**Installation Guide**

Contents

[Introduction 1](#_Toc429334479)

[ System structure 1](#_Toc429334480)

[Installation 2](#_Toc429334481)

[ Installation package 2](#_Toc429334482)

[ Chinese type tool(AVF) 3](#_Toc429334483)

[ Equipment configuration 4](#_Toc429334484)

[ Application 4](#_Toc429334485)

[ conf 5](#_Toc429334486)

[ log 5](#_Toc429334487)

[ mock-data 6](#_Toc429334488)

[ test 16](#_Toc429334489)

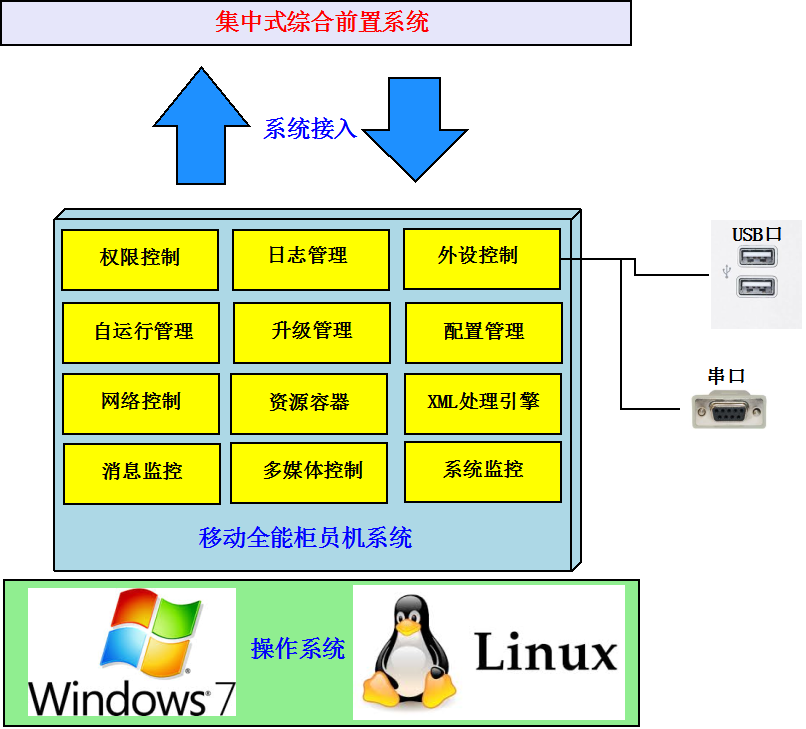
[ Desktop shortcut 19](#_Toc429334490)

## Introduction

The client system is built on java technology, so it intrinsically has a capability of running different platform. However, current smart equipment usually uses window7 as target platform, we suggest installing client system in win7 platform and require smart equipment must support this type of operation system. But if the customer wants to use diverse operation system such as Linux or Mac OS, we suggest that they should offer various drivers in those platform for smart equipment.

### System structure

The architecture picture shows the comprehensive construction of application based on xml-UI framework. Light blue box in picture is smart equipment, the whole client system is running in it.

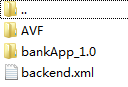


## Installation

In this paragraph, we will detail client installation. For a convenience, we presume the target operation system running in a smart equipment is window7. In the next parts, if we have no special designation, we all use window7 as default target operation system.

### Installation package

Installation package contains a lot of folders, which are for different file types. The basic package structure is described by the picture below.



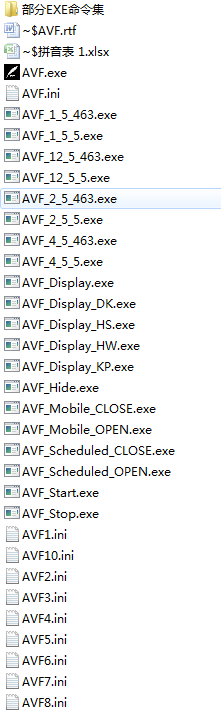
We interpret the installation package structure via following describes,

* Installation folder 🡪 such as ‘D:\bank\’
  + AVF 🡪 this folder includes Chinese type tool
  + bankApp\_1.0 🡪application package
  + backend.xml 🡪equipment configuration

### Chinese type tool(AVF)

This is a Chinese type tool, a lot of bank’s smart equipments have integrated this kind of type tool. User can utilize this to input account name that must be Chinese character.

AVF package is like picture below:



### Equipment configuration

This is configuration file of smart equipment, it includes many local device items used by application.

Sample of equipment configuration:

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

**<Transportation**

**xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'**

**xsi:noNamespaceSchemaLocation='TransportationSchema.xsd'>**

**<server>**

**<host>127.0.0.1</host> <!-- IP address of aggregation server -->**

**<port>8080</port>** **<!-- port of aggregation server -->**

**</server>**

**<registration>**

**<prsCode>sanlogin</prsCode> <!-- registration code to aggregation server -->**

**<terminalID>00000001</terminalID> <!—equipment’s ID -->**

**<token>ABCD</token> <!-- equipment’s token -->**

**<heartbeat>100000</heartbeat> <!-- heartbeat frequency between equipment and aggregation server -->**

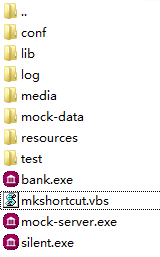
**<branchno>600000</branchno> <!—equipment’s branch ID -->**

**</registration>**

**</Transportation>**

### Application

This is application software package. It includes all software functionalities, just like picture below:

****

We interpret the application package structure via following describes,

* Application folder 🡪 such as ‘D:\bank\app’
  + conf 🡪 this folder includes GUI schema
  + lib 🡪application running library
  + log 🡪application running log files
  + media 🡪wav files used by application
  + mock-data 🡪mock data of mock server
  + resource 🡪application icon files
  + test 🡪application automatic test suite
  + bank.exe 🡪application starter
  + mkshortcut.vbs 🡪shortcut creation
  + mock-server.exe 🡪mock server starter
  + silent.exe 🡪application automatic testing

#### conf

About this part, you can refer to ‘How to config UI.docx‘

#### log

About this part, you can look into product folder. Basic log record likes sample below,

**七月 14, 2015 11:07:50 下午 king.flow.net.TunnelBuilder$HeartBeatTask run**

**信息: Heartbeat message : <?xml version="1.0" encoding="UTF-8" standalone="yes"?><TLS><counter>1</counter><prscode>sanlogin</prscode><terminalid>00000001</terminalid><token>ABCD</token><branchno>600000</branchno><terminalstate>0</terminalstate><startid>1436886470677</startid><keyboardstate>0</keyboardstate><prtstate>1</prtstate><version>1.0</version></TLS>**

**七月 14, 2015 11:07:51 下午 king.flow.net.P2PTunnel$MessageClientHandler channelRead**

**信息: <?xml version="1.0" encoding="UTF-8" standalone="yes"?><TLS><retcode>0</retcode><terminalid>00000001</terminalid><okmsg>终端注册成功</okmsg><errmsg></errmsg><restart>0</restart><changekey>0</changekey></TLS>**

#### mock-data

We can showcase application in a single smart equipment without binding with front aggregation server. Actually this is implemented via a fake server instead of a real server. We design a mock server and set its data via JSON file, which includes returning data of each business operation. When user uses mock server, he will feel this equipment is connecting a real aggregation server.

Sample mock data as below:

**{**

**"terminal": {**

**"00000001": {**

**"pwdVerify": {**

**"password": "111111",**

**"errmsg": "输入的密码错误,操作无法进行"**

**},**

**"accountHalt": {**

**"errmsg": "账户已被挂失或冻结,操作无法进行"**

**},**

**"prscode": {**

**"sanlogin": {**

**"operation": {**

**"verifyPwd": false**

**},**

**"success": {**

**"okmsg": "终端注册成功"**

**}**

**},**

**"downloadkey": {**

**"operation": {**

**"verifyPwd": false**

**},**

**"success": {**

**"okmsg": {**

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

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

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

**}**

**}**

**},**

**"printPB": {**

**"operation": {**

**"verifyPwd": false**

**},**

**"success": {**

**"okmsg": [["20150701",**

**"RMB",**

**"累存",**

**"100000", 10, 1],**

**["20150701",**

**"RMB",**

**"累取",**

**"-50000", 10, 1],**

**["20150701",**

**"RMB",**

**"利息",**

**"3500", 10, 1]]**

**}**

**},**

**"queryTranDetail": {**

**"success": {**

**"okmsg": [["20010105",**

**"800.00",**

**"取款",**

**""],**

**["20020105",**

**"800.00",**

**"转账",**

**"6222000200119674589"],**

**["20030105",**

**"1600.00",**

**"转账",**

**"8999666533215698751"],**

**["20030105",**

**"2700.00",**

**"转账",**

**"2333698545698542688"],**

**["20030105",**

**"3200.00",**

**"转账",**

**"2555699877412589622"],**

**["20040105",**

**"8900.00",**

**"转账",**

**"2333666999987452136"],**

**["20040105",**

**"4500.00",**

**"转账",**

**"6222000144569875428"],**

**["20040105",**

**"600.00",**

**"转账",**

**"15478999633215647895"],**

**["20120105",**

**" 80.00",**

**"转账",**

**"6222000200119674589"],**

**["20120105",**

**"810.00",**

**"转账",**

**"6222000200119674589"],**

**["20140105",**

**"300.00",**

**"转账",**

**"6222000200119674589"],**

**["20150105",**

**"100.00",**

**"转账",**

**"2111555633321123698"],**

**["20150105",**

**"100.00",**

**"转账",**

**"9666333255569874563"],**

**["20150105",**

**"700.00",**

**"转账",**

**"6222333566998745632"],**

**["20150105",**

**"300.00",**

**"转账",**

**"6222000200119674581"]]**

**}**

**},**

**"qyTranDetail": {**

**"operation": {**

**"advancedTableQuery": {**

**"componentId": "3409"**

**}**

**},**

**"success": {**

**"okmsg": {**

**"1": {**

**"total": 2,**

**"current": 1,**

**"value": [["20010105",**

**"800.00",**

**"取款",**

**"",**

**"百度科技有限公司"],**

**["20020101",**

**"800.00",**

**"转账",**

**"6222000200119674589",**

**"新浪科技有限公司"],**

**["20020102",**

**"1800.00",**

**"转账",**

**"6222000200119674589",**

**"新浪科技有限公司"],**

**["20020103",**

**"2800.00",**

**"转账",**

**"6222000200119674589",**

**"新浪科技有限公司"],**

**["20020104",**

**"3800.00",**

**"转账",**

**"6222000200119674589",**

**"新浪科技有限公司"],**

**["20020105",**

**"4800.00",**

**"转账",**

**"6222000200119674589",**

**"新浪科技有限公司"],**

**["20020106",**

**"5800.00",**

**"转账",**

**"6222000200119674589",**

**"新浪科技有限公司"],**

**["20020107",**

**"6800.00",**

**"转账",**

**"6222000200119674589",**

**"新浪科技有限公司"],**

**["20020108",**

**"7800.00",**

**"转账",**

**"6222000200119674589",**

**"新浪科技有限公司"],**

**["20020109",**

**"8800.00",**

**"转账",**

**"6222000200119674589",**

**"新浪科技有限公司"],**

**["20020110",**

**"8800.00",**

**"转账",**

**"6222000200119674589",**

**"新浪科技有限公司"],**

**["20020111",**

**"10800.00",**

**"转账",**

**"6222000200119674589",**

**"新浪科技有限公司"],**

**["20020112",**

**"11800.00",**

**"转账",**

**"6222000200119674589",**

**"新浪科技有限公司"],**

**["20020114",**

**"12800.00",**

**"转账",**

**"6222000200119674589",**

**"新浪科技有限公司"],**

**["20010115",**

**"13900.00",**

**"转账",**

**"6222000200119674581",**

**"华为股份科技有限公司"]]**

**},**

**"2": {**

**"total": 2,**

**"current": 2,**

**"value": [["20010105",**

**"800.00",**

**"取款",**

**"",**

**"360科技有限公司"],**

**["20020101",**

**"800.00",**

**"转账",**

**"6222000200119674589",**

**"网易科技有限公司"],**

**["20020102",**

**"1800.00",**

**"转账",**

**"6222000200119674589",**

**"网易科技有限公司"],**

**["20020103",**

**"2800.00",**

**"转账",**

**"6222000200119674589",**

**"网易科技有限公司"],**

**["20020104",**

**"3800.00",**

**"转账",**

**"6222000200119674589",**

**"网易科技有限公司"],**

**["20020105",**

**"4800.00",**

**"转账",**

**"6222000200119674589",**

**"网易科技有限公司"],**

**["20020106",**

**"5800.00",**

**"转账",**

**"6222000200119674589",**

**"搜狐科技有限公司"],**

**["20020107",**

**"6800.00",**

**"转账",**

**"6222000200119674589",**

**"搜狐科技有限公司"],**

**["20020108",**

**"7800.00",**

**"转账",**

**"6222000200119674589",**

**"搜狐科技有限公司"],**

**["20020109",**

**"8800.00",**

**"转账",**

**"6222000200119674589",**

**"搜狐科技有限公司"],**

**["20020110",**

**"8800.00",**

**"转账",**

**"6222000200119674589",**

**"搜狐科技有限公司"],**

**["20020111",**

**"10800.00",**

**"转账",**

**"6222000200119674589",**

**"搜狐科技有限公司"],**

**["20020112",**

**"11800.00",**

**"转账",**

**"6222000200119674589",**

**"搜狐科技有限公司"],**

**["20020114",**

**"12800.00",**

**"转账",**

**"6222000200119674589",**

**"搜狐科技有限公司"],**

**["20010115",**

**"13900.00",**

**"转账",**

**"6222000200119674581",**

**"中兴股份科技有限公司"]]**

**}**

**}**

**}**

**},**

**"qygetQuery": {**

**"success": {**

**"okmsg": "<html>付款账号: 6222000022119675684<br>付款户名: 韵达<br>可用余额: 98665.00元<br>充值号: 0012566987<br>充值金额: 900.00元</html>"**

**}**

**},**

**"paysalary": {**

**"success": {**

**"okmsg": "工资发放成功"**

**}**

**}**

**}**

**}**

**}**

**}**

#### test

We offer a comprehensive automatic test framework for application, this test framework is independent of real front aggregation server. If user design test suite and test case, test framework will run those in current equipment, so user will validate his business correction.

Test suite and test case is defined by XML file as GUI component, you can learn it from sample below.

Test suite sample:

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

**<testsuits xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'**

**xsi:noNamespaceSchemaLocation='TestDefine.xsd' loop='2'> <!--loop用来表示执行次数, 如果没有这个属性，表示无限次执行-->**

**<testcase-path>./test/testcase\_查询/test\_case.xml</testcase-path>**

**</testsuits>**

Testcase sample**:**

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

**<testsuits xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'**

**xsi:noNamespaceSchemaLocation='TestDefine.xsd'>**

**<testcase>**

**<name>账户查询测试</name> <!--测试用例名称-->**

**<description>测试描述：通过某个账户的查询来获取余额信息</description> <!--测试用例功能描述-->**

**<steps> <!--测试用例执行全部步骤-->**

**<step> <!--测试用例某一步骤-->**

**<component>102</component> <!--测试用例对应的控件ID-->**

**<pre-wait>3</pre-wait> <!--命令执行前设定等待时间，单位为秒. 0表示不需要等待-->**

**<action> <!--控件执行动作-->**

**<name>Click</name> <!--执行动作名称, Click表示点击按钮-->**

**<parameter></parameter> <!--执行动作参数，不填表示不需要参数-->**

**</action>**

**<post-wait>3</post-wait> <!--命令执行后设定等待时间，单位为秒-->**

**</step>**

**<step>**

**<component>207</component>**

**<pre-wait>0</pre-wait>**

**<action>**

**<name>Click</name>**

**<parameter></parameter>**

**</action>**

**<post-wait>3</post-wait>**

**</step>**

**<step>**

**<component>330336</component>**

**<pre-wait>0</pre-wait>**

**<action>**

**<name>Click</name>**

**<parameter></parameter>**

**</action>**

**<post-wait>3</post-wait>**

**</step>**

**<step>**

**<component>203</component>**

**<pre-wait>0</pre-wait>**

**<action>**

**<name>Click</name>**

**<parameter></parameter>**

**</action>**

**<post-wait>3</post-wait>**

**</step>**

**</steps>**

**</testcase>**

**</testsuits>**

### Desktop shortcut

Application will create a shortcut in current desktop of window operation system, if user want to restart application, he can close current application process in operation system manager and double click this shortcut to restart application.

Windows Shortcut creation script as below:

**REM run this script to create bank app shortcut in desktop**

**REM two parameters mkshortcut /target:"F:\free\bankApp-project\bank.exe" /shortcut:"bank"**

**REM set oShellLink = WshShell.CreateShortcut(Wscript.Arguments.Named("shortcut") & ".lnk")**

**REM one parameter mkshortcut /target:"F:\free\bankApp-project"**

**set WshShell = WScript.CreateObject("WScript.Shell" )**

**set oShellLink = WshShell.CreateShortcut("bank.lnk")**

**targetFolder = WScript.Arguments.Named("target")**

**oShellLink.TargetPath = targetFolder & "\bank.exe"**

**oShellLink.WorkingDirectory = targetFolder**

**oShellLink.WindowStyle = 1**

**oShellLink.Save**

**REM shortcut has been saved in folder of subprocess startuped in java runtime, that means you must know what's current java running folder or set it in Runtime.getRuntime().exec(makeShortcut, null, workingFolder)**

**REM WScript.Echo WshShell.SpecialFolders("Desktop")**

**REM retrieve current user's desktop path, and move created shortcut to desktop**

**desktop = WshShell.SpecialFolders("Desktop")**

**REM WScript.Echo "cmd /C move " & targetFolder & "\bank.lnk " & desktop & "\bank.lnk"**

**WshShell.run "cmd /C move " & targetFolder & "\bank.lnk " & desktop & "\bank.lnk"**