NIXXIS CONTACT SUITE

Nixxis Web Client

Agent Module

Version 1.0

Date	***Description***	***User***	***Version***
11/01/2024	Document Creation	Nixxis	1.0

Introduction

This document outlines the deployment process and the limitations of the web version of Nixxis Client. While the web version provides accessibility and flexibility, certain functionalities and constraints should be considered for a comprehensive understanding.

1. Supported browsers:

Chrome, Edge and Safari.

Note for Safari:

It is important to note that the NCS web client does not support older versions of Safari, such as Safari - 13.1.2 on MacOS Catalina. This is due to the presence of Ui glitches related to the user interface when running web client on Safari with older versions.

The reasons:

- From an end-user perspective, Safari 13 has been considered outdated and deprecated since 2020.
- From a technical perspective, Safari 13 may not support the CSS used for web Clients, leading to UI-related bugs. However, Safari 16 has updated some CSS features, preventing these UI bugs from occurring

It is recommended to use newer versions of Safari, such as Safari - 16.6

2. User Interface

2.1 Responsiveness and Performance

- **Desktop Application:** Achieves native-level responsiveness and optimal performance.
- **Web Application:** May experience slight delays and variations in performance, depending on the network connectivity.

2.2 Interface language

Language set in administration for the agent is not taken into consideration.

By default, the client uses the browser language. The user can either select English or French on the login screen. The selection is saved until cleared manually by the user or browser settings.

2.3 Access management

A user defined in Nixxis Administration can log in the web client even if he does not have an agent role.

3. Media handling

The web version can only handle inbound and outbound voice calls at this time.

4. IP Phone

The web version does not have an integrated softphone. It must be used with a softphone or a physical IP phone.

5. CORS Issues & Same-Origin Policy

Cross-Origin Resource Sharing (CORS) is a security feature implemented by web browsers to protect users from potential security vulnerabilities that can arise when web pages make requests to a different domain than the one that served the web page. The core principle is to prevent unauthorized cross-origin requests, ensuring that only trusted domains can interact with each other.
The Same-Origin Policy is a security measure enforced by web browsers that restricts web pages from making requests to a different domain than the one that served the web page.

Explanation: Due to security concerns outlined by the Same-Origin Policy and CORS, web browsers restrict the embedding of content from one domain into an iframe or HTML5 data object if the target domain does not explicitly allow such embedding.
This results in the script (web page) not being displayed inside the Nixxis Web Client.
When the thin client is deployed, a tool is available for testing webpages which can be displayed inside the thin client.

The tool can be accessed at this address: http(s)://web-client-url/corstester.html

Recommendation: If integration with external resources is necessary, ensure that the target domain explicitly supports CORS by including the appropriate headers in its HTTP responses. This can be achieved through server-side configurations (on the external resource side) to enable controlled cross-origin access.

6. Click to call limitations

Click to call functions are not available as on the desktop client as the “protocol handlers” are defined by the web browser.

Installation of the web client

1. Prerequisites

A running instance of NCS server.

2. Deployment

1. Download the latest version from: https://github.com/NixxisIntegration/NcsThinClient/blob/main/NcsWebClient.1.0.0.zip
2. Put the content in \CrAppServer\staticfiles\ (note that the content should be copied here directly, not in a subfolder)
3. Open the file \CrAppServer\http.config and make sure the following keys are present with the values below in the “agent” application:

```
<add key="filesPath" value="StaticFiles" />
<add key="filesSettings" value="specificExt='.txt.html.htm.js.vbs.css';cacheControl='private';cacheDuration='500'" />
<add key="defaultFile" value="CrLoginScreen.htm" />
<add key="agtFile" value="CrAgent.htm"/>
<add key="loginFile" value="CrLoginScreen.htm" />
```

4. The web client should now be accessible at http://appserver-IP-or-FQDN:8088/agent/

3. Setup of HTTPS

Https can be setup either using IIS or Nginx.

3.1 Prerequisites

Before starting the setup, make sure you have the following:

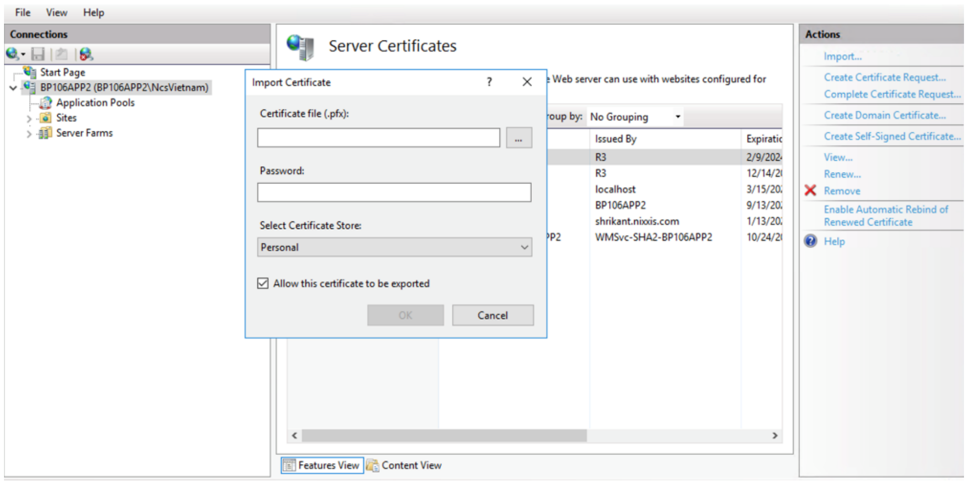
- SSL certificate files:
  - Certificate.pem (SSL certificate)
  - Private key.key (Private key)
- Nginx installed (for Nginx setup)
- IIS installed (for IIS setup)
  - Install the URL Rewrite module for IIS from https://www.iis.net/downloads/microsoft/url-rewrite.

3.2 Setting up IIS with HTTPS

- Convert .pem and private key files to a .pfx file

```
openssl pkcs12 -export -out YourDomain.pfx -inkey YourDomain.key -in YourDomain.pemopenssl
pkcs12 -export -out YourDomain.pfx -inkey YourDomain.key -in
YourDomain.pem
```

- Open IIS Manager
- Select your server in the Connections panel and double-click on "Server Certificates" in the center panel.
- Click "Import" in the Actions panel on the right.
- Browse to and select the **YourDomain.pfx** file.


File
View
Help
Connections
Start Page
BP106APP2 (BP106APP2\NcsVietnam)
Application Pools
Sites
Server Farms
Server Certificates
Web server can use with websites configured for
Features View
Content View
Actions
Import...
Create Certificate Request...
Complete Certificate Request...
Create Domain Certificate...
Create Self-Signed Certificate...
View...
Renew...
Remove
Enable Automatic Rebind of Renewed Certificate
Help

Import Certificate

Certificate file (.pfx):

Password:

Select Certificate Store: Personal

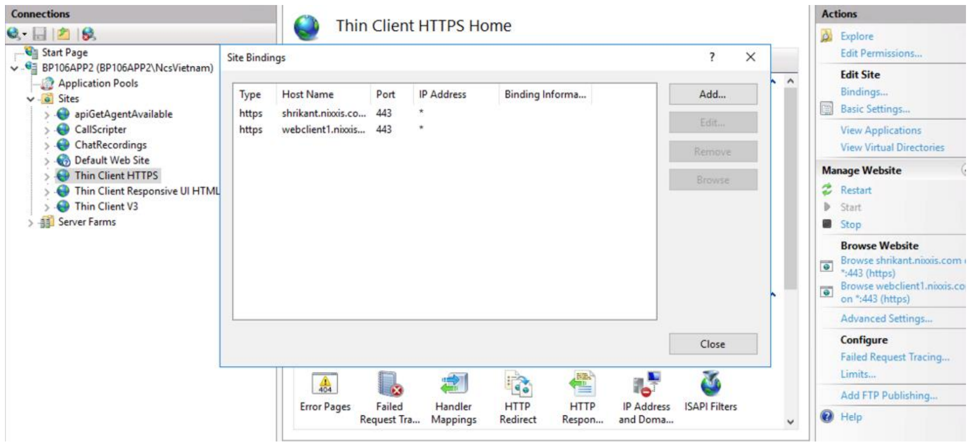
☑ Allow this certificate to be exported

OK | Cancel

roup by: No Grouping

	Issued By	Expiratio
	R3	2/9/202
	R3	12/14/2
	localhost	3/15/20
	BP106APP2	9/13/20
	shrikant.nixxis.com	1/13/20
P2	WMSvc-SHA2-BP106APP2	10/24/2

- In the Actions panel on the right, click on "Add Website."
    - Fill in the required information:
    - Host name: Enter your domain name (e.g., demo.nixxis.com).
    - Click OK to create the new site.
- In the Actions panel on the right, click on "Bindings."
- Click "Add" to add a new binding.
- Select Type: HTTPS, IP Address: All Unassigned, Port: 443, and select the SSL certificate you imported earlier.



Site Bindings

Type	Host Name	Port	IP Address	Binding Informa...
https	shrikant.nixxis.co...	443	*	
https	webclient1.nixxis...	443	*	

Add... | Edit... | Remove | Browse | Close

- Click OK to save the binding.

Go to root folder of website (In this example C:\NixxisV3\CrAppServer\staticfiles) and create a file **web.config.** With the content:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <system.webServer>
        <rewrite>
            <rules>
                <clear />
                <rule name="Redirect home page" enabled="true"
stopProcessing="true">
                    <match url="^$" />
                    <conditions logicalGrouping="MatchAll"
trackAllCaptures="false" />
                    <action type="Redirect"
url="https://{HTTP_HOST}/agent/CrAgent.htm" />
                </rule>
                <rule name="Agent Request" stopProcessing="true">
                    <match url="^agent/(.*)" />
                    <conditions logicalGrouping="MatchAll"
trackAllCaptures="false" />
                    <serverVariables>
                        <set name="X-Forwarded-Proto" value="https" />
                        <set name="Host" value="{HTTP_HOST}" />
                    </serverVariables>
                    <action type="Rewrite"
url="http://10.x.x.x:8088/agent/{R:1}" logRewrittenUrl="true" />
                </rule>
                <rule name="Reject if no Query String" enabled="true"
stopProcessing="true">
                    <match url="(.*)" />
                    <conditions logicalGrouping="MatchAll"
trackAllCaptures="false">
                        <add input="{QUERY_STRING}" pattern="^$" />
                    </conditions>
                    <action type="CustomResponse" statusCode="403"
subStatusCode="403" statusReason="Forbidden"
statusDescription="Forbidden to access" />
                </rule>
                <rule name="All requests" enabled="true"
stopProcessing="true">
                    <match url="(.*)" />
                    <conditions logicalGrouping="MatchAll"
trackAllCaptures="false">
                        <add input="{QUERY_STRING}" pattern=".+" />
                    </conditions>
                    <serverVariables>
                        <set name="Host" value="{HTTP_HOST}" />
                        <set name="X-Forwarded-Proto" value="https" />
                    </serverVariables>
                    <action type="Rewrite"
url="http://10.x.x.x:8088/{R:1}" appendQueryString="true"
logRewrittenUrl="true" />
                </rule>
            </rules>
        </rewrite>
        <directoryBrowse enabled="true" />
        <defaultDocument>
            <files>
                <add value="crAgent.htm" />
            </files>
        </defaultDocument>
    </system.webServer>
</configuration>
```

***Replace http://10.x.x.x:8088 with the IP of the CrAppServer IP + port.*

This IIS URL Rewrite configuration is designed to handle various URL patterns and redirect or rewrite them based on specific conditions. Below is the break down of each rule and its purpose:

Redirect home page:

- Matches requests to the root URL (^$).
- Redirects to https://{HTTP_HOST}/agent/CrAgent.htm.

Agent Request:

- Matches requests starting with /agent/.
- Sets X-Forwarded-Proto to "https" and Host to {HTTP_HOST}.
- Rewrites the URL to http://10.x.x.x:8088/agent/{R:1}.

Reject if no Query String:

- Matches all requests.
- If there is NO QUERY STRING, it responds with a custom 403 Forbidden status.

All requests:

- Matches all requests (.*).
- Checks if there is any query string ({QUERY_STRING} pattern=".+").
- Sets Host to {HTTP_HOST} and X-Forwarded-Proto to "https".
- Rewrites the URL to http://10.x.x.x:8088/{R:1} and appends the original query string.

Notes:

The configuration uses server variables (X-Forwarded-Proto and Host) to simulate the presence of HTTPS (X-Forwarded-Proto) and set the correct host for the backend server (Host).

It logs the rewritten URLs for debugging purposes (logRewrittenUrl="true").

Directory browsing is enabled (<directoryBrowse enabled="true" />).

Restart the IIS server to apply the changes.

3.3 Setting up Nginx with HTTPS

Either Nginx or IIS should be used for HTTPS, not both concurrently. Running both simultaneously may lead to conflicts between them.

- Copy the SSL certificate and private key to a secure location on the Nginx server.
- Update the Nginx configuration file (usually found at C:/nginx/nginx.conf). Add or replace the default web server as the configuration below:

```
server {
    access_log logs/demo_access.log;
    error_log logs/demo_error.log;
    listen       443 ssl;
    ## Nixxis URL
    set $rpn_nixxis_url                 demo.nixxis.com;
    ## Local IP of AppServer
    set $rpn_nixxis_ip                  10.x.x.x;
    set $rpn_nixxis_port                8088;
    set $rpn_nixxis_scheme              http;
    set $rpn_nixxis_service_length  1398;

    ## FQDN
    set $public_nixxis_url              demo.nixxis.com;
    ## Public IP of AppServer
    set $public_nixxis_ip               151.x.x.x;
    set $public_nixxis_port             443;
    set $public_nixxis_scheme           https;


    ssl_protocols TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref:
POODLE
    ssl_prefer_server_ciphers on;
    ssl_ciphers 'TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:ECDHE-
RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384';
```

```
    keepalive_timeout    60;
    ssl_session_cache    shared:SSL:10m;
    client_body_timeout 10s;
    ssl_session_timeout  10m;
    server_tokens on;
    chunked_transfer_encoding       off;
    server_name  $public_nixxis_url;
    set $rpn_nixxis_backend
$rpn_nixxis_scheme://$rpn_nixxis_url:$rpn_nixxis_port;
    ## If certificate in "c:\nginx\conf\"
    ssl_certificate      fullchain1.pem;
    ssl_certificate_key  privkey1.pem;



    location / {

        if ($query_string = ""){
                return 403;

        }

        proxy_http_version 1.1;
        proxy_next_upstream error timeout invalid_header http_500
http_502 http_503 http_504;


        ### Set headers ####
        proxy_set_header        Accept-Encoding   "gzip, deflate";
        proxy_set_header        Host $rpn_nixxis_url;
        proxy_set_header        Connection "";
        proxy_set_header        X-Forwarded-For
$proxy_add_x_forwarded_for;
        # Content-Length has to customized function of hostname
        # Require by service url
        if ($query_string ~ service) {
                add_header 'Content-Length'
$rpn_nixxis_service_length;
        }

        sub_filter_types text/xml;
        # Replace URL for service list
        sub_filter $rpn_nixxis_scheme://$rpn_nixxis_url
$public_nixxis_scheme://$public_nixxis_url;
        #sub_filter
$rpn_nixxis_scheme://$rpn_nixxis_ip  $public_nixxis_scheme://$public_n
ixxis_url;

        sub_filter_once off;
        proxy_pass
$rpn_nixxis_scheme://$rpn_nixxis_ip:$rpn_nixxis_port;
    }


    location ~ ^/agent/(.*) {
        proxy_http_version 1.1;
        proxy_hide_header       Accept-Encoding;
        proxy_set_header        Host $rpn_nixxis_url;
        proxy_set_header        X-Forwarded-For
$proxy_add_x_forwarded_for;
        proxy_set_header        X-Forwarded-Proto https;
        proxy_hide_header       Content-Encoding;
        proxy_no_cache 1;
        proxy_pass
$rpn_nixxis_scheme://$rpn_nixxis_ip:$rpn_nixxis_port/agent/$1$is_args$
query_string;
    }


location ~ ^/internal-server {
    proxy_http_version 1.1;
    proxy_hide_header       Accept-Encoding;
    proxy_set_header        Host $http_host;
    proxy_set_header        X-Forwarded-For
$proxy_add_x_forwarded_for;
    proxy_set_header        X-Forwarded-Proto https;
    proxy_hide_header       Content-Encoding;
    proxy_no_cache 1;
     proxy_pass $rpn_nixxis_scheme://$rpn_nixxis_ip:$nodejs_port
/get$is_args$query_string;
}

}
```

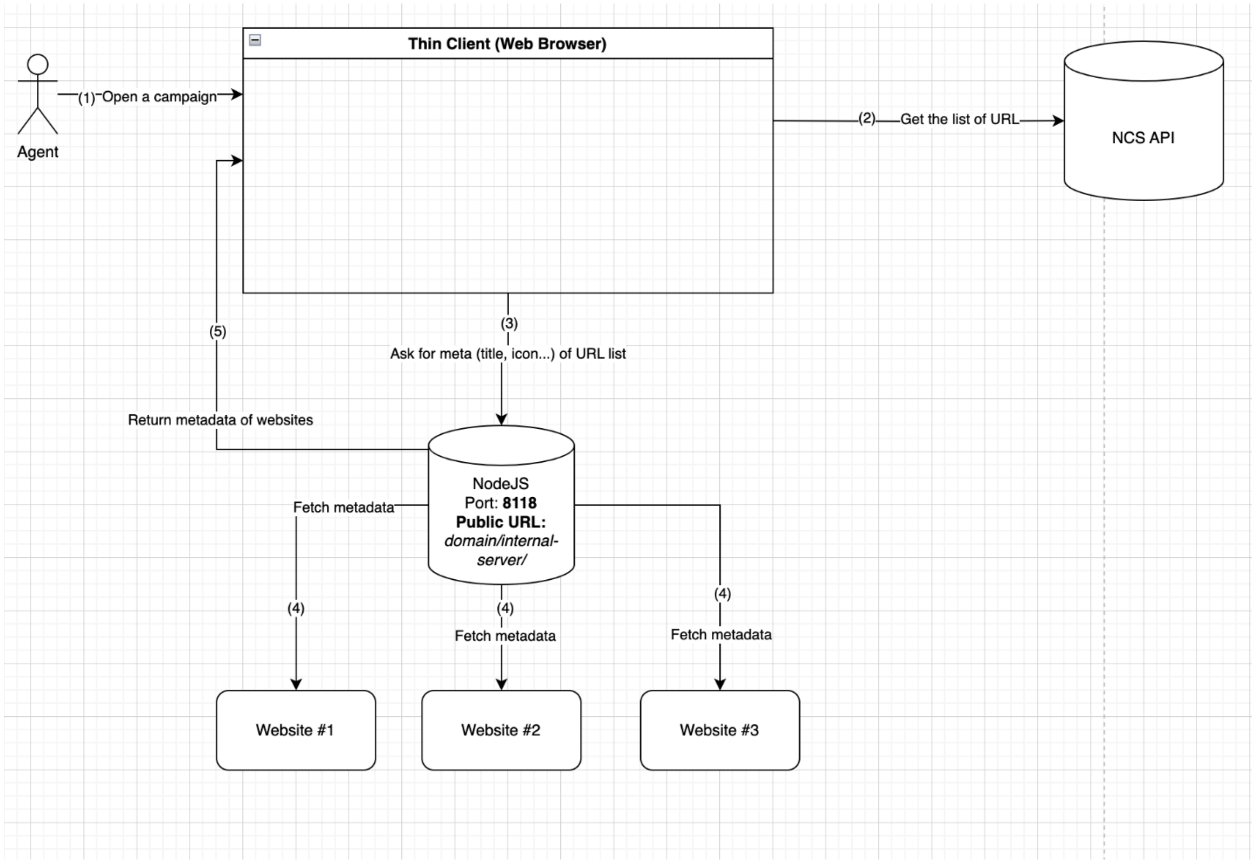
****Update the variables based on your environment:*

- `$rpn_nixxis_url, $public_nixxis_url : Public FQDN.`
- `$rpn_nixxis_ip: Local IP of server running CrAppServer.`
- `$rpn_nixxis_port  : CrAppServer port.`
- `$public_nixxis_ip : Public IP.`
- `$public_nixxis_port: Public port(443)`
- $ssl_certificate,ssl_certificate_key: `Path to certificate and private key.`
- `$nodejs_port: Port on which Nodejs server is listening`

Start **nginx.exe**

4. Setting up NodeJS

A NodeJS server is used only to fetch metadata for script URLS.



4.1 Install Node.js

Install Node.js from the official website : [Node.js]

4.2 Install PM2:

Open terminal or command prompt and run the following command to install PM2 globally:

```
npm install -g pm2
```

This command installs PM2 globally, allowing it to be used it from any directory in the system.

4.3 Deploy NodeJs application

Create a folder named “nixxis_wpx” inside the Nixxis installation folder (Drive:\Nixxis\)

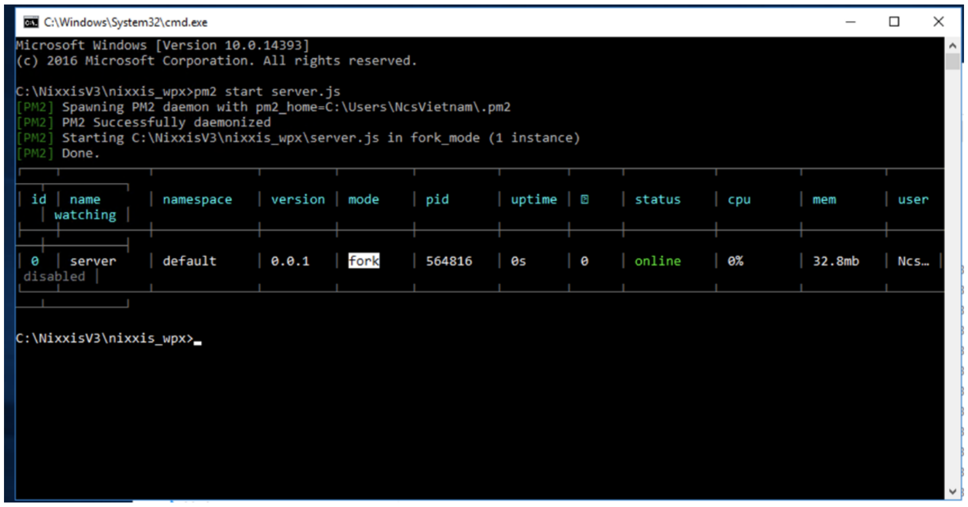
Download package from:
https://github.com/NixxisIntegration/NcsThinClient/blob/main/nixxis_wpx.1.0.0.zip

Unblock the zip archive and extract its content to \Nixxis\nixxis_wpx

4.4 Start Application with PM2

Navigate to source application's directory using the command prompt, and start Nixxis-wpx application using PM2:

```
pm2 start server.js
```



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\NixxisV3\nixxis_wpx>pm2 start server.js
[PM2] Spawning PM2 daemon with pm2_home=C:\Users\NcsVietnam\.pm2
[PM2] PM2 Successfully daemonized
[PM2] Starting C:\NixxisV3\nixxis_wpx\server.js in fork_mode (1 instance)
[PM2] Done.
| id | name      | namespace | version | mode | pid    | uptime | ↺ | status | cpu | mem    | user |
|    | watching  |
| 0  | server    | default   | 0.0.1   | fork | 564816 | 0s     | 0 | online | 0%  | 32.8mb | Ncs… |
 disabled |

C:\NixxisV3\nixxis_wpx>_
```

The application will run on port **8118**.

Note: https://github.com/NixxisIntegration/NcsThinClient/

5 Testing

Access the web client through HTTPS in one of the supported browsers: https://{domain}/agent/

6 Sample configuration files

Sample configuration files for web.config and nginx.conf are available here:

https://github.com/NixxisIntegration/NcsThinClient/tree/main/sampleconfig

7 eScript integration

Adaptations have been made to Nixxis_eScript.js for it to work with the web client. It is available here: https://github.com/NixxisIntegration/NcsThinClient/blob/main/eScript/Nixxis_eScript.js

Note that this JS works only with the web client. A new version compatible with the web and desktop client will be soon be release.