How do I add hypermedia controls to a resource?

Background

We've already discussed how to add hypermedia to a resource using the standard HAL *_links* and *_embedded* reserved properties. These properties will cover the majority of your hypermedia needs. However, in this same article we introduced the *AddFormAction* service to address hypermedia requirements that go beyond the design intent of the *_links* and *_embedded* properties.

To understand the *AddFormAction* service, it is important to understand the inspiration behind its design: *HTML*. As most developers will recognize, HTML is both a *media type* and a *markup language*. Because it is a media type, we can borrow from its well defined and understood ***hypermedia controls*** when defining similar capabilities for the JSON media type. In HTML, we find that *hypermedia controls* are used for three general purposes:

1. To identify related resources through hyperlinks that allow the client to change the application state (e.g., the <a> tag).
2. To embed resources from another URI into the current resource (e.g., the <img> tag).
3. To provide the client input and submit capabilities to interact with the current resource (e.g., the <form> tag.)

It would seem that HAL covers the first two purposes reasonably well. The *_links* property provides the same purpose as the **`<a>`** tag and the *_embedded* p roperty provides the same purpose as tags like **`<img>`** (which is just one of a few tags that embed resources). However, the third purpose - the ability to provide richer interaction with the resource via form-like controls - is not addressed by HAL.

New Reserved Property: _forms

In keeping with the spirit of HTML hypermedia controls, the SRP HTTP Framework supports a custom property called ***_forms***. It serves the same purpose as the HTML **`<form>`** tag and borrows some of the same element names. Here is an example of what *_forms* hypermedia controls looks like:

```
{
   "_forms":{
      "addPhone":{
         "method":"POST",
         "action":"https://www.examples.org/api/user/matthew/phone",
         "title":"Add Phone",
         "fields":{
            "type":{
               "default":"Cell",
               "required":true,
               "visible":true
            },
            "number":{
               "default":"",
               "required":true,
               "visible":true
            }
         }
      }
   }
}
```

The *_forms* property contains one or more *sub-properties*. Each *sub-property* name defines a unique *form action* and is typically named in a verb+noun format (e.g., *addPhone* in the above example). The *sub-property* values are themselves additional sub-property name/value pairs.

Each defined *form action* contains the following *sub-properties*:

- *method* to indicate the HTTP method used when making the request to the server. (Required)
- *action* to indicate the target URI for the request. (Required)
- *title* to provide a user-friendly label for the *form action*. (Optional)
- *fields* to identify relevant properties from the primary resource object and to document how they are to be handled. (Optional)

In the above example, the intent of this *form action* is to describe how a new phone number can be added to the current resource. Clients should be able to consume this meta data and discover that this is done by sending a JSON object containing the *type* and *number* properties via a POST to the indicated URI.

fields Sub-Property

The *fields* sub-property contains one or more *sub-properties*. Each of these *sub-properties* are the names of a *property* in the primary resource object. In the above example, *type* and *number* are expected to be properties within the primary resource object. The *sub-property* values are themselves additional sub-property name/value pairs.

Each defined *field name* contains one or more of the following *sub-properties*:

- *default* to indicate the value of the resource object property that should be submitted unless overridden by the client.
- *required* to indicate that this resource object property must have a value in order for the *form action* to be accepted.
- *visible* to indicate if this resource object property should be visible to the client.

Hierarchical Outline

To help visualize the _forms structure, here is a general outline:

- _forms
    - *form action 1*
        - method
        - action
        - title
        - fields
            - *field 1*
                - default
                - required
                - visible
            - *field 2*
                - default
                - required
                - visible
    - *form action 2*
        - method
        - action
        - title
        - fields
            - *field 1*
                - default
                - required
                - visible
            - *field 2*
                - default
                - required
                - visible

Using the *AddFormAction* Service

With our background out of the way, we can now demonstrate how to call the ***AddFormAction*** service. To make this easy, we'll implement the *form action* that appears in the above example:

```
API customers.ID.GET

    KeyID           = EndpointSegment

    ColumnNames     = 'FIRST_NAME' : @FM : 'LAST_NAME' : @FM : 'ADDRESS' : @FM : 'CITY' : @FM : 'STATE' : @FM :
'ZIP'
    PropertyNames   = 'firstName' : @FM : 'lastName' : @FM : 'address' : @FM : 'city' : @FM : 'state' : @FM :
'zipCode'
    // Create a JSON object in memory.
    objResource     = HTTP_Resource_Services('GetObject', 'CUSTOMERS', KeyID, ColumnNames, PropertyNames)

    If Error_Services('NoError') then
        // Add _forms sub-property hypermedia control.
        Fields          = 'type' : @FM : 'number'
        FieldProperties = 'Cell' : @VM : True$ : @VM : True$ : @FM : '' : @VM : True$ : @VM : True$
        HTTP_Resource_Services('AddFormAction', objResource, 'addPhone', 'POST', FullEndpointURL, 'Add Phone',
Fields, FieldProperties)
    end

    If Error_Services('NoError') then
        // Serialize the JSON object.
        jsonResource    = HTTP_Resource_Services('GetSerializedResource', objResource)
        // Set the response body with the serialized JSON object and set the Content-Type response header.
        HTTP_Services('SetResponseBody', jsonResource, False$, 'application/hal+json')
    end else
        // There is an error condition so call the SetResponseError service.
        HTTP_Services('SetResponseError', '', '', 500, Error_Services('GetMessage'), FullEndpointURL)
    end

end api
```