unify Harmonize your enterprise

OpenScape UC Application V7 Web Client 1.0 SDK

Programming Guide

A31003-S5070-R102-3-7620

Our Quality and Environmental Management Systems are implemented according to the requirements of the ISO9001 and ISO14001 standards and are certified by an external certification company.

Reference No.: A31003-S5070-R102-3-7620

unify Harmonize your enterprise

unify.com

Contents

Contents

History of Changes

Date	Changes	Reason
2012-07-09	Initial creation	
2014-01-09	The name “Unify” has been introduced.	
2014-04-30	“Standard Duplex (large) deployment” has been replaced by “Large Deployment”. The Very Large Deployment has been introduced.	
2014-04-30	The name “Web Client” has been replaced by the name “Web Client 1.0”.	FRN7955

1 Introduction

1.1 General

The OpenScape UC Application Web Client 1.0 SDK enables a user to change the appearance of the OpenScape UC Application's web interface according to his wishes and to add additional functionality to it:

- Customization
  - Background colors of the web portal
  - Background images in the web portal
  - Logo in the web portal
  - Language dependent modifications
- Integration
  - Click to Dial
  - Click to Release
  - Click to Accept

UNIFY GMBH & CO. KG GRANTS THE USER ACCESS TO THE OPENSCAPE UC APPLICATION SDK "AS IS" AND WITHOUT ANY WARRANTY WHATSOEVER. THE USER BEARS THE UNLIMITED RISK FOR THE USE OF THE OPENSCAPE UC APPLICATION SDK. IMPLEMENTING OTHER FEATURES THAN THOSE DEFINED IN THIS DOCUMENT MAY CAUSE SEVERE DAMAGE TO THE SYSTEM, INCLUDING LOSS OF DATA.

UNIFY GMBH & CO. KG MAKES NO GUARANTEE THAT THE OPENSCAPE UC APPLICATION SDK FUNCTIONS SATISFY THE DEMANDS OF THE USER, THAT OPENSCAPE UC APPLICATION SDK INTEROPERATE AS SELECTED BY THE USER, THAT THESE WILL OPERATE WITHOUT INTERRUPTION AND WITHOUT FAULTS OR THAT ALL SOFTWARE ERRORS CAN BE RECTIFIED.

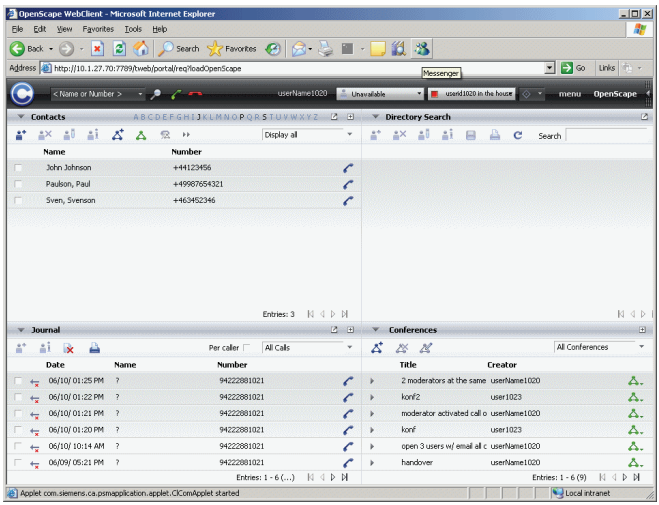
1.2 Addressees

This documentation is for professional services of Unify Software and Solutions GmbH & Co. KG and selected partners. I implement additional UC functionalities provided by the OpenScape UC Application Web Client 1.0 SDK, you need to have a basic understanding of HTML programming and JavaScript programming. If you want to modify the OpenScape UC Application Web Client 1.0 GUI, only picture paths and/or color values in config files have to be edited.

2 Customization

The OpenScape UC Application Web Client 1.0 SDK provides a simple way to adapt the web portal to an existing corporate style of a company by changing the background colors, the logo and the background images of the web portal. It is not necessary to have CSS knowledge. The offered properties are activated in a config file by removing the comment character # in front of the corresponding property and by entering the appropriate RGB value, e.g. `#f8fff8`, for the wanted color or the image path to the wanted company logo.

The following picture shows the OpenScape UC Application Web Client 1.0 before customization.



NOTE: If you want to disable the home page editor, i. e. remove the **Home Page** menu item, open `<install>/WebSpace/Portal/webapps/tweb/WEB-INF/language/common/customSettings.properties`
in an editor, go to the the line
`CUSTOM.SUPPORT_EDIT_HOMEPAGE = true`
change the value `true` to `false` and save the file.

2.1 Supported Web Browsers

The released browsers are documented in the *OpenScape UC Application Plannning Guide*.

- JavaScript must be activated.
- Pop-up blockers must be deactivated.
- Ad blockers must be deactivated.
- JRE 1.5 or 1.6 must be used.

2.2 Customization Files

Customizing the OpenScape UC Application Web Client 1.0 GUI is done by editing the `customStyles.properties` file and the `styles_<lang>.properties` files. The `customStyles.properties` file determines language independent features and uses the `styles_<lang>.properties` files which determine language dependent features. A `styles_<lang>.properties` file has the same functionality and structure as the `customStyles.properties` file but it is only valid for the specified language, i.e. a start for a `customStyles.properties` file is to make a copy of the `customStyles.properties` file. Thus, it is possible to implement a customization for one language and a completely different customization for another language. The language used by the browser determines which `styles_<lang>.properties` file is used. The `styles_<lang>.properties` files must be in the same directory as the `customStyles.properties` file.

In order to prevent the customization files from being overwritten, when the OpenScape UC Application is updated, they have to be copied to a customized path (see step 2f on page 12).

Execute the following steps for customization:

1. The `customStyles.properties` file and the `styles_<lang>.properties` files are stored in the default directory:

   `<install>/WebSpace/Portal/webapps/tweb/WEB-INF/language/styles/`

   The default value of `<install>` on a Linux operating system is `/opt/siemens/HiPathCA`.

   NOTE: In Large Deployment, the OpenScape front-end services (web servers) do not reside on the application computer (UC backend computer) but on one or several separate computers, the front-end computers. If you deploy this scenario, this step and the following step must not be executed on the application computer but on one of the front-end computers. In Very Large Deployment, execute the next step in each cluster on one the front-end computers.

   See the *OpenScape UC Application Planning Guide* for more information about deployment scenarios.

   ---

2. Create a new directory, the customized path, where the customization files should be copied to by executing the substeps below. The name of the directory, e.g. `MyHtml`, is arbitrary but its location is not.

   a) Go to the directory `<install>/WebSpace/Portal/webapps/tweb/`.

   b) Create the customized path by executing the command `mkdir MyHtml`.

   c) This directory has to be registered in the configuration file `<install>/config/common/global.cfg`. Open this file in an editor and set the `CustomizedPath` property to the name of the created directory, i. e. `MyHtml`.

   ```
   CustomizedPath=MyHtml
   ```

   IMPORTANT: Consider that directoy names and file names always are case-sensitive in Linux.

   ---

   d) Create the subdirectory `language/styles` in the `MyHtml` directory.

e) Go into the `styles` directory.

f) Copy the `customStyles.properties` file and all `styles_<lang>.properties` files from the default directory into this directory.

The files in this directory will never be overwritten by an update of the OpenScape UC Application but the files in the default directory will be. The OpenScape front-end services always look first whether there are a `customStyles.properties` file and `styles_<lang>.properties` files in this directory and uses them. Only if the OpenScape front-end services cannot find these files, they look for these files in the default directory.

g) Edit the files according to the information and instructions in the sections Section 2.3, “Customizable Properties”, on page 13, Section 2.4, “Proceedings for Customization”, on page 23 and Section 2.3.3, “Samples”, on page 18.

3. If you deploy Large Deployment or Very Large Deployment, the following additional substeps have to be executed.

a) Compress the files and subdirectories in the following directory into a ZIP file:

`<install>/WebSpace/Portal/webapps/tweb/MyHtml`

Do not compress this directory itself into a ZIP file.

b) Name the file ZIP file `MyHtml.zip`, i.e. the first part of the file name has to be the name of the customized path that has been assigned to the `CustomizedPath` property in the `<install>/config/common/global.cfg` file (see step 2c on page 11).

IMPORTANT: Make sure that the file name has the same uppercase letters and lower case letters as the corresponding directory.

---

c) Rename the file to `MyHtml.war`.

d) Copy this file into the `<install>/config/services/default/Httpd/Portal/` directory **on the application computer**.

e) Open the `<install>/config/services/default/Httpd/Portal/http-custom.cfg` file on the application computer and set the `CUSTOM_WARS` property to `tweb/MyHtml`.

```
<?x set CUSTOM_WARS = "tweb/MyHtml" ?>
```

Now, whenever the OpenScape front-end services are restarted on a front-end computer, and if the `MyHtml.war` file on the application computer is newer than the `MyHtml.war` file on the front-end computer, the `MyHtml.war` file is copied from the application computer to this front-end computer and unpacked. The `MyHtml.war` file on a front-end computer always gets the same creation date as the the `MyHtml.war` file on the application computer.

2.3 Customizable Properties

The following lines show a sample for the `<install>/WebSpace/Portal/webapps/tweb/WEB-INF/language/styles/customStyles.properties` file. It is on the front-end computer in Large Deployment and Very Large Deployment or on the application computer in the other deployment scenarios:

```
# Set CUSTOM.STYLES_ENABLED = true to enable customizing via properties below
CUSTOM.STYLES_ENABLED = false

# Portal customizing properties. Remove comment character # to enable property.
#CUSTOM.PORTAL_MENUBAR_IMAGE = /CaUIs/CmOS/images/bg/world_background.jpg
#CUSTOM.PORTAL_MENUBAR_BGCOLOR = #366dad
#CUSTOM.PORTAL_MENUBAR_HEIGHT = 44
#CUSTOM.LOGOIMAGE_COMPANY = /CaUIs/CmOS/images/logos/world_header.jpg
#CUSTOM.LOGOIMAGE_WIDTH = 72
#CUSTOM.LOGOIMAGE_HEIGHT = 42
#CUSTOM.LOGOIMAGE_MARGIN = 0

# Portlets customizing properties
#CUSTOM.PORTLET_LIST_BGCOLOR = #eeffee
#CUSTOM.PORTLET_LIST_FGCOLOR= #003300
#CUSTOM.PORTLET_LIST_SELECTED_BGCOLOR = #bbffbb
#CUSTOM.PORTLET_LIST_SELECTED_FGCOLOR = #001100
#CUSTOM.PORTLET_LIST_DETAILED_BGCOLOR= #f8fff8
#CUSTOM.PORTLET_LIST_DETAILED_FGCOLOR = #001100
#CUSTOM.PORTLET_LIST_BORDERCOLOR_TOP = #ffffff
#CUSTOM.PORTLET_LIST_BORDERCOLOR_BOTTOM = #bdd3cc
#CUSTOM.PORTLET_DIALOG_BGCOLOR = #eeffee
#CUSTOM.PORTLET_DIALOG_FGCOLOR = #225522
#CUSTOM.PORTLET_CAPTION_BGCOLOR= #448844
#CUSTOM.PORTLET_CAPTION_FGCOLOR= #ffffff
#CUSTOM.PORTLET_CAPTION_DISABLED_FGCOLOR = #225522
#CUSTOM.PORTLET_CAPTION_INACTIVE_FGCOLOR = #aaeeaa
#CUSTOM.PORTLET_CAPTION_MOUSEOVER_FGCOLOR = #bbffbb
#CUSTOM.PORTLET_CAPTION_BORDERCOLOR_TOP= #66aa66
#CUSTOM.PORTLET_CAPTION_BORDERCOLOR_BOTTOM = #cccccc

# Common customizing properties
#CUSTOM.MESSAGEBOX_BGCOLOR= #aaeeaa
#CUSTOM.MESSAGEBOX_FGCOLOR= #001100
#CUSTOM.POPUPDRAGGERBAR_BGCOLOR = #366dad
#CUSTOM.POPUPDRAGGERBAR_FGCOLOR = #ffffff

# Login page customizing properties
# Position attributes for one coordinate: top, left, right, bottom, center, 40px
#CUSTOM.LOGINIMAGE_COMPANY = /CaUIs/CmOS/images/logos/world_header.jpg
#CUSTOM.LOGINIMAGE_POSITION = right bottom
```

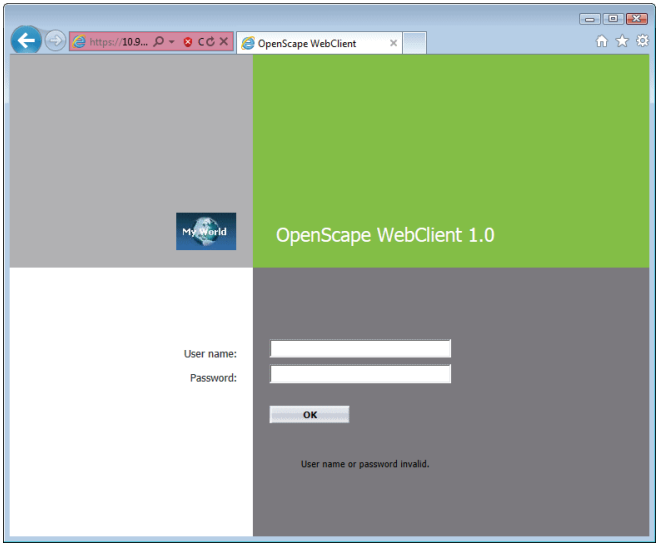
NOTE: If there are two lines in the `customStyles.properties` file defining the same property, the last line is valid, i.e. `#ffffff` overwrites `#366dad` in the following example:

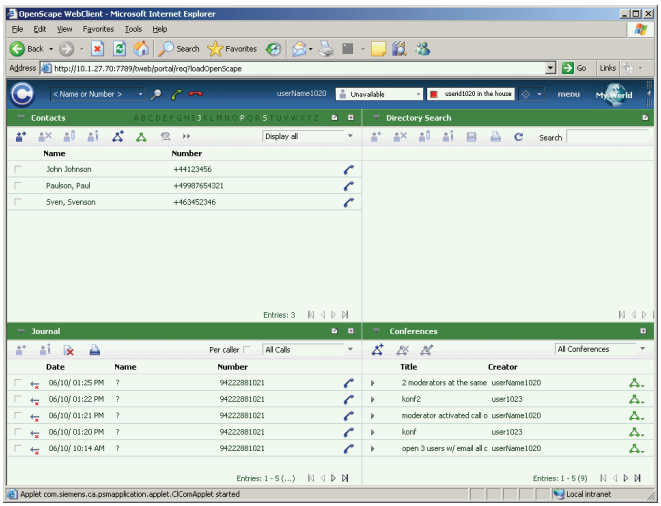
```
CUSTOM.PORTAL_MENUBAR_BGCOLOR = #366dad
CUSTOM.PORTAL_MENUBAR_BGCOLOR = #ffffff
```

NOTE: Do not append a comment to a line. This makes the line invalid.
Example:

```
CUSTOM.PORTLET_CAPTION_BORDERCOLOR_BOTTOM = #cccccc #Old value
```

The following figures show the OpenScape UC Application Web Client 1.0 with the basic customization, i.e. with the `customStyles.properties` file shown above but the `CUSTOM.STYLES_ENABLED` property set to `true` and all occurrences of `#CUSTOM` replaced by `CUSTOM`.



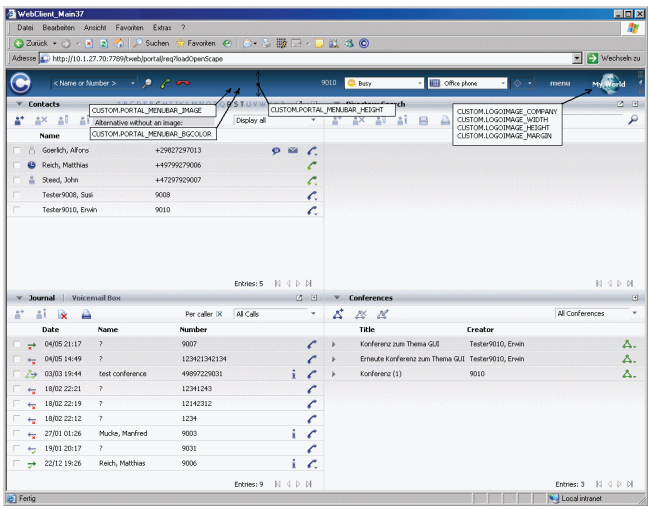


2.3.1 Portal Properties

The following table and figure describe the meaning of the portal properties that are available in the `customStyles.properties` file.

Item	Property	Description
1	CUSTOM.STYLES_ENABLED	If CUSTOM.STYLES_ENABLED is true, customization can be done by using the other properties. If it is false, no customization is activated.
2	CUSTOM.PORTAL_MENUBAR_IMAGE	Path and file name of the picture to be shown in the menubar
3	CUSTOM.PORTAL_MENUBAR_BGCOLOR	Background color of the menubar.
4	CUSTOM.PORTAL_MENUBAR_HEIGHT	Height of the menu bar
5	CUSTOM.LOGOIMAGE_COMPANY	Path and file name of the company logo to be shown in the right corner of the menu bar
6	CUSTOM.LOGOIMAGE_WIDTH	Width of the company logo
7	CUSTOM.LOGOIMAGE_HEIGHT	Height of the company logo
8	CUSTOM.LOGOIMAGE_MARGIN	Margin between company logo and border of the menu bar

Table 1 Portal Properties

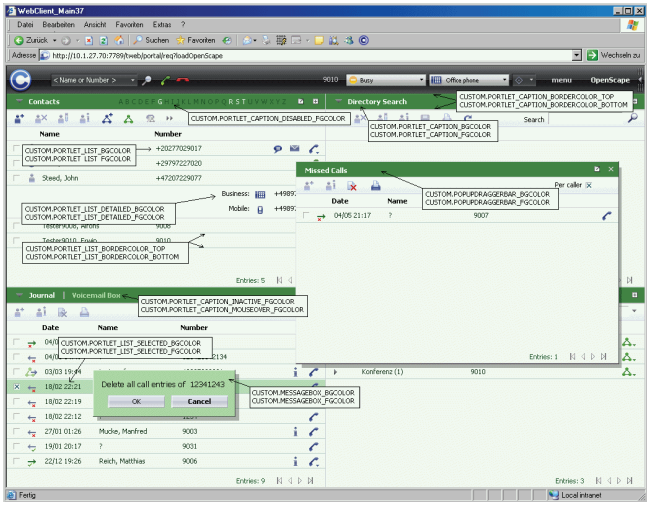


2.3.2 Portlet Properties

The following table and figure describe the meaning of the portlet properties that are available in the `customStyles.properties` file.

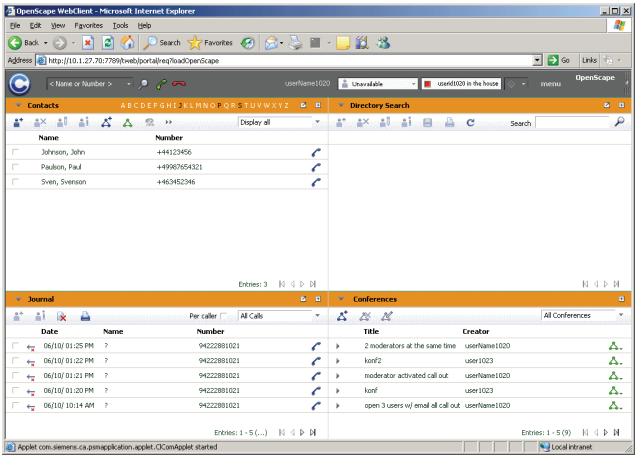
Item	Property	Description
1	CUSTOM.PORTLET_LIST_BGCOLOR	Background color of a list, e.g. user names list
2	CUSTOM.PORTLET_LIST_FGCOLOR	Foreground color of a list
3	CUSTOM.PORTLET_LIST_SELECTED_BGCOLOR	Background color of a selected item in a list
4	CUSTOM.PORTLET_LIST_SELECTED_FGCOLOR	Foreground color of a selected item in a list
5	CUSTOM.PORTLET_LIST_DETAILED_BGCOLOR	Background color of list details
6	CUSTOM.PORTLET_LIST_DETAILED_FGCOLOR	Foreground color of list details
7	CUSTOM.PORTLET_LIST_BORDERCOLOR_TOP	Color of the border line above a list entry
8	CUSTOM.PORTLET_LIST_BORDERCOLOR_BOTTOM	Color of the border line below a list entry
9	CUSTOM.PORTLET_DIALOG_BGCOLOR	Background color of a dialog
10	CUSTOM.PORTLET_DIALOG_FGCOLOR	Foreground color of a dialog
11	CUSTOM.PORTLET_CAPTION_BGCOLOR	Background color of the caption
12	CUSTOM.PORTLET_CAPTION_FGCOLOR	Foreground color of the caption
13	CUSTOM.PORTLET_CAPTION_DISABLED_FGCOLOR	Foreground color of not used components in the caption, e.g. a character that is not used as an intial character of a name in the contacts list
14	CUSTOM.PORTLET_CAPTION_INACTIVE_FGCOLOR	Foreground color of an inactive component in the caption, e.g. voicemail box in the journal
15	CUSTOM.PORTLET_CAPTION_MOUSEOVER_FGCOLOR	Foreground color of a character in the caption when the mouse moves over it
16	CUSTOM.PORTLET_CAPTION_BORDERCOLOR_TOP	Color of the border line above the caption
17	CUSTOM.PORTLET_CAPTION_BORDERCOLOR_BOTTOM	Color of the border line below the caption
18	CUSTOM.MESSAGEBOX_BGCOLOR	Background color of a message box
19	CUSTOM.MESSAGEBOX_FGCOLOR	Foreground color of a message box
20	CUSTOM.POPUPDRAGGERBAR_BGCOLOR	Background color of the caption of a pop-up window
21	CUSTOM.POPUPDRAGGERBAR_FGCOLOR	Foreground color of the caption of a pop-up window

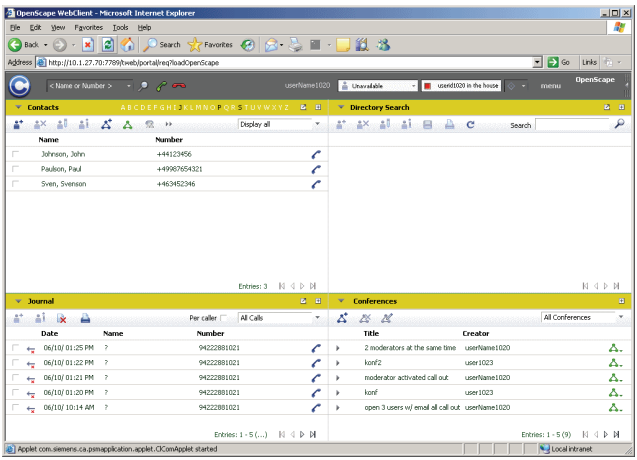
Table 2 Portlet Properties

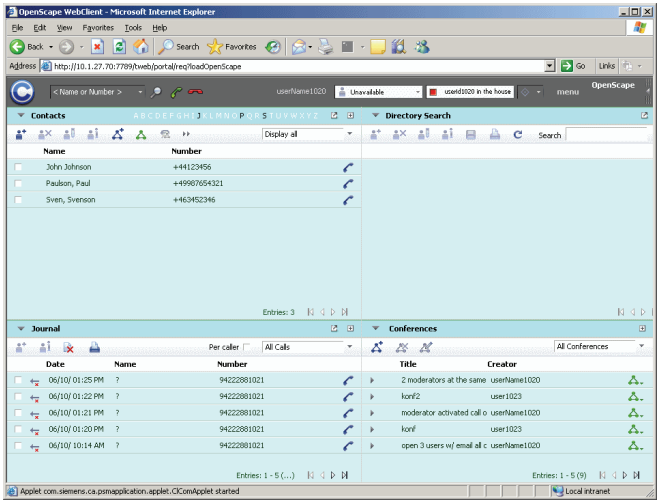

WebClient_Main37
CUSTOM.PORTLET_CAPTION_BORDERCOLOR_TOP
CUSTOM.PORTLET_CAPTION_BORDERCOLOR_BOTTOM
CUSTOM.PORTLET_CAPTION_DISABLED_FGCOLOR
CUSTOM.PORTLET_CAPTION_BGCOLOR
CUSTOM.PORTLET_CAPTION_FGCOLOR
CUSTOM.PORTLET_LIST_BGCOLOR
CUSTOM.PORTLET_LIST_FGCOLOR
CUSTOM.PORTLET_LIST_DETAILED_BGCOLOR
CUSTOM.PORTLET_LIST_DETAILED_FGCOLOR
CUSTOM.PORTLET_LIST_BORDERCOLOR_TOP
CUSTOM.PORTLET_LIST_BORDERCOLOR_BOTTOM
CUSTOM.POPUPDRAGGERBAR_BGCOLOR
CUSTOM.POPUPDRAGGERBAR_FGCOLOR
CUSTOM.PORTLET_CAPTION_INACTIVE_FGCOLOR
CUSTOM.PORTLET_CAPTION_MOUSEOVER_FGCOLOR
CUSTOM.PORTLET_LIST_SELECTED_BGCOLOR
CUSTOM.PORTLET_LIST_SELECTED_FGCOLOR
CUSTOM.MESSAGEBOX_BGCOLOR
CUSTOM.MESSAGEBOX_FGCOLOR
Delete all call entries of 12341243
OK
Cancel

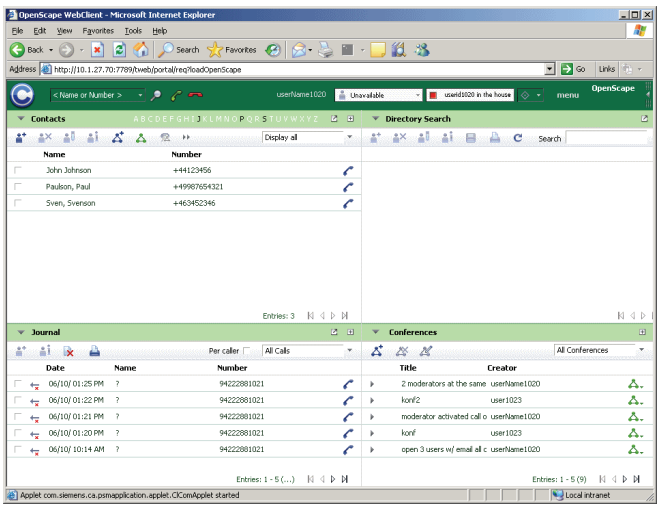
2.3.3 Samples

The following figures show examples of customized OpenScape UC Application Web Client 1.0 GUIs.






OpenScape WebClient - Microsoft Internet Explorer
Contacts
Directory Search
Journal
Conferences
John Johnson
Paulson, Paul
Sven, Svenson
Applet com.siemens.ca.psmapplication.applet.ClComApplet started


OpenScape WebClient - Microsoft Internet Explorer
Contacts
Directory Search
Journal
Conferences
John Johnson
Paulson, Paul
Sven, Svenson
Applet com.siemens.ca.psmapplication.applet.ClComApplet started

2.3.4 Login Page Properties

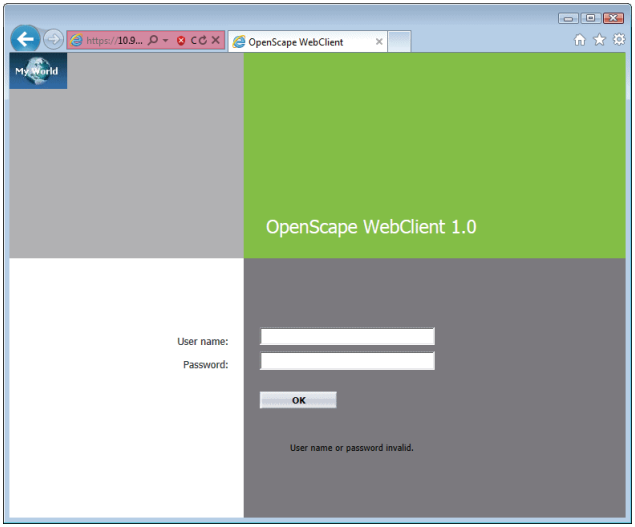
The following properties can be used to customize the login page of the OpenScape UC Application Web Client 1.0.

Item	Property	Description
1	CUSTOM.LOGINIMAGE_COMPANY	Path and file name of the picture to be shown in the upper left part of the login page

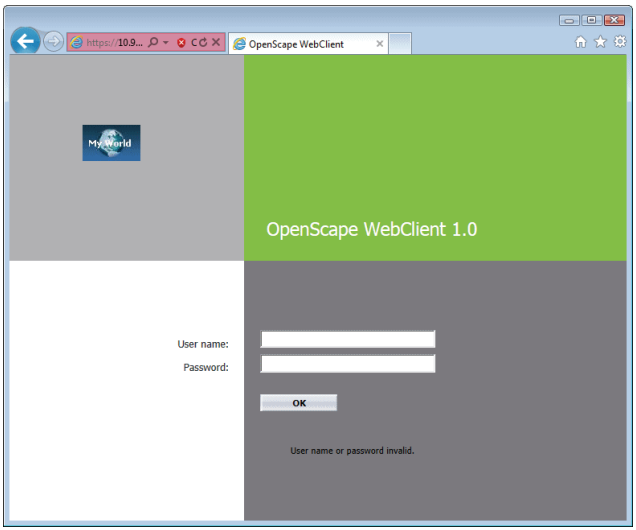
2	CUSTOM.LOGINIMAGE_POSITION	This property indicates the position of the the picture CUSTOM.LOGINIMAGE_COMPANY within the upper left part of the login page. It always consists of two values being separated by a blank. The first value indicates the horizontal position, the second value indicates the vertical position. The values `top`, `left`, `right`, `bottom` and absolute pixel values are allowed. `px` is the unit of measurement for absolute values. The upper left corner is the point of origin of the coordinate system.
		Examples:
		`LOGINIMAGE_POSITION = right bottom`
		`LOGINIMAGE_POSITION = left top`
		`LOGINIMAGE_POSITION = center center`
		`LOGINIMAGE_POSITION = 40px center`
		`LOGINIMAGE_POSITION = 40px 40px`
		If you use absolute values, make sure that there is no space between the number and the unit of measurement.
		If you use the value pair `right bottom`, there is a 15 pixel wide margin right of the picture and a 18 pixel high margin below the picture.

Table 3 Login Page Properties

Sample login page with the `CUSTOM.LOGINIMAGE_POSITION` property set to `left top`:


https://10.9...
OpenScape WebClient
OpenScape WebClient 1.0
User name:
Password:
OK
User name or password invalid.

Sample login page with the `CUSTOM.LOGINIMAGE_POSITION` property set to `center center`:



2.4 Proceedings for Customization

1. Edting `applicationContext.xml`

   f) Open the

   `<install>/WebSpace/Portal/webapps/tweb/WEB-INF/applicationContext.xml`

   file in an editor.

   g) Look for the following line indicating the `bundleConfiguration` section:

   ```
   <bean id="bundleConfiguration"
   class="com.siemens.ca.context.support.CaResourceBundleConfiguration">
   ```

   h) Look for the `cacheSeconds` property in this section.

   ```
   <property name="cacheSeconds" value="3600"/>
   ```

   i) Set the `cacheSeconds` property to a value of `5`.

   All customStyles files are cached by the page generator FreeMarker, i. e. changes take only effect, if the web engine is restarted or if the property `cacheSeconds` in the `bundleConfiguration` section of the

   `WEB-INF/applicationContext.xml`

   file is set to a value less than `3600` seconds.

2. Restart the web engine on the front-end computer by executing the following commands.

   ```
   /etc/init.d/symphoniad stop WebClient_FE
   /etc/init.d/symphoniad start WebClient_FE
   ```

NOTE: If you use Small Deployment, execute the following commands instead on the application computer:

```
/etc/init.d/symphoniad stop WebClient
/etc/init.d/symphoniad start WebClient
```

NOTE: Remember that Linux commands always are case-sensitive.

NOTE: If you want to check whether the OpenScape front-end services (WebClient) is running, execute the following command.

```
/etc/init.d/symphoniad status
```

The output is for example as follows:

```
TomcatServletContainer running
FrameworkContainer running
NotificationManager running
ActiveMQ running
UDPForwarder running
WebClient running
```

Step 3 on page 24 to step 5 on page 24 describe the language independent customization, step 6 on page 25 describes the language dependent customization.

3. Open the `customStyles.properties` file in the

   `<install>/WebSpace/Portal/webapps/MyHtml/WEB-INF/language/styles/`

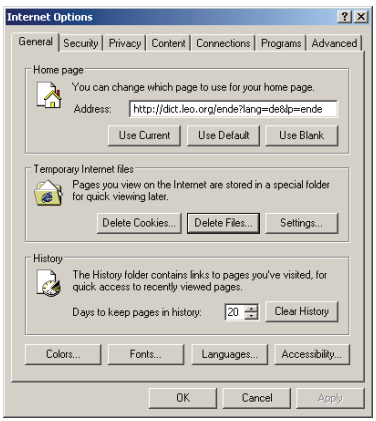
   directory (compare step 2 on page 11) in an editor, set the `CUSTOM.STYLES_ENABLED` property to `true` and remove the leading `#` character in this line. This causes all other properties to be considered and to influence the graphical user interface.

4. Customizing a property

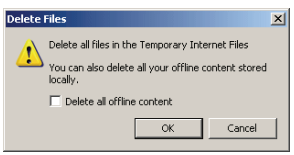
   a) Activate a property for customization by removing the leading `#` character in the appropriate line in the `customStyles.properties` file and change the value as wanted.

   b) Clear the browser cache.

   To do so in the Microsoft Internet Explorer, select **Tools** > **Internet Options...**.



   Click the **Delete Files...** button in the **Temporary Internet files** section.



   Click the **OK** button.

   Click the **OK** button.

   c) Wait for `cacheSeconds` seconds and reload the browser.

   d) Verify whether the modification suits your needs.

5. Repeat step 4 for all properties you want to customize.

6. If you want to create a language dependent customization, repeat step 3 to step 5 in an analog way for the `styles_<lang>.properties` files. See Section 2.2, “Customization Files”, on page 11 for more infor about these files.
7. Reset the `cacheSeconds` property in the

   `<install>/WebSpace/Portal/webapps/tweb/WEB-INF/applicationContext.xml`

   file to a value of `3600` seconds or higher (compare to step 1 on page 23).
8. If you deploy Large Deployment or Very Large Deployment, execute step 3 on page 12.
9. Restart the web engine on the front-end computer by executing the following commands.

```
/etc/init.d/symphoniad stop WebClient_FE
/etc/init.d/symphoniad start WebClient_FE
```

NOTE: If you use Small Deployment, execute the following commands instead on the application computer:

```
/etc/init.d/symphoniad stop WebClient
/etc/init.d/symphoniad start WebClient
```

NOTE: If you use Large Deployment and there are several front-end computers or if you use Very Large Deployment, this action has to be done on each front-end computer.

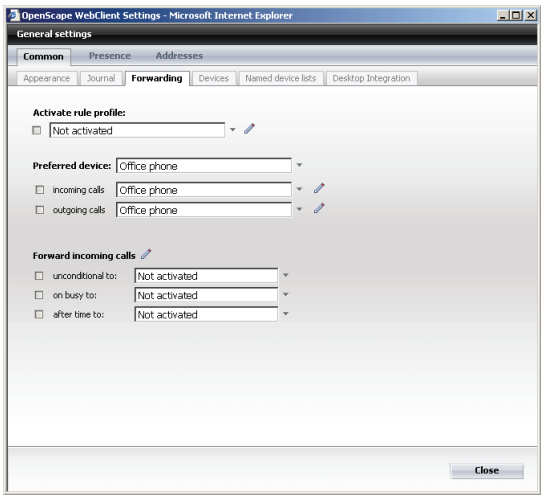
---

3 Integration

The OpenScape UC Application Web Client 1.0 SDK provides a quick and easy way to embed common UC functionality into a web application. However, for full UC integration capabilities use the programing guide *OpenScape UC Application SOAP SDK*.

IMPORTANT: Be sure that the preferred device of the user that is or will be logged in to OpenScape UC Application is set to the correct device. It is indicated and can be set in the main menu of the OpenScape UC Application Web Client 1.0 or at **Menu** > **General...** > **Common** > **Forwarding** > **Preferred device** in the OpenScape UC Application Web Client 1.0. Consider that the feature click to accept only works, if no preferred device is set (see Section 3.4, “Click to Accept”, on page 43).

With a click on the small triangle on the right margin of the combo box you can configure the device settings.



NOTE: The usage of port 7789 is described in the sections below. If you apply SSL, use the configured SSL port instead and use HTTPS instead of HTTP. The standard port for SSL is 8443.

NOTE: In all deployment scenarios being different from Large Deployment and Very Large Deployment, there is no front-end computer and the OpenScape front-end services reside on the application computer, i.e. `<FrontEndComputer>` in the following instructions has to be replaced by the name of the application computer.

See the *OpenScape UC Application Planning Guide* for more information about deployment scenarios.

3.1 Supported Web Browsers

The released browsers are documented in the *OpenScape UC Application Plannning Guide*.

- JavaScript must be activated.
- Pop-up blockers must be deactivated.
- Ad blockers must be deactivated.
- JRE 1.5 or 1.6 must be used.

3.2 Click to Dial

Click to dial means that you can initiate a phone call by clicking a link that is implemented on an arbitrary HTML page. This section describes how this link can be implemented. An arbitrary HTML page cannot know which OpenScape UC Application user account should be used to initiate the phone call. So you either already must be logged in to OpenScape UC Application in another web browser window before executing click to dial or you will be prompted in a web browser window to do so after clicking to dial. After clicking the link, the preferred device of the logged in user is called first. When his device is picked up, the call to the destination extension is established.

3.2.1 Click to Dial without using JavaScript

This section describes the simplest way to make a call for test purposes. There are two options that can be used to make a call:

3.2.1.1 Hidden Mode

In hidden mode, the modification of the GUI and the graphical interaction with the user should be reduced to an absolute minimum. In the ideal case, the GUI would not change at all while actions, e. g. make a call, release a call and accept a call are executed. The hidden mode mainly is deployed by a third party application being capable of starting an HTTP request in a browser control and supervising it.

An example for minimal GUI interaction is the case that a user is not logged in yet. In this case, the login page is returned by the front-end server (web server) to the application. The application must be able to make the browser control visible when showing the login page. It also must be able to make the browser control invisible again after successful login.

An example for an application using the hidden mode is an application using the Windows user account for automatic authentication (PKI) and sending the HTTP request when a phone number on a customer's web page is clicked. In this case, the phone would ring but the web browser's GUI would not change.

If you want to use the hidden mode without using JavaScript, the HTTP request

`http://<FrontEndComputer>:7789/tweb/portal/req?call=<number>`

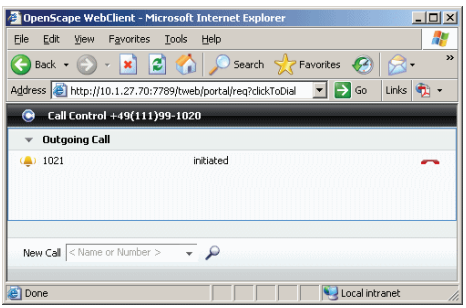
must be executed in order to dial a phone number. `<FrontEndComputer>` has to be replaced by the name of the front-end computer and `<number>` has to be replaced by the phone number to be called.

The hidden mode is intended for "unvisible" browser windows, e. g. an IFrame sized 1x1 in a HTML page or set to hidden by using CCS. Although this browser window is invisible, it is a correct browser window that can send HTTP requests. Since it is the intention that the web browser windows remain unchanged and no new web browser windows, e. g. a web browser window showing the login page, are opened while the phone call is processed, the web browser user already should be logged in, if possible.

If the web browser user is logged in, the front-end server's (web server) answer to the HTTP request that is stated above only contains a simple confirmation page that is not intended to be made visible to the web browser user. This answer only should be used as a confirmation whether the sent HTTP request has been sent correctly. The answer does not give any information about whether the phone call has been executed.

IMPORTANT: However, a developer using the hidden mode has to reckon, that the web browser user might not be logged in yet and authentication is needed. In this case, a login page is sent back. Since this login page is not visible, the developer has to provide an authentication mechanism, e. g. the developer can make the login page to be recognized as an answer and make this page visible. When the web browser user has logged in, the login page should be made invisible again.

If the call control window is returned, the visible mode (see Section 3.2.1.2, "Visible mode", on page 31) should be used instead of the hidden mode.



If you want to execute one of the following actions, use the visible mode (see Section 3.2.1.2, "Visible mode", on page 31) instead of the hidden mode:

- Check whether the phone call has been executed
- Recognize whether the called party is occupied
- Forward the phone call

The HTTP request

```
http://<FrontEndComputer>:7789/tweb/portal/req?call=<number>
```

delivers a task confirmation but no confirmation whether the call could be established. The call establishing may fail because the telephone number could not be reached. If you want to have a confirmation, use the command

```
http://<FrontEndComputer>:7789/tweb/portal/req?call=<number>&returntype=text
```

to make a call. The confirmation, e. g.

```
responseCode: 0
```

(mime-type: plain/text) is returned. The value `0` indicates success, any other value indicates a failure.

All HTTP requests stated above can be entered in the address field of the web browser for test purposes.

3.2.1.2 Visible mode

The visible mode is used when modifications of the GUI and interactions with the GUI user are wanted. This means that e. g. the call control information should be made visible. If you do not want to use JavaScript, use the following HTTP to make a call.

`http://<FrontEndComputer>:7789/tweb/portal/req?clickToDial=<number>`

`<FrontEndComputer>` has to be replaced by the name of the front-end computer and `<number>` has to be replaced by the phone number to be called.

NOTE: This request always opens a new web browser window with call control features for the OpenScape UC Application. This window does not check whether there is another window with an already opened OpenScape UC Application Web Client 1.0.

---

1. Enter the HTTP request into the address field of a web browser or implement it in a HTML file and click the link in a browser.

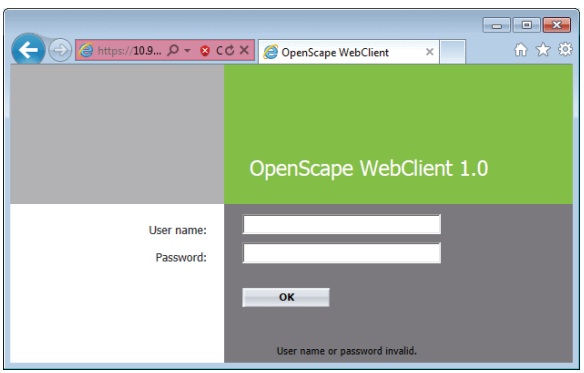
   Example:

   ```
   <html>
   <body>
   <a href="http://<FrontEndComputer>:7789/tweb/portal/
   req?clickToDial=<number>">Call <number> without JavaScript with visible call control
   information</a>
   </body>
   </html>
   ```

   Again, substitute `<FrontEndComputer>` and `<number>` by appropiate values.

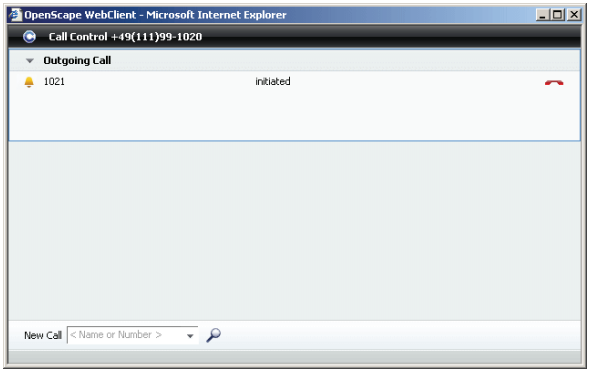
2. Depending on the situation you are in, the following happens.

   - The following login page is shown, if no OpenScape UC Application Web Client 1.0 login cookies are stored for the browser you are using. These cookies are not available, if you either have not logged in yet or the cookies have been deleted after a successful login.

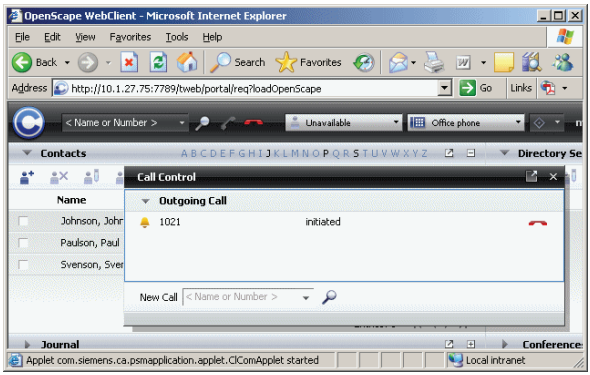
     

     Enter your user name and password. A new window with the screenshot in the next figure opens.

- If there are OpenScape UC Application Web Client 1.0 login cookies stored for the browser you are using but you are not logged in, the call control dialog opens in a new window.



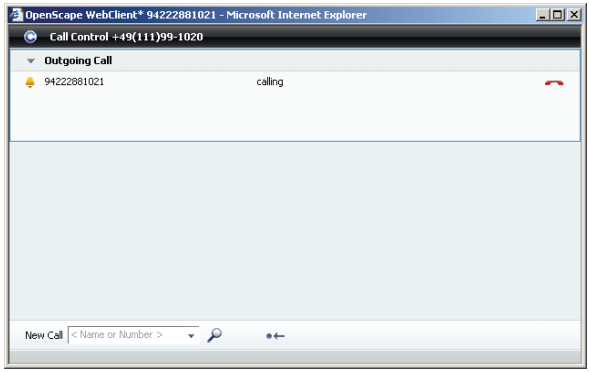
- If there are OpenScape UC Application Web Client 1.0 login cookies stored for the browser you are using and you are already logged in, the call control dialog is not opened in a new window opens but within the OpenScape UC Application Web Client 1.0 window.



  Both kinds of call control dialogs have the same functionality.

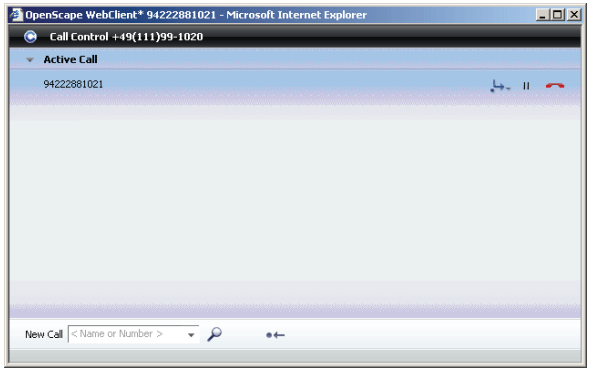
The preferred device of the logged in user rings.

3. Pick up the preferred device of the logged in user.



A call between the preferred device of the logged in user and the destination device (extension `<number>`) automatically is established.

4. Pick up the destination device.



The phone call is established.

3.2.2 Click to Dial using JavaScript

3.2.2.1 Hidden Mode

This section describes how to write an HTML page to establish a telephone call using JavaScript. This page works in hidden mode, i. e. it does not open any new window.

1. Implement the introduction of the HTML file.
```
<html>
<head>
<title>Hidden Call Sample</title>
```
2. Define the `doHiddenCall()` function handing over a string indicating the action you want to perform and the phone number you want to call to the `setRequestToHiddenIFrame()` function.
```
<script type="text/javascript">
function doHiddenCall( number )
{
   setRequestToHiddenIFrame( "/tweb/portal/req?call=" + encodeURIComponent(number) );
}
```
3. The `setRequestToHiddenIFrame()` function gets a part of the HTTP request that should be sent and assigns the complete HTTP request to the `src` attribute of the hidden IFrame that should send this HTTP request.
```
function setRequestToHiddenIFrame( request )
{
   var myIFrame = window.document.getElementById("myHiddenIFrame");
   myIFrame.src = "http://<FrontEndComputer>:7789" + request;
}
</script>
</head>
```
   Again, substitute `<FrontEndComputer>` by the appropiate value. If using SSL, replace `http` by `https` and replace port `7789` by port `8443`.
4. Start the body of the HTML page.
```
<body style="margin:20px";>
Samples using a hidden IFrame:
```
5. Define a button calling the `doHiddenCall()` function when it is clicked. This function defined above forwards the phone number to be called towards the hidden IFrame.
```
<p><button onClick="doHiddenCall('<number>')">Call phone <number></button></p>
```
   Again, substitute `<number>` by the appropiate value.
6. Define the hidden IFrame that should send the HTTP request.
```
<iframe style="display:none;" id="myHiddenIFrame" src=""></iframe>
```
7. Do not forget the closing tags for all HTML pages.
```
</body>
</html>
```

3.2.2.2 Visible Mode

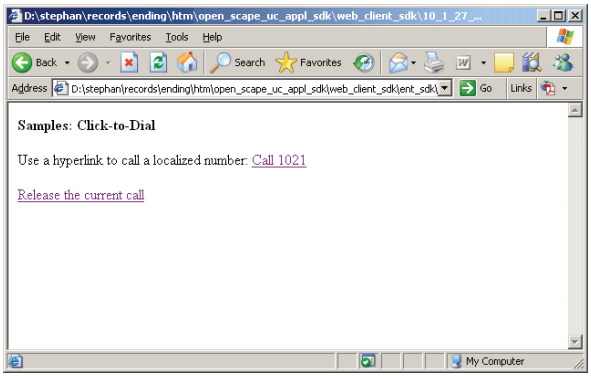
This section describes how to write an HTML page using JavaScript to establish a phone call. This page works in visible mode, i. e. it opens a new browser windows to show call control information.

Basic Proceeding

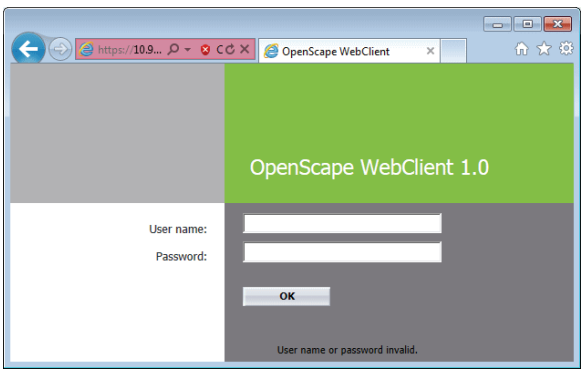
There are just a few steps to execute in order to integrate click-to-dial functionality in an arbitrary HTML page.

1. Implement an introduction of the HTML file.
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!-- Saved from url=(0087)http://10.1.27.75:7789/tweb/portal/req?getPage=/CaUIs/CmOS/
start/sampleClickToDial.html -->
<HTML>
<HEAD>
<META http-equiv=Content-Type content="text/html; charset=utf-8">
```
2. Make the script on the OpenScape UC Application server known to the web browser.
```
<script src="http://<FrontEndComputer>:7789/tweb/public/req?getClickToDialScript="
type="text/javascript"></script>
```
Substitute `<FrontEndComputer>` by the name or the IP address of the front-end computer . If using SSL, replace `http` by `https` and replace port `7789` by port `8443`.
3. Implement the `initOnLoad()` method to initialize the OpenScape UC Application web server for the getClickToDial script.
```
<SCRIPT src="" type=text/javascript>
function initOnLoad()
{
```
4. Check whether the click-to-dial script (getClickToDialScript) is loaded.
```
if( window.setCallServerUrl == null )
{
alert( "Click-to-Dial script could not be loaded from the " +
"web client installation running on the target host (localhost?)" );
return;
}
```
5. Initialize the getClickToDialScript script.
```
setCallServerUrl( "http://<FrontEndComputer>:7789" );
}
</SCRIPT>
</HEAD>
```
Substitute `<FrontEndComputer>` by the name or the IP address of the front-end computer . If using SSL, replace `http` by `https` and replace port `7789` by port `8443`.
6. Start implementing the body of the HTML page by assigning the function to be loaded when loading the HTML page and by writing a headline.
```
<BODY onload=initOnLoad()>
<B>Samples: Click-to-Dial</B>
```
7. The following hyperlink calls the indicated phone number when clicked on. Substitute `<number>` by the local destination extension to be called, e. g. `1021`.
```
<P>
Use a hyperlink to call a localized number:
<A href="javascript:makeCall('<number>')">Call <number></A>
</P>
```

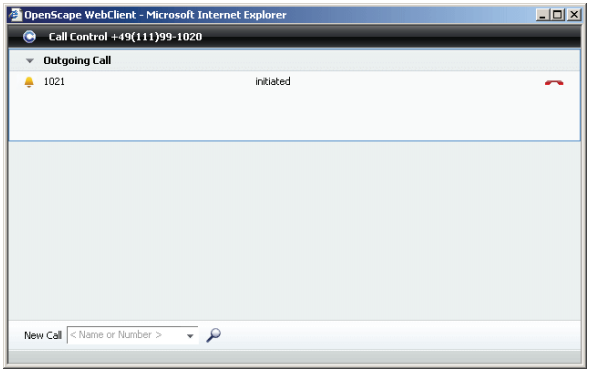
8. Do not forget the closing tags of every HTML page.
```
</BODY>
</HTML>
```
9. Save the the HTML file.
10. Be sure that JavaScript is enabled in the web browser.
11. Open the HTML file in a web browser.



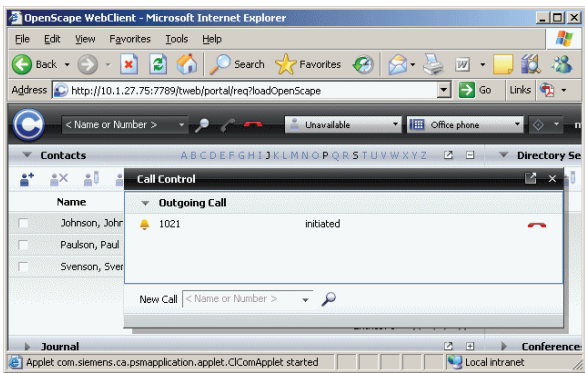
12. Click the first link in your HTML page. Depending on the situation you are in, the following happens.
    - The following login page is shown, if no OpenScape UC Application Web Client 1.0 login cookies are stored for the browser you are using. These cookies are not available, if you either have not logged in yet or the cookies have been deleted after a successful login.



      Enter your user name and password. A new window with the screenshot in the next figure opens.
    - If there are OpenScape UC Application Web Client 1.0 login cookies stored for the browser you are using but you are not logged in, the call control dialog opens in a new window.



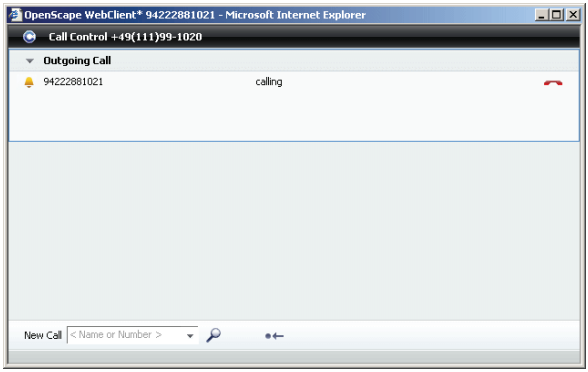
- If there are OpenScape UC Application Web Client 1.0 login cookies stored for the browser you are using and you are already logged in, the call control dialog is not opened in a new window opens but within the OpenScape Web Client 1.0 window.



Both kinds of call control dialogs have the same functionality.

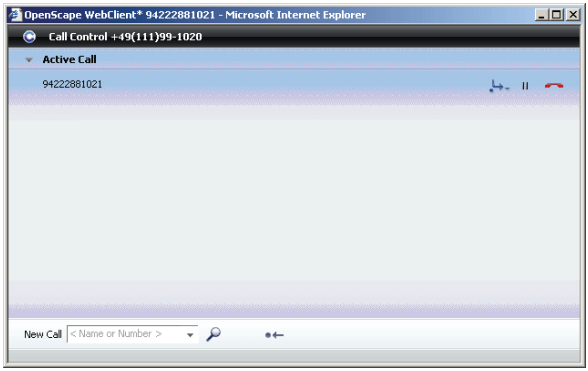
The preferred device of the logged in user rings.

13. Pick up the preferred device of the logged in user.



A call between the preferred device of the logged in user and the destination device (extension `<number>`) automatically is established.

14. Pick up the destination device.



The phone call is established.

Additional Features

1. If no single sign on is installed and you want to open a browser window only for login when a call is established in step 7 on page 35, extend the `initOnLoad()` function in step 3 on page 35 by calling the `setHiddenCallEnabled(true)` function.

   If `setHiddenCallEnabled(true)` has been called,

   - and a call control window already is opened, calling `makeCall()` is executed without opening a new call control window. The already opened call control window is moved to the foreground.
   - and the OpenScape UC Application web portal already is opened in a browser window, calling `makeCall()` is executed without opening a new call control window. The already opened window is moved to the foreground. The call control information is shown in the already opened window.
   - a call control window has not been opened yet and `makeCall()` is called, the call control window is opened with the correct size.

2. The `setHiddenCallEnabled()` function has to be called as well, if you want to make a hidden request to perform the phone call. If you want to do so, define the following function in addition to the `initOnLoad()` function.

   ```
   function configureHiddenMakeCallRequest( checkBox )
   {
       setHiddenCallEnabled( checkBox.checked );
   }
   ```

   Then implement the following checkbox.

   ```
   <P>
   Use "hidden" request to perform the call:
   <INPUT onclick=configureHiddenMakeCallRequest(this); type=checkbox
   name=useHiddenMakeCall></INPUT>
   </P>
   ```

3. Instead of using the hyperlink in step 7 on page 35 you can use a field where the phone number to be called can be entered.

   ```
   <P>
   <FORM name=inputForm onsubmit="makeCall( this.number.value ); return false;">
      1) Enter a number:
      <INPUT size=16 name=number></INPUT>
      <BUTTON type=submit>Call</BUTTON>
   </FORM>
   </P>
   ```

NOTE: Furthermore, each OpenScape UC Application Web Client 1.0 installation contains a sample HTML page which can be loaded at the following URL:

`http://<FrontEndComputer>:7789/tweb/portal/req?getPage=/CaUIs/CmOS/start/sampleClickToDial.html`

This file is stored in the following directory:

`<install>/WebSpace/Portal/webapps/tweb/CaUIs/CmOS/start`

3.3 Click to Release

3.3.1 Click to Release without using JavaScript

3.3.1.1 Hidden Mode

The hidden mode does not use any GUI modification and interaction. See Section 3.2.1.1, “Hidden Mode”, on page 29 for more details.

The HTTP request

```
http://<FrontEndComputer>:7789/tweb/portal/req?hangup
```

releases a call. It should only be executed if no visible browser window should be opened.

NOTE: Consider that this request has the same impact as hanging up. Thus, if you make a consultation call, this request not only releases the consultation call but also the held call.

This HTTP request delivers a task confirmation but no confirmation whether the call could be released. If you want to have a confirmation, use the command

```
http://<FrontEndComputer>:7789/tweb/portal/req?hangup&returntype=text
```

to release a call. The confirmation, e. g.

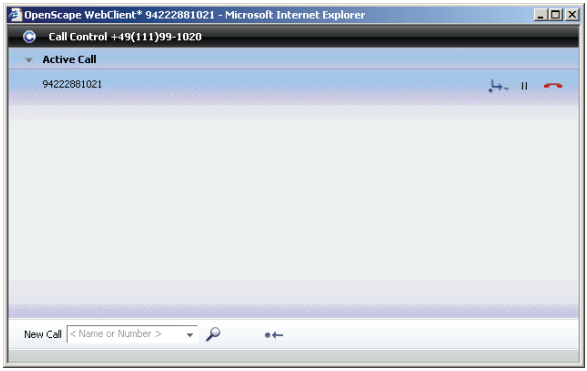
```
responseCode: 0
```

(mime-type: plain/text) is returned. The value `0` indicates success, any other value indicates a failure.

NOTE: All HTTP requests stated above can be entered in the address field of the web browser for test purposes.

3.3.1.2 Visible Mode

If you want to release a call, hang up or click the red telephone icon in the call control window.



3.3.2 Click to Release using JavaScript

3.3.2.1 Hidden Mode

1. Extend the `<script type="text/javascript">` section described in Section 3.2.2.1, “Hidden Mode”, on page 34 by the `doHiddenDisconnect()` function.

```
function doHiddenDisconnect()
{
   setRequestToHiddenIFrame( "/tweb/portal/req?hangup" );
}
```

2. Extend the `<body>` section by the following button:

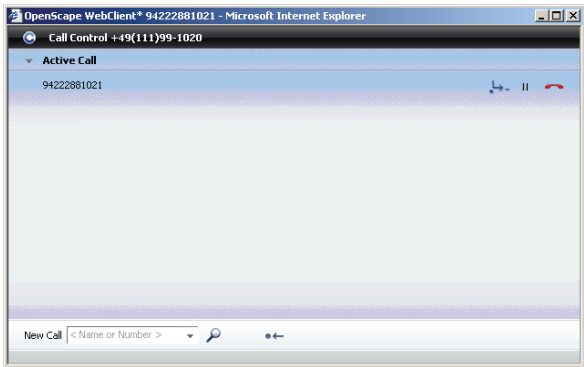
```
<p><button onClick="doHiddenDisconnect()">Release the current call</button></p>
```

3.3.2.2 Visible Mode

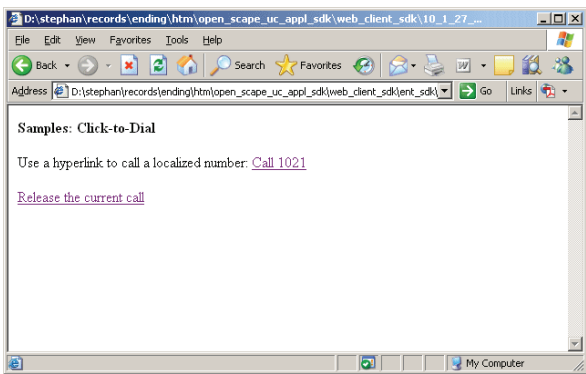
1. Extend the `<body>` section in Section 3.2.2.2, “Visible Mode”, on page 35 by the following hyperlink:

```
<P>
<A href="javascript:disconnectCall();">Release the current call</A>
</P>
```

2. To release a call,
   - hang up,
   - click the red phone icon or



   - click the second link in your HTML page.

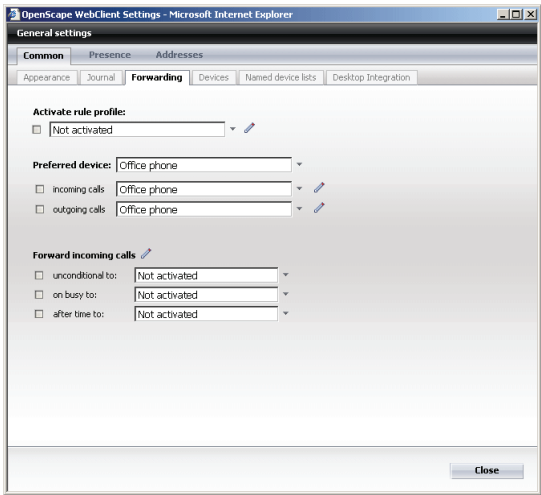


3.4 Click to Accept

Use the click to accept feature to accept an incoming phone call at an OpenScape UC Application user's phone and switches the phone's loudspeaker on. This is initiated e. g. by clicking on a link on an arbitrary HTML page extended by the code described below.

IMPORTANT: The click to accept feature only works if no preferred device is set for the user. When this user's preferred device is set, the execution of this HTTP request is ignored by the server and the loudspeaker is not switched on. The web browser user is not notified that the HTTP request has not succeeded. No error message is generated.
The same happens if this request cannot be assigned correctly to a phone call, e. g. if there are two incoming phone calls at the same time.
The preferred device is indicated and can be set in the main menu of the OpenScape UC Application Web Client 1.0 or at **Menu** > **General...** > **Common** > **Forwarding** > **Preferred device** in the OpenScape UC Application Web Client 1.0.

With a click on the small triangle on the right margin of the combo box you can configure the device settings.



3.4.1 Click to Accept without using JavaScript

3.4.1.1 Hidden Mode

The hidden mode does not use any GUI modification and interaction. See Section 3.2.1.1, “Hidden Mode”, on page 29 for more details.

When an OpenScape UC Application user is logged in to the OpenScape UC Application Web Client 1.0, the HTTP request

`http://<FrontEndComputer>:7789/tweb/portal/req?answer`

accepts a call to the original phone of this user and the loudspeaker of this phone is switched on.

This HTTP request should only be executed if no visible browser window should be opened.

This HTTP request delivers a task confirmation but no confirmation whether the call could be accepted. If you want to have a confirmation, use the command

`http://<FrontEndComputer>:7789/tweb/portal/req?answer&returntype=text`

to accept a call. The confirmation, e. g.

`responseCode: 0`

(mime-type: plain/text) is returned. The value `0` indicates success, any other value indicates a failure.

NOTE: All HTTP requests stated above can be entered in the address field of the web browser for test purposes.

Instead of accepting the call, you can block it using the

`http://<FrontEndComputer>:7789/tweb/portal/req?`hangup

request, if the PBX supports this. See Section 3.3.1.1, “Hidden Mode”, on page 40 for details about this request.

3.4.2 Click to Accept using JavaScript

3.4.2.1 Hidden Mode

This section describes how to write an HTML page to accept a telephone call using JavaScript. This page works in hidden mode, i. e. it does not open any new window.

1. Extend the `<script type="text/javascript">` section described in Section 3.2.2.1, “Hidden Mode”, on page 34 by the `doHiddenAccept()` function.
```
function doHiddenAccept()
{
   setRequestToHiddenIFrame( "/tweb/portal/req?answer" );
}
```
2. Extend the `<body>` section by the following button:
```
<p><button onClick="doHiddenAccept()">Accept an incoming call</button></p>
```

Integration

Click to Accept