Developer's Guide

RISC add-ons

Inhaltsverzeichnis

Adding own JavaScript Files

Where to place and how to register own .js files

When calling a RISC-page (e.g. "http://xyz/xyz/abc.risc?ccstyle=defaultrisc") then internally an HTML page is generated at runtime that contains the corresponding HTML and JavaScript to start the page. Inside this generated HTML page the JavaScript-files (*.js) are listed that are part of the page processing.

By default all JavaScript files are included that form the RISC-client. But you may define own ".js" files on your own and add them to the list. The purpose of own files is to either extend or sometimes override the default JavaScript that comes with the RISC-client.

Add .js files to /webcontent/eclnt/risc/plugin

Any ".js"-file that you add into this directory will automatically be loaded. There is no additional registration required.

Registering own JavaScript files inside system.xml

You may add own JavaScript files by updating the system.xml configuration file, located in "/webcontent/eclntjsfserver/config/system.xml".

Example:

```
<system>

...
...

    <riscclientscript src="xxx/yyy/zzzz.js" type="text/javascript" />
    <riscclientscript src="aaa/bbb/ccc.js" type="text/javascript"/>

...
...

</system>
```

The *.js files that you add must reside within the webcontent directory of your project:

```
<project>
  webcontent
    aaa
      bbb
        ccc.js
    xxx
      yyy
        zzzz.js
```

Rules / Conventions

CaptainCasa uses two the namespaces for all its variables / functions that are defined on window-level ("global level"):

- "RISC*" for all RISC related issues – this is the rendering of components
- "CC*" for all CaptainCasa related issues – this is the binding level between the rendering and the server side processing

Do not use this namespace on global level!

Overriding / Influencing the RISC Client by own JavaScript

The JavaScript sources that you add are loaded after the loading of the default RISC JavaScript files. This means: you may override certain behavior of the RISC client.

While JavaScript itself is 100% open and allows you to do “everything” we strongly recommend to only extend the behavior of the RISC client where we explicitly provide possibilities for extension!

Multi Language / Internationalization

The RISC client may be configured by the server side to use a certain language / country information. While the language is selecting the client side literals, the country defined the formatting of values (e.g. dates and big decimal numbers). You may extends by adding the following information:

```
RISCI18NDataFormat.DE =
{
    decimalSeparator: ",",
    thousandsSeparator: ".",
    dateSeparator: ".",
    timeSeparator: ":",
    ymdSequence: "dmy"
};

RISCI18NDataLiterals.de =
{
    monthNames: [ "Januar", "Februar", "M\u00E4rz", "April", "Mai", "Juni",
"Juli", "August", "September", "Oktober", "November", "Dezember"],
    weekdayNamesShort: ["Mo", "Di", "Mi", "Do", "Fr", "Sa", "So"],
    literals:
    {
        ctrl: "Strg",
        alt: "Alt",
        shift: "Shift",
        messageIncorrectValueReset: "Ung\u00fcltiger Wert \"${value}\" - der Wert
wurde auf seinen vorigen Stand zur\u00fckgesetzt.",
        hintIncorrectValueReset: "Das Eingabeformat ist: \"${format}\"",
        hintFromIncorrectValueReset: "Der minimale Wert ist: \"${value}\"",
        hintToIncorrectValueReset: "Der maximale Wert ist: \"${value}\"",
        messageIncorrectValue: "Ung\u00fcltiger Wert \"${value}\" - bite
pr\u00fcfen Sie Ihre Eingabe.",
        ok: "OK",
        color_rgb: "RGB",
        color_red: "Rot",
        color_green: "Gr\u00fcn",
        color_blue: "Blau",
        color_hexcode: "Hex Code",
        color_palette: "Palette",
        selectFiles: "Zu sendende Dateien ausw\u00E4hlen...",
        fileupload_clear: "Liste l\u00f6schen",
        fileupload_clearFile: "Entfernen",
        fileupload_selectFiles: "Datei ausw\u00E4hlen...",
        fileupload_selectedFiles: "Ausgew\u00E4hlte Datei(en):",
        fileupload_upload: "Zum Server senden",
        fileupload_cancel: "Abbrechen",
        fileupload_maxfilesize: "Maximale Uploadgr\u00f6\u00dfe (${value} kB)
\u00fcberschritten.",
        fileupload_maxsinglefilesize: "Maximal Dateigr\u00f6\u00dfe (${value}
kB) \u00fcberschritten.",
        fileupload_maxnumberoffiles: "Maximale Dateianzahl (${value})
\u00fcberschritten.",
        filechooser_clear: "Liste l\u00f6schen",
        filechooser_clearFile: "Entfernen",
        filechooser_selectFiles: "Datei ausw\u00E4hlen...",
        filechooser_selectedFiles: "Ausgew\u00E4hlte Datei(en):",
        filechooser_upload: "\u00dcbernehmen",
        filechooser_cancel: "Abbrechen",
        formatAdviceYear: "JJJJ",
        formatAdviceMonth: "MM",
        formatAdviceDay: "TT",
        formatAdviceHour: "SS",
        formatAdviceMinute: "MM",
        formatAdviceSecond: "ss",
```

```
        copyToClipboard: "Kopieren Sie den Inhalt mit ctrl-c in das Clipboard.",
        copyToClipboardTitle: "In das Clipboard kopieren",
        today: "Heute",
        todayShort: "Heute",
        confirmExit: "Wollen Sie diese Anwendung wirklich verlassen?",
        controlButton: "Schaltfl\u00e4che",
        controlButton_selected: "aktiv",
        controlButton_unselected: "inaktiv",
        controlCheckBox: "Auswahl",
        controlCheckBoxValue_true: "ein",
        controlCheckBoxValue_false: "aus",
        controlCheckBoxValue_undefined: "ungesetzt",
        controlField: "Feld",
        controlFieldFormat_decimal: "Zahl",
        controlFieldFormat_integer: "Ganze Zahl",
        controlFieldFormat_date: "Datum",
        controlFieldFormat_time: "Uhrzeit",
        controlFieldFormat_datetime: "Datum mit Uhrzeit",
        controlFieldFormat_color: "Farbe",
        controlHeaderLabel: "Spaltenkopf",
        controlHeaderLabel_sort_ascending: "aufsteigend sortiert",
        controlHeaderLabel_sort_descending: "absteigend sortiert",
        controlLabel: "Text",
        controlOutlookbarButton: "Akkordeonauswahl",
        controlProgressBar: "Fortschrittsanzeige",
        controlRadioButton: "Mehrfachoption",
        controlRadioButtonValue_true: "ein",
        controlRadioButtonValue_false: "aus",
        controlSingleTreeNode: "Knoten",
        controlSingleTreeNodeLevel: "Ebene",
        controlSingleTreeNodeStatus0: "offen",
        controlSingleTreeNodeStatus1: "geschlossen",
        controlSingleTreeNodeStatus2: "Endknoten",
        controlTabbedLineButton: "Bereichsauswahl",
        controlSlider: "Schieberegler",
        controlSlider_min: "min",
        controlSlider_max: "max",
        zzzzz: "zzzzz"
    }
};
```

Please note: Charactes with ASCII code >=128 have to be escaped!

And please also note: this is the language / country setting of the client. Which may be totally different from the language / country setting on server side. Please check the Developer's Guide for more information on Internationalization.

Predefined exits

There are predefined exits within the default RISC-client. An exit is an API-call that the RISC-client executes to some outside JavaScript-code. If the outside code exists then it is called, if not then a default behavior is executed.

All Exits – View /webcontentcc/eclnt/risc/plugin/plugin.js_template

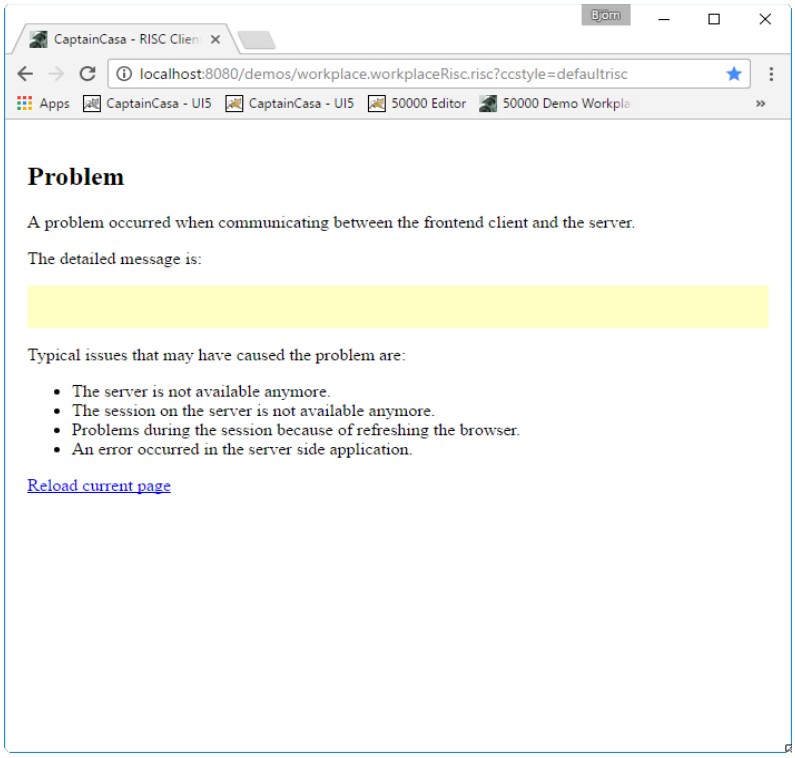
The file “plugin.js_template” is the one that lists all the exits that are provided by the RISC-client. For implementing the exits on your own you may just copy the “plugin.js_template” file as “/webcontent/eclnt/risc/plugin/plugin.jsp” into you project and then implement these exits that you want to implement. You do not have to implement all exits, so you may comment out or remove the corresponding JavaScript-methods.

The number of exits is quite low at the moment (Sep 2017) – but will be increased step by step.

Exit - Providing an own Error Page Layout

In case the client runs into some error (e.g. the server is not reachable), it shows some

error page:



You may customize this layout by overriding the following method:

```
/**
 * Returns the HTML that is output if an error occurs during communication.
 */
CCPlugin.createClientErrorPageHTML = function(pReloadUrl,pMessage)
{
    var sb = new Array();
    sb.push("<div style='font-family:Open
Sans;width:100%;height:100%;padding:20px;background-color:#FFFFFF;-moz-box-sizing:
border-box;-webkit-box-sizing: border-box;box-sizing: border-
box;overflow:auto'>");
    sb.push("<p style='font-family:Open Sans;font-size:25px; font-
weight:bold'>Problem</p>");
    sb.push("<p style='font-family:Open Sans;font-size:12px'>");
    sb.push("A problem occurred when communicating between the client and the
server.");
    sb.push("</p>");
    if (pMessage != null && pMessage != "")
    {
    sb.push("<hr>");
    sb.push("<p style='font-family:Open Sans;font-size:12px'>");
    sb.push("Details:");
    sb.push("</p>");
    sb.push("<p>");
    sb.push("<div style='width=100%;background-color:#F0F0F0;padding:10;font-
family:Courier New;font-size:11px;border:0px #c0c0c0 solid'>"+pMessage+"</div>");
    sb.push("</p>");
    }
    sb.push("<hr>");
    sb.push("<p style='font-family:Open Sans;font-size:12px'>");
    sb.push("Typical issues that may have caused the problem are:");
    sb.push("<ul>");
    sb.push("<li style='font-family:Open Sans;font-size:12px'>The server is not
available anymore.</li>");
    sb.push("<li style='font-family:Open Sans;font-size:12px'>The session on the
server is not available anymore.</li>");
    sb.push("<li style='font-family:Open Sans;font-size:12px'>An error occurred in
the server side application.</li>");
    sb.push("</ul>");
    sb.push("</p>");
    sb.push("<hr>");
    sb.push("<p>");
    sb.push("<a
href='javascript:RISCConfirmExit.avoidBlocking();location.reload();' style='font-
```

```
size:12px; font-weight:bold'>Reload current page</a>");
    sb.push("</p>");
    sb.push("</div>");
    return sb.join("\n");
};
```

Recording http traffic so that CaptainCasa can replay

The CaptainCasa server environment comes with a special feature that is inevitably important for solving complex bugs within the CaptainCasa frontend processing – that (as usual...) only happen inside your own environment and cannot be reproduced at CaptainCasa side.

There is a certain recording of all http traffic that is exchanged between the browser and the server. The recorded files can then be sent to CaptainCasa so that CaptainCasa can replay the frontend – the requests are responded from the files that you recorder. The result: CaptainCasa can exactly replay your scenario.

Security Issues

The recorded information contains all the traffic between your browser and your server. If the content contains critical information (including passwords, business information, …) then this is logged as consequence – and readable as clear text. So only use the recording feature in demo systems with demo user accounts...! If in doubt about security, please contact CaptainCasa.

Configuration

Recording (and replaying) is done by a servlet filter which is part of the normal CaptainCasa server environment.

Register the filter in the web.xml as follows:

```
  <filter>
    <filter-name>org.eclnt.jsfserver.util.CompressionFilter</filter-name>
    <filter-class>org.eclnt.jsfserver.util.CompressionFilter</filter-class>
  </filter>
  <filter>
    <filter-name>org.eclnt.jsfserver.util.ResponseLoggerFilter</filter-name>
    <filter-class>org.eclnt.jsfserver.util.ResponseLoggerFilter</filter-class>
  </filter>
  <filter>
    <filter-name>org.eclnt.jsfserver.util.SecurityFilter</filter-name>
    <filter-class>org.eclnt.jsfserver.util.SecurityFilter</filter-class>
  </filter>
  <filter>
    <filter-name>org.eclnt.jsfserver.util.ThreadingFilter</filter-name>
    <filter-class>org.eclnt.jsfserver.util.ThreadingFilter</filter-class>
  </filter>

  <filter-mapping>
    <filter-name>org.eclnt.jsfserver.util.CompressionFilter</filter-name>
    <url-pattern>*.jsp</url-pattern>
  </filter-mapping>
  <filter-mapping>
    <filter-name>org.eclnt.jsfserver.util.CompressionFilter</filter-name>
    <url-pattern>*.xml</url-pattern>
  </filter-mapping>
  <filter-mapping>
    <filter-name>org.eclnt.jsfserver.util.ResponseLoggerFilter</filter-name>
    <url-pattern>*</url-pattern>
  </filter-mapping>
  <filter-mapping>
    <filter-name>org.eclnt.jsfserver.util.SecurityFilter</filter-name>
    <url-pattern>*.jsp</url-pattern>
  </filter-mapping>0
    <filter-mapping>
    <filter-name>org.eclnt.jsfserver.util.ThreadingFilter</filter-name>
    <url-pattern>*.jsp</url-pattern>
  </filter-mapping>
```

Your filter configuration might look a little bit different (e.g. you might not have the security filter installed). Please install the filter “behind” the CompressionFilter and “in front of” the SecurityFilter and ThreadingFilter. Please note: the sequence of filters is defined by the sequence of filter-mapping-definitions – and NOT by the sequence of filter-definitions.

Please also pay attention: the filter MUST NOT be active in production scenarios! It is only to be used in development scenarios. (Otherwise any traffic is recorded!)

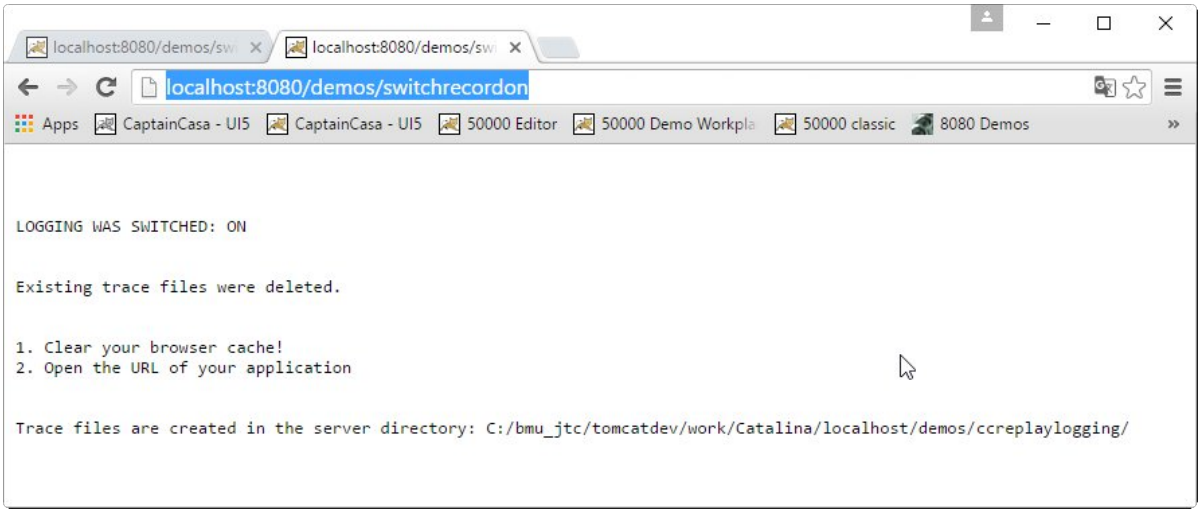
(Actually we added some protection mechanism here...: the filter is only active if the query is directed to the address http(s)://localhost:xxxxx. So you cannot invoke the filter if working from outside the local environment. But, despite of this, we strongly recommend to deactivate the filter in production scenarios!)

Recording

We explain the sequence of steps by using as example our demo workplace, which is available as “http://localhost:8080/demos/workplace.workplaceRisc.risc?ccstyle=defaultrisc” after installation.

Step 1 – Switch the recording to “on”

To do so you just need to call URL “http://localhost:8080/demos/switchrecordon”:



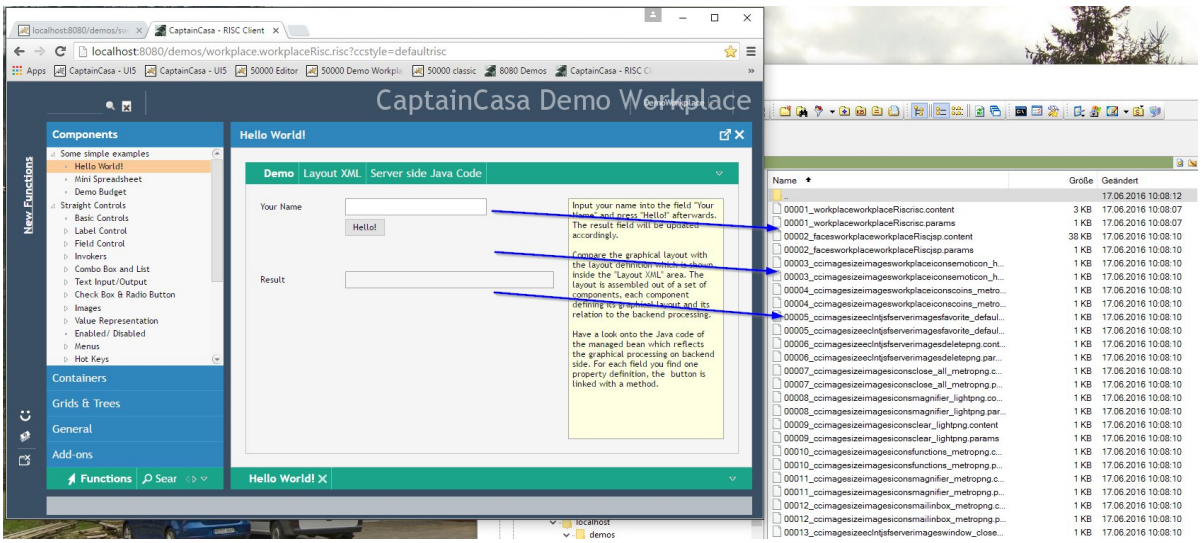
The text within the screen tells you the directory, in which the recorded files are placed.

Step 2 – Clear your browser cache

It is important that all requests are recorded that are required to run your page against the recorded log afterwards. So: clear your browser cache!

Step 3 – Run the URL of your application

Now you may start (e.g. in a second tab) the URL of your application just as normal. In the background on server side each request/response is recorded.



Per request/response two files are written:

- The content of the response (".content")
- Parameters of the response (".params")

Step 4 - Switch the recording to "off"

Call URL "http://localhost:8080/demos/switchrecordoff":

Step 5 - Send the recorded information to CaptainCasa

Now you may zip all of the recorded files to one file - and send it to CaptainCasa - together with the start-URL that you used.

...you want to replay on your own?

Well, replaying typically is done on CaptainCasa side, but you may want to do it for any reason on your own.

Just call URL "http://localhost:8080/demos/switchreplayon" and now the server will not work normally anymore but will fulfill all request by finding responses in the recorded files.

When calling your application URL "http://localhost:8080/demos/workplace.workplaceRisc.risc?ccstyle=defaultrisc" then all responses are coming "immediately" from the file system.

Finally switch the replay mode off by calling

"http://localhost:8080/demos/switchreplayoff".

Transferring Enterprise Client from Swing/FX to RISC

There's a high level of compatibility between Swing/FX client and RISC client. Nevertheless there are some issues to take care of.

The way to go...

Install the latest version

When coming from the JavaFX/Swing environment then please download the most up to date version from http://www.CaptainCasa.com/java. This is the “big” download, in which also the Swing and FX-client is included. The “normal” download only includes the RISC-HTML client anymore.

Upgrade your project

Just normally upgrade your project. This is typically done by opening the project within the CaptainCasa toolset. A dialog will appear in which you are informed that your project needs some update – and a corresponding button needs to be pressed.

Update your web.xml

There are a couple of new items in the web.xml that need to be added. Copy all the CaptainCasa items from web.xml_template into your web.xml.

Reload/Redeploy your application

Use the “Reload” within the toolset to redeploy your application.

Start your page

Start your outest page (starting page) within the browser in the following way:

```
http://<host>:<port>/<webapp>/[<dir>.]<page>.risc?ccstyle=defaultrisc
```

Example: you run the application on the default Tomcat, your application has the name “testapp”. Inside your application there is an “outest.jsp”, directly located in the webcontent-directory. In this case the URL is:

```
http://localhost:50000/testapp/outest.risc?ccstyle=defaultrisc
```

Example: the same situation but now the outest page is located in the “webcontent/pages”-directory:

```
http://localhost:50000/testapp/pages.outest.risc?ccstyle=defaultrisc
```

Incompatibilities

BGPAINT Attribute

- The BGPAINT attribute is not supported with all the options that were available in the Swing/FX environment. There is a certain compatibility level – but the best idea is to switch to these BGPAINT-commands that are available for all platforms (error, mandatory – and all the commands starting with “bg”, e.g. “bgimage”).

FILEUPLOAD* components

- In Swing/FX the file name always was transferred with its full absolute path (e.g. “c:/xyz/xyz/abc.txt”). In RISC the client only transfers the file name – due to security reasons (e.g. “abc.txt”).

Removing all “non-RISC”-parts

Files to be removed from your project

When having used CaptainCasa Enterprise Client with its Swing or with its JavaFX client then your project contains some files which are obsolete when “only” using the RISC client anymore.

These files may be removed from the project in order to decrease its size and in order to tidy up. When using the default project layout then these files are part of the “webcontentcc” directory – so they are part of the content, that is added by CaptainCasa into your project.

```
webcontentcc
      eclnt
            lib                        <== remove
            libfx                      <== remove
            ui5                        <== remove
      eclntjsfserver
            ht                         <== remove
            htstyle                    <== remove
            javainstall                <== remove
            styles
                  default              <== remove
                  defaultfx            <== remove
                  ccdark               <== remove
                  cceditor             <== remove
                  cceditorlight        <== remove
                  cceditorlightfx      <== remove
                  ccmetrofx            <== remove
                  cctouch              <== remove
                  cctouchlight         <== remove
      WEB-INF
            lib
                  jnlp-servlet.jar     <== remove
```

Cleaning up your Installation

The best was to clean up your CaptainCasa installation is:

- Remove the existing installation
  - You may “remove” your existing installation by just renaming the directory of the installation – in this case you still have your old installation pre-served.
- Install CaptainCasa-RISC in the directory, in which your installation was done

There is only one type of content which you need to preserve, i.e. which you have to save before removing and which you have to bring in again after installation: the project definitions of the CaptainCasa toolset.

So please save all files in...

```
<installdir>
      server
            tomcattools
                  webapps
                        editor
                              config
                                    projects
                                          *.xml        <== to be saved
```

...before removing the installation. And reapply the files at the same location after having installed CaptainCasa-RISC.

After re-install: re-deploy your projects

Part of the CaptainCasa installation is a Tomcat that is used for deploying your projects.

```
<installdir>
      server
            tomcat
```

This tomcat is “empty” if having removed the existing installation and having re-installed. In order to deploy your projects you need to open the project within the CaptainCasa toolset and you need to execute the “Reload Server” function.