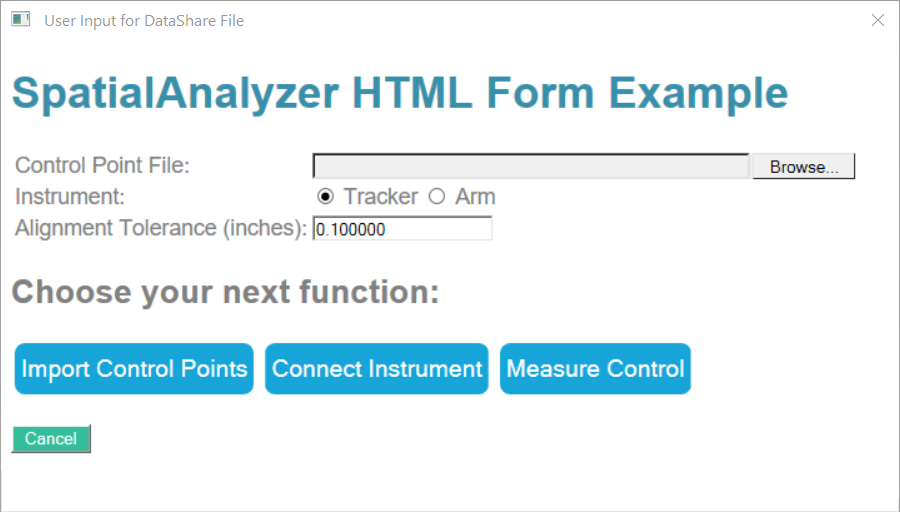
HTML Forms in SpatialAnalyzer MPs



Contents

[Introduction 2](#_Toc54691133)

[Basic Implementation 2](#_Toc54691134)

[Supported Input Types 2](#_Toc54691135)

[Text 2](#_Toc54691136)

[Number 2](#_Toc54691137)

[Radio 3](#_Toc54691138)

[File 3](#_Toc54691139)

[Checkbox 3](#_Toc54691140)

[Advanced Techniques 4](#_Toc54691141)

[Pre-Populating Forms 5](#_Toc54691142)

[Using CSS and JavaScript for HTML Forms 5](#_Toc54691143)

[Unexpected Styling 6](#_Toc54691144)

[Linking Form “Save” and “Cancel” to Custom Buttons 6](#_Toc54691145)

[Example: InspectPart.mp 7](#_Toc54691146)

# Introduction

Measurement Plans are a powerful tool to automate and standardize metrology processes using SpatialAnalyzer. In applications that require flexibility, choice, and varying values, it is often necessary to prompt the user for information. HTML forms allow this information to be gathered in one dialog, rather than a series of prompts from the MP. This technique can make the MP feel more polished and dramatically improve the user experience.

# Basic Implementation

There are three components necessary to include an HTML form in MP:

1. MP: The measurement plan that contains a Load HTML Form step.
2. HTML: The form that allows user input.
3. datashare: A text file that facilitates the information transfer between the HTML and MP.

The MP is created in SpatialAnalyzer and will likely have many steps preceding and following the call to the HTML. The HTML is a separate file that can be created in a text editor or any number of free editors that allow for faster formatting and testing than a text editor (i.e. Notepad) would. The datashare file can have a “.ds” or “.txt” extension and can be Binary if you do not want the users to make manual edits outside of the MP framework.

## Supported Input Types

The HTML language offers many options for formatting data entry forms, but there are some requirements for input elements that make them compatible with MP.

The MP identifies the data type to be written to a datashare file by the first letter of the id attribute.

* String (S)
* Integer (I)
* Double (D)
* Boolean (B)

The <input> tag is used to get the desired value from the user, of a type determined by the first letter of the id attribute. The following are input types supported.

### Text

This can be a String, Integer or Double value. The following example will yield a String in the datashare file called selectedFunction.

<input type="text" id="S3" name="selectedFunction" value="">

The HTML window will typically display a box for which the user can type their response.

### Number

“Number” typically displays a box similar to “Text”, but the HTML window will prevent the user from entering non-numerical values.

<tr>  
 <td>Alignment Tolerance (inches):</td>  
 <td><input type="number" id = "D1" name="alignmentTol(in)"

value="{alignmentTol}"></td>  
</tr>

The resulting HTML display of the table elements above looks as follows.



The user-entered alignmentTol(in) value will be stored in the datashare file as a Double.

### Radio

When the intent is to choose one option out of two or more, a “radio” button may be a good way to present your decision to the user. Each choice is presented as an input element with the same id.

<td>Instrument:</td>  
<td><input align="left" type="radio" id="S2" name="Instrument" value="Tracker" checked> Tracker   
<input align="left" type="radio" id="S2" name="Instrument" value="Arm"> Arm</td>

The table row consisting of the two elements above will look as follows, with the user able to check either Tracker or Arm.



### File

It is possible to obtain the path to a user-selected file with the HTML scheme. To the datashare and MP, the result is a String.

<input type="file" id = "S1" size="55" name="pathToControlPts" value="{pathToControl}">

The HTML will display a box that upon selection, pops a browse to file dialog. Once the file is selected, the path will be displayed in the box.



### Checkbox

If the user is meant to answer a series of yes or no questions, it may be helpful to provide checkboxes. This returns a Boolean to the datashare file.

Below is an example of a table in HTML that allows the user to select exactly what geometry relationship criteria to report.



A portion of that code is shown below.

<tr>  
 <td>X</td>  
 <td><input type="checkbox" id="B1\_N" name="B1\_N" value=""></td>  
 <td><input type="checkbox" id="B1\_M" name="B1\_M" value=""></td>  
 <td><input type="checkbox" id="B1\_D" name="B1\_D" value=""></td>  
 </tr>  
 <tr>  
 <td>Y</td>  
 <td><input type="checkbox" id="B2\_N" name="B2\_N" value=""></td>  
 <td><input type="checkbox" id="B2\_M" name="B2\_M" value=""></td>  
 <td><input type="checkbox" id="B2\_D" name="B2\_D" value=""></td>  
 </tr>  
 <tr>  
 <td>Z</td>  
 <td><input type="checkbox" id="B3\_N" name="B3\_N" value=""></td>  
 <td><input type="checkbox" id="B3\_M" name="B3\_M" value=""></td>  
 <td><input type="checkbox" id="B3\_D" name="B3\_D" value=""></td>  
 </tr>

# Advanced Techniques

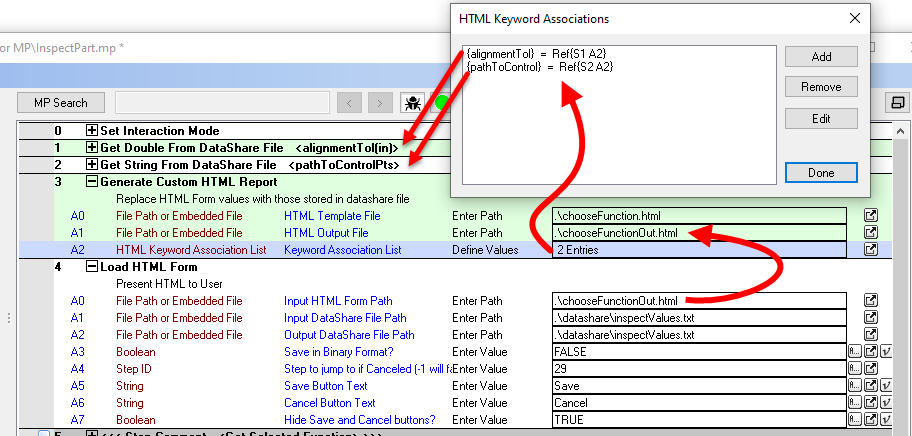
Once the mechanism behind the exchange of data between MP and HTML is understood, more advanced techniques can be employed to continue to improve the user experience. This document is not meant to be a primer in HTML or JavaScript, but there are some helpful tips below that SA users may find valuable.

## Pre-Populating Forms

Sometimes it is desirable to seed the HTML form with default values. Other times, the HTML form may be presented at many stages to the user over the course of a measurement or analysis job, and the previous selections need to be preserved to relieve the user from redundant data entry. In both cases, an existing datashare file and an additional step in the MP may be called for: Generate Custom HTML Report.

The HTML can be told to replace certain keywords with values defined in the MP. If the keyword to replace is within the value tag of an element, the form can be initially displayed with values defined from the MP. If these values are read from a datashare file like the example below, it is possible to pre-populate an HTML form with the user’s previous selections. With this method, the value held by a placeholder “{alignmentTol}” in the chooseFunction.html file can be displayed as the datashare value “alignmentTol(in)” when the user sees chooseFunctionOut.html. The curly brackets {} are not necessary for the keywords, but they are a helpful way to distinguish the pieces of the HTML that will be replaced by the MP.

<input type="number" id = "D1" name="alignmentTol(in)" value="{alignmentTol}">



## Using CSS and JavaScript for HTML Forms

A seasoned and complete language like HTML has many additional benefits that can be used to enhance formatting, data validation, and flow.

Cascading Style Sheets (CSS) can be used to style your form’s font, colors, fieldsets, tables, spacing and essentially every element’s appearance. A single CSS stylesheet can be linked to each HTML page displayed by the MP so that a uniform appearance is presented for the entire SpatialAnalyzer MP solution.

JavaScript can add a dynamic element to your form. It can be used to prompt for different information based on a user-selection within the form, validate input, and pass back button selection information to the MP. Javascripts can be contained in the header or in a separate directory defined in the header. An example of a simple JS script is below.

<script type="text/javascript">  
 **function** nextFunction(nextFun) **{**   
 document.getElementById(*"S3"*).value = nextFun;  
 **}**  
</script>

This function is run following a button onclick event.

<input type="button" class="classSaveButton buttonList" value="Measure Control" onclick="nextFunction(this.value)" />

The value of the button, i.e. “Measure Control”, is passed to the “nextFunction” script. Then, the value is written to the element with the id “S3” so that it is accessible to the datashare file. The “S3” input is contained within an element styled with a CSS class “hiddenInput” which renders it invisible to the user, but still accessible to the datashare.

(in header)

**.**hiddenInput **{**visibility**:**hidden **}**

(in body)

<div class="hiddenInput"> *<!--Allows datashare update from nextFunction function-->*  
 <input type="text" id="S3" name="selectedFunction" value="">  
 </div>

## Unexpected Styling

If a style does not look as you think it should, or a script does not run as expected, it may be that the browser window is rendering the code with the inappropriate or outdated settings. This can be overcome by including the following tag in the header of your HTML to elicit a more desirable browser behavior.

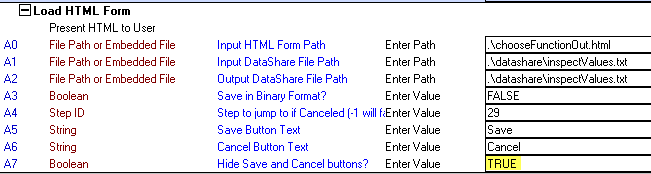
<meta http-equiv="x-ua-compatible" content="IE=edge">

## Linking Form “Save” and “Cancel” to Custom Buttons

The default form presented from the MP contains Save and Cancel buttons as part of the window. In some applications, the button selected by the user is the choice for how to proceed with the MP. It is undesirable to have the operator select the button that picks the next function, for example, then also click the form’s “Save” button.

It is possible to hide both the “Save” and “Cancel” buttons from the HTML window and link these actions to custom buttons. Here are the three steps to achieve this added user friendliness:

1. Choose TRUE in Argument 7 of the Load HTML Form MP step (Hide Save and Cancel Buttons?)



1. Define two classes within your header or CSS stylesheet
   1. The class definition can be blank, or it can have additional styling.

<style>

**.**classSaveButton **{** **}**

**.**classCancelButton **{**background-color**:** #33BD9A**;** color**:** white**;}**

</style>

1. Apply the pertinent class to the buttons that you have created. Multiple buttons can contain the Save and Cancel classes.
   1. The class can be optionally combined with other classes with a space in between.

<input type="button" class="classSaveButton buttonList" value="Measure Control" onclick="nextFunction(this.value)" />

<input type="button" class="classCancelButton" value="Cancel" />

# Example: InspectPart.mp

Many of the code snippets and images presented in this document are taken from an example script and its supporting HTML and datashare file. This sample demonstrates the basic HTML interaction as well as the advanced topics described above.

There are three input types demonstrated: file, radio, and number. There are three functions to choose from which are displayed as buttons that have the “classSaveButton” class so that the HTML window is closed upon selection. There is also a Cancel button containing the “classCancelButton” class.

The “Import Control Points” button is the only function that does any real work with the code provided. The other two functions simply demonstrate the structure of the MP and notify the user of their selection. If “Import Control Points” is selected and the user has browsed to a valid text file with points in the PointName X Y Z format, then the MP will import those points into a group matching the file name. Sample point files are provided in the control directory.

Following the Load HTML Form step, the selectedFunction string is read from the datashare file. A series of String Comparison steps are used to determine where to jump within the MP to achieve the function that the user selected. A more functional example would point to subroutines corresponding to the selected function, but this MP structure can be adapted for your purposes and coding style.

