**Communication Protocol Version1.0**

Contents

[Introduction 1](#_Toc428804703)

[ Packet structure 2](#_Toc428804704)

[Protocol 2](#_Toc428804705)

[ Heartbeat 2](#_Toc428804706)

[ Cipher 3](#_Toc428804707)

[ Upgrade 5](#_Toc428804708)

[ Restart 5](#_Toc428804709)

[ General Message 6](#_Toc428804710)

## Introduction

As the smart self-service equipment need to be deployed in different positions distributed around a city, this requires us to give a kind of communication way which can send request from smart equipment to the front-end aggregation server, as well as receive transaction result.

The architecture picture shows the comprehensive construction of application based on xml-UI framework.



### Packet structure

We define a common TCP/IP packet structure that comprises of three parts, the first is command code, the second is packet length, and the third is XML payload.

Standard Msg Packet as below,

1. Cmd code is 4 bytes integer type
2. Msg length is 4 bytes integer type
3. Msg content is string type

-----------------------------------------------------------------

| | | |

| cmd code | Msg Length | Msg Content |

| | | |

-----------------------------------------------------------------

System uses three default type of cmd codes, they are

GENERAL\_MSG\_CODE = 0; //common message

REGISTRY\_MSG\_CODE = 1; //terminal registration message

KEY\_DOWNLOAD\_MSG\_CODE = 2; //download secret key message

User can define cmd code via config file as well.

## Protocol

For convenience of communication, we utilize text message as content, and text message takes standard XML and JSON format to represent different type of stuff, such as array structure, string structure and so on.

### Heartbeat

Heartbeat is very important and plain-vanilla message which keeps client know if the front aggregation server is alive. If there is no valid heartbeat response coming from server, the client will prompt erroneous message to user.

Sample heartbeat message sent from client:

**<?xml version="1.0" encoding="UTF-8" standalone="yes"?>**

**<TLS>**

**<counter>3</counter> <!-- global counter number to represent message from this client -->**

**<prscode>sanlogin</prscode> <!-- action of message -->**

**<terminalid>00000001</terminalid> <!-- the ID of equipment -->**

**<token>usjdfjksa75ks832okudsjd94877akjdua</token> <!-- the token of equipment -->**

**<branchno>test\_bank\_division</branchno> <!-- the alias of organization -->**

**<terminalstate>0</terminalstate> <!-- the state of equipment -->**

**<startid>1440743273124</startid> <!-- the startup time of equipment -->**

**<keyboardstate>1</keyboardstate> <!-- the state of embedded keyboard -->**

**<prtstate>1</prtstate> <!-- the state of embedded printer -->**

**<version>Unknown</version> <!-- the software version of equipment -->**

**</TLS>**

Sample heartbeat message come from aggregation server:

**<?xml version="1.0" encoding="UTF-8" standalone="yes"?>**

**<TLS>**

**<retcode>0</retcode> <!-- response code, 0 is successful -->**

**<terminalid>00000001</terminalid> <!-- the ID of equipment -->**

**<okmsg>终端注册成功</okmsg> <!-- the content of successful response message -->**

**<errmsg/> <!-- the conent of erroneous response message -->**

**<restart>0</restart> <!-- the signal of restart equipment, 0 is none of restart -->**

**<changekey>0</changekey> <!-- the signal of update cipher key of keyboard, 0 is none of change -->**

**</TLS>**

### Cipher

There is a keyboard in client equipment, and this keyboard has encryption functionality. But the equipment need to download the latest and mutative encryption key from aggregation server when it’s brought up. We use another xml message to describe this type message.

Sample encryption key message sent from client:

**<?xml version="1.0" encoding="UTF-8" standalone="yes"?>**

**<TLS>**

**<counter>3</counter> <!-- global counter number to represent message from this client -->**

**<prscode>downloadkey</prscode> <!-- action of message -->**

**<terminalid>00000001</terminalid> <!-- the ID of equipment -->**

**<token>usjdfjksa75ks832okudsjd94877akjdua</token> <!-- the token of equipment -->**

**<branchno>test\_bank\_division</branchno> <!-- the alias of organization -->**

**<terminalstate>0</terminalstate> <!-- the state of equipment -->**

**<startid>1440743273124</startid> <!-- the startup time of equipment -->**

**<keyboardstate>1</keyboardstate> <!-- the state of embedded keyboard -->**

**<prtstate>1</prtstate> <!-- the state of embedded printer -->**

**<version>Unknown</version> <!-- the software version of equipment -->**

**</TLS>**

Downloading message is like heartbeat message except prscode is ‘downloadkey’.

Sample downloading message come from aggregation server:

**<?xml version="1.0" encoding="UTF-8" standalone="yes"?>**

**<TLS>**

**<retcode>0</retcode>**

**<terminalid>00000001</terminalid>**

**<okmsg>**

**{**

**"maKey": "01234789ABCDEF",**

**"masterKey": "01234789ABCDEF",**

**"workSecretKey": "01234789ABCDEF"**

**}**

**</okmsg> <!-- the content of successful response message -->**

**</TLS>**

Response message is a complex construction including XML and JSON. Aggregation server puts encryption keys into okmsg field, and encryption keys are constructed by JSON format based on {key:value}.

### Upgrade

This is about equipment software upgrade/downgrade requirement. As these equipment are distributed in different bank outlets, we cannot upgrade or downgrade software manually. So we will send software update message from aggregation server to all equipment, and those equipment will fetch designated software package to update after receiving upgrade/downgrade message.

Sample upgrade/downgrade message come from aggregation server:

**<?xml version="1.0" encoding="UTF-8" standalone="yes"?>**

**<TLS>**

**<retcode>0</retcode>**

**<terminalid>0X12347890ABCDEF</terminalid>**

**<update>1</update> <!—signal of change, 0 is none of update, 1 is update-->**

**<updatefile> <!—path is software package url and md5 is result of package md5 caculating, version is software’s internal id-->**

**{**

**"path": "http://10.117.5.10/bankApp-upgrade.zip",**

**"version": "2.1",**

**"md5": "7f90530983ff438ff57dcf11ab342ba4"**

**}**

**</updatefile>**

**</TLS>**

### Restart

This is about software application and equipment restart operation. As a remotely deployed equipment, there is a central application management platform to control its status. If there is any problem in an equipment, administrator will take an action to tackle it. The administrator can decide to restart application or the whole equipment depending on the root cause of problem.

Sample application restart message come from aggregation server:

**<?xml version="1.0" encoding="UTF-8" standalone="yes"?>**

**<TLS>**

**<retcode>0</retcode> <!-- response code, 0 is successful -->**

**<terminalid>00000001</terminalid> <!-- the ID of equipment -->**

**<restart>0</restart> <!-- 0 is none of restart, 1 means the application will restart -->**

**</TLS>**

Sample system restart message come from aggregation server:

**<?xml version="1.0" encoding="UTF-8" standalone="yes"?>**

**<TLS>**

**<retcode>0</retcode> <!-- response code, 0 is successful -->**

**<terminalid>00000001</terminalid> <!-- the ID of equipment -->**

**<systemrestart>0</systemrestart> <!-- 0 is none of restart, 1 means the system will restart -->**

**</TLS>**

### General Message

General message is plain-vanilla conversation carrier in this system. All conducts in current system belong to this category, such as query, transfer cash and so forth. These message will catch component’s value and tweak them into an xml string. With a cmd code appending, we will send this string to aggregation server.

Sample general message sent from client:

**<?xml version="1.0" encoding="UTF-8" standalone="yes"?>**

**<TLS>**

**<counter>3</counter>**

**<uid>20150831170718866243</uid> <!—message unique serial number-->**

**<prscode>queryAcount</prscode>**

**<terminalid>00000001</terminalid>**

**<branchno>test\_bank\_division</branchno>**

**<N\_330332>2015-08-31</N\_330332> <!-- component tag and value, tag consists of ‘N\_’+ component id -->**

**<N\_330334>2015-08-31</N\_330334>**

**</TLS>**

Sample general message come from aggregation server:

**<?xml version="1.0" encoding="UTF-8" standalone="yes"?>**

**<TLS>**

**<retcode>1</retcode> <!—the signal of conduct -->**

**<terminalid>00000001</terminalid>**

**<okmsg/> <!—the successful result of conduct -->**

**<errmsg>输入的密码错误,操作无法进行</errmsg> <!—the cause of conduct failure -->**

**<prtmsg/>**

**<cargo/>**

**</TLS>**