**How to Configure GUI**

[GUI Construction 2](#_Toc468784854)

[ Window 3](#_Toc468784855)

[ Page 7](#_Toc468784856)

[ Component 8](#_Toc468784857)

[ Button 9](#_Toc468784858)

[ ComboBox 9](#_Toc468784859)

[ Label 10](#_Toc468784860)

[ TextField 11](#_Toc468784861)

[ PasswordField 11](#_Toc468784862)

[ Date 12](#_Toc468784863)

[ Table 13](#_Toc468784864)

[ AdvancedTable 15](#_Toc468784865)

[ VideoPlayer 17](#_Toc468784866)

[ Grid 18](#_Toc468784867)

[ Action configuration 19](#_Toc468784868)

[ customizedAction 20](#_Toc468784869)

[ jumpPanelAction 20](#_Toc468784870)

[ setFontAction 21](#_Toc468784871)

[ cleanAction 21](#_Toc468784872)

[ limitInputAction 22](#_Toc468784873)

[ useTipAction 22](#_Toc468784874)

[ showTableAction 23](#_Toc468784875)

[ showComboBoxAction 24](#_Toc468784876)

[ swipeCardAction 24](#_Toc468784877)

[ swipe2In1CardAction 25](#_Toc468784878)

[ playMediaAction 26](#_Toc468784879)

[ rwFingerPrintAction 27](#_Toc468784880)

[ openBrowserAction 27](#_Toc468784881)

[ runCommandAction 27](#_Toc468784882)

[ virtualKeyboardAction 28](#_Toc468784883)

[ setPrinterAction 28](#_Toc468784884)

[ sendMsgAction 29](#_Toc468784885)

[ insertICardAction 31](#_Toc468784886)

[ writeICardAction 33](#_Toc468784887)

[ moveCursorAction 34](#_Toc468784888)

[ printPassbookAction 35](#_Toc468784889)

[ playVideoAction 35](#_Toc468784890)

[ showClockAction 35](#_Toc468784891)

[ uploadFileAction 36](#_Toc468784892)

[ balanceTransAction 37](#_Toc468784893)

[ encryptKeyboardAction 38](#_Toc468784894)

[ ejectCardAction 38](#_Toc468784895)

[ withdrawCardAction 39](#_Toc468784896)

[ hideAction 39](#_Toc468784897)

[ showGridAction 39](#_Toc468784898)

[ System supported driver types 40](#_Toc468784899)

[ System variable 40](#_Toc468784900)

## GUI Construction

A general GUI includes three parts, top level frame, middle level page and inner component. That means a whole GUI needs a window, a load of panels and enormous non-container components.

If you wanna configure an application running on JVM, you need to declare a window firstly. And then you need to describe pages switching in this window. On a time point, window will only show a page. That means you need to define page relationship depending on business requirements. In each page, you can lay out different components, such as button, table etc. how to design a page, you need to consider page business purpose. Following paragraph will narrate detailed configuration of each plain-vanilla part.

### Window

A window is the outer frame or the top level container, it can be a Frame or Dialog. Currently, a window only contain panel type, meanwhile only one panel will be show on one time point.

A window can be represented as a xml file, as a default setting, system will access ‘./conf/xml\_window.xml’ in running folder as startup entry.

A typical and ordinary entry file ‘xml\_window.xml’ is as below

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

<window

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

xsi:noNamespaceSchemaLocation='WindowXmlSchema.xsd'> <!--window schema and namespace declaration -->

<id>0</id> <!--window container id, an unique number. As top frame we suggest giving its zero value -->

<type>Frame</type> <!--window type, optional value is Frame or Dialog -->

<ui-style> <!--window UI style setting -->

<lookandfeel> <!--GUI look&feel name, this is optional. if no setting, default value will be used -->

com.jgoodies.looks.windows.WindowsLookAndFeel

</lookandfeel>

<font> <!--System default font setting, optional setting. -->

<name>Fangsong</name>

<size>21</size>

<style>BOLD</style>

</font>

<background>/image/5.jpg</background> <!--System background setting, optional setting -->

<progress>/image/loader\_orange\_256.gif</progress> <!--System progress animation setting, optional setting -->

<color>115,164,209</color> <!--System basic color setting, you can change base color via RGB value. optional setting -->

<textcolor>64,0,64</textcolor> <!--System text color setting, optional setting -->

</ui-style>

<driver> <!--System peripheral drivers setting, optional setting -->

<device>

<type>magnetCard</type> <!--peripheral device type -->

<port>com3</port> <!--serial port name-->

<dll>emptydll</dll> <!--driver dll name -->

</device>

<device>

<type>ICCard</type>

<port>com3</port>

<dll>emptydll</dll>

</device>

<device>

<type>sensorCard</type>

<port>com3</port>

<dll>emptydll</dll>

</device>

<device>

<type>printer</type>

<port>com3</port>

<dll>emptydll</dll>

</device>

<device>

<type>keyboard</type>

<port>com3</port>

<dll>emptydll</dll>

</device>

</driver>

<attribute> <!--Window attribute setting -->

<text>Application of Bank</text> <!-- title text -->

<icon>/image/bank-icon-256.png</icon> <!--window icon -->

</attribute>

<menubar> <!--Menu bar setting, optional setting. If you need no menu, remove it -->

<menu>

<id>-10</id>

<name>Operation of Conduction</name>

<item>

<id>-11</id>

<name>User Management</name>

<action>

<jumpPanelAction>

<nextPanel>200</nextPanel>

</jumpPanelAction>

</action>

</item>

<item>

<id>-12</id>

<name>Device Management</name>

<action>

<jumpPanelAction>

<nextPanel>300</nextPanel>

</jumpPanelAction>

</action>

</item>

</menu>

<menu>

<id>-20</id>

<name>About</name>

<enable>true</enable>

<item>

<id>-21</id>

<name>Version</name>

<action>

<customizedAction>

<className>king.flow.action.customization.AboutAction</className>

</customizedAction>

</action>

</item>

</menu>

</menubar>

<contents> <!--Page setting, this mandatory setting, you must assign page files here-->

<page>./conf/100/panel-100.xml</page> <!--page file link, if you wanna load panel in a window, you should add a link of panel file here -->

<page>./conf/200/panel-200.xml</page>

<page>./conf/300/panel-300.xml</page>

<page>./conf/300/panel-330.xml</page>

<page>./conf/300/panel-360.xml</page>

<page>./conf/900/panel-900.xml</page>

<page>./conf/900/panel-970.xml</page>

</contents>

</window>

That’s a standard window as picture below, it’s constructed wholly based on configuration above



### Page

Page is really container of component, which is loaded by window and contains all working components such as button, table , combobox and so on.

The system just support one type page, which is Panel, a middle-level container. A panel cannot show solely, only when a window has loaded it, it can show.

A typical and ordinary page configuration is a single xml file as below,

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

<panel

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

xsi:noNamespaceSchemaLocation='../WindowXmlSchema.xsd'> <!-- Panel declaration -->

<id>100</id> <!--Panel id, an unique number -->

<type>Panel</type>

<active>true</active> <!--Panel initial show on/off setting, true means this panel will show firstly in a window, false means hide after loaded. Only one page should be true among lots of pages-->

<background>/image/5.jpg</background> <!--Panel background setting, optional setting -->

<component> <!-- Component setting in a panel container-->

<id>101</id> <!-- Component unique id -->

<type>Label</type> <!-- Component type, current supported type are Button, Table, AdvancedTable, ComboBox, Label, TextField, PasswordField, TextArea, Date -->

<attribute> <!-- Component attribute setting-->

<rect> <!--Component rectangle attribute -->

<x>600</x>

<y>0</y>

<width>600</width>

<heigh>150</heigh>

</rect>

<text>Welcome to use bank application</text> <!--Component text property, optional setting -->

<icon>/image/5.ico</icon> <!-- Component icon property, optional setting -->

<debug >false</debug> <!--debug attribute, true will show component border, false no border show. Optional setting -->

</attribute>

<action> <!-- Component action setting, optional setting -->

<fontSetAction>

<fontSize>45</fontSize>

</fontSetAction>

</action>

</component>

<component>

<id>104</id>

<type>ComboBox</type>

<attribute>

<rect>

<x>700</x>

<y>400</y>

<width>300</width>

<heigh>40</heigh>

</rect>

</attribute>

<action>

<comboShowAction>

<items>CNY/Yuan, USD/Dollar, JPY/Yen</items>

</comboShowAction>

<mediaPlayAction>

<media>./media/goodbye.wav</media>

</mediaPlayAction>

</action>

</component>

</panel>

### Component

Component is the plain-vanilla unit of GUI system. A page is composed of some components, and current supported types include Button, Table, AdvancedTable, ComboBox, Label, TextField, PasswordField, and Date.

#### Button

***Sample configuration:***

<component>

<id>102</id>

<type>Button</type>

<attribute>

<text>查询业务</text>

<rect>

<x>200</x>

<y>300</y>

<width>300</width>

<heigh>50</heigh>

</rect>

<debug> <!-- show the component bound -->

<color>255,255,0</color> <!-- set the component bound using this color -->

</debug>

</attribute>

</component>



#### ComboBox

***Sample configuration:***

<component>

<id>104</id>

<type>ComboBox</type>

<attribute>

<rect>

<x>700</x>

<y>400</y>

<width>300</width>

<heigh>40</heigh>

</rect>

</attribute>

<action>

<showComboBoxAction>

<items>CNY/人民币, USD/美元, JPY/日元</items>

</showComboBoxAction>

</action>

</component>



#### Label

***Sample configuration:***

<component>

<id>101</id>

<type>Label</type>

<attribute>

<rect>

<x>600</x>

<y>0</y>

<width>600</width>

<heigh>150</heigh>

</rect>

<text>欢迎使用自助银行查询业务</text>

</attribute>

<action>

<fontSetAction>

<fontSize>45</fontSize>

</fontSetAction>

</action>

</component>



#### TextField

***Sample configuration:***

<component>

<id>107</id>

<type>TextField</type>

<attribute>

<rect>

<x>400</x>

<y>500</y>

<width>300</width>

<heigh>40</heigh>

</rect>

</attribute>

</component>



#### PasswordField

***Sample configuration:***

<component>

<id>105</id>

<type>PasswordField</type>

<attribute>

<rect>

<x>400</x>

<y>400</y>

<width>300</width>

<heigh>40</heigh>

</rect>

</attribute>

</component>



#### Date

***Sample configuration:***

<component>

<id>302</id>

<type>Date</type>

<attribute>

<rect>

<x>180</x>

<y>30</y>

<width>300</width>

<heigh>50</heigh>

</rect>

</attribute>

</component>



#### Table

***Sample configuration:***

<component>

<id>311</id>

<type>Table</type>

<attribute>

<rect>

<x>0</x>

<y>100</y>

<width>300</width>

<heigh>450</heigh>

</rect>

<debug>false</debug>

</attribute>

<action>

<showTableAction>

<columnNames>流水号, 名称, 开始时间, 结束时间, 费用</columnNames>

</showTableAction>

</action>

</component>



If you need a scroll bar aside a table, you need to put a table component in a decorator as below, and otherwise there is no scroll bar showing with this table.

***Sample configuration:***

**<decorator>**

**<id>310</id>**

**<type>ScrollPanel</type>**

**<attribute>**

**<rect>**

**<x>0</x>**

**<y>100</y>**

**<width>1600</width>**

**<heigh>600</heigh>**

**</rect>**

**<debug>false</debug>**

**</attribute>**

<component>

<id>311</id>

<type>Table</type>

<attribute>

<rect>

<x>0</x>

<y>100</y>

<width>300</width>

<heigh>450</heigh>

</rect>

<debug>false</debug>

</attribute>

<action>

<fontSetAction>

<fontSize>15</fontSize>

</fontSetAction>

</action>

<action>

<showTableAction>

<columnNames>流水号, 名称, 开始时间, 结束时间, 费用</columnNames>

</showTableAction>

</action>

</component>

**</decorator>**

****

#### AdvancedTable

***Sample configuration:***

<component>

<id>330337</id>

<type>AdvancedTable</type>

<attribute>

<rect>

<x>0</x>

<y>95</y>

<width>1275</width>

<heigh>600</heigh>

</rect>

</attribute>

<action>

<showTableAction>

<columnNames>流水号, 名称, 开始时间, 结束时间, 费用</columnNames>

</showTableAction>

</action>

<action>

<sendMsgAction>

<cmdCode>330</cmdCode>

<prsCode>queryAcount</prsCode>

<conditions>330332,330334</conditions>

<nextStep>

<nextPanel>330</nextPanel>

<display>330337</display>

</nextStep>

<exception>

<nextPanel>970</nextPanel>

<display>971</display>

</exception>

<checkRules>

<notEqual>

<more>330334</more>

<less>330332</less>

<errMsg>查询开始时间不能够大于结束时间</errMsg>

</notEqual>

</checkRules>

</sendMsgAction>

</action>

</component>



#### VideoPlayer

***Sample configuration:***

<component>

<id>1005</id>

<type>VideoPlayer</type>

<attribute>

<rect>

<x>90</x>

<y>200</y>

<width>645</width>

<heigh>360</heigh>

</rect>

</attribute>

<action>

<playVideoAction>

<media>./media/video/Smart-Community.mp4</media>

<replayInterval>30</replayInterval>

</playVideoAction>

</action>

</component>



#### Grid

***Sample configuration:***

<component>

<id>33115</id>

<type>Grid</type>

<attribute>

<rect>

<x>100</x>

<y>400</y>

<width>680</width>

<heigh>300</heigh>

</rect>

</attribute>

<action>

<showGridAction>

<row>1</row>

<column>2</column>

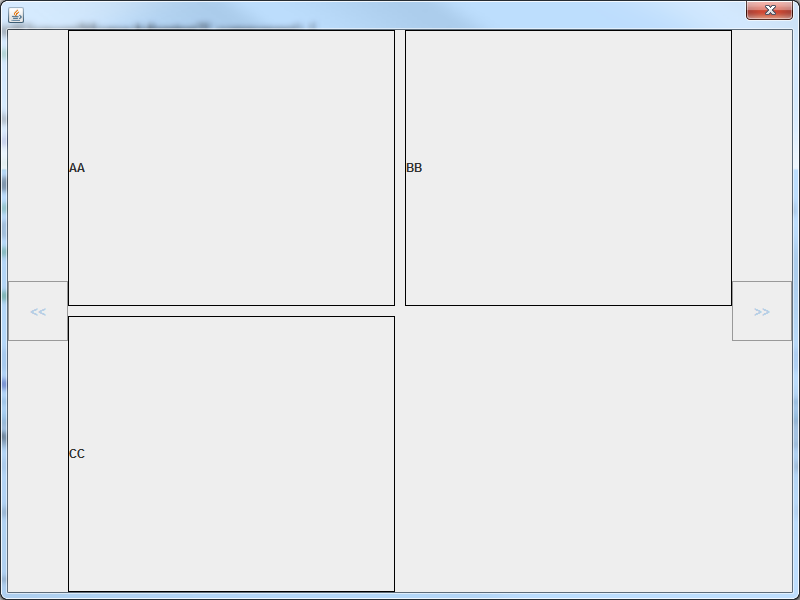
<hgap>15</hgap>

<vgap>15</vgap>

</showGridAction>

</action>

</component>



#### NumericPad

***Sample configuration:***

<component>

<id>97</id>

<type>NumericPad</type>

<attribute>

<rect>

<x>300</x>

<y>700</y>

<width>200</width>

<heigh>200</heigh>

</rect>

</attribute>

<action>

<numericPadAction>

<targetId>8</targetId>

</numericPadAction>

</action>

</component>



### Action configuration

After GUI is constructed, you need to configure component action to deal with business requirements. There are several action type here, each of them can do specific functionality.

The table below describes action configuration relationship with component, ‘√’means this action can be configured with this component.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Button | Combo Box | Label | Text Field | Password Field | Table | Advanced Table | Video Player | Grid | Numeric  Pad |
| jumpPanelAction | √ |  |  |  |  |  |  |  |  |  |
| setFontAction | √ | √ | √ | √ | √ | √ | √ |  |  |  |
| cleanAction | √ |  |  |  |  |  |  |  |  |  |
| limitInputAction |  |  |  | √ | √ |  |  |  |  |  |
| useTipAction | √ | √ | √ | √ | √ | √ |  |  |  |  |
| showTableAction |  |  |  |  |  | √ | √ |  |  |  |
| showComboBoxAction |  | √ |  |  |  |  |  |  |  |  |
| swipeCardAction |  | √ |  |  |  |  |  |  |  |  |
| Swipe2In1CardAction |  | √ |  |  |  |  |  |  |  |  |
| playMediaAction | √ | √ |  | √ | √ |  |  |  |  |  |
| rwFingerPrintAction |  |  |  | √ | √ |  |  |  |  |  |
| openBroswerAction | √ |  |  |  |  |  |  |  |  |  |
| runCommandAction | √ |  |  |  |  |  |  |  |  |  |
| virtualKeyboardAction | √ |  |  | √ |  |  |  |  |  |  |
| setPrinterAction | √ |  |  |  |  |  |  |  |  |  |
| sendMsgAction | √ |  |  |  |  |  | √ |  |  |  |
| insertICardAction | √ |  |  |  |  |  |  |  |  |  |
| writeICardAction | √ |  |  |  |  |  |  |  |  |  |
| moveCursorAction | √ | √ |  | √ | √ |  |  |  |  |  |
| printPassbookAction | √ |  |  |  |  |  |  |  |  |  |
| playVideoAction |  |  |  |  |  |  |  | √ |  |  |
| showClockAction |  |  | √ |  |  |  |  |  |  |  |
| uploadFileAction | √ |  |  |  |  |  |  |  |  |  |
| balanceTransAction | √ |  |  |  |  |  |  |  |  |  |
| encryptKeyboardAction |  |  |  |  | √ |  |  |  |  |  |
| eject2In1CardAction | √ |  |  |  |  |  |  |  |  |  |
| hideAction | √ |  |  |  |  |  |  |  |  |  |
| showGridAction |  |  |  |  |  |  |  |  | √ |  |
| numericPadAction |  |  |  |  |  |  |  |  |  | √ |
|  |  |  |  |  |  |  |  |  |  |  |

#### customizedAction

This is an entry for outer developers, they can develop customized action plugin and put into current GUI system. Moreover, third party user can implement special functionality to meet ad-hoc requirement.

#### jumpPanelAction

This action is about page jumping, currently only button/menu can configure this action. That says if you have set this action with component, when you click button/menu, the next page you wanna go will come out.

***Sample configuration:***

<action>

<jumpPanelAction>

<nextPanel>200</nextPanel>

<nextCursor>336</nextCursor> <! --optional setting, designate component getting cursor after jumping page -->

<trigger>201704</trigger> <! --optional setting, designate trigger component, button or combobox -->

</jumpPanelAction>

</action>

#### setFontAction

This action is used to set component text font

***Sample configuration:***

<action>

<setFontAction>

<fontName>Fangsong</fontName> <!--optional setting -->

<fontSize>18</fontSize>

<fontStyle>BOLD</fontStyle> <!--optional setting, valid value is BOLD/PLAIN/ITALIC -->

<fontColor>255,255,0</fontColor> <!--optional setting, set font color -->

<fontBgColor>255,0,255</fontBgColor> <!--optional setting, set font background color -->

</setFontAction>

</action>

#### cleanAction

When you leave some page, you hope to clean all data in that page, you can use this action.

***Sample configuration:***

<action>

<cleanAction>

<conditions>311,307, 337</conditions><!-- clean value of component 311,307 and 337 -->

</cleanAction>

</action>

#### limitInputAction

This action will limit input format in text field and password field component

***Sample configuration:***

<action>

<limitInputAction>

<length>6</length> <!-- limit 6 characters input in one text component -->

</limitInputAction>

</action>

<action>

<limitInputAction editable="false"> <!-- set component editable attribute -->

<length>100</length>

<enableCashLimit>true</enableCashLimit> <!-- set cash format as 0.00 -->

</limitInputAction>

</action>

<action>

<limitInputAction editable="true"> <!-- set component editable attribute -->

<length>100</length>

<enableNumberLimit>true</enableNumberLimit> <!-- set only input number(0-9) -->

</limitInputAction>

</action>

#### useTipAction

Make component show tip when cursor moves to this component

***Sample configuration:***

<action>

<useTipAction>

<tip>&lt;html&gt;&lt;center&gt;&lt;h1&gt;Hello,&lt;/h1&gt;&lt;h2 style=&#39;color: red; font-style: italic;&#39;&gt;World&lt;/h2&gt;&lt;b&gt;Modified&lt;br/&gt;ToolTipText&lt;/b&gt;&lt;/center&gt;&lt;br/&gt;so simple.请输入账号密码</tip> <!-- tip content -->

</useTipAction>

</action>



#### showTableAction

This action allows to show a table in a page

***Sample configuration:***

<action>

<showTableAction>

<columnNames>流水号, 名称, 开始时间, 结束时间, 费用</columnNames> <!--table column names -->

<sum>

<columnName>剩余号</columnName> --optional, sum denoted column

<display>34202</display> --optional, display sum result

</sum>

</showTableAction >

</action>



#### showComboBoxAction

This action allows to show a combobox in a page

***Sample configuration:***

<action>

<showComboBoxAction>

<items>/, ACTION1/刷接触式卡, ACTION2/刷磁条卡, ACTION3/刷IC卡</items> <!—current system only support three types, ACTION1（SENSOR）,ACTION2（MAGNET）,ACTION3（IC）,ACTION4（2In1） -->

</showComboBoxAction>

</action>



#### swipeCardAction

This action allows to combobox selection action link to general card-reading driver

System reserved action constants as below:

**ACTION1** : Sensor card-reading

**ACTION2** : Magnet card-reading

**ACTION3** : IC card-reading

**ACTION4** : 2in1 card-reading

***Sample configuration:***

<action>

<showComboBoxAction>

<items>/, ACTION2/刷磁条卡, ACTION3/插入IC卡</items>

</showComboBoxAction>

<swipeCardAction>

<nextCursor>333</nextCursor> <!-- next component cursor focuses on after swipe card operation -->

<editable>true</editable> <!—whether combobox is editable and allow to keyboard input -->

</swipeCardAction>

</action>

***Debug action configuration:***

<swipe2In1CardAction>

<nextCursor>370004</nextCursor>

<editable>true</editable>

<mediaTip>tip.wav</mediaTip>

<animationTip>/image/loader\_red\_256.gif</animationTip>

<!—if you set debug, you will get cardId value after you swipe card -->

<debug>

<cardId>622248596587123</cardId>

</debug>

</swipe2In1CardAction>

Debug has an attribute ‘limit’, default value is true. If you set false, you can manually input cardId via keyboard instead of swiping card. Like this,

<debug limit=”false”>

<cardId>622248596587123</cardId>

</debug>

#### swipe2In1CardAction

This action allows to combobox selection action link to 2in1 card-reading driver

***Sample configuration:***

<action>

<showComboBoxAction>

<items>/, ACTION4/刷卡</items>

</showComboBoxAction>

<swipe2In1CardAction>

<nextCursor>370004</nextCursor> <!-- next component cursor focuses on after swipe card operation -->

<editable>true</editable> <!--whether combobox is editable and allow to keyboard input, optional setting -->

<mediaTip>./media/tip.wav</mediaTip> <!--audio file prompt user to draw back card, optional setting-->

<animationTip>/image/card.gif</animationTip> <!--animation file prompts user to draw back card, optional setting-->

</swipe2In1CardAction>

</action>

***Debug action configuration:***

<swipe2In1CardAction>

<nextCursor>370004</nextCursor>

<editable>true</editable>

<mediaTip>tip.wav</mediaTip>

<animationTip>/image/loader\_red\_256.gif</animationTip>

<!—if you set debug, you will get cardId value after you swipe card -->

<debug>

<cardId>622248596587123</cardId>

</debug>

</swipe2In1CardAction>

Debug has an attribute ‘limit’, default value is true. If you set false, you can manually input cardId via keyboard instead of swiping card. Like this,

<debug limit=”false”>

<cardId>622248596587123</cardId>

</debug>

#### playMediaAction

Play a media when you move a cursor to this component

***Sample configuration:***

<action>

<playMediaAction>

<media>./media/goodbye.wav</media>

</playMediaAction>

</action>

#### rwFingerPrintAction

Read or write finger print when you put finger on peripheral device

***Sample configuration:***

<action>

<rwFingerPrintAction>

<nextCursor>104</nextCursor> <!-- next component cursor focuses on after swipe card operation -->

<writePrint>true</writePrint> <!—whether this is finger print registry operation -->

</rwFingerPrintAction>

</action>

#### openBrowserAction

Open a browser instance to access a web site

***Sample configuration:***

<action>

<openBrowserAction>

<url>http://www.baidu.com</url>

</openBrowserAction>

</action>

#### runCommandAction

Run local command offered by local operation system.

***Sample configuration:***

<action>

<runCommandAction>

<command>notepad</command> <!-- open notepad -->

</runCommandAction>

</action>



#### virtualKeyboardAction

Open Chinese character type tool.

***Sample configuration:***

<action>

<virtualKeyboardAction>

<start>AVF\_Display.exe</start>

<stop>AVF\_Hide.exe</stop>

</virtualKeyboardAction>

</action>

#### setPrinterAction

Setting receipt format, including receipt header and tail content.

***Sample configuration:***

<action>

<setPrinterAction>

<header>通商银行</header>

<tail>此凭条仅供参照自助银行,如有异议请联系客服中心:96669</tail>

</setPrinterAction>

</action>

***Debug action configuration:***

<setPrinterAction>

<header>淮南通商银行</header>

<tail>此凭条仅供参照自助银行,如有异议请联系客服中心:96669</tail>

<!—if you set debug, you will get printing you want -->

<debug>

<prompt>打印完成</prompt>

</debug>

</setPrinterAction>

#### sendMsgAction

This action is about communication task. When you configure this action with a component that usually is button, action will collect component values user designates and send to convergent server.

***Sample configuration:***

<action>

<sendMsgAction>

<prsCode>queryAcount</prsCode> <!-- operation PRS code -->

<conditions>302,304,307,333,334,335</conditions> <!—those are data need to send up to server, and we need to catch those values from these components -->

<nextStep>

<nextPanel>300</nextPanel> <!-- operation is successful, go to next page -->

<display>311</display> <!-- on successful page, show result in this component -->

<nextCursor>333</nextCursor> <!-- optional parameter, designate next cursor position -->

</nextStep>

<exception>

<nextPanel>970</nextPanel> <!-- operation is failed, go to error page -->

<display>971</display> <!-- in error page, show result to user -->

</exception>

<checkRules> <!-- rule checking before sending data to server -->

<notNull>

<content>333</content> <!-- component input must not be empty -->

<errMsg>密码不能为空</errMsg>

</notNull>

<notNull>

<content>334</content>

<errMsg>确认密码不能为空</errMsg>

</notNull>

<notNull>

<content>335</content>

<errMsg>账户不能为空</errMsg>

</notNull>

<validateCJK>

<content>335</content> <!-- component input must be Chinese character -->

<errMsg>账户名称必须是中文</errMsg>

</validateCJK>

<validateCJK>

<content>336</content>

<errMsg>输入必须是中文</errMsg>

</validateCJK>

<template>

<content>307</content> <!-- component input must start with pattern value -->

<pattern>6222</pattern>

<errMsg>当前使用非本行发行卡片，请使用本行卡</errMsg>

</template>

<equal>

<conditions>333,334</conditions> <!-- components input must be equal -->

<errMsg>两次输入的密码不一致,请重新输入</errMsg>

</equal>

<notEqual> <!-- component input must not be equal -->

<more>304</more>

<less>302</less>

<errMsg>查询开始时间不能够大于结束时间</errMsg>

</notEqual>

</checkRules>

</sendMsgAction>

</action>

***Advanced action configuration:***

If you need to show result in multiple components, for example, you want to use some result as query criteria in next step. You just set multiple component ids in display tag, as long as the server returns a json array as result, system will show them on corresponding component in order.

<nextStep>

<nextPanel>300</nextPanel> <!-- operation is successful, go to next page -->

<display>311, 312</display> <!-- show result array in those components -->

</nextStep>

#### insertICardAction

When you hit a button and hope to enter IC operation page, I need to make you know inserting card firstly. So I will play tip animation and wait user to insert IC card. And then, I can read card information as well as open operation page.

***Sample configuration:***

<action>

<insertICardAction>

<cardType>gzCard</cardType>

<nextStep> <!-- insert a valid IC, jumping to next page -->

<nextPanel>370</nextPanel>

<display>370005, 370006</display>

<trigger>3700051</ trigger> <!—optional, trigger action -->

</nextStep>

<exception> <!-- no valid IC found, jumping to a page-->

<nextPanel>200</nextPanel>

<display>201</display>

<trigger>3700051</ trigger>

</exception>

<animationTip>/image/card.gif</animationTip> <!--animation file prompt user to insert card, optional setting-->

</insertICardAction>

</action>

***Debug action configuration:***

<action>

<insertICardAction>

<cardType>gzCard</cardType>

<nextStep>

<nextPanel>370</nextPanel>

<display>370005, 370006</display>

***<debug>***

***Alice***

***Third-Block-Sunnyveil***

***</debug>***

</nextStep>

<exception>

<nextPanel>200</nextPanel>

<display>201</display>

</exception>

<animationTip>/image/card.gif</animationTip>

</insertICardAction>

</action>

***Hop Sample configuration:***

<action>

<insertICardAction>

<nextStep> <!-- insert a valid IC, jumping to next page -->

<nextPanel>370</nextPanel>

<display>370005, 370006</display>

<hop>370007</hop> <!-- button id in next page, allow to jump over next page370 -->

</nextStep>

<exception> <!-- no valid IC found, jumping to a page-->

<nextPanel>200</nextPanel>

<display>201</display>

</exception>

<animationTip>/image/card.gif</animationTip> <!--animation file prompt user to insert card, optional setting-->

</insertICardAction>

</action>

#### writeICardAction

Just like sendMsgAction, you can configure this action to write data to IC card after successful transaction. The difference between writeICardAction and sendMsgAction is specific function, because there probably is an error happening in writing an IC card, so we have to send a balanced message rolling back previous transaction to convergent server.

This action configuration is totally same as [sendMsgAction](#_sendMsgAction)

***Sample configuration:***

<action>

<writeICardAction>

<cardType>gzCard</cardType>

<cardAction>

<prsCode>pay\_ICbiao</prsCode>

<conditions>201801214,201801216</conditions>

<nextStep>

<nextPanel>201888</nextPanel>

<display>201886</display>

</nextStep>

<exception>

<nextPanel>201899</nextPanel>

<display>201896</display>

</exception>

<checkRules>

<notNull>

<content>201801214</content>

<errMsg>请点击卡号框右侧按钮进行读卡</errMsg>

</notNull>

<notNull>

<content>201801216</content>

<errMsg>请用密码键盘输入密码</errMsg>

</notNull>

</checkRules>

</cardAction>

</writeICardAction>

</action>

#### moveCursorAction

This action will move cursor when you hit up/down arrow key.

***Sample configuration:***

<action>

<moveCursorAction>

<upCursor>106</upCursor>

<downCursor>105</downCursor>

</moveCursorAction>

</action>

#### printPassbookAction

This action will implement passbook-printing function

***Sample configuration:***

<action>

<printPassbookAction>

<tableId>50571</tableId> <!-- table component id which will show records -->

</printPassbookAction>

</action>

#### playVideoAction

This action will implement video-playing function, standard MPEG-4 format works well

***Sample configuration:***

<action>

<playVideoAction>

<media>./media/video/Smart-Community.mp4</media>

<replayInterval>30</replayInterval> <!—interval between two playing, unit is second -->

</playVideoAction>

</action>

#### showClockAction

This action will implement clock-showing function

***Sample configuration:***

<action>

<setFontAction>

<fontSize>28</fontSize>

</setFontAction>

<showClockAction>

<!-- you can learn how to define date-time format -->

<!-- https://docs.oracle.com/javase/8/docs/api/java/util/Formatter.html#Formatter-java.util.Locale-

default is <format>%1$tc</format> -->

<format>%1$tT</format>

</showClockAction>

</action>



#### uploadFileAction

This action will allow user to choose a file from external usb disk and upload to server

***Sample configuration:***

<action>

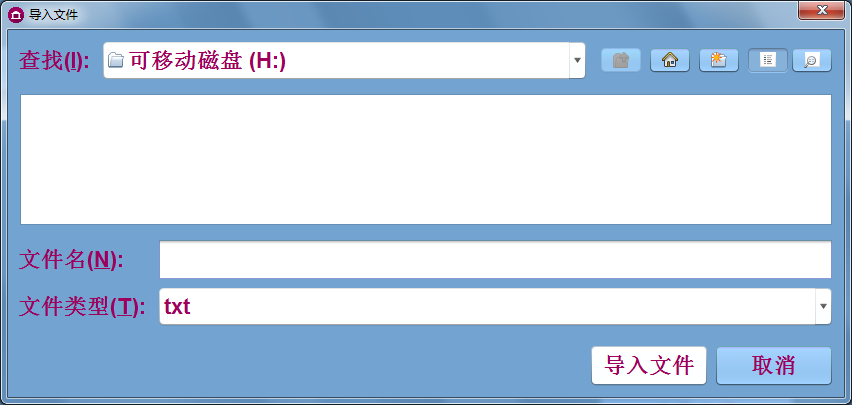
<uploadFileAction>

<filter>txt</filter>

<uploadPath>ftp://127.0.0.1/usb/upload/</uploadPath>

</uploadFileAction>

</action>



#### balanceTransAction

This action will allow user to launch transactional message flow between client and server, if the communication is timeout, client will send another balanced message against previous one.

This action configuration is totally same as [sendMsgAction](#_sendMsgAction)

***Sample configuration:***

<action>

<balanceTransAction>

<prsCode>pay\_pubiao</prsCode>

<conditions>201501014,201501016,201501012</conditions>

<nextStep>

<nextPanel>201588</nextPanel>

<display>201586</display>

</nextStep>

<exception>

<nextPanel>201599</nextPanel>

<display>201596</display>

</exception>

<checkRules>

<notNull>

<content>201501014</content>

<errMsg>请点击卡号框右侧按钮读取银行卡</errMsg>

</notNull>

<notNull>

<content>201501016</content>

<errMsg>请用密码键盘输入密码</errMsg>

</notNull>

</checkRules>

</balanceTransAction>

</action>

#### encryptKeyboardAction

This action will allow user to use keyboard encryption mode for password component. When encryptKeyboardAction is set on password, any true character will be not shown in password, keyboard will use hardware encryption to get user’s credentials.

***Sample configuration:***

<action>

<encryptKeyboardAction>

<moneyId>201501012</moneyId> --set money component id, it value will be calculated MAC . Mandatory configuration

<nextTrigger>201801219</nextTrigger > --set next step button id, optional configuration

<cancelTrigger>201801219</cancelTrigger > --set back step button id, optional configuration

</encryptKeyboardAction>

</action>

#### ejectCardAction

This action will allow to forcibly eject bank card injected.

***Sample configuration:***

<action>

<ejectCardAction>

<cardType>HISCard</cardType> ---card type

</ejectCardAction>

</action>

#### withdrawCardAction

This action will allow to forcibly withdraw bank card injected.

***Sample configuration:***

<action>

<withdrawCardAction>

<cardType>HISCard</cardType>

</withdrawCardAction>

</action>

#### hideAction

This action will allow to hide current application system.

***Sample configuration:***

<action>

<hideAction/>

</action>

#### showGridAction

This action will allow to define shape of grid.

***Sample configuration:***

<action>

<showGridAction>

<row>1</row> --line count per page

<column>2</column> --column count per page

<hgap>15</hgap> --horizon cell gap

<vgap>15</vgap> --vertical cell gap

</showGridAction>

</action>

#### numericPadAction

This action will allow to define shape of grid.

***Sample configuration:***

<action>

<numericPadAction>

<targetId>8</targetId> --designate target component id, only support TextField

</numericPadAction>

</action>

### System supported driver types

<xs:enumeration value="magnetCard"/>

<xs:enumeration value="ICCard"/>

<xs:enumeration value="sensorCard"/>

<xs:enumeration value="twoInOneCard"/>

<xs:enumeration value="printer"/>

<xs:enumeration value="keyboard"/>

<xs:enumeration value="gzCard"/>

<xs:enumeration value="PIDCard"/>

<xs:enumeration value="HISCard"/>

<xs:enumeration value="PatientCard"/>

<xs:enumeration value="pkg8583"/>

<xs:enumeration value="Unknown"/>

### System variable

System variable definition is useful information in current system. You can get it via ${var\_name} just like environment variable in operation system. Current system variable table as below

|  |  |
| --- | --- |
| Var Name | Description |
| ${TERMINAL\_ID} | Current terminal id of machine |
|  |  |