Version 0.50

路边停车服务器端通信协议

1. 概述 1

1.1. 目的 1
1.2. 阅读对象 1

2. 协议概述 2

2.1. 基础协议 2
2.2. 协议中通信方式 2
2.3. 协议基本数据格式 2

3. 系统相关协议 5

3.1. 客户端软件注册 5
3.1.1. 数据格式 5
3.1.2. 注册请求 6
3.1.3. 注册请求处理结果 6
3.1.4. 注册确认 7
3.1.5. 注册确认处理结果 7
3.2. 认证码获取 7
3.3. 客户端在线信号 9

4. 用户相关协议 10

4.1. 用户类型信息获取 10
4.2. 用户列表下载 11
4.3. 用户详细信息下载 13
4.4. 用户信息上传 15
4.5. 删除用户 17
4.6. 用户登录登出 18
4.7. 用户密码修改 19

5. 传感器信息传递协议 21

5.1. 客户端主动从服务器获取车辆进出信息 21
5.2. 以 IP 广播形式发送车辆进出信息到客户端 23

6. 应用相关协议 24

6.1. 车辆拍照信息 24
6.2. 车辆登记信息下载 25
6.3. 车辆计费信息 26

6.4. 图片上传下载 28

7. DETAILED REVISION HISTORY 30

CHANGES FROM VERSION 0.40 TO VERSION 0.50 30
CHANGES FROM VERSION 0.30 TO VERSION 0.40 30
CHANGES FROM VERSION 0.20 TO VERSION 0.30 31
CHANGES FROM VERSION 0.10 TO VERSION 0.20 31
FIRST VERSION 0.10 31

1. 概述

1.1. 目的

为方便路边停车 App 的移动设备端与 Web 服务器端之间的通信，制定该协议。

1.2. 阅读对象

路边停车 App 移动设备端开发人员。

路边停车服务器端开发人员。

2. 协议概述

2.1. 基础协议

本协议是建立在 HTTP 协议 1.1 版本(Hypertext Transfer Protocol -- HTTP/1.1)基础之上，制定的应用层协议；HTTP 协议 1.1 版本的具体标准参见 rfc2616 技术文档。

本协议的数据按照 JSON 数据交换格式组织；JSON 数据交换格式的标准参见 rfc4627 技术文档。

本协议中的文本数据编码格式按照 UTF-8 编码格式进行编码；UTF-8 的标准参见 rfc3629 技术文档。

本协议中涉及到的非文本数据按照 BASE64 编码格式编码成文本，再作为文本进行传输；BASE64 的相关标准参加 rfc1421、rfc2045、rfc822 技术文档。

传感器信息相关的通信由于其特殊性，不遵循以上基础协议，具体参见传感器通信相关章节。

2.2. 协议中通信方式

本协议中，通信的双方为路边停车服务器端软件和路边停车 App 移动设备端软件，分别简称为“服务器”和“客户端”。

通信中，总是由客户端主动发起请求，服务器端返回响应结果的方式进行通信。

客户端采用 POST 请求向服务器提交客户端的数据，服务器收到 POST 请求并处理后返回结果；客户端也可以采用 GET 请求获取服务器端的处理结果。

2.3. 协议基本数据格式

通信中，如果是在 HTTP 协议层出现通信错误，按照 HTTP 错误码标准返回错误原因；如果在 HTTP 层未出现通信错误，则按照如下的数据格式请求和响应数据。

POST 的数据基本格式：

```
{
    "client": "0001",
    "serial": "1234567890",
    "credit": " E2AF32258C4B1ADE398FFBD7E89FFFE271A0EEB5 ",
    "type": "enter",
    "content":
    {
        // Data to be transferred
    }
}
```

在客户端通过 POST 方法提交的数据中，均按照以上的 JSON 封装格式提交数据。其中 client 既可以为客户端机器码，也可以为客户端编号(十进制的数字串)，客户端编号由服务器一次性分配，并保持不变，直到服务器管理员手动修改才会变更。如果恰好存在一个客户端的机器码为十进制数字串，并且

该数字恰好与另一个客户端的编号相同，则需要服务器管理员手动调整客户端编号，避开类似的歧义。

serial 为客户端自定义的编号，并且只能用数字、字母和下划线，字母不区分大小写；serial 的长度理论上不受限制。无论 POST 是否成功，任何两次 POST 请求中的 serial 必须不相同。

credit 为服务器端分配给客户端的认证码，该认证码由服务器分配给客户端，作为客户端合法 POST 请求的凭证，同一个认证码的可使用次数和有效时间是有限制的，该限制由服务器设置并管理(默认值是非常大)，客户端一般无需关心。

鉴于在不同应用环境中对安全的要求不同，服务器对认证码的验证是一个可选过程。通常情况下，服务器不对认证码的合法性进行验证，即 credit 可以为任意值。在特别重视安全的应用环境中，如果要验证认证码的合法性，需要由系统管理员对服务器进行设置。

type 为提交的数据类型，服务器根据该类型可以更容易辨别如何处理客户端的请求，具体的类型字段在其后的内容中逐一说明。

如果服务器端收到多于一次 client 和 serial 均相同的 POST 请求，将视为重复 POST 请求，将只接纳首次可识别的请求(服务器收到 credit 号码最后一个引号并且该 credit 号码是合法的视为可识别)，其余的重复请求均被拒绝。

服务器得到 POST 数据后，进行处理，然后返回处理结果，同时将处理结果保存，供客户端通过 GET 方法获取,因此 POST 的返回结果和 GET 的结果在格式上是一致的，并且其内容通常也相同。客户端也可以请求资源“?client=客户端&serial=序号”的方式获取服务器对 POST 的处理结果。比如 *GET ip?client=001&serial=123 HTTP/1.1*，即可获取客户端编号为 001，serial 为 123 的 POST 请求的处理结果。服务器对 POST 结果的 GET 响应数据格式如下。

GET 的数据基本格式：

```
{
    "client": "0001",
    "serial": "1234567890",
    "content":
    {
        // Data to be transferred
    },
    "exception":
    {
        "type": "error / warning",
        "code": "404",
        "detail": "page not find"
    }
}
```

其中 client 是服务器对该客户端的编号，为一个正整数；如果提交 POST 请求的客户端为无效的客

户端，则 GET 响应中 client 字段为空。

serial 与客户端 POST 请求时所提交的 serial 相同。

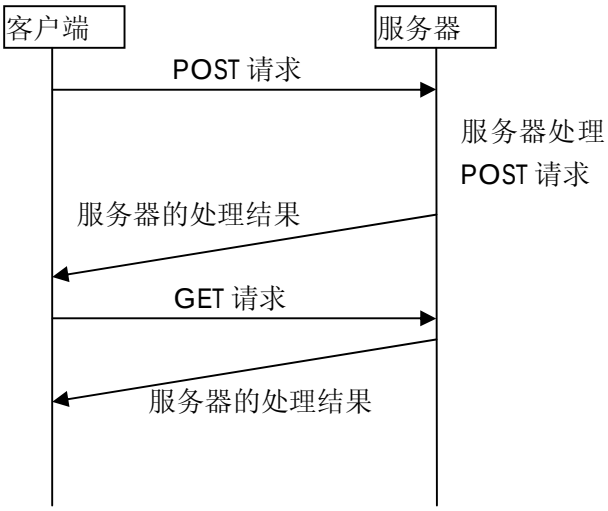
content 为返回的具体内容，content 内部的数据格式与 POST 中的 type 有关。如果没有任何值可返回，content 字段有可能不存在。

exception 为非正常 POST 的异常报告，其中的 type 可取的字段为 error 与 warning，表示异常的类型；code 为异常代码，任何两个不同含义的 code 均不相同，无论 type 的值是否相同；detail 是 code 的详细解释，为一个字符串。

content 字段和 exception 属于可选字段，即有可能某个字段不存在，但至少存在一个字段。

客户端只需遵循 POST 请求在 GET 请求之前即可，GET 处理结果的顺序可以是任意的。客户端可以多次对同一 POST 请求 GET 处理结果，但每次 GET 的结果内容可能不相同，详细信息参见各具体协议的描述。服务器只对 POST 请求的处理结果在较长的一定时间段内保存，之后服务器可能会删除该结果，客户端再通过 GET 请求获取时将会得不到任何结果。

交互的过程如下图所示。



3. 系统相关协议

3.1. 客户端软件注册

客户端只有正确的经过软件注册才可以使用，软件注册既可以在服务器端由管理员完成，也可以由客户端发起注册请求。对于已经在服务器端完成注册的客户端，则客户端的注册请求过程时可选的。

软件注册分为两步：软件注册请求和软件注册确认。客户端先提交注册请求信息，然后获取请求处理结果，再提交注册确认信息，最后获取注册确认处理结果。

3.1.1. 数据格式

软件注册的 POST 提交格式：

```
{
    "client": " deviceID / client number ",
    "serial": "deviceID+serial / serial ",
    "credit": "",
    "type": "register",
    "content":
    {
        "subtype": "request / verify",
        "device": "deviceID",
        "serial": "empty / register serial number",
        "activate": "",
        "expiration": ""
    }
}
```

软件注册的 GET 响应格式：

```
{
    "client": "deviceID / client number",
    "serial": "deviceID+serial ",
    "content":
    {
        "subtype": "unconfirmed",
        "device": "deviceID",
        "serial": " empty / register serial number",
        "activate": "2012-10-10 00:00:00",
        "expiration": "2112-10-09 18:03:26"
    }
}
```

类型字段"type"固定为 register，表示客户端进行软件注册；

内容字段"content"包含五个子字段：subtype、device、serial、activate 和 expiration，subtype 为注册的类型； device 为客户端机器码；serial 为注册序列号，该序列号由软件发布商向软件使用者提供；activate 为软件有效期起始时间，expiration 为软件有效期结束时间。

"content"中的 subtype 字段，在 POST 格式中，注册请求为 request，注册确认为 verify；在 GET 响应格式中，有四种可选字段：illegal、processing 、unconfirmed 和 confirmed，具体含义如下：

- 如果软件未注册或注册信息不合法，则 subtype 为 illegal。
- 如果服务器正在处理注册信息，则 subtype 为 processing。
- 如果客户端的注册请求信息被成功验证，但客户端还为提交注册确认信息，则 subtype 为 unconfirmed。
- 如果客户端的注册确认信息被成功验证，则 subtype 为 confirmed。

3.1.2. 注册请求

软件注册请求时，由于客户端还未获得服务器分配的客户端编号，因此客户端编号"client"可以使用客户端机器码；

客户端注册时首先向服务器提供注册请求信息，信息格式按照“软件注册的 POST 提交格式”；认证码"credit"字段需要合法(如果服务器设定为要求验证 credit 的合法性)，该值的来源可能与通信过程无关，比如客户端软件设置的固定值，或由人工向客户端输入；subtype 字段为 request。

如果服务器软件支持服务器端注册，则序列号、activate 和 expiration 三项可以为空；如果不支持服务器端注册，则序列号、activate 和 expiration 三项必须如实填写，服务器以此信息进行核对。

3.1.3. 注册请求处理结果

服务器收到客户端的软件注册请求后，处理请求信息，返回并保存结果，供客户端以 GET 请求获取。服务器端产生的处理结果按照“软件注册的 GET 响应格式”，subtype 字段为 illegal、processing 或 unconfirmed 三者之一。

如果 subtype 字段为 illegal 或 processing，"content"中的 serial、activate、expiration 三项字段内容为空。

如果 subtype 字段为 unconfirmed，client 为服务器分配给客户端的编号，客户端妥善保存该编号，用于以后的通信。

如果 subtype 字段为 unconfirmed，activate、expiration 两项字段分别为软件有效期起始时间和软件有效期结束时间。如果服务器软件支持服务器端注册，则在首次 GET 注册结果中，序列号"serial"字段包含注册序列号，如果服务器端软件不支持服务器端注册，或不是首次 GET 注册结果，则序列号字段为空。

3.1.4. 注册确认

客户端在提交注册确认信息之前，可以多次发起注册请求(用不同的请求序号"serial")，并获取请求处理结果，但服务器只会保存最后一次注册请求的处理结果。

如果客户端获取完整的注册请求处理结果，则可以向服务器提交注册确认信息，注册确认信息格式按照“软件注册的 POST 提交格式”; credit 字段需要合法(如果服务器设定为要求验证 credit 的合法性)，该值的来源可能与通信过程无关，比如客户端软件设置的固定值，或由人工向客户端输入；subtype 字段必须为 verify。注册确认信息中的序列号("content"中的"serial")可以为空，无需如实提供。

注册确认信息中的 activate 和 expiration 需要和服务器中记录的一致，才能成功。

3.1.5. 注册确认处理结果

服务器在收到客户端提交的合法注册确认信息后，返回并保存注册确认结果，数据格式按照 “软件注册的 GET 响应格式”，subtype 字段为 illegal、processing 或 confirmed 三者之一。如果 subtype 字段为 illegal，客户端需要重新提交注册确认信息。

服务器在生成 subtype 字段为 confirmed 的注册确认结果后，不再接受相同设备机器码及相同注册码的注册请求，除非服务器管理员对服务器手动作修改。

3.2. 认证码获取

在客户端软件注册过程中，根据软件注册序列号按照特定的哈希算法已生成几个认证码，并在软件注册的最后阶段应用。但因为一些特定的原因，可能不适合长期固定使用这几个认证码进行通信，比如在需要特别考虑安全的环境中，或为了服务器方便验证通信的合法性，这时需要客户端从服务器端获取更多的认证码。

鉴于在不同应用环境中对安全的要求不同，认证码获取是一个可选过程。通常情况下，服务器不对 POST 请求中的认证码进行合法性验证。

认证码获取协议只包含两部分：认证码获取请求、认证码获取请求的响应。客户端请求认证码时，通过 POST 方式向服务器提交请求数据。

客户端请求认证码 POST 提交格式：

```
{
    "client": " deviceID / client number ",
    "serial": " serial ",
    "credit": "credit code",
    "type": "credit",
    "content":
    {
        "subtype": "request",
        "max": "1000",
        "encryption": "none"
```

```
        }
    }
```

其中的 credit 用客户端已有的合法认证码，type 为 credit，content 包含三项子字段：subtype、max、encryption。subtype 固定为 request，表示请求获取认证码；max 是一个数字，表示客户端一次可以接纳的认证码数量，以免服务器响应 GET 请求时返回巨量认证码使得客户端没有能力处理；encryption 是加密的类型，指明返回的认证码使用何种加密方式进行加密，可选的字段有：none、DES、3DES、AES、RSA 等，其中 none 表示不加密，如果 encryption 字段不为 none，则客户端必须有能力正确解密服务器返回的认证码数据(比如使用相同的密码解密)。

服务器收到客户端的请求后，返回并保存处理结果，供客户端通过 GET 方式获取。

认证码获取请求的 GET 响应格式：

```
{
    "client": "client number",
    "serial": "serial",
    "content":
    {
        "subtype": "success",
        "sum": "3",
        "list":
        [
            {
                " order": "01",
                "value": "1A9FCBF8"
            },
            {
                " order": "02",
                "value": "2369B092"
            },
            {
                " order": "03",
                "value": "D3FB4E10"
            }
        ]
    }
}
```

content 包含三个子项：subtype、sum、list。subtype 指明服务器的处理结果，包含 processing、success 和 failure 三个可选项。sum 字段为一个数字，指明服务器返回的认证码个数。list 为一个列表，其中的每个成员包含 order 和 value 两个子项，order 是服务器对此次返回的认证码的一个编号，从 1 开始，value 是与 order 对应的认证码。如果 subtype 的值为 processing 或 failure，则 sum 为 0，list

中为空，仅有一对中括号。

3.3. 客户端在线信号

为了方便客户端和服务器进行通信状态监控，客户端可以向服务器报告在线状态。客户端在线信号是一个可选过程，是否提交客户端在线信号由客户端自行决定，服务器无法对其强制要求。

客户端在线 POST 提交格式：

```
{
    "client": " deviceID / client number ",
    "serial": " serial ",
    "credit": "credit code",
    "type": "online",
    "content":
    {
        "time": "2012-10-03 12:01:22",
    }
}
```

该 POST 数据中，type 为 online，content 只包含 time 一项内容，为客户端本机时间。

服务器收到客户端的合法在线数据后，进行记录，返回并保存处理结果，并在客户端 GET 其结果时生成响应结果。

客户端在线 GET 响应格式：

```
{
    "client": "client number ",
    "serial": " serial ",
    "content":
    {
        "time": "2012-10-03 12:01:22.123456",
    }
}
```

type 为 online，content 只包含 time 一个子项。

如果 subtype 为 online，time 为服务器响应 GET 请求的即时时间，该时间在服务器得到客户端 GET 请求时立即生成，精度为微秒，在忽略网络传输延迟的情况下，可作为客户端核对机器时间用；

time 也有可能为零值，为 0000-00-00 00:00:00.000000，此时应该是服务器正在处理更重要的事务，来不及处理该请求。

4. 用户相关协议

为了方便通过客户端管理用户信息，客户端可以下载和上传用户信息。用户信息上传与下载的 POST 格式应该符合以下基本格式：

```
{
    "client": " deviceID / client number ",
    "serial": " serial ",
    "credit": "credit code",
    "type": "user ",
    "content":
    {
        "subtype": "list"
        // other info
    }
}
```

type 的值固定为 user，subtype 的值可以为 group 、list、get、append、modify、delete、login、logout、change_password，分别表示获取用户类型信息、获取用户列表、获取用户详细信息、添加用户详细信息、修改用户详细信息、删除用户、用户登录、用户登出、修改用户密码。content 中除 subtype 字段外，还可能有其他字段，具体参见以下各节详细描述。

4.1. 用户类型信息获取

用户类型信息获取用于获得服务器对用户进行了哪些分类，其 POST 请求中的 content 字段如下：

```
"content":
{
    "subtype": " group "
}
```

其中仅包含 subtype 一个子字段，固定为 group。

服务器获得客户端的合法请求后，返回并保存处理结果，供客户端以 GET 方式获取。

用户类型信息获取 GET 响应格式：

```
{
    "client": "client number ",
    "serial": " serial ",
    "content":
```

```
        {
            "subtype": "success",
            "sum": "3",
            "list":
            [
                {
                    "group": "manager"
                },
                {
                    "group": "worker"
                }
            ]
        }
    }
```

content 中包含三个子项：subtype、sum、list。

subtype 可选的字段为 processing、success 和 failure，分别代表服务器正在处理、获取成功、获取失败。

sum 表示获取的组数量。

list 是一个列表，其中每个成员包含 group 一个子项，代表类型名。如果 sum 为 0，则 list 内容为空，仅有一对中括号。

4.2. 用户列表下载

用户列表下载用于客户端获取指定类型用户的一些基本信息，以方便用户登录和客户端离线管理用户登录，其 POST 请求中的 content 字段如下：

```
"content":
{
    "subtype": " list ",
    "group": " manager "
}
```

其中 subtype 子字段固定为 list，group 子字段为有效的用户类型；group 子字段是可选的，如果没有 group 字段，则表示获取所有类型的用户列表。

服务器获得客户端的合法请求后，返回并保存处理结果，供客户端以 GET 方式获取。

用户列表下载 GET 响应格式：

```
{
    "client": "client number ",
    "serial": " serial ",
    "content":
    {
        "subtype": "success",
        "sum": "3",
        "list":
        [
            {
                "user": "me",
                "group": "manager",
                "password": " 576E3DEFD5CB258D0514F59F89B91571 ",
                "hash": " D36E060779E9EAF26E9F7DAD28AD8554 "
            },
            {
                "user": "you",
                "group": " worker ",
                "password": " 5AA5AD3BBB153D091D00AE67D51B7DC1",
                "hash": " 5A7069D04E6E04F4A62B52B1327681F1"
            },
            {
                "user": "other",
                "group": " worker ",
                "password": " 641E0234E16895DAC578AEBAA4241EDA",
                "hash": " C40D0967B0DA6A4B6F7A47F3DA2D21A6"
            }
        ]
    }
}
```

content 中包含三个子项：subtype、sum、list。

subtype 可选的字段为 processing、success 和 failure，分别代表服务器正在处理、获取成功、获取失败。

sum 表示获取的用户数量。

list 是一个列表，其中每个成员包含 user、group、password、hash 四个子项，分别代表用户名、用户类型、加密后的用户密码、用户照片校验值。

如果该用户没有照片，则 hash 字段为空字符串；如果 sum 为 0，则 list 内容为空，仅有一对中括号。

客户端如果要获取用户的照片信息，可以使用空的用户名(字符串"")作为操作者下载图片，具体参见“图片上传下载”章节。

4.3. 用户详细信息下载

用户详细信息下载用于客户端获取指定用户的详细信息，以方便通过客户端核对用户的详细信息，其 POST 请求中的 content 字段如下：

```
"content":
{
    "subtype": " get ",
    "operator ": "me",
    "user": ["me", " you ", " other "]

}
```

其中 subtype 子字段固定为 get，operator 子字段为已成功登录的用户，如果用户未登录则不会得到成功的返回结果；user 子字段是一个列表，指出要获取哪些用户的详细信息。

特例：如果 user 字段为空(仅有一对中括号)，则服务器会返回所有类型所有用户的详细信息。

服务器获得客户端的合法请求后，返回并保存处理结果，供客户端以 GET 方式获取。

用户详细信息下载 GET 响应格式：

```
{
    "client": "client number ",
    "serial": " serial ",
    "content":
    {
        "subtype": "success",
        "sum": "3",
        "list":
        [
            {
                "user": "me",
                "number": "001",
```

```
        "name": "张三",
        "id": "510104201201011810",
        "company": "宝马公司",
        "street": "春熙路",
        "group": "manager",
        "photo_bytes":"1024",
        "photo_hash": "1A7A023D0C186CCF3E5468E97BD75AB9CAEC96B8",
        "photo":"
                TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGlz
                IHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2Yg
                dGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGlu
                dWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRo
                ZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS"
    },
    {
        "user": "you",
        "number": "002",
        "name": "李四",
        "id": "510104201201011810",
        "company": "宝马公司",
        "street": "春熙路",
        "group": " worker ",
        "photo_bytes":"1024",
        "photo_hash": "1A7A023D0C186CCF3E5468E97BD75AB9CAEC96B8",
        "photo":"
                TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGlz
                IHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2Yg
                dGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGlu
                dWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRo
                ZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS"
    },
    {
        "user": "other",
        "number": "003",
        "name": "王五",
        "id": "510104201201011810",
        "company": "宝马公司",
        "street": "春熙路",
        "group": " worker ",
        "photo_bytes":"1024",
```

"photo_hash": "1A7A023D0C186CCF3E5468E97BD75AB9CAEC96B8",
"photo":"
TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGlz
IHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2Yg
dGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGlu
dWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRo
ZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS"
}
]
}
}

content 中包含三个子项：subtype、sum、list。

subtype 可选的字段为 processing、success 和 failure，分别代表服务器正在处理、获取成功、获取列表失败。

sum 表示获取的用户数量。

list 是一个列表，其中每个成员包含 user、number、name、id、company、street、group、photo_bytes、photo 九个子项，分别代表用户名、用户工号、用户姓名、用户有效证件号码、用户所在公司名称、用户服务的街道名称、用户类型、用户照片大小、用户照片。

photo_bytes 指明照片在 BASE64 编码之前的数据大小，单位为字节；photo_hash 是照片在 BASE64 编码之前的校验值；photo 是照片的 BASE64 编码数据。

如果该用户没有照片，则 photo_bytes 字段为零，photo_hash 和 hash 字段为空字符串。

如果 sum 为 0，则 list 内容为空，仅有一对中括号。

4.4. 用户信息上传

如果在客户端新增或修改了用户信息，需要向服务器上传，其 POST 请求中的 content 字段如下：

```
"content":
{
    "subtype": " append / modify ",
    "operator ": "me",
    "user":
    [
        {
            "user": "me",
            "number": "001",
            "name": "张三",
```

```
        "id": "510104201201011810",
        "company": "宝马公司",
        "street": "春熙路",
        "group": "manager",
        "photo_bytes":"1024",
        "photo_hash": "1A7A023D0C186CCF3E5468E97BD75AB9CAEC96B8",
        "photo":"
                TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGlz
                IHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2Yg
                dGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGlu
                dWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRo
                ZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS"
    },
    {
        "user": "you",
        "number": "002",
        "name": "李四",
        "id": "510104201201011810",
        "company": "宝马公司",
        "street": "春熙路",
        "group": " worker ",
        "photo_bytes":"1024",
        "photo_hash": "1A7A023D0C186CCF3E5468E97BD75AB9CAEC96B8",
        "photo":"
                TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGlz
                IHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2Yg
                dGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGlu
                dWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRo
                ZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS"
    },
    {
        "user": "other",
        "number": "003",
        "name": "王五",
        "id": "510104201201011810",
        "company": "宝马公司",
        "street": "春熙路",
        "group": " worker ",
        "photo_bytes":"1024",
        "photo_hash": "1A7A023D0C186CCF3E5468E97BD75AB9CAEC96B8",
```

"photo":"
TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGlz
IHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2Yg
dGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGlu
dWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRo
ZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS"

}
]
}

其中 subtype 子字段为 append 或 modify，分别代表新增用户和修改现有用户信息；operator 子字段为已成功登录的用户，如果用户未登录则不会得到成功的返回结果；user 子字段是一个列表，指出要添加哪些用户或修改哪些用户的详细信息，user 子字段各项的含义与“用户详细信息下载”返回值中 list 子字段的含义相同。如果 user 字段为空(仅有一对中括号)，则不会添加或修改任何用户的信息。

添加的新用户没有设置密码，新用户可以用任何密码登录成功。

服务器获得客户端的合法请求后，返回并保存处理结果，供客户端以 GET 方式获取。

用户信息上传 GET 响应格式和“用户详细信息下载”GET 响应格式相同，返回的是完整的用户详细信息。

如果只有部分用户信息添加或修改成功，则返回值中只包含操作成功的那部分用户的信息，并且会在 exception 中给出警告。

4.5. 删除用户

当删除用户时，需要向服务器提交删除用户的请求，其 POST 请求中的 content 字段如下：

```
"content":
{
    "subtype": " delete",
    "operator ": "me",
    "user": ["me", " you ", " other "]
}
```

其中 subtype 子字段固定为 delete，operator 子字段为已成功登录的用户，如果用户未登录则不会得到成功的返回结果；user 子字段是一个列表，指出要获取哪些用户的详细信息。如果 user 字段为空(仅有一对中括号)，则不会删除任何用户。

服务器获得客户端的合法请求后，返回并保存处理结果，供客户端以 GET 方式获取。

删除用户 GET 响应格式：

{

```
"client": "client number ",
"serial": " serial ",
"content":
{
    "subtype": "success",
    "sum": "3",
    "list":
    [
        {
            "user": "me"
        },
        {
            "user": "you"
        },
        {
            "user": "other"
        }
    ]
}
}
```

content 中包含三个子项：subtype、sum、list。

subtype 可选的字段为 processing、success 和 failure，分别代表服务器正在处理、获取成功、获取列表失败。

sum 表示删除的用户数量。

list 是一个列表，其中每个成员只包含 user 一个子项，代表已经成功删除的用户的用户名。

如果只有部分用户被删除，则返回值中只包含操作成功的那部分用户的用户名，并且会在 exception 中给出警告。

4.6. 用户登录登出

用户要在客户端软件上操作某些功能，需要向服务器登录；用户从客户端退出时，需要向服务器发送登出信息，服务器以此作为判断用户操作某些功能的权限之一，并记录用户的登录情况。

用户登录登出 POST 请求中的 content 字段如下：

```
"content":
{
    "subtype": " login ",
    "operator ": "me",
```

```
        "password": "CF3E5468E97BD75AB9C"
    }
```

其中 subtype 子字段为 login 或 logout，分别代表用户登录和用户登出；password 为 operator 的登录密码加密后的值，登出时，password 可以为空。

用户登录登出 GET 响应格式：

```
{
    "client": "client number ",
    "serial": " serial ",
    "content":
    {
        "subtype": "success"
    }
}
```

该 GET 响应中，content 只包含 subtype 一个子项，subtype 可选的值为 processing、success 和 failure，分别代表服务器正在处理、登录/登出成功、登录/登出失败。

4.7. 用户密码修改

用户登录成功后，可以修改密码，客户端将密码修改请求发送至服务器，服务器返回操作结果。

新用户添加成功后，新用户也应当及时修改密码，否则有账户被别人盗用的风险。

用户密码修改 POST 请求中的 content 字段如下：

```
"content":
{
    "subtype": "change_password",
    "operator ": "me",
    "password":
    {
        "old": "CF3E5468E97BD75AB9C",
        "new": "2C2BB2CBC4C010EE1E1"
    }
}
```

其中 subtype 子字段固定为 change_password，operator 子字段为已成功登录的用户，如果用户未登录则不会得到成功的返回结果；password 包含 old 和 new 两个子项，分别为 operator 的旧登录密码加密后的值和新登录密码加密后的值。

用户密码修改 GET 响应格式和用户登录登出 GET 响应格式相同。

5. 传感器信息传递协议

无线地磁传感器用于检测车辆进入和离开车位。地磁传感器检测到车辆进入或离开车位时，将检测结果实时的发送至地磁节点控制器。地磁节点控制器将检测结果及时的发送到服务器，由服务器记入数据库中。由于客户端无法预知服务器何时收到地磁检测结果，而检测结果应当及时传到客户端，客户端软件才能及时的提醒操作员进行相应操作。由于地磁检测信息传递的特殊性和即时性要求，完全遵循 HTTP 协议无法达到预期的通信效果，只能为其制定特殊的通信协议。

考虑到地磁节点控制器有可能远离服务器的情况，客户端获知车辆进出信息的方式可以有两种：一种是客户端通过 http 协议主动从服务器去获取，另一种是使用临近客户端的设备发送 IP 广播数据包，客户端只需侦听。

5.1. 客户端主动从服务器获取车辆进出信息

客户端主动从服务器获取车辆进出信息需要客户端向服务器发送 POST 请求。

POST 提交格式：

```
{
    "client": " deviceID / client number ",
    "serial": " serial ",
    "credit": "credit code",
    "type": "sensor",
    "content":
    {
        "subtype": " enter "
    }
}
```

该 POST 请求中 type 为 sensor，content 中只包含 subtype 一个子项，其值为 enter 或 exit，分别代表获取的车辆进入车位的信息和获取车辆离开车位的信息。当 subtype 为 enter 时，获取的信息仅包括车辆已经在车位上，但还没有客户端对其进行拍照也没有对其收费的信息；当 subtype 为 exit 时，获取的信息包括所有车辆已经离开车位(无论是否已对其收费或拍照)，并且还没有新的车辆到达该车位的信息。

服务器获得客户端的合法请求后，返回并保存处理结果，供客户端以 GET 方式获取。

服务器按以下格式返回结果：

```
{
    "client": "client number ",
    "serial": " serial ",
    "content":
    {
        "subtype": "success",
        "sum": "3",
        "list":
        [
            {
                "street": "99",
                "station": "1",
                "serial": "123",
                "enter":"2012-10-01 12:01:22",
                "exit":"2012-10-01 13:01:22"
            },
            {
                "street": "99",
                "station": "2",
                "serial": "456",
                "enter":"2012-10-01 12:01:22",
                "exit":"2012-10-01 13:01:22"
            },
            {
                "street": "99",
                "station": "3",
                "serial": "789",
                "enter":"2012-10-01 12:01:22",
                "exit":"2012-10-01 13:01:22"
            }
        ]
    }
}
```


content 中包含三个子项：subtype、sum、list。

subtype 可选的字段为 processing、success 和 failure，分别代表服务器正在处理、获取成功、获取列表失败。

sum 表示获取的进出信息的数量。

list 是一个列表，其中每个成员包含 street、station、serial、enter、exit 五个子项，分别表示街道

编号、车位编号、车辆在该车位停放的先后顺序、进入时间、离开时间。如果 POST 请求中 subtype 为 enter ，则返回的结果中 exit 项为空字符串。

5.2. 以 IP 广播形式发送车辆进出信息到客户端

(内容待完善)

6. 应用相关协议

6.1. 车辆拍照信息

当有车辆进入或离开停车位时，客户端向服务器提交车辆出入信息，服务器以此信息作为日志记录和计费。车辆拍照信息仅供服务器记录，作为查阅凭证，是否上传拍照信息不影响对车辆收费。

车辆出入 POST 提交格式：

```
{
    "client": " deviceID / client number ",
    "serial": " serial ",
    "credit": "credit code",
    "type": "stay",
    "content":
    {
        "subtype": "enter",
        "operator": "me",
        "street": "001",
        "station": "123",
        "serial": "789",
        "plate": "川 A88888",
        "time": "2012-10-03 12:01:22",
        "photo_bytes":"1024",
        "photo_hash": "1A7A023D0C186CCF3E5468E97BD75AB9CAEC96B8",
        "photo":"
            TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGlz
            IHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2Yg
            dGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGlu
            dWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRo
            ZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS"
    }
}
```

该 POST 请求中 type 为 stay，content 中包含十个子项。subtype 可选的值为 care 和 pay，分别代表对进入车位的车辆拍照、车主打算结帐；operator 是信息提交者，应该是登录客户端的用户名；street 是街道号；station 是车位号；serial 是车辆在该车位上停放的先后顺序；street、station、serial 均应该与接收到的传感器信号中的值一致；plate 是车辆的车牌号码，如果没有车牌号，该字段为空；time 是对进入车位的车辆拍照、对打算离开车位的车辆拍照时间，该时间应该与操作员拍照时间一致；photo_bytes 指明照片在 BASE64 编码之前的数据大小，单位为字节；photo_hash 是照片在 BASE64 编码之前的校验值，可以是 CRC32，MD5，SHA1，SHA2 等其中之一，服务器会根据校验值的长度判

断是何种校验值；photo 是照片的 BASE64 编码数据。

服务器获得客户端的合法请求后，返回并保存处理结果，供客户端以 GET 方式获取。

车辆出入 GET 响应格式：

```
{
    "client": "client number ",
    "serial": " serial ",
    "content":
    {
        "street": "001",
        "station": "123",
        "serial": "0123456",
        "plate": "川 A88888",
        "enter": "2012-10-03 12:01:22",
        "pay": "2012-10-04 12:00:22",
        "exit": "2012-10-04 12:01:22",
        "enter_hash": "815AB0BB5DB486B5CCCBDA86C59600AEF3928256",
        "pay_hash": "931574AE30F903A8CE3D618F8F439BF0B38CCB4C"
    }
}
```

content 包含九个子项。street 是街道号；station 是车位号；serial 是车辆在该车位上停放的先后顺序；street、station、serial 均应该与接收到的传感器信号中的值一致；plate 是车辆的车牌号码，如果没有车牌号，该字段为空；enter 是车辆进入车位的时间；pay 是对打算离开车位的车辆拍照时间；exit 是车辆离开车位的时间，与传感器检测的时间相同；enter_hash 是对进入车位的车辆拍照照片的校验值，pay_hash 是车主结帐时车辆照片的校验值，校验算法与该客户端以 POST 方式提交的车辆出入信息中的照片校验算法相同。

pay、exit、pay_hash 有可能没有实际的值，如果还没有实际值，则设置为空字符串。

将以上几个值作为 GET 响应中的信息，客户端不仅可以核对提交的时间和校验值，而且还可以实现由一个客户端提供车辆进入时间和，另一个客户端进行收费的灵活功能。

6.2. 车辆登记信息下载

如果车主的车辆信息已经提前记录，或服务器能连接到交通管理等部门的服务器，则客户端可以获取车辆的登记信息。

车辆登记信息 POST 提交格式：

```
{
    "client": " deviceID / client number ",
    "serial": " serial ",
```

```
        "credit": "credit code",
        "type": "vehicle",
        "content":
        {
            "subtype": "request",
            "plate": "川 A88888"
        }
    }
```

该 POST 请求中 type 为 vehicle，content 中包含 subtype 和 plate 两个子项。Subtype 固定为 request，plate 为车牌号。

服务器获得客户端的合法请求后，返回并保存处理结果，供客户端以 GET 方式获取。

车辆登记信息 GET 响应格式：

```
{
    "client": "client number ",
    "serial": " serial ",
    "content":
    {
        "plate": "川 A88888",
        "owner": "王五",
        "style": "宝马拖拉机",
        "situation": "套牌，被盗，肇事逃逸",
        "car_hash": "AD64923EB9C19ADE6EE24242751EFED7",
        "owner_hash": "ED1474DCB860DE0BE7875A8B02118010"
    }
}
```

content 包含六个子项。plate 为车牌号；owner 为车主姓名，style 为车型，situation 为车辆违章状况，car_hash 是车辆图片的校验值，owner_hash 是车主照片的校验值。

6.3. 车辆计费信息

当对车辆进行计费并对车主收费时，需要向服务器提交计费信息，以供服务器核对和记录。

车辆计费 POST 提交格式：

```
{
    "client": " deviceID / client number ",
    "serial": " serial ",
    "credit": "credit code",
    "type": "charge",
```

```
    "content":
    {
        "subtype": " prepay ",
        "operator": "me",
        "street": "001",
        "station": "123",
        "plate": "川 A88888",
        "enter": "2012-10-03 12:01:22",
        "pay": "2012-10-04 12:00:22",
        "price": "25.00"
    }
}
```

该 POST 请求中 type 为 charge，content 中包含八个子项。subtype 可选的值为 prepay、pay、add，分别代表预收费用、结算费用、补收费用；operator 是收费员，应该是登录客户端的用户名；street 是街道号，station 是车位号，street 和 station 两个值与传感器信号中的值一致；plate 是车辆的车牌号码，如果没有车牌号，该字段为空；enter 是车辆进入车位的时间；pay 是收费时间，停车时间按此时间作为截止时间；price 是收费金额，以人民币元为单位。

车辆计费信息的街道号、车位号、进入时间必须与传感器检测到的值一致，服务器以此确定唯一的计费条件。

服务器获得客户端的合法请求后，返回并保存处理结果，供客户端以 GET 方式获取。

车辆计费 GET 响应格式：

```
{
    "client": "client number ",
    "serial": " serial ",
    "content":
    {
        "street": "001",
        "station": "123",
        "plate": "川 A88888",
        "enter": "2012-10-03 12:01:22",
        "pay": "2012-10-04 12:00:22",
        "money":
        {
            "total": "250000",
            "now": "250000",
            "should": "250000"
        }
```

```
        }
    }
```

content 包含六个子项。street 是街道号，station 是车位号，street 和 station 两个值与传感器信号中的街道号一致；plate 是车辆的车牌号码，如果没有车牌号，该字段为空；enter 是车辆进入车位的时间；pay 是结帐时间，停车时间按此时间作为截止时间；

money 是收费金额，以人民币元为单位，包括 total、now、should 三个子项，分别代表总共已经收取的金额(包括本次已收的在内)、本次收取的金额、应该收取的总金额。例如：一辆车停了两小时，停车费标准为 2 元/小时，停车前车主已经预付 10 元停车费，结帐时只退还了 5 元，则提交收费信息后，total、now、should 的值分别为 5、-5、4，由于 total 和 should 不相同，因此帐目还未结清。

客户端可以发送 now 为 0 的收费信息来查询已经收取了多少费用。

6.4. 图片上传下载

客户端向服务器上传图片或从服务器下载图片时，需要提交相关数据，服务器才能正确处理请求。

图片传输 POST 提交格式：

```
{
    "client": " deviceID / client number ",
    "serial": " serial ",
    "credit": "credit code",
    "type": "photo",
    "content":
    {
        "subtype": "upload",
        "operator": "me",
        "bytes": "819200",
        "hash": "AD64923EB9C19ADE6EE24242751EFED7",
        "photo": "
            TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGlz
            IHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2Yg
            dGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGlu
            dWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRo
            ZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS"
    }
}
```

该 POST 请求中 type 为 photo，content 中包含五个子项。subtype 可选的值为 upload 和 download，分别代表上传图片和下载图片；operator 为信息提交者，应该是登录客户端的用户名或者为空字符串(上传图片时服务器要验证用户权限，下载图片时服务器不验证)；bytes 是图片在 BASE64

编码前的大小，以字节为单位；hash 为照片的校验值，可以是 CRC32，MD5，SHA1，SHA2 等其中之一，服务器会根据校验值的长度判断是何种校验值，服务器也会根据该校验值将照片归类到客户端提交的其他信息中(比如用户信息、车辆出入信息等)，以形成完整的信息记录；photo 是图片经过 BASE64 编码后的数据，由于有了 bytes 字段，客户端或服务器可以在照片最后任意附加一些数据，以方便 BASE64 编码或其他特殊用途。

如果 subtype 的值为 download，bytes 和 photo 可以为空，服务器会忽略这两个值，hash 为要获取图片的校验值，服务器根据该值唯一确定要获取的图片内容。

服务器获得客户端的合法请求后，返回并保存处理结果，供客户端以 GET 方式获取。

图片传输 GET 响应格式：

```
{
    "client": "client number ",
    "serial": " serial ",
    "content":
    {
        "bytes": "819200",
        "hash": "AD64923EB9C19ADE6EE24242751EFED7",
        "photo": "
        TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGlz
        IHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2Yg
        dGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGlu
        dWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRo
        ZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS"
    }
}
```

content 包含三个子项。bytes 是图片在 BASE64 编码前的大小，以字节为单位；hash 为照片的校验值，校验算法根据客户端提供的 POST 信息去判定；photo 是图片经过 BASE64 编码后的数据。

如果是客户端要上传图片，则 photo 为空，以减少数据传输量，降低网络传输压力。

7. Detailed Revision History

Changes from Version 0.40 to Version 0.50

- Author: 石洪兴
- Date: 2012-12-06
- Approver: 石洪兴
- Change (+/-) Description
  a) (/) 修改 2.2.节“协议中通信方式”中服务器端返回结果的方式。
  b) (/) 修改 2.3.节“协议基本数据格式”POST 格式中的 client 格式，使得允许以机器码为 client。
  c) (/) 修改 2.3.节“协议基本数据格式”POST 格式中的 credit 限制，使得默认情况下客户端无需对 credit 的合法性负责。
  d) (/) 修改 2.3.节“协议基本数据格式”GET 格式的内容，添加 exception 字段，并允许 content 字段为可选字段。
  e) (/) 修改 2.3.节“协议基本数据格式”中数据交互示意图。
  f) (/) 修改 3.1. 节“客户端软件注册”部分内容，并允许软件注册过程为客户端的一个可选过程。
  g) (/) 修改 3.1. 2. 节“注册请求”和 3.1.4.节“注册确认”部分内容，更改了软件注册过程中验证码的来源方式。
  h) (-) 删除了 3.3.节“客户端连接与断连”部分，其余部分章节的编号同时自动调整。
  i) (/) 修改 3.3. 节“客户端在线信号”的返回数据格式，并且允许“客户端在线信号”为一个可选过程。
  j) (/) 重写“用户相关协议”章节，将用户列表、用户详细信息、用户登录、密码修改等归类为 user 类型。
  k) (/) 重写“传感器信息传递协议”章节。
  l) (/) 修改 6.1. 节“车辆进出信息”为“车辆拍照信息”，并修改其 POST 格式及 GET 结果字段的含义。
  m) (/) 修改 6.3. 节“车辆计费信息”中 POST 格式和 GET 结果格式，修改后的功能允许对车辆多次收费，直到收费正确为止。

Changes from Version 0.30 to Version 0.40

- Author: 石洪兴
- Date: 2012-11-16
- Approver: 石洪兴
- Change (+/-) Description
  n) (/) 修改 6.3 节的 POST 格式中 price 字段的单位(“分”变为“元”)。

Changes from Version 0.20 to Version 0.30

- Author: 石洪兴
- Date: 2012-11-13
- Approver: 石洪兴
- Change (+/-) Description
  a) (+) 在 4.1.2 节的 POST 格式中增加“operator”子字段。
  b) (/) 修改 4.1.2 节的 GET 格式中 photo 字段为 photo_bytes 和 photo_hash。
  c) (-) 合并 4.3 节“登录登出”的两个类型为一个“action”类型，并添加 subtype 以区分具体动作。
  d) (+) 在 6.1 节的 POST 格式中增加“operator”子字段。
  e) (+) 在 6.3 节的 POST 格式中增加“operator”子字段。
  f) (/) 修改 6.4 节的 POST 格式中对 subtype 为 download 的描述，之前的描述是错误的。

Changes from Version 0.10 to Version 0.20

- Author: 石洪兴
- Date: 2012-11-10
- Approver: 石洪兴
- Change (+/-) Description
  a) (/) 修改 2.3 节括号中对“可识别的请求”的描述。
  b) (/) 修改 2.3 节对 GET 请求格式的描述。

First Version 0.10

- Author: 石洪兴
- Date: 2012-10-15
- Approver: 石洪兴
- Change (+/-) Description
  a) (+) 创建文档，起始版本号 0.10