Converting an EGX300 Custom Page to a Com'X 510 Custom Page

This document describes the steps necessary to convert an EGX300 custom web page (HTML file) to a custom application on the Com'X 510 (HTML, JavaScript, and CSS files).

In this document

Additional information

- *Creating Com'X 510 Custom Web Pages*, 7EN72-0199
- *Com'X 510 User Manual*, DOCA0098EN
- *PowerLogic™ Ethernet Gateway EGX300 Reference Guide*, 63230-319-230

Schneider Electric
35 Rue Joseph Monier
92500 Rueil Malmaison – France
www.schneider-electric.com

Safety information

Important information

Read these instructions carefully and look at the equipment to become familiar with the device before trying to install, operate, service or maintain it. The following special messages may appear throughout this manual or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER

DANGER indicates a hazardous situation which, if not avoided, **will result in** death or serious injury.

⚠ WARNING

WARNING indicates a hazardous situation which, if not avoided, **could result in** death or serious injury.

⚠ CAUTION

CAUTION indicates a hazardous situation which, if not avoided, **could result in** minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

Please note

Electrical equipment should be installed, operated, serviced and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction, installation, and operation of electrical equipment and has received safety training to recognize and avoid the hazards involved.

Safety Precautions

⚠ WARNING

INACCURATE DATA RESULTS

- Do not incorrectly configure the software, as this can lead to inaccurate reports and/or data results.
- Do not base your maintenance or service actions solely on messages and information displayed by the software.
- Do not rely solely on data displayed in the software to determine if the device is functioning correctly or meeting all applicable standards and requirements.
- Do not use data displayed in the software as a substitute for proper workplace practices or equipment maintenance.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

NOTICE

IMPAIRED COM'X PERFORMANCE

Do not add more than 10 MB of files to Custom Page Management.

Failure to follow these instructions can impair Com'X performance.

Introduction

EGX300 custom page content is generally in one html file. You must separate this content into the following files for use on the Com'X 510:

- An HTML snippet encapsulated in a container for page layout
- One or more JavaScript files that define the page's functionality
- A cascading stylesheet

In addition to these files, you will need an initialization file. The contents of the **init.js** file are described in *Creating Com'X 510 Custom Web Pages*, reference 7EN72-0199. A sample **init.js** file that corresponds to the example below is included in this document.

Example

Below are excerpts from the sample page provided in the EGX300 custom web page reference guide (63230-319-230). The example page displays CM3000 frequency and current values in a table.

Relevant portions of the EGX300 sample HTML file are annotated below. Strikethrough text has been deleted and blue text has been added/edited. Use this example as a guideline for converting your existing EGX300 pages.

A few things to note before you begin:

- Server-parsed web pages (EGX300 .shtml files) are no longer supported.
- The .html file for a Com'X custom page is a div container only. So, you must remove the following: <!DOCTYPE> declaration and the <html>, <head>, <title>, <body>, and <form> tags from your existing file.
- We recommend also removing style elements and creating a .css file for styling.
- window.onload does not fire when the custom page is loaded, since the Com'X page is not a complete web page. Instead, _application.show() gets called when the page is shown after the initial page load.

HTML function onload="getData();" translates as follows in the new .js file:

- `getData()` becomes `_demoPageShow()`
- `onload` becomes `_application.show()`

See page 6 for syntax.

Save the contents of this new as a separate .html file. This .html file must be referenced in the initialization file.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd\">
<!-- Copyright 2009 Schneider Electric. All Rights
Reserved. -->

<html>
   <head>
      <title>CM3000 - Slave Device 3 - XMLHTTP</title>
   </head>
   <body style="font-family:Arial" onload="getData();">
      <form name="view_form">
<div id="demoApp">
         <div id="time_spot" style="text-align:
center"></div>
         <table border="1" cellspacing="0" style="width:
600px; margin-left:auto; margin-right:auto">

...

         </table>
         <br /><hr style="width:66%; height:1px" /><div
style="text-align: center; font-size: small">&copy; 2009
SchneiderElectric. All rights reserved.</div>
      </div>

</form>
      <script type="text/javascript">
```

Save this content and all that follows in the required .js file. This .js file must be referenced in the initialization file.

Define variables.

```
(function (OneESP) {

      //Name of this specific application
      var APPLICATION_ID = "plTagDemo";

      var _application =
OneESP.applicationsManager.getApplication(APPLICATION_ID
);

      var PL_TAGS_URL = "/UE/Post__PL__Data?_domain=" +
APPLICATION_ID;

      var _showCount = 0;
      var _hideCount = 0;
      var _parentShow = _application.show;
      var _parentHide = _application.hide;
      var sampleRate = 5*1000;

      if (window.XMLHttpRequest) {
         // If IE7, Mozilla, Safari, and so on: Use
native object
         var xmlhttp= new XMLHttpRequest();
      }
      else {
         if (window.ActiveXObject) {
            // ...otherwise, use the ActiveX control for
IE5.x and IE6
            var xmlhttp = new
ActiveXObject("Microsoft.XMLHTTP");
         }
      }
```

These functions are called when the page is shown or hidden.

```
      function _plTagDemoShow() {
            getData();
      }

      function _plTagDemoHide() {
            xmlhttp.abort();
      }
```

Replaces `onload="getData();"` in the <body> tag above.

```
      // This is the onload call, when your application
is loaded
      _application.show = function() {
            ++_showCount;
            _hideCount = 0;
            if(_showCount === 1) {
                  _parentShow.call(_application);
                  _demoPageShow.call(_application);
            }
      };

      // This is the onunload call, when your
application is unloaded.
      _application.hide = function() {
```

```
            ++_hideCount;
            _showCount = 0;
            if(_hideCount === 1) {
                  _demoPageHide.call(_application);
                  _parentHide.call(_application);
            }
      };
      function getData() {

         var postString = "PL_" + "_3" +
"^3209,3210,3211,3208,1180,1100,1103,1104" + "__PL";
         doRead(xmlhttp, updateData, postString,
getData);
      }
      function doRead(xmlhttpObj, functionName, postString
doneFn, postString, reloadFn) {
         var sampleRate = 5*1000;
         try{
            var temp;
            var Data = new Array();

            xmlhttp.open("POST", PL_TAGS_URL, true);



            xmlhttp.setRequestHeader("Content-Type",
"application/text");

            xmlhttp.onreadystatechange  = function() {
               if(xmlhttp.readyState == 4){
                  try{
                     temp= xmlhttp.responseText;
                     Data=temp.split(",");
                  }
                  catch(exception){
                     ProcessError(xmlhttp.responseText);
                     return;
                  }

                  if(Data.length > 2){
                     functionName(Data);

                     setTimeout(reloadFn, sampleRate)
                  }
                  else{
                     ProcessError(Data);
                  }
               }
            }
            xmlhttp.send(postString)
         }

...

         // Put data on the page
```

NOTE: The `xmlhttp` variable is no longer passed because is global.

`reloadFn` function replaces hard-coded `getData` below.

Change all instances of variable `xmlhttpObj` to the global variable `xmlhttp`.
Replace `"Post__PL__Data"` with `PL_TAGS_URL`

Replace `application/x-www-form-urlencoded` with `application/text`

Change "`getData()`" to the variable parameter above: `reloadFn`.

```
        document.getElementById("frequency").innerHTML
= Frequency;

document.getElementById("currentphasea").innerHTML =
CurrentPhaseA;

document.getElementById("currentneutral").innerHTML =
CurrentNeutral;

document.getElementById("currentground").innerHTML =
CurrentGround;
        document.getElementById("time_spot").innerHTML
= TheTime;
      }
      _application.show();

}(window.OneESP));



      </script>
   </body>
</html>
```

HTML File

Below are the resulting HTML file contents.

```
<div id="content">
      <div id="time_spot"></div>
            <table id="demo_table">
                  <tr>
                     <td colspan="3"
id="table_title">CM3000 - Slave Device 3</td>
                  </tr>
                  <tr>
                     <td class="col1">Frequency</td>
                     <td class="col2" id="frequency"></td>
                     <td class="col3">Hz</td>
                  </tr>
                  <tr>
                     <td class="col1">Current Phase A</td>
                     <td class="col2"
id="currentphasea"></td>
                     <td class="col3">Amps</td>
                  </tr>
                  <tr>
                     <td class="col1">Current Neutral</td>
                     <td class="col2"
id="currentneutral"></td>
                     <td class="col3">Amps</td>
                  </tr>
                  <tr>
                     <td class="col1">Current Ground</td>
                     <td class="col2"
id="currentground"></td>
                     <td class="col3">Amps</td>
                  </tr>
            </table>
      <br />
      <hr/><div id="copyright">&copy; 2016
SchneiderElectric. All rights reserved.</div>
</div>
```

JavaScript File

Below are the resulting JavaScript file contents.

```
 (function (OneESP) {
      //Name of this specific application
      var APPLICATION_ID = "plTagDemo";

      var _application =
OneESP.applicationsManager.getApplication(APPLICATION_ID);

      var PL_TAGS_URL = "/UE/Post__PL__Data?_domain=" +
APPLICATION_ID;

      var _showCount = 0;
      var _hideCount = 0;
```

```
	var _parentShow = _application.show;
	var _parentHide = _application.hide;
	var sampleRate = 5*1000;



	if (window.XMLHttpRequest) {
	// If IE7, Mozilla, Safari, and so on: Use native
object
		var xmlhttp = new XMLHttpRequest();
	}
	else {
		if (window.ActiveXObject) {
			// ...otherwise, use the ActiveX control
for IE5.x and IE6
			var xmlhttp = new
ActiveXObject("Microsoft.XMLHTTP");
		}
	}

	function _plTagDemoShow() {
		getData();
	}

	function _plTagDemoHide() {
		xmlhttp.abort();
	}

	_application.show = function() {
		++_showCount;
		_hideCount = 0;
		if(_showCount === 1) {
			_parentShow.call(_application);
			_plTagDemoShow.call(_application);
		}
	};

	_application.hide = function() {
		++_hideCount;
		_showCount = 0;
		if(_hideCount === 1) {
			_plTagDemoHide.call(_application);
			_parentHide.call(_application);
		}
	};

	function getData() {
		var postString =
"PL__3^S3209,3210,3211,3208,1180,1100,1103,1104__PL";
		doRead(updateData, postString, getData);
	}

	function doRead(doneFn, postString, reloadFn) {
		try{
			var temp;
			var Data = new Array();
```

```
                  xmlhttp.open("POST", PL_TAGS_URL, true);
                  xmlhttp.setRequestHeader("Content-Type",
"application/text");

                  xmlhttp.onreadystatechange  = function()
{
                        if(xmlhttp.readyState == 4){
                              try {

      temp=xmlhttp.responseText;
                                    Data=temp.split(",");
                              }
                              catch(exception) {

      ProcessError(xmlhttp.responseText);
                                    return;
                              }

                              if(Data.length > 2) {
                                    doneFn(Data);
                                    setTimeout(reloadFn,
sampleRate)
                              }
                              else {
                                    ProcessError(Data);
                              }
                        }
                  }
                  xmlhttp.send(postString)
            }
                  catch(exception){
                  ProcessError(exception);
            }
      }

      function processError(errTxt) {
            alert(errTxt);
      }

      function updateData(Registers)
      {

            // Assign Scale Factors
            ScaleFactorA = Registers[0];  // Current Scale
Factor
            ScaleFactorB = Registers[1];  // Neutral
Current Scale Factor
            ScaleFactorC = Registers[2];  // Ground Current
Scale Factor

            // Assign Nominal Frequency
            NominalFrequency = Registers[3];

            // Assign Data Values
            Frequency      = Registers[4];
```

```
                  CurrentPhaseA  = Registers[5];
                  CurrentNeutral = Registers[6];
                  CurrentGround  = Registers[7];

                  // Get the current date and time
                  TheTime = new Date();

                  // Scale Frequency
                  if(NominalFrequency != 400) {
                        Frequency *= 0.01;
                  }
                  else {
                        Frequency *= 0.10;
                  }

                  // Scale Phase A Current
                  CurrentPhaseA *= Math.pow(10, ScaleFactorA);

                  // Scale Neutral Current
                  if (CurrentNeutral == 32768) {
                        CurrentNeutral = "N/A";
                  }
                  else {
                        CurrentNeutral *= Math.pow(10,
ScaleFactorB);
                  }

                  // Scale Ground Current
                  if (CurrentGround == 32768)   {
                        CurrentGround = "N/A";
                  }
                  else {
                        CurrentGround *= Math.pow(10,
ScaleFactorC);
                  }

                  // Put data on the page
                  document.getElementById("frequency").innerHTML
= Frequency;

      document.getElementById("currentphasea").innerHTML =
CurrentPhaseA;

      document.getElementById("currentneutral").innerHTML =
CurrentNeutral;

      document.getElementById("currentground").innerHTML =
CurrentGround;
                  document.getElementById("time_spot").innerHTML
= TheTime;
      }

      _application.show();

}(window.OneESP));
```

Stylesheet

We recommend using a CSS instead of inline styling. Below is the CSS for this example.

```
#demo_table {
      border: 1px;
      border-spacing: 0;
      width: 600px;
      margin-left: auto;
      margin-right: auto;
}

#demo_table td {
      border: 1px solid black;
      padding: 5px;
}

#table_title {
       width: 600px;
       text-align: center;
       font-weight: bold;
       font-size: x-large;
}

#time_spot {
      text-align: center;
}

#copyright {
      text-align: center;
      font-size: small;
}

.col1 {
      width: 300px;
      text-align: center;
}

.col2 {
      width: 90px;
      text-align: center;
}

.col3 {
      width: 100px;
      text-align: center;
}

hr {
      width: 66%;
      height: 1px";
      text-align: center;
      margin-left: auto;
      margin-right: auto;
}
```

Initialization File

You must create an **init.js** file when converting EGX300 pages. Below is the file that corresponds to this example. Refer to *Creating Com'X 510 Custom Web Pages*, 7EN72-0199, for more information on custom page requirements.

```
(function (OneESP) {
      var APPLICATION_ID = "plTagDemo";
      var _application =
OneESP.applicationsManager.getApplication(APPLICATION_ID);
      var _prefix = "applications/" + APPLICATION_ID;
      var _sys_prefix = "/system/http"

      _application.prepare(
            //Set the application title
            "Com'X PL Tag Demo",

            //Set the HTML Content file
            _prefix + "/plTagDemo.html",

            //Set the required JS files
            [
                  _sys_prefix + "/js/API_ComX_General.js",
                  _prefix + "/plTagDemo.js",

            ],
            //Set the required CSS files
            [
                  _prefix + "/plTagDemo.css"
            ]
      );

}(window.OneESP));
```