```
1
2
3
4  //
5    ================================================================================================
6    =========
7  //
8    ================================================================================================
9    =========
10 // Each Form has this onclick function call for updating the table(s) in a form
11
12 function commitTableRecs(formID, tablename, action)
13 {
14
15   var table_info = {} ;  // Given an table name it supplies the PK field name
16
17   //
18     =*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*
19     =*=*=*=*=
20   // Go pay a visit to PHP to get the name of the ID Field on the form currently loaded
21   // DO AJAX with Success callback to handle the variables
22
23   var php_params = "job=GetTableIDs" ;
24
25   $('#debug').html(php_params) ;
26
27   $.ajax({
28     url: '/cmdb/PHP/do_FormCommit_work.php',
29     method:     "GET" ,
30     cache:      false ,
31     data:       php_params  ,
32     dataType:   'JSON'
33     }).done(function(data)
34     {
35       console.log('AJAXed to do_FormCommit_work.php for '+action+'TableRecs() on
36         '+tablename+' to get table_info[], data.status='+data.status) ;
37
38       table_info = (typeof data === 'string') ? JSON.parse(data) : data ;  // e.g.:
39         table_info['ModemConfig'] => modem_config_id
40
41
42       extract_fields_for_commit(table_info, formID, tablename, action) ;
43
44     }      ) ;
45
46 }
47
48
49 function extract_fields_for_commit(table_info, formID, tablename, action)
50 {
```

```
51
52  // We're going to be passing a lot of info (objects & arrays) around in this jArray
      array
53
54
55  var jArray = [] ;
56  var job = {} ;
57  var object_list = [ '[0][job], [0][object_list]===this, [0][nFields]',
58                      '[1][{formSpecs}]: USAGE: [1][formID], [1][parent_table]',
59                      '[2][{table_ids}]: USAGE: [2][ModemConfig]=>modem_config_id, ...',
60                      '[3]...[nFields][{fieldSpecs}]:  USAGE: [3][name], [3][type],
61                        [3][value], [3][orig_value], [3][tablename]',
62                      '[3+nFields]...[?] [{pkValues}]:  USAGE: [n][name], [n][value]' ] ;
63  var formSpecs = {} ;
64
65  job['job'] = 'SubmitFor'+action ;
66  job['object_list'] =  object_list ;
67  job['nFields'] = 0 ;
68  formSpecs['formID'] = formID ;
69  formSpecs['parent_table'] = tablename ;
70
71
72  // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      - - - - -
73  // Unlike just below where I do similar check/set, there is no such table "GeneralNote".
74
75  var inserting_a_general_note = 0 ;
76  if ( tablename === 'GeneralNote' )
77  {
78     tablename = 'Note' ;
79     inserting_a_general_note = 1 ;
80     formSpecs['parent_table'] = tablename ;
81  }
82
83
84  jArray.push(job) ;
85  jArray.push(formSpecs) ;
86  jArray.push(table_info) ;
87
88  // We don't want to update a Note table row with the row# of the noted table, so we have
      to do
89  // some special handling in this case.
90  // Two fixes are necessary, keeping in mind that as of now the Note form appears above
      the working
91  // form except for ModemConfig.  Being above or below makes a difference as to when we
      will, (in
92  // the normal flow of this code), know what table to assign to each field.
93  // Along with adding 'data-tablename="parent tablename"' to the Note table's fields,
      (which will
94  // set the correct table for Note's fields, a couple other tricks are needed.
95
96
97
98
99
100
```

```
101  var updating_a_note = 0 ;
102  var inserting_a_reminder = 0 ;
103
104  if ( tablename === 'Note' )
105  {
106    tablename = $('#note_src_table').val() ; // Non-note fields will now have correct
107      ownership.
108
109    // However, in the case of SwapModems, we have to not keep the here-confusing
110      '(On/Off)' designation
111    // which is set in callbackNote() in FormSearchScripts.js, just after the tablename
112    if ( tablename.indexOf(' ') !== -1 )
113    { tablename = tablename.substring(0, tablename.indexOf(' ')) ; }
114
115    updating_a_note=1 ;
116  }
117  // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
118    - - - - -
119  if ( tablename === 'Reminder' )
120  {
121    tablename = $('#reminder_ref_table').val() ;  // Non-reminder fields will now have
122      correct ownership.
123    inserting_a_reminder = 1 ;
124  }
125
126  //
127    ------------------------------------------------------------------------------------
128    ---------
129  // Get all of the active form's values
130
131  $('form input, form select, form textarea, form checkbox')
132   .each(
133    function(index)
134    {
135      var input = $(this);
136      var type = input.prop('type') ;
137      var name = input.attr('name') ;
138
139      switch (type.substr(0,6))
140      {
141      case 'select':
142      case 'text':
143      case 'number':
144      case 'date':
145      case 'checkb':
146      case 'textar':
147      case 'email':
148      case 'hidden':
149
150        if ( type.substr(0,6) !== 'hidden' || ( type.substr(0,6) === 'hidden' && (name ===
```

```
          'table_row' ||
                                                                                name ===
                                                                                  '
                                                                                  note_t
                                                                                  able_r
                                                                                  ow') )
                                                                                  )
        {

          var value = input.val() ;  // Why this and not .text() works for this textarea
            but not the orig, I don't know
          if ( type === 'checkbox' )
          {
            value = 0 ;
            var temp =  input.prop('checked') ;
            if ( temp !== false ) { value = 1  ; }  // The test seems to return 1 or '',
              else false
          }
          if ( name === 'wireless_no' && value.indexOf('(') !== -1 )
          {
            value = value.strip_phone_format() ;
          }
          if ( name === 'simm_identifier' || name === 'IMEI' || name === 'IMSI' )
          {
            value = value.replace(' ', '') ;
          }

          if ( type !== 'textarea' )
          { var orig_value = $('input[name=orig_'+name).val() ; }
          else
          { var orig_value = $('#orig_'+name).text() ; }
          if ( typeof orig_value === 'undefined') { orig_value = '' ; }

          // So PHP no burp w/o a value here
          if ( name === 'table_row' || name === 'note_table_row' )
          { orig_value = value ; }

          var parent_table = input.data('tablename') ;  //   <-- c/o form input with
            data-tablename="somevalue" within its < input ></ input >
                                                  //   May be null (undefined) if
                                                    not set in the form
          if ( typeof parent_table === 'undefined' ) { parent_table = tablename ; }

          // More special circumstances
          // After the note DIV has slid out of view its form elements are still found
            HERE, so we
          // have to handle the do/don't include here.  (My jQuery exclusion didn't work.)
          // When updating a Note we do not want to update its referencing table

          if ( ( !updating_a_note      && parent_table !== 'Note'     || updating_a_note
```

```
201             && parent_table === 'Note') &&
202                ( !inserting_a_reminder && parent_table !== 'Reminder' ||
203                  inserting_a_reminder && parent_table === 'Reminder')  )
204           {
205             console.log('Type: ' + type + '    Name: ' + name + '    Value: ' + value + '
206               OrigValue: '+orig_value+'   tablename: '+parent_table );
207
208             var fieldSpecs = {} ;
209             fieldSpecs['name'] = name ;
210             fieldSpecs['type'] = type ;
211             fieldSpecs['value'] = value ;
212             fieldSpecs['orig_value'] = orig_value ;
213             fieldSpecs['tablename'] = parent_table ;
214
215             jArray.push(fieldSpecs) ;
216           }
217         }
218
219       break ;
220       //
221         ------------------------------------------------------------------------------
222         -------
223       default:
224         console.log('SOMETHING ELSE  Type: ' + input.attr('type') + '    Name: ' +
225           input.attr('name') + '    Value: ' + input.val());
226
227       }  // EO Switch
228
229     }   );  // EO Form Inputs .each()
230
231   //
232     ------------------------------------------------------------------------------
233     ---------
234   // When inserting a note that is not tied to any table, the field `note_src_table_id` is
235     meaningless.
236   // What we want to do is use that variable to contain the # of GeneralNote s inserted.
237
238   if ( inserting_a_general_note === 1 )
239   {
240     var php_params = 'job=SearchUsing&formToProcess=GeneralNote&task=GetCount' ;
241     var sql_data ;
242
243     $.ajax({
244     url: '/cmdb/PHP/do_FormCommit_work.php',
245     method:     "GET" ,
246     cache:      false ,
247     data:       php_params  ,
248     dataType:   'JSON'
249     }).done(function(data)
250     {
```

```
      console.log('AJAXed to do_FormCommit_work.php for '+action+'TableRecs() on
        GeneralNote to get COUNT(*), data.status='+data.status) ;

      sql_data = (typeof data === 'string') ? JSON.parse(data) : data ;

      var fieldSpecs = {} ;
      var parent_table = 'Note' ;

      fieldSpecs['name'] = 'note_src_table_id' ;
      fieldSpecs['type'] = 'text' ;
      fieldSpecs['value'] = Number(sql_data.count_of_general_notes) + 1 ;
      fieldSpecs['orig_value'] = 0 ;
      fieldSpecs['tablename'] = 'Note' ;

alert('name='+fieldSpecs['name']+'  type='+fieldSpecs['type']+'
  value='+fieldSpecs['value']+'  orig_value='+fieldSpecs['orig_value']+'
  tablename='+fieldSpecs['tablename']) ;

      jArray.push(fieldSpecs) ;

      console.log('Type: ' + fieldSpecs['type'] + '    Name: ' + fieldSpecs['name'] + '
        Value: ' + fieldSpecs['value'] + '   OrigValue: '+fieldSpecs['orig_value']+'
        tablename: '+parent_table );

      // Needed below.  Just don't have the #table_row value here.
$(formID+' p').append('<input type="hidden" ID="table_row" name="table_row"
  value="'+fieldSpecs['value']+'" />') ;

    }      ) ;
 }

 //
   ------------------------------------------------------------------------------
   ---------
 // Gotta get a little messy here now.
 // When dropdowns are hidden and in their place are text fields, both are picked up
   here.
 // We want the fieldnameTF (text field) version's of the data, so we have to find those
   array
 // elements, get their value, set that into the DD field's value, then remove the 'TF'
   element.

 var fTFname, fTFnamelen, fname, fTFvalue ;
 var fToRemove = [] ;

 for (var f=3; f<jArray.length; f++)
 {
   fTFname = jArray[f]['name'] ;
   fTFnamelen = fTFname.length ;
```

```
301     if ( fTFname.substring(fTFnamelen-2) === 'TF' )
302     {
303       fname = fTFname.substring(0,fTFnamelen-2) ;
304       fTFvalue = jArray[f]['value'] ;             //console.log('fname='+fname+'
305         fTfvalue='+fTFvalue) ;
306
307       for (var fnbr=3; fnbr<jArray.length; fnbr++)
308       {
309         if ( jArray[fnbr]['name'] == fname )
310         {
311           jArray[fnbr]['value'] = fTFvalue ;  //console.log('fnbr='+fnbr+'
312             jArray[fnbr][name]='+jArray[fnbr]['name']+'
313             jArray[fnbr][value]='+jArray[fnbr]['value']) ;
314           $('a:link #rebuild_dropdowns').css('color', '#FF0000') ;
315         }
316       }
317       fToRemove.push(f) ;
318       //console.log('f='+f+'  jArray[f][name]='+jArray[f]['name']) ;
319     }
320   }
321
322   if ( fToRemove.length)
323   {
324     for (var f=fToRemove.length-1; f>=0; f--)
325     {
326       // console.log('f='+f+'  fToRemove[f]='+fToRemove[f]);
327       jArray.splice(fToRemove[f],1) ;
328     }
329   }
330
331
332   //
333     ------------------------------------------------------------------------------------
334     ---------
335
336   var nFields = jArray.length - 3 ;  // 3 Objects stuffed on < these fields
337   jArray[0]['nFields'] = nFields ;
338
339   //
340     ====================================================================================
341     =========
342   // Now must get from #invisible DIV the current PK values
343
344   $('#invisible .other_table_ids ')
345    .each(
346     function()
347     {
348       var input = $(this);
349       var name = input.attr('name') ;
350       var value = input.val() ;
```

```
351
352       var pkValues = {} ;
353       pkValues['name']  = name  ;
354       pkValues['value'] = value ;
355
356       jArray.push(pkValues) ;
357     }   ) ;
358
359     // Now get the as-loaded PK values
360
361   $('#invisible .otid_orig ')
362    .each(
363     function()
364     {
365       var input = $(this);
366       var name = input.attr('name') ;
367
368       if ( $('#'+name).val() !== 'undefined' )
369       {
370         var pkValues = {} ;
371         pkValues['name']  = name  ;
372         pkValues['value'] = $('#'+name).val() ;
373
374         jArray.push(pkValues) ;
375       }
376     }   ) ;
377
378   //
       ------------------------------------------------------------------------------------------
       ---------
379   // Let's not forget, though not an 'other' table ID, it is a PK we definitely need to
       know about for
380   // our purposes here.
381   // Yes, it's already there on the parent form, hidden, but let's also put in the format
       of other 'other table ids'.
382
383   var pkValues = {} ;
384   pkValues['name']  = table_info[tablename]  ;
385   pkValues['value'] = $('form #table_row').val() ;
386   jArray.push(pkValues) ;
387
388   if ( updating_a_note || tablename === 'ModemConfig')
389   {
390     pkValues['name']  = 'note_id'  ;
391     pkValues['value'] = $('form #note_table_row').val() ;
392     jArray.push(pkValues) ;
393   }
394
395   //
       ==========================================================================================
396
```

```
401       =======
402     // Prepare for POSTing
403
404     var jsonString = JSON.stringify(jArray) ;
405
406     //
          ------------------------------------------------------------------------------------
407       -------
408     // On return from this Ajax invocation the DB will be revised w/ updated/new values
409
410     console.log('AJAXing to do_FormCommit_work.php with json array for '+action+'.');
411
412     $.ajax({
413       url: '/cmdb/PHP/do_FormCommit_work.php', //   do_FormCommit_work  bounce_post
414       method:     "POST" ,
415       data:       {posted:jsonString}
416       }).done(function(data)
417       {
418         console.log('Successfully performed '+action.toUpperCase()+'.') ;
419
420         var results = (typeof data === 'string') ? JSON.parse(data) : data ;
421
422         if ( !updating_a_note && !inserting_a_reminder )
423         {
424           manage_form_fields(tablename, 'backup') ;  // This installs the visible fields'
425             invisible bu counterparts, class="backup"
426         }
427
428         // After a commit manage_form_fields will take care of all normal visible fields in
429           the form,
430         // but what's left are:  The one/two hidden #table_row and maybe #note_table_row,
431           AND the two
432         // other classes we have to address now.  We have to fix the dropdowns that point to
433           other tables.
434         // The DD changes did upon change, update the .other_table_ids class of values, and
435           those were
436         // used to note any changes from the as-loaded values found in class .otid_orig.
437         // NOW those .otid_orig values have to be set so future DD changes will be noted.
438
439         $('.otid_orig').each(function()
440         {
441           var otid_name =  $(this).attr('name') ;
442           var name = otid_name.substr(10) ;
443
444           var curr_value = $('#'+name).val() ;
445
446           $(this).val(curr_value) ;
447
448         }) ;
449
450
```

```
      if ( updating_a_note ) { cancelNote() ;  }
      if ( inserting_a_reminder ) { cancelReminder() ;  }

      alert(results) ;

    }       ) ;

}



//
  ==========================================================================================
  =========
//
  ==========================================================================================
  =========
// FormSearchScripts.js calls this function after loading new values using jQuery into
  form page

function manage_form_fields(tablename, operation, fields)
{
  // Purpose is to manage values in a form.
  // Once a form is initially loaded, this function is called.
  // Use 'backup' operation to make input backups in hidden DIV installing as-loaded
    values.
  // Use 'update' with the passed fields array to just backup those fields

  if ( fields === 'undefined' ) { fields = [] ; }


  var php_params = "job=GetFormSpecs&formToProcess="+tablename ;
  // $('#debug').html(php_params) ;
  console.log('Arrived in manage_form_fields("'+tablename+'", "'+operation+'")');

  operation = operation.toLowerCase() ;

  //
    =*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*
    =*=*=*=*=
  // Go pay a visit to PHP to get the specifications on the form currently loaded
  // DO AJAX with Success callback to handle the variables

  $.ajax({
    url: '/cmdb/PHP/do_FormCommit_work.php',
    method:     "GET" ,
    cache:      false ,
    data:       php_params  ,
    dataType:   'JSON'
    }).done(function(data)
```

```
501       {
502
503         var formSpecs = (typeof data === 'string') ? JSON.parse(data) : data ;
504
505         console.log('AJAXed to do_FormCommit_work.php for
506           manage_form_fields('+tablename+', '+operation+'), data.status='+data.status) ;
507
508         // Back from our visit now, so let's march through that list of fields getting
509         //   their as-loaded values
510
511         // If this is empty, then all name="orig_fieldname" and  class="backup" inputs
512           will be cleared
513         // and repopulated with current form values, but in the case of the Note table, we
514           only want
515         // to clear that sub-form's value, not remove all of the originally as-loaded
516           values for the
517         // other form fields.
518
519         var fields_to_clear_and_set = [] ;
520         if ( tablename === 'Note' )
521         {
522           fields_to_clear_and_set.push('note_src_table') ;
523           fields_to_clear_and_set.push('note_src_table_id') ;
524           fields_to_clear_and_set.push('note_date_entered') ;
525           fields_to_clear_and_set.push('note_date_effective') ;
526           fields_to_clear_and_set.push('info_co_id') ;
527           fields_to_clear_and_set.push('note') ;
528         }
529         else if ( tablename === 'Reminder' )
530         {
531           fields_to_clear_and_set.push('reminder_ref_table') ;
532           fields_to_clear_and_set.push('reminder_clue') ;
533           fields_to_clear_and_set.push('reminder_for') ;
534           fields_to_clear_and_set.push('date_rmdr_created') ;
535           fields_to_clear_and_set.push('date_to_remind') ;
536           fields_to_clear_and_set.push('date_of_importance') ;
537           fields_to_clear_and_set.push('reminder_status') ;
538           fields_to_clear_and_set.push('reminder_detail') ;
539         }
540         else if ( Array.isArray(fields) && fields.length > 0 )
541         {
542           fields_to_clear_and_set = fields ;
543         }
544
545         //
546           ------------------------------------------------------------------------------
547           ---------
548         switch (operation)
549         {
550           case 'backup':  //
```

```
551             ------------------------------------------------------------------------
552
553             var form_values = get_form_values(formSpecs, 'current') ; //
554               form_values[property]=value  OBJECT
555
556             insert_new_html5_inputs(formSpecs, form_values, fields_to_clear_and_set) ;
557
558           break ;
559
560           case 'update':  // ------------ NOT USED I BELIEVE
561             ---------------------------------------
562
563             var form_values = get_form_values(formSpecs, 'current') ;
564             // Make on the form a hidden input of similar
565             //   name to hold each of these initial values
566
567             insert_new_html5_inputs(formSpecs, form_values, fields) ;
568 //            $('#debug').html('') ;
569
570           break ;
571
572         }
573
574
575       }) ;
576
577 }
578 //
    ==========================================================================================
    =========
579 //
    ==========================================================================================
    =========
580 // Function manage_form_fields() calls this to return with an assoc array of field names &
    values
581
582 function get_form_values(formSpecs, get_which)
583 {
584
585   var nfields = Object.keys(formSpecs).length ;
586
587   var name, type, subtype, temp ;
588   var values = {} ;
589
590   for (var f=0; f<nfields; f++)
591   {
592     name = formSpecs[f]['name'] ;
593     type = formSpecs[f]['type'] ;
594     subtype = formSpecs[f]['subtype'] ;
595
596
597
598
599
600
```

```
601
602     if ( get_which === 'backup' )
603     {
604       name = 'orig_'+name ;
605       values[name] = $( 'form input[name="'+name+'"]' ).val()  ;
606     }
607     else
608     {
609
610       if (type === '<select>')
611       { // The below works only because for <select> elements I included ID= with value =
612         'name' value
613
614         values[name] = $('form #'+name+" option:selected").val()  ;
615       }
616       else if ( type === '<textarea>' )
617       {
618         values[name] = $( 'form textarea[name="'+name+'"]' ).val().replace(/"/g,
619           "&quot;").replace(/'/, "&apos;")  ; // Using .text() messes w/ what is returned.
620           Go figure.
621       }
622       else
623       {
624         if ( subtype === 'checkbox' )
625         {
626           temp=0;
627           if ( $( ' form input[name="'+name+'"]' ).prop('checked') === true ) { temp=1 ; }
628           values[name] = temp  ;
629         }
630         else
631         {
632           values[name] = $( 'form input[name="'+name+'"]' ).val() ;
633         }
634       }
635     }  // EO backup or not
636
637     if ( name === 'wireless_no' && values[name].indexOf('(') !== -1 )
638     {
639       values[name] = values[name].strip_phone_format() ;
640     }
641     if ( name === 'simm_identifier' || name === 'IMEI' || name === 'IMSI' )
642     {
643       values[name] = values[name].replace(' ', '') ;
644     }
645     // console.log(name+'   '+values[f]) ;
646   }
647
648   return values ;
649
650 }
```

```
651 //
652   ================================================================================================
653   =========
654 //
655   ================================================================================================
656   =========
657 // Function manage_form_fields() calls this to insert as-loaded values for later
658   comparison
659
660 function insert_new_html5_inputs(formSpecs, form_values, which_fields)
661 {
662
663   if ( which_fields.length === 0 )
664   {
665     $('#invisible .backup').remove() ;  // Remove all input elements in that DIV with that
666       class
667   }
668
669   var nfields = Object.keys(formSpecs).length ;
670   var name, subtype, value ;
671
672   for (var f=0; f<nfields; f++)
673   {
674     name = formSpecs[f]['name'] ;
675     subtype = formSpecs[f]['subtype'] ;
676
677     value = form_values[name] ;
678
679     if ( subtype !== 'hidden' && ( which_fields.length === 0 || which_fields.indexOf(name)
680       !== -1 ) )
681     {
682       $('#invisible [name=orig_'+name+']').remove() ;
683       if ( formSpecs[f]['type'] !== '<textarea>' )
684       { $('#invisible').append('<input type="text" name="orig_'+name+'" ID="orig_'+name+'"
685         class="backup" value="'+value+'" />'+"\n") ; }
686       else
687       { $('#invisible').append('<textarea name="orig_'+name+'" ID="orig_'+name+'"
688         class="backup">'+value+'</textarea>'+"\n") ; }  // Want to preserve line feeds
689
690     }
691
692   }
693 }
694 //
695   ================================================================================================
696   =========
697 //
698   ================================================================================================
699   =========
700 // This commitAircraftModemSwap function is markedly different than other INSERT/DELETEs.
```

```
// Rather here we just have to point the 'off' Modem to Aircraft with ID=1 => Unassigned, and
// point the 'on' Modem to the Aircraft.

function commitAircraftModemSwap()
{
  var table_row_Aircraft, table_row_ModemOff, table_row_ModemOn ;
  var php_params = "job=commitAircraftModemSwap" ;
  var result ;

  table_row_Aircraft = $("#form_4ModemSwap_Aircraft p #table_row_Aircraft").val() ;
  table_row_ModemOff = $("#form_4ModemSwap_ModemOff p #table_row_ModemOff").val() ;
  table_row_ModemOn  = $("#form_4ModemSwap_ModemOn  p #table_row_ModemOn").val() ;
  if ( table_row_Aircraft === '' || isNaN(table_row_Aircraft) || table_row_ModemOff === ''
    || isNaN(table_row_ModemOff) || table_row_ModemOn === '' || isNaN(table_row_ModemOn) )
  {
    alert("Please use [Search] to populate the Aircraft and Modem forms before clicking
      [Commit].")
    throw 'Missing A/C or Modem(s) IDs so cannot perform swap.' ;
  }

  //
    ================================================================================
    =========

  php_params +=
    '&aircraftID='+table_row_Aircraft+'&modemOffID='+table_row_ModemOff+'&modemOnID='+
    table_row_ModemOn ;
  $('#debug').html(php_params) ;

  $.ajax({
    url: '/cmdb/PHP/do_FormCommit_work.php',
    method:     "GET" ,
    cache:      false ,
    data:       php_params  ,
    dataType:   'JSON'
    }).done(function(data)
    {
      console.log('AJAXed to do_FormCommit_work.php for commitAircraftModemSwap() on
        ModemConfig rows, data.status='+data.status) ;

      result = data ; //(typeof data === 'string') ? JSON.parse(data) : data ;  // e.g.:
        table_info['ModemConfig'] => modem_config_id

      alert(result) ;

      $('#4ModemSwapSubmit').prop('disabled', true) ;  //

    }      ) ;

```

```
751 }
752
```