Remote Control App

Introduction.

Remote Control gives you a level of operational control over the player even when you are away from your machine. It works by regularly downloading and processing an instruction script so that you can jump to a different page within the project, place a message on the logs or quit the page. It can be remotely controlled through a website.

Purpose

The Remote Control project plugin provides the ability to remotely control the Acquire Player from your website. The App works by regularly downloading and processing an **"instruction script"**.
This script is a list of instructions for the Acquire Player, such as **"jump to page"**, in XML format. See Appendix 1.

Usage

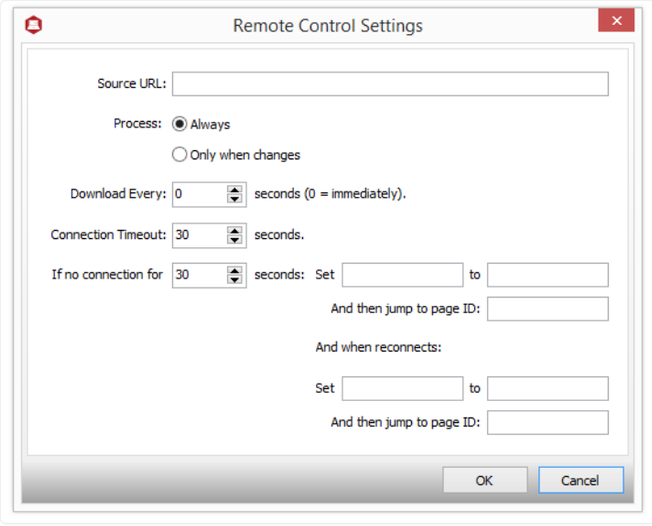
This system allows someone to control the Acquire Player directly from a website using pre-determined layouts and playlists and for emergency messaging etc.
As a simple example your website might allow a user to input data, such as a message to display, and press **'Go'**. When **'Go'** is pressed the website should write the script file to include that message etc. The plugin will then download the script and apply it.
See Appendix 2 for an example.

The script can contain instructions to:

- Set an Acquire variable †
- Jump to a page
- Place a message in the Acquire Logs
- Quit the page
- Time out the page

† Including **$$_ChannelScript** variable to switch playlists.

App Options.



Source URL: Enter the web address for the script to be downloaded from.
Process: The downloaded script can be processed every time they are downloaded or only if there has been a changed.
Download Every: Specify how often the script is downloaded from the URL.

as:lightbulb **Tip:** If your web server is using the Long Polling method this value should be 0. See Appendix 3.

Connection Timeout: The maximum time for a connection attempt to the URL before giving up.

as:lightbulb **Tip:** If your web server is using the Long Polling method this value should be higher than your web server timeout. See Appendix 3.

If no connection for: Enter an Acquire variable to set and/or a page to jump to when there has not been a successful connection to the URL for the enter number of seconds.
Enter another variable and page for when the connection is re-established.

Appendix 1 – The script XML format.
Here is an example script that demonstrates the format of the XML that your website must supply.

```
<?xml version="1.0" encoding="UTF-8" ?>
<script>
    <variable name="$$var1">1</variable>
    <variable name="$$var2">2</variable>
    <variable name="$$var3">3</variable>
    <jump>PageID</jump>
    <log code="123">my message 1</variable>
    <log code="321">my message 2</variable>
    <quit />
    <timeout />
</script>
```

Items are processed in this order:
Variables: The name will be populate with the value e.g. **$$var1=”1”**.
Page Jumps: The page identified by the value will be jumped to.

Logs: The *value* contains the message to logged under the code which should be a number in the range 0 to 999.
Quits:
Timeouts: When there are multiple items of the same sort they are processed top to bottom as they appear in the script.

Appendix 2 – Example remote control website.
Here is a very simple example of a remote control website.
The user enters a **Message** and **PageID**. When **'Go'** is pressed that data is written to the rc.xml script that the plugin will download and process.

index.html: Shows Message and Page ID text boxes and a Go button. Submits entered data to go.php:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/> <title>Show Message</title>
</head>
<body>
    <form id='messageform' method="post" action="go.php">
    Message: <input name="message" type="text" /><br/>
    Page ID: <input name="pageid" type="text" ><br/>
    <input type="submit" value="Go" />
</form>
</body>
</html>
```

go.php: Saves passed Message and PageID data to the script and returns to index.html.

```
<?php
    $message=@$_POST["message"];
    $pageid=@$_POST["pageid"];
    $file = fopen("rc.xml", "w");
    fwrite($file, '<?xml version="1.0" encoding="UTF-8" ?>');
    fwrite($file, '<script>');
    fwrite($file, '<variable name="$$message">'.$message.'</variable>');
    fwrite($file, '<jump>'.$pageid.'</jump>');
    fwrite($file, '</script>'); fclose($file);
    header("Location: index.html"); ?>
?>
```

The plugin will download the **rc.xml**, set *$$message* to the value entered and jump to the specified page. That page may contain an element to display **$$message**.

Appendix 3 – Long Polling.
The plugin regularly polls the website for the script file. The website server normally sends the script file on every request. As a connection takes place on every request this impacts on the speed of script delivery. However, you can configure your web server application to deliver the script as soon as possible using **"Long Polling"**.

This means that upon a request for the script your server does not send it immediately. Instead it should hold the request and wait until the script changes (e.g. when a user has pressed **'Go'** on your webpage and the script has been written.) or after a suitable timeout.

If you use this method then the plugin must:

- **Set the 'Download Every' value to 0:** This is the time between download attempts and setting it to zero will make sure that the next connection after a download is immediate.
- **Set the 'Connection Timeout':** to a value higher than your web server Long Polling timeout. This will prevent the plugin from assuming there is a bad connection because the request is taking too long.