数据定长，18Bytes

ip	port
IpString	4Bytes

相关请求 REQ_SRVCTRL_ADJUST_SRVIP。

5.7.24 设备连接信息 DeviceConnInfoT

```
struct DeviceConnInfoT
{
    IpString        ip;             // 设备 IP
    SnString        devSn;          // 设备序列号
    int             token[6];       // 设备上保留的令牌值，预留供 6 种，现使用 3 种
};
```

数据定长，104Bytes。

ip	devSn	token[6]
16Bytes	64Bytes	6 种 token，现在使用 3 种。类型索引见 EnumTokenType

5.7.25 设备信息 DeviceInfoT

```
struct DeviceInfoT
{
    IpString            ip;             // IP
    EnumWorkMode        mode;           // 工作模式，联网或脱机
    EnumWorkStatus      status;         // 工作状态，无、操作员、识别、省电
    SnString            devSn;          // 设备序列号
    VersionString       version;        // 软件版本号
    TimeString          time;           // 设备当前时间
};
```

数据定长，128Bytes

ip	mode	status	devSn	version	time
16Bytes	4Bytes	4Bytes	64Bytes	20Bytes	20Bytes

相关请求 ACK_ GET_DEV_INFO。

5.7.26 获得指定识别记录 GetRecogRecordT

```
struct GetRecogRecordT
```

```
{
    EnumGetRecord       getType;            // 获取类型，全部，已发送，未发送
    int                 startTimeValid;     // 起始时间是否有效，
                                            // 0——无效，1——有效
    int                 endTimeValid ;      // 结束时间是否有效，
                                            // 0——无效，1——有效
    TimeString          startTime;          // 起始时间
    TimeString          endTime;            // 截止时间
};
```

数据定长，52Bytes

getType	startTimeValid	endTimeValid	startTime	endTime
4Bytes	4Bytes	4Bytes	20Bytes	20Bytes

相关请求 REQ_GET_RECOG_RECORD。

5.8 请求列表

请求中文名称	请求英文名称	请求码	请求说明	请求流向
连接相关				
连接测试（心跳请求）	REQ_KEEP_ALIVE	0x0001	服务器测试与设备的连接状况	服务器->客户端
响应连接测试	ACK_KEEP_ALIVE	0x4001	回应连接状况测试请求	客户端->服务器
数据操作令牌相关				
设备同步服务器数据操作令牌	REQ_SYNC_TOKEN	0x0101	设备向服务器要求同步数据操作令牌	客户端->服务器
响应设备同步服务器数据操作令牌	ACK_ SYNC_TOKEN	0x4101	回应设备向服务器要求同步数据操作令牌请求	服务器->客户端
设备连接参数	REQ_DEV_CONNECT	0x0121	设备与服务器连接成功后，向服务器发送标识本设备当前情况的请求	客户端->服务器
响应设备连接参数	ACK_DEV_CONNECT	0x4121	服务器回应设备连接参数请求	服务器->客户端
人员信息相关				
增加人员信息	REQ_ADD_PERSON_INFO	0x0201	增加人员信息	双向
响应增加人员信息	ACK_ADD_PERSON_INFO	0x4201	回应增加人员信息请求	双向
修改人员信息	REQ_UPDATE_PERSON_INFO	0x0202	修改人员信息	双向
响应修改人员信息	ACK_UPDATE_PERSON_INFO	0x4202	回应修改人员信息请求	双向
删除人员信息	REQ_DELETE_PERSON_INFO	0x0203	删除人员信息	双向
响应删除人员信息	ACK_DELETE_PERSON_INFO	0x4203	回应删除人员信息请求	双向
获得人员列表	REQ_GET_PERSON_LIST	0x0211	获得人员列表	服务器->客户端
响应获得人员列表	ACK_GET_PERSON_LIST	0x4211	响应获得人员列表	客户端->服务器端
增加人员照片	REQ_ADD_PERSON_PHOTO	0x0204	增加人员照片	双向
响应增加人员照片	ACK_ADD_PERSON_PHOTO	0x4204	回应增加人员照片请求	双向
修改人员照片	REQ_UPDATE_PERSON_PHOTO	0x0205	修改人员照片	双向
响应修改人员照片	ACK_UPDATE_PERSON_PHOTO	0x4205	回应修改人员照片请求	双向
删除人员照片	REQ_DELETE_PERSON_PHOTO	0x0206	删除人员照片	双向
响应删除人员照片	ACK_DELETE_PERSON_PHOTO	0x4206	回应删除人员照片请求	双向
注意：现在系统实现中，将人员和管理员的管理合在一起，不通过请求码来区分				
增加管理员信息	REQ_ADD_ADMIN_INFO	0x0221	增加管理员信息	双向
响应增加管理员信息	ACK_ADD_ADMIN_INFO	0x4221	回应增加管理员信息请求	双向
修改管理员信息	REQ_UPDATE_ADMIN_INFO	0x0222	修改管理员信息	双向
响应修改管理员信息	ACK_UPDATE_ADMIN_INFO	0x4222	回应修改管理员信息请求	双向

删除管理员信息	REQ_DELETE_ADMIN_INFO	0x0223	删除管理员信息	双向
响应删除管理员信息	ACK_DELETE_ADMIN_INFO	0x4223	回应删除管理员信息请求	双向
增加管理员照片	REQ_ADD_ADMIN_PHOTO	0x0224	增加管理员照片	双向
响应增加管理员照片	ACK_ADD_ADMIN_PHOTO	0x4224	回应增加管理员照片请求	双向
修改管理员照片	REQ_UPDATE_ADMIN_PHOTO	0x0225	修改管理员照片	双向
响应修改管理员照片	ACK_UPDATE_ADMIN_PHOTO	0x4225	回应修改管理员照片请求	双向
删除管理员照片	REQ_DELETE_ADMIN_PHOTO	0x0226	删除管理员照片	双向
响应删除管理员照片	ACK_DELETE_ADMIN_PHOTO	0x4226	回应删除管理员照片请求	双向
虹膜特征更新相关（服务器端向客户端更新虹膜特征）				
增加虹膜特征	REQ_ ADD_PERSON_IRIS	0x0301	增加虹膜数据	服务器端->客户端
响应增加虹膜特征	ACK_ ADD_PERSON_IRIS	0x4301	回应增加虹膜数据请求	客户端->服务器端
删除虹膜特征	REQ_ DELETE_PERSON_IRIS	0x0302	删除虹膜数据	服务器端->客户端
响应删除虹膜特征	ACK_ DELETE_PERSON_IRIS	0x4302	回应删除虹膜数据请求	客户端->服务器端
注意：现在系统实现中，将人员和管理员的管理合在一起，不通过请求码来区分				
增加管理员虹膜特征	REQ_ ADD _ADMIN_IRIS	0x0311	增加管理员虹膜特征	服务器端->客户端
响应增加管理员虹膜特征	ACK_ ADD _ADMIN_IRIS	0x4311	回应增加管理员虹膜特征请求	客户端->服务器端
删除管理员虹膜特征	REQ_ DELETE _ADMIN_IRIS	0x0312	删除管理员虹膜特征	服务器端->客户端
响应删除管理员虹膜特征	ACK_ DELETE_ADMIN_IRIS	0x4312	回应删除管理员虹膜特征请求	客户端->服务器端
客户端向服务器端上传虹膜注册数据	REQ_UPLOAD_PERSON_IRIS	0x0321	客户端向服务器端上传客户端注册的虹膜数据	客户端->服务器端
响应客户端向服务器端上传虹膜注册数据	ACK_UPLOAD_PERSON_IRIS	0x4321	响应客户端向服务器端上传在客户端注册的注册数据请求	服务器端->客户端
识别记录相关				
上传识别记录	REQ_UPLOAD_RECOG_RECORD	0x0401	上传识别记录	客户端->服务器端
响应上传识别记录	ACK_UPLOAD_RECOG_RECORD	0x4401	回应上传识别记录请求	服务器端->客户端
实时发送识别记录	REQ_SEND_RECOG_RECORD	0x0421	实时发送识别记录	客户端->服务器端
响应实时发送识别记录	ACK_SEND_RECOG_RECORD	0x4421	回应实时发送识别记录请求	服务器端->客户端
发送指定识别记录	REQ_GET_RECOG_RECORD	0x0441	获得设备指定识别记录	服务器端->客户端
响应发送指定识别记录	ACK_GET_RECOG_RECORD	0x4441	响应获得设备指定识别记录请求	客户端->服务器端
服务器控制相关				
远程控制关机	REQ_SRVCTRL_SHUTDOWN	0x0661	服务器控制客户端关机	服务器端->客户端

响应远程控制关机	ACK_SRVCTRL_ SHUTDOWN	0x4661	回应服务器控制客户端关机请求	客户端->服务器端
远程控制重启	REQ_SRVCTRL_RESET	0x0662	服务器控制客户端重启	服务器端->客户端
响应远程控制重启	ACK_SRVCTRL_ RESET	0x4662	回应服务器控制客户端重启请求	客户端->服务器端
远程控制时间调整	REQ_SRVCTRL_ADJUST_TIME	0x0663	服务器控制客户端调整时间	服务器端->客户端
响应远程控制时间调整	ACK_SRVCTRL_ ADJUST_TIME	0x4663	回应服务器控制客户端调整时间请求	客户端->服务器端
远程控制修改设备 IP	REQ_SRVCTRL_ADJUST_DEVIP	0x0664	服务器控制修改设备 IP	服务器端->客户端
响应远程控制修改设备 IP	ACK_SRVCTRL_ ADJUST_DEVIP	0x4664	回应服务器控制修改设备 IP	客户端->服务器端
远程控制修改设备服务器 IP	REQ_SRVCTRL_ADJUST_SRVIP	0x0665	服务器控制修改设备对应的服务器 IP	服务器端->客户端
响应远程控制修改设备服务器 IP	ACK_SRVCTRL_ ADJUST_SRVIP	0x4665	回应服务器控制修改设备对应的服务器 IP 请求	客户端->服务器端
获得设备信息	REQ_GET_DEV_INFO	0x0671	服务器查询设备信息	服务器端->客户端
响应获得设备信息	ACK_GET_DEV_INFO	0x4671	回应服务器查询设备信息请求	客户端->服务器端
设备获得服务器时间	REQ_GET_SRV_TIME	0x0681	设备请求获得服务器当前时间	客户端->服务器端
响应设备获得服务器时间	ACK_GET_SRV_TIME	0x4681	回应设备要求获得服务器当前时间请求	服务器端->客户端
集群识别* （有星号标记的暂时不做）				
图像方式集群识别	REQ_CLUSTER_IMAGE_IDENTIFY	0x0701	发送图像实现集群识别	客户端->集群服务
响应图像方式集群识别	ACK_CLUSTER_IMAGE_IDENTIFY	0x4701	回应发送图像实现集群识别请求	集群服务->客户端
设置图像方式集群识别控制参数	REQ_CLUSTER_IMAGE_IDEN_CONFIG	0x0711	设置集群服务端图像识别时的识别控制参数	客户端->集群服务
响应设置图像方式集群识别控制参数	ACK_CLUSTER_IMAGE_IDEN_CONFIG	0x4711	回应设置集群服务端图像识别时的识别控制参数	集群服务->客户端
特征方式集群识别	REQ_CLUSTER_FEATURE_IDENTIFY	0x0711	发送特征实现集群识别	客户端->集群服务
响应特征方式集群识别	ACK_CLUSTER_FEATURE_IDENTIFY	0x4711	回应发送特征实现集群识别请求	集群服务->客户端
设置特征方式集群识别控制参数	REQ_CLUSTER_FEATURE_IDEN_CONFIG	0x0712		客户端->集群服务
响应设置特征方式集群识别控制参数	ACK_CLUSTER_FEATURE_IDEN_CONFIG	0x4712		集群服务->客户端

5.9 规约具体格式

注 1：本节列出请求的具体格式，本次可能不实现的请求没有在这里列出，可在以后需要的时候补充。

注 2：本节只列出请求名称，其对应的请求码请参见“请求列表”。

注 3：请求子码固定为 0xaaaa。

注 4：服务器发送大数据量请求才需要等待对方回应（比如人员、特征和照片）；等到回应后才继续发送新的请求（也是需要等待回应的请求）。

注 5：如果是数据量小的请求，随时发送即可，且不需要等待回应。

5.9.1 一般回应格式

大部分请求的回应帧格式如没有特殊说明，则均按照如下形式。

请求帧：

如果错误码=0，表示对应的请求成功，格式为

同步头	请求码	请求子码	实际数据长度	错误码	传送的数据	校验
0x5353 (2 Bytes)	 (2 Bytes)	0xaaaa (2 Bytes)	4 (4 Bytes)	0 (4Bytes)	（无数据）	 (2 Bytes)

请求帧字节长度固定为：16Bytes。

如果错误码<0，表示对应的请求失败，并给出错误描述，格式为

同步头	请求码	请求子码	实际数据长度	错误码	传送的数据	校验
0x5353 (2 Bytes)	 (2 Bytes)	0xaaaa (2 Bytes)	264 (4 Bytes)	 (4Bytes)	ErrorString (260Bytes)	 (2 Bytes)

请求帧字节长度固定为：276Bytes。

5.9.2 连接测试（心跳）请求

请求方向：服务器端→客户端

此请求用于测试服务器与设备之间是否仍存在有效连接。

请求名称：REQ_KEEP_ALIVE

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353		0xaaaa	0	（无数据）	

(2 Bytes)	(2 Bytes)	(2 Bytes)	(4 Bytes)		(2 Bytes)

请求帧字节长度固定为：12Bytes。

5.9.3 响应连接测试（响应心跳）

请求方向：服务器端←客户端

此请求回应连接测试（心跳）请求，表示服务器与设备之间的连接仍旧有效。

请求名称：ACK_KEEP_ALIVE

请求帧格式参见“一般回应格式”。

5.9.4 数据操作令牌相关请求及响应

5.9.4.1 设备同步服务器数据操作令牌请求

请求方向：客户端→服务器端。

此请求用于设备向服务器请求数据操作令牌同步。

请求名称：REQ_SYNC_TOKEN

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	 (2 Bytes)	0xaaaa (2 Bytes)	4 (4 Bytes)	token (4Bytes)	 (2 Bytes)

请求帧字节长度固定为：16Bytes。

其中，传送的数据’token’为设备上目前保留的操作令牌值。

5.9.4.2 响应设备同步服务器数据操作令牌

请求方向：客户端←服务器端。

此请求用于回应设备向服务器请求数据操作令牌同步的请求，可能成功或失败。

请求名称：ACK_ SYNC_TOKEN

请求帧格式参见“一般回应格式”。

如果回应成功，则后面可能会有以下三种可能

操作	请求	原因
无	无	服务器与设备令牌已经同步
发送单个人员或特征或人员照片	人员或特征或照片请求	服务器与设备令牌不同步，但二者间数据相差不大(<=10)

开始批量发送同步数据	REQ_SYNC_MASS_BEGIN	服务器与设备令牌不同步，但二者间数据相差较大(>10)，因此进行批量数据同步，同步结束后，有批量同步结束请求

5.9.4.3 设备连接参数请求

请求方向：客户端→服务器端。

此请求用于客户端通知服务器端，设备连接参数及设备当前保留的令牌值。

请求名称：REQ_DEV_CONNECT

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	(2 Bytes)	0xaaaa (2 Bytes)	104 (4 Bytes)	设备连接信息 DevConnInfoT (104Bytes)	(2 Bytes)

请求帧字节长度固定为：116Bytes。

设备连接信息请参见 DevConnInfoT。

5.9.4.4 响应设备连接参数

请求方向：客户端←服务器端。

此请求用于设备通知的连接参数请求，可能成功或失败。

请求名称：ACK_ DEV_CONNECT

请求帧格式参见“一般回应格式”。

5.9.5 人员信息相关请求及响应

5.9.5.1 增加人员信息请求

请求方向：服务器端↔客户端，任意一方均可发起此请求。

此请求用于向接收方增加新的人员信息。

请求名称：REQ_ADD_PERSON_INFO

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	(2 Bytes)	0xaaaa (2 Bytes)	传送的数据长度 (4 Bytes)	人员信息数组 (4Bytes+332Bytes*个数)	(2 Bytes)

请求帧字节长度不是固定值，以传送的实际数据长度为准。

单个人员的数据结构参见 PersonBaseT。

此请求中，PersonBaseT 数据结构中的 personType= ordinaryPerson。

实际传送的数据结构为：

typedef list<PersonBaseT> lstPerson;

len	PersonBaseT	PersonBaseT	PersonBaseT	……
4Bytes	384Bytes	332Bytes	332Bytes	……

其中，len 表示后面的人员数组长度，即有多少个人员的数据。

5.9.5.2 响应增加人员信息

请求方向：服务器端↔客户端，任意一方均可响应。

此请求用于此请求用于回应增加人员信息的请求，可能成功或失败。

请求名称：ACK_ADD_PERSON_INFO

请求帧格式参见“一般回应格式”。

5.9.5.3 修改人员信息请求

请求方向：服务器端↔客户端，任意一方均可发起此请求。

此请求用于向接收方修改已有的人员信息。

请求名称：REQ_UPDATE_PERSON_INFO

请求帧参见“增加人员信息请求”。

以 PersonBaseT 结构中的 id 作为人员的唯一标识，即修改的哪个人的信息。

5.9.5.4 响应修改人员信息

请求方向：服务器端↔客户端，任意一方均可响应。

此请求用于回应修改已有的人员信息的请求，可能成功或失败。

请求名称：ACK_UPDATE_PERSON_INFO

请求帧参见“响应增加人员信息”。

5.9.5.5 删除人员信息请求

请求方向：服务器端↔客户端，任意一方均可发起此请求。

此请求用于向接收方删除已有的人员信息。

请求名称：REQ_DELETE_PERSON_INFO

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353		0xaaaa	传送的数据长度	人员 UUID 数组	

(2 Bytes)	(2 Bytes)	(2 Bytes)	(4 Bytes)	(4Bytes+16Bytes*个数)	(2 Bytes)

请求帧字节长度不是固定值，以传送的实际数据长度为准。

实际传送的数据结构为：

typedef list<UUID> lstPersonId;

len	id	id	……
4Bytes	16Bytes UUID	16Bytes UUID	……

其中，len 表示后面的人员 UUID 长度，即有多少个人员的 UUID。

5.9.5.6 响应删除人员信息

请求方向：服务器端↔客户端，任意一方均可响应。

此请求用于回应删除人员信息的请求，可能成功或失败。

请求名称：ACK_DELETE_PERSON_INFO

请求帧格式参见“一般回应格式”。

5.9.5.7 获得人员列表

请求方向：服务器端→客户端。

此请求用于服务器向设备获取当前设备上加载的人员信息。

请求名称：REQ_GET_PERSON_LIST

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	 (2 Bytes)	0xaaaa (2 Bytes)	0 (4 Bytes)	无 0Bytes	 (2 Bytes)

请求帧字节长度固定为：12Bytes。

5.9.5.8 响应获得人员列表

请求方向：服务器端→客户端。

此请求用于设备响应服务器获取当前设备上加载的人员信息。

请求名称：ACK_GET_PERSON_LIST

请求帧：

当错误码为 0 时（表示请求成功执行，回应中包括人员的信息）

同步头	请求码	请求子码	实际数据长度	错误码	传送的数据	校验
0x5353 (2 Bytes)	 (2 Bytes)	0xaaaa (2 Bytes)	传送的数据长度 (4 Bytes)	0 (4Bytes)	PersonSimpleT 数组 (4+148Bytes*个数)	 (2 Bytes)

请求帧字节长度不是固定值，以传送的实际数据长度为准。

实际传送的数据结构为：

typedef list<PersonSimpleT> lstPerson;

len	PersonSimpleT	PersonSimpleT	……
4Bytes	148Bytes	148Bytes	……

其中，len 表示后面的人员数据长度，即有多少个人员的数据。

其中人员信息的数据结构参见 PersonSimpleT。

当错误码不等于 0，表示请求未成功执行，请求帧格式参见“一般回应格式”。

5.9.5.9 增加人员照片请求

请求方向：服务器端↔客户端，任意一方均可发起此请求。

此请求用于向接收方增加新的人员照片，照片数据是按人单独发送的，即一次请求只发送一个人的照片。

请求名称：REQ_ADD_PERSON_PHOTO

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	(2 Bytes)	0xaaaa (2 Bytes)	传送的数据长度 (4 Bytes)	单个人员照片信息 （变长）	(2 Bytes)

请求帧字节长度不是固定值，以传送的实际数据长度为准。

单个人员照片数据结构参见 PersonPhotoT 的定义。

5.9.5.10 响应增加人员照片

请求方向：服务器端↔客户端，任意一方均可响应。

此请求用于回应增加人员照片的请求，可能成功或失败。

请求名称：ACK_ADD_ PERSON_PHOTO

请求帧格式参见“一般回应格式”。

5.9.5.11 修改人员照片请求

请求方向：服务器端↔客户端，任意一方均可发起此请求。

此请求用于向接收方更新已有人员照片，照片数据是按人单独发送的，即一次请求只发送一个人的照片。

请求名称：REQ_UPDATE_PERSON_PHOTO

请求帧参见“增加人员照片请求”。

以 PersonPhotoT 结构中的 personId 作为唯一标志，表示修改的是哪个人的照片。

5.9.5.12 响应修改人员照片

请求方向：服务器端↔客户端，任意一方均可响应。

此请求用于回应修改人员照片的请求，可能成功或失败。

请求名称：ACK_ UPDATE_PERSON_PHOTO

请求帧格式参见“一般回应格式”。

5.9.5.13 删除人员照片请求

请求方向：服务器端↔客户端，任意一方均可发起此请求。

此请求用于向接收方删除已有人员照片，可以一次删除多个人员的照片。

请求名称：REQ_DELETE_PERSON_PHOTO

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	(2 Bytes)	0xaaaa (2 Bytes)	传送的数据长度 (4 Bytes)	人员 UUID 数组 (4Bytes+16Bytes*数组长度)	(2 Bytes)

请求帧字节长度不是固定值，以传送的实际数据长度为准。

5.9.5.14 响应删除人员照片

请求方向：服务器端↔客户端，任意一方均可响应。

此请求用于回应删除人员照片的请求，可能成功或失败。

请求名称：ACK_ DELETE_PERSON_PHOTO

请求帧格式参见“一般回应格式”。

5.9.6 管理员信息相关请求及响应

请注意：目前系统中将管理员信息和照片的请求及响应的处理合并到人员信息请求及响应处理，以数据结构中的 personType 来区分。没有以请求来区分。

5.9.6.1 管理员信息的相关请求及响应

包括增加、修改及删除管理员信息；

包括对增加、修改及删除管理员信息的响应。

请求帧格式请参见对应的对普通人员的操作请求及响应。

此类请求中，PersonBaseT 数据结构中的 personType= adminPerson。

请求码列表如下：

增加管理员信息请求	REQ_ADD_ADMIN_INFO
响应增加管理员信息	ACK_ADD_ADMIN_INFO
修改管理员信息请求	REQ_UPDATE_ADMIN_INFO

响应修改管理员信息	ACK_UPDATE_ADMIN_INFO
删除管理员信息请求	REQ_DELETE_ADMIN_INFO
响应删除管理员信息	ACK_DELETE_ADMIN_INFO

5.9.6.2 管理员照片的相关请求及响应

包括增加、修改及删除管理员照片；

包括对增加、修改及删除管理员照片的响应。

请求帧格式请参见对应的对普通人员照片的操作请求及响应。

增加管理员照片请求	REQ_ADD_ADMIN_PHOTO
响应增加管理员照片	ACK_ADD_ADMIN_PHOTO
修改管理员照片请求	REQ_UPDATE_ADMIN_PHOTO
响应修改管理员照片	ACK_UPDATE_ADMIN_PHOTO
删除管理员照片请求	REQ_DELETE_ADMIN_PHOTO
响应删除管理员照片	ACK_DELETE_ADMIN_PHOTO

5.9.7 虹膜特征更新相关请求及响应

对设备来说，虹膜特征更新会在本机数据库中保留，并更新到本机内存中。

5.9.7.1 增加虹膜特征请求

请求方向：服务器端→客户端。

此请求用于服务器端向客户端请求增加联网虹膜特征。

请求名称：REQ_ ADD_PERSON_IRIS

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	(2 Bytes)	0xaaaa (2 Bytes)	传送的数据长度 (4 Bytes)	实时更新虹膜特征数组 (4Bytes+552Bytes*个数)	(2 Bytes)

请求帧字节长度不是固定值，以传送的实际数据长度为准。

单个实时更新虹膜特征的数据结构参见 SyncIrisDataT。

实际传送的数据结构为：

typedef list<SyncIrisDataT> lstSyncIrisData;

len	SyncIrisDataT	SyncIrisDataT	SyncIrisDataT	……

4Bytes	552Bytes	552Bytes	552Bytes	……

其中，len 表示后面的虹膜特征数组长度，即有多少个虹膜特征的数据。

5.9.7.2 响应增加虹膜特征

请求方向：客户端→服务器端。

此请求用于回应增加虹膜特征的请求，可能成功或失败。如果成功，表明客户端已经将发来的虹膜特征增加到内存虹膜特征中。

请求名称：ACK_ ADD_PERSON_IRIS

请求帧格式参见“一般回应格式”。

5.9.7.3 删除虹膜特征请求

请求方向：服务器端→客户端。

此请求用于服务器端向客户端请求删除联网虹膜特征。

请求名称：REQ_ DELETE_PERSON_IRIS

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	(2 Bytes)	0xaaaa (2 Bytes)	传送的数据长度 (4 Bytes)	虹膜特征 UUID 数组 (4Bytes+16Byts*个数)	(2 Bytes)

请求帧字节长度不是固定值，以传送的实际数据长度为准。

5.9.7.4 响应删除虹膜特征

请求方向：客户端→服务器端。

此请求用于客户端回应删除联网虹膜特征请求，可能成功或失败。如果成功，表明客户端已经删除了要求删除的虹膜。

请求名称：ACK_ DELETE_PERSON_IRIS

请求帧格式参见“一般回应格式”。

5.9.8 管理员虹膜特征相关请求及响应

请注意：目前系统中将管理员特征的请求及响应的处理合并到人员特征请求及响应处理。没有以请求来区分。

包括增加及删除管理员虹膜特征；

包括对增加及删除管理员虹膜特征的响应。

请求帧格式请参见对应的对普通人员虹膜特征的请求及响应。

请求码列表如下：

增加管理员虹膜特征请求	REQ_ ADD_ADMIN_IRIS
响应增加管理员虹膜特征	ACK_ ADD_ADMIN_IRIS
删除管理员虹膜特征请求	REQ_ DELETE_ADMIN_IRIS
响应删除管理员虹膜特征	ACK_ DELETE_ADMIN_IRIS

5.9.9 上传虹膜注册数据请求及响应

5.9.9.1 上传虹膜注册数据请求

请求方向：客户端→服务器端

此请求用于客户端向服务器端上传在本地注册的虹膜特征。此请求的虹膜数据中包含注册特征、注册图像等全部的虹膜注册信息。

请求名称：REQ_UPLOAD_PERSON_IRIS

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	(2 Bytes)	0xaaaa (2 Bytes)	传送的数据长度 (4 Bytes)	UploadIrisDataT 数组 (4Bytes+309982Byts* 个数)	(2 Bytes)

请求帧字节长度不是固定值，以传送的实际数据长度为准。

单幅上传虹膜注册数据的结构参见 UploadIrisDataT。

实际传送的数据结构为：

typedef list<UploadIrisDataT> lstUploadIrisData;

len	UploadIrisDataT	UploadIrisDataT	UploadIrisDataT	……
4Bytes	309982Bytes	309982Bytes	309982Bytes	……

其中，len 表示后面的虹膜注册数据数组长度，即有多少个虹膜注册数据。

5.9.9.2 响应上传虹膜注册数据

请求方向：服务器端→客户端。

此请求用于服务器端回应上传虹膜注册数据请求，可能成功或失败。如果成功，表明服务器端已经向数据库中保存了之前上传的虹膜注册数据。

请求名称：ACK_UPLOAD_PERSON_IRIS

请求帧格式参见“一般回应格式”。

5.9.10 识别记录相关请求及响应

5.9.10.1 上传识别记录请求

请求方向：客户端→服务器端

此请求用于客户端向服务器端上传设备上的识别记录。

请求名称：REQ_UPLOAD_RECOG_RECORD

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	(2 Bytes)	0xaaaa (2 Bytes)	传送的数据长度 (4 Bytes)	识别记录 RecogRecordT 录数组 (4Bytes+120Bytes*个数)	(2 Bytes)

请求帧字节长度不是固定值，以传送的实际数据长度为准。

单独一条识别记录的数据结构参见 RecogRecordT。

实际传送的数据结构为：

typedef list<RecogRecordT> lstRecogRecord;

len	RecogRecordT	RecogRecordT	RecogRecordT	……
4Bytes	120Bytes	120Bytes	120Bytes	……

5.9.10.2 响应上传识别记录

请求方向：服务器端→客户端

此请求用于服务器端向客户端响应识别记录上传请求，可能成功或失败。如果成功，则表明之前上传请求中列出的所有识别记录都已经被服务器端处理（例如，保存到数据库中）。

请求名称：ACK_UPLOAD_RECOG_RECORD

请求帧格式参见“一般回应格式”。

5.9.10.3 实时发送识别记录请求

请求方向：客户端→服务器端

此请求用于客户端向服务器端实时上传设备上的识别记录。在联网模式下，设备会向服务器端实时上传识别记录。

请求名称：REQ_SEND_RECOG_RECORD

请求帧格式参见“上传识别记录请求”中的帧格式。

5.9.10.4 响应实时发送识别记录

请求方向：服务器端→客户端

此请求用于服务器端向客户端响应识别记录实时发送请求，可能成功或失败。如果成功，则表明之前上传请求中列出的所有识别记录都已经被服务器端处理（例如，保存到数据库中）。

请求名称：ACK_SEND_RECOG_RECORD

请求帧格式参见“一般回应格式”。

5.9.10.5 获得识别记录请求

请求方向：服务器端→客户端

此请求用于服务器端要求客户端上传指定的识别记录。

请求名称：REQ_GET_RECOG_RECORD

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	 (2 Bytes)	0xaaaa (2 Bytes)	52 (4 Bytes)	GetRecogRecordT (52Bytes)	 (2 Bytes)

请求帧字节长度固定为 64Bytes。

获得指定识别记录请求的数据结构参见 GetRecogRecordT。

5.9.10.6 响应获得识别记录

请求方向：客户端→服务器端

此请求用于客户端响应服务器端要求获得指定识别记录的请求，可能成功或失败。如果成功，则回应中包含符合要求的识别记录（个数可能为 0）。

请求名称：ACK_GET_RECOG_RECORD

请求帧：

如果响应成功，错误码为 0，发送数据为符合要求的识别记录数组

同步头	请求码	请求子码	实际数据长度	错误码	传送的数据	校验
0x5353 (2 Bytes)	 (2 Bytes)	0xaaaa (2 Bytes)	4 (4 Bytes)	0 (4Bytes)	ClusterIdenResultT 数组 (4+152Bytes*个数)	 (2 Bytes)

请求帧字节长度不是固定值，以传送的实际数据长度为准。

单独一条识别记录的数据结构参见 RecogRecordT。

如果响应不成功，错误码不等于 0，请求帧格式参见“一般回应格式”。

5.9.11 集群识别相关请求及回应

5.9.11.1 设置图像方式集群识别控制参数请求

请求方向：客户端→集群服务端

此请求用于客户端向集群服务端发出设置图像集群识别控制参数请求。

请求名称：REQ_CLUSTER_IMAGE_IDEN_CONFIG

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	(2 Bytes)	0xaaaa (2 Bytes)	8 (4 Bytes)	ClusterImageIdenConfig (8Bytes)	(2 Bytes)

请求帧定长，为 20Bytes。

图像方式集群识别请求控制参数数据结构参见 ClusterImageIdenConfigT。

5.9.11.2 响应设置图像方式集群识别控制参数

请求方向：客户端←集群服务端

此请求用于集群服务器端响应客户端设置图像集群识别控制参数的请求。

请求名称：ACK_ CLUSTER_IMAGE_IDEN_CONFIG

请求帧格式参见“一般回应格式”。

5.9.11.3 图像方式集群识别请求

请求方向：客户端→集群服务端

此请求用于客户端向集群服务端发出以图像方式进行识别的请求。

请求名称：REQ_CLUSTER_IMAGE_IDENTIFY

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	(2 Bytes)	0xaaaa (2 Bytes)	传送的数据长度 (4 Bytes)	ClusterImageItemT 数组 (308286Bytes*个数)	(2 Bytes)

请求帧字节长度不是固定值，以传送的实际数据长度为准。

图像集群识别的单条数据结构参见 ClusterImageItemT。

5.9.11.4 响应图像方式集群识别

请求方向：客户端←集群服务端

此请求用于集群服务端回应以图像方式进行识别的请求，给出识别结果。

请求名称：ACK_CLUSTER_IMAGE_IDENTIFY

请求帧：

如果本次成功识别，错误码=0

同步头	请求码	请求子码	实际数据长度	错误码	传送的数据	校验
0x5353 (2 Bytes)	(2 Bytes)	0xaaaa (2 Bytes)	4 (4 Bytes)	0 (4Bytes)	ClusterIdenResultT 数组 (4+152Bytes*个数)	(2 Bytes)

请求帧字节长度不是固定值，以传送的实际数据长度为准。

如果本次识别出错，错误码<0

同步头	请求码	请求子码	实际数据长度	错误码	传送的数据	校验
0x5353 (2 Bytes)	(2 Bytes)	0xaaaa (2 Bytes)	4 (4 Bytes)	(4Bytes)	错误描述字符串 (260Bytes)	(2 Bytes)

注：如果仅仅是没有匹配成功的数据，错误码=0，ClusterIdenResultT 数组长度为 0。这种情况不算请求出错。

单条图像方式集群识别结果的数据结构参见 ClusterIdenResultT。

识别成功时，实际传送数据结构为：

typedef　list< ClusterIdenResultT > lstResult;

回应为匹配成功的人员列表，每个 id 的识别结果列表最大不超过 maxMatchNum，每个 id 的识别结果按匹配份数降序排列。

len	ClusterIdenResultT	ClusterIdenResultT	ClusterIdenResultT	……
4Bytes	152Bytes	152Bytes	152Bytes	……

5.9.11.5　设置特征方式集群识别控制参数请求

请求方向：客户端→集群服务端

此请求用于客户端向集群服务端发出设置特征集群识别控制参数请求。

请求名称：REQ_CLUSTER_FEATURE_IDEN_CONFIG

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	(2 Bytes)	0xaaaa (2 Bytes)	28 (4 Bytes)	ClusterFeatureIdenConfig (28Bytes)	(2 Bytes)

请求帧定长，为 40Bytes。

图像方式集群识别请求控制参数数据结构参见 ClusterFeatureIdenConfigT。

5.9.11.6　响应设置图像方式集群识别控制参数

请求方向：客户端←集群服务端

此请求用于集群服务器端响应客户端设置特征集群识别控制参数的请求。

请求名称：ACK_ CLUSTER_FEATURE_IDEN_CONFIG

请求帧格式参见“一般回应格式”。

5.9.11.7 特征方式集群识别请求

请求方向：客户端→集群服务端

此请求用于客户端向集群服务端发出以特征方式进行识别的请求。

请求名称：REQ_CLUSTER_FEATURE_IDENTIFY

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	(2 Bytes)	0xaaaa (2 Bytes)	4 (4 Bytes)	ClusterFeatureItemT 数组 (520Bytes*个数)	(2 Bytes)

请求帧字节长度不是固定值，以传送的实际数据长度为准。

特征集群识别的单条数据结构参见 ClusterFeatureItemT。

5.9.11.8 响应特征方式集群识别

请求方向：客户端←集群服务端

此请求用于集群服务端回应以图像方式进行识别的请求，给出识别结果。

请求名称：ACK_CLUSTER_FEATURE_IDENTIFY

请求帧参见“响应图像方式集群识别”。

5.9.12 服务器控制客户端相关请求及响应

5.9.12.1 远程控制关机及重启相关请求及响应

请求方向：服务器端→客户端

响应方向：客户端→服务器端

此类请求意义通过请求名称即可了解。

请求名称，请查阅请求列表。

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	(2 Bytes)	0xaaaa (2 Bytes)	0 (4 Bytes)	无 (0Bytes)	(2 Bytes)

请求帧字节长度固定为：12Bytes。

响应的请求帧格式参见“一般回应格式”。

5.9.12.2 远程控制调整时间请求

请求方向：服务器端→客户端

服务器端通过此请求可调整客户端时间。

请求名称：REQ_SRVCTRL_ADJUST_TIME

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	 (2 Bytes)	0xaaaa (2 Bytes)	20 (4 Bytes)	时间字符串 (g_timeLen Bytes)	 (2 Bytes)

请求帧字节长度固定为：32Bytes。

时间字符串格式：YYYY-MM-DD hh:mm:ss，不足两位的数字用 0 补齐。

5.9.12.3 响应远程控制调整时间

请求方向：服务器端←客户端。

此请求用于客户端回应远程控制调整时间请求。

请求名称：ACK_SRVCTRL_ADJUST_TIME

请求帧格式参见“一般回应格式”。

5.9.12.4 远程控制修改设备 IP 请求

请求方向：服务器端→客户端

服务器端通过此请求调整客户端 IP。

请求名称：REQ_SRVCTRL_ADJUST_DEVIP

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	 (2 Bytes)	0xaaaa (2 Bytes)	48 (4 Bytes)	IpSettingT (48Bytes)	 (2 Bytes)

请求帧字节长度固定，为 60Bytes。

其中 IP 设置数据结构参见 IpSettingT 。

5.9.12.5 响应远程控制修改设备 IP

请求方向：服务器端←客户端。

此请求用于客户端回应远程控制调整设备 IP 请求。

请求名称：ACK_SRVCTRL_ADJUST_DEVIP

请求帧格式参见“一般回应格式”。

5.9.12.6 远程控制修改设备服务器 IP 请求

请求方向：服务器端→客户端

服务器端通过此请求调整客户端连接的服务器 IP。

请求名称：REQ_SRVCTRL_ADJUST_SRVIP

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	 (2 Bytes)	0xaaaa (2 Bytes)	18 (4 Bytes)	ListenIpT (18Bytes)	 (2 Bytes)

请求帧字节长度固定，为 30Bytes。

其中 IP 设置数据结构参见 ListenIpT 。

5.9.12.7 响应远程控制修改设备服务器 IP

请求方向：服务器端←客户端。

此请求用于客户端回应远程控制调整设备连接的服务器 IP 请求。

请求名称：ACK_SRVCTRL_ADJUST_SRVIP

请求帧格式参见“一般回应格式”。

5.9.12.8 获得设备信息请求

请求方向：服务器端→客户端

服务器端通过此请求获得设备一些当前的基本信息。

请求名称：REQ_GET_DEV_INFO

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	 (2 Bytes)	0xaaaa (2 Bytes)	0 (4 Bytes)	无 (0Bytes)	 (2 Bytes)

请求帧字节长度固定，为 12Bytes。

5.9.12.9 响应获得设备信息

请求方向：服务器端←客户端。

此请求用于客户端回应服务器获得设备信息请求。

请求名称：ACK_ GET_DEV_INFO

请求帧：

当错误码为 0 时（表示请求成功执行，回应中包括设备的信息）

同步头	请求码	请求子码	实际数据长度	错误码	传送的数据	校验
0x5353		0xaaaa	132	0	设备信息 DeviceInfoT	

(2 Bytes)	(2 Bytes)	(2 Bytes)	(4 Bytes)	(4Bytes)	(128Bytes)	(2 Bytes)

请求帧字节长度固定，为 144Bytes。

其中设备信息数据结构参见 DeviceInfoT。

当错误码为其他值时，表示请求执行有错误，回应中包括错误信息。其请求帧格式参见“一般回应格式”。

5.9.12.10 获得服务器时间请求

请求方向：客户端→服务器端

客户端通过此请求获得服务器当前时间。

请求名称：REQ_GET_SRV_TIME

请求帧：

同步头	请求码	请求子码	实际数据长度	传送的数据	校验
0x5353 (2 Bytes)	(2 Bytes)	0xaaaa (2 Bytes)	0 (4 Bytes)	无 (0Bytes)	(2 Bytes)

请求帧字节长度固定，为 12Bytes。

5.9.12.11 响应获得服务器时间

请求方向：客户端←服务器端。

此请求用于服务器端回应设备端获得服务器时间请求。

请求名称：ACK_ GET_SRV_TIME

请求帧：

当错误码为 0 时（表示请求成功执行，回应中包括服务器当前时间）

同步头	请求码	请求子码	实际数据长度	错误码	传送的数据	校验
0x5353 (2 Bytes)	(2 Bytes)	0xaaaa (2 Bytes)	24 (4 Bytes)	0 (4Bytes)	服务器时间 TimeString (20Bytes)	(2 Bytes)

请求帧字节长度固定，为 36Bytes。

当错误码为其他值时，表示请求执行有错误，回应中包括错误信息。其请求帧格式参见“一般回应格式”。

6 讨论修改意见

6.1 2013-12-02 第一次讨论修改意见

1、 请求码和请求子码均改为 2 个字节

2、 本版本请求子码选择一个独特的数据，如 0xaaaa，而不用 0

3、 请求头中，请求序数字段可删除

4、 增加心跳请求，一般是服务器端发往客户端的

5、 虹膜特征导入/导出与虹膜特征实时更新请求分开

6、 请求码最好按照功能分组，功能相近的分在一组，便于以后增加新请求，请求码不至于太混乱

7、 对齐方式明确为没有字节间隙方式的对齐

8、 需求——设备上的按钮功能都可以通过服务器方操作——是否要在本次实现？但此需求在之前的设计文档及需求文档上似乎都没有体现？

9、 校验可选择加和方式校验或 CRC 校验。但本次确定涉及到图片、特征等的请求不做真正的校验，保留校验位，填写一个无意义的数据即可

10、 实时更新特征前，增加一个握手请求，便于设备准备好接收实时更新的数据

11、 增加发送照片的请求（从人员更新请求中分离出发送照片的请求），且照片信息是一个人一个人地发，不是一起发的

12、 更新虹膜特征请求分成三个请求：增加、更新和删除虹膜特征

13、 唯一标识符是 GUID，不是 UUID

14、 人员数据结构中的卡号长度设置为 16 个字节

15、 删除虹膜请求按照 featureId 指定删除对象

16、 注册特征定长为 512；识别特征定长为 1024；虹膜图像定长为 640*480

17、 增加虹膜特征时，如果 featureId 已存在，则返回错误，标识哪些 featureId 已存在

18、 更新识别记录请求结构中增加识别记录 ID

19、 识别结果 eyeFlag 标志保留 0——双眼；1——左眼；2——右眼，与识别 API 的标识保持一致

20、 文档中的“集成模式”有误，应为“集群模式”

21、 集群识别请求拆分为：特征比对请求和图像比对请求

22、 网络规约中到底字符串用定长方式，还是变长方式？

23、 集群识别方式是否要在本次实现？

6.2 2014-01-08 第二次讨论修改意见

1、 明确网络上传输的图像数据都是带调色板的完整 BMP 数据，且图像的实际大小还是固定为 640*480。

2、 根据 1，修改网络上传输的图像数据格式和大小。

3、 去掉需求中，在服务器端远程控制设备一切操作的要求，改为只实现联网识别和联网注册。

4、 提出设备工作模式为：脱机工作模式和联网工作模式两种。

5、 脱机模式下，设备行为类似一个用于门禁的设备。设备识别时，已注册特征使用设备上数据库中的已注册特征，从服务器更新来的特征数据不直接进入特征内存区，而是保存到设备的本地数据库中。待服务器发指令要求设备更新特征时，设备再从本地数据库中读取特征。脱机模式下，识别记录保存在设备本地数据库，不需要实时向服务器上传识别记录。脱机模式下，注册结果保存在设备本地数据库中。

6、 联网模式下，设备行为类似现在的 220A 设备。设备识别时，已注册特征使用从服务器发来的已注册特征，从服务器更新来的特征数据直接进入特征内存区，且特征不保存到设备的本地数据库中（暂定）。联网模式下，识别记录保存在设备本地数据库，并会实时向服务器上传识别记录。联网模式下，注册结果发给服务器，由服务器处理注册结果。

7、 脱机模式和联网模式是本次新提出来的一个比较新的概念，而且在本次讨论中已形成了比较一致的意见。对本次会议的讨论结果有另文介绍，这类文档保存在概要设计文档中，详情请查阅概要设计文档，并以概要设计文档为准。

8、 由于 4 的明确，需要增加控制设备进入联网、退出联网模式的请求；增加重新加载特征请求（脱机模式下）。

9、 增加注册端的概念，设备与注册端之间有新增加的请求。

10、 对服务端对设备的控制不需要预先设置“进入远程控制”，去掉远程控制这种模式，而是使用“联网工作模式”这样的概念。

11、 把可以向用户开放的和不向客户开放的请求分开，可以向用户开放的写在前面，便于后期挑选可给用户的内容。

12、 去掉对虹膜特征修改的请求，虹膜特征的操作，只有增加和删除

13、 唯一标识改为 UUID。

14、

7 主要修改内容

7.1 第一次讨论后主要修改内容

本次修改完成时间：2014-01-02

1、 按照“2013-12-02 第一次讨论修改意见”修改

2、 所有的更新操作均分解为：增加、修改、删除三个操作

3、 对普通人员和管理员操作分开

4、 特征更新和特征实时更新分开

5、 增加服务端对客户端的控制操作

7.2 第二次讨论后主要修改内容

本次修改完成时间：2014-01-19

1、 去掉联网方式下对管理员特征的更新请求

2、 修改网络上传输的图像类型为 BMP 格式

3、 按照讨论中商议的联网和脱机工作模式，整理出相关请求

4、 将请求分类，加深一级目录，避免目录过长，不易查找

5、 唯一标识符是 UUID，不是 GUID

7.3 2014-03-20 主要修改内容

1、 集群识别请求，将发送的图像（特征）数据与识别控制参数分开。因此增加了请求（设置识别控制参数），修改了数据结构

2、 所有部门 id 是 int 类型，不是 UUID 类型

7.4 2014-04-23 主要修改内容

1、 设备工作模式切换及不同模式下工作方式方案修改，删除远程控制设备注册、识别等功能；

2、 完善服务器与设备间的人员、特征同步方案，采用操作令牌方式，修改相应请求

3、 删除部门交互请求

4、 增加服务器与设备间数据交互内容，包括服务器获取设备上装载人员、服务器获取设备信息、设备获得服务器时间等