How do I add a sub-resource to a resource?

If you have already learned how to add a property or a sub-property to a resource object, you already know the basics of how these *GetObject* companion services work. As one might expect, there is an *AddSubResource* service available to **add a sub-resource to a resource**. In fact, there are two principle services available to the developer. The other service is named *AddSubResourceObject*. Both of these are explained below.

AddSubResource Service

The calling signature for the ***AddSubResource*** service is virtually identical to the *AddSubProperties* service (i.e., the *plural* version of the *AddSubProperty* service). The key difference is that when the *AddSubProperties* service is called more than once for the same parent *property* name and the same *sub-property* names, the new *sub-property* values replace the old ones. The *AddSubResource* service, on the other hand, adds (or appends) new *sub-resource* objects to the existing array of *sub-resources* associated with the parent *property*. Here is an example of what the code might look like:

```
objResource     = HTTP_Resource_Services('GetObject')
If Error_Services('NoError') then
    SubResourcePropertyNames    = 'type'     : @FM : 'address'            : @FM : 'city'        : @FM :
'county'  : @FM : 'state' : @FM : 'zip'
    SubResourcePropertyValues   = 'Mailing'  : @FM : 'PO Box 1234'        : @FM : 'New Orleans' : @FM :
'Orleans' : @FM : 'LA'    : @FM : '70116'
    HTTP_Resource_Services('AddSubResource', objResource, 'address', SubResourcePropertyNames,
SubResourcePropertyValues)
    SubResourcePropertyValues   = 'Shipping' : @FM : '6649 N Blue Gum St' : @FM : 'New Orleans' : @FM :
'Orleans' : @FM : 'LA'    : @FM : '70116'
    HTTP_Resource_Services('AddSubResource', objResource, 'address', SubResourcePropertyNames,
SubResourcePropertyValues)
    // Serialize the JSON object.
    jsonResource    = HTTP_Resource_Services('GetSerializedResource', objResource)
end
```

Because we are adding entire resources with each call to the *AddSubResource* service, the entire sub-resource must be passed in at once. Our resulting resource object will look like this:

```
{
   "address":[
      {
         "type":"Mailing",
         "address":"PO Box 1234",
         "city":"New Orleans",
         "county":"Orleans",
         "state":"LA",
         "zip":"70116"
      },
      {
         "type":"Shipping",
         "address":"6649 N Blue Gum St",
         "city":"New Orleans",
         "county":"Orleans",
         "state":"LA",
         "zip":"70116"
      }
   ]
}
```

Actual code will likely loop through each item of an AMV group or each row as Key IDs are retrieved from a database cursor. Each iteration would call the *A ddSubResource* service.

Developers also have access to the ***AddSubResources*** service, which is a a wrapper around the *AddSubResource* service. It allows values to be passed in as an @RM/@FM array so multiple sub-resource objects can be added in a single call.

AddSubResourceObject Service

The *AddSubResource* service works like the *AddProperty* and *AddSubProperty* service in that serialized data is expected to be passed in through the arguments. The ***AddSubResourceObject*** service allows the developer to pass in an object handle. This is particularly useful when the sub-resource object itself is too complex to be created using the *AddSubResource* service alone (which only knows how to create a flat sub-resource of property names and values). In the *What is a sub-resource?* article we note that sub-resources can be as complex as any resource. In fact, it is possible for sub-resources to contain sub-resources and so on.

Therefore, in order to add a complex *sub-resource* to a resource object, it is necessary to build the *sub-resource object* using other services (e.g., *GetObjec*
, *AddProperty*, *AddSubProperty*, etc.) and then use the object handle in the *AddSubResourceObject* service. Here is how the above code could be re-written using the *AddSubResourceObjec*t service:

```
objResource     = HTTP_Resource_Services('GetObject')
If Error_Services('NoError') then
    PropertyNames   = 'type'     : @FM : 'address'            : @FM : 'city'        : @FM : 'county'  : @FM :
'state' : @FM : 'zip'
    PropertyValues  = 'Mailing'  : @FM : 'PO Box 1234'        : @FM : 'New Orleans' : @FM : 'Orleans' : @FM :
'LA'    : @FM : '70116'
    objSubResource  = HTTP_Resource_Services('GetObject')
    HTTP_Resource_Services('AddProperties', objSubResource, PropertyNames, PropertyValues)
    HTTP_Resource_Services('AddSubResourceObject', objResource, 'address', objSubResource)
    PropertyValues  = 'Shipping' : @FM : '6649 N Blue Gum St' : @FM : 'New Orleans' : @FM : 'Orleans' : @FM :
'LA'    : @FM : '70116'
    objSubResource  = HTTP_Resource_Services('GetObject')
    HTTP_Resource_Services('AddProperties', objSubResource, PropertyNames, PropertyValues)
    HTTP_Resource_Services('AddSubResourceObject', objResource, 'address', objSubResource)
    // Serialize the JSON object.
    jsonResource    = HTTP_Resource_Services('GetSerializedResource', objResource)
end
```

Admittedly, the *AddSubResourceObject* service appears to require more code to implement than the *AddSubResource* service. For simple *sub-resource* objects this is true. But as already noted, the *AddSubResourceObject* service is intended for more complex *sub-resource* objects. It is also quite useful when another routine is responsible for creating the object handle and this is returned conveniently to a calling routine and used in the *AddSubResourceObject* service.

At this time we will introduce the ***AddSubResourceObjects*** service. This is a wrapper around the *AddSubResourceObject* service and it accepts an @FM delimited list of object handles. This is quite useful when a separate routine returns multiple objects handles at once. The below code demonstrates this in action using the *GetObjects* service and the *AddSubResourcesObjects* service:

```
objResource     = HTTP_Resource_Services('GetObject')
If Error_Services('NoError') then
    objSubResources = HTTP_Resource_Services('GetObjects', 'CONTACTS', 'SELECT CONTACTS WITH FIRST_NAME
STARTING "H"', 'FIRST_NAME' : @FM : 'LAST_NAME', 'firstName' : @FM : 'lastName')
    HTTP_Resource_Services('AddSubResourceObjects', objResource, 'contacts', objSubResources)
    // Serialize the JSON object.
    jsonResource    = HTTP_Resource_Services('GetSerializedResource', objResource)
end
```

When this code is tested against the default *CONTACTS* table that ships with the SRP HTTP Framework, the following resource object is produced:

```
{
   "contacts":[
      {
         "firstName":"Harrison",
         "lastName":"Haufler"
      },
      {
         "firstName":"Haydee",
         "lastName":"Denooyer"
      },
      {
         "firstName":"Heike",
         "lastName":"Berganza"
      },
      {
         "firstName":"Helga",
         "lastName":"Fredicks"
      },
      {
         "firstName":"Herman",
         "lastName":"Demesa"
      },
      {
         "firstName":"Herminia",
         "lastName":"Nicolozakes"
      },
      {
         "firstName":"Hillary",
         "lastName":"Skulski"
      },
      {
         "firstName":"Howard",
         "lastName":"Paulas"
      }
   ]
}
```