

Introduction

This tutorial is meant as an introduction to creating HTML/CSS templates based on the Webclient JavaScript library which can be used via the Trace Universal Control Hub (UCH) to interact with connected target devices such as televisions and satellite boxes through an online DHTML interface (graphic or text).

For the purpose of this tutorial, you need to have downloaded two other projects from Trace: The [Universal Control Hub for Java \(UCHj\)](#), and the [SatelliteBox simulation](#). The SatelliteBox simulation is an applet simulating a SATELLITE BOX that can be controlled from the UCH. A target adapter for the SATELLITE BOX is included in the UCH package. Target adapters for other devices (including real devices) may be developed on top of the UCH reference implementation, but this is not in the scope of this tutorial.

Server and Device Connection

This tutorial assumes that you have successfully installed the UCH and SatelliteBox packages provided by Trace. Please read accompanying readme files for details how to do this. Limited instructions have been provided here:

In order to control a fake device via the UCH server using your new interface, you need to do some initial setup work.

First, install Tomcat and copy UCH.war into your web applications directory. For more information on installing Tomcat, check out (<http://www.devx.com/webdev/Article/16416>).

Once Tomcat is installed and the UCH application has been unpacked and is running, simply run the fake SATELLITE BOX device. The fake device and the Tomcat server do not necessarily have to be on the same computer, but they should be on the same local network (with no firewall between them). After running the fake device and the Tomcat server together, your web client interface should automatically connect to the UCH server and allow you to control the fake target device.

Design

This tutorial is not meant to be a design tutorial, but I've included the step here to illustrate the fact that visual design ranging from simple to advanced interfaces can be used to create fully functioning internet remote controls for media devices.

The design shown below is the one I will be using throughout this tutorial. Though this design is not the simplest possible, it is also not the most complex. It allows for a reasonable display of the capacities of the Webclient library operating through an HTML/JavaScript webpage.

The following design will utilize the satellite box's functions which control the satellite box's **power**, **channel**, and **volume** settings. The available functions of a target device are to be specified by its developer/manufacturer in a "socket description". The appendix shows the complete socket description of the satellite box.

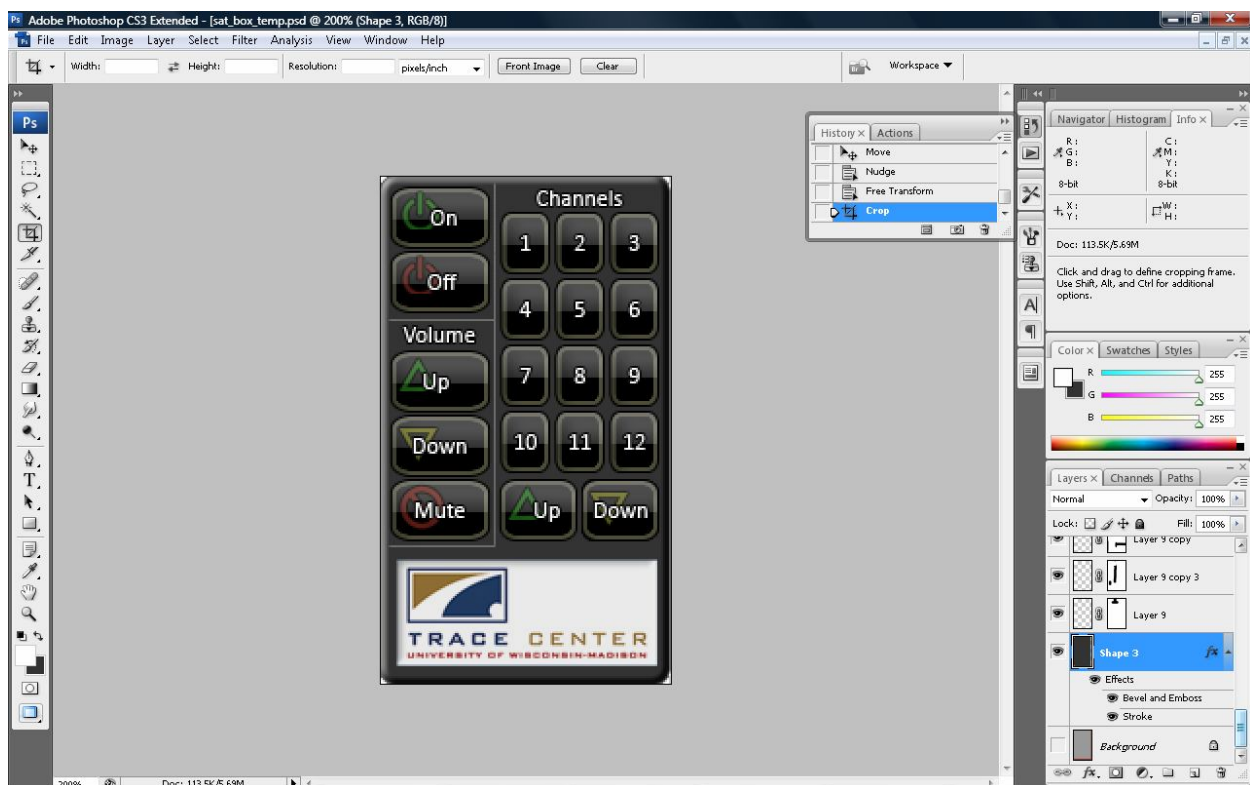


Figure 1: The figure shows a control panel with three groups of buttons, and the Trace logo. The first group of buttons consists of an "On" and an "Off" button. The second, of a "Volume Up" button, a "Volume Down" button, and a "Mute button." The third is a 3x4 array of buttons each corresponding to a different numbered channel, as well as "Channel Up" and "Channel Down" buttons.

Accessibility

It is important to keep users with disabilities in mind when designing an interface. You may want to consider creating multiple versions of the same interface to accommodate for users with poor eyesight, among other things.

One of the nice things about the URC framework is that user interfaces can be easily exchanged (that's why we also call them "pluggable user interfaces").

For example, here is a high contrast version of the above design made for people with poor eyesight:

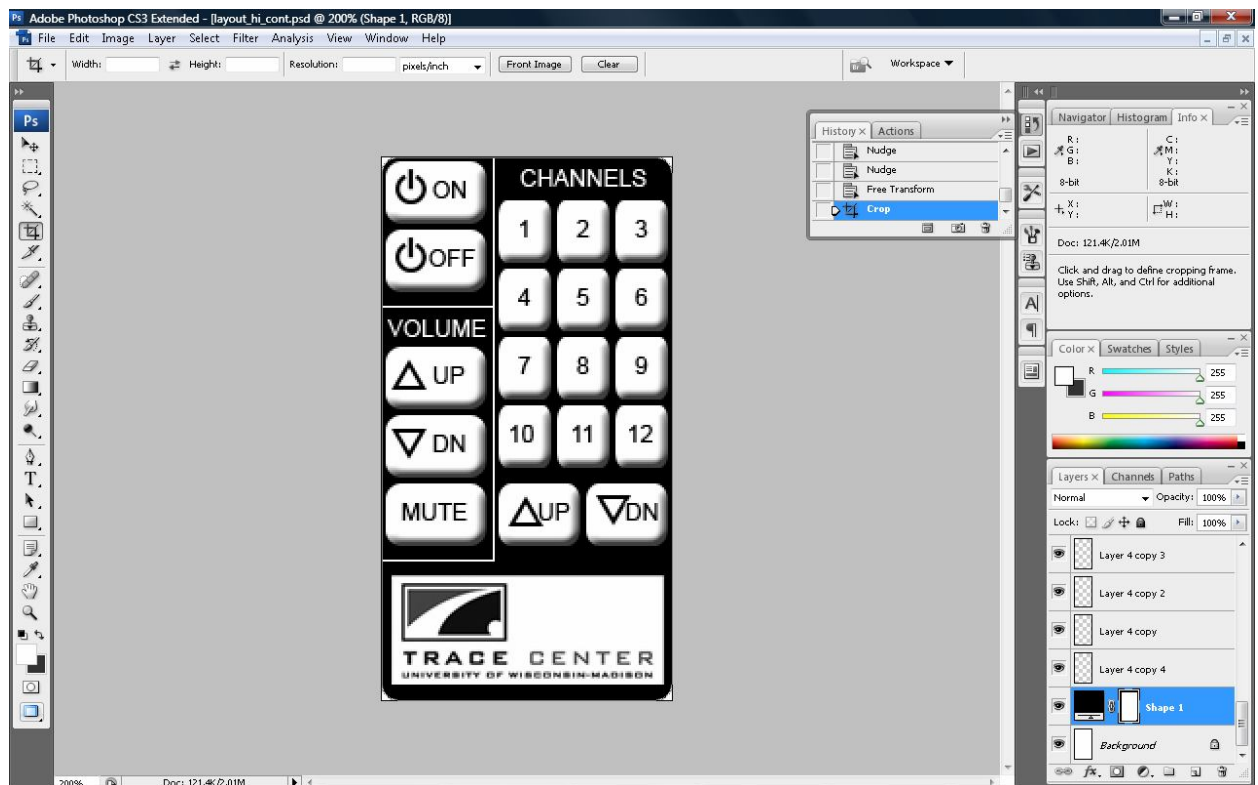


Figure 2: The figure shows a control panel with three groups of buttons, and the Trace logo. The buttons are the same as in figure 1, but are rendered in black and white only, including the logo.

Implementation

<head>

Create a new .html file and open it for modification in whichever .html editor you prefer.

Various JavaScript files which interact with the backend socket and namespace definitions will be made available to you. The <head> area of your .html page must link to these files as follows:

```
<HEAD>
  <TITLE>SAMPLE CONTROLLER</TITLE>
  <META HTTP-EQUIV="CONTENT-TYPE" CONTENT="TEXT/HTML; CHARSET=ISO-8859-1"/>
  <LINK REL="STYLESHEET" HREF="SATELLITEBOX.CSS" TYPE="TEXT/CSS" />

  <SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT"
SRC="ORG_MYURC_WEBCLIENT.JS">
    </SCRIPT>
  <SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT"
SRC="ORG.MYURC.SOCKET.JS">
    </SCRIPT>
  <SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT"
SRC="ORG.MYURC.SCHEMA.JS">
    </SCRIPT>
  <SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT"
SRC="ORG_MYURC_URCHTTP.JS">
    </SCRIPT>
  <SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT" SRC="ORG_MYURC_LIB.JS">
    </SCRIPT>
</HEAD>
```

The <link> tag here connects to a CSS file which handles the layout of visual elements. You can look at the CSS file if you wish, but its creation will not be included in this tutorial.

The <script> tags which follow provide the page with the functions it needs to interact with the Trace UCH and your socket-enabled interactive device.

In order for the web remote interface to function properly the following JavaScript commands must also be included in the **<head>** area of the document:

```
<script language="javascript" type="text/javascript">
  // List of sockets that the client wants to receive updates from (i.e. open
  sessions with). sockets = [ "http://res.myurc.org/upnp/demo/satelliteBox/socket" ];
  updateInterval = 2000;    // Poll for updates every 2 seconds.

  function init()
  {
    try { org_myurc_webclient_init(sockets, updateInterval); }
    catch (ex) { alert(typeof ex === "string" ? ex : ex.message); }
  }

  function finalize()
  {
    org_myurc_webclient_finalize();
  }
</script>
```

These JavaScript functions will be called when the .html page loads and unloads. The **sockets** variable defines, for the UCH, which sockets will be in use by the interface (this is needed for receiving updates from the UCH). The **updateInterval** specifies the polling rate for events from the UCH. **2000** means it polls for updates every **2000** ms, i.e. every **2** seconds.

With that, short of any changes to the CSS for design purposes, all necessary changes to the .html file's **<head>** section have been made.

<body>

In order for the interface to function properly the **<body>** tag itself must appear like the following:

```
<body onLoad="init()" onUnload="finalize()">
```

This tag calls the necessary JavaScript functions that we just defined in the **<head>** section of the page.

At this point you should write your CSS and get the visual display functioning properly with regular HTML and CSS code.

DO NOT change any of the code from the **<head>** or **<body>** sections that you wrote in earlier steps of this tutorial.

Your design should now appear as you like in your chosen web browser.

Interactive Elements

This design uses images to simulate the appearance of buttons. You can look at the CSS if you don't know how this was done and are interested in simulating the effect.

In order to pass data to the interactive device being used you need to employ the **onclick** event handler of the images you are using as buttons. The **onclick** event handler calls the **setValue** function from **org_myurc_webclient.js** which was imported in the **<head>** section of the page.

Note: To make our interface keyboard-accessible, we also need to define an **onkeypress** event handler with the same content as the **onclick** event handler.

Here is an example which sets the **powerMode** value to **ON**:

```

```

org_myurc_webclient_setValue function requires two parameters:

1. A **socketPath** to the value you wish to update. You'll notice that the first part (the socket name) is what we used in the sockets variable in the header code and the remainder (after the '#' sign) is the path to the value to be updated. The satellite box makes the following variable paths available to be updated with the included values or value like them, with the exception of **/previousChannel**, which is a read-only field:
 - a. **/powerMode**
 - i. **ON**
 - ii. **STANDBY**
 - b. **/activeChannel**

- i. Any integer between 1 and 12
 - c. /previousChannel (read-only)
 - i. Any integer between 1 and 12
 - d. /volume
 - i. Any integer between 0 and 100
 - e. /mute
 - i. true
 - ii. false
2. The value you wish to use from the ones listed above.

The available values for **activeChannel/previousChannel** will likely change depending on the data being used, but this general outline will still apply.

You may have noticed that this framework doesn't directly allow for incrementing the volume, as the buttons would suggest. In order to accomplish that you need to have the button call a JavaScript function that you've written which reads in the current value of **volume**, increments it, and then stores the new value in **volume**. Such a function may look like the following:

```
function volumeInc()
{
    var curVol =
    parseInt(org_myurc_webclient_getValue('http://res.myurc.org/upnp/demo/satelliteBox/socket
#/volume'));
    curVol      += 5;
    org_myurc_webclient_setValue('http://res.myurc.org/upnp/demo/satelliteBox/socket#
/volume', curVol)
}
```

Reactive Elements

In addition to having an interface capable of sending messages to a given device through the UCH socket definitions, you can create an interface which reacts to the current state of that device.

For example, through JavaScript you can create an interface which highlights the button corresponding to the current active channel. You can also highlight the mute button if the device is currently muted. These examples are among the least complicated; the UCH system, through JavaScript implementation, allows for far-ranging and creative interfaces which react dynamically to the state of your device.

In order to employ *reactive elements*, which change according live information from the UCH, you need to include a function called **onpostvalue()** in the **<head>** of your page:

```
function onpostvalue(path, value)
{
    switch (path)
    {
        case "http://res.myurc.org/upnp/demo/satelliteBox/socket#/powerMode":
            document.getElementById('poweronbutton').src =
                (thisValue == 'ON' ? 'imgs/power_on_on.jpg' : 'imgs/power_on_off.jpg');
            document.getElementById('poweroffbutton').src =
                (thisValue == 'ON' ? 'imgs/power_off_off.jpg' : 'imgs/power_off_on.jpg');
            break;

        case "http://res.myurc.org/upnp/demo/satelliteBox/socket#/mute":
            document.getElementById('volumemutebutton').src =
                (thisValue == 'true' ? 'imgs/volume_mute_on.jpg' :
                'imgs/volume_mute_off.jpg');
            break;
    }
}
```

Please note that this is not the complete **onpostvalue()** function from the source file. I have included part of it here for illustrative purposes. Basically, what the function does is, whenever a socket variable is given a new value by the interface the UCH calls the **onpostvalue()** function that you have defined. When it does so, the switch statement contained within the **onpostvalue()** function checks to see what socket variable was updated and, depending on the value, performs the given operation.

For example, in this particular **onpostvalue()** function, if the value of the socket variable **mute** (in this case, the full path of **mute** is <http://res.myurc.org/upnp/demo/satelliteBox/socket#/mute> as you can see) is updated, the function checks the posted value and changes the look of the visual "Mute" button on the interface (it will glow if the socket variable is set to **true** and it will appear normal if otherwise).

Here is the interface in action when the connected device has been set to **'ON'** and **channel 6** :

May 5, 2008

[TUTORIAL: HOW TO CREATE AN HTML TEMPLATE WITH THE WEBCLIENT LIBRARY]

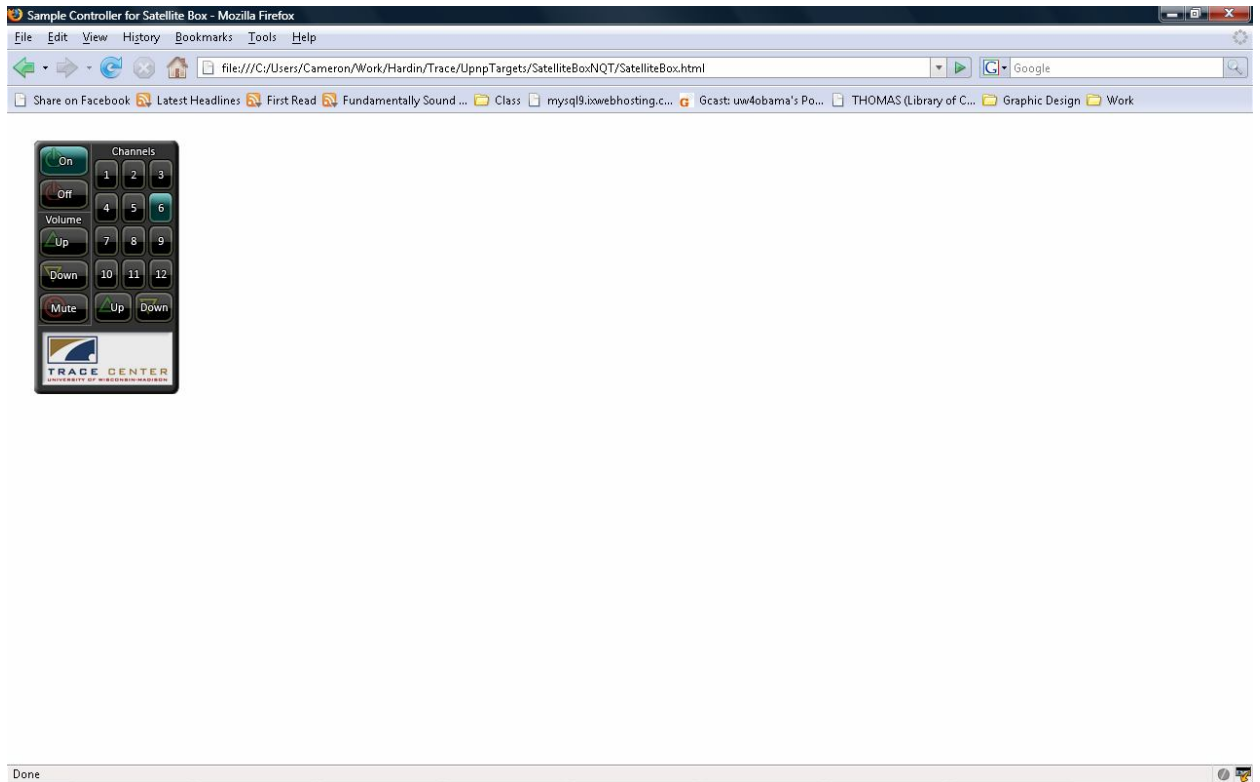


Figure 3: The figure shows the controller with two buttons (the "ON" button and the button corresponding to Channel 6) highlighted. This is an example of how visual elements of the design can react to the behavior of the UCH.

Once finished, the web remote interface can be accessed via any internet-connected device running any standard web browser. Here is a shot of the design running on the Apple iPhone:



Figure 4: The figure features the full-color interface design displayed on the Apple iPhone.

Appendix: Socket Description of the Satellite Box

You can find the socket description of the satellite box in the 'res' directory of SatelliteBoxNQT.zip, as file 'satellite_box.uis'. A socket description follows the syntax of the socket description language, an XML language defined by ISO/IEC 24752-2:2008, User Interface Socket Description.

Here is a complete listing of 'satellite_box.uis':

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  UI socket description (based on ISO/IEC 24752-2) for a Satellite Box.
  The socket description is not tailored to a specific device, but aims
  to reflect functions that are common for all satellite boxes.
-->

<uisocket about="http://res.myurc.org/upnp/demo/satelliteBox/socket"
  id="socket"
  xmlns="http://myurc.org/ns/uisocketdesc"
  xmlns:uis="http://myurc.org/ns/uisocketdesc"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://myurc.org/ns/uisocketdesc http://myurc.org/ns/uisocketdesc
    http://purl.org/dc/elements/1.1/ http://dublincore.org/schemas/xmls/qdc/2006/01/06/dc.xsd
    http://purl.org/dc/terms/ http://dublincore.org/schemas/xmls/qdc/2006/01/06/dcterms.xsd"
  <dc:creator>Jon Hardin</dc:creator>
  <dc:contributor>Gottfried Zimmermann</dc:contributor>
  <dc:publisher>Intelli-Computing Consulting</dc:publisher>
  <dc:rights>Copyright 2008 The Board of Regents of the University of Wisconsin System</dc:rights>
  <dcterms:conformsTo>http://myurc.org/iso24752-2/2007</dcterms:conformsTo>
  <dcterms:modified>2008-05-02</dcterms:modified>

  <!-- Part 1 of Satellite Box socket - power on/off -->

  <variable id="powerMode" type="powerModeType" includesRes="true">
    <dc:description xml:lang="en">
      This socket element allows to adjust the power mode of the device.
      The powerModeType is a string enumeration 'ON', 'STANDBY':
      - 'ON' corresponds to an activated Satellite Box device.
      - 'STANDBY' corresponds to a deactivated Satellite Box that may still be awoken from the network.
    </dc:description>
  </variable>

  <!-- Part 2 of Satellite Box socket - channel selection -->

  <variable id="activeChannel" type="channelType">
    <dc:description xml:lang="en">
      This variable may be used to set the active channel shown by the Satellite.
      A channel is identified by a unique integer between 1 and 12. Channel number 0 is used if
      the device is on standby mode.
      The variable may only be written if the power mode of the device is 'ON'.
      Before switching the channel, the target adaptor saves the current channel to 'previousChannel'
      in order to allow to return to the previously active channel.
      The integer value may be used to fetch long labels and icons from a resource sheet.
    </dc:description>
    <dependency>
      <relevant>value('powerMode') eq 'ON'</relevant>
      <write>value('powerMode') eq 'ON'</write>
    </dependency>
  </variable>

  <variable id="previousChannel" type="channelType">
```

```

    <dc:description xml:lang="en">
      Specifies the channel number of the previously selected channel.
      This variable is read-only (i.e. it cannot be changed by the user).
    </dc:description>
  <dependency>
    <relevant>value('powerMode') eq 'ON'</relevant>
    <write>>false()</write>
  </dependency>
</variable>

<command id="selectPreviousChannel" type="uis:basicCommand">
  <dc:description xml:lang="en">
    Command for switching back to the previously selected channel.
    It can only be activated if the device is on.
    This command switches the values of activeChannel and previousChannel socket variables.
  </dc:description>
  <param idref="activeChannel" dir="inout"/>
  <param idref="previousChannel" dir="inout"/>
  <dependency>
    <relevant>value('powerMode') eq 'ON'</relevant>
    <write>value('powerMode') eq 'ON'</write>
  </dependency>
</command>

<!-- Part 3 of Satellite Box socket - volume control (volume level, mute) -->

<variable id="volume" type="volumeType">
  <dc:description xml:lang="en">
    Volume of the Satellite audio output may be set in range 0 to 100 (inclusive bounds), stepsize 1.
    May only be set if the device is 'ON'.
    When the volume is changed, the mute is deactivated (if it was on).
  </dc:description>
  <dependency>
    <relevant>value('powerMode') eq 'ON'</relevant>
    <write>value('powerMode') eq 'ON'</write>
  </dependency>
</variable>

<variable id="mute" type="xsd:boolean">
  <dc:description xml:lang="en">
    Activates and deactivates the mute function.
    Can only be changed if the device is on.
  </dc:description>
  <dependency>
    <relevant>value('powerMode') eq 'ON'</relevant>
    <write>value('powerMode') eq 'ON'</write>
  </dependency>
</variable>

<schema xmlns="http://www.w3.org/2001/XMLSchema">

  <simpleType name="channelType" id="channelTypeId">
    <restriction base="xsd:unsignedInt">
      <minInclusive value="0"/>
      <maxInclusive value="12"/>
    </restriction>
  </simpleType>

  <simpleType name="powerModeType" id="powerModeTypeId">
    <restriction base="xsd:string">
      <enumeration value="ON"/>
      <enumeration value="STANDBY"/>
    </restriction>
  </simpleType>

  <simpleType name="volumeType" id="volumeTypeId">
    <restriction base="xsd:unsignedInt">
      <minInclusive value="0"/>
      <maxInclusive value="100"/>
    </restriction>
  </simpleType>

```

May 5, 2008

[TUTORIAL: HOW TO CREATE AN HTML TEMPLATE WITH THE WEBCLIENT LIBRARY]

```
    </restriction>
  </simpleType>

</schema>

</uiSocket>
```