ChangeParametersMultipleElementFlatInput Sample Code

Borland® Delphi

```
Var ParamList, ValueList : OLEVariant

// Set ParamList up to modify the MW output of generators
ParamList := VarArrayCreate([1,3], varOleStr);
ParamList[1] := 'BusNum';
ParamList[2] := 'GenID';
ParamList[3] := ‘GenMW’;

// ValueList is setup with MW values for generators at buses 1 and 2
NumObjects := 2;
NumFields := NumObjects * 3;
ValueList := VarArrayCreate([1,NumFields], varOleStr);
For k := 0 to 1 do begin
ValueList[3*k+1] := k;
ValueList[3*k+2] := ‘1’;
ValueList[3*k+3] := k*10;
End;

// Make the ChangeParametersMultipleElementFlatInput call
Output = SimAuto.ChangeParametersMultipleElementFlatInput('Sim_Solution_Options', _
ParamList, NumObjects, ValueList)
```

Microsoft® Visual Basic for Applications

```
' Set ParamList up to modify the MW output of generators
```

```
Dim ParamList As Variant

ParamList = Array("BusNum", "GenID", "GenMW")


' ValueList is setup with MW values for generators at buses 1 and 2

NumObjects = 2

NumFields = NumObjects * 3

Dim ValueList(NumObjects) As Variant

For k = 0 to 1

ValueList(3*k+1) = k

ValueList[3*k+2] := ‘1’;

ValueList[3*k+3] := k*10;

Next


' Make the ChangeParametersMultipleElementFlatInput call

Output = SimAuto.ChangeParametersMultipleElementFlatInput("Sim_Solution_Options", _

ParamList, NumObjects, ValueList)
```

ChangeParametersMultipleElement Sample Code

```
Borland® Delphi

Var ParamList, ValueList : OLEVariant


// Set ParamList up to modify the MW output of generators

ParamList := VarArrayCreate([1,3], varOleStr);

ParamList[1] := 'BusNum';

ParamList[2] := 'GenID';

ParamList[3] := ‘GenMW’;
```

```
// ValueList is setup with MW values for generators at buses 1 and 2
ValueList := VarArrayCreate([1,2], varOleStr);
For k := 1 to 2 do begin
IndValueList := VarArrayCreate([1,3], varOleStr);
IndValueList[1] := k;
IndValueList[2] := '1';
IndValueList[3] := k*10;
ValueList[k] := IndValueList;
End;

// Make the ChangeParametersMultipleElement call
Output = SimAuto.ChangeParametersMultipleElement('Sim_Solution_Options', _
ParamList, ValueList)
```

Microsoft® Visual Basic for Applications

```
' Set ParamList up to modify the MW output of generators
Dim ParamList As Variant
ParamList = Array("BusNum", "GenID", "GenMW")

' ValueList is setup with MW values for generators at buses 1 and 2
Dim ValueList(2) As Variant
For k = 0 to 1
Dim IndValueList As Variant
IndValueList = Array(k+1,"1",(k+1)*10)
ValueList(k) = IndValueList
Next
```

```
' Make the ChangeParametersMultipleElement call
Output = SimAuto.ChangeParametersMultipleElement("Sim_Solution_Options", _
ParamList, ValueList)
```

ChangeParametersSingleElement Sample Code

Borland® Delphi

```
Var ParamList, ValueList : OLEVariant

// Set ParamList up to modify the maximum number of iterations
// and the system base for the power flow simulations
ParamList := VarArrayCreate([1,2], varOleStr);
ParamList[1] := 'MaxItr';
ParamList[2] := 'SBase';

// ValueList is setup with 41 and 410 for MaxItr and SBase,
// respectively
ValueList := VarArrayCreate([1,2], varOleStr);
ValueList[1] := 41;
ValueList[2] := 410;

// Make the ChangeParametersSingleElement call
Output = SimAuto.ChangeParametersSingleElement('Sim_Solution_Options', _
ParamList, ValueList)
```

Microsoft® Visual Basic for Applications

```
' Set ParamList up to modify the maximum number of iterations
```

```
' and the system base for the power flow simulations
Dim ParamList As Variant
ParamList = Array("MaxItr", "Sbase")

' ValueList is setup with 41 and 410 for MaxItr and SBase,
' respectively
Dim ValueList As Variant
ValueList = Array(45, 90)

' Make the ChangeParametersSingleElement call
Output = SimAuto.ChangeParametersSingleElement("Sim_Solution_Options", _
ParamList, ValueList)
```

Matlab®

```
% Set ParamList up to modify the maximum number of iterations
% and the system base for the power flow simulations
ParamList = {'MaxItr' 'Sbase'};

% values is setup with 41 and 410 for MaxItr and SBase,
% respectively
values = [41 410];

% Convert the values matrix to a set of cells for passing
% through the COM interface
ValueList = num2cell(values);

' Make the ChangeParametersSingleElement call
Output = SimAuto.ChangeParametersSingleElement('Sim_Solution_Options', _
```

ParamList, ValueList)

CloseCase Function: Sample Code

Borland® Delphi

```
Output := SimAuto.CloseCase();
```

Microsoft® Visual Basic for Applications

```
Output = SimAuto.CloseCase()
```

Matlab®

```
Output = SimAuto.CloseCase
```

GetFieldList Function: Sample Code

Microsoft® Visual Basic for Applications

```
Dim objecttype As String

' Object type to obtain
objecttype = "branch"

' Make the GetField call
Output = SimAuto.GetFieldList(objecttype)
```

Matlab®

```
% Object type to obtain
objecttype = 'branch';
```

% Make the GetField call

Output = SimAuto.GetFieldList(objecttype);

LoadState Function: Sample Code

Microsoft® Visual Basic for Applications

' Make the LoadState call

Output = SimAuto.LoadState()

Matlab®

% Make the LoadState call

Output = SimAuto.LoadState();

OpenCase Function: Sample Code

Borland® Delphi

Output := SimAuto.OpenCase('c:\simauto\examples\b7opf.pwb');

if (string(Output[0]) <> '') then

StatusBar1.Panels[1].Text := 'Error: ' + string(Output[0]);

else

begin

StatusBar1.Panels[1].Text := 'Open Case successful.';
// Perform activities with opened case

end;

Microsoft® Visual Basic for Applications

```
Output = SimAuto.OpenCase("c:\simauto\examples\b7opf.pwb")

If output(0) <> "" Then

MsgBox(output(0))

Else

' Perform activities with the opened case

End If
```

Matlab®

```
Output = SimAuto.OpenCase('c:\simauto\examples\b7opf.pwb')

%If the first cell in Output ~= '', then that means an error

%occurred.

if ~(strcmp(Output{1},''))

disp(Output{1})

else

%Otherwise, no errors. Perform activities.
disp('Open Case successful')

end
```

ProcessAuxFile Function: Sample Code

Microsoft® Visual Basic for Applications

```
Dim filename As String


' Setup name of aux file to run

filename = "c:\b7opf_ctglist.aux"


' Make the processAuxFile call
```

```
Output = SimAuto.ProcessAuxFile(filename)
```

Matlab®

```
% Setup name of aux file to run
filename = 'c:\b7opf_ctglist.aux';

% Make the processAuxFile call
Output = SimAuto.ProcessAuxFile(filename);
```

RunScriptCommand Function: Sample Code

Microsoft® Visual Basic for Applications

```
Dim scriptcommand As String

' Set script command to cause Simulator to enter Run Mode
scriptcommand = "EnterMode(RUN)"

' Make the RunScriptCommand call
Output = SimAuto.RunSCriptCommand(scriptcommand);

' Set script command to cause Simulator to perform a single,
' standard solution
scriptcommand = "SolvePowerFlow(RECTNEWT)"

' Make the RunScriptCommand call
Output = SimAuto.RunSCriptCommand(scriptcommand);
```

Matlab®

```
% Set script command to cause Simulator to enter Run Mode
scriptcommand = 'EnterMode(RUN)';

% Make the RunScriptCommand call
Output = SimAuto.RunSCriptCommand(scriptcommand);

% Set script command to cause Simulator to perform a single,
% standard solution
scriptcommand = 'SolvePowerFlow(RECTNEWT)';

% Make the RunScriptCommand call
Output = SimAuto.RunSCriptCommand(scriptcommand);
```

SaveCase Function: Sample Code

Microsoft® Visual Basic for Applications

```
' Save the case as a PWB file
Output = SimAuto.SaveCase("c:\b7opfcopy.pwb", "PWB", true)

' Save the case as a PTI file
Output = SimAuto.SaveCase("c:\b7opfcopy.raw", "PTI", true)
```

Matlab®

```
% Setup name of PWB file to write
filenamepwb = 'c:\b7opfcopy.pwb';
```

```
% Setup name of PTI file to write
filenamepti = 'c:\b7opfcopy.raw';

% Make the SaveCase call for the PWB file
Output = SimAuto.SaveCase(filenamepwb, ‘PWB’, true);

% Make the SaveCase call for the PTI file
Output = SimAuto.SaveCase(filenamepti, ‘PWB’, true);
```

SaveState Function: Sample Code

Microsoft® Visual Basic for Applications

```
' Make the SaveState call
Output = SimAuto.SaveState()
```

Matlab®

```
% Make the SaveState call
Output = SimAuto.SaveState();
```

SendToExcel Function: Sample Code

Microsoft® Visual Basic for Applications

```
Dim FieldList As Variant

' Setup fieldlist to send the bus number, gen id and gen agc to Excel
FieldList = Array("pwBusNum", "pwGenID", "pwGenAGCAble")
```

```
' Make the SendToExcel call
' By specifying the parameter FieldList, only the three fields
' for each generator will be returned
Output = SimAuto.SendToExcel("gen", "", "FieldList")

' Sending the string "all" instead of a fieldlist array
' writes all predefined fields to the Excel spreadsheet
Output = SimAuto.SendToExcel("gen", "", "all")
```

Note: This function call will send the values of the fields in FieldList to an Excel workbook for all the generators in the load flow case. If a filter name had been passed instead of an empty string, Simulator would have located and used a pre-defined advanced filter and applied it to the information if it was found.

Matlab®

```
% Setup fieldlist to send the bus number, gen id and gen agc to Excel
fieldlist = {'pwBusNum' 'pwGenID' 'pwGenAGCAble' };

% Make the SendToExcel call
Output = SimAuto.SendToExcel('gen', '' , FieldList);

% Sending the string 'all' instead of a fieldlist array
% writes all predefined fields to the Excel spreadsheet
Output = SimAuto.SendToExcel('gen', '', 'all');
```

Note: This function call will send the values of the fields in FieldList to an Excel workbook for all the generators in the load flow case. If a filter name had been passed instead of an empty string, Simulator would have located and used a pre-defined advanced filter and applied it to the information if it was found.

TSGetContingencyResults Sample Code

Sample code (VB):

```
Set mySimAuto = New pwrworld.SimulatorAuto
Output = mySimAuto.OpenCase("G:\wscc_9busCacheTest.pwb")
If Output(0) <> "" Then
DisplayErrorMessage Output(0)
End If

Dim objFieldList As Variant
objFieldList = Array("Plot 'Gen_Rotor Angle'", "Bus 4 | frequency")
Output = mySimAuto.TSGetContingencyResults("ctgname1", objFieldList, "0.0", "10.0")
If Output(0) <> "" Then
DisplayErrorMessage Output(0)
End If
Set SimAuto = Nothing
```

Sample code (Matlab):

```
%Initialize SimAuto object
SimAuto = actxserver('pwrworld.SimulatorAuto');
%Initialize SimAutoOutput which is used to store the output of every method call
clear SimAutoOutput;
SimAutoOutput = {''};
%Open the case
```

```
SimAutoOutput = SimAuto.OpenCase('G:\wscc_9busCacheTest.pwb');
if ~(strcmp(SimAutoOutput{1},''))
disp(SimAutoOutput{1})
else
disp('OpenCase successful')
end
%%
% Here we get the results for all of the angles directly into Matlab via SimAuto
%%
newCtgName = 'ctgName';
objFieldList = {'"Plot ''Gen_Rotor Angle''"' };
SimAutoOutput = SimAuto.TSGetContingencyResults(newCtgName, objFieldList , '0.0', '10.0');
if ~(strcmp(SimAutoOutput{1},''))
disp(SimAutoOutput{1})
else
disp('GetTSResultsInSimAuto successful')

%Get the results
localResults = SimAutoOutput{3};

%Get the header variables to use for plot labels
Header = SimAutoOutput{2};

% Convert a matrix of strings into a matrix of numbers and plot them
localResultsMat = str2double(localResults);
plot(localResultsMat(:,1), localResultsMat(:,[2:size(localResultsMat,2)]));
title(['Transient stability results for ', newCtgName]);
TimeSeriesNames = Header([4:size(Header,1)],2);
```

```
legend(TimeSeriesNames); % these labels come from the HEADER
xlabel('time')
ylabel('Generator rotor angle (degrees)');
end
%Delete (close) the COM object.
delete(SimAuto);
```

WriteAuxFile Function: Sample Code

```
Microsoft® Visual Basic for Applications
Dim FieldList As Variant
Dim auxfilename As String

' Setup FieldList to send the bus number, gen id and gen agc
FieldList = Array("pwBusNum", "pwGenID", "pwGenAGCAble")

' Aux file to write to
auxfilename = "c:\businfo.aux"

' Make the WriteAuxFile call
' By specifying the parameter FieldList, only the three fields
' for each generator will be returned
Output = SimAuto.WriteAuxFile(auxfilename, "", "gen", true, FieldList)

' Sending the string "all" instead of the FieldList array
```

' writes all predefined fields to the Excel spreadsheet

Output = SimAuto.SendToExcel(auxfilename, "", "gen", true, "all")

Note: This function call will send the values of the fields in FieldList to an auxiliary file for all the generators in the load flow case. If a filter name had been passed instead of an empty string, Simulator would have located and used a pre-defined advanced filter and applied it to the information if it was found.

Matlab®

% Setup FieldList to send the bus number, gen id and gen agc

fieldlist = {'pwBusNum' 'pwGenID' 'pwGenAGCAble' };

% Aux file to write to

auxfilename = 'c:\businfo.aux';

% Make the WriteAuxFile call

Output = SimAuto.WriteAuxFile(auxfilename, '', 'gen', true, FieldList);

% Sending the string 'all' instead of the FieldList array

% writes all predefined fields to the .aux file

Output = SimAuto.WriteAuxFile(auxfilename, '', 'gen', true, 'all');

Note: This function call will send the values of the fields in FieldList to an auxiliary file for all the generators in the load flow case. If a filter name had been passed instead of an empty string, Simulator would have located and used a pre-defined advanced filter and applied it to the information if it was found.
