

Introduction to PowerWorld Simulator: Interface and Common Tools



I15: SimAuto Overview



2001 South First Street
Champaign, Illinois 61820
+1 (217) 384.6330

support@powerworld.com
<http://www.powerworld.com>

SimAuto Overview



- SimAuto is a Microsoft Component Object Model (COM) automation server that allows PowerWorld Simulator to be controlled from an external application (COM client)
- COM clients can be written in any object-oriented programming language that runs in MS Windows and allows external type libraries (e.g. Python, Visual Basic for Applications, Visual C++, .NET Framework, MATLAB, and many more)
- Example Applications
 - Use Visual Basic for Applications to load generator cost data and unit commitment from Access or SQL, solve the OPF, and write solutions back to the database
 - Use MATLAB or Python to calculate generator parameters over time and solve a sequence of instantaneous power flows in Simulator

Using SimAuto



- Reference the Simulator Type Library in your programming environment
- Connect to the Automation Server
- Interface with Simulator using SimAuto Functions

ChangeParametersMultipleElement
ChangeParametersMultipleElementFlatInput
ChangeParametersMultipleElementRect
ChangeParametersSingleElement
GetCaseHeader
GetParametersMultipleElement
GetParametersMultipleElementFlatOutput
GetParametersMultipleElementRect
GetParametersSingleElement

GetFieldList
GetSpecificFieldList
GetSpecificFieldMaxNum
ListOfDevices
ListOfDevicesAsVariantStrings
ListOfDevicesFlatOutput
LoadState
CloseCase
OpenCase

OpenCaseType
ProcessAuxFile
RunScriptCommand
RunScriptCommand2
SaveCase
SaveState
SendToExcel
TSGetCongencyResults
WriteAuxFile

Visual Basic OPF Demonstration



- Excel VBA application allows user to perform several operations in any order
 - Solve OPF
 - Scale case
 - Write generator records to Excel
- Open Excel file *ExampleSimAutoVB02.xls* and select **Enable Content** if prompted below ribbon (code will not function if Excel settings do not allow the use of macros)
- Click **Run Main Form** button, then **Open Connection**, then **Open Case** to activate the other options

Visual Basic OPF Demonstration



Run Main Form

PowerWorld Automation Server Examples (VB)

Execute Example Quit

Open Connection Close Connection

Open Case Directory: C:\Program Files\PowerWorld\Simulator\Cases

Close Case File name: b7opf.pwb

Get Gen Parameters

OPF

Scale Case

Send Gen Info to Excel

Opened Case Successfully!

GEN PARAMETERS					
Bus#	ID	Status	AGC	MW	MVAR
1	1	Closed	YES	149.6	16.3
2	1	Closed	YES	200.0	46.9
4	1	Closed	YES	16.4	22.6
6	1	Closed	YES	200.2	-6.4
7	1	Closed	YES	200.4	39.0

OPF executed successfully!

SimAuto Tips



- This example and others may be downloaded from the PowerWorld website
- Code for examples may be accessed in Excel by selecting **Developer** Ribbon tab → **Visual Basic**
- Actions in Simulator are usually taken with the RunScriptCommand function; its arguments are SCRIPT actions described in the *Auxiliary File Format* document (In Simulator, **Window** ribbon tab → **Auxiliary File Format**)
- Simulator objects and data fields are accessed as they are in DATA sections of script files
 - Reference object names and fields with identical syntax
 - Check Variable Names or Concise Variable Names as headers in the GUI
 - In Simulator, **Window** ribbon tab → **Export Case Object Fields**

Select SimAuto Functions



- `ChangeParametersSingleElement(ObjectType, ParamList, Values)`
 - `ObjectType` : String
 - The type of object for which parameters are being changed, e.g. “BUS”.
 - `ParamList` : Variant
 - A variant array storing strings that are Simulator object field variables, e.g. “BusNum”.
 - Must contain the key fields for the objecttype.
 - `Values` : Variant
 - A variant array storing variants (integer, string, single, etc.) that are the values for each of the fields in the `ParamList`.
 - Output
 - Returns any errors in the first element, i.e. `Output(0)`

Select SimAuto Functions



- `ChangeParametersMultipleElement(ObjectType, ParamList, ValueList)`
 - `ObjectType` : String
 - The type of object for which parameters are being changed, e.g. “BUS”.
 - `ParamList` : Variant
 - A variant array storing strings that are Simulator object field variables, e.g. “BusNum”.
 - Must contain the key fields for the objecttype.
 - `ValueList` : Variant
 - A variant array storing arrays of variants.
 - Create variant arrays (one for each element being changed) with values corresponding to the fields in `ParamList`. Insert each of these variant arrays into `ValueList`.
 - Output
 - Returns any errors in the first element, i.e. `Output(0)`

Select SimAuto Functions



- ChangeParametersMultipleElement – Sample VBA Code

```
Dim ValueList(1), ParamList as Variant
```

```
Dim Output as Variant
```

```
ParamList = Array("BusNum", "AreaName")
```

```
ValueList(0) = Array(1,"Right")
```

```
ValueList(1) = Array(2,"Left")
```

```
Output = SimAuto.ChangeParametersMultipleElement("BUS",  
                                                  ParamList,ValueList)
```

Select SimAuto Functions



- `ChangeParametersMultipleElementFlatInput(ObjectType, ParamList, NoOfObjects, ValueList)`
 - `ObjectType` : String
 - Type of object for which parameters are being changed
 - `ParamList` : Variant
 - A variant array storing strings that are Simulator object field variables, e.g. “BusNum”.
 - Must contain the key fields for the object type.
 - `NoOfObjects` : Integer
 - Number of devices for which values are being passed
 - `ValueList` : Variant
 - Single-dimensional variant array storing a list of variants (integer, single, string, etc.) representing the values corresponding to `ParamList` for all devices being changed
 - All parameters for the first object are listed first followed by all of the parameters for the second object, etc.
 - `ValueList = Array(Obj1Param1, Obj1Param2, ...Obj1ParamM, Obj2Param1, Obj2Param2, ... Obj2ParamM, Obj3Param1, ...ObjNParam1, ...ObjNParamM)`
 - Output
 - Returns any errors in the first element, i.e. `Output(0)`

Select SimAuto Functions



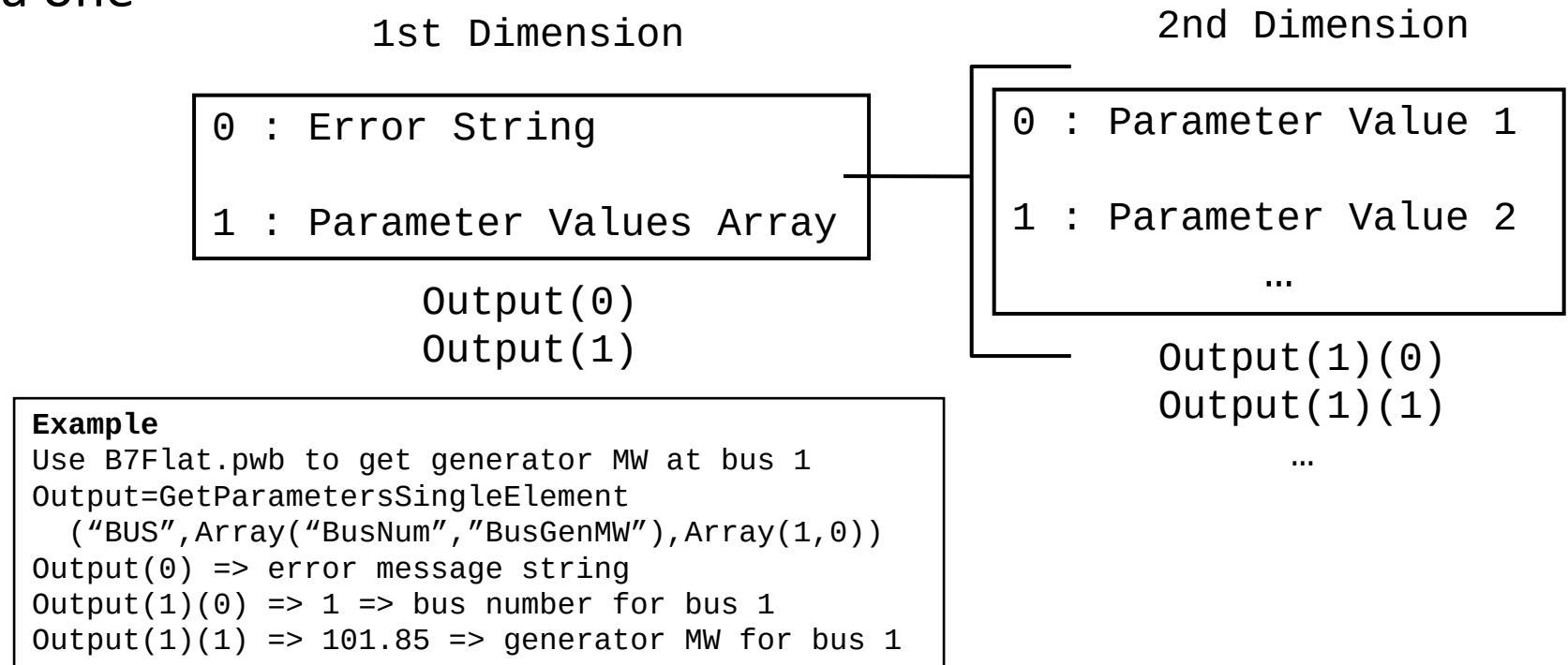
- GetParametersSingleElement(ObjectType, ParamList, Values)
 - ObjectType : String
 - The type of object for which parameters are being retrieved, e.g. “BUS”.
 - ParamList : Variant
 - A variant array storing strings that are Simulator object field variables, e.g. “BusNum”.
 - Must contain the key fields for the object type.
 - Values : Variant
 - A variant array storing variants (integer, string, single, etc.) that are the values for each of the fields in the ParamList.
 - Values must be passed in for the key fields
 - Values other than key fields should be set to zero

Select SimAuto Functions



- GetParametersSingleElement Output

- First element contains any errors
- Second element is a one-dimensional array containing values corresponding to fields specified in ParamList



Select SimAuto Functions

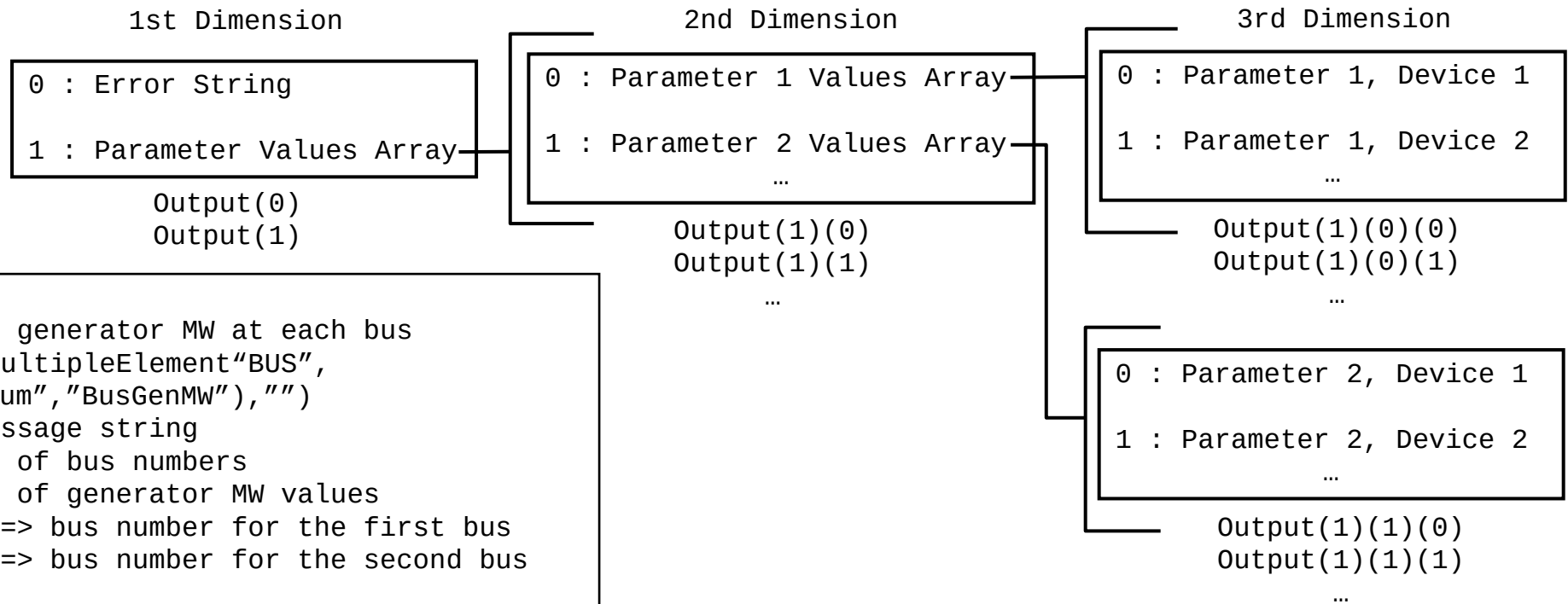


- **GetParametersMultipleElement(ObjectType, ParamList, FilterName)**
 - **ObjectType : String**
 - The type of object for which parameters are being retrieved, e.g. “BUS”.
 - **ParamList : Variant**
 - A variant array storing strings that are Simulator object field variables, e.g. “BusNum”.
 - Must contain the key fields for the object type.
 - **FilterName : String**
 - Name of a pre-defined advanced filter that will limit the objects returned.
 - Pass an empty string to return all objects of the specified type.
 - **Output**
 - Set of nested arrays containing the parameter values for the device type requested
 - Number of arrays returned depends on the number of fields in ParamList

Select SimAuto Functions



- GetParametersMultipleElement Output



Example

```
Use B7Flat.pwb to get generator MW at each bus
Output=GetParametersMultipleElement"BUS",
      Array("BusNum","BusGenMW"),"")
Output(0) => error message string
Output(1)(0) => array of bus numbers
Output(1)(1) => array of generator MW values
Output(1)(0)(0) => 1 => bus number for the first bus
Output(1)(0)(1) => 2 => bus number for the second bus
...
Output(1)(0)(6) => 7 => bus number for the seventh bus
Output(1)(1)(0) => 101.85 => gen MW for the first bus
Output(1)(1)(1) => 170.08 => gen MW for the second bus
...
Output(1)(1)(6) => 170.08 => gen MW for the seventh bus
```

Select SimAuto Functions



- GetParametersMultipleElementRect Output

1st Dimension

0 : Error String

1 : Parameter Values Rectangular Array
N x M where N = number of objects
M = number of fields

Output(0)
Output(1)

Output(1)(0,0)
Output(1)(0,1)

...
Output(1)(0,M-1)
Output(1)(1,0)
Output(1)(1,1)

...
Output(1)(N-1,M-1)

Example

Use B7Flat.pwb to get generator MW at each bus

```
Output=GetParametersMultipleElementRect  
      ("BUS",Array("BusNum","BusGenMW"),"")
```

Output(0) => error message string

Output(1) => 2-dim array of bus numbers and generator MW values

Output(1)(0,0) => 1 => bus number for the first bus

Output(1)(0,1) => 101.85 => gen MW for the first bus

Output(1)(1,0) => 2 => bus number for the second bus

Output(1)(1,1) => 170.08 => gen MW for the second bus

...

Output(1)(6,0) => 7 => bus number for the seventh bus

Output(1)(6,1) => 200.65 => gen MW for the seventh bus

2nd Dimension

Index	0	1	-	M-1
0	Object 1, Value 1	Object 1, Value 2	-	Object 1, Value M
1	Object 2, Value 1	Object 2, Value 2	-	Object 2, Value M
-	-	-	-	-
N-1	Object N, Value 1	Object N, Value 2	-	Object N, Value M

Select SimAuto Functions



- GetParametersMultipleElementFlatOutput (ObjectType,ParamList,FilterName)
 - Inputs are handled in the same manner as GetParametersMultipleElement
 - Output
 - Single-dimensional array instead of nested arrays
 - Array(errorstring, NumberOfObjectsReturned, NumberOfFieldsPerObject, Ob1Fld1, Ob1Fld2, ..., Ob(n)Fld(m-1), Ob(n)Fld(m))

Select SimAuto Functions



- GetFieldList(ObjectType)
 - Returns all fields associated with a given object type.
 - ObjectType : String
 - Type of object for which fields are requested, e.g. “BUS”.
 - Output
 - First element is the error string
 - Second element is an n x 4 array of fields
 - Similar to information obtained from Export Case Object Fields...
 - (n,0) specifies the key and required fields
 - » Key - *1*,*2*, etc.
 - » Secondary Key – *A*, *B*, etc.
 - » Required – **
 - (n,1) variablename of the field
 - (n,2) type of data stored in the field (integer, string, real)
 - (n,3) field description

Select SimAuto Functions

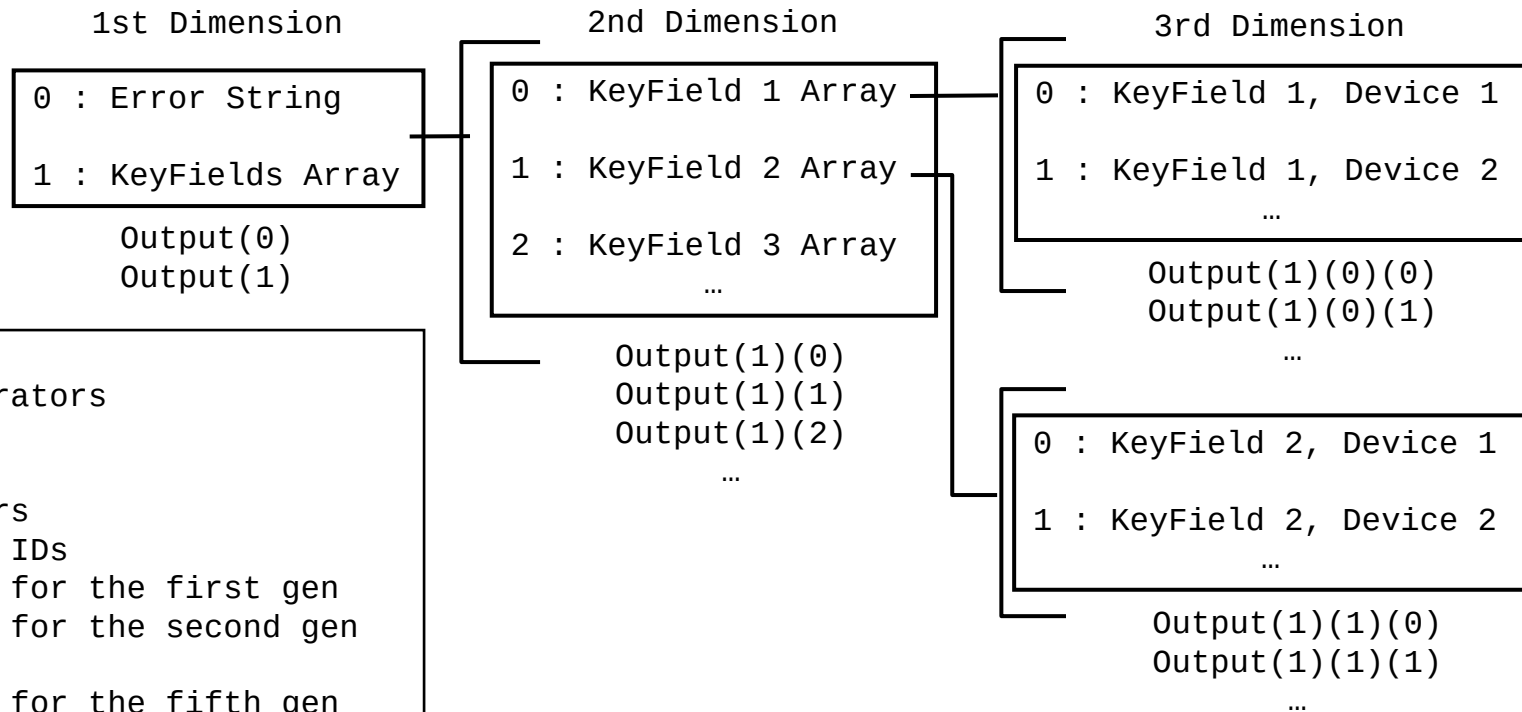


- ListOfDevices(ObjectType,FilterName)
 - ObjectType : String
 - Type of object for which devices are being acquired.
 - FilterName : String
 - Name of a pre-defined advanced filter that will limit the objects returned.
 - Pass an empty string to return all objects of the specified type.
 - Output
 - Set of nested arrays containing the key field values for the type of object requested.
 - Number of arrays returned depends on the object type selected.
 - Values in the arrays are strongly typed, i.e. bus numbers are returned as long integers instead of as a variant
 - Use ListOfDevicesAsVariantStrings to return values as variants

Select SimAuto Functions



- ListOfDevices Output



Example

```
Use B7Flat.pwb to get list of generators
Output = ListOfDevices("GEN","")
Output(0) => error message string
Output(1)(0) => array of bus numbers
Output(1)(1) => array of generator IDs
Output(1)(0)(0) => 1 => bus number for the first gen
Output(1)(0)(1) => 2 => bus number for the second gen
...
Output(1)(0)(4) => 7 => bus number for the fifth gen
Output(1)(1)(0) => "1" => gen ID for the first gen
Output(1)(1)(1) => "1" => gen ID for the second gen
...
Output(1)(1)(4) => "1" => gen ID for the fifth gen
```

Select SimAuto Functions



- ListOfDevicesFlatOutput(ObjectType, FilterName)
 - Inputs same as ListOfDevices
 - Output
 - Single-dimensional array of variants
 - Array(errorString, NumberOfObjectsReturned, NumberOfFieldsPerObject, Ob1Fld1, Ob1Fld2,..., Ob(n)Fld(m-1), Ob(n), Fld(m))

Select SimAuto Functions



- RunScriptCommand(Statements)
- Executes one or more script statements
 - Each statement must end in a semicolon
 - Ex. *RunScriptCommand("Entermode(PowerFlow); SolvePowerFlow(DC); SolvePrimalLP;")* → changes to power flow mode, solves the DC power flow, and then solves the OPF
- Returns an error string or "" if it executes without errors

Blank Page

Blank Page

Blank Page