# ChangeParametersMultipleElementFlatInput Function

The ChangeParametersMultipleElementFlatInput function allows you to set parameters for multiple objects of the same type in a case loaded into the Simulator Automation Server. This function is very similar to the ChangeParametersMultipleElement, but uses a single dimensioned array of values as input instead of a multi-dimensioned array of arrays. The advantage of this function is that it is much faster to change multiple elements with a single function call than it is to repeatedly call ChangeParametersSingleElement multiple times. An additional advantage over ChangeParametersMultipleElement is that you can still take advantage of the speed improvement, even if the programming language you are using does not support multi-dimensioned arrays.

**Function Prototype**

**ChangeParametersMultipleElementFlatInput(ObjectType, ParamList, NoOfObjects, ValueList)**

Parameter Definitions

**ObjectType : String** The type of object you are changing parameters for.

**ParamList : Variant** A variant array storing strings (COM Type BSTR). This array stores a list of PowerWorldâ object field variables, as defined in the section on PowerWorld Object Fields. The ParamList must contain the key field variables for the specific device, or the device cannot be identified.

**NoOfObjects** You must pass an integer number of devices that are passing values for. SimAuto will automatically check that the number of parameters for each device (counted from ParamList) and the number of objects integer correspond to the number of values in value list (counted from ValueList.)

**ValueList : Variant** A variant array storing a list of variants. Value list can be an array with many values, as it is a single dimensioned array of all values for all devices that are being changed. The structure of the ValueList array is such that all of the parameters for the first object are listed first, then all parameters for the second object, and so on. The parameters must be in the same order as given in ParamList. In other words, your array would look like:

ValueList = Array(Obj1Param1, Obj1Param2, Obj2Param1, Obj2Param2,Obj3Param1, …, ObjNParam1, ObjNParam2)

Output

ChangeParametersMultipleElementFlatInput only returns the first element in Output, the

error string.

Notes

If the number of parameters given in ParamList multiplied by the number of objects passed does not equal the total number of values in ValueList, SimAuto will abort the function call.

---

# ChangeParametersMultipleElement Function

The ChangeParametersMultipleElement function allows you to set parameters for multiple objects of the same type in a case loaded into the Simulator Automation Server. This function is very similar to the ChangeParametersSingleElement, but allows for modifying multiple elements with a single function call. The advantage of this function is that it is much faster to change multiple elements with a single function call than it is to repeatedly call ChangeParametersSingleElement multiple times.

**Function Prototype**

**ChangeParametersMultipleElement(ObjectType, ParamList, Values)**

Parameter Definitions

**ObjectType : String** The type of object you are changing parameters for.

**ParamList : Variant** A variant array storing strings (COM Type BSTR). This array stores a list of PowerWorldâ object field variables, as defined in the section on PowerWorld Object Fields. The ParamList must contain the key field variables for the specific device, or the device cannot be identified.

**ValueList : Variant** A variant array storing arrays of variants. This is the difference between the multiple element and single element change parameter functions. This array stores a list of arrays of values matching the fields laid out in ParamList. You construct ValueList by creating an array of variants with the necessary parameters for each device, and then inserting each individual array of values into the ValueList array. SimAuto will pick out each array from ValueList, and calls ChangeParametersSingleElement internally for each array of values in ValueList.

Output

ChangeParametersMultipleElement only returns the first element in Output, the error string.

Notes

The ParameterList and each array of values stored in ValueList must be the same size, as each parameter must have a corresponding value to be assigned.

---

# ChangeParametersSingleElement Function

The ChangeParametersSingleElement function allows you to set a list of parameters for a single object in a case loaded into the Simulator Automation Server. In addition to changing parameters for objects, this function can also be used to set options for some of the Simulator tools, such as ATC and OPF. This function is identical in setup to the GetParametersSingleElement function, with the exception that the Values array must contain a value for each field variable given in the ParamList array.

**Function Prototype**

**ChangeParametersSingleElement(ObjectType, ParamList, Values)**

Parameter Definitions

**ObjectType : String** The type of object you are changing parameters for.

**ParamList : Variant** A variant array storing strings (COM Type BSTR). This array stores a list of PowerWorldâ object field variables, as defined in the section on PowerWorld Object Fields. The ParamList must contain the key field variables or label for the specific device, or the device cannot be identified.

**Values : Variant** A variant array storing variants. This array can store any type of information (integer, string, etc.) in each array position. A value should be passed for each field variable given in the ParamList. The Values array must contain the key field values for the specific device, or the device cannot be identified.

Output

ChangeParametersSingleElement only returns the first element in Output, the error string.

Notes

The ParameterList and Values arrays must be the same size, as each parameter must have a corresponding value to be assigned.

---

# CloseCase Function

The CloseCase function is used to close a load flow case loaded in the Simulator Automation Server. This function should be called at some point after the OpenCase function.

Function Prototype

**CloseCase()**

Parameter Definitions

No parameters are passed.

Output

CloseClase returns only one element in Output—any errors which may have occurred when attempting to close the case.

---

# GetFieldList Function

The GetFieldList function is used to find all fields contained within a given object type.

Function Prototype

**GetFieldList(ObjectType)**

Parameter Definitions

**ObjectType : String** The type of object for which the fields are requested.

Output

GetFieldList returns two elements of the Output array. The first element, as with the other functions, returns any errors that might have occurred. The second element of the Output array contains an n x 4 array of fields. The layout of this array is virtually identical to the output obtained by going to **Help > Export Case Object Fields** from the main menu of Simulator. The first column, corresponding to the (n,0) column in the field array, specifies which fields are key fields for the object. The second column, (n,1), contains the internal name of the field. The third column, (n,2), contains the type of data stored in the string (e.g. String, Integer, Real). The fourth column, (n,3), contains the display-friendly name of the field.

---

# GetParametersMultipleElementFlatOutput Function

This function operates the same as the GetParametersMultipleElement function, only with one notable difference. The values returned as the output of the function are returned in a single-dimensional vector array, instead of the multi-dimensional array as described in the GetParametersMultipleElement topic. The function returns the parameter values for the device type requested, and the number of values returned depends on the ParamList and any advanced filter that impacts the number of devices returned.

The format of the output array is the following:

[errorString, NumberOfObjectsReturned, NumberOfFieldsPerObject, Ob1Fld1, Ob1Fld2, …, Ob(n)Fld(m-1), Ob(n)Fld(m)]

The data is thus returned in a single dimension array, where the parameters NumberOfObjectsReturned and NumberOfFieldsPerObject tell you how the rest of the array is populated. Following the NumberOfObjectsReturned parameter is the start of the data. The data is listed as all fields for object 1, then all fields for object 2, and so on. You can parse the array using the NumberOf… parameters for objects and fields.

---

# GetParametersMultipleElement Function

The GetParametersMultipleElement function is used to request the values of specified fields for a

set of objects in the load flow case. The function can return values for all devices of a particular type, or can return values for only a list of devices of a particular type based on an advanced filter defined for the loaded case.

Function Prototype

**GetParametersMultipleElement(ObjectType, ParamList, FilterName)**

Parameter Definitions

**ObjectType : String** The type of object you are changing parameters for.

**ParamList : Variant** A variant array storing strings. This array stores a list of PowerWorldâ object field variables, as defined in the section on PowerWorld Object Fields. The ParamList must contain the key field variables or label for the specific device, or the device cannot be identified. The remaining field variables in the array define which values to retrieve from Simulator.

**FilterName : String** The name of an advanced filter defined in the load flow case open in the Simulator Automation Server. If no filter is desired, then simply pass an empty string. If a filter name is passed but the filter cannot be found in the loaded case, the server will default to returning all objects in the case of type ObjType.

Output

GetParametersMultipleElement returns a set of nested arrays containing the parameter values for the device type requested. The number of arrays of values returned depends on the number of fields in ParamList.

The Output structure of GetParametersMultipleElement is shown in the following figure.

As you can see, to access the first parameter value for the first device, Output[1][0][0] would be the correct array index. For example, the bus number for the first bus would be stored at Output[1][0][0] after calling Output = GetParametersMultipleElement('Bus',fieldarray, ''), and assuming that we have fieldarray = Array(pwBusnum, pwBusName).

---

# GetParametersSingleElement Function

The GetParametersSingleElement function is used to request the values of specified fields for a particular object in the load flow case. For returning field values for multiple objects, you can use a loop to make repeated calls to the GetParametersSingleElement function, and pass the object and desired field information for each object. This function is identical in setup to the ChangeParameters function, with the exception that the Values array will be updated with the values for the field variables defined in ParamList.

Function Prototype

**GetParametersSingleElement(ObjectType, ParamList, Values)**

Parameter Definitions

**ObjectType : String** The type of object you are changing parameters for.

**ParamList : Variant** A variant array storing strings. This array stores a list of PowerWorldâ object field variables, as defined in the section on PowerWorld Object Fields. The ParamList must contain the key field variables or label for the specific device, or the device cannot be identified. The remaining field variables in the array define which values to retrieve from Simulator.

**Values : Variant** A variant array storing variants. This array can store any type of information (integer, string, etc.) in each array position. Values must be passed for the key field variables in ParamList, in the same array position. The remaining field positions in the Values array should be set to zero.

Output

GetParametersSingleElement returns both the first element in Output—containing any errors occurring during execution of the function—and a second element in Output. The second element returned in the Output structure is a one dimensional array containing the values corresponding to the fields specified in ParamList.

The Output structure of GetParametersSingleElement is shown in the following figure.

---

# GetParameters Function

This function is maintained in versions of Simulator later than version 9 for compatibility with

Simulator version 9. This function is replaced by GetParametersSingleElement. See also GetParametersMultipleElement.

---

# GetSpecificFieldList Function

The GetSpecificFieldList function is used to return identifying information about specific fields used by an object type.

Function Prototype

**GetSpecificFieldList(ObjectType, FieldList)**

Parameter Definitions

**ObjectType : String** The type of object for which fields are requested.

**FieldList : Variant** A variant array storing strings. This array stores a list of object field variables, as defined in the section on PowerWorld Object Fields. Specific variablenames along with location numbers can be specified. To return all fields using the same variablename, use "*variablename*:ALL" instead of the location number that would normally appear after the colon. If all fields should be returned, a single parameter of "ALL" can be used instead of specific variablenames.

Output

GetSpecificFieldList returns a two dimensional variant array. The first dimension, Output(0) contains the error string. The second dimension, Output(1), is a rectangular array indexed from 0 with a size of n x 4 where n is the number of fields that are returned. The specific information that is returned is as follows:

Output(1)(n,0) - variablename:location

Output(1)(n,1) - field (this is the identifier that is displayed when looking through the list of available fields for an object in a case information display in the GUI)

Output(1)(n,2) - column header (this is the column header that appears in a case information display in the GUI)

Output(1)(n,3) - field description

# GetSpecificFieldMaxNum Function

The GetSpecificFieldMaxNum function is used to return the maximum number of a fields that use a particular variablename for a specific object type.

Function Prototype

**GetSpecificFieldMaxNum(ObjectType, Field)**

Parameter Definitions

**ObjectType : String** The type of object for which information is being requested.

**Field : String** The variablename for which the maximum number of fields is being requested. This should just be the variablename and should exclude the location number that can be included to indicate different fields that use the same variablename, i.e. do not include the colon and number that can be included when identifying a field.

Output

An integer that specifies the maximum number of fields that use the same variablename for a particular object type. Fields are identified in the format variablename:location when multiple fields use the same variablename. The output indicates the maximum number that the location can be. Generally, fields are identified starting from 0 and going up to the maximum number, but keep in mind that values within this range might be skipped and not used to indicate valid fields.

---

# ListOfDevicesAsVariantStrings Function

This function operates the same as the ListOfDevices function, only with one notable difference. The values returned as the output of the function are returned as Variants of type String. The ListOfDevices function was errantly released returning the values strongly typed as Integers and Strings directly, whereas all other SimAuto functions returned data as Variants of type String. This function was added to also return the data in the same manner. This solved some compatibility issues with some software languages.

---

# ListOfDevicesFlatOutput Function

This function operates the same as the ListOfDevices function, only with one notable difference. The values returned as the output of the function are returned in a single-dimensional vector array, instead of the multi-dimensional array as described in the ListOfDevices topic. The function returns the key field values for the device, typically in the order of bus number 1, bus number 2 (where applicable), and circuit identifier (where applicable). These are the most common key fields, but some object types do have other key fields as well.

The format of the output array is the following:

[errorString, NumberOfObjectsReturned, NumberOfFieldsPerObject, Ob1Fld1, Ob1Fld2, …, Ob(n)Fld(m-1), Ob(n)Fld(m)]

The data is thus returned in a single dimension array, where the parameters NumberOfObjectsReturned and NumberOfFieldsPerObject tell you how the rest of the array is populated. Following the NumberOfObjectsReturned parameter is the start of the data. The data is listed as all fields for object 1, then all fields for object 2, and so on. You can parse the array using the NumberOf… parameters for objects and fields.

---

# ListOfDevices Function

The ListOfDevices function is used to request a list of objects and their key fields from the Simulator Automation Server. The function can return all devices of a particular type, or can return only a list of devices of a particular type based on an advanced filter defined for the loaded case. This function is best used in conjunction with a looping procedure and the ChangeParameters or GetParametersSingleElement functions to process a group of devices.

Function Prototype

**ListOfDevices(ObjType, filterName)**

Parameter Definitions

**ObjType : String** The type of object for which you are acquiring the list of devices.

**FilterName : String** The name of an advanced filter defined in the load flow case open in the Simulator Automation Server. If no filter is desired, then simply pass an empty string. If the filter cannot be found, the server will default to returning all objects in the case of type ObjType.

Output

ListOfDevices returns a set of nested arrays containing the key field values for the device type requested. The number of arrays of values returned depends on the object type selected. For instance, buses have only one key field (the bus number) so calling ListOfDevices for buses will return only one array of values—the bus numbers. On the other hand, calling ListOfDevices for branches will return three arrays of values—the "From" bus, "To" bus, and ID—for each branch in the case meeting the specified filter.

The arrays containing the key field values for each device are arranged as shown in the following figure.

As you can see, to access the first key field value for the first device, Output[1][0][0] would be the correct array index. For example, the bus number (which is the bus key field) for the first bus would be stored at Output[1][0][0] after calling Output = ListOfDevices('Bus', '').

One unique limitation of the ListOfDevices function from other SimAuto functions is that this is the only function that returns the output as strongly typed variables. The bus numbers are always returned as Long Integers, and the Circuit ID values are returned as strings. This was actually an oversight during the design of SimAuto. In all other SimAuto functions, the values are returned as Variant types, with each value within the variant being a string. This was the intended operation for this function as well. Since the Automation Server interface was released with the errant inclusion of the ListOfDevices function, it could not be modified. Therefore, another function, ListOfDevicesAsVariantStrings, has been created. This function returns all values in variant variables, with each as a string within the variant type.

---

# LoadState Function

LoadState is used to load the system state previously saved with the SaveState function. Note that LoadState will not properly function if the system topology has changed due to the addition or removal of the system elements.

Function Prototype

**LoadState()**

Parameter Definitions

No parameters are passed.

Output

LoadState returns only one element in Output—any errors which may have occurred when attempting to execute the function.

---

# OpenCase Function

The OpenCase function will load a PowerWorldâ Simulator load flow file into the Simulator Automation Server. This is equivalent to opening a file using the **File > Open Case** menu option in Simulator.

Function Prototype

**OpenCase(FileName)**

Parameter Definitions

**FileName : String** The name of the PowerWorldâ Simulator case file to be loaded into the Simulator Automation Server. This string includes the directory location and full file name.

Output

OpenCase returns only one element in Output—if the file cannot be found or an error occurs while reading the file.

---

# ProcessAuxFile Function

The ProcessAuxFile function will load a PowerWorldâ Auxiliary file into the Simulator Automation Server. This allows you to create a text file (conforming to the PowerWorldâ Auxiliary file format) that can list a set of data changes and other information for making batch changes in Simulator

Function Prototype

**ProcessAuxFile(FileName)**

Parameter Definitions

**FileName : String** The name of the PowerWorldâ Auxiliary file to be loaded into the Simulator Automation Server. This string includes the directory location and full file name.

Output

ProcessAuxFile returns only one element in Output—any errors which may have occurred when attempting to load the file.

---

# RunScriptCommand Function

The RunScriptCommand function is used to execute a list of script statements. The script actions are those included in the script sections of the Auxiliary Files. If an error occurs trying to run a script command, an error will be returned through EString.

Function Prototype

**RunScriptCommand(Statements)**

Parameter Definitions

**Statements : String** The block of script actions to be executed. Each script statement must end in a semicolon. The block of script actions should **not** be enclosed in curly braces.

Output

RunScriptCommand returns only one element in Output—any errors which may have occurred when attempting to load or run the auxiliary file.

---

# SaveCase Function

The SaveCase function is used to save a case previously loaded in the Simulator Automation Server using the OpenCase function. The function allows you to specify a file name and a format for the save file.

Function Prototype

**SaveCase(FileName, EString, FileType, Overwrite)**

Parameter Definitions

**FileName : String** The name of the file you wish to save as, including file path.

**FileType : String** A string indicating the format of the written case file. An empty string will return an error. The following list is the currently supported list of string identifiers and the file types they represent.

"PTI23" PTI version 23 (raw)
"PTI24" PTI version 24 (raw)
"PTI25" PTI version 25 (raw)
"PTI26" PTI version 26 (raw)
"PTI27" PTI version 27/28 (raw)
"PTI29" PTI version 29 (raw)
"PTI30" PTI version 30 (raw)
"PTI31" PTI version 31 (raw)
"GE" GE PSLF (epc)
"IEEE" IEEE common format (cf)
"PWB70" PowerWorld Binary version 7.0 (pwb)
"PWB" PowerWorld Binary (most recent) (pwb)

**Overwrite : Boolean** A Boolean value which indicates whether to overwrite a file if FileName already exists. If Overwrite is set to false and the file specified by FileName already exists, SaveCase will return an error message and do nothing to the file.

Output

SaveCase returns only one element in Output—any errors which may have occurred when attempting to save the case.

---

## SaveState Function

SaveState is used to save the current state of the power system. This can be useful if you are interested in comparing various cases, much as the Difference Flows feature works in the Simulator application.

Function Prototype

**SaveState()**

Parameter Definitions

No parameters are passed.

Output

SaveState returns only one element in Output—any errors which may have occurred when attempting to execute the function.

---

## SendToExcel Function

The SendToExcel function can be called to send data from the Simulator Automation Server to an Excel spreadsheet. The function is flexible in that you can specify the type of object data you want to export, an advanced filter name for a filter you want to use, and as many or as few field types as desired that are supported by the type of object. The first time this function is called, a new instance of Excel will be started, and the data requested will be pasted to a new sheet. For each subsequent call of this function, the requested data will be pasted to a new sheet within the same workbook, until the workbook is closed.

Function Prototype

**SendToExcel(ObjectType, FilterName, FieldList)**

Parameter Definitions

**ObjectType : String** A string describing the type of object for which your are requesting data.

**FilterName : String** The name of an advanced filter which was previously defined in the case before being loaded in the Simulator Automation Server. If no filter is desired, then simply pass an empty string. If a filter name is passed but the filter cannot be found in the loaded case, no filter is used.

**FieldList : Variant** This parameter must either be an array of fields for the given object or the string "all". As an array, FieldList contains an array of strings, where each string represents an object field variable, as defined in the section on PowerWorld Object Variables. If, instead of an array of strings, the single string "all" is passed, the Simulator Automation Server will use predefined default fields when exporting the data.

Output

SendToExcel returns only one element in Output—any errors which may have occurred when attempting to execute the function.

---

# TSGetContingencyResults Function

TSGetCongencyResults function

The TSGetContingencyResults function is used to read transient stability results into an external program (i.e. Matlab or VB) using SimAuto.

This function is analogous to the script command TSGetResults, where rather than saving out results to a file, the results are passed back directly to the SimAuto COM object and may be further processed by an external program. As with TSGetResults, this function should only be used after the simulation is run (for example, use this after running script commands TSSolveAll or TSSolve).

Function Prototype

**TSGetContingencyResults(CtgName, ObjFieldList, StartTime, StopTime)**

Parameter Definitions

**CtgName : String** The contingency to obtain results from. Only one contingency be obtained at a time.

**ObjFieldList: Variant** A variant array of strings which may contain plots, subplots, or individual object/field pairs specifying the result variables to obtain.

**StartTime : String** The time in seconds in the simulation to begin retrieving results. If not specified, the start time of the simulation is used.

**StopTime : String** The time in seconds in the simulation to stop retrieving results. If not specified, the end time of the simulation is used.

Output

The SimAuto output for this function is a Variant which contains three levels. Output(0) displays an error message, if any. Output(1) displays the header which describes the variables that were saved out. Output(2) displays the time series data (numerical results from simulation), corresponding to the variables described in the header. MATLAB indexing begins at 1 instead of 0, so add one to all the indices shown when using MATLAB.

---

# WriteAuxFile Function

The WriteAuxFile function can be used to write data from the case in the Simulator Automation Server to a PowerWorldâ Auxiliary file. The function is flexible in that you can specify the type of object data you want to export, an advanced filter name for a filter you want to use, and as many or as few field types as desired that are supported by the type of object. In addition, you can specify a new file name for each call to WriteAuxFile, or you can specify the same file name and append the data to the file.

Function Prototype

**WriteAuxFile(FileName, FilterName, ObjectType, EString, ToAppend, FieldList)**

Parameter Definitions

**FileName : String** The name of the PowerWorldâ Auxiliary file you wish to save.

**FilterName : String** The name of an advanced filter which was previously defined in the case before being loaded in the Simulator Automation Server. If no filter is desired, then simply pass an empty string. If a filter name is passed but the filter cannot be found in the loaded case, no filter is used.

**ObjectType : String** A string describing the type of object for which your are requesting data.

**ToAppend : Boolean** If you have given a file name of an auxiliary file that already exists, then the file will either be appended to or overwritten according to the setting of this parameter. If ToAppend is *False* and the file already exists, WriteAuxFile will return an error message and do nothing to the file.

**FieldList : Variant** This parameter must either be an array of fields for the given object or the string "all". As an array, FieldList contains an array of strings, where each string represents an object field variable, as defined in the section on PowerWorld Object Variables. If, instead of an array of strings, the single string "all" is passed, the Simulator Automation Server will use predefined default fields when exporting the data.

Output

WriteAuxFile returns only one element in Output—any errors which may have occurred when attempting to execute the function.

---