Automating PowerWorld with Python

Auxiliary Files and Scripting Tips

December 9-12, 2019

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

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

Motivation

- Provide hints about more advanced/obscure options that can be used with auxiliary files and SimAuto
- Emphasize where to get help about object types and fields used in auxiliary files and SimAuto
- Provide Python example when an unknown number of fields is returned

© 2019 PowerWorld Corporation

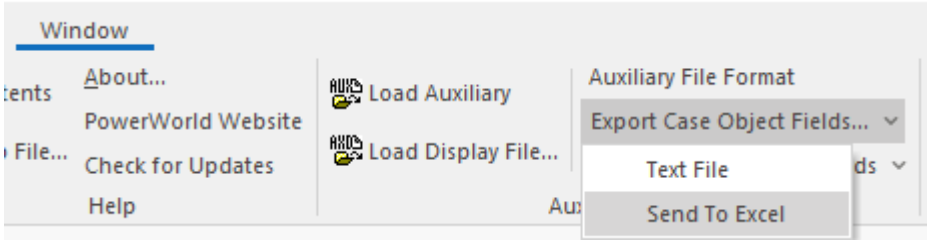
Available Online Help

- https://www.powerworld.com/WebHelp/Content/Other_Documents/Auxiliary-File-Format.pdf
  - Most up-to-date description of all available script commands and other auxiliary file related information
- https://www.powerworld.com/WebHelp/
  - Most up-to-date help information for all of Simulator
  - This contains the SimAuto documentation

© 2019 PowerWorld Corporation

Export Case Object Fields (More Documentation)

- The only comprehensive documentation about all of the object types and fields that are available



© 2019 PowerWorld Corporation

Export Case Object Fields

< Included in Difference Case comparison
***** Required for creating object
1, 2, 3 Primary key
A, B, C Secondary key

For the majority of fields the description found here is the most comprehensive information that you will find describing them

	A	B	C	D	E	F	
1	Object Type	SUBDATA Allowed	Key/Required F	Variable Name	Concise Variable Name	Type of Variable	Description
19011	Gen						
19012		BidCurve					
19013		ReactiveCapability					
19014				ABCPhaseAngle	FaultCurAngleA	Real	Phase A
19015				ABCPhaseAngle:1	FaultCurAngleB	Real	Phase B
19016				ABCPhaseAngle:2	FaultCurAngleC	Real	Phase C
19017				ABCPhaseI	FaultCurMagA	Real	Phase A
19018				ABCPhaseI:1	FaultCurMagB	Real	Phase B
19019				ABCPhaseI:2	FaultCurMagC	Real	Phase C
19020			<	AllLabels	AllLabels	String	This represents a comma-separated list
19021			<	AreaName	AreaName	String	It is possible for the terminal bus to bel
19022			<	AreaName:1	BusAreaName	String	It is possible for the terminal bus to bel
19023			<	AreaNum	AreaNumber	Integer	It is possible for the terminal bus to bel
19024			<	AreaNum:1	BusAreaNumber	Integer	It is possible for the terminal bus to bel
19025			<	BAName	BAName	String	It is possible for the terminal bus to bel
19026			<	BAName:1	BusBAName	String	It is possible for the terminal bus to bel
19027			<	BANumber	BANumber	Integer	It is possible for the terminal bus to bel
19028			<	BANumber:1	BusBANumber	Integer	It is possible for the terminal bus to bel
19029			<	BreakerDelay	BreakerDelay	Real	Breaker time delay in seconds
19030				BreakerGroupNum	BreakerGroupNum	Integer	ID of the Bus's breaker group of the bus
19031				BusCat	BusCat	String	Shows how the bus is being modeled in
19032				BusgenericSensP	SensdValuedPinj	Real	Sensitivity: Injection dValue/dP of Bus
19033				BusgenericSensQ	SensdValuedQinj	Real	Sensitivity: Injection dValue/dQ of Bus
19034				BusGenericSensV	SensdValuedVset	Real	Sensitivity: Injection dValue/dVsetpoin
19035				BusKVVolt	kV	Real	Voltage: kV Actual of the bus
19036				BusLossSensMW	LossSensMW	Real	Sensitivity of the MW losses with respe
19037				BusMCMW	BusMargCostMW	Real	OPF: Marginal MW Cost. May be interpr
19038			<	BusName	BusName	String	Name of the bus
19039			*A*<	BusName_NomVolt	BusNameNomkV	String	Name_Nominal kV of the bus
19040			<	BusNomVolt	NomkV	Real	The nominal kv voltage specified as par
19041			*1*<	BusNum	BusNum	Integer	Number of the bus
19042				BusObjectOnline	Online	String	YES only if the generator Status = CLOSE
19043				BusOwnerName	BusOwnerName	String	Name of the Owner of the attached bus

© 2019 PowerWorld Corporation

SimAuto Functions for Obtaining Field Information

- SimAuto functions for obtaining field details
  - GetFieldList(ObjectType)
    - Return all fields and details for a particular object type
    - Second element, Output[1], is an n x 5 array of 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
      - [n][4] concise variablename of the field

© 2019 PowerWorld Corporation

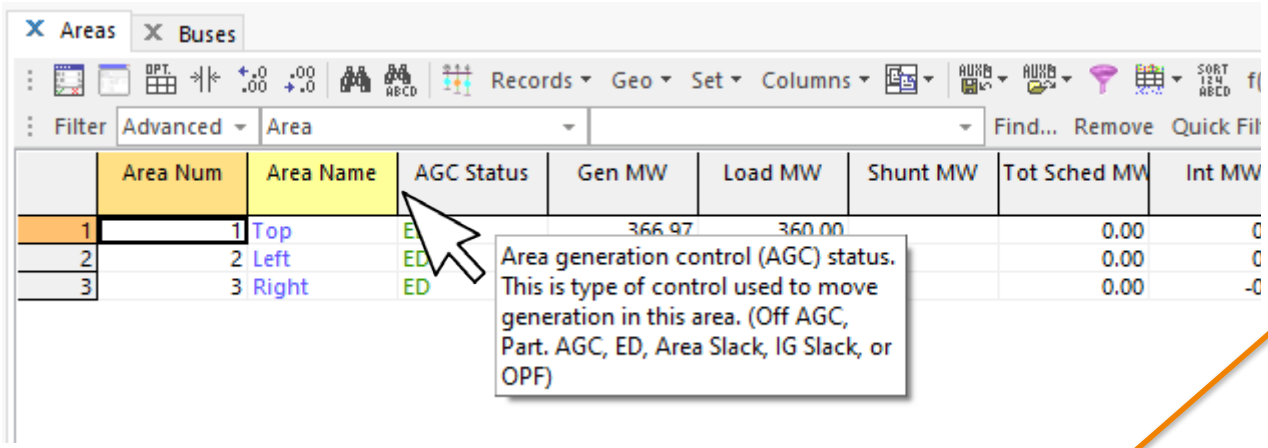
SimAuto Functions for Obtaining Field Information

- GetSpecificFieldList(ObjectType, FieldList)
  - Return specified fields and details for a particular object type
  - Second element, Output[1], is an n x 5 array of 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
    - [n][4] concise variablename of the field

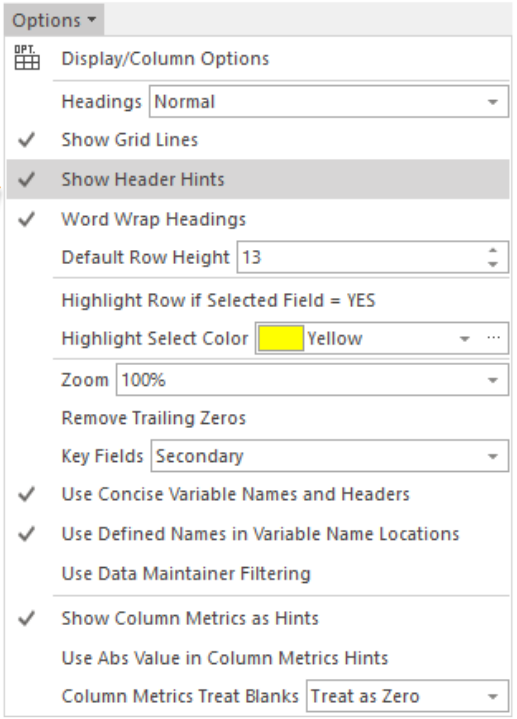
© 2019 PowerWorld Corporation

Hover Hints

- Hover over column headers in case information displays to see the description of fields



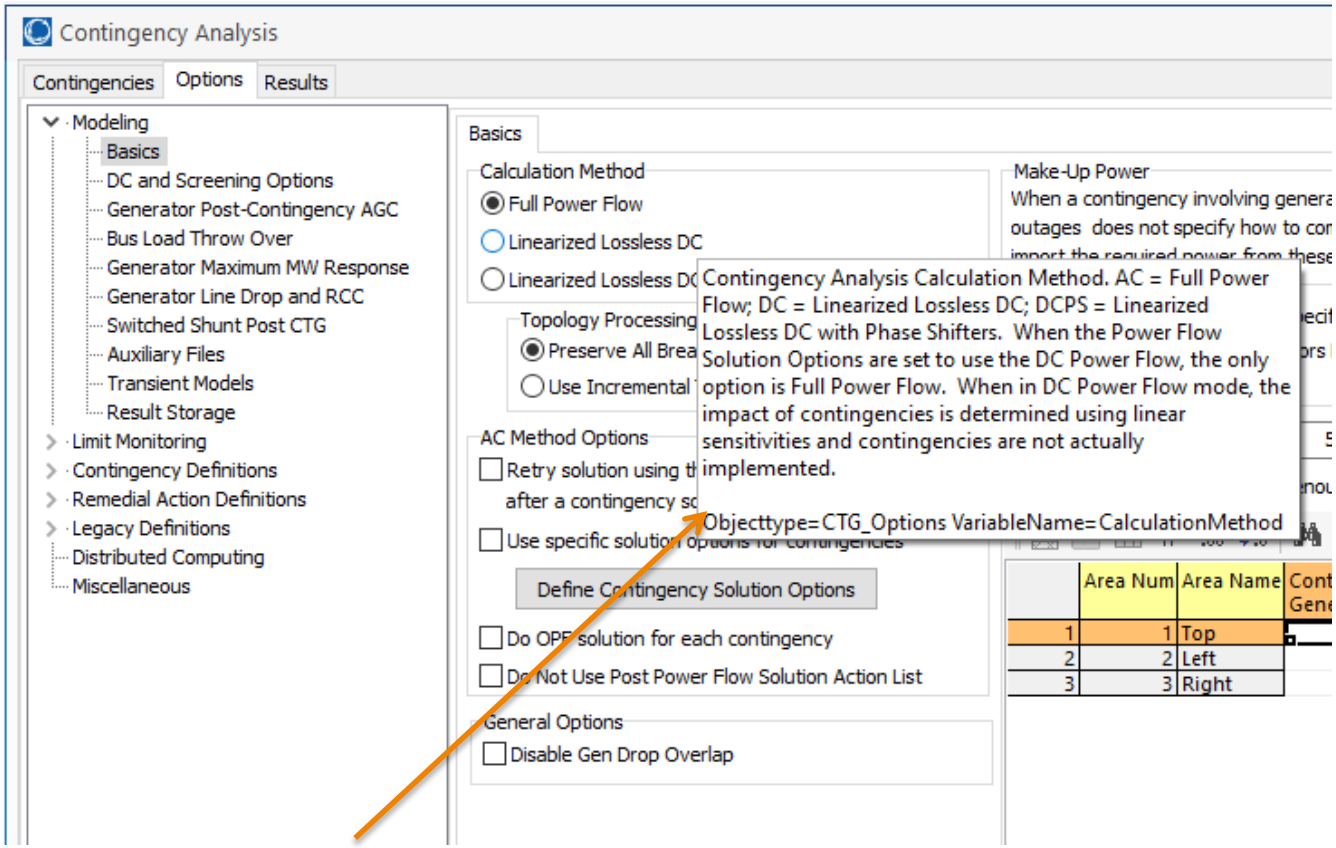
Hints can be enabled/disabled by toggling **Show Header Hints** option



© 2019 PowerWorld Corporation

Hover Hints

- Some dialogs have hints that pop up when hovering over option fields
- These hints provide the object type and variable name for setting this option in an auxiliary file data section or script command

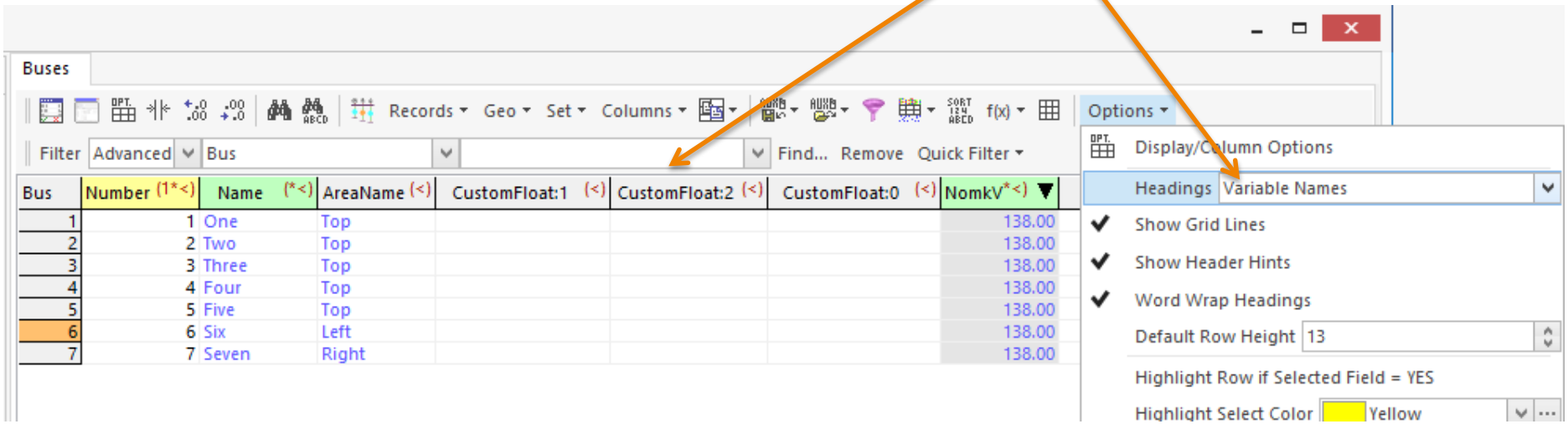


Objecttype and VariableName to be used in an auxiliary file

© 2019 PowerWorld Corporation

Specifying Field Variable Names

- Variable Names are used within auxiliary file DATA and SCRIPT sections to identify fields
- For help in creating auxiliary files, column headers within the Model Explorer can be set to show the variable names instead



© 2019 PowerWorld Corporation

Specifying Field Variable Names

- Variables follow the naming convention `variablename:location` where `location` is an integer indicating the exact variable to be used
- This convention is considered the legacy convention where the same `variablename` was re-used by tacking on the `location`
- Starting with Simulator version 19, *concise* variable names have been created in an attempt to get rid of the `location` in as many variables as possible and to make the names more obvious

© 2019 PowerWorld Corporation

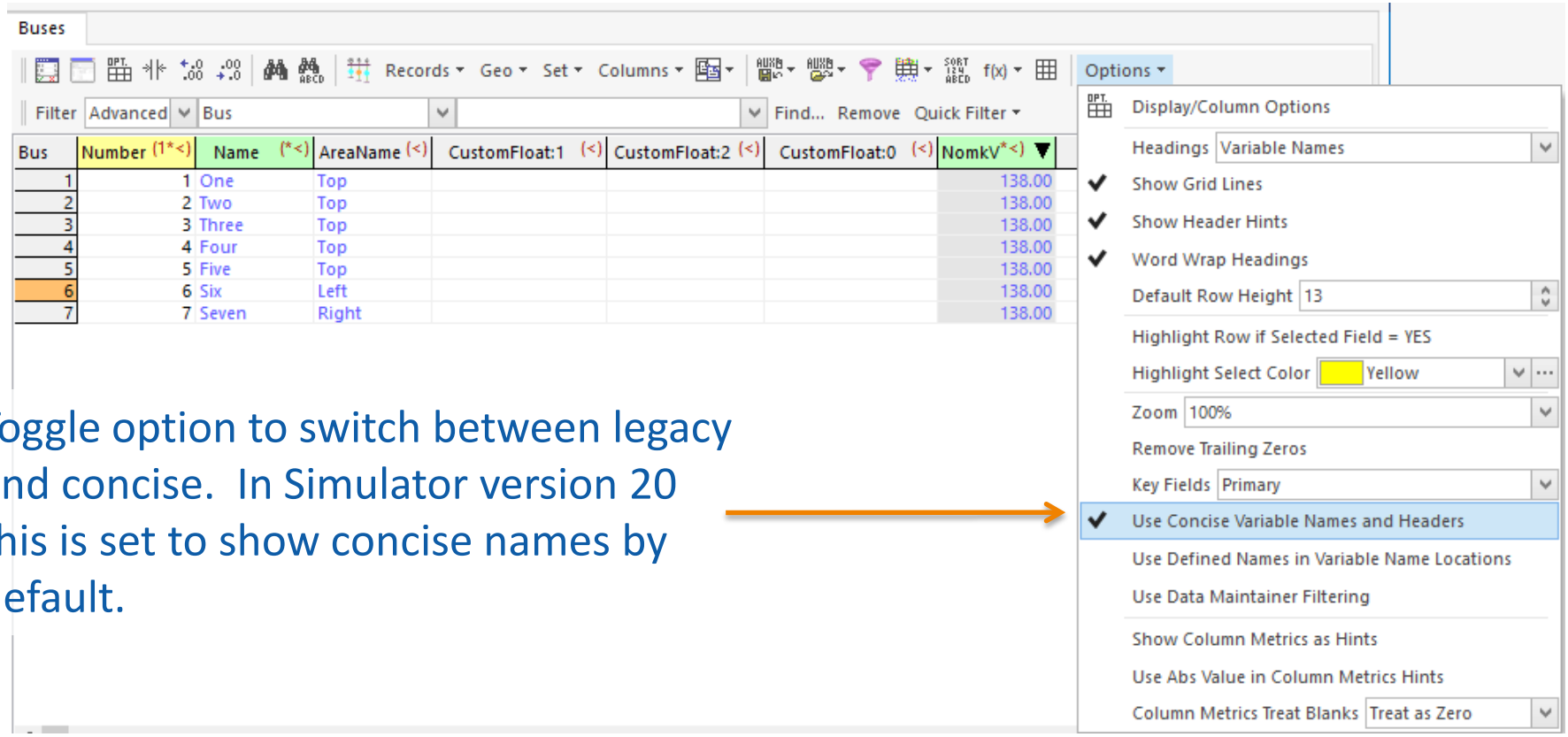
Specifying Field Variable Names Legacy vs. Concise

- The legacy variables to indicate the MW flow at the from end and to ends of a branch are:
  - `LineMW:0` (:0 is typically omitted because :0 is assumed if no location specified)
  - `LineMW:1`
- The same variables in the concise format are:
  - `MWFrom`
  - `MWTo`

© 2019 PowerWorld Corporation

Specifying Field Variable Names Legacy vs. Concise

- Both versions of a field variable name can be shown in the Model Explorer



Toggle option to switch between legacy and concise. In Simulator version 20 this is set to show concise names by default.

© 2019 PowerWorld Corporation

Specifying Field Variable Names Number Formatting

- `variablename:location:digits:decimals`
  - `digits` specifies the total number of digits
    - Default depends on how the field is being used: 12 when saving to file and 32 when setting a value
  - `decimals` specifies the digits to the right of the decimal
    - Default depends on how the field is being used: 6 when saving to file and 16 when setting a value
  - When using concise variable names both `digits` and `decimals` must be specified if specified at all
    - `location` can be omitted whether or not `digits` and `decimals` are specified
  - When using legacy variable names `location`, `digits`, and `decimals` can be omitted but the previous parameters must be specified for a given parameter to be included

© 2019 PowerWorld Corporation

Specifying Field Variable Names

- `Location` indicators continue to exist for some variable names
- Typically these are fields for which a dynamic number exists
  - Fields that are available for most objects
    - `CustomExpression`, `CustomFloat`, `CustomInteger`, `CustomString`, `CustomExpressionStr`, `CalcFied`, `DataCheck`, and `DataCheckAggr`
  - Some example fields that are available for specific objects
    - Bus – `MargCostMWValue`, `MargControl`, `SensdValuedPinjMult`
    - Branch – `PTDFMult`, `LODFMult`, `LODFInterfaceMult`
    - Gen – `BidMW`, `BidMWHr`, `SensMultMeterMultControl`

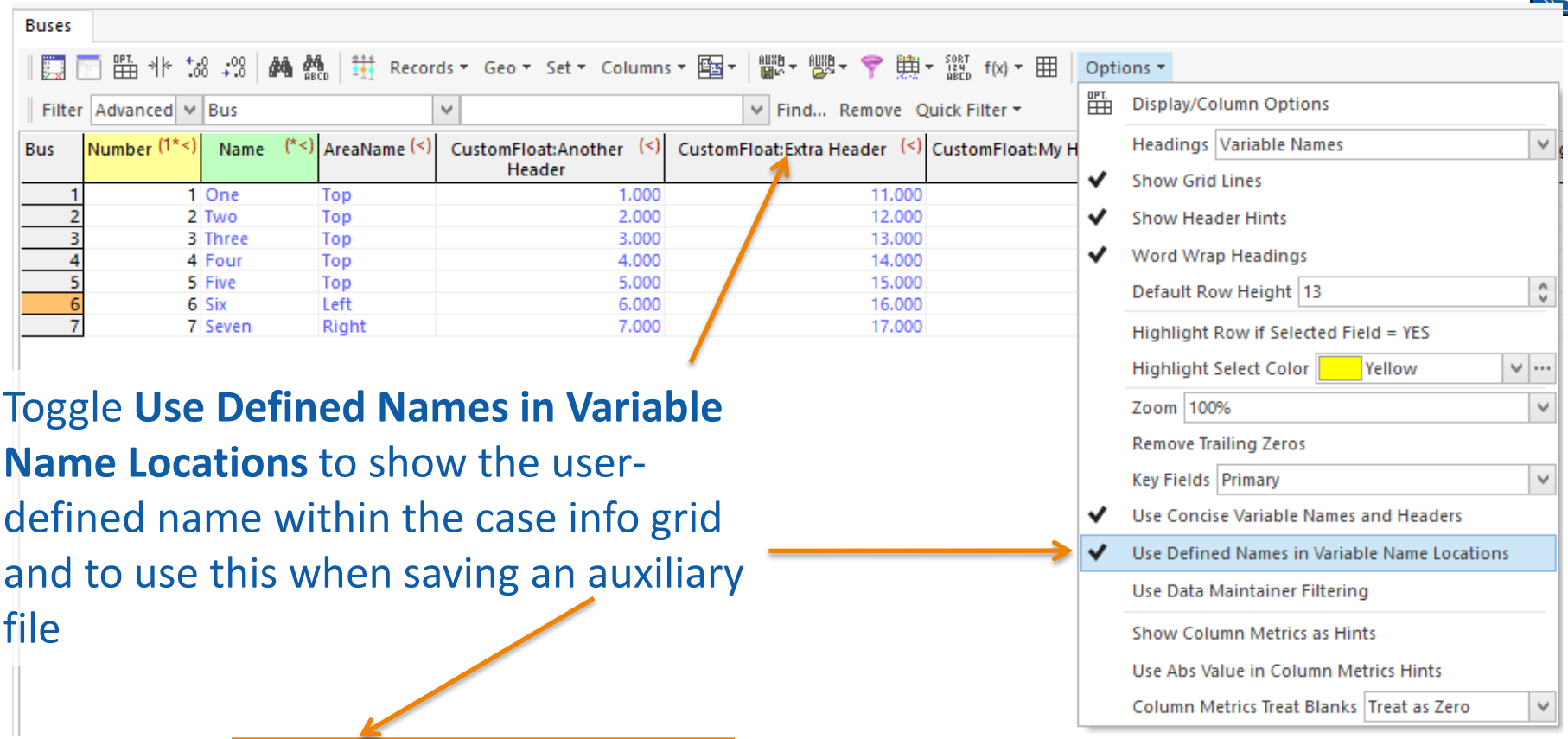
© 2019 PowerWorld Corporation

Specifying Field Variable Names

- To better identify dynamic fields, the `Location` can be replaced with `Location_by_name` for fields that have user-defined names
- This could help prevent conflicts in data where the same `Location` by number is being referenced in different auxiliary files but these should really be different fields
- The fields can be read and written by Simulator when used in auxiliary files
  - `CustomExpression`, `CustomFloat`, `CustomInteger`, `CustomString`, `CustomExpressionStr`, `CalcFied`, `DataCheck`, and `DataCheckAggr`
  - `"CustomFloat:`*`My Header`*`"`
  - `"CalcField:`*`My calculated field name`*`"`
  - `"DataCheckAggr:Bus '`*`DataCheck Name`*`'"`

© 2019 PowerWorld Corporation

Specifying Field Variable Names



Toggle **Use Defined Names in Variable Name Locations** to show the user-defined name within the case info grid and to use this when saving an auxiliary file

```
Bus (Number,Name,AreaName,CustomFloat:Another Header,CustomFloat:Extra Header,
   CustomFloat:My Header,NomkV,Vpu,kV,Vangle,LoadMW,LoadMvar,GenMW,GenMvar,
   ShuntMvar,ActG,ActB,AreaNumber,ZoneNumber)
{
    1 "One"          "Top"    1.000   11.000   21.000   138.00  1.05000  144.900     6.09 "" ""   101.85     5.25 ""     0.00     0.00        1        1
    2 "Two"          "Top"    2.000   12.000   22.000   138.00  1.04000  143.520     4.22    40.00    20.00   170.08    33.24 ""     0.00     0.00        1        1
    3 "Three"        "Top"    3.000   13.000   23.000   138.00  0.99269  136.991     0.99   110.00    40.00 "" "" ""     0.00     0.00        1        1
    4 "Four"         "Top"    4.000   14.000   24.000   138.00  1.00000  138.000     1.46    80.00    30.00    95.03    19.99 ""     0.00     0.00        1        1
    5 "Five"         "Top"    5.000   15.000   25.000   138.00  1.00665  138.917    -0.83   130.00    40.00 "" "" ""     0.00     0.00        1        1
    6 "Six"          "Left"   6.000   16.000   26.000   138.00  1.04000  143.520     2.81   200.00     0.00   200.33    -6.59 ""     0.00     0.00        2        1
    7 "Seven"        "Right"  7.000   17.000   27.000   138.00  1.04000  143.520     0.00   200.00     0.00   200.65    51.29 ""     0.00     0.00        3        1
}
```

Specifying Field Variable Names Saving All Fields for a Variable

- What if you don't know how many fields exist for a particular variable name?
  - Multiple PTDF or LODF calculation produces a matrix of results based on the inputs selected
- Replace the `Location` identifier with the keyword `ALL`
  - `PTDFMult:ALL` or `LODFMult:ALL`
- Available with script commands that allow specifying fields to save to file
  - `CTGSaveViolationMatrices`, `SaveData`, `SaveDataWithExtra`, `SaveObjectFields`, and `SendToExcel`

© 2019 PowerWorld Corporation

Specifying Field Variable Names Saving All Fields for a Variable

- Available with SimAuto functions that get fields
  - `GetParametersMultipleElement`, `GetParametersMultipleElementRect`, `GetParametersMultipleElementFlatOutput`, `GetParametersSingleElement`, `GetSpecifiedFieldList`, `SendToExcel`, and `WriteAuxFile`
- Save the LODF matrix results to a CSV file

```
SaveData("myfile.CSV",CSVColHeader,Branch,[BusNumFrom,BusNumTo,
Circuit,LODFMult:ALL],[],,[],NO);
```

Use filetype = CSVColHeader to identify the monitored branches by the normal names instead of variable names

© 2019 PowerWorld Corporation

PW_Multiple_TLR_Script.aux

Multiple TLR Scripting Example

- Problem
  - Calculate TLR sensitivities for all branches in the B7Flat.pwb case for two different scenarios
  - Save results in CSV files or send result to same Excel workbook
  - Return results without knowing how many branches are studied
- Solution
  - Script commands
    - SelectAll
    - CalculateTLRMultipleElement
    - SendToExcel
    - DeleteFile
    - SaveData
    - SetData
    - SolvePowerFlow
  - Significant fields
    - SensdValuedPInjMult:ALL

© 2019 PowerWorld Corporation

Multiple TLR Scripting Example

```
SCRIPT
{
  OpenCase("B7Flat.PWB");
  SelectAll(Branch, "");
  CalculateTLRMultipleElement(Branch, SELECTED, BUYER, [SLACK], DC);
  //SendToExcel(Bus, [Number, Name, SensdValuedPInjMult:ALL], "", YES,
  //            "MyTLRFile.xlsx", "Original B7Flat", [], [], []);
  DeleteFile("MyTLROutput.CSV");
  SaveData("MyTLROutput.CSV", CSVColHeader, Bus, [Number, Name,
           SensdValuedPInjMult:ALL:8:4], [], "", [], No);
}
SCRIPT
{
  OpenCase("B7Flat.PWB");
  SetData(Branch, [BusNumFrom, BusNumTo, Circuit, Status], [2, 4, "1", "OPEN"], "");
  SolvePowerFlow;
  SelectAll(Branch, "");
  CalculateTLRMultipleElement(BRANCH, SELECTED, BUYER, [SLACK], DC);
  //SendToExcel(Bus, [Number, Name, SensdValuedPInjMult:ALL], "", YES,
  //            "MyTLRFile.xlsx", "Line 2 to 4 Open", [], [], []);
  DeleteFile("MyTLROutput_Line_2_to_4_Open.CSV");
  SaveData("MyTLROutput_Line_2_to_4_Open.CSV", CSVColHeader, Bus, [Number, Name,
           SensdValuedPInjMult:ALL:8:4], [], "", [], No);
}
```

Select all branches to include in calculations

For large case use advanced filtering to limit the branches

Return results for all branches for which TLRs were calculated

© 2019 PowerWorld Corporation

Python Example

- Problem
  - Calculate TLR sensitivities for all branches in the B7Flat.pwb case for two different scenarios
  - Display a table of the results for the two scenarios side-by-side for comparison
  - Return results without knowing how many branches are studied
- Solution
  - SimAuto functions
    - OpenCase
    - GetSpecificFieldList
    - GetParametersMultipleElementRect
    - ChangeParametersSingleElement
    - RunScriptCommand
  - Script commands
    - SelectAll
    - CalculateTLRMultipleElement
    - SolvePowerFlow
  - Significant fields
    - SensdValuedPInjMult:ALL

© 2019 PowerWorld Corporation

Python Example

- Run multiple element TLR calculation on unknown number of elements

```
simauto_output = local_simauto_obj.RunScriptCommand('SelectAll(BRANCH, "");')
simauto_output =
local_simauto_obj.RunScriptCommand('CalculateTLRMultipleElement(BRANCH, SELECTED,
BUYER, [SLACK], DC);')
```

For real cases should use filtering instead of calculating for all branches

- Return results for all elements using the column headers that appear within the Simulator GUI

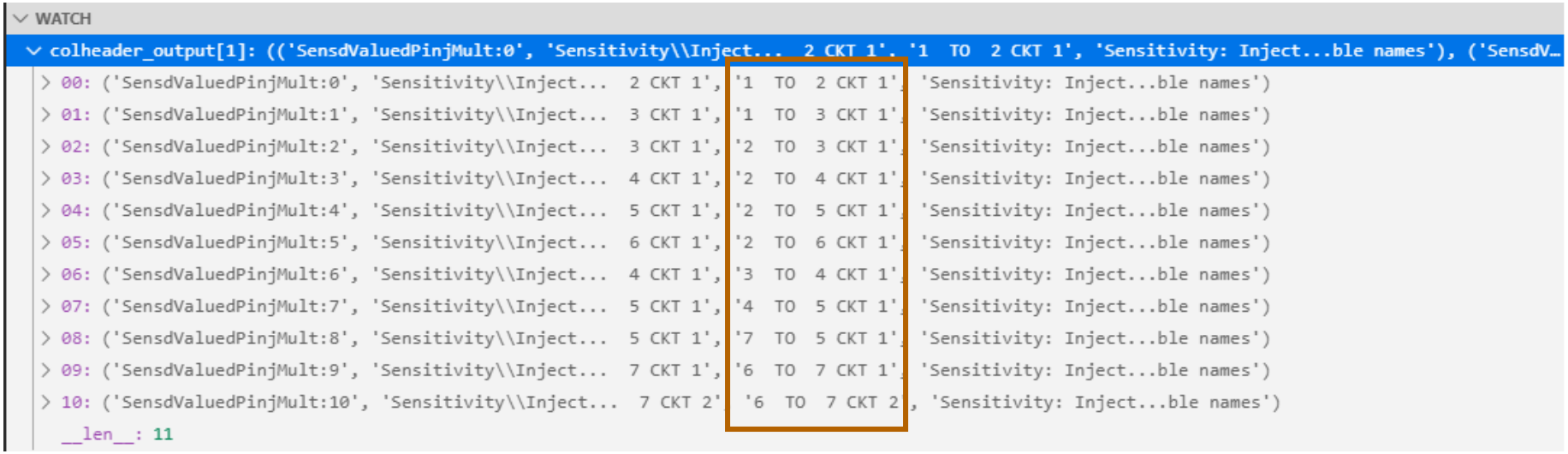
```
colheader_output = local_simauto_obj.GetSpecificFieldList('Bus',['SensdValuedPinj
Mult:ALL'])
tlr_output.append(local_simauto_obj.GetParametersMultipleElementRect('Bus',
['Number', 'Name', 'SensdValuedPInjMult:ALL'], ""))
```

© 2019 PowerWorld Corporation

Python Example

```
colheader_output =
  local_simauto_obj.GetSpecificFieldList('Bus',['SensdValuedPinjMult:ALL'])
```

- GetSpecificFieldList(ObjectType, FieldList)
  - Output[0] contains error string
  - Output[1] is an n x 4 array of fields
    - Output[1][n][0] – variablename:location
    - Output[1][n][1] – field identifier (identifier that appears when looking at list of available fields)
    - Output[1][n][2] – column header
    - Output[1][n][3] – field description



© 2019 PowerWorld Corporation

Python Example

```
tlr_output.append(local_simauto_obj.GetParametersMultipleElementRect('Bus',
['Number', 'Name', 'SensdValuedPInjMult:ALL'], ""))
```

Get the results for all of the calculated TLR values

```
header_labels = ['Number', 'Name']
    for i in range(len(local_colheader_output[1])):
        for k in range(len(local_tlr_output)):
            header_labels.append(local_colheader_output[1][i][2] + ' S' + str(k))
```

Getting the length of the results, Output[1], from GetSpecificFieldList will determine the value of n

Specify the labels for the result table to use the column headers

© 2019 PowerWorld Corporation

Python Example

- GetParamtersMultipleElementRect(ObjectType, ParamList, FilterName)
  - Output[0] contains error string
  - Output[1] is a two-dimensional array with each row representing an object and the corresponding columns representing each field in ParamList for that object

```
field_count = 0
for i in range(len(local_tlr_output[0][1])):           # this gives the number of devices
    for j in range(len(local_tlr_output[0][1][i])) :   # this gives the number of fields
        for k in range(len(local_tlr_output)):         # this is the number of scenarios
            if (j > 1):
                result_table.setItem(i, j + k + field_count,
                 QTableWidgetItem('{0:.4f}'.format(float(local_tlr_output[k][1][i][j]))))
            elif (k == 0):
                result_table.setItem(i, j, QTableWidgetItem(local_tlr_output[k][1][i][j]))
        if (j > 1):
            field_count = field_count + 1
    field_count = 0
```

local_tlr_output[0][1][i] returns the number of fields for the ith object. Assumption here is that the same number of fields will exist for both of the scenarios.

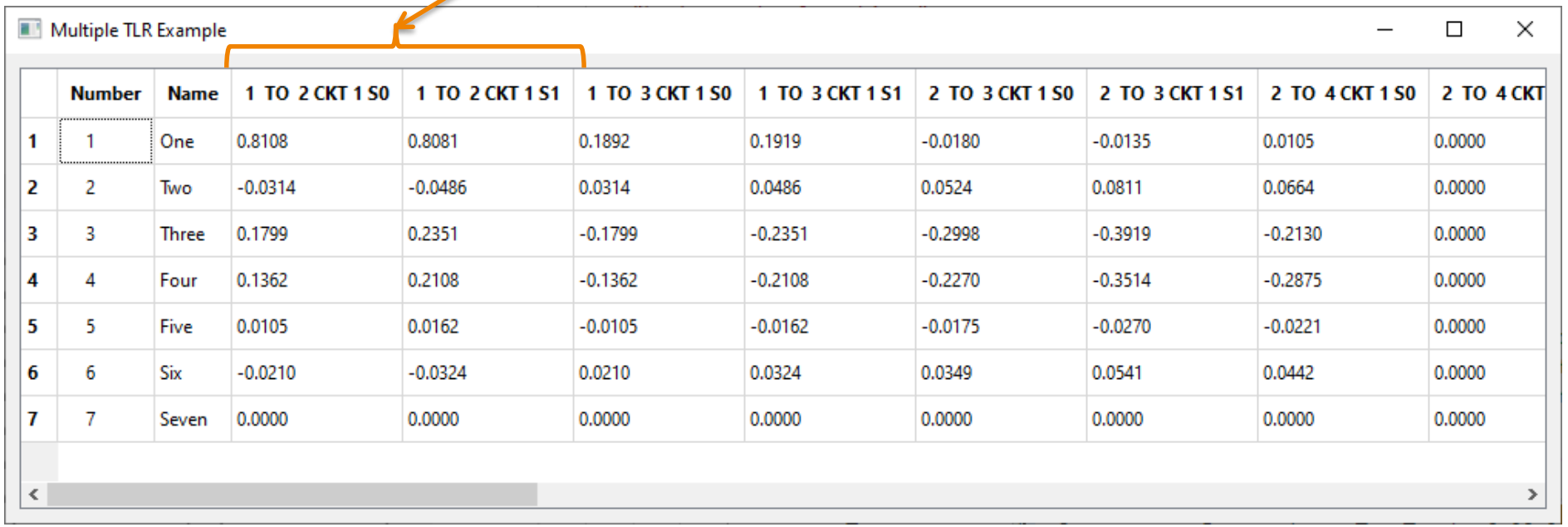
Getting the length of the results, Output[1], from GetParametersMultipleElementRect will determine how many objects are returned.
local_tlr_output[0] are the results for the first scenario, which means that local_tlr_output[0][1] contains the results. Assumption here is that there is at least one scenario.

TLR sensitivity for scenario k, object i, field j

© 2019 PowerWorld Corporation

Python Example

Sensitivity for the same branch for different scenarios



Multiple TLR Example

	Number	Name	1 TO 2 CKT 1 S0	1 TO 2 CKT 1 S1	1 TO 3 CKT 1 S0	1 TO 3 CKT 1 S1	2 TO 3 CKT 1 S0	2 TO 3 CKT 1 S1	2 TO 4 CKT 1 S0	2 TO 4 CKT
1	1	One	0.8108	0.8081	0.1892	0.1919	-0.0180	-0.0135	0.0105	0.0000
2	2	Two	-0.0314	-0.0486	0.0314	0.0486	0.0524	0.0811	0.0664	0.0000
3	3	Three	0.1799	0.2351	-0.1799	-0.2351	-0.2998	-0.3919	-0.2130	0.0000
4	4	Four	0.1362	0.2108	-0.1362	-0.2108	-0.2270	-0.3514	-0.2875	0.0000
5	5	Five	0.0105	0.0162	-0.0105	-0.0162	-0.0175	-0.0270	-0.0221	0.0000
6	6	Six	-0.0210	-0.0324	0.0210	0.0324	0.0349	0.0541	0.0442	0.0000
7	7	Seven	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

© 2019 PowerWorld Corporation

Specifying Field Variable Names Saving All Fields for an Object Type

- What if you just want everything?
- Replace entire list of fields with keyword ALL
- Available with script commands that allow specifying fields to save to file
  - `CTGSaveViolationMatrices`, `SaveData`, `SaveDataWithExtra`, `SaveObjectFields`, and `SendToExcel`
- Available with SimAuto functions that get fields
  - `GetParametersMultipleElement`, `GetParametersMultipleElementRect`, `GetParametersMultipleElementFlatOutput`, `GetParametersSingleElement`, `GetSpecifiedFieldList`, `SendToExcel`, and `WriteAuxFile`

© 2019 PowerWorld Corporation

Specifying Field Variable Names Saving All Fields for an Object Type

- Available with SimAuto functions that get fields
  - `GetParametersMultipleElement`, `GetParametersMultipleElementRect`, `GetParametersMultipleElementFlatOutput`, `GetParametersSingleElement`, `GetSpecifiedFieldList`, `SendToExcel`, and `WriteAuxFile`
- Saving all fields for a Branch

```
SaveData("myfile.CSV", CSVColHeader,
Branch,[ALL],[],,[],NO);
```

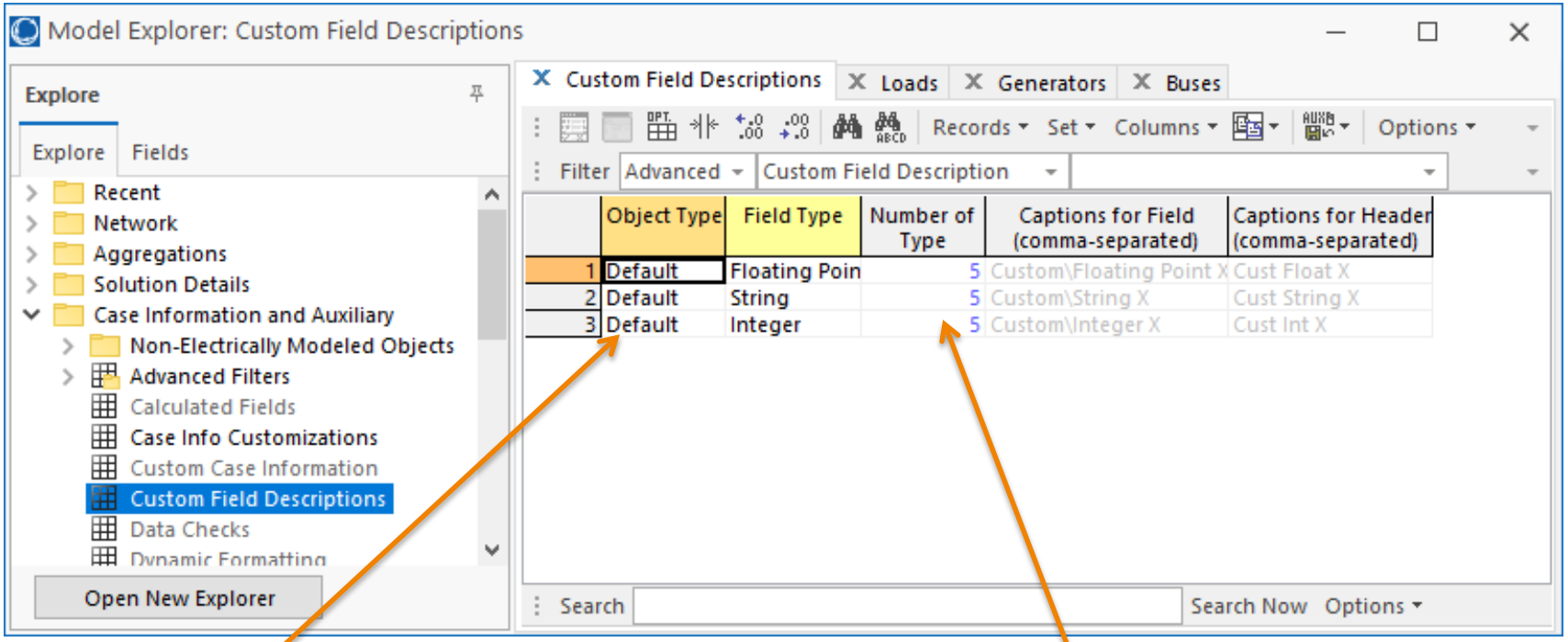
© 2019 PowerWorld Corporation

Custom Fields

- Simulator has many fields, but sometimes that is not enough
- Custom fields are available for data that does not fit anywhere else
- Custom fields can also be useful for temporary storage of values
  - e.g., store the pre-change flow to compare against post-change flow
- By default 5 Custom Floating Point, Custom Integer, and Custom String fields are available for most objects
- Number and names of each type of field can be customized on an object type basis

© 2019 PowerWorld Corporation

Custom Field Descriptions



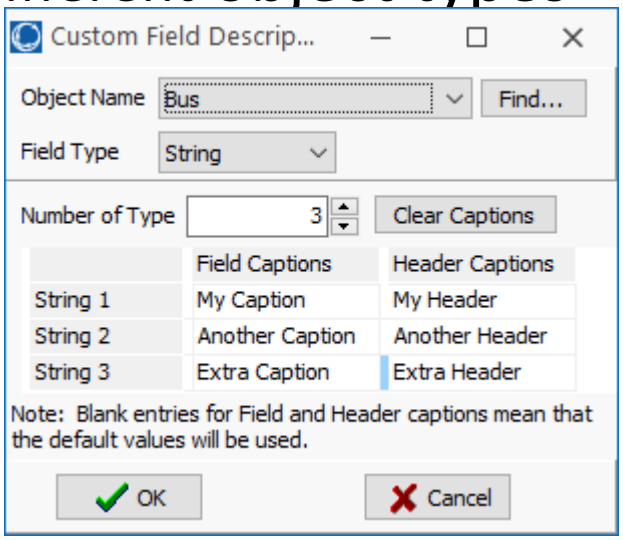
Default object types are always listed. These apply to all object types.

Number of default fields of a particular type can be changed without additional customization.

© 2019 PowerWorld Corporation

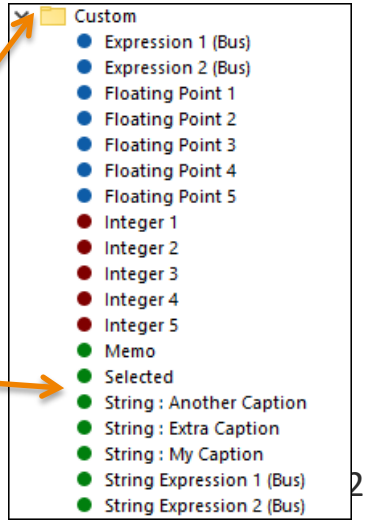
Custom Field Descriptions: Custom Floats, Integers, and String

- Custom settings can be specified for different object types by right clicking and choosing **Insert**
  - Choose the **Object Name**
  - Choose the **Field Type**
  - Choose the **Number of Type**
  - Define **Field Captions**
  - Define **Header Captions**



	Object Type	Field Type	Number of Type	Captions for Field (comma-separated)	Captions for Header (comma-separated)
1	Default	Floating Poin	5	Custom\Floating Point X	Cust Float X
2	Default	String	5	Custom\String X	Cust String X
3	Default	Integer	5	Custom\Integer X	Cust Int X
4	Bus	String	3	My Caption,Another Caption,Extra Caption	My Header,Another Header,Extra Header

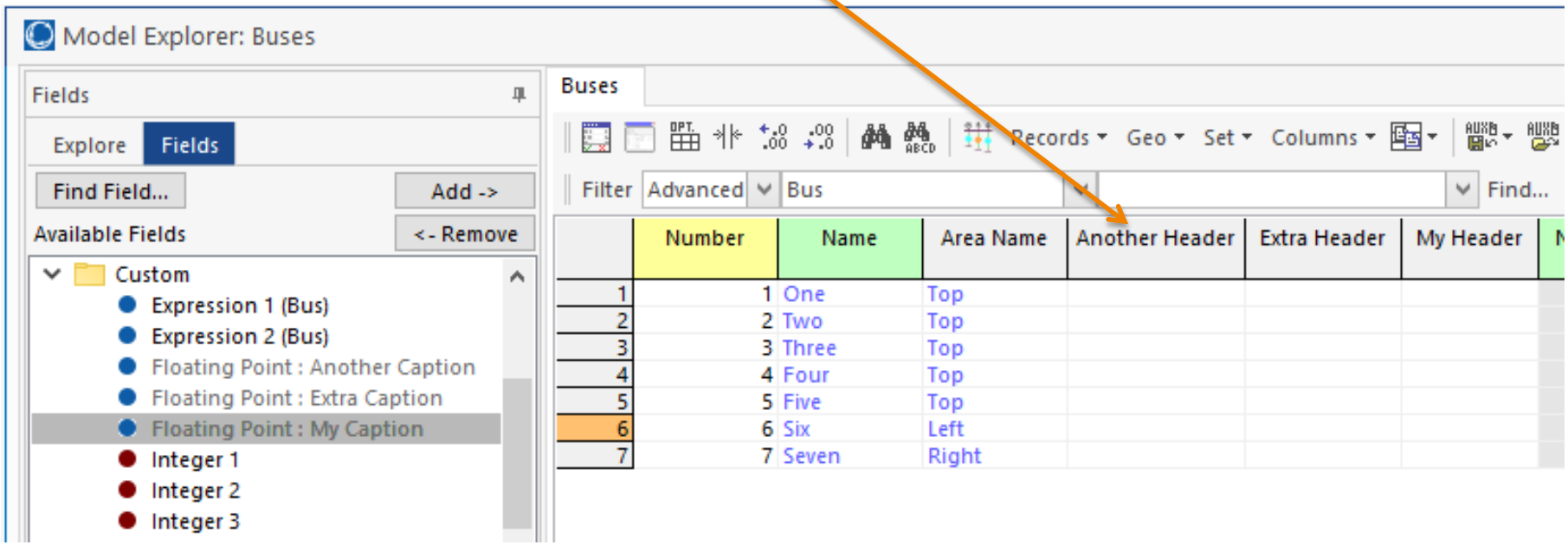
Folder view of available bus fields becomes



© 2019 PowerWorld Corporation

Custom Field Descriptions

Column headings reflect the user-defined Header Captions

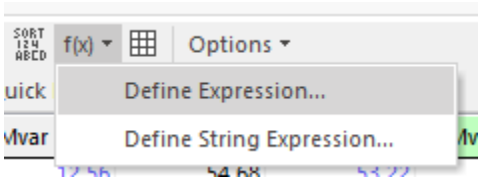


© 2019 PowerWorld Corporation

Expressions

- Custom Expressions (Custom String Expressions)
  - Allow calculation to be performed on fields of the object type for which it is defined
  - Model Field, Model Expression, and Model String Expression fields can also be included in the expression function
- Model Expressions (Model String Expressions)
  - Allow calculation on specific fields of specific objects
- Provide a way of performing calculations and updating data within Simulator
- Useful with contingency actions and remedial actions for specifying the value of the applied action

© 2019 PowerWorld Corporation



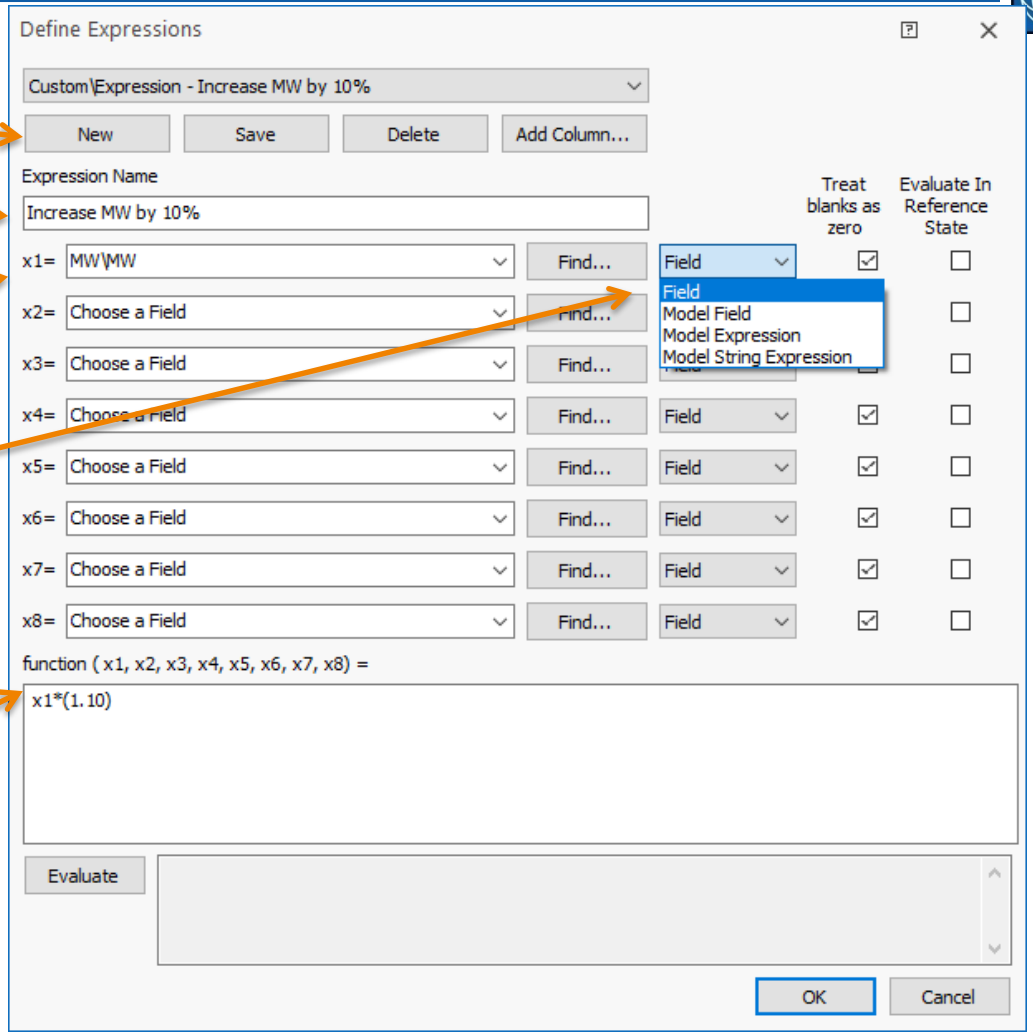
Expressions

Click **New** to add a new expression and then give it a meaningful name

Assign fields to expression variables

Fields can be a **Field** of the particular object type or access fields for particular model objects

Define the function for the expression



https://www.powerworld.com/WebHelp/#MainDocumentation_HTML/Functions_and_Operators_Available.htm

© 2019 PowerWorld Corporation

Expressions

- Example of increasing load by 10%
  - Create expression *Increase MW by 10%*
  - Update the load by the result of the expression using script command

```
SetData(Load, [MW], ["@CustomExpression"], ALL);

SetData(Load, [MW], ["@CustomExpression:Increase MW by
10%"], ALL);
```

© 2019 PowerWorld Corporation

Supplemental Data

- Original intent was to allow extra information to be stored that is associated with display objects
- Can be used as a user-defined container object similar to other aggregation objects like areas, zones, substations, etc.
  - Details of this will be left for other discussions
- Within the context of script commands and auxiliary files can be used as a place to store user-defined variables
  - Reference these using the **Special Keywords in Script Commands** syntax, **Specifying Field Values in Script Commands** syntax, and as part of Custom Expressions and Model Expressions

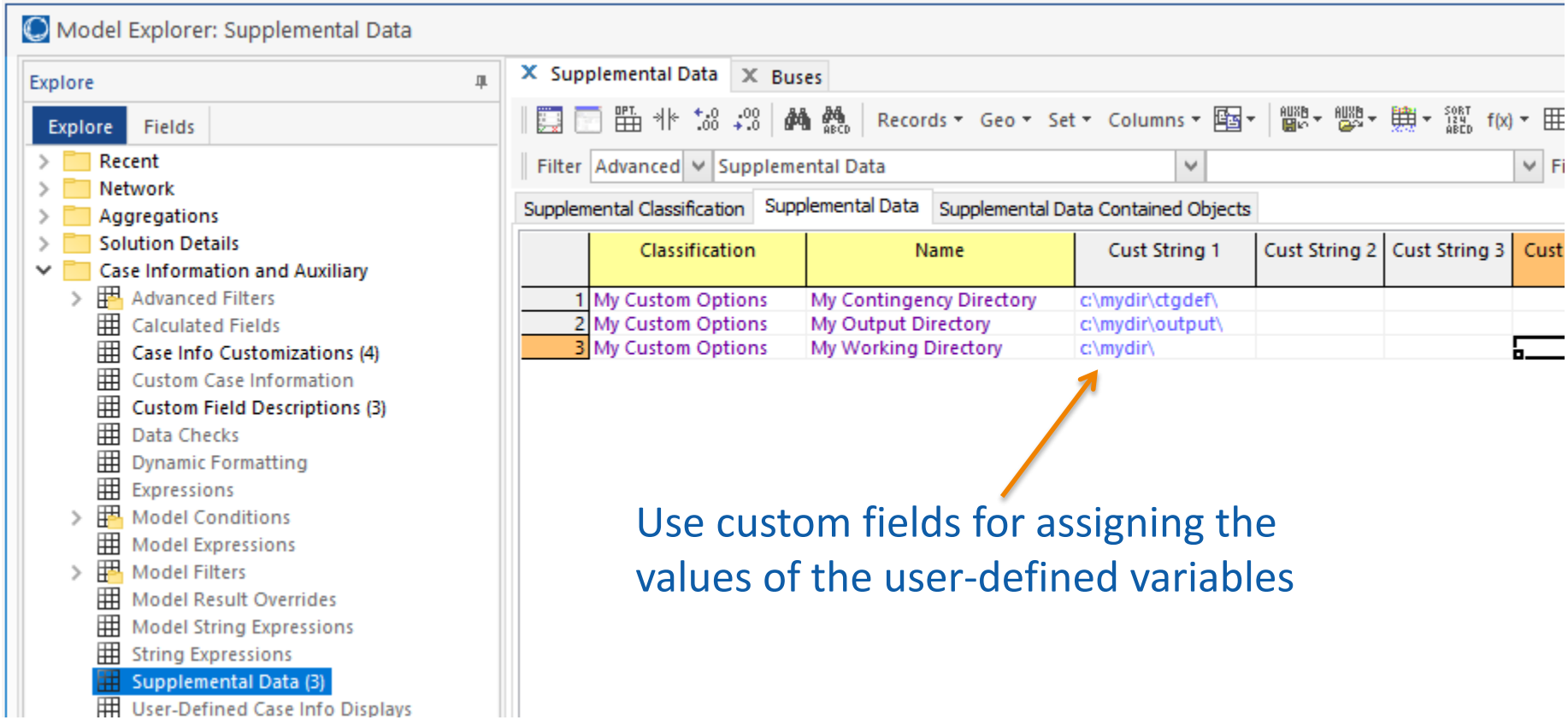
© 2019 PowerWorld Corporation

Supplemental Data

- Two objects need to be defined
  - **Supplemental Classification**
    - This is used as the category to group the data
    - Example – *My Custom Options*
  - **Supplemental Data**
    - Individual pieces of information that belong to a Supplemental Classification
    - Assign to a Supplemental Classification and provide a Name
    - Example – **Classification** = *My Custom Options*, **Name** = *My Working Directory*

© 2019 PowerWorld Corporation

Supplemental Data



Use custom fields for assigning the values of the user-defined variables

© 2019 PowerWorld Corporation

Special Keywords in Script Commands

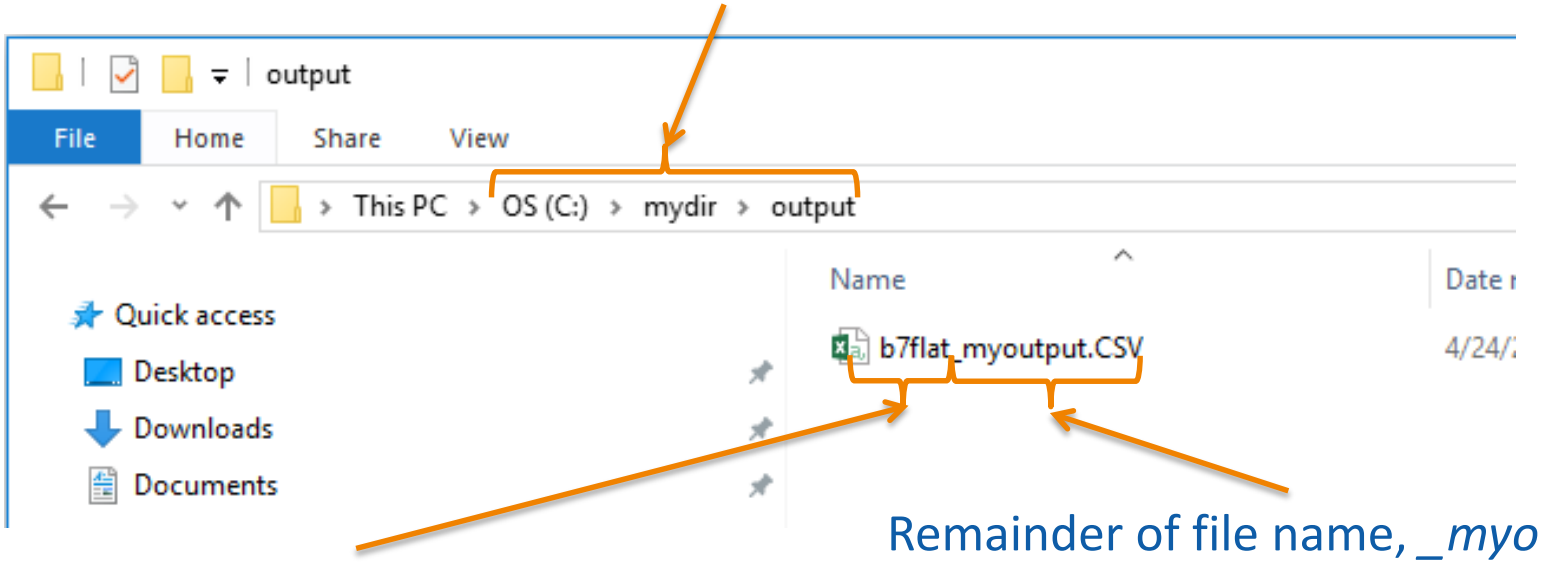
- Allowed as part of file name parameters and other text parameters
  - Generally, these cannot be used for specifying the values to set for specific fields
- The following keywords will be replaced with their actual value when used by a script command
  - `@DATETIME`, `@DATE`, `@TIME`, `@BUILDDATE`, `@VERSION`, and `@CASENAME`
  - `@MODELFIELD<objecttype 'key1' 'key2' 'key3' variablename:digits:rod>`

© 2019 PowerWorld Corporation

Special Keywords in Script Commands

```
SaveData("@MODELFIELD<SupplementalData 'My Custom
Options' 'My Output Directory'
CustomString>@CASENAME_myoutput.CSV", CSV, Bus,
[Number, NomkV, Vpu, Vangle], [],,[],NO);
```

Directory is specified by `@MODELFIELD<SupplementalData 'My Custom Options' 'My Output Directory' CustomString>`



@CASENAME includes *b7flat* from *b7flat.pwb*

Remainder of file name, *_myoutput.CSV*, is remainder of the script command parameters that is not a keyword

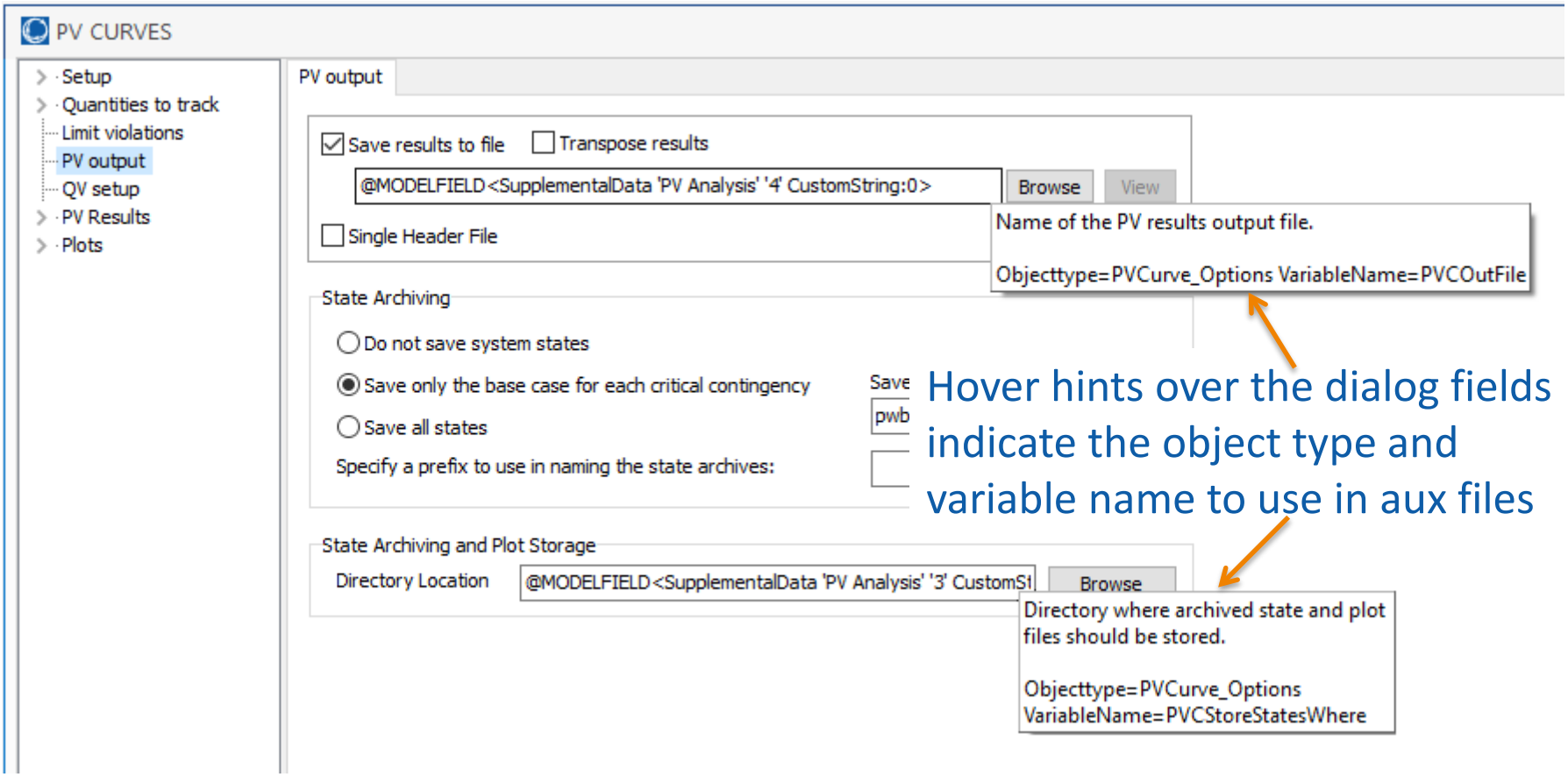
© 2019 PowerWorld Corporation

Special Keywords in Field Values

- These are the exceptions because generally the special keywords are only used in file names and text fields as parameters in script commands
- The fields where this is allowed currently only include fields that specify output file names or directories
- The keywords will be converted at the time that the fields are accessed to determine the name of the directory or file
- The following fields can include the special keywords (these will also show up in relevant dialog fields in the GUI)
  - Transient_Options: ExpDirectory
  - PVCurve_Options: PVCOutFile, PVCStoreStatesWhere
  - QVCurve_Options: QVOutputDir
  - CTG_Options: CTGPostSolAuxFile, PostPostAuxFile, CTGResultStorageFile:1
  - Contingency: PostCTGAuxFile
  - Sim_Environment_Options: SEOSpecifiedAUXFile:0, SEOSpecifiedAUXFile:1, SEOSpecifiedAUXFile:2
  - MessLog_Options: LogAutoFileName

© 2019 PowerWorld Corporation

Special Keywords in Field Values Example




```
PVCurve_Options (PVCOutFile,PVCStoreStatesWhere)
{
"@MODELFIELD<SupplementalData 'PV Analysis' '4' CustomString:0>" "@MODELFIELD<SupplementalData 'PV Analysis' '3' CustomString:0>"
}
```

© 2019 PowerWorld Corporation

Specifying Field Values

- When setting the value of a field other fields can be referenced by using special formatting
- The following can be used in script commands and data sections
  - `"@variablename:location:digits:decimals"`
    - Sets the field to another field within the same object
  - `"&ModelExpressionName:digits:decimals"`
    - Sets the field to the result of a model expression
  - `"&objecttype 'key fields' variablename:location:digits:decimals"`
    - Sets the field to the value of the specified field for the specified object

© 2019 PowerWorld Corporation

Specifying Field Values Examples

```
SetData(Load, [MW], ["@CustomExpression"], ALL);
```

Set the MW value of all loads to the custom expression result for that load

```
Load (BusNum,BusName,ID,MW)
{
     2 "Two"   "1 " "@CustomExpression"
     3 "Three" "1 " "@CustomExpression"
     4 "Four"  "1 " "@CustomExpression"
     5 "Five"  "1 " "@CustomExpression"
     6 "Six"   "1 " "@CustomExpression"
     7 "Seven" "1 " "@CustomExpression"
}
```

```
SetData(Load, [MW], ["&Gen 2 '1' MW:8:2"], "<Device>Load 2 '1'");
```

Set the MW value of the load at bus 2 with ID = 1 to the MW output of the generator at bus 2 with ID = 1

```
Load (BusNum,BusName,ID,MW)
{
     2 "Two" "1 " "&Gen 2 '1' MW:8:2"
}
```

© 2019 PowerWorld Corporation

Filtering

- Many script commands take a filter as input to determine the objects on which the command acts
- Valid filter parameters are typically
  - "*FilterName*"
    - Name of Advanced Filter, Device Filter, or Secondary Filter (filtering across object types)
  - AREAZONE
    - Filter based on the area/zone/owner filter
  - SELECTED
    - Filter based on the Selected field = YES
  - ALL
    - Special filter for some script commands to include all objects of a specified object type

© 2019 PowerWorld Corporation

Filtering

- Device Filter
  - Instead of creating an Advanced Filter to return a particular object, reference the object directly through the device filter syntax
    - "<DEVICE>objecttype 'key1' 'key2' 'key3’”

```
SetData(Load, [MW], ["&Gen 2 '1' MW:8:2"], "<Device>Load 2 '1'");
```

Set the MW value of the load at bus 2 with ID = 1 to the MW output of the generator at bus 2 with ID = 1

© 2019 PowerWorld Corporation

Filtering

- Filtering Across Object Types (Secondary Filter)
  - Reference an Advanced Filter for a different object type than the object being filtered
  - Allows reuse of filters
    - Example: define a bus filter and then use this to filter generators, loads, switched shunts, branches, or any other object that connects to a bus
    - If the object being filtered contains more than one of the filter object type OR is assumed
      - If a bus object is being filtered based on an area filter, the bus meets the filter if the area of the bus meets the filter
      - If an area object is being filtered based on a bus filter, the area meets the filter if ANY single bus in the area meets the filter
  - "<objecttype>filtername"

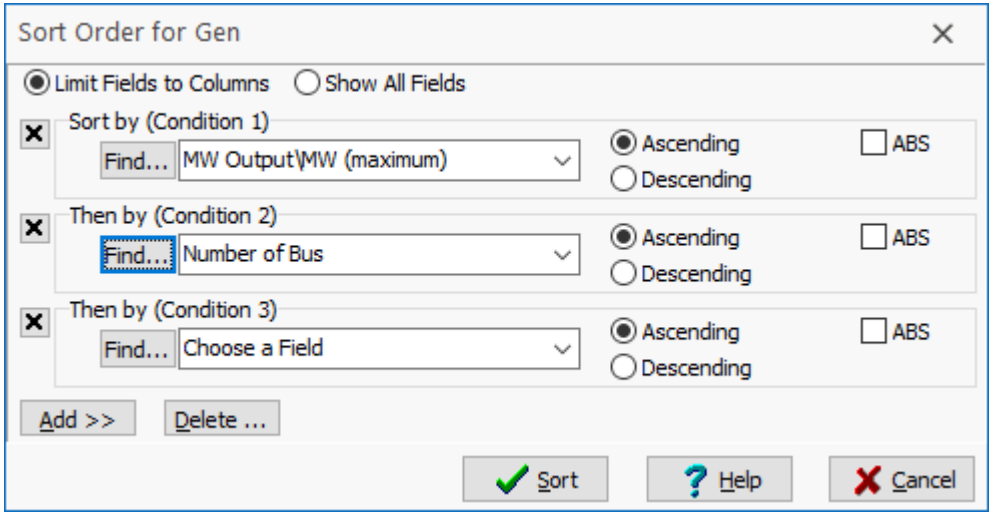
```
SetData(Load, [MW], ["@CustomExpression"], "<Bus>Nom kV > 138");
```

Set the MW value of a load if the terminal bus of the load meets the filter

© 2019 PowerWorld Corporation

Sorting

- Click on the **Advanced Sort** toolbar button to open the Advanced Sorting dialog for a case information display



© 2019 PowerWorld Corporation

Sorting

- Same sorting that can be applied to case information displays can be applied when saving data to file
  - Sort case information display and then use **Save As** options for the display
  - Specify a sort to determine the correct syntax for defining sort in script commands

```
SaveData("filename",filetype,objecttype,[fieldlist],
[subdatalist],filter,[SortFieldList],Transpose);
```

[SortFieldList] with format `[variablename1:+:0, variablename2:-:1]`

+ to sort ascending

- to sort descending

0 for case insensitive or no absolute value

1 for case sensitive or absolute value

© 2019 PowerWorld Corporation

Sorting

- **Sorted by** field in **Case Info Customizations** will contain the string to include in the script command

Case Info Customizations

Records Set Columns Options

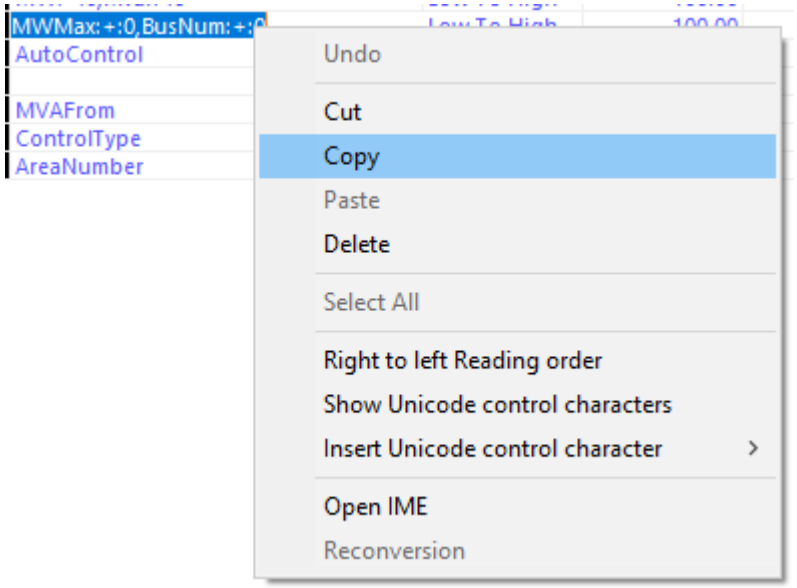
Filter Advanced Data Grid Find... Remove Quick Filter

	Name	Use Filters?	Filter Name	Nondefault Font?	Row Height	Font Name	Font Styles	Font Size	Font Color	Sorted by	Sort Dir	Zoom Level	Frozen Columns
1	TSModalModeDetail	YES		NO	13	Segoe UI		8		MagnitudeNormalized	High To Low	100.00	-1
2	GICXFormer	YES		NO	13	Segoe UI		8		GICQLosses	High To Low	100.00	-1
3	Area	YES		NO	13	Segoe UI		8		Number	Low To High	100.00	-1
4	AreaFilter	YES		NO	13	Segoe UI		8		Number	Low To High	100.00	-1
5	SuperArea	YES		NO	13	Segoe UI		8			High To Low	100.00	-1
6	TSGenEigenAnalysisGrid	YES		NO	13	Segoe UI		8		BusName	Low To High	100.00	-1
7	Bus	YES		NO	13	Segoe UI		8		Vpu	High To Low	100.00	-1
8	Load	YES		NO	13	Segoe UI		8		MW:+:0,Mvar:+:0	Low To High	100.00	-1
9	Gen	YES		NO	13	Segoe UI		8		MWMax:+:0,BusNum:+:0	Low To High	100.00	-1
10	Shunt	YES		NO	13	Segoe UI		8		AutoControl	Low To High	100.00	-1

© 2019 PowerWorld Corporation

Sorting

- It is always better to let Simulator do the formatting
  - Double click on the **Sorted by** field entry – this will select the field for editing directly within the case information display
  - Right click and choose **Copy** from the local menu




```
SaveData("MyGenerators.CSV",CSVColHeader,Gen,[BusNum,ID,MW,
Mvar,MWMax,MWMin,AGC],[],,[MWMax:+:0,BusNum:+:0],NO);
```

© 2019 PowerWorld Corporation

Very Useful Script Commands

- **`SelectAll`**`(objecttype, filter);`
  - Some script commands require that a filter be specified
  - This will set the *Selected* field for the specified `objecttype` to YES and allow the special SELECTED filter to be used
- **`UnSelectAll`**`(objecttype, filter);`
  - This will set the *Selected* field for the specified `objecttype` to NO to reset if for using the SELECTED filter on new selections
  - This will appear in some auxiliary files created from Simulator such as those created with the Difference Flows tool

© 2019 PowerWorld Corporation

Options by Value

- Transpose of options object type that often provides are more desirable format

```
Sim_Solution_Options
(CloseCBToEnergizeShunts,DCMvarComp,DCPhaseIgnore,DCXFCorrIgnore,DCModelType,
   DCApprox,DCLossComp,EconDispEnforceConvex,EconDispPenaltyFactors,
   CTGInterfaceEnforcement,DisableAngleRotation,DisableAngleSmoothing,
   DisableRestoreSolution,DisableRestoreState,DisableXFTapWrongSign,
   DynAssignSlack,EvalSolutionIsland,LossSenseFunc,MVABase,MVAConvergenceTol,
   PUConvergenceTol,DisableOptMult,DoOneIteration,FlatStartInit,MaxItr,
   MinVoltILoad,MinVoltSLoad,MWAGCIGOnlyAGCAble,MWAGCIGEnforceGenMWLimits,
   MWAGCIGPositiveLoad,MWAGCIGName,AGCToleranceMVA,AGCTolerance,MWAGCAreaIsland,
   MWAGCIGQMult,MWAGCIGPowerFact,MWAGCIGUseContantPF,LogID,LogMWColor,
   LogMvarColor,LogOPFLPColor,LogLTCColor,LogPhaseColor,LogShuntColor,LogNomkV,
   Show,LogMWSuppress,LogMvarSuppress,LogOPFLPSuppress,LogLTCSuppress,
   LogPhaseSuppress,LogShuntSuppress,ChkMWAGC,EnforceGenMWLimits,
   ConsolidationUse,LTCTapBalance,ChkDFACTS,ChkPhaseShifters,ChkSVCs,ChkShunts,
   ChkTaps,DisableGenMVRCheck,ChkVarBackoffImmediately,ChkVarImmediately,
   MaxItrVoltLoop,MinLTCSense,ModelPSDiscrete,PreventOscillations,
   MvarSharingAllocation,ShuntInner,TransformerStepping,ZBRThreshold)
{
"NO " "NO " "YES" "YES" RIgnore "NO " "NO " "NO " "YES" Never "NO " "NO " "NO " "NO "
"YES" "NO " "NO " "None" "100.0000000" 0.1000000 0.0010000 "NO " "NO " "NO "      50
0.5000000 0.7000000 "NO " "NO " "NO " "" 5.0000001 0.0500000 "Use Area/SuperArea ACE"
1.0000000 1.0000000 "NO " "Numbers" 16711680 8388736 16711680   32768 8388863   33023
"NO " "NO " "YES" "NO " "YES" "NO " "NO " "NO " "YES" "NO " "NO " "YES" "NO " "YES"
"YES" "YES" "YES" "NO " "YES" "NO "      20 0.0500000 "NO " "NO " "RegPerc" "YES"
"Coordinated" "0.0002"
}
```

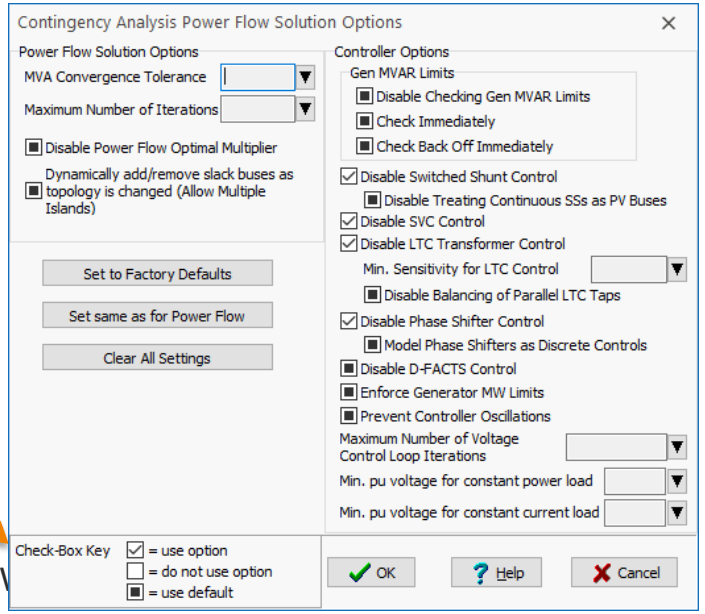
```
Sim_Solution_Options_Value (Option,Value)
{
"AGCToleranceMVA"                    "5"
"ChkDFACTS"                          "NO"
"ChkMWAGC"                           "YES"
"ChkPhaseShifters"                   "YES"
"ChkShunts"                          "YES"
"ChkSVCs"                            "YES"
"ChkTaps"                            "YES"
"ChkVarBackoffImmediately"           "YES"
"ChkVarImmediately"                  "NO"
"CloseCBToEnergizeShunts"            "NO"
"ConsolidationUse"                   "NO"
"CTGInterfaceEnforcement"            "Never"
"DCApprox"                           "NO"
"DCLossComp"                         "NO"
"DCModelType"                        "RIgnore"
"DCMvarComp"                         "NO"
"DCPhaseIgnore"                      "YES"
"DCXFCorrIgnore"                     "YES"
"DisableAngleRotation"               "NO"
"DisableAngleSmoothing"              "NO"
"DisableGenMVRCheck"                 "NO"
"DisableOptMult"                     "NO"
"DisableRestoreSolution"             "NO"
"DisableRestoreState"                "NO"
"DisableXFTapWrongSign"              "YES"
…
}
```

© 2019 PowerWorld Corporation

Options by Value
Contingency Options

- In the case of contingency options, options by value (CTG_Options_Value) provides access to fields that are not individually available in other format (CTG_Options)
  - Accessing contingency analysis power flow solution options is much easier using options by value

Default power flow solution options are used unless the option has been changed for contingency analysis



© 2019 PowerW

Contingency Options

CTG Options | Sim Solution Options Value | Sim Solution Options | CTG Options Value

Records | Set | Columns | f(x) | Optio

Filter: Advanced | CTG_Options | Find... | Remove | Quick Fil

	CTG Solution Options
1	ChkShunts = NO, ChkShunts:1 = NO, ChkTaps = NO, ChkPhaseShifters = NO

CTG Options by Value | CTG Options

Records | Set | Columns | f(x) | Options

Filter: Advanced | CTG_Options_Value | Find... | Remove | Quick Filter

CTG_Options_Value	FieldFolder	FieldName	Option (1)	Value
63	Screening	Number of Buses	ScreenNumBus	20
64	Screening	Store Voltage Violations	ScreenVoltStoreViol	NO
65	Solution Options	Voltage Control Loop\Check Phase Shifters	ChkPhaseShifters	NO
66	Solution Options	MW Control Loop\Enforce Generator MW limits	EnforceGenMWLimits	Default
67	Solution Options	General\Dynamically assign slack buses (Allow Multiple I	DynAssignSlack	Default
68	Solution Options	Inner Power Flow Loop\Disable Optimal Multiplier	DisableOptMult	Default
69	Solution Options	Voltage Control Loop\Switched Shunts in Inner Power F	ShuntInner	Default
70	Solution Options	Voltage Control Loop\Disable Gen MVAR Check?	DisableGenMVRCheck	Default
71	Solution Options	Inner Power Flow Loop\Convergence Tolerance in PU	PUConvergenceTol	Default
72	Solution Options	Voltage Control Loop\Prevent Controller Oscillations	PreventOscillations	Default
73	Solution Options	Inner Power Flow Loop\Convergence Tolerance in MVA	MVAConvergenceTol	Default
74	Solution Options	Voltage Control Loop\Generate Mvar Check Immediately	ChkVarImmediately	Default
75	Solution Options	Voltage Control Loop\Generate Mvar Check Backoff Imm	ChkVarBackOffImmediately	Default
76	Solution Options	Voltage Control Loop\Model phase-shifters as discrete (	ModelPSDiscrete	Default
77	Solution Options	Inner Power Flow Loop\Minimum PU Voltage for Consta	MinVoltSLoad	Default
78	Solution Options	Inner Power Flow Loop\Minimum PU Voltage for Consta	MinVoltILoad	Default
79	Solution Options	Voltage Control Loop\Minimum LTC Sensitivity	MinLTCSense	Default
80	Solution Options	Voltage Control Loop\Check Transformer Tap Ratios	ChkTaps	NO
81	Solution Options	Voltage Control Loop\Maximum Voltage Control Loop It	MaxItrVoltLoop	Default
82	Solution Options	Inner Power Flow Loop\Max Iterations	MaxItr	Default
83	Solution Options	Voltage Control Loop\Balance LTC Tap Ratios	LTCTapBalance	Default
84	Solution Options	Voltage Control Loop\Check SVCs	ChkSVCs	NO
85	Solution Options	Voltage Control Loop\Check Switched Shunts	ChkShunts	NO

Options by Value
Contingency Options

Default power flow solution options are used unless the option has been changed for contingency analysis; with CTG_Options only the options that are not default are specified in an auxiliary file

```
CTG_Options (CTGSolutionOptions)
{
"ChkShunts = NO, ChkShunts:1 = NO, ChkTaps = NO, ChkPhaseShifters = NO"
}
```

CTG_Options_Value is much easier format to use for changing only one of these options

```
CTG_Options_Value (Option,Value)
{
"ChkPhaseShifters"                    "NO"
"ChkShunts"                           "NO"
"ChkSVCs"                             "NO"
"ChkTaps"                             "NO"
}
```

© 2019 PowerWorld Corporation

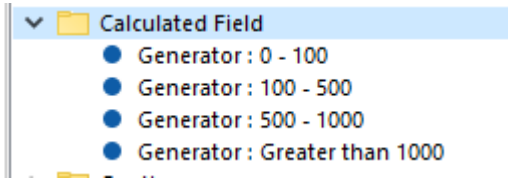
Calculated Fields

- Provide a way of performing an arithmetic operation on a group of objects of a single object type based on a specific field
- This calculation can then be applied to objects of a different object type
- The final results are based on how the object type in the calculated field definition relates to the object type to which the calculation is applied
  - Define a calculation on the branch object type and then apply this to the bus object type. The final result that is displayed with a bus will perform the calculation on all branches that have at least one terminal connected to that bus.

© 2019 PowerWorld Corporation

Calculated Fields Example

- Total Max MW of generators by range
  - Define calculated fields for the ranges of generators: 0 – 100 MW, 100 – 500 MW, 500 – 1000 MW, greater than 1000 MW
- Calculated fields will show up in the list of available fields for all applicable object types



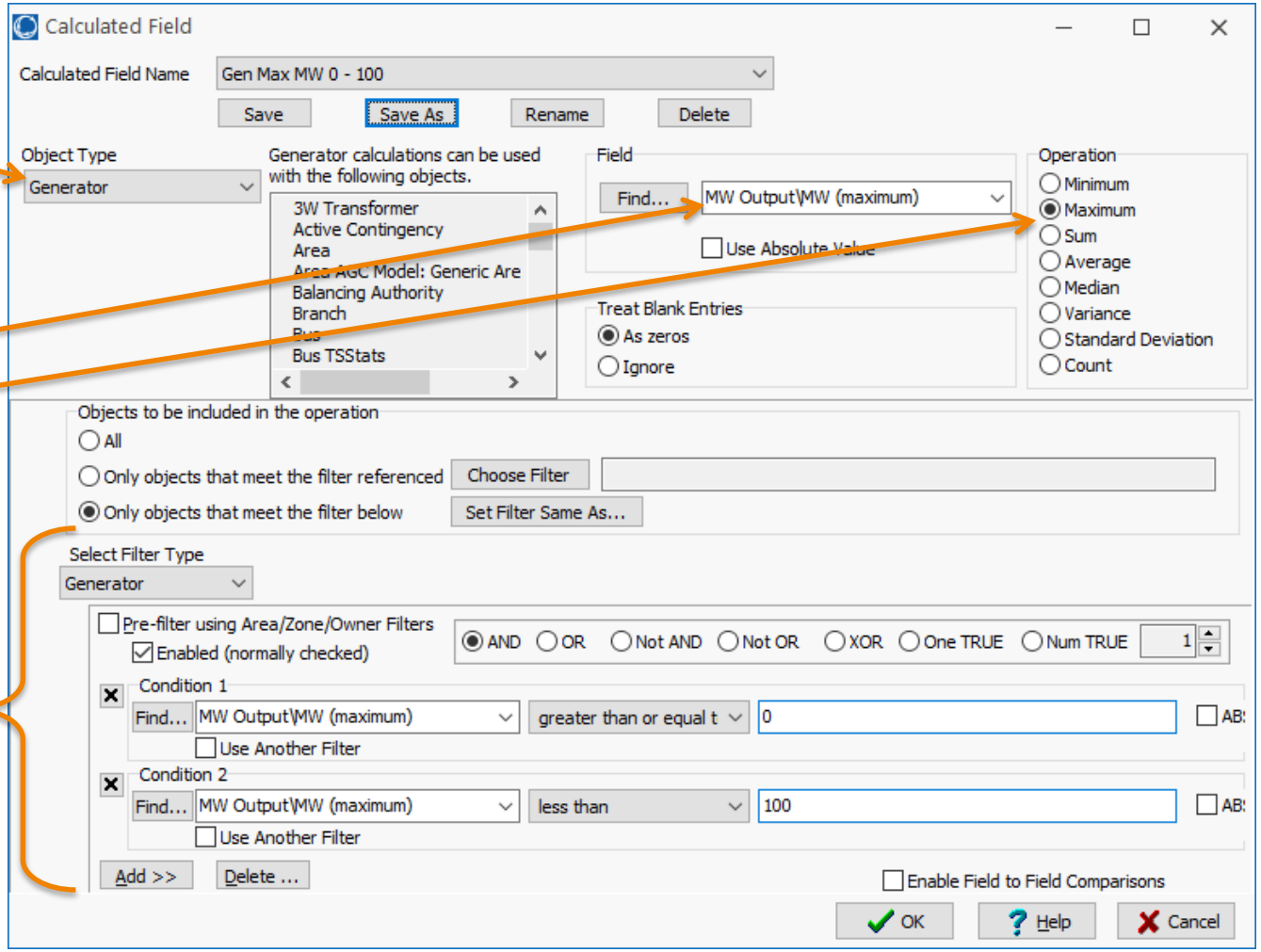
© 2019 PowerWorld Corporation

Calculated Fields Example

Object Type on which the calculation is performed

Field and Operation being applied

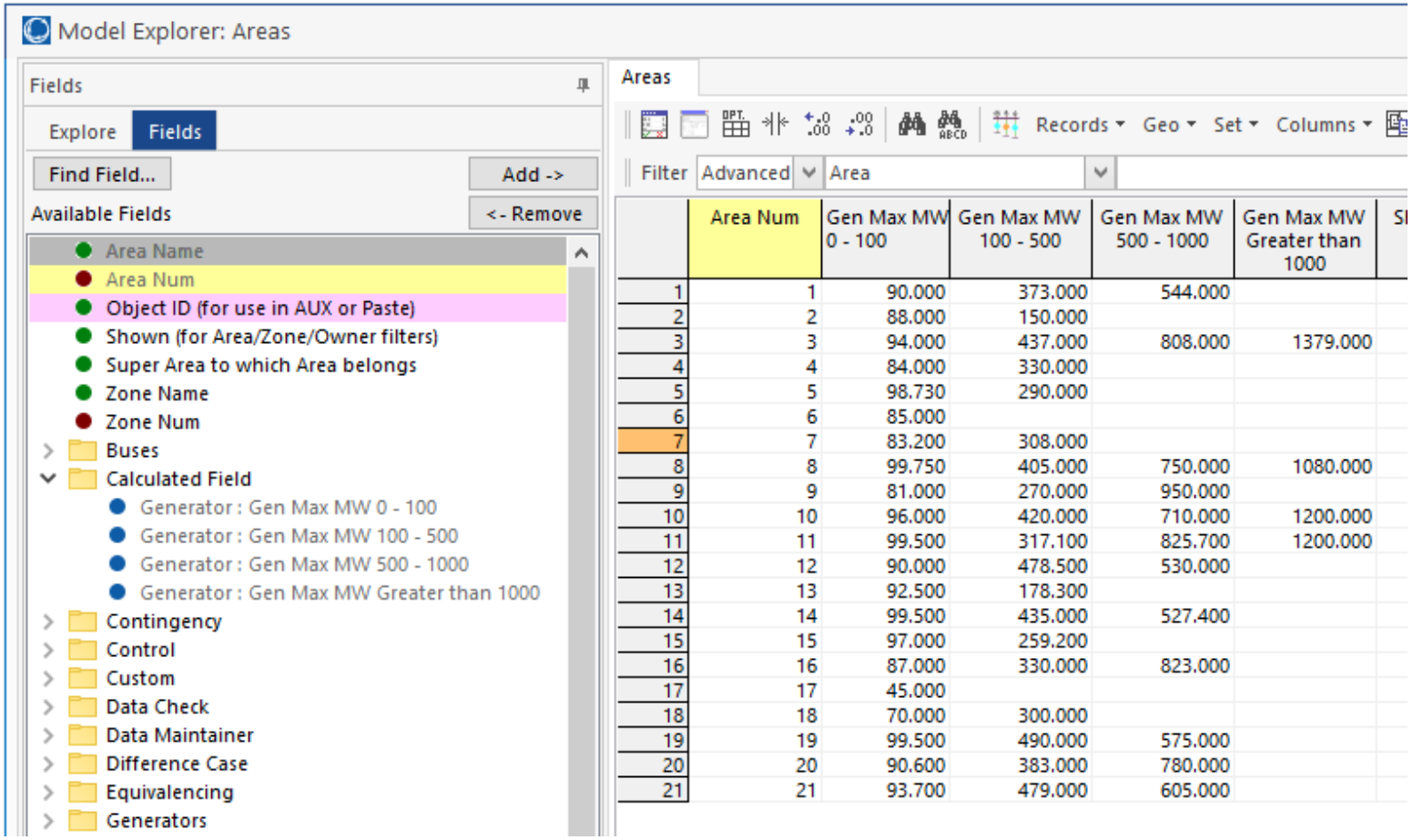
Filter is applied to the objects of Object Type to determine which ones are included in the calculation



© 2019 PowerWorld Corporation

Calculated Fields Example

When used with Area object type, the calculated results give the total Max MW of all generators in each area within the specified MW range

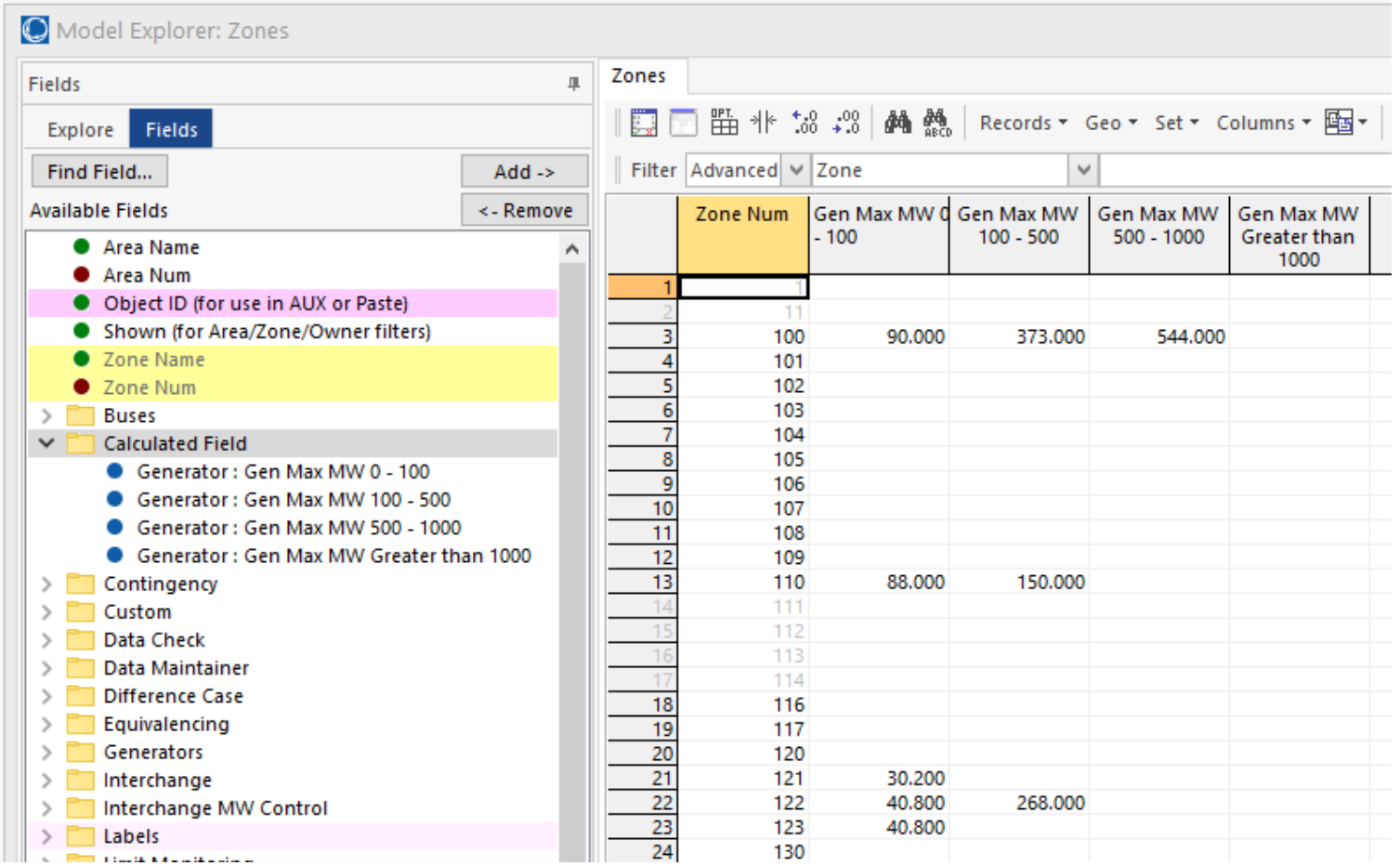


	Area Num	Gen Max MW 0 - 100	Gen Max MW 100 - 500	Gen Max MW 500 - 1000	Gen Max MW Greater than 1000
1	1	90.000	373.000	544.000	
2	2	88.000	150.000		
3	3	94.000	437.000	808.000	1379.000
4	4	84.000	330.000		
5	5	98.730	290.000		
6	6	85.000			
7	7	83.200	308.000		
8	8	99.750	405.000	750.000	1080.000
9	9	81.000	270.000	950.000	
10	10	96.000	420.000	710.000	1200.000
11	11	99.500	317.100	825.700	1200.000
12	12	90.000	478.500	530.000	
13	13	92.500	178.300		
14	14	99.500	435.000	527.400	
15	15	97.000	259.200		
16	16	87.000	330.000	823.000	
17	17	45.000			
18	18	70.000	300.000		
19	19	99.500	490.000	575.000	
20	20	90.600	383.000	780.000	
21	21	93.700	479.000	605.000	

© 2019 PowerWorld Corporation

Calculated Fields Example

When used with Zone object type, the calculated results give the total Max MW of all generators in each zone within the specified MW range



© 2019 PowerWorld Corporation

Calculated Fields

- Other examples
  - Return the buses in the island with the most buses
  - Set the system slack bus based on generator criteria
  - Maximum percent flow on all branches in an area or zone
  - Trip injection group with largest output
  - GIC flows in a substation
  - Losses by owner and nominal kV level
  - PV results with the lowest transfer level

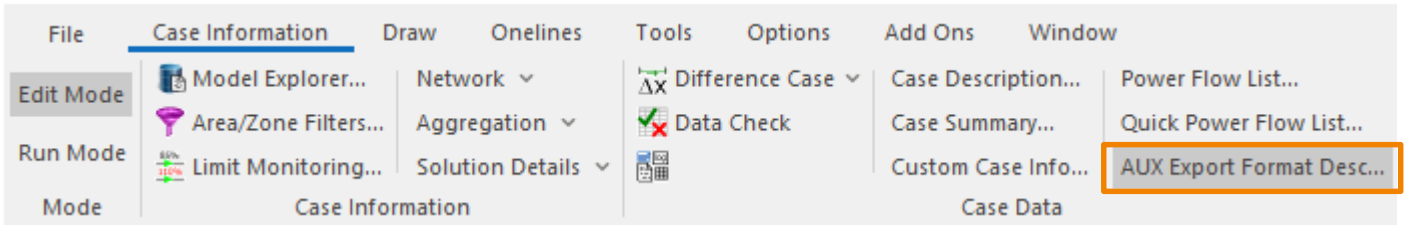
© 2019 PowerWorld Corporation

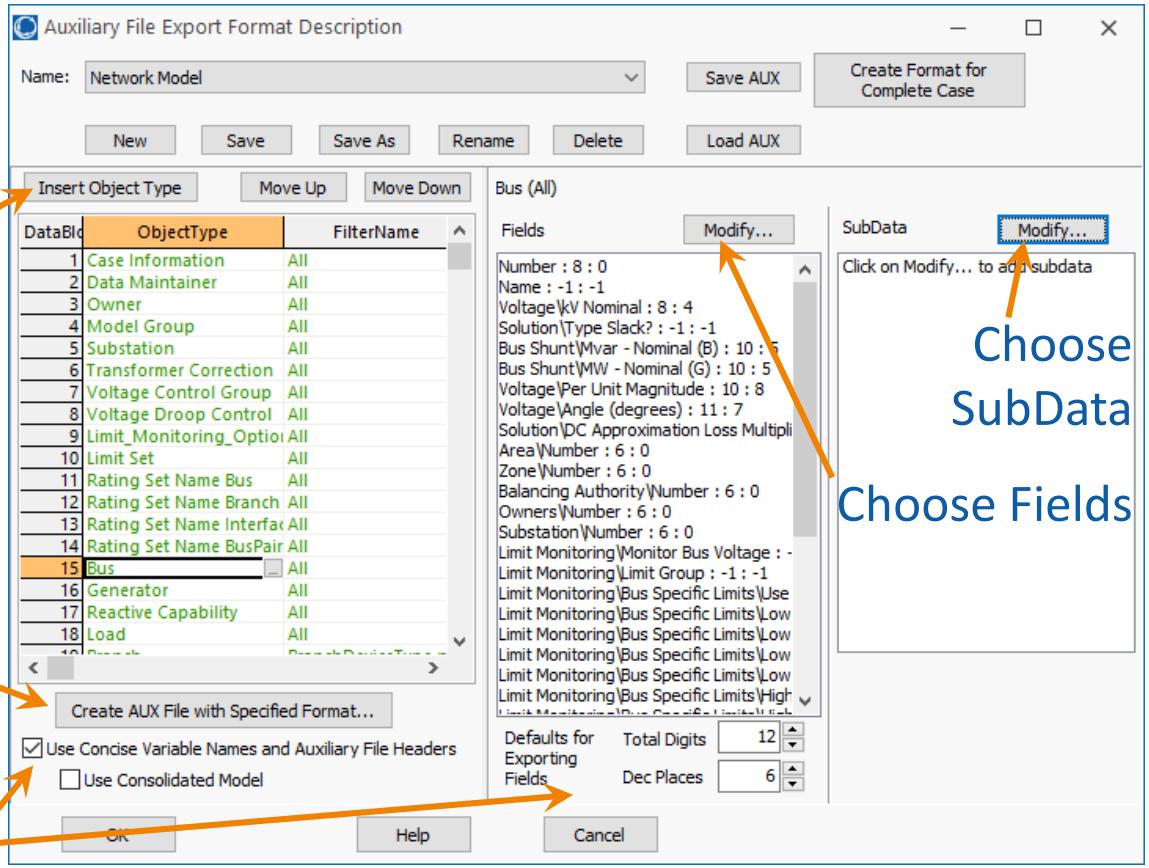
Auxiliary File Export Format Descriptions

- Allows the definition of a list of object types and fields to be saved in DATA sections of an auxiliary file
- These formats themselves can then be saved to an auxiliary file and used whenever needed
- Script command can be used to save an auxiliary file in a defined format
  - `SaveDataUsingExportFormat("filename", filetype, "FormatName", ModelToUse);`

© 2019 PowerWorld Corporation

Auxiliary File Export Format Descriptions





Choose Object Types

Choose SubData

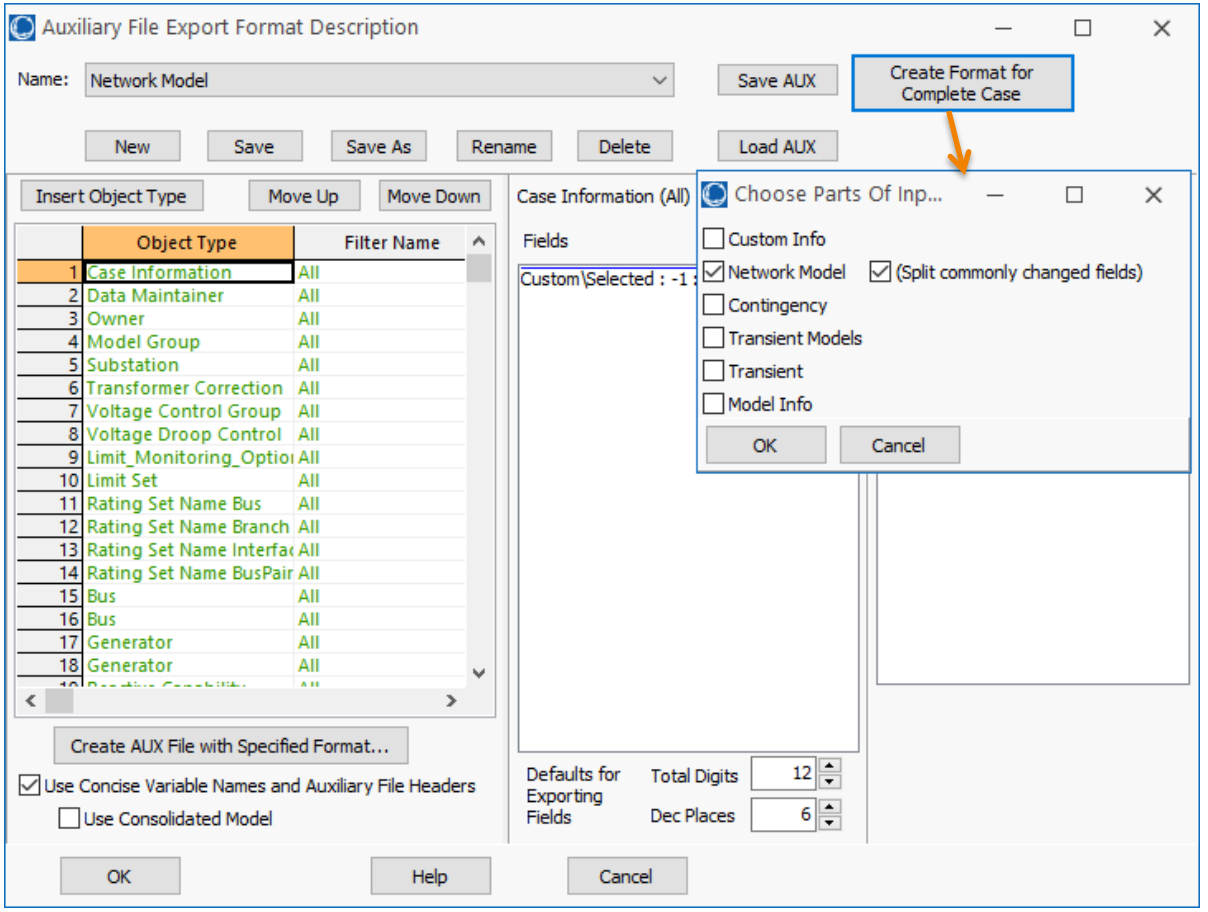
Choose Fields

Save an auxiliary file with the specified data

Set formatting and file type

Auxiliary File Export Format Descriptions

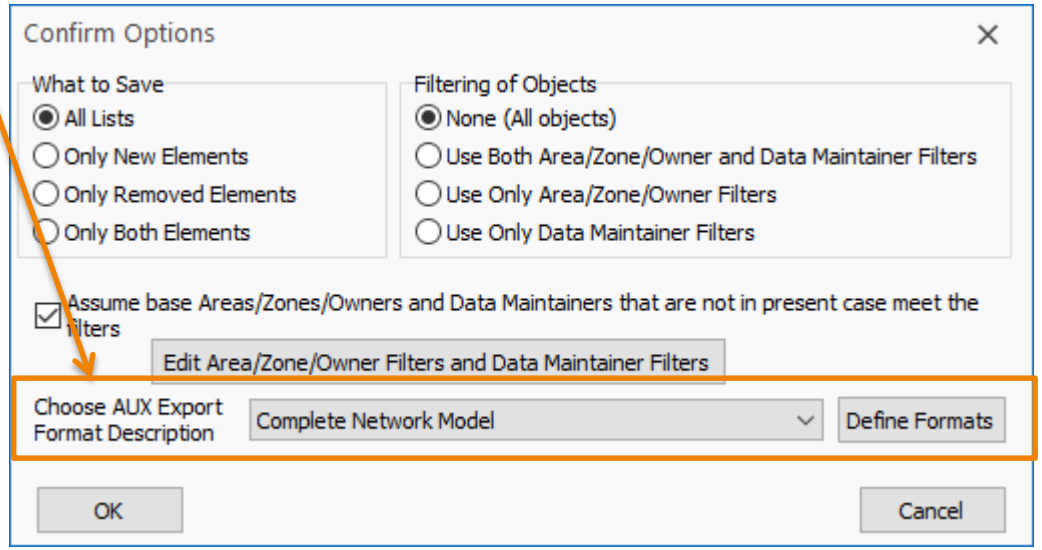
- Default formats can be used as a starting point for customizations



© 2019 PowerWorld Corporation

Auxiliary File Export Format Descriptions

- AUX export formats can be used with Present Case Topological Differences from the Base Case dialog to customize fields and objects saved



© 2019 PowerWorld Corporation

User Interface During Scripting

- Auxiliary file scripting is a batch process with no looping structure or condition checks
- The intention is to load an auxiliary file and walk away
- There are a few script commands and special syntax that allow user interaction with a GUI dialog during the scripting

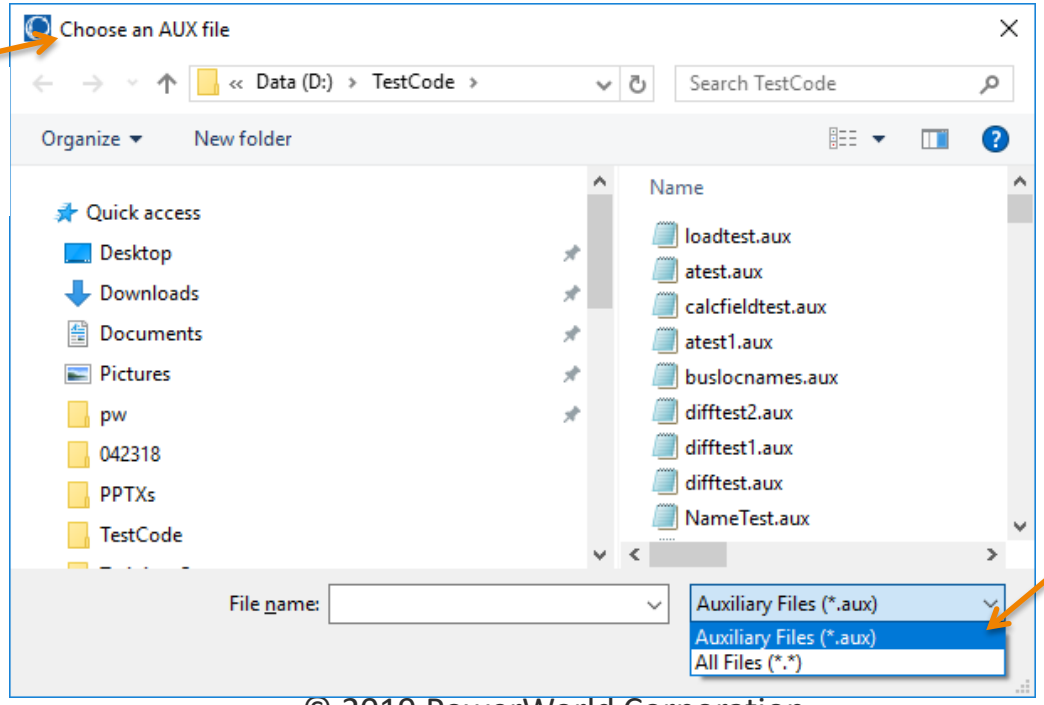
© 2019 PowerWorld Corporation

GUI During Scripting

- Special syntax within the `filename` parameter in script commands will open a dialog to choose a file
  - `"<PROMPT 'Caption' 'FileTypes' 'InitialDirectory'>"`

```
LoadAux("<PROMPT 'Choose an AUX file' 'Auxiliary Files
(*.aux)|*.aux|All Files (*.*)|*.*' 'D:\TestCode'>");
```

Specified **Caption**



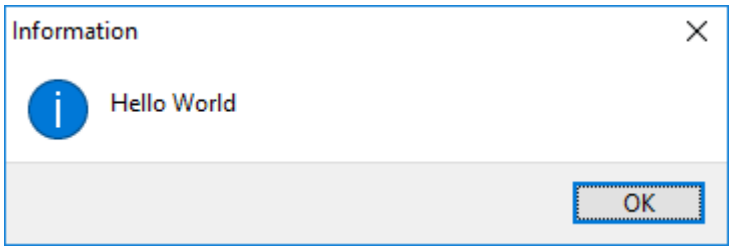
Specified **FileTypes**

© 2019 PowerWorld Corporation

GUI During Scripting

- `MessageBox("text");`
  - Provide a simple text message to the user with no user input to the scripting

```
MessageBox("Hello World");
```



© 2019 PowerWorld Corporation

GUI During Scripting

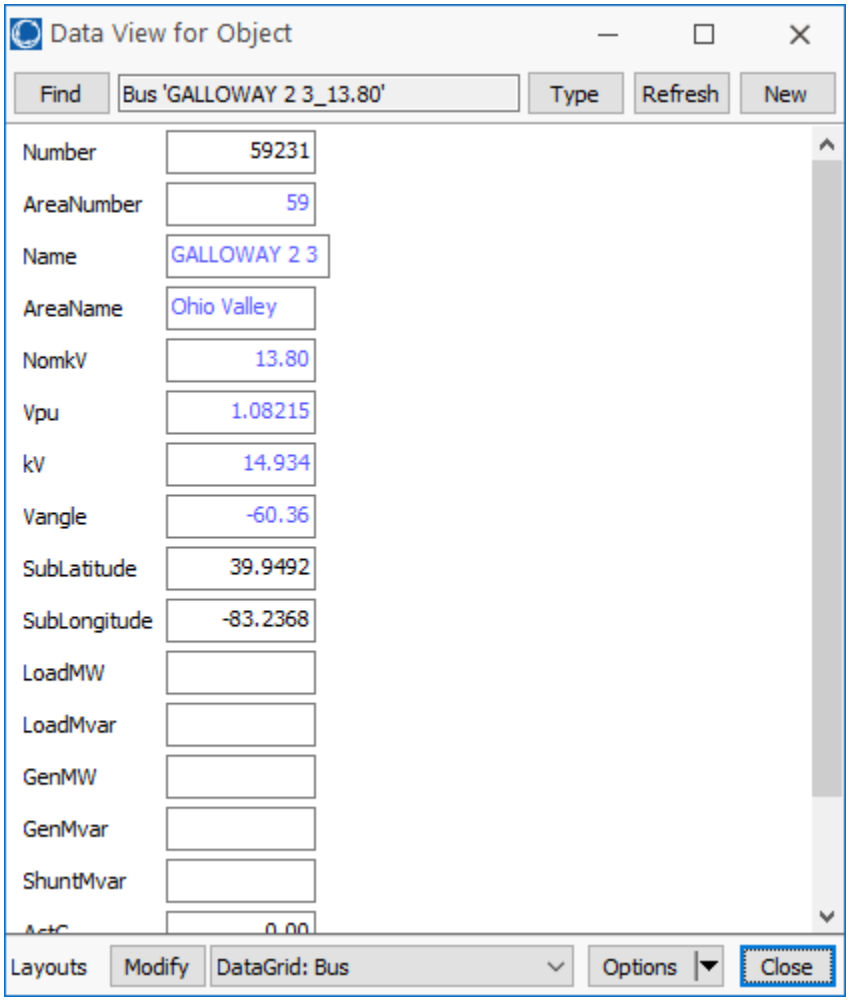
- `OpenDataView("ObjectIDString", "DataGridIDString");`
  - Use the Data View feature to display a dialog during scripting that will allow fields to be changed for a single object in a manner similar to how they are changed in a case information display
  - See the help documentation for full details on how to customize these data views
    - https://www.powerworld.com/WebHelp/Default.htm#MainDocumentation_HTML/Data_View.htm

© 2019 PowerWorld Corporation

GUI During Scripting

```
OpenDataView("Bus 59231", "DataGrid 'Bus'");
```

- Opens a dialog containing the same fields that are displayed in the Bus case information display in the Model Explorer
- Color coding for field values is the same as in a case information display
- User can change values for the fields that can be edited



© 2019 PowerWorld Corporation

GUI During Scripting

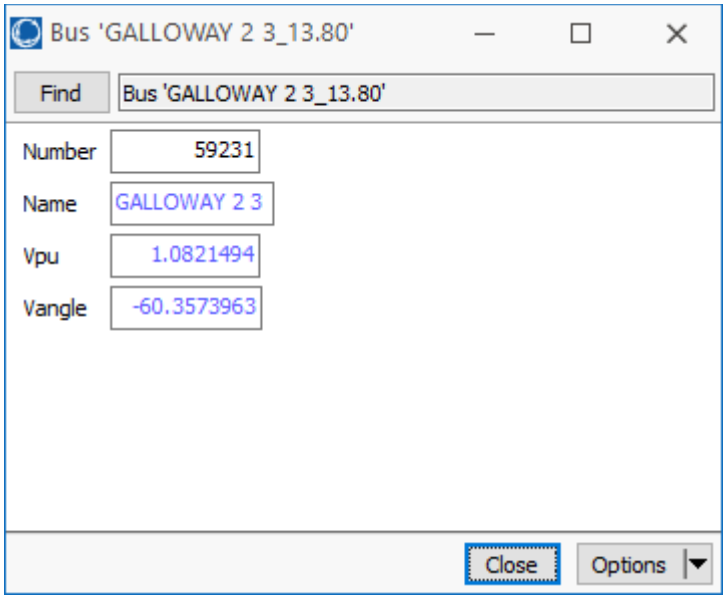
- `ObjectFieldsInputDialog("ObjectIDString",[fieldlist], lots of optional parameters);`
  - Create a custom dialog for displaying fields for a particular object without having to define a DataGrid object
  - See help documentation for details on how to specify the `lots of optional parameters`
    - https://www.powerworld.com/WebHelp/Content/Other_Documents/Auxiliary-File-Format.pdf

© 2019 PowerWorld Corporation

GUI During Scripting

```
ObjectFieldsInputDialog("Bus 59231",[Number, Name,
Vpu, Vangle]);
```

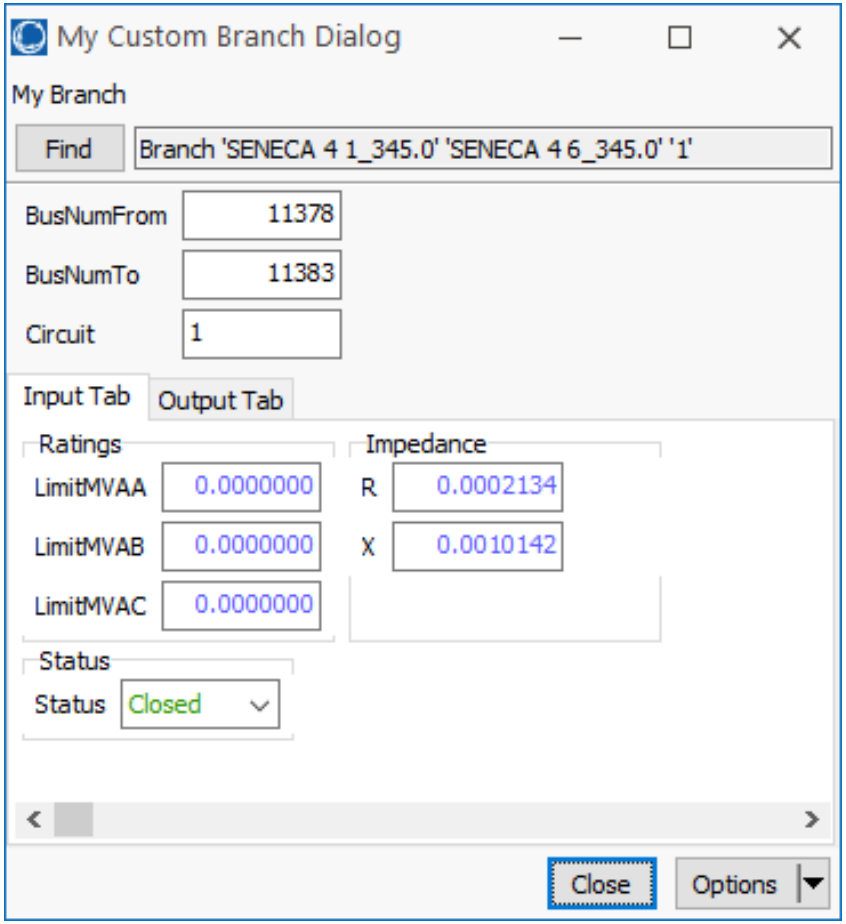
- Opens a dialog containing the fields defined in the script command
- Color coding for field values is the same as in a case information display
- User can change values for the fields that can be edited



© 2019 PowerWorld Corporation

GUI During Scripting

```
ObjectFieldsInputDialog("Branch
11378 11383 1", [BusNumFrom,
BusNumTo, Circuit, LimitMVAA,
LimitMVAB, LimitMVAC, R, X,
Status, MWFrom, MWTo, MvarFrom,
MvarTo, MVAFrom, MVATo], "My
Custom Branch Dialog", "My
Branch", [], [3, 9], ["Input
Tab","Output Tab"], [3,8,9,13],
["Ratings","Status","MW","MVA"],
[6,11], ["Impedance","Mvar"]);
```



© 2019 PowerWorld Corporation