Auxiliary File Format for Simulator 22

Last Updated: January 25, 2023

PowerWorld Corporation
2001 South First St
Champaign, IL 61820
(217) 384-6330
http://www.powerworld.com
info@powerworld.com

Table of Contents

Introduction ***1***

SCRIPT Section ***2***

Using Filters in Script Commands **2**

Specifying Special Keywords in Script Command Parameters **3**

Specifying File Names in Script Commands **3**

Specifying Field Variable Names in Script Commands **4**

Specifying Field Values in Script Commands **4**

General Actions **5**

Program Actions 5

CopyFile("oldfilename", "newfilename"); 5

DeleteFile("filename"); 5

ExitProgram; 5

LogAdd("text"); 5

LogAddDateTime("label", includedate, includetime, includemilliseconds); 5

LogClear; 5

LogSave("filename", AppendFile); 5

LogShow(DoShow); 6

RenameFile("oldfilename", "newfilename"); 6

SetCurrentDirectory("filedirectory", CreateIfNotFound); 6

StopAuxFile; 6

WriteTextToFile("filename", "text"); 6

Data Actions 7

CreateData(objecttype, [fieldlist], [valuelist]); 7

Delete(objecttype, filter); 7

DeleteDevice([ObjectIDString]); 7

DeleteIncludingContents(objecttype, filter); 7

ExportAreaSupplyCurves("filename", "User Defined String", NumPoints); 8

ImportData("filename", FileType, HeaderLine, CreateIfNotFound); 8

LoadAux("filename", CreateIfNotFound); 8

LoadAuxDirectory("filedirectory", "filterstring", CreateIfNotFound); 8

LoadCSV("filename", CreateIfNotFound); 9

LoadData("filename", DataName, CreateIfNotFound); 9

LoadScript("filename", ScriptName, CreateIfNotFound); 9

SaveData("filename", filetype, objecttype, [fieldlist], [subdatalist], filter, [SortFieldList], Transpose, Append); 9

SaveDataEPC("filename", objecttype, filter, GEFileType, SaveBuses, Append); 10

SaveDataUsingExportFormat("filename", filetype, "FormatName", ModelToUse); 11

SaveDataWithExtra("filename", filetype, objecttype, [fieldlist], [subdatalist], filter, [SortFieldList], [Header_List], [Header_Value_List], Transpose, Append); 11

SaveObjectFields("filename", objecttype, [fieldlist]); 12

SelectAll(objecttype, filter); 12

SendtoExcel(objecttype, [fieldlist], filter, UseColumnHeaders, "workbookname", "worksheetname", [SortFieldList], [Header_List], [Header_Value_List], ClearExisting, RowShift, ColShift); 12

SetData(objecttype, [fieldlist], [valuelist], filter); 14

UnSelectAll(objecttype, filter); 14

Case Related Actions 15

AutoInsertTieLineTransactions; 15
CalculateRXBGFromLengthConfigCondType(filter); 15
CaseDescriptionClear; 15
CaseDescriptionSet("text", Append); 16
ChangeSystemMVABase(NewBase); 16
ClearSmallIslands; 16
ConditionVoltagePockets(VoltageThreshold, AngleThreshold, filter); 16
CreateNewAreasFromIslands; 16
DetermineBranchesThatCreateIslands(Filter, StoreBuses, "filename", SetSelectedOnLines, FileType); 17
DeterminePathDistance([start], BranchDistMeas, BranchFilter, BusField); 17
DetermineShortestPath([start], [end], BranchDistanceMeasure, BranchFilter, Filename); 18
DoFacilityAnalysis ("Filename"); 18
DirectionsAutoInsert(Source, Sink, DeleteExisting, UseAreaZoneFilters, Start, Increment); 18
EnterMode(mode); 19
EstimateVoltages(filter); 19
GenForceLDC_RCC(filter); 19
InitializeGenMvarLimits; 19
InjectionGroupsAutoInsert; 19
InjectionGroupCreate("Name", objecttype, InitialValue, filter, Append); 20
InjectionGroupRemoveDuplicates; 20
InterfaceCreate("Name", DeleteExisting, ObjectType, Filter); 20
InterfaceRemoveDuplicates; 21
LoadEMS("filename", filetype); 21
NewCase; 21
OpenCase("filename", OpenFileType,[LoadTransactions,StarBus,HowToKeepDuplicates]); 21
OpenCase("filename", OpenFileType,[MSLine,VarLimDead,PostCTGAGC,MSLineDummyBus]); 21
RenameInjectionGroup("OldName", "NewName"); 22
SaveCase("filename", SaveFileType, [PostCTGAGC, UseAreaZone]); 22
SaveCase("filename", SaveFileType, [AddCommentForObjectLabels]); 22
SaveGenLimitStatusAction("filename"); 22
SaveJacobian("JacFileName", "JIDFileName", FileType, JacForm); 23
SaveYbusInMatlabFormat("filename", IncludeVoltages); 23
Scale(scaletype, basedon, [parameters], scalemarker); 23
SetGenPMaxFromReactiveCapabilityCurve(filter); 24
SetParticipationFactors(Method, ConstantValue, Object); 25
SetScheduledVoltageForABus([bus identifier], voltage); 25
SetSelectedFromNetworkCut(SetHow, [BusOnCutSide], BranchFilter, InterfaceFilter, DCLineFilter, Energized, NumTiers, InitializeSelected, [ObjectsToSelect], UseAreaZone, UsekV, MinkV, MaxkV, LowerMinkV, LowerMaxkV); 25
UpdateIslandsAndBusStatus; 26
WriteLimitMonitoringSettings("filename"); 26
Distributed Computing Actions 27
EnterDistMasterPassword(Password); 27
VerifyDistributedComputersAvailable; 27
Oneline Actions 28
CloseOneline("OnelineName"); 28
EditMultipleOnelineAction("Path", LinkType, SaveFileType); 28
ExportBusView("filename", "bus key", ImageType, Width, Height, [ExportOptions]); 28
ExportOneline("filename", "OnelineName", ImageType, "view", FullScreen, ShowFull, [ExportOptions]); 29
ExportOnelineAsShapeFile("filename", "OnelineName", "ShapeFileExportDescriptionName", UseLonLat, PointLocation); 30
ImportDDLAsTranslation("filename"); 30
LoadAXD("filename", "OnelineName", CreateIfNotFound) 30
OpenOneline("filename", "view", FullScreen, ShowFull, LinkMethod, Left, Top, Width, Height); 31

RelinkAllOpenOnelines; 31
SaveOneline("filename", "OnelineName", SaveFileType); 31
OpenBusView("Bus key", ForceNewWindow); 31
OpenSubView("Substation key", ForceNewWindow); 32
User Interface Actions 33
MessageBox("text"); 33
ObjectFieldsInputDialog("ObjectIDString", [fieldlist], "DialogCaption", "DialogExplain", [LabelCaptions], [TabBreaks], [TabCaptions], [RowBreaks], [RowCaptions], [ColBreaks], [ColCaptions]); 33
OpenDataView("ObjectIDString", "DataGridIDString"); 35
Customer-Specific Actions 36
ISONEInterfaceLimitCalculation("filename", variablename); 36

Edit Mode Actions 38
Case Related Actions 38
AppendCase("filename", OpenFileType, [StarBus, EstimateVoltages]); 38
AppendCase("filename", OpenFileType, [MSLine, VarLimDead, PostCTGAGC, EstimateVoltages]); 38
Combine([elementA], [elementB]); 39
DeleteExternalSystem; 39
Equivalence; 39
InterfaceFlatten("InterfaceName"); 39
InterfaceModifyIsolatedElements(filter); 39
InterfacesAutoInsert(Type, DeleteExisting, UseFilters, "Prefix", Limits); 40
MergeBuses([element], Filter); 40
MergeLineTerminals(Filter); 40
MergeMSLineSections(Filter); 41
Move([elementA], [destination parameters], HowMuch); 41
ReassignIDs(objecttype, field, filter, UseRight); 41
Remove3WXformerContainer(filter); 42
Renumber3WXFormerStarBuses("filename", Delimiter); 42
RenumberAreas(NumCI); 42
RenumberBuses(NumCI); 43
RenumberMSLineDummyBuses("filename", Delimiter); 43
RenumberSubs(NumCI); 43
RenumberZones(NumCI); 43
SaveExternalSystem("Filename", SaveFileType, WithTies); 43
SplitBus([element], NewBusNumber, InsertBusTieLine, LineOpen, BranchDeviceType); 44
TapTransmissionLine([element], PosAlongLine, NewBusNumber, ShuntModel, TreatAsMSLine, UpdateOnelines); 44

Run Mode Actions 45
Animate(DoAnimate); 45
CalculatePTDF([transactor seller], [transactor buyer], LinearMethod); 45
CalculatePTDFMultipleDirections(StoreForBranches, StoreForInterfaces, LinearMethod); 46
CalculateLODF([BRANCH nearbusnum farbusnum ckt], LinearMethod, PostClosureLCDF); 46
CalculateLODFMatrix(WhichOnes, filterProcess, filterMonitor, MonitorOnlyClosed, LinearMethod, filterMonitorInterface, PostClosureLCDF); 46
CalculateLODFScreening(filterProcess, filterMonitor, IncludePhaseShifters, IncludeOpenLines, UseLODFThreshold, LODFThreshold, UseOverloadThreshold, OverloadLow, OverloadHigh, DoSaveFile, FileLocation, CustomFieldHighLODF, CustomFieldHighLODFLine, CustomFieldHighOverload, CustomFieldHighOverloadLine, DoUseCTGName, CustomFieldOrigCTGName); 47
CalculateLODFAdvanced(IncludePhaseShifters, FileType, MaxColumns, MinLODF, NumberFormat, DecimalPoints, OnlyIncludingLinesIncreasing, "FileName", IncludeIslandingCTG); 49
CalculateShiftFactors([flow element], direction, [transactor], LinearMethod, SetOutOfServiceBuses, filter, AbortOnError); 49
CalculateShiftFactorsMultipleElement(TypeElement,WhichElement,direction,[transactor],LinearMethod); 50

CalculateVoltSense([BUS num]); 51
CalculateFlowSense([flow element], FlowType); 51
CalculateLossSense(FunctionType,AreaSALossReference,IslandLossReference); 51
CalculateVoltToTransferSense([transactor seller], [transactor buyer], TransferType, TurnOffAVR); 52
CalculateVoltSelfSense(filter); 52
LineLoadingReplicatorCalculate([Flow Element], [Injection Group], AGCOnly, DesiredFlow, Implement, LinearMethod, UseLoadMinMax, MaxMultiplier, MinMultiplier); 52
LineLoadingReplicatorImplement; 53
DeleteState(WhichState, StateName); 53
RestoreState(WhichState, StateName); 54
SetInterfaceLimitToMonitoredElementLimitSum(filter); 55
SetSensitivitiesAtOutOfServiceToClosest(filter); 55
StoreState(StateName); 55
ZeroOutMismatches(ObjectType); 55
Power Flow Related Actions 56
ClearPowerFlowSolutionAidValues; 56
DiffCaseClearBase; 56
DiffCaseKeyType(KeyType); 56
DiffCaseMode(diffmode); 56
DiffCaseSetAsBase; 57
DiffCaseShowPresentAndBase(How); 57
DiffCaseRefresh; 57
DiffCaseWriteCompleteModel ("filename", AppendFile, SaveAdded, SaveRemoved, SaveBoth, KeyFields, "ExportFormat", UseAreaZone, UseDataMaintainer, AssumeBaseMeet, IncludeClearPowerFlowSolutionAidValues, DeleteBranchesThatFlipBusOrder); 57
DiffCaseWriteBothEPC ("filename", GEFileType, UseAreaZone, BaseAreaZoneMeetFilter, Append, "ExportFormat"); 58
DiffCaseWriteNewEPC ("filename", GEFileType, UseAreaZone, BaseAreaZoneMeetFilter, Append); 58
DiffCaseWriteRemovedEPC ("filename", GEFileType, UseAreaZone, BaseAreaZoneMeetFilter, Append); 58
DoCTGAction([contingency action]); 59
DoCTGAction("contingency action"); 59
ResetToFlatStart (FlatVoltagesAngles, ShuntsToMax, LTCsToMiddle, PSAnglesToMiddle); 59
SolvePowerFlow (SolMethod, "filename1", "filename2", CreateIfNotFound1, CreateIfNotFound2); 60
Contingency Related Actions 61
CTGApply("ContingencyName"); 61
CTGAutoInsert; 61
CTGCalculateOTDF([transactor seller], [transactor buyer], LinearMethod); 61
CTGClearAllResults; 61
CTGCloneMany(filter, "Prefix", "Suffix", SetSelected); 61
CTGCloneOne("ctgname", "newctgname", "Prefix", "Suffix", SetSelected); 62
CTGComboDeleteAllResults; 62
CTGComboSolveAll(DoDistributed, ClearAllResults); 62
CTGCompareTwoListsofContingencyResults (PRESENT or "ControllingFilename",PRESENT or "ComparisonFilename"); 63
CTGConvertAllToDeviceCTG(KeepOriginalIfEmpty); 63
CTGCreateContingentInterfaces(filter, maxOption); 64
CTGCreateExpandedBreakerCTGs; 64
CTGCreateStuckBreakerCTGs(filter, AllowDuplicates, "PrefixName", IncludeCTGLabel, BranchFieldName, "SuffixName", "PrefixComment", BranchFieldComment, "SuffixComment"); 64
CTGDeleteWithIdenticalActions; 65
CTGJoinActiveCTGs(InsertSolvePowerFlow, DeleteExisting, JoinWithSelf, "filename"); 65
CTGPrimaryAutoInsert; 65
CTGProcessRemedialActionsAndDependencies(DoDelete, filter); 65
CTGProduceReport("filename"); 65

CTGReadFilePSLF("filename"); 66
CTGReadFilePTI("filename"); 66
CTGRelinkUnlinkedElements; 66
CTGRestoreReference; 66
CTGSaveViolationMatrices("filename", filetype, UsePercentage, [ObjectTypesToReport], SaveContingency, SaveObjects, FieldListObjectType, [FieldList], IncludeUnsolvableCTGs); 66
CTGSetAsReference; 67
CTGSolve("ContingencyName"); 67
CTGSolveAll(DoDistributed, ClearAllResults); 67
CTGSort([SortFieldList]); 67
CTGWriteAllOptions("filename", KeyField, UseSelectedDataMaintainer, SaveDependencies, UseAreaZoneFilters); 68
CTGWriteFilePTI("filename", BusFormat, TruncateCTGLabels, "filtername", Append); 69
CTGWriteResultsAndOptions("filename", [opt1, opt2, opt3, ..., opt19], KeyField, UseDATASection, UseConcise, UseObjectIDs, UseSelectedDataMaintainers, SaveDependencies, UseAreaZoneFilters); 69
Fault Related Actions 71
Fault([Bus num], faulttype, R, X); 71
Fault([BRANCH nearbusnum farbusnum ckt], faultlocation, faulttype, R, X); 71
FaultClear; 71
FaultMultiple(UseDummyBus); 71
LoadPTISEQData("filename", version); 71
ATC (Available Transfer Capability) Related Actions 72
ATCCreateContingentInterfaces(filter); 72
ATCDeleteAllResults; 72
ATCDeleteScenarioChangeIndexRange(ScenarioChangeType, [IndexRange]); 72
ATCDetermine([transactor seller], [transactor buyer], DoDistributed, DoMultipleScenarios); 72
ATCDetermineMultipleDirections(DoDistributed, DoMultipleScenarios); 73
ATCDetermineATCFor(RL, G, I, ApplyTransfer); 73
ATCDetermineMultipleDirectionsATCFor(RL, G, I); 73
ATCIncreaseTransferBy(amount); 73
ATCRestoreInitialState; 73
ATCSetAsReference; 73
ATCTakeMeToScenario(RL, G, I); 73
ATCWriteAllOptions("filename", AppendFile, KeyField); 73
ATCDataWriteOptionsAndResults("filename", AppendFile, KeyField); 73
ATCWriteResultsAndOptions("filename", AppendFile); 74
ATCWriteScenarioLog("filename", AppendFile, filter); 74
ATCWriteScenarioMinMax("filename", filetype, AppendFile, [fieldlist], Operation, OperationField, GroupScenario, [RLFilter], [GFilter], [IFilter], DirectionFilter, DoGroupDirection, filter, PreferIterativelyFound); 75
ATCWriteToExcel("worksheetname", [fieldlist]); 77
ATCWriteToText("filename", filetype, [fieldlist]); 77
GIC (Geomagnetically Induced Current) Related Actions 78
GICCalculate(MaxField, Direction, SolvePF); 78
GICClear; 78
GICLoad3DEfield(FileType, "FileName", SetupOnLoad); 78
GICTimeVaryingCalculate(TheTime,SolvePF); 78
GICTimeVaryingAddTime(NewTime); 78
GICTimeVaryingDeleteAllTimes; 78
GICTimeVaryingEFieldCalculate(TheTime,SolvePF); 78
GICTimeVaryingElectricFieldsDeleteAllTimes; 78
GICWriteOptions("FileName", KeyField); 78
ITP (Integrated Topology Processing) Related Actions 79
CloseWithBreakers(objecttype, filter or [object identifier], OnlyEnergizeSpecifiedObjects, [SwitchingDeviceTypes], CloseNormallyClosedDisconnects); 79

ExpandAllBusTopology; 80
ExpandBusTopology(BusIdentifier, TopologyType); 80
OpenWithBreakers(objecttype, filter or [object identifier], [SwitchingDeviceTypes], OpenNormallyOpenDisconnects); 80
SaveConsolidatedCase("filename", filetype, [BusFormat, TruncateCtgLabels, AddCommentsForObjectLabels]); 82
OPF (Optimal Power Flow) and SCOPF Related Actions 83
SolvePrimalLP("filename1", "filename2", CreateIfNotFound1, CreateIfNotFound2); 83
InitializeLP("filename1", "filename2", CreateIfNotFound1, CreateIfNotFound2); 83
SolveSinglePrimalLPOuterLoop("filename1", "filename2", CreateIfNotFound1, CreateIfNotFound2); 83
SolveFullSCOPF (BCMethod, "filename1", "filename2", CreateIfNotFound1, CreateIfNotFound2); 84
OPFWriteResultsAndOptions("filename"); 84
PV Related Actions 85
PVClear; 85
PVDataWriteOptionsAndResults("filename", AppendFile, KeyField); 85
PVDestroy; 85
PVQVTrackSingleBusPerSuperBus; 85
PVRun([elementSource], [elementSink]); 86
PVSetSourceAndSink([elementSource], [elementSink]); 86
PVStartOver; 86
PVWriteInadequateVoltages("filename", AppendFile, InadequateType); 86
PVWriteResultsAndOptions("filename", AppendFile); 87
RefineModel(objecttype, filter, Action, Tolerance); 87
QV Related Actions 88
QVDataWriteOptionsAndResults("filename", AppendFile, KeyField); 88
QVRun("filename", InErrorMakeBaseSolvable); 88
QVSelectSingleBusPerSuperBus; 88
QVWriteCurves("filename", IncludeQuantitiesToTrack, filter, Append); 89
QVWriteResultsAndOptions("filename", AppendFile); 89
TS (Transient Stability) Related Actions 90
TSAutoCorrect; 90
TSAutoInsertDistRelay(Reach, AddRelayAtFromBus, AddRelayAtToBus, DoTransferTrip, Shape, filter); 90
TSAutoInsertZPOTT(Reach, filter,); 91
TSCalculateCriticalClearTime([branch] or filter,); 91
TSCalculateSMIBEigenValues; 91
TSClearAllModels; 91
TSClearResultsFromRAM(ALL/SELECTED/"ContingencyName", ClearSummary, ClearEvents, ClearStatistics, ClearTimeValues, ClearSolutionDetails); 91
TSDisableMachineModelNonZeroDerivative(DerivativeThreshold); 91
TSGetVCurveData("FileName", filter); 92
TSGetResults("FileName", SINGLE/SEPARATE/JSIS, [Contingencies], [Plots, ObjectFields], StartTime, EndTime]); 92
TSInitialize(CheckInitialized); 92
TSLoadBPA("FileName"); 93
TSLoadGE("FileName", GENCCYN, EnableOutOfOrderModels); 93
TSLoadPTI("FileName", "MCREfilename", "MTRLOfilename", "GNETfilename", "BASEGENfilename", "MODREMOVEfilename); 93
TSLoadRDB("filename", ModelType, filter); 93
TSLoadRelayCSV("filename", ModelType, filter); 94
TSResultStorageSetAll(objecttype, YES/NO); 94
TSRunUntilSpecifiedTime("ContingencyName", [StopTime, StepSize, StepsInCycles, ResetStartTime, NumberOfTimeStepsToDo]); 94
TSSaveBPA("FileName", DiffCaseModifiedOnly); 94
TSSaveGE("FileName", DiffCaseModifiedOnly); 95
TSSavePTI("FileName", DiffCaseModifiedOnly); 95

TSSaveTwoBusEquivalent ("AuxFileName", [BUS]); 95
TSSolve("ContingencyName", [StartTime, StopTime, StepSize, StepInCycles]); 95
TSSolveAll(DoDistributed); 95
TSTransferStateToPowerFlow; 95
TSValidate; 95
TSWriteModels("FileName", DiffCaseModifiedOnly); 96
TSWriteOptions("FileName",[SaveDynamicModel, SaveStabilityOptions, SaveStabilityEvents, SaveResultsEvents, SavePlotDefinitions, SaveTransientLimitMonitors, SaveResultAnalyzerTimeWindow], KeyField); 96
TSSetSelectedForTransientReferences(SetWhat, SetHow, [ObjectType List],[ModelType List]); 96
TSSaveDynamicModels("FileName", FileType, ObjectType, Filter, Append); 96
Scheduled Actions Related Actions 97
IdentifyBreakersForScheduledActions(IdentifyFromNormalStatus); 97
SetScheduleView(ViewTime, ApplyActions, UseNormalStatus, ApplyWindow); 97
SetScheduleWindow(StartTime, EndTime, Resolution, ResolutionUnits); 97
Trainer Related Actions 98
ResetStatusChangeCount; 98
SuppressCommandDialog; 98

***DATA Section* 99**

Concise Auxiliary File Header 99

Legacy Auxiliary File Header 99

ObjectType 99

File_Type_Specifier 100

Required Fields and Create_if_not_found 100

List_of_Fields 100

Concise Field Variable Names 101

Legacy Field Variable Naming 101

Special Naming 101

Key Fields 101

Data List 102

Special Data List Entries 102

Special Identifiers for Model Fields in Data 102

Using Labels for Identification 103
Saving Auxiliary Files Using Labels 104
Loading Auxiliary Files SUBDATA Sections Using Labels 104
Special Use of Labels in SUBDATA 105
ATC Scenarios 105
ATC Extra Monitors 105
Model Conditions 105
Model Expressions 105
Bus Load Throw Over Records 105
Injection Group Participation Points 105

SubData Sections 106
ATC_Options 107
RLScenarioName 107
GScenarioName 107

IScenarioName 107
ATCMemo 107
ATCExtraMonitor 107
ATCFlowValue 107
ATCScenario 108
TransferLimiter 108
ATCExtraMonitor 108
AUXFileExportFormatData 109
DataBlockDescription 109
AUXFileExportFormatDisplay 109
DataBlockDescription 109
BGCalculatedField 109
Condition 109
Bus 110
MWMarginalCostValues 110
MvarMarginalCostValues 110
LPOPFMarginalControls 110
BusViewFormOptions 110
BusViewBusField 110
BusViewFarBusField 110
BusViewGenField 110
BusViewLineField 110
BusViewLoadField 110
BusViewShuntField 110
ColorMap 111
ColorPoint 111
Contingency 111
CTGElementAppend 111
CTGElement 111
LimitViol 123
Sim_Solution_Options 124
WhatOccurredDuringContingency 124
ContingencyMonitoringException 124
CTG_Options 124
Sim_Solution_Options 124
CTGElementBlock 125
CTGElement 125
CTGElementAppend 125
CustomColors 125
CustomColors 125
CustomCaseInfo 125
ColumnInfo 125
DataGrid 125
ColumnInfo 125
ColumnContourInfo 126
DynamicFormatting 127
DynamicFormattingContextObject 127
LineThicknessLookupMap 128
LineColorLookupMap 128
FillColorLookupMap 128
FontColorLookupMap 128
FontSizeLookupMap 128
BlinkColorLookupMap 128

XoutColorLookupMap 128
FlowColorLookupMap 128
SecondaryFlowColorLookupMap 128
Filter 129
Condition 129
Gen 130
BidCurve 130
ReactiveCapability 130
GeoDataViewStyle 131
TotalAreaValueMap 131
RotationRateValueMap 131
RotationAngleValueMap 131
LineThicknessValueMap 132
GlobalContingencyActions 132
CTGElementAppend 132
CTGElement 132
HintDefValues 132
HintObject 132
InjectionGroup 133
PartPoint 133
Interface 134
InterfaceElement 134
KMLExportFormat 135
DataBlockDescription 135
LimitSet 135
LimitCost 135
Load 135
BidCurve 135
LPVariable 136
LPVariableCostSegment 136
ModelCondition 136
Condition 136
ModelExpression 136
LookupTable 136
ModelFilter 137
ModelCondition 137
MTDCRecord 138
MTDCBus 138
MTDCConverter 138
MTDCTransmissionLine 139
MultiSectionLine 140
Bus 140
BusRenumber 141
Nomogram 141
InterfaceElementA 141
InterfaceElementB 141
NomogramBreakPoint 142
NomogramInterface 142
InterfaceElement 142
Owner 142
Bus 142
Load 142
Gen 142

Branch 143
PostPowerFlowActions 143
CTGElementAppend 143
CTGElement 143
PWCaseInformation 143
PWCaseHeader 143
PWFormOptions 143
PieSizeColorOptions 143
PWLPOPFCTGViol 144
OPFControlSense 144
OPFBusSenseP 144
OPFBusSenseQ 144
PWLPTabRow 144
LPBasisMatrix 144
PWPVResultListContainer 145
PWPVResultObject 145
LimitViol 145
PVBusInadequateVoltages 145
PWQVResultListContainer 146
PWPVResultObject 146
QVCurve 146
QVPoints 146
QVCurve_Options 147
Sim_Solution_Options 147
RemedialAction 147
CTGElementAppend 147
CTGElement 147
SelectByCriteriaSet 147
SelectByCriteriaSetType 147
Area 148
Zone 148
ScreenLayer 148
ShapefileExportDescription 148
StudyMWTransactions 148
ImportExportBidCurve 148
SuperArea 149
SuperAreaArea 149
TSSchedule 149
SchedPoint 149
UserDefinedDataGrid 150
ColumnInfo 150

SCRIPT Section for Display Auxiliary File* *151

AXD Actions 151
AutoInsertBorders; 151
AutoInsertBuses(LocationSource, MapProjection, AutoInsertBranches, InsertIfNotAlreadyShown, "filename", InsertSelected); 151
AutoInsertInterfaces(InsertPieCharts, PieChartSize); 151
AutoInsertLineFlowObjects(MinkV, InsertOnlyIfNotAlreadyShown, LineLocation, Size, FieldDigits, FieldDecimals, TextPosition, ShowMW, ShowMvar, ShowMVA, ShowUnits, ShowComplex); 152
AutoInsertLineFlowPieCharts(MinkV, InsertOnlyIfNotAlreadyShown, InsertMSLines, Size); 152
AutoInsertLines(MinkV, InsertTextFields, InsertEquivObjects, InsertZBRPieCharts, InsertMSLines, ZBRImpedance, NoStubsZBRs, SingleCBZRs); 152

AutoInsertLoads(MinkV, InsertTextFields); 152
AutoInsertSwitchedShunts(MinkV, InsertTextFields); 152
AutoInsertSubStations(LocationSource, MapProjection, AutoInsertBranches, InsertIfNotAlreadyShown, "filename", InsertSelected); 153
AutoInsertAreas(MapProjection, InsertIfNotAlreadyShown, InsertSelected); 153
AutoInsertInjectionGroups(MapProjection, InsertIfNotAlreadyShown, InsertSelected); 153
AutoInsertOwners(MapProjection, InsertIfNotAlreadyShown, InsertSelected); 153
AutoInsertZones(MapProjection, InsertIfNotAlreadyShown, InsertSelected); 154
FixFlowArrowLineEnds("OnelineName", "LayerName"); 154
FixFlowArrowPosition("OnelineName", "LayerName"); 154
InsertConnectedBuses("BusIdentifier"); 154
LoadAXDFromAXD("filename", CreateIfNotFound) 155
PanAndZoomToObject("ObjectID", "DisplayObjectType", "DoZoom"); 155
ResetStubLocations(ZBRImpedance, NoStubsZBRs); 155
General Script Commands 156

DATA Section for Display Auxiliary File ***157***

Key Fields **157**

Special Data Sections **157**
GeographyDisplayOptions 157
Picture 157
PWFormOptions 157
View 158

SubData Sections **158**
ColorMap 158
CustomColors 158
DisplayDCTramisssionLine 159
DisplayInterface 159
DisplayMultiSectionLine 159
DisplaySeriesCapacitor 159
DisplayTransformer 159
DisplayTransmissionLine 159
Line 159
Line 159
DynamicFormatting 159
Filter 159
GeoDataViewStyle 159
PieChartGaugeStyle 160
ColorMap 160
PWFormOptions 160
SelectByCriteriaSet 160
UserDefinedDataGrid 160
View 160
ScreenLayer 160

Introduction

PowerWorld has incorporated the ability to import data to/from data sources other than power flow models into PowerWorld Simulator. The text file interface for exchanging data, as well as for executing a batch script command, is represented by the auxiliary files. The script language and auxiliary data formats are incorporated together. This format is described in this document.

Script/Data files are called data auxiliary files in Simulator and typically have the file extension .AUX. These files mostly contain information about power system elements and options for running the various tools within Simulator. They do not contain any information about the individual display objects contained on a one-line diagram. There are separate files called display auxiliary files that are available for importing display data to/from Simulator in a text format. These files are distinguished from the data auxiliary files by using the extension .AXD. The format for these two types of files is similar, but different object types are supported by each and require that the files be read separately.

Both file types will be generically referred to as auxiliary files. An auxiliary file may be comprised of one or more data or script sections. A data section provides specific data for a specific type of object. A script section provides a list of script actions for Simulator to perform. These sections have the following format:

```
SCRIPT ScriptName1
{
script_statement_1;
   .
script_statement_n;
}

object_type DataName1(list_of_fields)
{
data_list_1
   .
data_list_n
}

object_type DataName2(list_of_fields)
{
data_list_1
   .
data_list_n
}

SCRIPT ScriptName2
{
script_statement_1;
   .
script_statement_n;
}
```

Data sections start with the string representing the object_type, but older auxiliary files start with the syntax
DATA DataName(object_type, [list_of_fields])
See the description of the data section later in this document for more information.

Note that the keywords script, data, or a valid object_type must occur at the start of a text file line. Auxiliary files may contain more than one section. These sections always begin with the keyword script, data, or a valid object_type. Data sections are followed by an argument list enclosed in ( ). The actual data or script commands are then contained within curly braces { }. Strings are enclosed in straight quotes – note that smart quotes will not work (this might be encountered when copy/pasting script commands from another program). The Script commands available are described in the next main section. The data sections are then described after this. There are separate sections for describing the data sections for the data auxiliary files and the display auxiliary file.

SCRIPT Section

```
SCRIPT ScriptName
{
script_statement_1;
   .
script_statement_n;
}
```

Scripts may optionally contain a ScriptName. This enables you to call a particular SCRIPT by using the LoadScript action (see General Actions). After the optional name, the SCRIPT section begins with a left curly brace and ends with a right curly brace. Inside of this, script statements can be given. In general, a script statement has the following format

```
Keyword(arg1, arg2, ...);
```

- Statement starts with a keyword.
- The keyword is followed by an argument list which is encompassed in parentheses ( ).
- The arguments are separated by commas.
- If a single argument is a list of things, this list is encompassed by braces [ ].
- Statements end with a semicolon.
- Statements may take up several lines of the text file.
- You may put more than one statement on a single text line.

Those familiar with using Simulator will know that there is a RUN and EDIT mode in Simulator. Some features in Simulator are only available in one mode or the other. This functionality will be preserved in the script language. In earlier versions of the software, certain functionality was organized by the "submode" feature. While existing scripts designed to work with submodes will still function as before, moving between submodes is no longer necessary.

Various script commands require that you be in RUN or EDIT mode. If a script requires this, then the script will automatically change modes.

Using Filters in Script Commands

Many script commands allow the specification of a filtername. Only those objects meeting this filter will be selected for the specified action. Unless otherwise specified, a blank filter will select all objects. This filtername can be the name of an advanced filter. The filtername should be enclosed in double quotes. Advanced filters belonging to a different objecttype can also be used depending on the objectype in use. For example, if filtering generator objects a bus filter can also be used. When using an advanced filter that belongs to a different objecttype, the format of the filter is `"<Objecttype>filtername"` instead of just specifying the filtername itself.

The filtername can also be the name of a device filter. A device filter allows you to specify a particular object for filtering instead of a class of object. For example, you might want to return all buses that belong to a particular substation. You can specify the device filter for the particular substation and then apply this to the bus objects. The format of a device filter is `"<DEVICE>objecttype 'key1' 'key2' 'key3'"`.

In addition to a filtername, special keywords can be used to indicate the type of filter desired. These include the following:

AREAZONE : Only objects that meet the area/zone/owner filters will be selected for the specified action.
SELECTED : Only objects whose Selected field is YES will be selected for the specified action.

Specifying Special Keywords in Script Command Parameters

The special keywords @DATETIME, @DATE, @TIME, @BUILDDATE, @VERSION, @CASENAME, @CASEFILENAME, and @CASEFILEPATH may be used as part of the input for script command parameters. Generally, these are allowed as part of the file name input in commands that save or modify files, as well as, some other commands that take text as input. These special keywords will be replaced with their actual values when the script command is processed. @DATETIME will replace the keyword with the actual date and time in the format yyyymmdd_hhnnss-hhmm with the UTC offset included on the end of the time.

The special keyword @MODELFIELD can be used in combination with an object type and variable name so that any field of any object can be included in the text. The syntax for inserting the value of a model field in the text is the following: `@MODELFIELD<objecttype 'key1' 'key2' variablename:digits:rod>`.

Specifying File Names in Script Commands

In place of the "filename" parameter in any script command, specially formatted text can be used to indicate that the user should be prompted to choose the file. Depending on whether or not a file is being opened or saved, an Open or Save dialog will be presented for the user to choose the file. This will not work when using the SimAuto Add-on.

The special syntax of the filename parameter is generally:
`"<PROMPT 'Caption' 'FileTypes' 'InitialDirectory' 'CancelAction'>"`
The entire string must start with <PROMPT and end with >. The parameters after PROMPT are all optional, must be space delimited, must be enclosed in single quotes, and can be specified as follows:

Caption
: Caption to be placed at the top of the file dialog that appears. If omitted, either 'Save' or 'Open' is assumed based on how the prompt is used to access a file.

FileTypes
: List of file types and extensions. The list itself is composed of a pipe-delimited string (|) with the first string representing the first file type, the second string representing the first file extension, the third string representing the second file type, the fourth string representing the second file extension and so on. If no file types are specified, 'All Files (*.*)|*.*' is assumed.

InitialDirectory
: The initial directory in which the dialog should open.

CancelAction
: The CancelAction can either be 'Abort' or 'Continue', with 'Abort' being the default. If the Cancel button is clicked on the resulting dialog, the CancelAction specifies how to proceed. Setting this to 'Abort' will skip the command and abort any remaining auxiliary file commands. Setting this to 'Continue' will skip the command but continue processing the remaining auxiliary file commands.

Here is an example prompt using all options:
"<PROMPT 'Choose an AUX file' 'Auxiliary Files (*.aux)|*.aux|All Files (*.*)|*.*' 'c:\StudyDirectory\AUX Files' 'Abort'>"

The special keywords described in the Specifying Special Keywords in Script Command Parameters section can also be used as part of a filename.

Specifying Field Variable Names in Script Commands

See the Field Variable Naming (Legacy) topic in the DATA Section for general information about naming fields.

Within select script commands the keyword ALL can be used instead of using the location number of a field when specifying variable names as part of a field list. This will return all fields with the same variable name. This is intended to allow easier access to fields when the exact number of fields is not known, such as with multiple TLR (MultBusTLRSens:ALL) or PTDF (LinePTDFMult:ALL) results. This can be used with SaveData, SaveDataWithExtra, SaveObjectFields, and SendToExcel script actions.

Within select script commands the keyword ALL can be used instead of a list of fields. This will return all fields for a particular objecttype. This can be used with SaveData, SaveDataWithExtra, SaveObjectFields, and SendToExcel script actions.

Within these same Save related script commands for any transient stability data model objecttype (Machines, Exciters, etc…), the special strings may be used to export all the model input parameters for the stability model and the various primary, secondary or label key fields. These special strings are as follows.

AllModelParams	exports all the model input parameters and references to objects necessary to define model (Added starting in the August 16, 2022 patch of Simulator 22)
AllModelParamsPriKey	exports the primary keys for the objecttype and the same fields as AllModelParams (Added starting in the August 16, 2022 patch of Simulator 22)
AllModelParamsSecKey	exports the secondary keys for the objecttype and the same fields as AllModelParams (Added starting in the August 16, 2022 patch of Simulator 22)
AllModelParamsLabel	exports the label key for the objecttype and the same fields as AllModelParams (Added starting in the August 16, 2022 patch of Simulator 22)

Specifying Field Values in Script Commands

Several script commands require that a valuelist be specified to assign values to a corresponding fieldlist. Instead of specifying the values explicitly, special formatting is available to assign values from other fields. See the Special Data List Entries topic in the DATA Section for more information.

General Actions

Program Actions

Available regardless of the mode

```
CopyFile                    ("oldfilename", "newfilename");
DeleteFile                  ("filename");
ExitProgram;
LogAdd                      ("text");
LogAddDateTime              ("label", includedate, includetime, includemilliseconds);
LogClear;
LogSave                     ("filename", AppendFile);
LogShow                     (DoShow);
RenameFile                  ("oldfilename", "newfilename");
SetCurrentDirectory         ("filedirectory", CreateIfNotFound);
StopAuxFile;
WriteTextToFile             ("filename", "text");
```

CopyFile("oldfilename", "newfilename");

Use this action to copy a file from within a script.

"oldfilename"	:	The present file name. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.
"newfilename"	:	The new file name desired. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.

DeleteFile("filename");

Use this action to delete a file from within a script.

"filename"	:	The file name to delete. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.

ExitProgram;

Immediately exits the program with no prompts. Note that this command should not be used in scripts loaded through SimAuto, as the SimAuto instance should be managed from the script which created it.

LogAdd("text");

Use this action to add a personal message to the MessageLog.

"text"	:	The text that will appear as a message in the log.

LogAddDateTime("label", includedate, includetime, includemilliseconds);

Use this action to add the date and time to the message log

"label"	:	A string which will appear at the start of the line containing the date/time.
includedate	:	YES – Include the data or NO to not include.
includetime	:	YES – Include the time or NO to not include.
includemilliseconds	:	YES – Include the milliseconds or NO to not include.

LogClear;

Use this action to clear the Message Log.

LogSave("filename", AppendFile);

This action saves the contents of the Message Log to "filename".

"filename"	:	The file name to save the information to.
AppendFile	:	Set to YES or NO. YES means that the contents of the log will be appended to "filename". NO means that "filename" will be overwritten.

LogShow(DoShow);

Added in February 21, 2020 patch of Simulator 21

This action will show or hide the Message Log.

DoShow	:	Set to YES to show the Message Log. Set to NO to hide the Message Log.

RenameFile("oldfilename", "newfilename");

Use this action to rename a file from within a script.

"oldfilename"	:	The present file name. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.
"newfilename"	:	The new file name desired. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.

SetCurrentDirectory("filedirectory", CreateIfNotFound);

Use this action to set the current work directory.

"filedirectory"	:	The path of the working directory. See the Specifying Special Keywords in Script Command Parameters and Special Identifiers for Model Fields section for information on special keywords that can be used as part of the directory name.
CreateIfNotFound	:	Set to YES or NO. YES means that if the directory path cannot be found,the directory will be created. If this parameter is not specified, NO is assumed.

StopAuxFile;

Use this action to treat the remainder of the file after the command as a big comment. This includes any script commands inside the present SCRIPT block, as well as all remaining SCRIPT or DATA blocks.

WriteTextToFile("filename", "text");

Use this action to write text to a file. If the specified file already exists, the text will be appended to the file. Otherwise, it creates the file and writes the text to the file.

"filename"	:	The file path and name to save.
"text"	:	The text to be written to the file. Special keywords can be entered that will be replaced with their actual values. These include @BUILDDATE, @DATETIME, @DATE, @TIME, @VERSION, and @CASENAME.

Data Actions

Available regardless of the mode

```
CreateData                    (objecttype, [fieldlist], [valuelist]);
Delete                        (objecttype, filter);
DeleteDevice                  ("ObjectIDString");
DeleteIncludingContents       (objecttype, filter);
ExportAreaSupplyCurves        ("filename", "User Defined String", NumPoints);
ImportData                    ("filename", FileType, CreateIfNotFound);
LoadAux                       ("filename", CreateIfNotFound);
LoadAuxDirectory              ("filedirectory", "filterstring", CreateIfNotFound);
LoadCSV                       ("filename", CreateIfNotFound);
LoadData                      ("filename", DataName, CreateIfNotFound);
LoadScript                    ("filename", ScriptName, CreateIfNotFound);
SaveData                      ("filename", filetype, objecttype, [fieldlist], [subdatalist],
                              filter, [SortFieldList], Transpose, Append);
SaveDataEPC                   ("filename",objecttype,filter,GEFileType,SaveBuses,Append);
SaveDataUsingExportFormat     ("filename",filetype,"FormatName",ModelToUse);
SaveDataWithExtra             ("filename", filetype, objecttype, [fieldlist], [subdatalist],
                              filter, [SortFieldList], [Header_List], [Header_Value_List],
                              Transpose, Append);
SaveObjectFields              ("filename", objecttype, [fieldlist]);
SelectAll                     (objecttype, filter);
SendToExcel                   (objecttype, [fieldlist], filter, UseColumnHeaders,
                              "workbookname", "worksheetname", [SortFieldList], [Header_List],
                              [Header_Value_List], ClearExisting, ShiftRow, ShiftCol);
SetData                       (objecttype, [fieldlist], [valuelist], filter);
UnSelectAll                   (objecttype, filter);
```

CreateData(objecttype, [fieldlist], [valuelist]);

Use this action to create particular objects.

objecttype	:	The objecttype being created.
[fieldlist]	:	A list of fields to set with the object. The key fields and required fields must be specified.
[valuelist]	:	A list of values corresponding to the respective fields.

Delete(objecttype, filter);

Use this delete objects of a particular type. A filter may optionally be specified to only delete objects that meet a filter.

objecttype	:	The objecttype being selected.
filter	:	Optional parameter – default is to delete all objects of specified type See Using Filters in Script Commands section for more information on specifying the filter.

DeleteDevice([ObjectIDString]);

Use this action to delete a specific object.

[ObjectIDString]	:	The specific object to delete. The format is the object type followed by the key fields used to identify the object. Examples: DeleteDevice([Bus 234891]), DeleteDevice([Branch 1239 1234 "AB"]), and DeleteDevice([Interface "my interface name"]).

DeleteIncludingContents(objecttype, filter);

Use this to delete objects of a particular type and other objects that these contain. Currently, only multi-section lines (objecttype = MultiSectionLine) can be used with this command. The branches and dummy buses that belong to multi-section lines will also be deleted along with the multi-section lines. A filter may optionally be specified to only delete objects that meet a filter. The syntax is identical to the Delete(objecttype, filter); action above.

ExportAreaSupplyCurves("filename", "User Defined String", NumPoints);

Use this action to export Area Supply Curves to a CSV file. The output of the file will have 7 entries for each area for Fixed Gen MW, Fixed Load MW, Fixed Shunt MW, Losses MW, Variable Min MW, Variable Max MW, Variable Present MW, followed by a set of Bid MW/Price entries represents the supply curve for the variable MWs.

"filename.csv"	:	The name of the CSV file to which results will be written.
"User Defined String"	:	This is an optional parameter for specifying a user defined string written to each entry in the resulting CSV file. If this is omitted, blank will be assumed.
NumPoints	:	This is an optional parameter and is related to converting a cubic cost model into a piece-wise linear model. If this is omitted, 5 is the default.

ImportData("filename", FileType, HeaderLine, CreateIfNotFound);

Use this action to import data in various file formats that are not native to Simulator.

"filename"	:	Name of the file to import
FileType	:	Parameter that specifies the format of the data this is being read. Currently supported are two methods of importing CROW files as created by the Equinox Control Room Operations Window application. This is used with the Scheduled Actions add-on tool. CSV : Uses CSV Import Settings as specified in the Scheduled Actions dialog to read in a CROW CSV file. CROW : Uses the hardcoded format Scheduled Actions was originally programmed to import.
HeaderLine	:	Optional parameter to specify if the row of headers in the CSV file is on the first line (1) or second line (2). If left blank (or any other value is specified), it will use the setting last configured in the Scheduled Actions dialog.
CreateIfNotFound	:	Optional parameter that is NO by default. Set this to YES to create objects defined in the data if they do not already exist.

LoadAux("filename", CreateIfNotFound);

Use this action to load another auxiliary file from within a script.

"filename"	:	The filename of the auxiliary file being loaded.
CreateIfNotFound	:	Set to YES or NO. YES means that objects which cannot be found will be created while reading in DATA sections from filename. If this parameter is not specified, NO is assumed.

LoadAuxDirectory("filedirectory", "filterstring", CreateIfNotFound);

Use this action to load multiple auxiliary files from a specified directory. The auxiliary files will be loaded in alphabetical order by name.

"filedirectory"	:	The directory where the auxiliary files are located.
"filterstring"	:	Optional – if not specified then all files in the directory are loaded. If specified, only files meeting this filter will be loaded. This filtering supports normal Windows wildcard filtering.
CreateIfNotFound	:	Optional – default is NO Set to YES or NO. YES means that objects that cannot be found will be created while reading in DATA sections from the files.

LoadCSV("filename", CreateIfNotFound);

Use this action to load a CSV file that is formatted the same as the data sent to Excel in the Send All to Excel option found within a case information display, or by choose Save As CSV.

"filename"	:	The filename of the CSV file being loaded.
CreateIfNotFound	:	Set to YES or NO. YES means that objects which cannot be found will be created. If this parameter is not specified, NO is assumed.

LoadData("filename", DataName, CreateIfNotFound);

Use this action to load a named Script Section from another auxiliary file. This will open the auxiliary file denoted by "filename", but will only execute the script section specified.

"filename"	:	The filename of the auxiliary file being loaded.
DataName	:	The specific ScriptName from the auxiliary file which should be loaded.
CreateIfNotFound	:	Set to YES or NO. YES means that objects which cannot be found will be created while reading in DATA sections from filename. If this parameter is not specified, NO is assumed.

LoadScript("filename", ScriptName, CreateIfNotFound);

Use this action to load a named Script Section from another auxiliary file. This will open the auxiliary file denoted by "filename", but will only execute the script section specified.

"filename"	:	The filename of the auxiliary file being loaded.
ScriptName	:	The specific ScriptName from the auxiliary file which should be loaded.
CreateIfNotFound	:	Set to YES or NO. YES means that objects which cannot be found will be created while reading in SCRIPT sections from filename. If this parameter is not specified, NO is assumed.

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

Use this action to save data in a custom defined format.

"filename"	:	The file path and name to save.
filetype	:	There are several options for the filetype
		AUXCSV : save as a comma-delimited auxiliary data file.
		AUX : save as a space-delimited auxiliary data file.
		CSV : save as a normal CSV file without the AUX file syntax. The first few lines of the text file will represent the object name and field variable names.
		CSVNOHEADER : save as a normal CSV text file, without the AUX file formatting. The object name and field variable names are NOT included. This option is useful when appending data of the same object type and field list into a common file.
		CSVCOLHEADER : save as a normal CSV without the AUX syntax and with the first row showing column headers you would see in a case information display
objecttype	:	The type of object being saved.
[fieldlist]	:	A list of fields that you want to save. For numeric fields, the number of digits and the number of decimal places (digits to right of decimal) can be specified by using the following format for the field, `variablenamelegacy:location:digits:rod` or `concisename:digits:rod`. See the Specifying Field Variable Names in Script Commands topic for more information on specifying this list.
[subdatalist]	:	A list of the subdata objecttypes to save with each object record.

filter : Optional parameter – default is to save all objects of specified type

- blank : Save all objects of the specified type
- AREAZONE : Only objects that meet the area/zone/owner filters will be saved
- SELECTED : Only objects whose Selected field = YES will be saved
- "FilterName" : Only objects that meet the specified filter will be checked. See Using Filters in Script Commands section for more information on specifying the filtername.

[SortFieldList] : Optional parameter – the default is to do no sorting
This allows the specification of a sort order in which the data will be saved. The format is: [variablename1:+:0, variablename2:-:1] where

- variablename : the first parameter is the name of the field by which to sort. There is no limit to how many fields can be specified for sorting. For fields that require a location other than zero, variablename can be in the format fieldname:location.
- + or - : the second parameter indicates sort ascending for + and sort descending for -. This parameter must be specified.
- 0 or 1 : for the third parameter 0 means case insensitive and do not use absolute value, 1 mean case sensitive or use absolute value. This parameter is optional.

Transpose : Optional parameter – default is NO
Set this to YES or NO. Set to YES to transpose the columns and rows of the returned data. When transposed the values for the same field for all selected objects will appear in the same row. Transposing the data is only allowed for CSV filetypes and this option will default to NO for all other filetypes.

(Following added in the February 13, 2020 patch of Simulator 21)

Append : Optional parameter – default is YES
Set this to YES or NO. Set to YES to append data to an existing file. Set to NO to overwrite an existing file.

SaveDataEPC("filename", objecttype, filter, GEFileType, SaveBuses, Append);

Use this action to save data in the GE EPC format.

"filename" : The file path and name to save.

objecttype : The type of object being saved.

filter : Optional parameter – default is to save all objects of specified type
See Using Filters in Script Commands section for more information on specifying the filter.

GEFileType : Optional parameter – default is to save with the latest version. Valid options:
GE (latest version), GE14-GE21

SaveBuses : Optional parameter – default is NO
Set to YES or NO. Set to YES to save any buses associated with the regulated bus of generators or switched shunts so that the scheduled voltage can also be saved.

Append : Optional parameter – default is YES
Set to YES or NO. Set to YES to append data to an existing file. Set to NO to overwrite an existing file.

SaveDataUsingExportFormat("filename", filetype, "FormatName", ModelToUse);

Use this action to save data in a user-defined format that has previously been defined.

"filename" : The file to save the data to

filetype : There are several options for the filetype

- AUXCSV : save as a comma-delimited auxiliary data file.
- AUX : save as a space-delimited auxiliary data file.
- CSV : save as a normal CSV file without the AUX file syntax. The first few lines of the text file will represent the object name and field variable names.
- CSVNOHEADER : save as a normal CSV text file, without the AUX file formatting. The object name and field variable names are NOT included. This option is useful when appending data of the same object type and field list into a common file.
- CSVCOLHEADER : save as a normal CSV without the AUX syntax and with the first row showing column headers you would see in a case information display

FormatName : The name of the Object Export Format Description to use.

ModelToUse : Optional parameter that indicates the model to use.

- FULL : Full-topology model. This is the default if the parameter is omitted.
- CONSOLIDATED : Consolidated planning-type model. This option will only work with the Topology Processing add-on.

SaveDataWithExtra("filename", filetype, objecttype, [fieldlist], [subdatalist], filter, [SortFieldList], [Header_List], [Header_Value_List], Transpose, Append);

Use this action to save data in a custom defined format. User-specified fields and field values can also be specified in the output. The syntax is identical to the SaveData command with the following exceptions:

Filetype : There are several options for the filetype

- CSV : save as a normal CSV file without the AUX file syntax. The first few lines of the text file will represent the object name and field variable names.
- CSVNOHEADER : save as a normal CSV text file, without the AUX file formatting. The object name and field variable names are NOT included. This option is useful when appending data of the same object type and field list into a common file.
- CSVCOLHEADER : save as a normal CSV without the AUX syntax and with the first row showing column headers you would see in a case information display

Data cannot be saved using AUX or AUXCSV filetypes with this command.

[Header_List] : Optional parameter – default is that no extra headers are included
This allows the specification of user-defined fields that will appear in the output. Headers should be specified as a list of comma delimited strings. A string should be enclosed in double quotes if the string contains a comma. Header strings cannot be blank.

[Header_Value_List] : Optional parameter – default is that all values are blank
Allows the specification of the values that should be assigned to the user-defined fields specified by Header_List. If specified, there must be as many values specified as there are headers. If not specified, all values are blank. Each object will use the same specified value for the specified field. To use different values for different objects and save these in the same file, make use of the CSVNOHEADER file format and filtering. Special keywords can be entered that will be replaced with their actual values. These include @BUILDDATE, @DATETIME, @DATE, @TIME, @VERSION, and @CASENAME.

(Following added in the February 13, 2020 patch of Simulator 21)

Append : Optional parameter – default is YES
Set this to YES or NO. Set to YES to append data to an existing file. Set to NO to overwrite an existing file.

For the Header_List and Header_Value_List, the input should be formatted in a manner to indicate how it should be written to the CSV. Any strings enclosed in double quotes will be stripped of the enclosers. Any strings containing double double quotes will have them replaced with single double quotes.

SaveObjectFields("filename", objecttype, [fieldlist]);

Use this action to save a list of fields available for the specified objecttype to a CSV file. Format of the file is `variablename, field, col header, description`.

"filename" : The file path and name to save.
objecttype : The type of object for which fields should be saved.
[fieldlist] : List of fields for which information will be saved. See the Specifying Field Variable Names in Script Commands topic for more information on specifying this list.

SelectAll(objecttype, filter);

Use this to set the selected property of objects of a particular type to true. A filter may optionally be specified to only set this property for objects that meet a filter.

objecttype : The objecttype being selected.
filter : Optional parameter – default is to set all objects of specified type
AREAZONE : Only objects that meet the area/zone/owner filters will be selected
"FilterName" : Only objects that meet the specified filter will be checked. See Using Filters in Script Commands section for more information on specifying the filtername.

SendtoExcel(objecttype, [fieldlist], filter, UseColumnHeaders, "workbookname", "worksheetname", [SortFieldList], [Header_List], [Header_Value_List], ClearExisting, RowShift, ColShift);

Use this action to mimic the behavior of the Send to Excel option found within a case information display.

objecttype : The type of object for which fields should be saved.
[fieldlist] : List of fields for which information will be saved. See the Specifying Field Variable Names in Script Commands topic for more information on specifying this list.
filter : Optional parameter – default is to send all objects of specified type
See the Using Filters in Script Commands section for more information on specifying the filter.
UseColumnHeaders : Set to YES or NO. YES signifies that the first row shows the Column Header, NO signifies that field variable names are used.

"workbookname" : Path and name of the workbook to save or modify. If no path is specified, the workbook will be saved or opened from the current directory. If the workbook already exists, it will be modified with a new worksheet, or if the worksheet is specified and already exists, the worksheet will be overwritten. If using Excel 2007 or later *.xlsm filetypes can be specified.

"worksheetname" : Optional parameter to specify the worksheet name to save. If blank, a new worksheet will be created, if a value is specified it will overwrite the data in any existing worksheet of that name.

[SortFieldList] : Optional parameter – the default is to do no sorting
This allows the specification of a sort order in which the data will be saved. The format is: [variablename1:+:0, variablename2:-:1] where

- variablename : is the name of the field to sort by. There is no limit to how many fields can be specified for sorting. For fields that require a location other than zero , variablename can be in the format fieldname:location.
- + or - : for the second parameter indicates sort ascending for + and sort descending for -. This parameter must be specified.
- 0 or 1 : for the third parameter 0 means case insensitive and do not use absolute value, 1 mean case sensitive or use absolute value. This parameter is optional.

[Header_List] : Optional parameter – default is that no extra headers are included
This allows the specification of user-defined fields that will appear in the output. Headers should be specified as a list of comma delimited strings. A string should be enclosed in double quotes if the string contains a comma. Header strings cannot be blank.

[Header_Value_List] : Optional parameter – default is that all values are blank
Allows the specification of the values that should be assigned to the user-defined fields specified by Header_List. If specified, there must be as many values specified as there are headers. If not specified, all values are blank. Each object will use the same specified value for the specified field. Special keywords can be entered that will be replaced with their actual values. These include @BUILDDATE, @DATETIME, @DATE, @TIME, @VERSION, and @CASENAME.

ClearExisting : Optional parameter – default is YES. YES means to clear the existing sheet

RowShift : Optional parameter – default is 0. Set to a positive integer to shift the paste of data downwards from row 1. Negative values treated as 0.

ColShift : Optional parameter – default is 0. Set to a positive integer to shift the paste of data to the right from column A. Negative values treated as 0.

SetData(objecttype, [fieldlist], [valuelist], filter);

Use this action to set fields for particular objects. If a filter is specified, then it will set the respective fields for all objects which meet this filter. Otherwise, if no filter is specified, then the key fields must be included in the field list so that the object can be found.

- objecttype : The objecttype being set.
- [fieldlist] : A list of fields to update.
- [valuelist] : A list of values to which to set the corresponding fields.
- filter : Optional parameter – if omitted the key fields and corresponding values must be included to identify the one object to update
  - ALL : Set data for all objects
  - AREAZONE : Only objects that meet the area/zone/owner filters will be set
  - SELECTED : Only objects whose Selected field = YES will be set
  - "FilterName" : Only objects that meet the specified filter will be set. See Using Filters in Script Commands section for more information on specifying the filtername.

UnSelectAll(objecttype, filter);

Same as SelectAll, but this action sets the selectected properties to false.

Case Related Actions

Available regardless of the mode

```
AutoInsertTieLineTransactions;
CalculateRXBGFromLengthConfigCondType(filter);
CaseDescriptionClear;
CaseDescriptionSet            ("text", Append);
ChangeSystemMVABase           (NewBase);
ClearSmallIslands;
ConditionVoltagePockets       (VoltageTreshold, AngleThreshold, filter);
CreateNewAreasFromIslands;
DetermineBranchesThatCreateIslands(Filter, StoreBuses,"filename", SetSelectedOnLines, FileType);
DeterminePathDistance         ([start], BranchDistMease, BranchFilter, BusField);
DetermineShortestPath         ([start],[end], BranchDistMeas, BranchFilter, Filename);
DoFacilityAnalysis            ("filename");
DirectionsAutoInsert          (Source, Sink, DeleteExisting, UseDisplayFilters, Start,
                              Increment);
EnterMode                     (mode);
EstimateVoltages              (filter);
GenForceLDC_RCC               (filter);
InitializeGenMvarLimits;
InjectionGroupsAutoInsert;
InjectionGroupCreate          ("Name", objecttype, InitialValue, Filter, Append);
InjectionGroupRemoveDuplicates;
InterfaceCreate               ("Name", DeleteExisting, ObjectType, Filter);
InterfaceRemoveDuplicates;
LoadEMS                       ("filename", filetype);
NewCase;
OpenCase                      ("filename", openfiletype, [LoadTransactions, StarBus,
                              HowToKeepDuplicates]);
RenameInjectionGroup          ("OldName", "NewName");
SaveCase                      ("filename", savefiletype, [PostCTGAGC, UseAreaZone]);
SaveGenLimitStatusAction      ("filename");
SaveJacobian                  ("JacFileName", "JIDFileName", FileType, JacForm)
SaveYbusInMatlabFormat        ("filename", IncludeVoltages);
Scale                         (scaletype, basedon, [parameters], ScaleMarker);
SetGenPMaxFromReactiveCapabilityCurve(filter);
SetParticipationFactors       (Method, ConstantValue, Object);
SetScheduledVoltageForABus    ([bus identifier], voltage);
SetSelectedFromNetworkCut     (SetHow, [BusOnCutSide], BranchFilter, InterfaceFilter,
                              DCLineFilter, Energized, NumTiers, InitializeSelected,
                              [ObjectsToSelect], UseAreaZone, UsekV, MinkV, MaxkV, LowerMinkV,
                              LowerMaxkV);
UpdateIslandsAndBusStatus;
WriteLimitMonitoringSettings  ("filename");
```

AutoInsertTieLineTransactions;

Use this action todelete all existing MW transactions and set the unspecified MW interchange for each area to zero. It then automatically creates a MW transaction between each pair of connected areas with a MW transaction exactly equal to the sum of the tie-line flows.

CalculateRXBGFromLengthConfigCondType(filter);

Use this action the go through branches in the power system and automatically recalculate the per unit R, X, G, and B values using the TransLineCalc tool. The branches Conductor Type, Tower Configuration, and Line Length will be passed to the TransLineCalc tool and new R, X, G and B values will be calculated. This is only available if you have installed the TransLineCalc tool.

filter : Optional parameter – default is to check all branches
Check only branches that meet the specified filter. See the Using Filters in Script Commands section for more information on specifying the filtername.

CaseDescriptionClear;

Use this action clear the case description of the presently open case.

CaseDescriptionSet("text", Append);

Use this action to set or append text to the case description.

"text"	:	Specify the text to set/append to the case description.
Append	:	YES – will append the text specified to the existing case description. NO – will replace the case description.

ChangeSystemMVABase(NewBase);

Use this action to change the system MVA base to the specified value and update all internal data structures to store values on the new base.

NewBase	:	New power system base in MVA.

ClearSmallIslands;

Use this action to identify the largest island and de-energize all other islands. The largest island is the island with the most buses. Small islands are de-energized by setting the status of all generators in those island to open.

ConditionVoltagePockets(VoltageThreshold, AngleThreshold, filter);

The goal of this script command is to find pockets of buses that may have bad initial voltage estimates and to get a better voltage estimate of these buses based on assuming that the voltages on buses outside these pockets are good. It will identify pockets of buses bounded by branches that meet the condition that the absolute value of the voltage difference across the branch is greater than VoltageThreshold or the absolute value of the angle difference across the branch is greater than AngleThreshold and the branch meets the specified filter.

VoltageThreshold	:	Per-unit voltage difference (absolute value) that determines if a branch can be considered when determining groups of radial buses.
AngleThreshold	:	Angle difference in degrees (absolute value) that determines if a branch can be considered when determining groups of radial buses.
filter	:	This is an optional parameter that is used to specify which branches are checked. If omitted all branches are considered.

ALL	:	All branches will be checked
SELECTED	:	Only branches whose Selected field = YES will be checked
AREAZONE	:	Only branches that meet the area/zone/owner filters will be checked
"FilterName"	:	Only branches that meet the specified filter will be checked. See Using Filters in Script Commands section for more information on specifying the filtername.

CreateNewAreasFromIslands;

(Added in the April 8, 2022 patch of Simulator 22)

Use this action to create permanent areas that match the area Simulator creates temporarily while solving the power flow. New areas are created if and area is on AGC, spans multiple vilable islands, and only one of those islands has more than one area in it.

DetermineBranchesThatCreateIslands(Filter, StoreBuses, "filename", SetSelectedOnLines, FileType);

Use this action to determine the branches whose outage results in island formation. Note that setting the Selected field will overwrite the Selected fields.

Filter : This parameter is used to specify which branches are checked.
- ALL : means all branches will be checked
- SELECTED : means only branches whose Selected field = YES will be checked
- AREAZONE : means only branches that meet the area/zone/owner filters will be checked
- "FilterName" : means only branches that meet the specified filter will be checked. See the Using Filters in Script Commands section for more information on specifying the filtername.

StoreBuses : YES to store the buses in the island to the output file

"filename" : file to which the results will be written. The format of the file is based on the auxiliary file format. Each branch that was checked will be followed by the list of buses that are islanded. The branch and bus information will be written in appropriate auxiliary file DATA format. If this is left blank, SetSelectedOnLines will be assumed to be YES.

SetSelectedOnLines : YES to set the SELECTED field to YES for branches that create islands

FileType : Optional parameter used to specify the format of the file. This is AUX by default.
- AUX : The saved file is based on an auxiliary file data format. Each branch that causes an island appears in the file in the auxiliary file data format followed by a auxiliary file bus data section containing all of the buses that are islanded by the preceeding branch.
- CSV : The saved file is a comma-delimited text file. Each unique bus/branch pair appears on a single line. A unique bus/branch pair is determined by a bus that is islanded and a particular branch that causes it to be islanded. A header appears in the file specifying the fields used to identify the branch and bus in each record.

DeterminePathDistance([start], BranchDistMeas, BranchFilter, BusField);

Use this action to calculate a distance measure at each bus in the entire model. The distance measure will represent how far each bus is from the starting group specified. The distance measure can be related to impedance, geographical distance, or simply the number of nodes.

[start] : The starting location. The starting location may be either an Area, Zone, SuperArea, Substation, Injection Group, or Bus. Format of string is
- [Area Num], [Area "Name"], or [Area "label"]
- [Zone Num], [Zone "Name"], or [Zone "label"]
- [SuperArea "Name"] or [SuperArea "label"]
- [Substation Num] or [Substation "label"]
- [InjectionGroup "Name"] or [InjectionGroup "label"]

BranchDistMeas : is either X, Z, Length, Nodes, or a field variable name for a branch.
- X : means use the series reactance,
- Z : means use sqrt(X^2 + R^2),
- Length : means us the Length field, and
- Nodes : means treat each branch as a length of one.
- "Variablename" : Otherwise use any Branch object field variable name.

BranchFilter : is either All, Selected, Closed or the name of a branch Advanced Filter. This parameter is used to specify which branch can be traversed at all.

- All : means all branches can be traversed
- Selected : means only branches whose Selected field = YES can be traversed
- Closed : means only branches that are CLOSED can be traversed.
- "FilterName" : See the Using Filters in Script Commands section for more information on specifying the filtername.

BusField : is the variable name of a Bus field. This field is populated with the minimum distance from the Start Place to that bus. All buses in the start group will have a distance measure of zero. Buses which cannot be reached from the start group will have a distance measure of -1.

DetermineShortestPath([start], [end], BranchDistanceMeasure, BranchFilter, Filename);

Use this action to calculate the shortest path between a starting group and an ending group. The results will be written to a textfile specified by filename. In the text file, the first bus listed will be in the end grouping and the last bus listed will be the start grouping. The result text file will have a line for each bus passed. Each line will contain three entries delimited by a space: "Number DistanceMeasure Name".

[start] : same as the starting place for the DeterminePathDistance script command

[end] : same as the starting place for the DeterminePathDistance script command

BranchDistanceMeasure : same as for DeterminePathDistance script command

BranchFilter : same as for DeterminePathDistance script command

Filename : is a filename (may need to be enclosed in quotes) to which the results will be written.

DoFacilityAnalysis ("Filename");

Do Facility Analysis (Minimum Cut) is used to determine the branches that would isolate the Facility from the External region as specified in the Select Bus Dialog in the Simulator Tool dialog. It is assumed that the user will set the options before using the script command. The script will be used to identify the minimum number of branches that need to be opened or removed from the system in order to isolate the Facility (power system device) from an External region.

"Filename" : The auxiliary file to which the results will be written. The results will show the buses of the different paths in a data section consisting of the buses that form the respective path. Also it will show the branches of the minimum cut.

DirectionsAutoInsert(Source, Sink, DeleteExisting, UseAreaZoneFilters, Start, Increment);

Use this action to auto-insert directions to the case

Source : AREA, ZONE, or INJECTION GROUP – specifies what to use as source

Sink : AREA, ZONE, INJECTION GROUP, or SLACK – specifies what to use as sink.

DeleteExisting : YES – to delete existing direction; NO to not do that.

UseAreaZoneFilters : YES – to filter Area/Zones by filter.

Start : The starting number for the new directions added.

Incremement : The increment for subsequent directions.

EnterMode(mode);

This action will change the mode in which Simulator is operating. This is especially necessary when creating new case objects for which you are required to be in EDIT mode. Simulator will automatically change the mode to RUN for script actions that require that mode.

Mode : The mode to enter, either RUN or EDIT.

EstimateVoltages(filter);

Added in February 6, 2020 patch of Simulator 21

This will estimate voltages and angles at the buses that meet the filter based on voltages and angles at surrounding buses (these are buses that do not meet the filter). It is assumed that the voltages and angles are correct at the surrounding buses, and in order for this command to work, there must be some buses that do not meet the filter.

filter : See the Using Filters in Script Commands section for more information on specifying the filter. A valid filter must be specified. An error will result if a blank filter is specified.

GenForceLDC_RCC(filter);

Use this action to force generators in the case onto line drop / reactive current compensation. The present voltage at the point at which the generator is controlling based on the line drop/reactive current compensation impedance is calculated, and the setpoint of the generator is set to this value. If the absolute value of the line drop/reactive current compensation impedance is less than or equal to 2*10-6*MVA Base, the generator will regulate its terminal bus and the setpoint voltage is set to the present value of the terminal bus voltage. For a typical case with an MVA Base of 100 MVA, this value is 0.0002.

filter : Optional parameter – default is to set all generators
See the Using Filters in Script Commands section for more information on specifying the filter.

InitializeGenMvarLimits;

Use this action to initialize all generators in the case so that they are appropriately marked as being at Mvar limits or not. This could be useful if manually setting the Mvar output of generators or changing their limits.

InjectionGroupsAutoInsert;

Use this action to automatically insert injection groups according to the options specified in the "IG_AutoInsert_Options" object. The settings available with this object represent what is seen on the Auto Insert Injection Groups Dialog.

InjectionGroupCreate("Name", objecttype, InitialValue, filter, Append);

This action will create or modify an injection group with participation points of a single object type that meet a filter. Repeated calls to this script command can be used to define an injection group with different object types.

"Name"	:	Name of the injection group to create or modify.
objecttype	:	Type of object to be included in the injection group. Valid options are GEN, LOAD, SHUNT, or BUS.
InitialValue	:	Set this to a floating point value to indicate the value of the participation factor to use with each point. Special keywords can also be used to indicate dynamically determined values. (The participation point AutoCalc field is set to YES). The following options are available: Generators: PRESENT, MAX GEN INC, MAX GEN DEC, and MAX GEN MW Loads: LOAD MW Switched Shunts: MAX SHUNT INC, MAX SHUNT DEC, and MAX SHUNT MVAR All object types: <FIELD>variablename can be used to reference a field associated with the object in the participation point. <EXPRESSION>modelexpressionname can be used to reference a Model Expression.
filter	:	Specify a filter to select the objects to add to the injection group. See the Using Filters in Script Commands section for more information on specifying the filter.
Append	:	Optional parameter – default is YES. Set to YES or NO. Set to YES to add new participation points based on the current settings to an injection group that exists with the same Name. Set to NO to delete all existing points before adding new points to an injection group that already exists with the same Name.

InjectionGroupRemoveDuplicates;

(Added in the November 11, 2021 patch of Simulator 22)

This action will search through all injection groups in the case looking for injection groups with the same elements. For an injection group being the same means that the injection group would have the same behavior as a duplicate. This means that the elements must be the same. The elements must also have the same initial value, participation factor, and auto calculation settings.

InterfaceCreate("Name", DeleteExisting, ObjectType, Filter);

This action will create or modify an interface with elements of a single object type that meet a filter. Repeated calls to this script command can be used to define an interface with different object types.

"Name"	:	Name of the interface to create or modify.
DeleteExisting	:	Set to YES to delete an interface with the same Name. Set to NO to append elements to an interface with the same Name.
ObjectType	:	Type of object to be included in the interface. Valid options are BRANCH or INTERFACE.
Filter	:	Specify a filter to select the objects of the specified ObjectType to add to the interface. See the Using Filters in Script Commands section for more information on specifying the filter. Special filter keywords of SELECTED and AREAZONE cannot be used with this script command.

InterfaceRemoveDuplicates;

(Added in the November 11, 2021 patch of Simulator 22)

This action will search through all interfaces in the case looking for interfaces with the same elements. For an interface being the same means that the reported flow is the same. In addition to having all the same elements, the interface elements must also have the same monitoring direction and weighting.

LoadEMS("filename", filetype);

Use this to open any EMS file. This can be a full case, a contingency file, or remedial action scheme file. Simulator will determine the type of information being loaded and how to handle it based on the records in the file.

"filename"	:	Name of the file to open. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.
filetype	:	Type of file to be loaded. Currently the only option is AREVAHDB.

NewCase;

This action clears out the existing case and open a new case from scratch.

OpenCase("filename", OpenFileType,[LoadTransactions,StarBus,HowToKeepDuplicates**]);**

OpenCase("filename", OpenFileType,[MSLine,VarLimDead,PostCTGAGC,MSLineDummyBus]);

This action will open a case stored in "filename" of the type OpenFileType. Different sets of optional parameters apply for the PTI and GE file formats. The LoadTransactions and Star bus parameters are available for writing to RAW files. MSLine, VarLimDead, and PostCTGAGC are for writing EPC files.

"filename" : The file to be opened.

OpenFileType : An optional parameter indicating the format of the file being opened. If none is specified, PWB will be assumed. It may be one of the following strings:

PWB, PTI (latest version), PTI23-PTI34
GE (latest version), GE14-GE21, CF
AUX, UCTE, AREVAHDB, OPENNETEMS

LoadTransactions : valid for PTI RAW format only

- YES : load transactions when opening case.
- NO : do not load transactions when opening case.
- DEFAULT : follow default behavior.

StarBus : valid for PTI RAW format only

- NEAR : star buses are numbered starting after the near bus number
- MAX : star buses are numbered starting with the maximum bus number
- VALUE : star bus numbering will start at value

(Following added in July 10, 2020 patch of Simulator 21)

HowToKeepDuplicates : valid for PTI RAW format only

- LAST : keep only the LAST duplicate of any device in each section of a PTI RAW file
- ALL : keep ALL duplicates of any device in each section of a PTI RAW file, and make them each unique by dynamically changing their IDs

MSLine : valid for GE EPC format only

- MAINTAIN : maintain multi-section lines
- EQUIVALENCE : equivalence mult-section lines

VarLimDead : valid for GE EPC format only

- Number : set the GE var limit deadband

PostCTGACG : valid for GE EPC format only
set to YES to populate the generator field Post-CTG Prevent Response based on the EPC file's generator base load flag.

MSLineDummyBus : Optional parameter – default is specified with Simulator Options
valid for GE EPC format only
Specifies how the dummy bus numbers for multi-section lines are determined.
FROM : starting at the from bus number
MAX : starting with the maximum bus number
Value or range – starting with the specified value or will be numbered within the specified range. If an unused bus number within the specified range cannot be found, the numbering will start at the highest number specified in the range.

RenameInjectionGroup("OldName", "NewName");

This action will change the name of an existing injection group.

"OldName" : Name of the existing injection group.
"NewName" : New name of the existing injection group.

SaveCase("filename", SaveFileType, [PostCTGAGC, UseAreaZone]);

SaveCase("filename", SaveFileType, [AddCommentForObjectLabels]);

This action will save the case to "filename" in the format SaveFileType.

"filename" : The file name in which to save the information.
SaveFileType : An optional parameter saying the format of the file to be saved. If none is specified, then PWB will be assumed. It may be one of the following strings
PWB (latest version), PWB5-PWB20
PTI23-PTI34
GE14-GE21, CF
AUX, AUXSECOND, AUXLABEL, AUXNETWORK, UCTE
PostCTGAGC : An optional parameter, only valid for GE EPC format. If the Governor Response Limits field for a generator is set to Down Only or Fixed, the base load flag will be written as 1 or 2, respectively, and this option is ignored. This option is only used when a generator's Governor Response Limits field is set to Normal. If this option is set to YES and not ignored, the base load flag in the EPC file is based on the Post-Contingency Prevent AGC Response setting. If preventing post-contingency AGC, the base load flag is set to 2. If not preventing post-contingency AGC, the base load flag is set to 0.
UseAreaZone : An optional parameter, only valid for GE EPC format. YES limits the entries in the EPC file based on the area/zone/owner filter (NO by default)
AddCommentsForObjectLabels
: An optional parameter, only valid for PTI RAW format. Default is NO. YES adds object labels to the end of data records when saving a RAW file.

SaveGenLimitStatusAction("filename");

Use this action to save Mvar information about generators in a text file. The information saved includes the generator bus number, generator ID, Mvar, Max Mvar, Min Mvar, AVRable flag (user specified), and internal AVRable flag (set by Simulator). This information is useful for debugging.

"filename" : Name of the text file in which to save the generator information.

SaveJacobian("JacFileName", "JIDFileName", FileType, JacForm);

Use this action to save the Jacobian Matrix to a text file or a file formatted for use with Matlab

"JacFileName"	:	File in which to save the Jacobian.
"JIDFileName"	:	File to save a description of what each row and column of the Jacobian represents.
FileType	:	One of: M – Matlab form TXT – Text file EXPM – Save Jacobian in Exponential form (ex. 1.2E-2) in Matlab form
JacForm	:	One of: R – AC Jacobian in Rectangular coordinates P – AC Jacobian in Polar coordinates DC – B' matrix of DC power flow

SaveYbusInMatlabFormat("filename", IncludeVoltages);

Use this action to save the YBus to a file formatted for use with Matlab

"filename"	:	File in which to save the YBus.
IncludeVoltages	:	YES includes the per unit bus voltages in the file; NO does not include.

Scale(scaletype, basedon, [parameters], scalemarker);

Use this action to scale the load and generation in the system. This script command should be used in conjunction with the SCALE_OPTIONS object that specifies additional options necessary for the scaling that are not set through the script command.

scaletype	:	The objecttype begin scaled. Must be either LOAD, GEN, INJECTIONGROUP, or BUSSHUNT.
basedon	:	One of: MW : parameters are given in MW, MVAR units. FACTOR : parameters a factor to multiple the present values by.
[parameters]	:	These parameters have different meanings depending on ScaleType. LOAD : [MW, MVAR] or [MW]. If you want to scale load using constant power factor, then do not specifying a MVAR value. GEN : [MW] INJECTIONGROUP: [MW, MVAR] or [MW] . If you want to scale load using constant power factor, then do not specifying a MVAR value. BUSSHUNT : [GMW, BCAPMVAR, BREAMVAR]. The first values scales G shunt values, the second value scales positive (capacitive) B shunt values, and the third value scales negative (reactive) B shunt values The Scale script command also allows using the [parameters] input to specify the new value or scale factor through a field with the object type to scale. To use this option, the [parameters] input should contain field variable names instead of numeric values. When using a field rather than value, the scaling will be done by individual object rather than the aggregation of all objects selected for scaling.

scalemarker : This value specifies whether to look at an element's bus, area or zone to determine whether it should be scaled.

- BUS : Means that elements will be scaled according to the Scale property of the element's terminal bus.
- AREA : Means that elements will be scaled according to the Scale property of the element's Area. Note that it is possible for the area of a load, generator, or switched shunt to be different than the terminal bus's area.
- ZONE : Means that elements will be scaled according to the Scale property of the element's Zone. Note that it is possible for the zone of a load, generator, or switched shunt to be different than the terminal bus's zone.
- OWNER : Means that elements will be scaled according to the Scale property of the element's Owner. Note that it is possible for the Owner of a load, generator, or switched shunt to be different than the terminal bus's Owner.

Examples: Two different wasy to scale three areas to a particular load value

```
Script
{
  SetData(Area, [Scale], ["NO"], All);          // Sets the scale field of all areas
to NO
  SetData(Area, [Number, Scale], [111, "YES"]); // For area, set Scale=YES
  Scale(LOAD, MW, [1111.1],AREA);               // Do the scaling for particular area
  SetData(Area, [Number, Scale], [111, "NO"]);  // reset Scale back to NO
  SetData(Area, [Number, Scale], [333, "YES"]); // For area, set Scale=YES
  Scale(LOAD, MW, [3333.3],AREA);               // Do the scaling for particular area
  SetData(Area, [Number, Scale], [333, "NO"]);  // reset Scale back to NO
  SetData(Area, [Number, Scale], [444, "YES"]); // For area, set Scale=YES
  Scale(LOAD, MW, [4444.4],AREA);               // Do the scaling for particular area
  SetData(Area, [Number, Scale], [444, "NO"]);  // reset Scale back to NO
}
Script
{
  SetData(Area, [Scale], ["NO"], All);
  SetData(Area, [Number, Scale, CustomFloat:5], [111, "YES", 1111.1]);
  SetData(Area, [Number, Scale, CustomFloat:5], [333, "YES", 3333.3]);
  SetData(Area, [Number, Scale, CustomFloat:5], [444, "YES", 4444.4]);
  Scale(LOAD,MW,["CustomFloat:5"],AREA);
}
```

SetGenPMaxFromReactiveCapabilityCurve(filter);

Use this action to change the Maximum MW output of generators which use a capability curve, equal to the second-to-last MW point in the capability curve if the last Max Mvar point on the capability curve is smaller than 0.001 Mvar. If the present MW output is higher than this new Max MW value, then Max MW is set to the present MW output..

filter : optional parameter which is either Selected, Closed or the name of a branch Advanced Filter. This parameter is used to specify which generators are processed. If blank, all generators are processed.

- Selected : means only generators whose Selected field = YES can be processed
- AREAZONE : means process those generators that meet the area/zone/owner filters.
- "FilterName" : See the Using Filters in Script Commands section for more information on specifying the filtername.

SetParticipationFactors(Method, ConstantValue, Object);

Use this action to modify the generator participation factors in the case

Method : The formula used to calculate the participation factors for each generator. It may be one of the following strings:
MAXMWRAT – base factors on the maximum MW ratings.
RESERVE – base factors on the (Max MW rating – Present MW).
CONSTANT – set factors to a constant value.

ConstantValue : The value used if CONSTANT method is specified. If CONSTANT method is not specified, enter 0 (zero).

Object : Specify which generators to set the participation factor for.
[Area Num], [Area "name"], [Area "label"]
[Zone Num], [Zone "name"], [Zone "label"]
SYSTEM
AREAZONE or DISPLAYFILTERS

SetScheduledVoltageForABus([bus identifier], voltage);

Use this action to set the stored scheduled voltage, vsched, for a bus according to how this is defined in the EPC format. This value is not used by Simulator but is stored for purposes of writing out to an EPC file. The setpoint voltages for generators and switched shunts regulating the specified bus are also set to the new voltage. The regulation range for switched shunts is modified for the new setpoint voltage according to how this is defined in the EPC format: vband = (VHigh-VLow)/2 with newVHigh = voltage+vband and new VLow = voltage-vband.

[bus identifier] : specifies bus

voltage : the new voltage

SetSelectedFromNetworkCut(SetHow, [BusOnCutSide], BranchFilter, InterfaceFilter, DCLineFilter, Energized, NumTiers, InitializeSelected, [ObjectsToSelect], UseAreaZone, UsekV, MinkV, MaxkV, LowerMinkV, LowerMaxkV);

Use this action to set the Selected field of specified object types if they are on the specified side of a network cut created by specified branches, interfaces, and/or dc lines.

SetHow : Set to YES or NO. This is the value to which the Selected field will be set if an object is within the network cut.

[BusOnCutSide] : Specify the bus that is on the desired side of the network cut. Objects that are on the same side as this bus will have their Selected field set.

At least one of the following filters MUST not be blank:

BranchFilter : Specify a filter to select the branches that define the network cut. See the Using Filters in Script Commands section for more information on specifying the filter. A blank filter means that no branches are selected.

InterfaceFilter : Specify a filter to select the interfaces that define the network cut. See the Using Filters in Script Commands section for more information on specifying the filter. A blank filter means that no interfaces are selected.

DCLineFilter : Specify a filter to select the dc lines that define the network cut. See the Using Filters in Script Commands section for more information on specifying the filter. A blank filter means that no dc lines are selected.

Energized : Set to YES or NO. Set to YES to only include branches with a closed status when traversing branches to determine which side of the network cut each bus is on. This option does not apply to the branches that are specified to define the network cut. Those branches will not be traversed regardless of their status.

NumTiers	:	Once the network cut has been defined by a set of branches and the bus defining which side of the cut is being examined has been chosen, this value indicates that buses will be included within this number of tiers of the network cut boundary, on the opposite side of the cut as the specified bus. If the number of tiers is set to zero, the buses examined will only be those on the same side of the cut as the specified bus.
InitializeSelected	:	Set to YES or NO. Set to YES to set the Selected field for all objects to the opposite of the value specified in the SetHow parameter.
[ObjectsToSelect]	:	Comma separated list of object types enclosed in square brackets. These objects will have their Selected fields set if they are within the network cut. Valid options are: BRANCH, BUS, DCTRANSMISSIONLINE, GEN, LOAD, and SHUNT.
UseAreaZone	:	Optional parameter – default is NO. Set to YES or NO. Set to YES to only set the Selected field for objects that are within the network cut that meet the area/zone/owner filter.
UsekV	:	Optional parameter – default is NO. Set to YES or NO. Set to YES to only set the Selected field for objects that are within the network cut and within the nominal kV range specified.
MinkV	:	Optional parameter – default is 0. An object's nominal kV must be greater than or equal to this value to have its Selected field set if it is also in the network cut. Branches can have different nominal voltages at each terminal; the largest nominal voltage must be greater than or equal to this value.
MaxkV	:	Optional parameter – default is 9999. An object's nominal kV must be less than or equal to this value to have its Selected field set if it is also in the network cut. Branches can have different nominal voltages at each terminal; the largest nominal voltage must be less than or equal to this value.
LowerMinkV	:	Optional parameter – default is 0. This value is only used with branches. Branches can have different nominal voltages at each terminal; the smallest nominal voltage must be greater than or equal to this value.
LowerMaxkV	:	Optional parameter – default is 9999. This value is only used with branches. Branches can have different nominal voltages at each terminal; the smallest nominal voltage must be less than or equal to this value.

UpdateIslandsAndBusStatus;

Changes to branch and generator status impact islands and whether or not buses are connected. Islands and bus status are always updated at the beginning of a power flow solution if necessary, but this script command makes it convenient to update this information without requiring a power flow solution.

WriteLimitMonitoringSettings("filename");

Use this to save Limit Monitoring Settings to an auxiliary file.

"filename"	:	Name of the auxiliary file to save. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.

Distributed Computing Actions

Available regardless of the mode

```
EnterDistMasterPassword       (Password);
VerifyDistributedComputersAvailable;
```

EnterDistMasterPassword(Password);

Use this action to enter the master password used to unlock distributed machine login credentials.

Password : Password that must be specified to unlock the credentials.

VerifyDistributedComputersAvailable;

Allows distributed computer status to be verified.

Oneline Actions

Available regardless of the mode

```
CloseOneline                    ("OnelineName");
EditMultipleOnelineAction       ("Path", LinkType, SaveFileType);
ExportBusView                   ("filename", "bus key", ImageType, Width, Height,
                                [ExportOptions]);
ExportOneline                   ("filename", "OnelineName", ImageType, "view", FullScreen,
                                ShowFull, [ExportOptions]);
ExportOnelineAsShapeFile        ("filename", "OnelineName", "shapefileExportDescriptionName",
                                UseLonLat, PointLocation);
ImportDDLAsTranslation          ("filename");
LoadAXD                         ("filename", "OnelineName", CreateIfNotFound);
OpenOneline                     ("filename", "view", FullScreen, ShowFull, LinkMethod, Left, Top,
                                Width, Height);
RelinkAllOpenOnelines;
SaveOneline                     ("filename", "OnelineName", SaveFileType);
OpenBusView                     ("Bus Key", ForceNewWindow);
OpenSubstationView              ("Substation Key", ForceNewWindow);
```

CloseOneline("OnelineName");

Use this action to close an open oneline diagram without saving it. If the name is omitted, the last focused oneline diagram will be closed.

"OnelineName" : The name of the oneline diagram to close.

EditMultipleOnelineAction("Path", LinkType, SaveFileType);

Use this action to convert all files with a PWD extension in a specified directory to a new format. The files will be saved with the same name but with an extension appropriate for the SaveFileType.

"Path" : Specify a valid path where the files are located.

LinkType : Specify the key field identifier to use for linking objects in the oneline diagrams to a power flow case. Options are NUMBER, NAMENOMKV, and LABEL.

SaveFileType : Specify the new format for the oneline diagrams. Valid options are: PWB, PWB5-PWB20, and AUX.

ExportBusView("filename", "bus key", ImageType, Width, Height, [ExportOptions]);

(Added in the November 17, 2022 patch of Simulator 22)

Use this action to export an image of a bus view oneline diagram to a file.

"filename" : Name of the file in which the exported image will be saved. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.

"Bus key" : The specific bus. The format is the object type followed by the key fields

ImageType : The type of image to save. Valid options are: BMP, GIF, JPG, EMF, WMF, and PDF.

Width : Width of the saved image in pixels.

Height : Height of the saved image in pixels.

[ExportOptions] : Optional parameter
This is a comma separated list of options based on the ImageType that is being exported.

When exporting an image of type JPG, the following options can be specified:

ImageQuality : Quality of the image specified from 1 to 100 with 100 being the highest quality image. The larger the image quality the larger the resulting file will be. Default is 80.

ResolutionScalar : The resolution can be changed from the default resolution by adjusting by this scalar. To increase the resolution set the scalar to something greater than 1. Increasing the resolution will also increase the file size. Default is 1.

When exporting an image of type GIF, the following options can be specified:

NumFrames : GIF images can be animated by introducing multiple frames. This value specifies the number of frames. Default is 1.

FrameDelay : Number of seconds to wait between frames. Default is 0.1.

ResolutionScalar : The resolution can be changed from the default resolution by adjusting by this scalar. To increase the resolution set the scalar to something greater than 1. Increasing the resolution will also increase the file size. Default is 1.

ExportOneline("filename", "OnelineName", ImageType, "view", FullScreen, ShowFull, [ExportOptions]);

Use this action to export an image of the open oneline diagram to a file containing the specified image type.

"filename" : Name of the file in which the exported image will be saved. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.

"OnelineName" : The name of the oneline diagram to export. The oneline diagram must be open. Use the OpenOneline script command if necessary to open the appropriate oneline.

ImageType : The type of image to save. Valid options are: BMP, GIF, JPG, EMF, WMF.

"view" : Optional parameter. The view name that should be opened. Pass an empty string to denote no specific view.

FullScreen : Optional parameter with default of NO. Set to YES or NO. YES means that the oneline diagram will be open in full screen mode. If this parameter is not specified, then NO is assumed.

ShowFull : Optional parameter with default of NO. Set to YES to open the oneline and apply the Show Full option. Set to NO to open the oneline and leave the oneline as is.

[ExportOptions] : Optional parameter
This is a comma separated list of options based on the ImageType that is being exported.

When exporting an image of type JPG, the following options can be specified:

ImageQuality : Quality of the image specified from 1 to 100 with 100 being the highest quality image. The larger the image quality the larger the resulting file will be. Default is 80.

ResolutionScalar : The resolution can be changed from the default resolution by adjusting by this scalar. To increase the resolution set the scalar to something greater than 1. Increasing the resolution will also increase the file size. Default is 1.

When exporting an image of type GIF, the following options can be specified:

NumFrames : GIF images can be animated by introducing multiple frames. This value specifies the number of frames. Default is 1.

FrameDelay : Number of seconds to wait between frames. Default is 0.1.

ResolutionScalar : The resolution can be changed from the default resolution by adjusting by this scalar. To increase the resolution set the scalar to something greater than 1. Increasing the resolution will also increase the file size. Default is 1.

ExportOnelineAsShapeFile("filename", "OnelineName", "ShapeFileExportDescriptionName", UseLonLat, PointLocation);

Use this action to save an open oneline diagram to a shapefile.

"filename" : The file name of the shapefile to save.

"OnelineName" : The name of the oneline diagram to save to a shapefile. The oneline diagram must be open. Use the OpenOneline script command if necessary to open the appropriate oneline.

"ShapeFileExportDescriptionName" : Name of the ShapeFile Export Description to use when saving the shapefile.

UseLonLat : Set to YES or NO. YES means that the coordinates of objects on the oneline diagram will be saved using longitude,latitude. This will only be true if a valid map projection is in use with the oneline diagram. Otherwise, the coordinates will be saved in x,y. If this parameter is set to NO, the coordinates will be saved in x,y. If this parameter is not specified, YES is assumed.

PointLocation : Determines where points are specified – object centers, or the upper left corner. Specify “center” to define points as the shape centers, or “ul” to define them as the upper left corner of the shapes. If not specified, upper left is assumed.

ImportDDLAsTranslation("filename");

This loads an ESET DDL and converts some definitions into translations within Simulator. This currently only works for keyset definitions that set links to open onelines.

"filename" : The name of file to load.

LoadAXD("filename", "OnelineName", CreateIfNotFound)

Use this action to apply a display auxiliary file to an open oneline diagram.

"filename" : The file name of the display auxiliary file to load.

"OnelineName" : The name of the oneline diagram to which to apply the display auxiliary file. If the oneline is not already open, the OpenOneline script command can be used to open the appropriate oneline. If the specified oneline is not open, a new one will be created with the given name.

CreateIfNotFound : Set to YES or NO. YES means that objects which cannot be found will be created while reading in DATA sections from filename. If this parameter is not specified, NO is assumed.

OpenOneline("filename", "view", FullScreen, ShowFull, LinkMethod, Left, Top, Width, Height);

Use this action to open a oneline diagram. When using SimAuto, this action cannot be used to actually view a oneline. This script can be used in SimAuto to associate onelines with a PWB file. Any oneline that is opened using the script command and while the case is saved will opened in the GUI once the case is reopened.

"filename"	:	The file name of the oneline diagram to open. Wildcards are allowed when opening a DDL file type. This is useful for loading DDL files via browsing patch searches.
"view"	:	The view name that should be opened. Pass an empty string to denote no specific view.
FullScreen	:	Set to YES or NO. YES means that the oneline diagram will be open in full screen mode. If this parameter is not specified, then NO is assumed.
ShowFull	:	Optional parameter. Set to YES to open the oneline and apply the Show Full option. Set to NO to open the oneline and leave the oneline as is. Default is NO if not specified.
LinkMethod	:	Optional Parameter that controls oneline linking. LABELS, NAMENOMKV, and NUMBER will link using the respective key fields.
Left	:	Optional with default of 0. Value between 0 and 100 that indicates the location of the left edge of the oneline as a percentage of the Simulator/Retriever window width.
Top	:	Optional with default of 0. Value between 0 and 100 that indicates the top edge of the oneline as a percentage of the Simulator/Retriever window height.
Width	:	Optional with default of 0. Value between 0 and 100 that indicates the width of the oneline as a percentage of the Simulator/Retriever window width.
Height	:	Optional with default of 0. Value between 0 and 100 that indicates the height of the oneline as a percentage of the Simulator/Retriever window height.

RelinkAllOpenOnelines;

Making modifications to the power flow case could cause objects on a oneline from becoming unlinked. This action will attempt to relink all objects on all open onelines.

SaveOneline("filename", "OnelineName", SaveFileType);

Use this action to save an open oneline diagram to file

"filename"	:	The path and file name of the file to save. If a full path is not specified, then the file is saved to the current directory.
"OnelineName"	:	Name of the open oneline to save.
SaveFileType	:	Type of file to save. Valid options are AXD, PWB, and PWB5-PWB20. If omitted, PWB, which is the most recent version, will be assumed. Note the use of "PWB" instead of "PWD" is not a typo. The version of the PWD file corresponding to the PWB version will be used.

OpenBusView("Bus key", ForceNewWindow);

Opens the Bus View to a particular bus specified in the first parameter.

"Bus key"	:	The specific bus. The format is the object type followed by the key fields
ForceNewWindow	:	Optional with default of NO. Set to YES to force a new bus view to be opened regardless. If NO, then if a bus view is already open the command will update that bus view instead of opening a new one.

OpenSubView("Substation key", ForceNewWindow);

Opens the Substation View to a particular substation specified in the first parameter.

"Substation key"	:	The specific substation. The format is the object type followed by the key fields
ForceNewWindow	:	Optional with default of NO. Set to YES to force a new substation view to be opened regardless. If NO, then if a substation view is already open the command will update that substation view instead of opening a new one.

User Interface Actions

Available regardless of the mode

```
MessageBox                  ("text");
ObjectFieldsInputDialog     ("ObjectIDString", [fieldlist]);
OpenDataView                ("ObjectIDString", "DataGridIDString");
```

MessageBox("text");

Use this action to open a dialog box that will display the entered text. This script command will fail if using the SimAuto add-on.

"text"	:	Text that will appear in the dialog box.

ObjectFieldsInputDialog("ObjectIDString", [fieldlist], "DialogCaption", "DialogExplain", [LabelCaptions], [TabBreaks], [TabCaptions], [RowBreaks], [RowCaptions], [ColBreaks], [ColCaptions]);

Use this action to open a dialog box displaying the list of specified fields for the specified object. This will allow the fields to be modified in the same manner as they can through case information displays. This script command will fail if using the SimAuto add-on.

"ObjectIDString"	:	The specific object for which to display fields. The format is the object type followed by the key fields used to identify the object. Examples: "Bus 234891", "Gen 16445 'A'", "Branch 1239 1234 'AB'".
[fieldlist]	:	A list of fields to to display for the specified object.
"DialogCaption"	:	Optional with default of blank. This is the caption that will appear on the dialog.
"DialogExplain"	:	Optional with default of blank. This is an explanation that will appear in a text at the top of the dialog underneath the caption.
[LabelCaptions]	:	Optional with default of []. Inside brackets, you may enter a comma-delimited list of captions that will appear with the respective fields. The captions must be enclosed in double quotes if there are any commas in the string. If now label captions are specified, then the concise variable names will be used to indicate what each field is
[TabBreaks]	:	Optional with default of []. Inside brackets, you may enter a comma-delimited list of integers that indicate that a tab break occurs before the field at the particular index. The fields are indexed starting at zero. The dialog that appears will be created with the first "tab" representing a panel at the TOP of the dialog. This top panel will be made a fixed height so that all rows of fields can be seen. Any subsequent tabs will be placed inside a Tabbed control. The tabbed control will take up the remainder of the size of the dialog.
[TabCaptions]	:	Optional with default of []. Inside brackets, you may enter a comma-delimited list of captions that will appear with the respective tab break. Each tab break will represent a TAB on the tabbed control. These will be the captions. If nothing is specified, the captions will simply numbered.
[RowBreaks]	:	Optional with default of []. Inside brackets, you may enter a comma-delimited list of integers that indicate that a row break occurs before the field at the particular index. The fields are indexed starting at zero. Each tab of the dialog will be drawn with controls optionally grouped into rows and then these rows optionally grouped into columns. A particular "cell" of this table can then have multiple fields inside it.
[RowCaptions]	:	Optional with default of []. Inside brackets, you may enter a comma-delimited list of captions that will appear with the respective group box that starts with the field at this index. The group box will contain all fields up until the next Column or Row break. Blank captions are also allowed, in which case a group box is not drawn.

[ColBreaks] : Optional with default of []. Inside brackets, you may enter a comma-delimited list of integers that indicate that a column break occurs before the field at the particular index. The fields are indexed starting at zero. Each tab of the dialog will be drawn with controls optionally grouped into rows and then these rows optionally grouped into columns. A particular "cell" of this table can then have multiple fields inside it.

[ColCaptions] : Optional with default of []. Inside brackets, you may enter a comma-delimited list of captions that will appear with the respective group box that starts with the field at this index. The group box will contain all fields up until the next Column or Row break. Blank captions are also allowed, in which case a group box is not drawn.

The following image depicts what the resulting dialog would show for the following script command. Note the field list is abbreviated but for this example there are 21 fields listed in the same manner as other script commands.

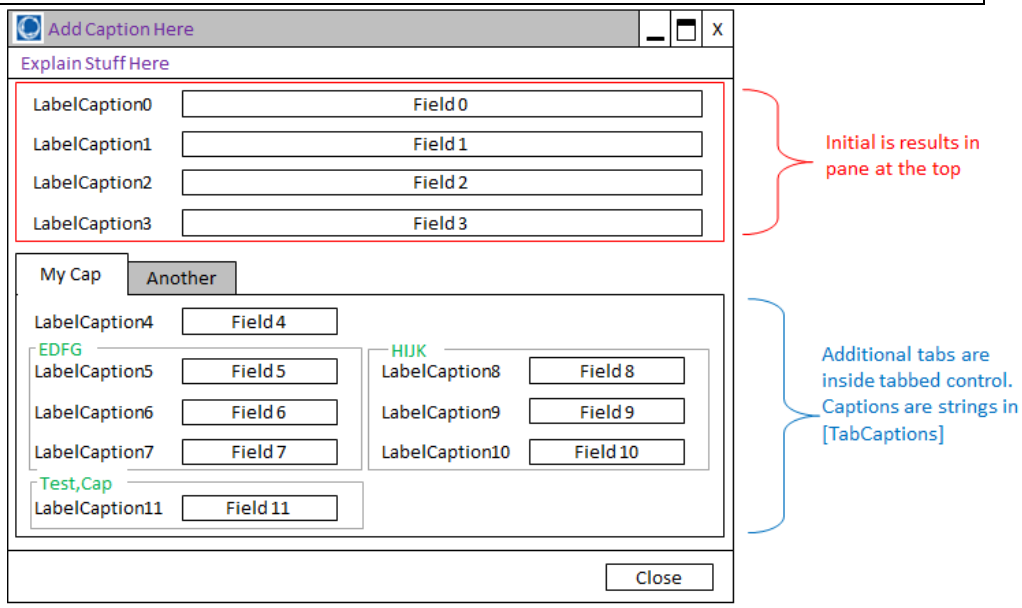
```
ObjectFieldsInputDialog("Branch 5 6 1", [Field0, … Field20],
     "Add Caption Here", "Explain Stuff Here", [],
     [4, 12], [My Cap,Another],
     [5,11,12,17,19], [EDFG,"Test,Cap","Heref",ABCD,""],
     [8,14,15,18,18], [HIJK,,LMNO,,XYZ]
     );
```

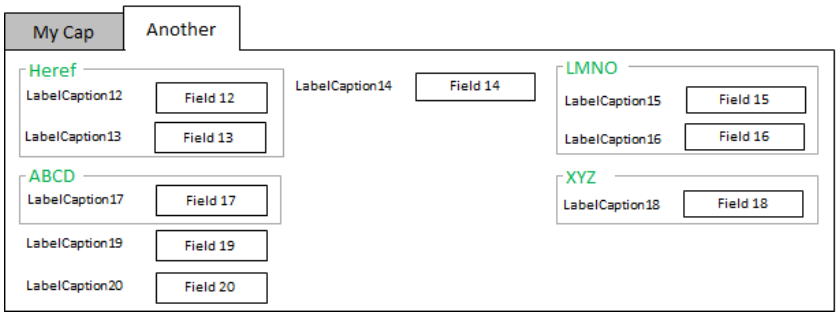
A few things of note in this example.

Column caption that goes with Index 14 is blank so no group box is drawn.

Column index 18 is listed twice in the ColBreaks which results in the empty column between Field 17 and 18.

RowBreak index 12 would seem to be unnecessary but provide the mechanism to add the caption "Heref".





OpenDataView("ObjectIDString", "DataGridIDString");

Use this action to open the Data View Dialog to a particular object using a particular set of customized string grid options. The string grid options determine the fields to show as well as whether to have any Tab, Row or Column breaks on the dialog in the same manner as is done for the ObjectFieldsInputDialog() script command.

"ObjectIDString" : The specific object for which to display fields. The format is the object type followed by the key fields used to identify the object. Examples: "Bus 234891", "Gen 16445 'A'", "Branch 1239 1234 'AB'".

"DataGridIDString" : Optional. If not specified dialog will open with the first customized grid. This is a reference to either a DataGrid or a UserDefinedDataGrid object. DataGrid objects store the customizations used on various case information displays in PowerWorld Simulator. A part of the customization for a DataGrid includes information about the Data View Layout (allows tab, row, and column breakers along with captions for tabs and group boxes). The format for this string is object type string DataGrid or UserDefinedDataGrid the key fields for that object (only a name for the DataGrid, and name followed by object type for UserDefinedDataGrid). Examples:

"DataGrid 'BranchRun'"
"DataGrid 'BranchEdit'"
"UserDefinedDataGrid 'My named grid' Bus"

Note: you may also simply enter a string showing the name of either the DataGrid or UserDefinedDataGrid. If you do this, then Simulator will first look for a DataGrid with that name. If a DataGrid is not found, then we will look for a UserDefinedDataGrid that matches the name specified and assumes the object type matches what is specified for the ObjectIDString.

Customer-Specific Actions

Available regardless of the mode

```
ISONEInterfaceLimitCalculation       ("filename", variablename);
```

ISONEInterfaceLimitCalculation("filename", variablename);

This script command has been developed specifically for ISO-NE to determine the limit of a specified interface based on a set of logical conditions being met on the given state of the power system.

"filename"	:	Name of the file containing all input data. This is a text-based format with the description of the contents given below.
variablename	:	Interface floating point field that will contain the calculated limit for each studied interface.

The file containing the input data contains specific sections to properly calculate an interface limit. Each section is identified by a keyword. Any sections with keywords that are not recognized will be ignored. The data for each section is enclosed in curly braces, { }. Curly braces must be on their own lines. Comments can be added using the double slash, //, notation at the start of a line. Only entire lines can be commented. The contents of an example file are shown below:

```
// Put comments here
Interface
{
MyInterface
}
ModelExpressions
{
Branch1_Online
GeneratorA_Online
GeneratorB_Offline
LoadGroup_Online
}
MatrixM
{
-0.5,0.5,  0.5,1.5, -9,9, -9,9
-0.5,0.5, -0.5,0.5, -9,9, -9,9
}
VectorL
{
123
456
}
MatrixA
{
0,0,350,100
0,0,120,80
}
VectorB
{
700
400
}
```

The section keywords and data contents are described below :

Interface	:	Name of the interface for which the limit is being calculated. Only a single interface limit can be calculated per input file.

ModelExpressions	:	List of Model Expressions that correspond to the columns of MatrixM and MatrixA. These Model Expressions must be specified by name as the key field for identifying them within the power flow case. There must be as many Model Expressions defined as there are columns of MatrixM and MatrixA (m).
MatrixM	:	This is an nxm matrix of floating point values. Each entry in the matrix contains two values that represent the minimum and maximum range of a power system value. It is possible that there will be no limit in either the minimum or maximum direction and -INF or INF can be used to indicate this instead of a floating point value.
VectorL	:	This is an nx1 vector of floating point values. This vector must be the same size as the number of rows in MatrixM.
MatrixA	:	This is an nxm matrix of floating point values. This matrix must be the same size as MatrixM. This matrix is optional and does not have to exist in the input file. If not specified, it will be assumed that all entries are 0 and it is the same size as MatrixM. Previous versions of the script command named this MatrixL1. To maintain older functionality, Simulator will look for either MatrixA or MatrixL1.
VectorB	:	This is an nx1 vector of floating point values. Each entry is the upper bound limit for the corresponding rows of MatrixM. To maintain older functionality, it is not required that this vector be included in the input data. If it is not included, the calculations will be done without imposing these limits.

Implementation of the limit calculation is determined by the following:

Each column of Matrix M corresponds to a Model Expression. All defined Model Expressions will be evaluated in the present power system state. Each row, *n*, of M is evaluated for the logical conditions defined for each cell entry. The logical condition for each cell entry *m* is: $min_m \leq Model\ Expression_m \leq max_m$. There is only one row, call this *p*, of M where all logical conditions are met.
The interface limit is then calculated based on row *p* using the following equation:

$$Limit_{prelim} = VectorL_p + \sum_{j=1}^{m} MatrixA_{p,j} \cdot V_j$$

where V_j is the value of $Model\ Expression_j$.
If VectorB has been specified,

$$Limit = \mathrm{Min}\{Limit_{prelim}, VectorB_p\}$$

otherwise,

$$Limit = Limit_{prelim}$$

If there is no row *p* where all logical conditions are met, the *Limit* will be set to 99999 if the input is specified using MatrixA. If the older format that uses MatrixL1 is specified, the *Limit* will be set to 0. To access the calculated *Limit* value, the field that is specified by the input variablename parameter is set to the *Limit*.
During the calculation, the *Selected* field will be set to *YES* for all Model Expressions that are used in the calculation and to *NO* for all Model Expressions that are not used as part of the calculation. The *Selected* field for the interface for which the limit is determined will be set to *YES*, and the *Selected* field will be set to *NO* for all other interfaces.

Edit Mode Actions

Case Related Actions

The following script commands are available while in Edit mode

```
AppendCase                    ("filename", OpenFileType, [StarBus, EstimateVoltages]);
AppendCase                    ("filename", OpenFileType, [MSLine, VarLimDead, PostCTGAGC,
                              EstimateVoltages]);
Combine                       ([elementA], [elementB]);
DeleteExternalSystem;
Equivalence;
InterfaceFlatten              ("InterfaceName");
InterfaceModifyIsolatedElements(filter);
InterfacesAutoInsert          (Type, DeleteExisting, UseFilters, Prefix, Limits);
MergeBuses                    ([element], Filter);
MergeLineTerminals            (Filter);
MergeMSLineSections           (Filter);
Move                          ([elementA], [destination parameter], HowMuch);
ReassignIDs                   (objecttype, field, filter, UseRight);
Remove3WXformerContainer      (filter);
Renumber3WXFormerStarBuses    ("filename", Delimiter);
RenumberAreas                 (NumCI);
RenumberBuses                 (NumCI);
RenumberMSLineDummyBuses      ("filename", Delimiter);
RenumberSubs                  (NumCI);
RenumberZones                 (NumCI);
SaveExternalSystem            ("filename", Savefiletype, withties);
SplitBus                      ([element], NewBusNumber, InsertBusTieLine, LineOpen,
                              BranchDeviceType);
TapTransmissionLine           ([element], PosAlongLine, NewBusNumber, ShuntModel, TreatAsMSLine,
                              UpdateOnelines);
```

AppendCase("filename", OpenFileType, [StarBus, EstimateVoltages]);

AppendCase("filename", OpenFileType, [MSLine, VarLimDead, PostCTGAGC, EstimateVoltages]);

Use this action to append a case to the currently open case. The optional parameters depend on the type of file being appended.

"Filename"	:	File name of the case to be appended
OpenFileType	:	PWB – case file is a PowerWorld Binary file GE – case file is a GE .epc file. GE*xx* where *xx* is the appropriate EPC file version number can also be used. PTI – case file is a PTI .raw file. PTI*xx* where *xx* is the appropriate RAW file version number can also be used. CF – case file is an IEEE common data format file
StarBus	:	Optional parameter – default is NEAR Only used for PTI RAW format, with the following options: NEAR – star buses are numbered starting after the near bus number MAX – star buses are numbered starting after the maximum bus number Value – star bus numbering will start at specified value
MSLine	:	Optional parameter – default is MAINTAIN Only used for the GE EPC format, with the following options: MAINTAIN – maintain multisection lines EQUIVALENCE – equivalence multisection lines
VarLimDead	:	Optional parameter – default is 2.0 Only used for the GE EPC format NUMBER – set the GE var limit deadband to this value
PostCTGACG	:	Optional parameter – default is NO Only used for the GE EPC format. Set to YES to populate the generator field Post-CTG Prevent Response based on the EPC file's generator base load flag.

EstimateVoltages : Optional parameter – default is YES
Used with either GE EPC or PTI RAW format with the following options:
YES – voltages and angles are estimated for new buses that are created when appending data to a case. Angle smoothing is done across new lines that are created when appending data to a case. These operations might be necessary if appending data that contains voltages that are not consistent to the case into which it is being appended or contains no voltages at all. This is the default.
NO – no voltage and angle estimates are done and no angle smoothing is done. This might be necessary when appending large sections of a case, i.e. such as a new island, or providing voltages that are already good estimates in the appended data.

Combine([elementA], [elementB]);

NOTE: THIS ACTION IS ONLY AVAILABLE FOR GENERATORS
Use this action to combine two generators, two loads, or two transmission line. Note that elementA and elementB must be of the same object type. You cannot combine a BRANCH and a LOAD.

[elementA] : The object that should be moved. See the format for [elementA] in the Move script command for information on the formatting of this string.

[elementB] : The object that element A should be combined with. Same format as for elementA.

DeleteExternalSystem;

This action will delete part of the power system. It will delete those buses whose property Equiv must is set true.

Equivalence;

This action will equivalence a power system. All options regarding equivalencing are handled by the Equiv_Options objecttype. Use the SetData action, or a DATA section to set these options prior to using the Equivalence() action. Also, remember that the property Equiv must be set true for each bus that you want to equivalence.

InterfaceFlatten("InterfaceName");

Interfaces can contain elements that are other interfaces. The "flattening" process will permanently modify an interface to contain the elements that the other interface contains and remove the element that is the other interface.

"InterfaceName" : Name of the interface to modify.

InterfaceModifyIsolatedElements(filter);

(Added in the September 30, 2021 patch of Simulator 22)
In interfaces with many BRANCH OPEN actions it is possible for some of the open actions to disconnect portions of the system that contain other open actions and other devices. This can be seen in the illustration below. **All** lines on the diagram represent interface BRANCH OPEN elements. However, the three green lines in the center are completely disconnected from the system by the other BRANCH OPEN interface elements (the ones in pink). The BRANCH OPEN elements also disconnect load. This situation is very hard to deal with numerically when calculating sensitivities. We address this by removing the BRANCH OPEN elements that are contained within the disconnected portion of the system. We also add OPEN actions

filter : Optional parameter – default is to apply command to all interfaces
Filter that specifies which interfaces to modify. See Using Filters in Script Commands section for more information on specifying the filter.

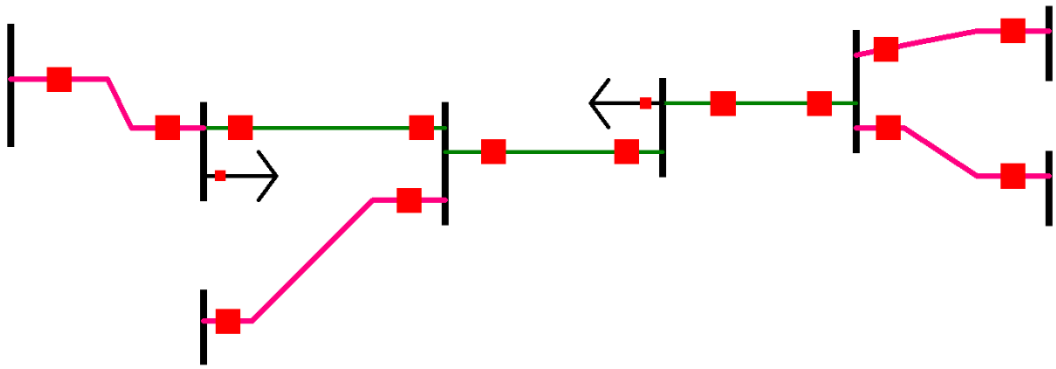


Figure 1: Interface BRANCH OPEN actions with disconnected subset

InterfacesAutoInsert(Type, DeleteExisting, UseFilters, "Prefix", Limits);

Use this action to auto-insert interfaces

Type	:	AREA – insert area-to-area tieline interfaces. ZONE – insert zone-to-zone tieline interfaces.
DeleteExsiting	:	YES – to delete existing interfaces; NO – to leave existing interfaces alone.
UseFilters	:	YES – to user Area/Zone Filters; NO – to insert for entire case.
"Prefix"	:	Enter a string which will be a prefix on the interface names.
Limits	:	ZEROS – to make all limits zero. AUTO – limits will be set to the sum of the branch limits. [lima, limb, limc, limd, …] – Enter 8 limits enclosed in brackets, separated by commas. This will set the limits as specified.

MergeBuses([element], Filter);

Use this action to merge buses

Element	:	The bus object that will be created when the buses meeting the Filter are merged.
Filter	:	Optional parameter – default is to merge all buses into a single bus
	AREAZONE	: Only buses that meet the area/zone/owner filters will be merged
	SELECTED	: Merge buses with Selected field = YES
	"FilterName"	: Only buses that meet the specified filter will be merged. See Using Filters in Script Commands section for more information on specifying the filtername.

MergeLineTerminals(Filter);

Use this action to merge line terminals. This action can be used to remove a line by merging the terminal buses of that line into a single bus. The only parameter of the script command is a filter parameter, which must be populated with either the name of an advanced filter (with the name in quotation marks) or the text SELECTED (with no quotation marks). If an advanced filter is given, then Simulator will find all branches that meet the advanced filter definition and will individually merge the line terminals of each line one at a time.

Filter	:	Any multi-section lines meeting this filter will be merged.
	"filtername"	: See the Using Filters in Script Commands section for more information on specifying the filtername.
	SELECTED	: Merge objects with Selected field = YES

MergeMSLineSections(Filter);

Use this action to eliminate multi-section line records. If possible, the individual sections will be merged into a single line record between the from and to bus and the multi-section line record will be removed. If a multi-section line contains series capacitors or transformers, the multi-section line record will be retained.

Filter : Any multi-section lines meeting this filter will be merged.
- "filtername" : See the Using Filters in Script Commands section for more information on specifying the filtername.
- SELECTED : Merge objects with Selected field = YES

Move([elementA], [destination parameters], HowMuch);

Use this action to move a generator, load, transmission line, or switched shunt.

[elementA] : The object that should be moved. Must be one of the following formats:
[GEN busnum id], [GEN "name_nomkv" id],
[GEN "label"]
[LOAD busnum id] , [LOAD "name_nomkv" id],
[LOAD "label"],
[BRANCH busnum1 busnum2 ckt],
[BRANCH "name_kv1" "name_kv2" ckt],
[BRANCH "label"]
[SHUNT busnum id], [SHUNT "name_nomkv" id],
[SHUNT "label"],
[MULTISECTIONLINE busnum1 busnum2 ckt],
[MULTISECTIONLINE "name_kv1" "name_kv2" ckt],
[MULTISECTIONLINE "label"],
[3WXFORMER busnum1 busnum2 busnum3 ckt],
[3WXFORMER "name_kv1" "name_kv2" "name_kv3" ckt]
[3WXFORMER "label"]

[destination parameters]
: These parameters have different meanings depending on object type of the element. Must use bus numbers here:
- GEN : [busnum id]
- LOAD : [busnum id]
- BRANCH : [busnum1 busnum2 ckt]
- SHUNT : [busnum id]
- MULTISECTIONLINE : [busnum1 busnum2 ckt]
- 3WXFORMER : [busnum1 busnum2 busnum3 ckt]

HowMuch : The amount of the element to move. A value of 100 indicates that 100% should be moved. This parameter is only valid for generators and loads. It is ignored for lines and switched shunts.

ReassignIDs(objecttype, field, filter, UseRight);

Use this action to set the IDs of specified objects to the first two characters of a specified field.

Objecttype : The type of object for which to assign IDs. BRANCH, GEN, LOAD, and SHUNT are allowed.

field : The field that contains the IDs that will be assigned. Only the first two characters of the field will be assigned. Field is specified in format `variablenamelegacy:location` or `concisename`

Filter : (optional) Any objects meeting this filter will have their IDs reassigned. Blank is the default value:
- Blank : means all objects will be modified
- ALL : means all objects will be modified

SELECTED : means only branches whose Selected field = YES will be modified

AREAZONE : means only branches that meet the area/zone/owner filters will be modified

"FilterName" : means only objects that meet the specified filter will be modified. See the Using Filters in Script Commands section for more information on specifying the filtername.

UseRight : Optional parameter – default is NO
Set to YES or NO. If set to YES, the last two characters of the specified field will be assigned.

Remove3WXformerContainer(filter);

Use this action to delete the three-winding transformers matching the specified filter while leaving the internal two-winding transformers intact.

Filter : (optional) Any three-winding transformers meeting this filter will be deleted. Default is blank:

Blank : means all three-winding transformers will be deleted

ALL : means all three-winding transformers will be deleted

SELECTED : means only three-winding transformers whose Selected field = YES will be deleted

AREAZONE : means only three-winding transformers that meet the area/zone/owner filters will be deleted

"FilterName" : means only three-winding transformers that meet the specified filter will be deleted. See the Using Filters in Script Commands section for more information on specifying the filtername.

Renumber3WXFormerStarBuses("filename", Delimiter);

Use this action to renumber star buses based on user-specified values.

"filename" : The name of the file containing the renumbering

Delimiter : Optional parameter – default is BOTH
Set to COMMA, SPACE, or BOTH to indicate the delimiter to use to separate data in the file.

The file may be comma or space delimited. The contents of the file should be formatted using the format below.

```
primary bus, secondary bus, tertiary bus, circuit, new starbus number, new starbus
name
```

or

```
pribusname_nomkV, secbusname_nomkV, terbusname_nomkV, circuit ID, newstarbusnum,
newstarbusname
```

Either the bus number or the name_nominalkV identifier may be used to identify the buses. Each bus may be identified using either method even for the same transformer. Lines starting with two slashses (//) will be ignored. The next two lines are sample file contents using different methods to identify buses.

```
11037, 11038, 11199, "1", 11202, "Ki star"
"WESTWING_500.00" "WESTWNGW_230.00" "WESTWG 4_34.50" "2" 99823 "KI STAR 3"
```

RenumberAreas(NumCI);

Renumber Areas using the new number for the Area located in the Custom Integer field of the area.

NumCI : Custom Integer field containing the new numbers.

RenumberBuses(NumCI);

Renumber Buses using the new number for the bus located in the Custom Integer field of the bus.

NumCI : Custom Integer field containing the new numbers.

RenumberMSLineDummyBuses("filename", Delimiter);

Use this action to renumber dummy buses or a multisection line based on user-specified values.

"filename" : The name of the file containing the renumbering

Delimiter : Optional parameter – default is BOTH
Set to COMMA, SPACE, or BOTH to indicate the delimiter to use to separate data in the file.

The file may be comma or space delimited. Buses may be identified using bus numbers or using the BusName_NominalkV combination. The file format is below:

```
from bus, to bus, circuit   //identifiy multi-section line
dummybusnumber1, dummybusname1
dummybusnumber2, dummybusname2
```

where the dummy bus numbers and names give the numbers and names that will be assigned for the dummy buses of a multi-section line. An example of the file contents is below:

```
40039 , 40141, 1       // ALFALFA   230 N BONNVL  230 #1
  49997, "ALFN B11"
40062 , 40699, 2     // ASHE R1   500 MARION    500 #2
  49990, ASHMAR21
  49989, ASHMAR22
  49988, ASHMAR23
```

RenumberSubs(NumCI);

Renumber Substations using the new number for the substation located in the Custom Integer field of the substation.

NumCI : Custom Integer field containing the new numbers.

RenumberZones(NumCI);

Renumber Zones using the new number for the Zone located in the Custom Integer field of the zone.

NumCI : Custom Integer field containing the new numbers.

SaveExternalSystem("Filename", SaveFileType, WithTies);

This action will save part of the power system to a "filename". It will save only those buses whose property Equiv must is set true.

filename : The file name to save the information to.

SaveFileType : An optional parameter saying the format of the file to be saved. If none is specified, then PWB will be assumed. May be one of the following strings
PWB, PWB5-PWB20
PTI23-PTI34
GE14-GE21, CF, AUX

WithTies : An optional parameter. The user must specify the file type explicitly in order to use the WithTies parameter. Allows saving a transmission line that ties a bus marked with Equiv as false and one marked true. This must be a string which starts with the letter Y, otherwise NO will be assumed.

SplitBus([element], NewBusNumber, InsertBusTieLine, LineOpen, BranchDeviceType);

Use this action to split buses

Element : Enter the description of which bus to split by enclosing in brackets the word bus and an identifier. The format looks as follows:
[BUS num]
[BUS "name_nomkv"]
[BUS "buslabel"]

NewBusNumber : This is the number of the new bus to create

InsertBusTieLine : Optional parameter – default is YES
Set to YES or NO. YES will insert a low impedance tie line between the buses; NO will not.

LineOpen : Optional parameter – default is NO
Set to YES or NO. YES set the status of the inserted bus tie to OPEN. NO will set the status of the inserted bus tie to CLOSED.

BranchDeviceType : Optional parameter – default is "Line"
Specify the Branch Device Type of the branch inserted for the bus tie. Options are: "Line", "Transformer", "Breaker", "Disconnect", "ZBR", "Fuse", "Load Break Disconnect", and "Ground Disconnect".

TapTransmissionLine([element], PosAlongLine, NewBusNumber, ShuntModel, TreatAsMSLine, UpdateOnelines);

Use this action to tap a transmission line by adding in a new bus and splitting the line in two.

Element : A description of the branch being tapped. The first bus listed will be treated as the nearbus that is used as the reference for the PosAlongLine. If the branch is identified by label, the from bus will be used as the reference for the PosAlongLine.
Enclose description in brackets:
[BRANCH busnum1 busnum2 ckt]
[BRANCH "name_kv1" "name_kv2" ckt]
[BRANCH "buslabel1" "buslabel2" ckt]
[BRANCH "label"]

PosAlongLine : The percent distance along the branch at which the line will be tapped.

NewBusNumber : The number of the new bus created at the tap point.

ShuntModel : Optional parameter – default is CAPACITANCE
How should the shunt charging capacitance values be handled for the split lines:
LINESHUNTS : Line shunts will be created (keeps exact power flow model).
CAPACITANCE : Convert shunt values capacitance in the PI model.

TreatAsMSLine : Optional parameter – default is NO
Set to YES or NO. If set to YES, the two newly created lines will be made part of a mulit-section line.

UpdateOnelines : Optional parameter – default is NO
Set to YES or NO. If set to YES, the original display transmission line object will be replaced with the tapped display transmission line and display bus objects on all open oneline diagrams.

Run Mode Actions

The following script commands are available while in Run Mode.

```
Animate                          (DoAnimate);
CalculatePTDF                    ([transactor seller], [transactor buyer], LinearMethod);
CalculatePTDFMultipleDirections(StoreValuesForBranches,StoreValuesForInterfaces,LinearMethod);
CalculateLODF                    ([BRANCH nearbusnum farbusnum ckt], LinearMethod,
                                 PostClosureLCDF);
CalculateLODFMatrix              (WhichOnes, filterProcess, filterMonitor, MonitorOnlyClosed,
                                 LinearMethod, filterMonitorInterface, PostClosureLCDF);
CalculateLODFScreening           (filterProcess, filterMonitor, IncludePhaseShifters,
                                 IncludeOpenLines, UseLODFThreshold, LODFThreshold,
                                 UseOverloadThreshold, OverloadLow, OverloadHigh, DoSaveFile,
                                 FileLocation, CustomFieldHighLODF, CustomFieldHighLODFLine,
                                 CustomFieldHighOverload, CustomFieldHighOverloadLine, DoUseCTGName
                                 CustomFieldOrigCTGName);
CalculateLODFAdvanced            (IncludePhaseShifters, FileType, MaxColumns, MinLODF,
                                 NumberFormat, DecimalPoints, OnlyIncludingLinesIncreasing,
                                 "FileName", IncludeIslandingCTG);
CalculateShiftFactors            ([flowelement], direction, [transactor], LinearMethod,
                                 SetOutOfServiceBuses, filter, AbortOnError);
CalculateShiftFactorsMultipleElement (TypeElement, WhichElement, direction, [transactor],
                                 LinearMethod);
CalculateVoltSense               ([BUS num]);
CalculateFlowSense               ([flowelement], FlowType);
CalculateLossSense               (FunctionType, AreaSALossReference,IslandLossReference);
CalculateVoltToTransferSense     ([transactor seller], [transactor buyer], TransferType,
                                 TurnOffAVR);
CalculateVoltSelfSense           (filter);
LineLoadingReplicatorCalculate([Flow Element], [Injection Group], AGCOnly, DesiredFlow,
                                 Implement, LinearMethod, UseLoadMinMax, MaxMultiplier,
                                 MinMultiplier);
LineLoadingReplicatorImplement;
DeleteState                      (WhichState, StateName);
RestoreState                     (WhichState, StateName);
SetInterfaceLimitToMonitoredElementLimitSum(filter);
SetSensitivitiesAtOutOfServiceToClosest(filter);
StoreState                       (StateName);
ZeroOutMismatches;
```

Animate(DoAnimate);

Use this action to animate all the open oneline diagrams.

DoAnimate : Set to YES or NO. YES means to start the animation of the open oneline diagrams, while NO means that the animation will be paused.

CalculatePTDF([transactor seller], [transactor buyer], LinearMethod);

Use this action to calculate the PTDF values between a seller and a buyer. You may optionally specify the linear calculation method. Note that the buyer and seller must not be same thing. If no Linear Method is specified, Lossless DC will be used.

[transactor seller] : The seller (or source) of power. There are six possible settings:
- [AREA num], [AREA "name"], [AREA "label"]
- [ZONE num], [ZONE "name"], [ZONE "label"]
- [SUPERAREA "name"], [SUPERAREA "label"]
- [INJECTIONGROUP "name"], [INJECTIONGROUP "label"]
- [BUS num], [BUS "name_nomkv"], [BUS "label"]
- [SLACK]

[transactor buyer] : The buyer (or sink) of power. There are six possible settings which are the same as for the seller.

LinearMethod : The linear method to be used for the PTDF calculation. The options are:
- AC : for calculation including losses
- DC : for lossless DC
- DCPS : for lossless DC that takes into account phase shifter operation

CalculatePTDFMultipleDirections(StoreForBranches, StoreForInterfaces, LinearMethod);

Use this action to calculate the PTDF values between all the directions specified in the case. You may optionally specify the linear calculation method. If no Linear Method is specified, Lossless DC will be used.

StoreForBranches : Specify YES to store the values calculated for each branch.
StoreForInterfaces : Specify YES to store the values calculated for each interface.
LinearMethod : the linear method to be used for the PTDF calculation. The options are:
- AC : for calculation including losses.
- DC : for lossless DC.
- DCPS : for lossless DC that takes into account phase shifter operation.

CalculateLODF([BRANCH nearbusnum farbusnum ckt], LinearMethod, PostClosureLCDF);

Use this action to calculate the Line Outage Distribution Factors (or the Line Closure Distribution Factors) for a particular branch. If the branch is presently closed, then the LODF values will be calculated, otherwise the LCDF values will be calculated. You may optionally specify the linear calculation method as well. If no Linear Method is specified, Lossless DC will be used.

[BRANCH nearbusnum farbusnum ckt]
: the branch whose status is being changed. Can also use strings
- [BRANCH "nearbusname_kv" "farbusname_kv" ckt]
- [BRANCH "nearbuslabel" "farbuslabel" ckt]
- [BRANCH "label"]

LinearMethod : The linear method to be used for the LODF calculation. The options are:
- DC : for lossless DC.
- DCPS : for lossless DC that takes into account phase shifter operation.

Note: AC is NOT an option for the LODF calculation.

PostClosureLCDF : Optional parameter – default is YES
Set to YES to calculate any line closure sensitivies relative to post-closure flow on the line being closed. This is known as the LCDF value.
Set to NO to calculate any line closure sensitivities based on calculating the flow on the line being closed from pre-closure voltages and angles. This is known at the MLCDF value.

CalculateLODFMatrix(WhichOnes, filterProcess, filterMonitor, MonitorOnlyClosed, LinearMethod, filterMonitorInterface, PostClosureLCDF);

Use this action to calculate the Line Outage Distribution Factors (or the Line Closure Distribution Factors) for a particular branch. If the branch is presently closed, then the LODF values will be calculated, otherwise the LCDF values will be calculated. You may optionally specify the linear calculation method as well. If no Linear Method is specified, Lossless DC will be used.

WhichOnes : Specify the type of sensitivities to be calculated.
- OUTAGES : Outage sensitivities will be calculated for those branches meeting the filterProcess.
- CLOSURES : Closure sensitivities will be calculated for those branches meeting the filterProcess.

filterProcess : Specify a filter for the branches for which the outages or closures will be implemented.
- ALL : All AC transmission lines.
- SELECTED : Only those branches whose Selected field is YES.
- AREAZONE : Only those branches meeting the area/zone filter.
- "filtername" : See the Using Filters in Script Commands section for more information on specifying the filtername.

filterMonitor : Specify a filter for the branches for which the impact of the outages or closures will be determined.

- ALL : All AC transmission lines.
- SELECTED : Only those branches whose Selected field is YES.
- AREAZONE : Only those branches meeting the area/zone filter.
- "filtername" : See the Using Filters in Script Commands section for more information on specifying the filtername.
- SAME : Same as set of branches to process as specified by filterProcess.

MonitorOnlyClosed : Set to YES to monitor only those branches that are closed. Set to NO to monitor branches regardless of their status.

LinearMethod : Optional parameter – default is DC
The linear method to be used for the LODF calculation.

- DC : for lossless DC.
- DCPS : for lossless DC that takes into account phase shifter operation.

Note: AC is NOT an option for the LODF calculation.

filterMonitorInterface : Optional parameter – default is to not monitor interfaces
Specify a filter for the interfaces for which the impact of the outages or closures will be determined. Using this option will add the individual lines in the interface to the list of lines to monitor.

- ALL : All interfaces.
- SELECTED : Only those interfaces whose Selected field is YES.
- AREAZONE : Only those interfaces meeting the area/zone filter.
- "filtername" : See the Using Filters in Script Commands section for more information on specifying the filtername.

PostClosureLCDF : Optional parameter – default is YES
Set to YES to calculate any line closure sensitivies relative to post-closure flow on the line being closed. This is known as the LCDF value.
Set to NO to calculate any line closure sensitivities based on calculating the flow on the line being closed from pre-closure voltages and angles. This is known at the MLCDF value.

CalculateLODFScreening(filterProcess, filterMonitor, IncludePhaseShifters, IncludeOpenLines, UseLODFThreshold, LODFThreshold, UseOverloadThreshold, OverloadLow, OverloadHigh, DoSaveFile, FileLocation, CustomFieldHighLODF, CustomFieldHighLODFLine, CustomFieldHighOverload, CustomFieldHighOverloadLine, DoUseCTGName, CustomFieldOrigCTGName);

Use this action to do the LODF Screening calculation. This calculation uses LODF/LCDF factors to determine how significant a branch open/close action will be on monitored lines. The significance of the action can be determined by LODF/LCDF magnitude or line loading on monitored lines. Significant single contingency actions can then be combined to form pairs of contingency actions that will be used to create new contingencies that can be saved to an auxiliary file.

filterProcess : Specify a filter for the branches for which the outage or closure impact will be determined.

- ALL : All AC transmission lines.
- AREAZONE : Only those branches meeting the area/zone filter.
- CTG : Only those branches included in any currently defined contingency.
- LIMITMONITOR : Only those branches meeting the Limit Monitoring Settings.
- SELECTED : Only those branches whose Selected field is YES.
- "filtername" : See the Using Filters in Script Commands section for more information on specifying the filtername.

filterMonitor : Specify a filter for the branches on which the impact of the outages or closures will be determined.

- ALL : All AC transmission lines.
- AREAZONE : Only those branches meeting the area/zone filter.
- LIMITMONITOR : Only those branches meeting the Limit Monitoring Settings.
- SAME : Same as branches to process specified by filterProcess.
- SELECTED : Only those branches whose Selected field is YES.
- "filtername" : See the Using Filters in Script Commands section for more information on specifying the filtername.

IncludePhaseShifters : Set to YES to calculate the LODF/LCDF values assuming that phase shifters are allowed to operate and will see no impact due to an outage or closure. Set to NO to not enforce the flow on phase shifters.

IncludeOpenLines : Set to NO to monitor only those branches that are closed. Set to YES to monitor branches regardless of their status.

UseLODFThreshold : Set to YES to screen outages/closures by LODF/LCDF magnitude. Set to NO to not screen by LODF/LCDF magnitudes.

LODFThreshold : Threshold above which LODF/LCDF magnitudes are considered significant.

UseOverloadThreshold : Set to YES to screen outages/closures by monitored branch loading. Set to NO to not screen by branch loading.

OverloadLow : Threshold above which a monitored branch loading is considered significant. This value should be entered as a percent.

OverloadHigh : Threshold below which a monitored branch loading is considered significant. This value should be entered as a percent.

DoSaveFile : Set to YES to save an auxiliary file of new contingencies created by joining pairs of significant single outage/closure actions. Set to NO to not save the file.

FileLocation : Specify a directory path where the auxiliary file containing new contingencies will be saved. The filename will be determined by Simulator.

CustomFieldHighLODF : Optional parameter – default is 0
Integer indicating which Custom Floating Point field for a processed branch will store the highest magnitude LODF/LCDF determined for any monitored branch.

CustomFieldHighLODFLine : Optional parameter – default is 0
Integer indicating which Custom String field for a processed branch will store the identifier for the monitored branch that has the highest magnitude LODF/LCDF.

CustomFieldHighOverload : Optional parameter – default is 0
Integer indicating which Custom Floating Point field for a processed branch will store the highest overload determined for any monitored branch.

CustomFieldHighOverloadLine : Optional parameter – default is 0
Integer indicating which Custom String field for a processed branch will store the identifier for the monitored branch that has the highest overload.

DoUseCTGName : Optional parameter – default is NO
Set to YES to use the available active contingency names available in the CTG Tool. If a contingency in the CTG Tool has the same branch as the only contingency element it will use the contingency name in the tool to create the new CTG Label for the new contingency combination of branches. Set to NO to only use the branch info to create the new CTG Label for the combination of branches.

CustomFieldOrigCTGName
: Optional parameter – default is 0
Integer indicating which Custom String field for a processed branch will store the name of the contingency from which the branch originated.

CalculateLODFAdvanced(IncludePhaseShifters, FileType, MaxColumns, MinLODF, NumberFormat, DecimalPoints, OnlyIncludingLinesIncreasing, "FileName", IncludeIslandingCTG);

Use this action to to mimic what is done on the Advanced LODF Calculation dialog in the GUI.

IncludePhaseShifters : Set to YES to calculate the LODF/LCDF values assuming that phase shifters are allowed to operate and will see no impact due to an outage or closure. Set to NO to not enforce the flow on phase shifters.

FileType : Either PROMOD or MATRIX. For PROMOD, save only "Monitored Branch, Contingency" pairs for PROMOD; and for MATRIX save Matrix as comma-delimited text file.

MaxColumns : Maximum number of columns per text file.

MinLODF : Only Save pairs with an LODF whose absolute value is greater than this minimum.

NumberFormat : LODF Number format, either, EXPONENTIAL or DECIMAL

DecimalPoints : Fixed Decimals Points.

OnlyIncludingLinesIncreasing: Only include monitored branches whose MW flow increases.

"FileName" : The name of the text file to write.

IncludeIslandingCTG : Optional parameter – default is YES
LODF values cannot be calculated for contingencies that will cause a new island to be created. When including these contingencies in the results, the LODF will be reported as a very large number to indicate that these values were not actually calculated. Set to NO to completely omit these contingencies from the results.

CalculateShiftFactors([flow element], direction, [transactor], LinearMethod, SetOutOfServiceBuses, filter, AbortOnError);

In Version 21 and earlier this script command was called **CalculateTLR**. Simulator 21 patches after January 20, 2021 will handle reading either the **CalculateShiftFactors** or **CalculateTLR**.

Use this action to calculate the Shift Factor Sensitivity values a particular flow element (transmission line or interface). You also specify one end of the potential transfer direction. You may optionally specify the linear calculation method. If no Linear Method is specified, Lossless DC will be used.

[flow element] : This is the flow element we are interested in. Choices are:
[INTERFACE "name"]
[INTERFACE "label"]
[BRANCH nearbusnum farbusnum ckt]
[BRANCH "nearbusname_kv" "farbusname_kv" ckt]
[BRANCH "nearbuslabel" "farbuslabel" ckt]
[BRANCH "label"]

direction : The type of the transactor. Either BUYER or SELLER.

[transactor] : The transactor of power. There are six possible settings:
[AREA num], [AREA "name"], [AREA "label"]
[ZONE num], [ZONE "name"], [ZONE "label"]
[SUPERAREA "name"], [SUPERAREA "label"]
[INJECTIONGROUP "name"], [INJECTIONGROUP "label"]
[BUS num], [BUS "name_nomkv"], [BUS "label"]
[SLACK]

LinearMethod : Optional parameter – default is DC
The linear method to be used for the calculation. The options are:
AC : for calculation including losses
DC : for lossless DC
DCPS : for lossless DC that takes into account phase shifter operation

SetOutOfServiceBuses : Optional parameter – default is NO
Set to YES or NO. If YES then set the sensitivities for out-of-service buses equal to the sensitivity to the value at the closest in-service bus. The "distance" to the in-service buses will be measured by the number of nodes. If an out-of-service bus is equally close to a set of buses, then the average of that set of buses will be used.

filter : Optional parameter – default is to include all buses
The filter that will determine the buses for which sensitivities will be set when SetOutOfServiceBuses = YES. In addition to meeting the filter, only out-of-service buses will be included.
blank : All buses
AREAZONE : Only those buses meeting the area/zone filter
SELECTED : Only those buses whose Selected field is YES
"filtername" : See the Using Filters in Script Commands section for more information on specifying the filtername.

AbortOnError : Optional parameter – default is YES
Set to YES or NO. If YES, the script command will fail and the auxiliary file containing the script command will terminate without processing the remainder of the file. If NO, an error message is printed to the message log but the command is not treated as failing and the remainder of the auxiliar file containing the command will be processed.

CalculateShiftFactorsMultipleElement(TypeElement,WhichElement,direction,[transactor],LinearMethod);

In Version 21 and earlier this script command was called **CalculateTLRMultipleElement**. Simulator 21 patches after January 20, 2021 will handle reading either the **CalculateShiftFactorsMultipleElement** or **CalculateTLRMultipleElement**.

Use this action to calculate the Shift Factor Sensitivity values a multiple elements similar to as is done on the TLR multiple elements dialog. You also specify one end of the potential transfer direction. You may optionally specify the linear calculation method. If no Linear Method is specified, Lossless DC will be used.

TypeElement : May be either INTERFACE, BRANCH, or BOTH

WhichElement : There are three choices which represent which elements of the TypeElement specified will have TLR calculations performed.
SELECTED : Only branches or interfaces with their Selected Field = YES will be used.
OVERLOAD : Only branches that are presently overloaded using their normal ratings will be used
CTGOVERLOAD : You must have first run the contingency analysis. A branch or interface is included in the calculation if it has been overloaded during at least one contingency.

Direction : the type of the transactor. Either BUYER or SELLER.

[transactor buyer] : the transactor of power. There are six possible settings.
[AREA num], [AREA "name"], [AREA "label"]
[ZONE num], [ZONE "name"], [ZONE "label"]
[SUPERAREA "name"], [SUPERAREA "label"]
[INJECTIONGROUP "name"], [INJECTIONGROUP "label"]
[BUS num], [BUS "name_nomkv"], [BUS "label"]
[SLACK]

LinearMethod : The linear method to be used for the calculation. The options are:
AC : for calculation including losses.
DC : for lossless DC.
DCPS : for lossless DC that takes into account phase shifter operation.

CalculateVoltSense([BUS num]);

This calculates the sensitivity of a particular buses voltage to real and reactive power injections at all buses in the system. (Note: this assumes that the power is injected at a given bus and taken out at the slack bus).

[BUS num] : the bus to calculate sensitivities for.

CalculateFlowSense([flow element], FlowType);

This calculates the sensitivity of the MW, MVAR, or MVA flow of a line or interface to an real and reactive power injections at all buses in the system. (Note: this assumes that the power is injected at a given bus and taken out at the slack bus).

[flow element] : This is the flow element we are interested in. Choices are:
[INTERFACE "name"]
[INTERFACE "label"]
[BRANCH busnum1bus num2 ckt]
[BRANCH "name_kv1" "name_kv2" ckt]
[BRANCH "buslabel1" "buslabel2" ckt]
[BRANCH "label"]

FlowType : The type of flow to calculate this for. Either MW, MVAR, or MVA.

CalculateLossSense(FunctionType,AreaSALossReference,IslandLossReference);

This calculates the loss sensitivity at each bus for an injection of power at the bus. The parameter FunctionType determines which losses are referenced.

FunctionType : This is the losses for which sensitivities are calculated.
NONE : all loss sensitivities will be set to zero
ISLAND : all loss sensitivities are referenced to the total loss in the island
AREA : For each bus it calculates how the losses in the bus' area will change (Note: this means that sensitivities at buses in two different areas cannot be directly compared because they are referenced to different losses)
AREASA : same as Each Area, but if a Super Area exists it will use this instead (Note: this means that sensitivities at buses in two different areas cannot be directly compared because they are referenced to different losses)
SELECTED : Calculates how the losses in the areas selected on the Loss Sensitivity Form will change

(Following parameters added in the July 14, 2022 patch for Simulator 22)

AreaSALossReference : Optional parameter - default is NO
This parameter will only be used if the FunctionType is AREA or AREASA. This option specifies whether or not the Cost of Energy, Losses, and Congestion Reference for each area or super area that is used for OPF calculations will be used in this calculation.

IslandLossReference : Optional parameter - default is EXISTING
This parameter specifies the loss reference that will be used when the FunctionType is ISLAND.
- EXISTING : Use existing loss sensitivities
- LOADS : MW value of all loads within an island will be used for weighting in the loss reference calculation
- "InjGroupName" : Name of injection group where the participation factor of each participant will be used for weighting in the loss reference calculation

CalculateVoltToTransferSense([transactor seller], [transactor buyer], TransferType, TurnOffAVR);

This calculates the sensitivity of bus voltage to a real or reactive power transfer between a seller and a buyer. The sensitivity is calculated for all buses in the system.

[transactor seller] : This is the seller (or source) of power. There are six possible settings:
[AREA num], [AREA "name"], [AREA "label"]
[ZONE num], [ZONE "name"], [ZONE "label"]
[SUPERAREA "name"], [SUPERAREA "label"]
[INJECTIONGROUP "name"], [INJECTIONGROUP "label"]
[BUS num], [BUS "name_nomkv"], [BUS "label"]
[SLACK]

[transactor buyer] : This is the buyer (or sink) of power. There are six possible settings, which are the same as for the seller.

TransferType : The type of power transfer. The options are:
- P : real power transfer
- Q : reactive power transfer
- PQ : both real and reactive power transfer. (Note: Real and reactive power transfers are calculated independently, but both are calculated.)

TurnOffAVR : Set to YES or NO. Set to YES to turn off AVR control for generators participating in the transfer. Set to NO to leave the AVR control unchanged for generators participating in the transfer.

CalculateVoltSelfSense(filter);

This calculates the sensitivity of a particular bus' voltage to real and reactive power injections at the same bus. (Note: This assumes that the power is injected at a given bus and taken out at the slack bus.)

filter : Optional parameter – default is to calculate sensitivities for all buses in the system
- AREAZONE : Only buses that meet the area/zone/owner filters will be included
- SELECTED : Only buses whose Selected field = YES will be included
- "FilterName" : Only buses that meet the specified filter will be included. See Using Filters in Script Commands section for more information on specifying the filtername.

LineLoadingReplicatorCalculate([Flow Element], [Injection Group], AGCOnly, DesiredFlow, Implement, LinearMethod, UseLoadMinMax, MaxMultiplier, MinMultiplier);

(Added in the August 31, 2021 patch of Simulator 22)

This command calculates an injection change list containing the injection changes needed to alter a line or interface flow to a desired value. The user supplies the element to manipulate the flow on, the injection group, and the desired line or interface flow. This command calculates the injection changes that will result in that line or interface flow. This command can optionally implement the changes, or the changes can be applied separately with the LineLoadingReplicatorImplement script command.

[Flow Element] : This is the flow element we are interested in. Choices are:
[INTERFACE "name"]
[INTERFACE "label"]
[BRANCH nearbusnum farbusnum ckt]
[BRANCH "nearbusname_kv" "farbusname_kv" ckt]
[BRANCH "nearbuslabel" "farbuslabel" ckt]
[BRANCH "label"]

[Injection Group] : Injection group containing elements that are available to move to implement the desired line or interface flow.

AGCOnly : YES indicates that only elements on AGC control in the injection group will move to implement the desired line flow. A value of NO indicates that all elements in the injection group can move as needed.

DesiredFlow : The new desired flow on the flow element.

Implement : YES indicates that the injection change should be implemented after it is calculated. NO indicates that the injection change will not be implemented. The list is stored in memory and may be implemented later with the LineLoadingReplicatorImplement command. This option lets you implement the change with one command if you have no need to check anything before implementing it.

LinearMethod : DC : for lossless DC.
DCPS : for lossless DC that takes into account phase shifter operation.

UseLoadMinMax : Optional parameter - default is YES
This option indicates that the maximum and minimum values specified with the load records should be used to limit the load movement while calculating the injection changes. A value of NO indicates that the values specified in the MaxMultiplier and MinMultiplier should be applied to set the load change limits. When set to NO the Max and Min multiplers are required.

MaxMultiplier : Optional parameter - default is 1.0
When UseLoadMinMax is set to NO this parameter is used as a scaling factor on the present load value to set the maximum limit that bounds the load's change.

MinMultiplier : Optional parameter - default is 1.0
When UseLoadMinMax is set to NO this parameter is used as a scaling factor on the present load value to set the minimum limit that bounds the load's change.

LineLoadingReplicatorImplement;

(Added in the August 31, 2021 patch of Simulator 22)
This command takes no parameters. It applies the changes in the injection change list calculated by the LineLoadingReplicatorCalculate command.

DeleteState(WhichState, StateName);

Apply this script command to delete a specified system state that is currently in memory. This script command will fail if the specified state has not been set. The following options are available for specifying which system state to delete:

WhichState : Optional parameter – default is USER. This determines which state to restore.
USER : Delete the user set system state that is set with the StoreState script command.

BEFOREFAILED : Before the power flow is solved, either through the GUI or the SolvePowerFlow script command, a system state will be stored in the event that the power flow solution fails. This pre-solution state is deleted with this option. If the power flow is successful in solving at any time after this pre-solution state is stored, this pre-solution state is removed automatically.

LASTSUCCESSFUL : If the power flow solution is successful when solving the power flow either through the GUI or the SolvePowerFlow script command, a system state will be stored with this successful solution. This post-solution state is deleted with this option.

StateName : Optional parameter – default is blank
This option is only used when WhichState = USER. This specifies a named system state that was stored using the StoreState script command. If no name is specified, the unnamed system state stored using StoreState will be deleted.

RestoreState(WhichState, StateName);

Apply this script command to restore a specified system state that is currently in memory. This script command will fail if the specified state has not been set. The following options are available for specifying which system state to restore:

WhichState : Optional parameter – default is USER. This determines which state to restore.

USER : Restore the user set system state that is set with the StoreState script command.

BEFOREFAILED : Before the power flow is solved, either through the GUI or the SolvePowerFlow script command, a system state will be stored in the event that the power flow solution fails. This pre-solution state is restored with this option. If the power flow is successful in solving at any time after this pre-solution state is stored, this pre-solution state is removed and cannot be restored.

LASTSUCCESSFUL : If the power flow solution is successful when solving the power flow either through the GUI or the SolvePowerFlow script command, a system state will be stored with this successful solution. This post-solution state is restored with this option.

StateName : Optional parameter – default is blank
This option is only used when WhichState = USER. This specifies a named system state that was stored using the StoreState script command. If no name is specified, the unnamed system state stored using StoreState will be restored.

SetInterfaceLimitToMonitoredElementLimitSum(filter);

This sets the limits of the interface to the sum of the limits of all branches within the interface. This only includes branches that are monitored and excludes any contingency branches. All limits A through H will be set.

Filter : This parameter is used to specify which interfaces have their limits set.

- ALL : all interfaces will be set
- SELECTED : only interfaces whose Selected field = YES will be set
- AREAZONE : only interfaces that meet the area/zone/owner filters will be set
- "FilterName" : only interfaces that meet the specified filter will be set. See the Using Filters in Script Commands section for more information on specifying the filtername.

SetSensitivitiesAtOutOfServiceToClosest(filter);

This will take the P Sensitivity and Q Sensitivity values calculated using the CalculateTLR, CalculateFlowSense, or CalculateVoltSense actions and then populate the respective values at out-of-service buses so that they are equal to the value at the closest in service bus. The "distance" to the in-service buses will be measured by the number of nodes. If an out-of-service bus is equally close to a set of buses, then the average of that set of buses will be used.

filter : Optional parameter – default is to include all buses
The filter that will determine the buses for which sensitivities will be set. In addition to meeting the filter, only out-of-service buses will be included.

- Blank : All buses
- AREAZONE : Only those buses meeting the area/zone filter
- SELECTED : Only those buses whose Selected field is YES
- "filtername" : See the Using Filters in Script Commands section for more information on specifying the filtername.

StoreState(StateName);

Apply this script command to store the current system state to memory. Use the RestoreState script command to restore the state. Multiple states can be stored by providing a unique name.

StateName : Optional parameter
Multiple states can be stored by providing a unique name for each state. If StateName is not specified, an unnamed state is stored.

ZeroOutMismatches(ObjectType);

With this script command, bus shunts or loads are changed at each bus that has a mismatch greater than the MVA convergence tolerance so that the mismatch at that bus is forced to zero.

ObjectType : Optional parameter – default is BUSSHUNT
Set to BUSSHUNT or LOAD to indicate how the mismatch should be adjusted. When set to BUSSHUNT the Bus Shunt fields at each bus where there is a mismatch are adjusted to eliminate the mismatches. When set to LOAD, a new load is added to a bus where there is a mismatch to eliminate the mismatches. The new loads are identified by a unique ID of Q1. If a load with this ID already exists, a unique ID is determined by incrementing the ID until a unique ID is found.

Power Flow Related Actions

```
ClearPowerFlowSolutionAidValues;
DiffCaseClearBase;
DiffCasKeyType                  (KeyType);
DiffCaseMode                    (diffmode);
DiffCaseSetAsBase;
DiffCaseShowPresentAndBase      (How);
DiffCaseRefresh;
DiffCaseWriteCompleteModel      ("filename", AppendFile, SaveAdded, SaveRemoved, SaveBoth,
                                KeyFields, "ExportFormat", UseAreaZone, UseDataMaintainer,
                                AssumeBaseMeet, IncludeClearPowerFlowSolutionAidValues);
DiffCaseWriteBothEPC            ("filename", GEFileType, UseAreaZone, BaseAreaZoneMeetFilter,
                                Append, "ExportFormat");
DiffCaseWriteNewEPC             ("filename", GEFileType, UseAreaZone, BaseAreaZoneMeetFilter,
                                Append);
DiffCaseWriteRemovedEPC         ("filename", GEFileType, UseAreaZone, BaseAreaZoneMeetFilter,
                                Append);
DoCTGAction                     ([contingency action]);
ResetToFlatStart                (FlatVoltagesAngles, ShuntsToMax, LTCsToMiddle, PSAnglesToMiddle);
SolvePowerFlow                  (SolMethod, "filename1", "filename2", CreateIfNotFound1,
                                CreateIfNotFound2);
```

ClearPowerFlowSolutionAidValues;

PowerWorld Simulator maintains several internal flags that keep track of which branches are closed or opened, as well as information to help estimate the generation change needed in a system after making changes to load or generation. PowerWorld uses this to help with various pre-processing steps in the power flow solution. This information is related to angle smoothing and generator MW estimation features of the power flow solution. Typically, this information is a great aid in getting successful power flow solutions, however in some circumstances you may be using an AUX file to edit information you know is good and would not want PowerWorld to modify the initial bus voltage and angle nor the generator MW outputs before a solution is attempted. To clear all this internally stored information so that PowerWorld does not do any of this, call the ClearPowerFlowSolutionAidValues script command.

DiffCaseClearBase;

Call this action to clear the base case for the difference flows abilities of Simulator.
In Version 21 and earlier this script command was called **DiffFlowClearBase**. Simulator 21 patches after January 20, 2021 will handle reading either the **DiffCaseClearBase** or **DiffFlowClearBase**.

DiffCaseKeyType(KeyType);

Use this action to change the key type that should be used when comparing fields when using the difference flows abilities of Simulator.

KeyType	:	String that starts with 'P' changes key field type to PRIMARY. String that starts with 'S' changes key field type to SECONDARY. String that starts with 'L' changes key field type to LABEL.

In Version 21 and earlier this script command was called **DiffFlowKeyType**. Simulator 21 patches after January 20, 2021 will handle reading either the **DiffCaseKeyType** or **DiffFlowKeyType**.

DiffCaseMode(diffmode);

Call this action to change the mode for the difference flows abilities of Simulator.

diffmode	:	String that starts with 'P' changes it to PRESENT. String that starts with 'B' changes it to BASE. String that starts with 'D' changes it to DIFFERENCE. String that starts with 'C' changes it to CHANGE.

In Version 21 and earlier this script command was called **DiffFlowMode**. Simulator 21 patches after January 20, 2021 will handle reading either the **DiffCaseMode** or **DiffFlowMode**.

DiffCaseSetAsBase;

Call this action to set the present case as the base case for the difference flows abilities of Simulator.
In Version 21 and earlier this script command was called **DiffFlowSetAsBase**. Simulator 21 patches after January 20, 2021 will handle reading either the **DiffCaseSetAsBase** or **DiffFlowSetAsBase**.

DiffCaseShowPresentAndBase(How);

Call this action with the parameter of either YES or NO to toggle the difference flows options "Show Present|Base in Difference and Change Mode".
In Version 21 and earlier this script command was called **DiffFlowShowPresentAndBase**. Simulator 21 patches after January 20, 2021 will handle reading either the **DiffCaseShowPresentAndBase** or **DiffFlowShowPresentAndBase**.

DiffCaseRefresh;

Call this action to refresh the linking between the base case and the present case. This should be used before saving data that identifies objects as being added or removed, especially if any topological differences have been made that affect the comparison.
In Version 21 and earlier this script command was called **DiffFlowRefresh**. Simulator 21 patches after January 20, 2021 will handle reading either the **DiffCaseRefresh** or **DiffFlowRefresh**.

DiffCaseWriteCompleteModel ("filename", AppendFile, SaveAdded, SaveRemoved, SaveBoth, KeyFields, "ExportFormat", UseAreaZone, UseDataMaintainer, AssumeBaseMeet, IncludeClearPowerFlowSolutionAidValues, DeleteBranchesThatFlipBusOrder);

In Version 21 and earlier this script command was called **DiffFlowWriteCompleteModel**. Simulator 21 patches after January 20, 2021 will handle reading either the **DiffCaseWriteCompleteModel** or **DiffFlowWriteCompleteModel**.
Use this action to create an auxiliary file that contains information about objects that have been added or removed when comparing the present case to the base case when using the difference case comparison. Fields that have changed for objects that exist in both the present and base case can also be written to this auxiliary file. This auxiliary file can then be used to modify cases with these same changes.

"filename"	:	Name of the auxiliary file to create.
AppendFile	:	Set to YES or NO. YES means to append the saved information to "filename". NO means that "filename" will be overwritten.
SaveAdded	:	Set to YES or NO. YES means to save the added objects to the file. NO means to exclude the added objects.
SaveRemoved	:	Set to YES or NO. YES means to save the removed objects to the file. NO means to exclude the removed objects.
SaveBoth	:	Set to YES or NO. YES means to save the changed fields for the objects that exist in both the present and base case. NO means to exclude objects that occure in both cases.
KeyFields	:	Optional parameter – default is Primary Set to Primary or Secondary to specify the key field identifiers to use for objects in the resulting file.
"ExportFormat"	:	Optional parameter – default is blank This is the name of the Auxiliary File Export Format Description to use for defining the object types and fields that should be included in the auxiliary file.
UseAreaZone	:	Optional parameter – default is NO Set to YES or NO. YES means to use the Area/Zone/Owner filter for including objects in the file. NO means to ignore this filter.
UseDataMaintainer	:	Optional parameter – default is NO Set to YES or NO. YES means to use the Data Maintainer filter for including objects in the file. NO means to ignore the Data Maintainer filter.

AssumeBaseMeet : Optional parameter – default is YES
Set to YES or NO. YES means that areas/zones/owners and data maintainers that are in the base case and not in the present case meet the Area/Zone/Owner and Data Maintainer filters.

(Following added in January 31, 2020 patch of Simulator 21)

IncludeClearPowerFlowSolutionAidValues
: Optional parameter – default is YES
Set to YES or NO. YES means that the ClearPowerFlowSolutionAidValues script command is included in the auxiliary file.

DeleteBranchesThatFlipBusOrder
: Optional parameter – default is NO
Set to YES or NO. YES means that branches that have the order of their from and to terminal buses flipped will be included as both removed and added objects. The branch with the old order will be removed and the branch with the new order will be added. This is always done for transformers, but it is optional for non-transformer branches.

DiffCaseWriteBothEPC ("filename", GEFileType, UseAreaZone, BaseAreaZoneMeetFilter, Append, "ExportFormat");

Call this action to save any elements in both the base and present cases as determined using the difference flows functionality. The elements are saved in the GE EPC format. See the DiffFlowWriteRemovedEPC script command for the descriptions of the parameters that are common among all script commands used to save difference case change files in the EPC format. Parameters that are specific to DiffCaseWriteBothEPC are described here:

"ExportFormat" : Optional parameter – default is blank.
This is the name of the Auxiliary File Export Format Description to use for defining the object types that should be included in the file. Only object types that are allowed in an EPC can be included and others will be skipped. Fields that are specified with the object types in the format will be used to determine if an object has changed based on those fields changing between the present and base case. If any field has changed for an object, the entire object will be written including all fields that are required in the EPC format.

In Version 21 and earlier this script command was called **DiffFlowWriteBothEPC**. Simulator 21 patches after January 20, 2021 will handle reading either the **DiffCaseWriteBothEPC** or **DiffFlowWriteBothEPC**.

DiffCaseWriteNewEPC ("filename", GEFileType, UseAreaZone, BaseAreaZoneMeetFilter, Append);

Call this action to save any new elements determined using the difference flows functionality. The elements are saved in the GE EPC format. See the DiffFlowWriteRemovedEPC script command for the descriptions of the parameters.

In Version 21 and earlier this script command was called **DiffFlowWriteNewEPC**. Simulator 21 patches after January 20, 2021 will handle reading either the **DiffCaseWriteNewEPC** or **DiffFlowWriteNewEPC**.

DiffCaseWriteRemovedEPC ("filename", GEFileType, UseAreaZone, BaseAreaZoneMeetFilter, Append);

In Version 21 and earlier this script command was called **DiffFlowWriteNewEPC**. Simulator 21 patches after January 20, 2021 will handle reading either the **DiffCaseWriteNewEPC** or **DiffFlowWriteNewEPC**.

Call this action to save any removed elements determined using the difference flows functionality. The elements are saved in the GE EPC format.

"filename" : The path and name of the file to save.

GEFileType : Optional parameter – default is to save with the latest version. Valid options:
GE (latest version), GE14-GE21

UseAreaZone : Optional parameter – default is NO.
Set to YES or NO. YES means to save only those objects that meet the Area/Zone/Owner filter.

BaseAreaZoneMeetFilter
: Optional parameter – default is NO.
Set to YES or NO. YES means that areas or zones that are not in the difference flows base case will be treated as meeting the Area/Zone filter if that is used.

Append : Optional parameter – default is YES.
Set to YES or NO. YES means to append data to an existing file. NO means to overwrite an existing file.

DoCTGAction([contingency action]);

DoCTGAction("contingency action");

Call this action to use the formats seen in the CTGElement subdata record for Contingency Data. Note that all actions are supported, except COMPENSATION sections are not allowed.
The action must be enclosed in either brackets or double quotes.

ResetToFlatStart (FlatVoltagesAngles, ShuntsToMax, LTCsToMiddle, PSAnglesToMiddle);

Use this action to initialize the Power Flow Solution to a "flat start." The parameters are all optional and specify a conditional response depending on whether the solution is successfully found. If parameters are not passed then default values will be used.

FlatVoltagesAngles : Set to YES or NO. YES means setting all the voltage magnitudes and generator setpoint voltages to 1.0 per unit and all the voltage angles to zero. Default Value = YES.

ShuntsToMax : Set to YES or NO. YES means to increase Switched Shunts Mvar half way to maximum. Default Value = NO.

LTCsToMiddle : Set to YES or NO. YES means setting the LTC Transformer Taps to middle of range. Default Value = NO.

PSAnglesToMiddle : Set to YES or NO. YES means setting Phase Shifter angles to middle of range. Default Value = NO.

SolvePowerFlow (SolMethod, "filename1", "filename2", CreateIfNotFound1, CreateIfNotFound2);

Call this action to perform a single power flow solution. The parameters are all optional and specify a conditional response depending on whether the solution is successfully found. If parameters are not passed then default values will be used.

SolMethod	:	The solution method to be used for the Power Flow calculation. The options are: RECTNEWT : for Rectangular Newton-Raphson. POLARNEWTON : for Polar Newton-Raphson. GAUSSSEIDEL : for Gauss-Seidel. FASTDEC : for Fast Decoupled. ROBUST : for attempting the robust solution process DC : for DC power flow calculation Default = RECTNEWT.
"filename1"	:	The filename of the auxiliary file to be loaded if there is a successful solution. You may also specify STOP, which means that all AUX file execution should stop under the condition. Default = "".
"filename2"	:	The filename of the auxiliary file to be loaded if there is a NOT successful solution. You may also specify STOP, which means that all AUX file execution should stop under the condition. Default = "".
CreateIfNotFound1	:	Set to YES or NO. YES means that objects which cannot be found will be created while reading in DATA sections of filename1. Default = NO.
CreateIfNotFound2	:	Set to YES or NO. YES means that objects which cannot be found will be created while reading in DATA sections of filename2. Default = NO.

Contingency Related Actions

```
CTGApply                        ("ContingencyName");
CTGAutoInsert;
CTGCalculateOTDF                ([transactor seller], [transactor buyer], LinearMethod);
CTGClearAllResults;
CTGCloneMany                    (filter, "Prefix", "Suffix", SetSelected);
CTGCloneOne                     ("ctgname", "newctgname", "Prefix", "Suffix", SetSelected);
CTGComboDeleteAllResults;
CTGComboSolveAll                (DoDistributed, ClearAllResults);
CTGCompareTwoListsofContingencyResults(PRESENT or "ControllingFilename", PRESENT or
                                "ComparisonFilename");
CTGConvertAllToDeviceCTG        (KeepOriginalIfEmpty);
CTGCreateContingentInterfaces (filter, maxOption);
CTGCreateExpandedBreakerCTGs;
CTGCreateStuckBreakerCTGs       (filter, AllowDuplicates, "PrefixName", IncludeCTGLabel,
                                BranchFieldName, "SuffixName", "PrefixComment",
                                BranchFieldComment, "SuffixComment");
CTGDeleteWithIdenticalActions;
CTGJoinActiveCTGs               (InsertSolvePowerFlow, DeleteExisting, JoinWithSelf, "filename");
CTGPrimaryAutoInsert;
CTGProcessRemedialActionsAndDependencies(DoDelete, filter);
CTGProduceReport                ("filename");
CTGReadFilePSLF                 ("filename");
CTGReadFilePTI                  ("filename");
CTGRelinkUnlinkedElements;
CTGRestoreReference;
CTGSaveViolationMatrices        ("filename", filetype, UsePercentage, [ObjectTypesToReport],
                                SaveContingency, SaveObjects, FieldListObjectType, [FieldList]);
CTGSetAsReference;
CTGSolve                        ("ContingencyName");
CTGSolveAll                     (DoDistributed, ClearAllResults);
CTGSort                         ([SortFieldList]);
CTGWriteAllOptions              ("filename", KeyField, UseSelectedDataMaintainers,
                                SaveDependencies);
CTGWriteFilePTI                 ("filename", BusFormat, TruncateCTGLabels, "filtername", Append);
CTGWriteResultsAndOptions       ("filename", [opt1…opt19], KeyField, UseDATASection, UseConcise,
                                UseObjectIDs, UseSelectedDataMaintainers, SaveDependencies);
```

CTGApply("ContingencyName");

Call this action to apply the actions in a contingency without solving the power flow.

"ContingencyName" : This is the name of the contingency to apply.

CTGAutoInsert;

This action will auto insert contingencies for you case. Prior to calling this action, all options for this action must be specified in the Ctg_AutoInsert_Options object using the SetData script command or DATA sections.

CTGCalculateOTDF([transactor seller], [transactor buyer], LinearMethod);

This action first performs the same action as done by the CalculatePTDF([transactor seller], [transactor buyer], LinearMethod) call. It then goes through all the violations found by the contingency analysis tool and determines the OTDF values for the various contingency/violation pairs.

CTGClearAllResults;

This action will delete all contingency violations and any contingency comparison results.

CTGCloneMany(filter, "Prefix", "Suffix", SetSelected);

(Added in the November 8, 2021 patch of Simulator 22)

This command creates copies of any contingencies returned by the filter. If neither the prefix nor suffix is supplied, the command will default to appending "- Copy" to the name of the contingency being cloned to define the new contingency name. Integer indices contained in parentheses, " (0)", " (1)", etc., will be added as needed until a unique contingency name is found.

filter : Opitonal parameter – by default all contingencies will be cloned
All contingencies meeting the filter will be cloned. See the Using Filters in Script Commands section for more information on specifying the filtername.

Prefix : Optional parameter – default is blank
Prefix to prepend to the existing contingency name

Suffix : Opitonal parameter – default tis blank
Suffix to append to the existing contingency name

SetSelected : Optional parameter – default is NO
If this value is YES, then as new contingencies are created for these clones, the Selected field of the new contingencies will be set to YES.

CTGCloneOne("ctgname", "newctgname", "Prefix", "Suffix", SetSelected);

(Added in the November 8, 2021 patch of Simulator 22)

This command creates a copy of a single existing contingency. If newctgname, prefix, and suffix are all blank, "- Copy" will added to the end of the existing contingency name to specify the name of the new contingency. Regardless of how the new contingency name is specified, integer indices contained in parentheses, " (0)", " (1)", etc., will be added as needed until a unique contingency name is found.

ctgname : name of contingency to clone

newctgname : Optional parameter – default is blank
Name of new contingency. If the special keyword @CTGName is specified it will be replaced with the name of the existing contingency that is being cloned.

Prefix : Optional parameter – default is blank
Prefix to prepend to the existing contingency name. This will only be used if newctgname is blank.

Suffix : Optional parameter – default is blank
Suffix to append to the existing contingency name. This will only be used if newctgname is blank.

SetSelected : Optional parameter – default is NO
If this value is YES, then as new contingencies are created for these clones, the Selected field of the new contingencies will be set to YES.

CTGComboDeleteAllResults;

(Added in the November 18, 2021 patch of Simulator 22)

Deletes all results that are associated with contingency combination analysis. This includes violations, what occurred, combination summary results, and injection sensitivities.

CTGComboSolveAll(DoDistributed, ClearAllResults);

(Added in the November 18, 2021 patch of Simulator 22)

Call this command to run contingency combination analysis for all primary and regular/secondary contingencies that are set to not be skipped.

DoDistributed : Optional parameter – default is NO
Set to YES or NO. If set to YES, distributed methods will be used to solve contingency combination analysis if the Distributed Contingency Analysis add-on is installed. Distributed analysis requires the proper configuration and security settings to work.

ClearAllResults : Optional parameter – default is YES
Set to YES or NO. If set to YES, all existing contingency combination results will be cleared even if a primary contingency is marked to be skipped. If set to NO, only those primary contingencies that are marked to not be skipped will have their results cleared.

CTGCompareTwoListsofContingencyResults (PRESENT or "ControllingFilename",PRESENT or "ComparisonFilename");

This command compares two different contingency result lists. The first parameter is to set the Controlling List. The second parameter is to set the Comparison List.

PRESENT or "ControllingFilename"

: PRESENT wil set the present contingency analysis results as the Controlling List. If the results are in a file then you can set the path to the list as the "ControlingFilename". See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.

PRESENT or "ComparisonFilename"

: PRESENT wil set the present contingency analysis results as the Comparison List. If the results are in a file then you can set the path to the list as the "ComparisonFilename". See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.

The file types allowed are: Simulator Contingency File (*.aux), Simulator (Ver 5,6,7) Contingency Files (*.ctg), PTI Contingency Files (*.con), PTI Load Throw Over Files (*.thr;*.dat), and GE Contingency Files (*.otg).

CTGConvertAllToDeviceCTG(KeepOriginalIfEmpty);

This command is intended for use with full topology models, where breakers and disconnects are defined in addition to generators, loads, transmission lines, and so on. This function would have no affect on a traditional planning model representation, which has no breakers or disconnects explicitly defined.

The purpose of the function is to allow the user to take a contingency set that is defined with outages of breakers and disconnects in a full topology model and convert them to outages of the traditional planning model elements, such as generators, loads, transmission lines, etc. This would be used in conjunction with the ability to save a full topology model as a consolidated model. A consolidated model reduces the full model down to a traditional planning model by examining the breaker and disconnect statuses, and reducing the system down by consolidating breakers and disconnects that are in service. The resulting model is a smaller model with the traditional planning elements represented, but breakers and disconnects have been removed and nodes aggregated into bus representations. This function will also take the breaker and disconnect statuses and convert contingencies defined with the breakers and disconnects and convert them into contingencies of the planning model devices affected by opening the original breakers and disconnects. Thus you could create a contingency set that is defined for the consolidated model, and can be run on the consolidated model with the same results as if the original contingency set is run on the full topology model.

The parameter KeepOriginalIfEmpty is a YES or NO option to retain or not the original contingency definitions for any contingencies that do not end up isolating any devices. This is an optional parameter that is NO by default if it is not specified.

Note that the contingency set generated depends on the statuses of the breakers and disconnects, and that the contingencies created will be different for different statuses of breakers and disconnects in the full topology model.

CTGCreateContingentInterfaces(filter, maxOption);

This command creates an interface based on contingency violations. The contingency elements are included as contingent elements in the new interface, and the violated element is included as a monitored element.

filter	:	This is the name of an Advanced Filter. Only violation objects of type ViolationCTG that meet the named filter will be used to create new interfaces.
maxOption	:	Set to BRANCH, CTG, BRANCHCTG, or leave blank to select all violations. BRANCH – for each branch, this selects the worst violation out of all contingency violations for that branch. CTG – for each contingency, this selects the worst violation out of all branch violations for that contingency. BRANCHCTG – union of violations selected in both BRANCH and CTG.

CTGCreateExpandedBreakerCTGs;

This will convert any "Open with Breakers" or "Close with Breakers" contingency actions into OPEN or CLOSE actions on explicit breakers. This will permanently modify the contingency definitions.

CTGCreateStuckBreakerCTGs(filter, AllowDuplicates, "PrefixName", IncludeCTGLabel, BranchFieldName, "SuffixName", "PrefixComment", BranchFieldComment, "SuffixComment");

This command creates new contingencies from contingencies that have explicit breaker outages defined. New contingencies will be created by treating each breaker as stuck in turn. The new contingencies will be comprised of all existing elements, minus the stuck breaker outage, plus open actions for breakers that are identified to isolate the stuck breakers. Only branches with Branch Device Type of Breaker will be considered in determining the stuck breakers.

All of the following parameters are optional. If not specified, the defaults will be used.

filter	:	Only contingencies that meet the specified filter will be set. See the Using Filters in Script Commands section for more information on specifying the filtername. Default is to process all contingencies.
AllowDuplicates	:	Set to YES or NO. YES means that contingencies with the same actions as existing or newly created contingencies will be allowed. Default is NO.

"PrefixName", IncludeCTGLabel, BranchFieldName, and "SuffixName" are used to name the new contingencies in the format: `PrefixName_Contingency Label_BranchFieldName_SuffixName`.

"PrefixName"	:	string that is used as the prefix of the new contingency name. Default is blank.
IncludeCTGLabel	:	Set to YES or NO. YES means that the name of the existing contingency will be used as part of the new contingency. Default is YES.
BranchFieldName	:	variablename of the Branch field whose value will be used in the naming of the new contingency in the format `variablenamelegacy:location:digits:rod` or `concisename:digits:rod`. The Branch used to evaluate the variablename is the stuck breaker. Default is blank.
"SuffixName"	:	string that is used as the suffix of the new contingency name. Default is "STK".

"PrefixComment", BranchFieldComment, and "SuffixComment" are used to create a comment for new contingency actions in the format: `PrefixComment_BranchFieldComment_SuffixComment`.

"PrefixComment"	:	string that is used as the prefix of the new contingency action comment. Default is blank.

BranchFieldComment : variablename of the Branch field whose value will be used in the naming of the new contingency action comment in the format `variablenamelegacy:location:digits:rod` or `concisename:digits:rod`. The Branch used to evaluate the variablename is the breaker in the new contingency action. Default is blank.

"SuffixComment" : string that is used as the suffix of the new contingency action comment. Default is blank.

CTGDeleteWithIdenticalActions;

(Added in the January 23, 2023 patch of Simulator 22)
This action deletes contingencies that have identical actions. The first contingency alphabetically is retained while all others with identical actions are deleted. Messages are added to the log detailing which contingencies are deleted.

CTGJoinActiveCTGs(InsertSolvePowerFlow, DeleteExisting, JoinWithSelf, "filename");

This command creates new contingencies that are a join of the current contingency list and a list read in from an auxiliary file or the current list itself. Contingencies with their Skip field set to YES will not be included in the join.

InsertSolvePowerFlow : Set to YES or NO. YES means to insert the solve power flow solution action between the joined contingency actions.

DeleteExisting : Set to YES or NO. YES means to delete the existing contingencies and only keep the joined contingencies.

JoinWithSelf : Set to YES or NO. YES means that the current contingency list will be joined with itself instead of contingencies specified in a file. If set to YES, the "filename" parameter does not have to be specified.

"filename" : Name of auxiliary file containing contingencies to join with the current contingency list. This does not have to be specified if JoinWithSelf = YES.

CTGPrimaryAutoInsert;

(Added in the October 7, 2022 patch of Simulator 22)
This action will auto insert Primary Contingencies. Prior to calling this action, all options for this action must be specified in the Ctg_AutoInsert_Options object using the SetData script command or DATA sections.

CTGProcessRemedialActionsAndDependencies(DoDelete, filter);

Added in the May 12, 2020 patch of Simulator 21
Remedial Actions and any Model Conditions, Model Filters, Model Expressions, and Model Planes being used by the specified Remedial Actions will be deleted or have their Selected field set to YES. Model Conditions, Model Filters, Model Expressions, and Model Planes that are being used by other objects or not being used by a specified Remedial Action will not be deleted or marked.

DoDelete : Set to YES to delete remedial actions and dependencies. Set to NO to set the Selected field to YES instead of deleting.

filter : Optional parameter - default is blank
Only Remedial Actions that meet the specified filter will be processed. See the Using Filters in Script Commands section for more information on specifying the filtername. AREAZONE is not a valid filter. Default is to process all remedial actions.

CTGProduceReport("filename");

Produces a text-based contingency analysis report using the settings defined in CTG_Options.

CTGReadFilePSLF("filename");

Use this action to load a file in the PSLF OTG format and create contingencies.

"filename"	:	Name of the file to read. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.

CTGReadFilePTI("filename");

Use this action to load a file in the PTI CON format and create contingencies.

"filename"	:	Name of the file to read. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.

CTGRelinkUnlinkedElements;

This will attempt to relink unlinked elements in the contingency records.

CTGRestoreReference;

Call this action to reset the system state to the reference state for contingency analysis.

CTGSaveViolationMatrices("filename", filetype, UsePercentage, [ObjectTypesToReport], SaveContingency, SaveObjects, FieldListObjectType, [FieldList], IncludeUnsolvableCTGs);

This command will save contingency violations in a matrix format. Multiple files can be created for each type of violation as well as a file showing all contingencies that have violations and the objects that are violations under each contingency.

"filename"	:	Base name of the files to save. It is possible to save multiple files. This file name will be appended with an underscore and name of the object type that is being saved in the file.
filetype	:	There following options are available:
CSVNOHEADER	:	save as a normal CSV text file, without the AUX file formatting. The object name and field variable names are NOT included.
CSVCOLHEADER	:	save as a normal CSV without the AUX syntax and with the first row showing column headers you would see in a case information display
UsePercentage	:	Set to YES or NO. Set to YES the values will be reported as a percentage loading. If NO, the actual values will be reported.
[ObjectTypesToReport]	:	Comma delimited list of object types to include in the results. Options are BRANCH, BUS, INTERFACE, and CUSTOMMONITOR. If saving results by contingency (rows show contingencies that contain violations), the columns will only contain objects of these specified types. If saving results by object (row show objects that are violated under any contingency), files will only be created for these specified types.
SaveContingency	:	Set to YES or NO. Set to YES to save a file containing all contingencies that have at least one violation. The results will be reported as contingencies in rows and the violated elements in columns. Only violations of the specified [ObjectTypesToReport] will be included. Contingencies that have no violations because they failed to solve can be included by setting the IncludeUnsolvableCTGs to YES.
SaveObjects	:	Set to YES or NO. Set to YES to save a file for each of the object types specified in [ObjectTypesToReport]. The objects will be in rows and the columns will be the contingencies under which the objects are violated.

FieldListObjectType : Optional parameter – default is blank.
Additional fields can be included depending on the object type that is being saved in the rows of each file. This parameter specifies the object type associated with the FieldList. Valid options are BRANCH, BUS, INTERFACE, CUSTOMMONITOR, or CONTINGENCY. As an example, suppose that this parameter is set to BRANCH and a FieldList is specified. If SaveObjects is also YES, a file will be saved showing branch violations in each row of the file. In addition to columns showing contingencies and the violation value of the branch, columns will be added showing the value of the branch in the present system state for each of the fields in the FieldList.

FieldList : Optional parameter – default is blank.
Additional fields can be included depending on the object type that is being saved in the rows of each file. This parameter specifies the additional fields to save. See the FieldListObjectType parameter for an explanation of how these extra fields are saved to file.

IncludeUnsolvableCTGs
: Optional parameter – default is NO.
Set to YES or NO. Set to YES to include contingencies that have been processed but did not solve either because the power flow failed or an abort action took place. The Solved field will be automatically added to the file that lists results by contingency. Set to NO to only include contingencies that solved. This option is only relevant if the SaveContingency option is set to YES.

CTGSetAsReference;

Call this action to set the present system state as the reference for contingency analysis.

CTGSolve("ContingencyName");

Call this action solve a particular contingency. The contingency is denoted by the "Contingency Name" parameter. The system state remains in the post-contingency state following the application of this command.

CTGSolveAll(DoDistributed, ClearAllResults);

Call this action to solve all of the contingencies that are not marked to be skipped.

DoDistributed : Optional parameter – default is NO
Set to YES or NO. If set to YES, distributed methods will be used to solve contingency analysis if the Distributed Contingency Analysis add-on is installed. Distributed analysis requires the proper configuration and security settings to work.

ClearAllResults : Optional parameter – default is YES
Set to YES or NO. If set to YES, all existing contingency results will be cleared even if a contingency is marked to be skipped. If set to NO, only those contingencies that are marked to not be skipped will have their results cleared.

CTGSort([SortFieldList]);

(Added in the January 23, 2023 patch of Simulator 22)
This command sorts the contingencies stored in Simulator's internal data structure. This is different than sorting contingencies in case information displays in the GUI or sorting data when it is written to an auxiliary file. Contingencies are processed in the order in which they are stored in the internal data structure, and they are not sorted by default in this structure; contingencies are added to the internal data structure in the order in which they are created. This could be significant for other actions like CTGJoinActiveCTGs if the goal is to join contingencies alphabetically.

[SortFieldList] : Optional parameter – the default is to sort alphabetically by contingency name
This allows the specification of a sort order based on the available fields for the contingency object type. The format is: [variablename1:+:0, variablename2:-:1] where

- variablename : the first parameter is the name of the field by which to sort. There is no limit to how many fields can be specified for sorting. For fields that require a location other than zero, variablename can be in the format fieldname:location.
- + or - : the second parameter indicates sort ascending for + and sort descending for -. This parameter must be specified.
- 0 or 1 : for the third parameter 0 means case insensitive and do not use absolute value, 1 means case sensitive or use absolute value. This parameter is optional.

CTGWriteAllOptions("filename", KeyField, UseSelectedDataMaintainer, SaveDependencies, UseAreaZoneFilters);

Writes out all information related to contingency analysis as an auxiliary file using concise variable names and headers. Data is written using DATA sections instead of SUBDATA sections.

"filename" : Name of the auxiliary file to save.

KeyField : Optional parameter – default is PRIMARY
Indicates the identifier that should be used for the data. Valid entries are PRIMARY, SECONDARY, or LABEL. PRIMARY will save using bus numbers and other primary key fields. SECONDARY will save using bus name and nominal kV and other secondary fields. LABEL will save using device labels. If no labels are specified then the primary key field will be used.

UseSelectedDataMaintainer : Optional parameter – default is NO
Set to YES or NO. YES means to save only the information belonging to Data Maintainers where the Selected field is set to YES. NO means to save all information. Default is NO.

SaveDependencies : Optional parameter – default is NO
Set to YES or NO. YES means that all relevant objects that are required to define the selected objects will also be saved. NO means to only save the selected objects.

UseAreaZoneFilters : Optional parameter – default is NO
Set to YES or NO. YES means to save only the information for objects that meets the Area, Zone, Owner filters for Contingency Options and Limit Monitoring Settings related to the Area, Zone, Bus, Gen, and Shunt Objects. (Opt2 and Opt3 of CTGWriteResultsAndOptions)

See the CTGWriteResultsAndOptions script command for a list of the option settings that are considered. The equivalent options are set as follows for this script command:

```
Opt1 = NO , Opt2 = YES, Opt3 = YES, Opt4 = YES ,Opt5 = NO ,Opt6 = NO, Opt7 = NO, Opt8
= YES, Opt9 = YES , Opt10 = NO , Opt11 = NO, Opt12 = YES, Opt13 = YES, Opt14 = YES,
Opt15 = YES, Opt16 = YES, Opt17 = NO, Opt18 = YES, Opt19 = YES
```

The UseObjectIDs parameter with the CTGWriteResultsAndOptions script command is set automatically for this script command. The equivalent setting is YES_MS_3W.

CTGWriteFilePTI("filename", BusFormat, TruncateCTGLabels, "filtername", Append);

Write contingencies to file in the PTI CON format.

"filename" : The name of the text file to write out.

BusFormat : How to identify buses:

- Number : Using numbers
- Name8 : Using BusName_NomkV strings truncated to 8 characters
- Name12 : Using BusName_NomkV strings truncated to 12 characters

TruncateCTGLabels : Set to YES or NO. YES means that the contingency labels will be truncated after 12 characters.

(Following added in the April 1, 2021 patch of Simulator 22)

"filtername" : Optional – default is blank and all contingencies will be saved
This filter will be applied to the Contingency object type to specify which contingencies should be saved to file.
See the Using Filters in Script Commands section for more information on specifying the filtername.

Append : Optional – default is NO
Set to YES or NO. YES means to append the saved information to "filename". NO means that "filename" will be overwritten.

CTGWriteResultsAndOptions("filename", [opt1, opt2, opt3, ..., opt19], KeyField, UseDATASection, UseConcise, UseObjectIDs, UseSelectedDataMaintainers, SaveDependencies, UseAreaZoneFilters);

Writes out all information related to contingency analysis as an auxiliary file.

"filename" : Name of the auxiliary file to save.

[opt1, opt2, ..., opt19] : Each entry in the Option Settings parameter is either a YES or NO entry corresponding to the following options. These are all optional parameters, so if they are not specified or blank, the default entry given for each will be used.

- Opt1 : Save Unlinked Contingency Actions, default = NO
- Opt2 : Save Contingency Options, default = YES
- Opt3 : Save Limit Monitoring Settings, default = NO
- Opt4 : Save General Power Flow Solution Options, default = YES
- Opt5 : Save List Display Settings, default = NO
- Opt6 : Save Contingency Results, default = YES
- Opt7 : Save Inactive Violations, default = YES
- Opt8 : Save Interface Definitions, default = NO
- Opt9 : Save Injection Group Definitions, default = NO
- Opt10 : Save Distributed Computing Options, default = YES
- Opt11 : Suppress Gen and Load options when writing out Options, default = NO
- Opt12 : Save Contingency Definitions, default = YES
- Opt13 : Save Remedial Actions and Global Actions, default = YES
- Opt14 : Save Custom Monitor Definitions, default = YES
- Opt15 : Save Model Conditions, Model Filters, and Model Expressions, default = YES
- Opt16 : Save Advanced Filters used as part of Custom Monitors, Model Conditions, Remedial Actions, and Global Actions, default = YES
- Opt17 : Save Limit Cost Functions with Limit Sets, default = YES
- Opt18 : Automatically Convert Contingency Blocks and Global Actions, default = NO
- Opt19 : Save Voltage Control Groups, default = YES

KeyField : Optional parameter – default is PRIMARY
Indicates the identifier that should be used for the data. Valid entries are PRIMARY, SECONDARY, or LABEL. PRIMARY will save using bus numbers and other primary key fields. SECONDARY will save using bus name and nominal kV and other secondary fields. LABEL will save using device labels. If no labels are specified then the primary key field will be used.

UseDATASection : Optional parameter – default is NO
Set this to YES or NO. If YES, data that by default is specified using SUBDATA sections will instead be specified using DATA sections. For example, the actions that define a contingency by default are specified using a SUBDATA section. If choosing to use the DATA section instead, each action will be specified in a DATA record belonging to the ContingencyElement objecttype.

UseObjectIDs : Optional parameter – default is NO
Possible settings are YES, NO, YES_MS, YES_3W, and YES_MS_3W.

- YES : Any input with YES means to use the ObjectID field when writing objects with contingency settings instead of using multiple key fields to identify an object. The advantage to using ObjectIDs is that you only have one field to be used as an identifier rather than a changing number of fields that depends on the type of object. This will simplify your auxiliary file.
- NO : This means to use the specified key fields to identify an object.
- MS : Any input with MS means to write out a multi-section line by identifying it by the from bus, to bus, and circuit ID of the multi-section followed by the number of the particular section. This follows the PSLF format. If not writing out in this manner, individual sections will be written based on their from bus, to bus, and circuit ID.
- 3W : Any input with 3W means to write out a three-winding transformer using the buses at the three terminals of the transformer followed by the circuit ID with the first bus listed being the particular winding that is desired. If not writing out in this manner, a particular winding will be written with its terminal bus, the star bus of the transformer, and the circuit ID of the transformer.

UseSelectedDataMaintainer : Optional parameter – default is NO
Set to YES or NO. YES means to save only the information belonging to Data Maintainers where the Selected field is set to YES. NO means to save all information.

SaveDependencies : Optional parameter – default is NO
Set to YES or NO. YES means that all relevant objects that are required to define the selected objects will also be saved. NO means to only save the selected objects.

UseAreaZoneFilters : Optional parameter – default is NO
Set to YES or NO. YES that to save only the information for objects that meets the Area, Zone, Owner filters for Contingency Options and Limit Monitoring Settings related to the Area, Zone, Bus, Gen, and Shunt Objects. (Opt2 and Opt3)

Fault Related Actions

```
Fault                          ([BUS num], faulttype, R, X);
Fault                          ([BRANCH nearbusnum farbusnum ckt], faultlocation, faulttype, R,
                               X);
FaultClear;
FaultMultiple                  (UseDummyBus);
LoadPTISEQData                 ("filename", version);
```

Fault([Bus num], faulttype, R, X);

Fault([BRANCH nearbusnum farbusnum ckt], faultlocation, faulttype, R, X);

Call this function to calculate the fault currents for a fault. If the fault element is a bus then do not specify the fault location parameter. If the fault element is a branch, then the fault location is required.

[BUS num] : This specifies the bus at which the fault occurs. You may also specify the bus using secondary keys or labels.

[BUS "name_nomkv"]
[BUS "label"]
[BRANCH nearbusnum farbusnum ckt]
: This specifies the branch on which the fault occurs. You may also specify the branch using secondary keys or labels.

[BRANCH "name_kv1" "name_kv2" ckt]
[BRANCH "buslabel1" "buslabel2" ckt]
[BRANCH "label"]

Faultlocation : This specifies the percentage distance along the branch where the fault occurs. This percent varies from 0 (meaning at the nearbus) to 100 (meaning at the far bus)

Faulttype : This specified the type of fault which occurs. There are four options:

- SLG : Single Line To Ground fault
- LL : Line to Line Fault
- 3PB : Three Phase Balanced Fault
- DLG : Double Line to Group Fault.

R, X : These parameters are optional and specify the fault impedance. If none are specified, then a fault impedance of zero is assumed.

FaultClear;

Clears a single fault that has been calculated with the Fault script command.

FaultMultiple(UseDummyBus);

Runs fault analysis on a list of defined faults.

UseDummyBus : Optional parameter – default is NO
Set to YES or NO. If YES, dummy buses should be created and inserted at the specified percent location for branch faults. Faults will be calculated at the dummy buses. If NO, the fault will be calculated at the branch terminal bus that is closest to the specified location.

LoadPTISEQData("filename", version);

Added in the October 21, 2019 patch of Simulator 21

Loads sequence data in the PTI format.

"filename" : Name of file containing sequence data.

version : Integer representing the PTI version of the SEQ file to open.

ATC (Available Transfer Capability) Related Actions

```
ATCCreateContingentInterfaces (filter);
ATCDataWriteOptionsAndResults ("filename", AppendFile, KeyField);
ATCDeleteAllResults;
ATCDeleteScenarioChangeIndexRange    (ScenarioChangeType, [IndexRange]);
ATCDetermine                  ([transactor seller], [transactor buyer], DoDistributed,
                              DoMultipleScenarios);
ATCDetermineMultipleDirections(DoDistributed, DoMultipleScenarios);
ATCDetermineATCFor            (RL, G, I, ApplyTransfer);
ATCDetermineMultipleDirectionsATCFor(RL, G, I);
ATCIncreaseTransferBy         (amount);
ATCRestoreInitialState;
ATCSetAsReference;
ATCTakeMeToScenario           (RL, G, I);
ATCWriteResultsAndOptions     ("filename", AppendFile);
ATCWriteScenarioLog           ("filename", AppendFile, filter);
ATCWriteScenarioMinMax        ("filename", filetype, AppendFile, [fieldlist], Operation,
                              OperationField, GroupScenario, [RLFilter], [GFilter], [IFilter],
                              DirectionFilter, DoGroupDirection, filter,
                              PreferIterativelyFound);
ATCWriteToExcel               ("worksheetname", [fieldlist]);
ATCWriteToText                ("filename", filetype, [fieldlist]);
```

ATCCreateContingentInterfaces(filter);

This command creates an interface based Transfer Limiter results from an ATC run. Each Transfer Limiter is comprised of a Limiting Element/Contingency pair. Each interface is then created with contingent elements from the contingency and the Limiting Element included as the monitored element.

filter : This is the name of an Advanced Filter. Only objects of type TransferLimiter that meet the named filter will be used to create new interfaces.

ATCDeleteAllResults;

Deletes all ATC results including TransferLimiter, ATCExtraMonitor, and ATCFlowValue object types.

ATCDeleteScenarioChangeIndexRange(ScenarioChangeType, [IndexRange]);

ATC scenarios are defined by RL (line rating and zone load), G (generator), and I (interface rating) changes. This command allows the deletion of entries within one of these change types by specifying the indices of the changes that should be deleted.

ScenarioChangeType : RL, G, or I to indicate the scenario change type to delete.

IndexRange : Comma-delimited list of integer ranges that must be enclosed in square brackets. The indices start at 0.

To delete line rating and zone load changes for indices 0 to 2, 5, and 7 to 9 the following is the appropriate syntax:

```
ATCDeleteScenarioChangeIndexRange(RL, [0-2, 5, 7-9]);
```

ATCDetermine([transactor seller], [transactor buyer], DoDistributed, DoMultipleScenarios);

Use this action to calculate the Available Transfer Capability (ATC) between a seller and a buyer. The buyer and seller must not be the same. Other options regarding ATC calculations should be set with the ATC_Options object type. If the distributed ATC add-on is installed, the optional DoDistributed flag may bet set to indicate that the ATC should be solved using the distributed methods.

[transactor seller] : The seller (or source) of power. There are six possible settings:
[AREA num], [AREA "name"], [AREA "label"]
[ZONE num], [ZONE "name"], [ZONE "label"]
[SUPERAREA "name"], [SUPERAREA "label"]

[INJECTIONGROUP "name"], [INJECTIONGROUP "label"]
[BUS num], [BUS "name_nomkv"], [BUS "label"]
[SLACK]

[transactor buyer] : The buyer (or sink) of power. There are six possible settings, which are the same as for the seller.

DoDistributed : Optional parameter – default is NO
Set to YES to use the distributed ATC solution method.

DoMultipleScenarios : Optional parameter – default is set to YES if scenarios are defined
Set to YES to process each defined scenario.

ATCDetermineMultipleDirections(DoDistributed, DoMultipleScenarios);

Use this action to calculate the Available Transfer Capability (ATC) for all defined directions. Other options regarding ATC calculations should be set with the ATC_Options object type. If the distributed ATC add-on is installed, the optional DoDistributed flag may bet set to indicate that the ATC should be solved using the distributed methods.

DoDistributed : Optional parameter – default is NO
Set to YES to use the distributed ATC solution method.

DoMultipleScenarios : Optional parameter – default is NO
Set to YES to process each defined scenario for all defined directions.

ATCDetermineATCFor(RL, G, I, ApplyTransfer);

Call this action to determine the ATC for Scenario RL, G, I.

ApplyTransfer : Optional parameter – default is NO
Set to YES or NO. Set this value to YES to leave the system state at the transfer level that was determined. When using the Iterated Linear then Full Contingency solution method, the system state will retain the transfer level but the contingency will not be applied.

ATCDetermineMultipleDirectionsATCFor(RL, G, I);

Call this action to determine the ATC for Scenario RL, G, I for all defined directions.

ATCIncreaseTransferBy(amount);

Call this action to increase the transfer between the seller and buyer.

ATCRestoreInitialState;

Call this action to restore the initial state for the ATC tool.

ATCSetAsReference;

Call this action to set the present system state to the reference state for ATC analysis.

ATCTakeMeToScenario(RL, G, I);

Call this action to set the present case according to Scenario RL, G, I.

ATCWriteAllOptions("filename", AppendFile, KeyField);

Renamed to ATCDataWriteOptionsAndResults in December 9, 2021 patch of Simulator 22

ATCDataWriteOptionsAndResults("filename", AppendFile, KeyField);

Added in the May 4, 2020 patch of Simulator 21

Writes out all information related to ATC analysis to an auxiliary file. Saves the same information as the ATCWriteResultsAndOptions script command. Auxiliary file is formatted using the concise format for DATA section headers and variable names. Data is written using DATA sections instead of SUBDATA sections.

"filename" : Name of the auxiliary file to save. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.

AppendFile : Optional parameter - default is YES
YES means to append results to existing "filename." NO means to overwrite "filename" with the results.

KeyField : Optional parameter – default is PRIMARY
Indicates the identifier that should be used for the data. Valid entries are PRIMARY, SECONDARY, or LABEL. PRIMARY will save using bus numbers and other primary key fields. SECONDARY will save using bus name and nominal kV and other secondary fields. LABEL will save using device labels. If no labels are specified then the primary key field will be used.

ATCWriteResultsAndOptions("filename", AppendFile);

Writes out all information related to ATC analysis to an auxiliary file. This includes Contingency Definitions, Remedial Action Definitions, Limit Monitoring Settings, Solution Options, ATC Options, ATC results, as well as any Model Criteria that are used by the Contingency and Remedial Action Definitions.

(Following changes made with the May 4, 2020 patch of Simulator 21)
Contingency and Remedial Action definitions will always be saved along with dependencies and only those object types that are dependencies will be saved. This means that if a Remedial Action definition uses a Model Filter, that Model Filter along with any Model Filter Conditions, i.e. other Model Filters or Model Conditions, will be saved. If a model criteria object is not being used by a Remedial Action or Contingency it will not be saved. Objects that are saved if they are dependencies include: Model Conditions, Model Filters, Model Planes, Model Expressions, Model Result Overrides, Interfaces, Injection Groups, Calculated Fields, and Expressions.

Dependencies for the ATC setup are also included. This includes Injection Groups that are used as the seller or buyer, Interfaces that are used in ATC Extra Monitor definitions, and Interfaces that are used in Multiple ATC Scenario definitions.

"filename" : Name of the auxiliary file to save. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.

AppendFile : Optional parameter - default is YES
YES means to append results to existing "filename." NO means to overwrite "filename" with the results.

ATCWriteScenarioLog("filename", AppendFile, filter);

Added in the July 7, 2021 patch of Simulator 22
Writes out detailed log information for ATC Multiple Scenarios to a text file. If no scenarios have been defined, no file will be created; this is not treated as a fatal error, and an auxiliary file containing this command will continue to process subsequent commands.

"filename" : Name of log file. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.

AppendFile : Optional parameter - default is NO
YES means to append log information to existing "filename." NO means to overwrite "filename" with the log information.

filter : Optional parameter – default is blank
Log information will only be written for ATC scenarios that meet the specified filter. See the Using Filters in Script Commands section for more information on specifying the filtername. Default is to write the log for all ATC scenarios.

ATCWriteScenarioMinMax("filename", filetype, AppendFile, [fieldlist], Operation, OperationField, GroupScenario, [RLFilter], [GFilter], [IFilter], DirectionFilter, DoGroupDirection, filter, PreferIterativelyFound);

Added in the June 10, 2022 patch of Simulator 22

Writes out TransferLimiter results from multiple scenario ATC calculations. The results are grouped based on the input parameters, and the minimum, maximum, or minimum and maximum limiter from each group is written to file.

- "filename" : Name of file. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.
- filetype : There are several options for the filetype
  - AUXCSV : save as a comma-delimited auxiliary data file
  - AUX : save as a space-delimited auxiliary data file
  - CSV : save as a normal CSV file without the AUX file syntax. The first few lines of the text file will represent the object name and field variable names.
  - CSVNOHEADER : save as a normal CSV text file, without the AUX file formatting. The object name and field variable names are NOT included.
  - CSVCOLHEADER : save as a normal CSV without the AUX syntax and with the first row showing the normal column headers seen in a case information display
- AppendFile : YES means to append to existing "filename." NO means to overwrite "filename."
- [fieldlist] : Comma-delimited list of fields to save. For numeric fields, the number of digits and the number of decimal places (digits to right of decimal) can be specified by using the following format for the field, `variablenamelegacy:location:digits:rod` or `concisename:digits:rod`. See the Specifying Field Variable Names in Script Commands topic for more information on specifying this list.
- Operation : This is the operation to perform on each grouping of transfer limiters. The OperationField specifies the value to use when performing the operation.
  - MIN : Write the minimum limiter for each grouping. If there are limiters for the grouping, at most one limiter per grouping will be written to the file.
  - MAX : Write the maximum limiter for each grouping. If there are limiters for the grouping, at most one limiter per grouping will be written to the file.
  - MIN/MAX : Write the minimum and maximum limiter per grouping. If there are limiters for the grouping, at most two limiters per grouping will be written to the file. There will only be one limiter if the minimum and maximum limiters are the same.
- OperationField : Variable name of the field to use for the value in the operation. Valid entries are `TransferLimit` or any available `ATC_ExtraMonitor` field.

GroupScenario : Indicates which type of scenario to use for the grouping.

- None : The results are not grouped by scenarios. All scenario results are included as one group. Direction grouping also applies.
- RL : The results are grouped by RL scenario. For each RL scenario all results for the G and I scenarios are included. Direction grouping also applies.
- G : The results are grouped by G scenario. For each G scenario all results for the RL and I scenarios are included. Direction grouping also applies.
- I : The results are grouped by I scenario. For each I scenario all results for the RL and G scenarios are included. Direction grouping also applies.

[RLFilter] : Optional parameter – default is blank.
Determines which RL scenarios are included in the groupings. If left blank, all RL scenarios are included. Specify the scenarios using an integer range list enclosed in square brackets, which is a comma-separated list of either a single integer or a range of integers such as [1, 3-4, 7]. This parameter is ignored if **filter** is not blank.

[GFilter] : Optional parameter – default is blank
Determines which G scenarios are included in the groupings. If left blank, all G scenarios are included. Specify the scenarios using an integer range list enclosed in square brackets, which is a comma-separated list of either a single integer or a range of integers such as [1, 3-4, 7]. This parameter is ignored if **filter** is not blank.

[IFilter] : Optional parameter – default is blank
Determines which I scenarios are included in the groupings. If left blank, all I scenarios are included. Specify the scenarios using an integer range list enclosed in square brackets, which is a comma-separated list of either a single integer or a range of integers such as [1, 3-4, 7]. This parameter is ignored if **filter** is not blank.

DirectionFilter : Optional parameter – default is IGNORE
A single direction or multiple directions can be run for multiple scenarios. This parameter determines which set of directions to write in the results.

- IGNORE : Use only the single direction results in the groupings. Scenario grouping also applies.
- ALL : Use all of the multiple direction results in the groupings. The **DoGroupDirection** parameter will determine if groupings will be formed for each direction or if all directions will be combined before determining the scenario groupings.
- "Direction Name" : Only use the specified direction in the groupings. Scenario grouping also applies.

DoGroupDirection : Optional parameter – default is YES
This parameter is used only if **DirectionFilter** = ALL. Set to YES if groupings should be formed for each of the multiple direction results. Within a direction grouping, scenarios will also be grouped based on specified parameters. Set to NO if all directions are considered together and the only grouping that should be done is based on scenario groupings.

filter : Optional parameter – default is blank
Groupings will be formed only for TransferLimiters that meet this filter. **RLFilter**, **GFilter**, and **IFilter** will be ignored when specifying this filter, but all other options will apply. See the Using Filters in Script Commands section for more information on specifying the filtername.

PreferIterativelyFound : Optional parameter – default is YES
Set to YES so that transfer limiters that have been iteratively found will take precedence when determining the limiter that meets the selected operation for each grouping. If there are no limiters that have been iteratively found or this is set to NO, the first limiter that is found that meets the operation will be reported for each grouping.

ATCWriteToExcel("worksheetname", [fieldlist]);

Sends ATC analysis results to an Excel spreadsheet. This script command is available only for Multiple Scenarios ATC analysis.

"worksheetname" : The name of the Excel sheet where the results will be sent to.

(Following added in October 11, 2019 patch of Simulator 21)

fieldlist : Optional parameter
If specified, the results will be saved including this list of fields. If not specified, the results will be saved with the fields that are specified with the TransferLimiter DataGrid. It is possible that the results could be blank if the TransferLimiter DataGrid has not been initialized by either opening the ATC dialog or loading the DataGrid settings from an auxiliary file. To make sure that results are stored and the desired fields are included, it is suggested that the fieldlist option be used.

ATCWriteToText("filename", filetype, [fieldlist]);

This is used with Multiple Scenario ATC analysis. Multiple files are created with "filename" as the primary identifier and the Interface scenario label appended to the end of the filename. Separate files are created for each of the Interface scenarios. Results inside the files are separated into sections based on the number of Rating/Load scenarios.

"filename" : Primary identifier for the name of the file in which to save the results. "filename" gets appended with the Interface scenario label to complete the filename.

filetype : Either TAB or CSV. This indicates the delimiter to use when writing out the file(s). This is an optional parameter with TAB being the default if omitted.

(Following added in October 11, 2019 path of Simulator 21)

fieldlist : Optional parameter
If specified, the results will be saved including this list of fields. If not specified, the results will be saved with the fields that are specified with the TransferLimiter DataGrid. It is possible that the results could be blank if the TransferLimiter DataGrid has not been initialized by either opening the ATC dialog or loading the DataGrid settings from an auxiliary file. To make sure that results are stored and the desired fields are included, it is suggested that the fieldlist option be used.

GIC (Geomagnetically Induced Current) Related Actions

```
GICCalculate                    (MaxField, Direction, SolvePF);
GICClear;
GICLoad3DEField                 (FileType, "FileName", SetupOnLoad);
GICTimeVaryingCalculate         (TheTime, SolvePF);
GICTimeVaryingAddTime           (NewTime);
GICTimeVaryingDeleteAllTimes;
GICTimeVaryingEFieldCalculate (TheTime, SolvePF);
GICTimeVaryingElectricFieldsDeleteAllTimes;
GICWriteOptions                 ("FileName", KeyField);
```

GICCalculate(MaxField, Direction, SolvePF);

Calculates the "Single Snapshot" using GICSolution Options

MaxField	:	Maximum Electric Field in Volts/km
Direction	:	Storm Direction, Degrees from 0 to 360
SolvePF	:	Select YES or NO to include GIC in the Power Flow

GICClear;

Clear GIC Values

GICLoad3DEfield(FileType, "FileName", SetupOnLoad);

Loads GIC data including time varying fields.

FileType	:	Type of file to be loaded. Options are CSV, B3D, JSON, and DAT.
FileName	:	Name of the file to be loaded
SetupOnLoad	:	Select YES to run procedure to setup time varying series after loading file or NO to skip the setup process.

GICTimeVaryingCalculate(TheTime,SolvePF);

Calculates "Time Varying Input" GIC at specified time

TheTime	:	Specified Time Point
SolvePF	:	Select YES or NO to include GIC in the Power Flow

GICTimeVaryingAddTime(NewTime);

Adds a new input values at specified time

NewTime	:	New Time for new input values

GICTimeVaryingDeleteAllTimes;

Delete All Input time varying voltage input values

GICTimeVaryingEFieldCalculate(TheTime,SolvePF);

Run the GIC solution using the time varying non-uniform AER data.

TheTime	:	Specified Time Point
SolvePF	:	Select YES or NO to include GIC in the Power Flow

GICTimeVaryingElectricFieldsDeleteAllTimes;

Added in the July 28, 2022 patch of Simulator 22.

Clear all the time varying electric field input values.

GICWriteOptions("FileName", KeyField);

Calculates the "Single Snapshot" using GICSolution Options

FileName	:	Name of Aux file name to write out the options
KeyField	:	KeyField indicates the identifier that should be used for the data. Valid entries are PRIMARY, SECONDARY, or LABEL. The default setting is PRIMARY. PRIMARY will save using bus numbers and other primary key fields. SECONDARY will save using bus name and nominal kV and other secondary fields. LABEL will save using device labels. If no labels are specified then the primary key field will be used.

ITP (Integrated Topology Processing) Related Actions

```
CloseWithBreakers              (objecttype, filter or [object identifier],
                               OnlyEnergizeSpecifiedObjects, [SwitchingDeviceTypes],
                               CloseNormallyClosedDisconnects);
ExpandAllBusTopology;
ExpandBusTopology              (BusIdentifier, TopologyType);
OpenWithBreakers               (objecttype, filter or [object identifier],
                               [SwitchingDeviceTypes], OpenNormallyOpenDisconnects);
SaveConsolidatedCase           ("filename", filetype, [BusFormat, TruncateCtgLabels,
                               AddCommentsForObjectLabels]);
```

CloseWithBreakers(objecttype, filter or [object identifier], OnlyEnergizeSpecifiedObjects, [SwitchingDeviceTypes], CloseNormallyClosedDisconnects);

This action is used to specify which objects are to be energized by closing breakers and to actually close those breakers. The status of an object will be set to closed if necessary in addition to closing the breakers. If only the status of an object needs to be changed to close an object, that will occur without requiring any breakers to be closed.

objecttype : Objects that are valid to be energize. Only allowed for Buses, Generators, Loads, Transmission Lines, Switched Shunts, DC Lines, Injection Groups, and Interfaces.

Filter : The second parameter can either be a filter specification or an object identifier. When specifying a filter, the following options are available:

- SELECTED : only objects whose Selected field = YES will be energized
- AREAZONE : only objects that meet the area/zone/owner filters will be energized
- "FilterName" : only objects that meet the specified filter will be energized. See the Using Filters in Script Commands section for more information on specifying the filtername.

[object identifier] : The second parameter can either be a filter specification or an object identifier. When using an object identifier, the objecttype is applicable and no further specification of the type needs to be included with the object identifier as is done with some other script commands. The following describe the possible objecttypes and identifier options:

- BUS : [busnum]
  ["name_nomkv"]
  ["label"]
- GEN : [busnum id]
  ["name_nomkv" id]
  ["buslabel" id]
  ["label"]
- LOAD : [busnum id]
  ["name_nomkv" id]
  ["buslabel"]
  ["label"]
- BRANCH : [busnum1 busnum2 ckt]
  ["name_kv1" "name_kv2" ckt]
  ["buslabel1" "buslabel2" ckt]
  ["label"]
- SHUNT : [busnum id]
  ["name_nomkv" id]
  ["buslabel" id]
  ["label"]

INJECTIONGROUP: ["name"]
INTERFACE : ["name"]
DCLINE : [num rectnum invnum]
[num "rectnam_nomkv" "invname_nomkv"]
[num "rectlabel" "invlabel"]
["label"]

OnlyEnergizeSpecifiedObjects
: optional parameter, default is NO.
YES : No extra objects in addition to those specified in the filter can be energized. Each object will be evaluated individually.
NO : Extra objects could be energized in addition to those specified if a group of breakers required to energize a specified object also causes other objects to be energized. All objects will be evaluated collectively for determining which objects can be energized, i.e. breakers that cause one object to be energized might also be needed for another object to be energized.

[SwitchingDeviceTypes]
: optional parameter, default is "Breaker". This is a comma-separated list naming the Branch Device Types for switching devices that should be included when determining which devices to close to energize objects. Options include "Breaker" and "Load Break Disconnect".

CloseNormallyClosedDisconnects
: optional parameter, default is NO.
YES : When searching for the specified SwitchingDeviceTypes and a Disconnect is encountered that is open but normally closed, it will be closed and the search for open switching devices will continue past the Disconnect. Additionally, Disconnects that are in series with any open devices of the specified SwitchingDeviceTypes will be closed if they are normally open.
NO : Only switching devices of the specified SwitchingDeviceTypes will be closed.

ExpandAllBusTopology;

This action is used to expand the topology around all buses in the case according to a topology type that is specified with a custom string field (currently Custom String 5) for the bus. New breakers and nodes (buses) will be inserted as necessary.

ExpandBusTopology(BusIdentifier, TopologyType);

This action is used to expand the topology around the specified bus according to the specified topology type. New breakers and nodes (buses) will be inserted as necessary.

BusIdentifier : A bus can be identified in one of these formats: BUS busnum, BUS name_nomkv, BUS label.

TopologyType : These types of breaker configurations are allowed: DOUBLEBUSDOUBLEBREAKER, MAINTRANSFER, RINGBUS, BREAKERANDAHALF, SINGLEBUS, and SECTIONALIZEBUS.

OpenWithBreakers(objecttype, filter or [object identifier], [SwitchingDeviceTypes], OpenNormallyOpenDisconnects);

This action is used to specify which objects are to be disconnected by opening breakers and to actually open those breakers.

objecttype : Objects that are valid to be disconnected. Only allowed for Buses, Generators, Loads, Transmission Lines, Switched Shunts, DC Lines, Injection Groups, and Interfaces.

Filter : The second parameter can either be a filter specification or an object identifier. When specifying a filter, the following options are available:

- SELECTED : only objects whose Selected field = YES will be energized
- AREAZONE : only objects that meet the area/zone/owner filters will be energized
- "FilterName" : only objects that meet the specified filter will be energized. See the Using Filters in Script Commands section for more information on specifying the filtername.

[object identifier] : The second parameter can either be a filter specification or an object identifier. When using an object identifier, the objecttype is applicable and no further specification of the type needs to be included with the object identifier as is done with some other script commands. The following describe the possible objecttypes and identifier options:

BUS	[busnum] ["name_nomkv"] ["label"]
GEN	[busnum id] ["name_nomkv" id] ["buslabel" id] ["label"]
LOAD	[busnum id] ["name_nomkv" id] ["buslabel"] ["label"]
BRANCH	[busnum1 busnum2 ckt] ["name_kv1" "name_kv2" ckt] ["buslabel1" "buslabel2" ckt] ["label"]
SHUNT	[busnum id] ["name_nomkv" id] ["buslabel" id] ["label"]
INJECTIONGROUP	["name"]
INTERFACE	["name"]
DCLINE	[num rectnum invnum] [num "rectnam_nomkv" "invname_nomkv"] [num "rectlabel" "invlabel"] ["label"]

[SwitchingDeviceTypes]

: Optional parameter – default is "Breaker"
This is a comma-separated list naming the Branch Device Types for switching devices that should be included when determining which devices to open to disconnect objects. Options include "Breaker" and "Load Break Disconnect".

OpenNormallyOpenDisconnects
: optional parameter, default is NO.
YES : When searching for the specified SwitchingDeviceTypes and a Disconnect is encountered that is closed but normally open, it will be opened and the search for closed switching devices will terminate along that path.
NO : Only switching devices of the specified SwitchingDeviceTypes will be opened.

SaveConsolidatedCase("filename", filetype, [BusFormat, TruncateCtgLabels, AddCommentsForObjectLabels]);

This action saves the full topology model into a consolidated case.

"filename" : The name of the consolidated case file to be saved.

Filetype : Optional parameter to specifiy the type of the file to be saved. If omitted, the latest version of the PWB will be used.
PWB : save a pwb file with the most recent version
PWBX: save a pwb with version X
PTIXX : save the file with PTI version XX, where XX is between 23 and 33
GEXX : save the file with GE PSLF version XX, where XX is between 14 and 19

BusFormat : optional parameter used to specifiy the bus identifier format in the .CON file used to store contingencies when saving a PTI file
Number : identify buses using number
Name8 : identify buses using the Name_kV identifier truncated to 8 characters
Name12 : identify buses usingthe Name_kV identifier truncated to 12 characters

TruncateCTGLabels : optional parameter used to specify if the contingency labels should be truncated to 12 characters when saving the contingencies in PTI format
YES : truncate the contingency labels to 12 characters
NO : do not truncate the contingency labels

AddCommentsForObjectLabels
: (optional) YES adds object labels to the end of data records when saving a RAW file. (default NO)

OPF (Optimal Power Flow) and SCOPF Related Actions

```
SolvePrimalLP                  ("filename1", "filename2", CreateIfNotFound1, CreateIfNotFound2);
InitializePrimalLP             ("filename1", "filename2", CreateIfNotFound1, CreateIfNotFound2);
SolveSinglePrimalLPOuterLoop   ("filename1", "filename2", CreateIfNotFound1, CreateIfNotFound2);
SolveFullSCOPF                 (BCMethod, "filename1", "filename2", CreateIfNotFound1,
                               CreateIfNotFound2);
OPFWriteResultsAndOptions      ("filename");
```

SolvePrimalLP("filename1", "filename2", CreateIfNotFound1, CreateIfNotFound2);

Call this action to perform a primal LP OPF solution. The parameters are all optional and specify a conditional response depending on whether the solution is successfully found. If parameters are not passed then default values will be used.

"filename1"	:	The filename of the auxiliary file to be loaded if there is a successful solution. You may also specify STOP, which means that all AUX file execution should stop under the condition. Default Value = "".
"filename2"	:	The filename of the auxiliary file to be loaded if there is a NOT successful solution. You may also specify STOP, which means that all AUX file execution should stop under the condition. Default Value = "".
CreateIfNotFound1	:	Set to YES or NO. YES means that objects which cannot be found will be created while reading in DATA sections of filename1. Default Value = NO.
CreateIfNotFound2	:	Set to YES or NO. YES means that objects which cannot be found will be created while reading in DATA sections of filename2. Default Value = NO.

InitializeLP("filename1", "filename2", CreateIfNotFound1, CreateIfNotFound2);

This commands clears all the structures and results of previous primal LP OPF solutions. The parameters are all optional and specify a conditional response depending on whether the solution is successfully found. If parameters are not passed then default values will be used.

"filename1"	:	The filename of the auxiliary file to be loaded if there is a successful solution. You may also specify STOP, which means that all AUX file execution should stop under the condition. Default Value = "".
"filename2"	:	The filename of the auxiliary file to be loaded if there is a NOT successful solution. You may also specify STOP, which means that all AUX file execution should stop under the condition. Default Value = "".
CreateIfNotFound1	:	Set to YES or NO. YES means that objects which cannot be found will be created while reading in DATA sections of filename1. Default Value = NO.
CreateIfNotFound2	:	Set to YES or NO. YES means that objects which cannot be found will be created while reading in DATA sections of filename2. Default Value = NO.

SolveSinglePrimalLPOuterLoop("filename1", "filename2", CreateIfNotFound1, CreateIfNotFound2);

This action is basically identical to the SolvePrimalLP action, except that this will only perform a single optimization. The SolvePrimalLP will iterate between solving the power flow and an optimization until this iteration converges. This action will only solve the optimization routine once, then resolve the power flow once and then stop.

SolveFullSCOPF (BCMethod, "filename1", "filename2", CreateIfNotFound1, CreateIfNotFound2);

Call this action to perform a full Security Constrained OPF solution. The parameters are all optional and specify a conditional response depending on whether the solution is successfully found. If parameters are not passed then default values will be used.

BCMethod	:	The solution method to be used for solving the base case. The options are: POWERFLOW : for single power flow algorithm (default) OPF : for the optimal power flow algorithm.
"filename1"	:	The filename of the auxiliary file to be loaded if there is a successful solution. You may also specify STOP, which means that all AUX file execution should stop under the condition. Default Value = "".
"filename2"	:	The filename of the auxiliary file to be loaded if there is a NOT successful solution. You may also specify STOP, which means that all AUX file execution should stop under the condition. Default Value = "".
CreateIfNotFound1	:	Set to YES or NO. YES means that objects which cannot be found will be created while reading in DATA sections of filename1. Default Value = NO.
CreateIfNotFound2	:	Set to YES or NO. YES means that objects which cannot be found will be created while reading in DATA sections of filename2. Default Value = NO.

OPFWriteResultsAndOptions("filename");

Writes out all information related to OPF analysis as an auxiliary file. This includes Limit Monitoring Settings, options for Areas, Buses, Branches, Interfaces, Generators, SuperAreas, OPF Solution Options.

PV Related Actions

```
PVClear;
PVDataWriteOptionsAndResults  ("filename", AppendFile, KeyField);
PVDestroy;
PVQVTrackSingleBusPerSuperBus;
PVRun                         ([elementSource], [elementSink]);
PVSetSourceAndSink            ([elementSource], [elementSink]);
PVStartOver;
PVWriteInadequateVoltages     ("filename", AppendFile, InadequateType);
PVWriteResultsAndOptions      ("filename", AppendFile);
RefineModel                   (objecttype, filter, Action, Tolerance);
```

Changes were made with Simulator version 14 to eliminate the need for a PV study name. To maintain functionality with any existing processes that users might have in place using older script definitions, scripts from older versions of Simulator will still be supported if the name is specified. However, the name will just be ignored. The script formats given here reflect the changes for versions 14 and later.

The PVCreate script required in previous versions is no longer necessary starting with Simulator version 14. Versions starting with 14 will still recognize this action if is included and will simply set the source and sink for the study. This does the same thing as PVSetSourceAndSink.

It is highly recommended that for any new processes the new script formats specified here be used.

PVClear;

Call the function to clear all the results of the PV study.

PVDataWriteOptionsAndResults("filename", AppendFile, KeyField);

Added in December 16, 2021 patch of Simulator 22

Writes out all information related to PV analysis to an auxiliary file. Saves the same information as the PVWriteResultsAndOptions script command. Auxiliary file is formatted using the concise format for DATA section headers and variable names. Data is written using DATA sections instead of SUBDATA sections.

"filename"	:	Name of the auxiliary file to save. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.
AppendFile	:	Optional parameter - default is YES YES means to append results to existing "filename." NO means to overwrite "filename" with the results.
KeyField	:	Optional parameter – default is PRIMARY Indicates the identifier that should be used for the data. Valid entries are PRIMARY, SECONDARY, or LABEL. PRIMARY will save using bus numbers and other primary key fields. SECONDARY will save using bus name and nominal kV and other secondary fields. LABEL will save using device labels. If no labels are specified then the primary key field will be used.

PVDestroy;

Call the function to destroy the PV study. This will remove all results and prevent any restoration of the initial state that is stored with the PV study.

PVQVTrackSingleBusPerSuperBus;

If the topology processing add-on is installed, then this script command can be used to reduce the number of monitored buses. The script action examines each monitored value for each bus and determines if that bus is part of a super bus and selects monitored buses so that only the pnode is monitored.

PVRun([elementSource], [elementSink]);

Call this function start the PV study and optionally specify the source and sink elements.

[elementSource]	:	Optional parameter – default to using element already set The source of power for the PV study. Only injection groups can be used: [INJECTIONGROUP "name"] or [INJECTIONGROUP "label"]
[elementSink]	:	Optional parameter – default to using element already set The sink of power for the PV study. Only injection groups can be used: [INJECTIONGROUP "name"] or [INJECTIONGROUP "label"]

PVSetSourceAndSink([elementSource], [elementSink]);

Call the function to specify the source and sink elements to perform the PV study.

[elementSource]	:	The source of power for the PV study. There is only one possible setting: [INJECTIONGROUP "name"] or [INJECTIONGROUP "label"]
[elementSink]	:	The sink of power for the PV study. There is only one possible setting, which is the same as for the source.

PVStartOver;

Call the function to start over the PV study. This includes clear the activity log, clear results, restore the initial state, set the current state as initial state, and initialize the step size.

PVWriteInadequateVoltages("filename", AppendFile, InadequateType);

Call this action to save PV Inadequate Voltages in a CSV file.

"filename"	:	Name of the CSV file to save. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.
AppendFile	:	Optional parameter – default is YES Set this to YES or NO. Setting this to YES will cause the data to be appended to an existing file. Setting this to NO will cause any existing file to be overwritten.
InadequateType	:	Optional parameter – default is LOW Set this to HIGH or LOW to indicate the type of inadequate voltages to save to file.

PVWriteResultsAndOptions("filename", AppendFile);

Writes out all information related to PV analysis to an auxiliary file. This includes Contingency Definitions, Remedial Action Definitions, Solution Options, PV Options, PV results, ATC Extra Monitors, as well as any Model Criteria that are used by the Contingency and Remedial Action Definitions.

(Following changes made with the December 9, 2021 patch of Simulator 22)
Contingency and Remedial Action definitions will always be saved along with dependencies and only those object types that are dependencies will be saved. This means that if a Remedial Action definition uses a Model Filter, that Model Filter along with any Model Filter Conditions, i.e. other Model Filters or Model Conditions, will be saved. If a model criteria object is not being used by a Remedial Action or Contingency it will not be saved. Objects that are saved if they are dependencies include: Model Conditions, Model Filters, Model Planes, Model Expressions, Model Result Overrides, Interfaces, Injection Groups, Calculated Fields, and Expressions.

Dependencies for the PV setup are also included. This includes Injection Groups that are used as the seller or buyer and Interfaces that are used as part of interface ramping options.

"filename"	:	Name of the auxiliary file to save. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.
AppendFile	:	Optional parameter – default is YES Set this to YES or NO. Setting this to YES will cause the data to be appended to an existing file. Setting this to NO will cause any existing file to be overwritten.

RefineModel(objecttype, filter, Action, Tolerance);

Call this function to refine the system model to fix modeling idiosyncrasies that cause premature loss of convergence during PV and QV studies.

Objecttype	:	The objecttype being selected: AREA ZONE
Filter	:	Specify a filter to limit the objects that will be included. Blank : Select all objects of specified type AREAZONE : Only objects that meet the area/zone/owner filters will be selected SELECTED : Only objects whose Selected field = YES will be selected "FilterName" : Only objects that meet the specified filter will be selected. See Using Filters in Script Commands section for more information on specifying the filtername.
Action	:	The way the model will be refined. Choices are: TRANSFORMERTAPS : Fix all transformer taps at their present values if their Vmax – Vmin is less than or equal to the user specified tolerance. SHUNTS : Fix all shunts at their present values if their Vmax – Vmin is less than or equal to the user specified tolerance. OFFAVR : Remove units from AVR control, thus locking their MVAR output at its present value if their Qmax – Qmin is less or equal to the user specified tolerance.
Tolerance	:	Tolerance value.

QV Related Actions

```
QVDataWriteOptionsAndResults  ("filename", AppendFile, KeyField);
QVRun                         ("filename",  InErrorMakeBaseSolvable);
QVSelectSingleBusPerSuperBus;
QVWriteCurves                 ("filename", IncludeQuantitiesToTrack, filter, Append);
QVWriteResultsAndOptions      ("filename", AppendFile);
```

QVDataWriteOptionsAndResults("filename", AppendFile, KeyField);

Added in December 16, 2021 patch of Simulator 22

Writes out all information related to QV analysis to an auxiliary file. Saves the same information as the QVWriteResultsAndOptions script command. Auxiliary file is formatted using the concise format for DATA section headers and variable names. Data is written using DATA sections instead of SUBDATA sections.

"filename" : Name of the auxiliary file to save. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.

AppendFile : Optional parameter - default is YES
YES means to append results to existing "filename." NO means to overwrite "filename" with the results.

KeyField : Optional parameter – default is PRIMARY
Indicates the identifier that should be used for the data. Valid entries are PRIMARY, SECONDARY, or LABEL. PRIMARY will save using bus numbers and other primary key fields. SECONDARY will save using bus name and nominal kV and other secondary fields. LABEL will save using device labels. If no labels are specified then the primary key field will be used.

QVRun("filename", InErrorMakeBaseSolvable);

Call the function to start a QV study for the list of buses whose SELECTED field is set to YES.

"filename" : This specifies the file to which to save a comma-delimited version of the results.

InErrorMakeBaseSolvable
: This specifies whether to perform a solvability analysis of the base case if the pre-contingency base case cannot be solved. If not specified, then YES is assumed.

QVSelectSingleBusPerSuperBus;

If the QV tool is being used on a full topology model, this action can be used to modifiy the monitored buses. This action examines the monitored buses and sets the monitored status so that only one bus is monitored for each pnode.

QVWriteCurves("filename", IncludeQuantitiesToTrack, filter, Append);

Added in April 27, 2021 patch of Simulator 22

This will save a comma-separated text file with the QV curve points.

"filename" : Name of the comma-separated file to save. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.

IncludeQuantitiesToTrack : Set this to YES or NO. Set this to YES to also include any Quantities to Track along with the QV curve points.

filter : Specify a filter that is applied to the QVCurve object type. Specifying a blank will select all curve results. See Using Filters in Script Commands section for more information on specifying the filtername. AREAZONE filtering is ignored with the QVCurve object type and will return all results.

Append : Set this to YES or NO. Setting this to YES will cause the data to be appended to an existing file. Setting this to NO will cause any existing file to be overwritten.

QVWriteResultsAndOptions("filename", AppendFile**);**

Writes out all information related to QV analysis to an auxiliary file. This includes Contingency Definitions, Remedial Action Definitions, Solution Options, QV Options, QV results, as well as any Model Criteria that are used by the Contingency and Remedial Action Definitions.

(Following changes made with the December 9, 2021 patch of Simulator 22)

Contingency and Remedial Action definitions will always be saved along with dependencies and only those object types that are dependencies will be saved. This means that if a Remedial Action definition uses a Model Filter, that Model Filter along with any Model Filter Conditions, i.e. other Model Filters or Model Conditions, will be saved. If a model criteria object is not being used by a Remedial Action or Contingency it will not be saved. Objects that are saved if they are dependencies include: Model Conditions, Model Filters, Model Planes, Model Expressions, Model Result Overrides, Interfaces, Injection Groups, Calculated Fields, and Expressions.

"filename" : Name of the auxiliary file to save. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.

Added in the December 16, 2021 patch of Simulator 22

AppendFile : Optional parameter – default is YES
Set this to YES or NO. Setting this to YES will cause the data to be appended to an existing file. Setting this to NO will cause any existing file to be overwritten.

TS (Transient Stability) Related Actions

```
TSAutoCorrect;
TSAutoInsertDistRelay          (Reach, AddRelayAtFromBus, AddRelayAtToBus, DoTransferTrip, Shape,
                               filter);
TSAutoInsertZPOTT              (Reach, filter);
TSCalculateCriticalClearTime   ([branch] or filter);
TSCalculateSMIBEigenValues;
TSClearAllModels;
TSClearResultsFromRAM          (ALL/SELECTED/"ContingencyName", ClearSummary, ClearEvents,
                               ClearStatistics, ClearTimeValues, ClearSolutionDetails);
TSDisableMachineModelNonZeroDerivative(DerivativeThreshold);
TSGetVCurveData                ("FileName", filter);
TSGetResults                   ("FileName", Single/Separate, [contingencies], [plots], StartTime,
                               EndTime);
TSInitialize                   (CheckInitialized);
TSLoadBPA                      ("FileName");
TSLoadGE                       ("FileName", GENCCYN, EnableOutOfOrderModels);
TSLoadPTI                      ("FileName", "MCREfilename", "MTRLOfilename", "GNETfilename",
                               "BASEGENfilename", "MODREMOVEfilename");
TSLoadRDB                      ("FileName", ModelType, filter);
TSLoadRelayCSV                 ("FileName", ModelType, filter);
TSResultStorageSetAll          (objecttype, YES/NO);
TSRunUntilSpecifiedTime        ("ContingencyName", [StopTime, StepSize, StepsInCycles,
                               ResetStartTime, NumberofTimeStepstoDo]);
TSSaveBPA                      ("FileName", DiffCaseModifiedOnly);
TSSaveGE                       ("FileName", DiffCaseModifiedOnly);
TSSavePTI                      ("FileName", DiffCaseModifiedOnly);
TSSaveTwoBusEquivalent         ("AuxFileName",[BUS]);
TSSolve                        ("ContingencyName", [StartTime, StopTime, StepSize,StepInCycles]);
TSSolveAll                     (DoDistributed);
TSTransferStateToPowerFlow;
TSValidate;
TSWriteModels                  ("FileName", DiffCaseModifiedOnly);
TSWriteOptions                 ("FileName",[SaveDynamicModel, SaveStabilityOptions,
                               SaveStabilityEvents, SaveResultsSettings, SavePlotDefinitions],
                               KeyField);
TSSetSelectedForTransientReferences(SetWhat, SetHow, [ObjectType List], [ModelType List]);
TSSaveDynamicModels            ("FileName", FileType, ObjectType, Filter, Append);
```

TSAutoCorrect;

Runs the auto correction of parameters for a transient stability run. If there are still validation errors after running this script that would prevent the stability simulation from running, then the remainder of a script will be aborted.

TSAutoInsertDistRelay(Reach, AddRelayAtFromBus, AddRelayAtToBus, DoTransferTrip, Shape, filter);

Inserts DistRelay models on the lines meeting the specified filter.

Reach	:	Zone 1 reach
AddRelayAtFromBus	:	Add relay at the FROM bus: YES/NO.
AddRelayAtToBus	:	Add relay at the TO bus: YES/NO.
DoTransferTrip	:	Do transfer trip: YES/NO
Shape	:	Shape: 0 - Circle; 1 - Rectangle; 2 - Reactance Distance; 3 - Impedance Distance.
filter	:	Lines meeting this filter will have DistRelay models inserted. See the Using Filters in Script Commands section for information on specifying the filter.

TSAutoInsertZPOTT(Reach, filter,);

Inserts ZPOTT models on the lines meeting the specified filter.

Reach	:	Zone 1 reach
filter	:	Lines meeting this filter will have ZPOTT models inserted. See the Using Filters in Script Commands section for information on specifying the filter.

TSCalculateCriticalClearTime([branch] or filter,);

Use this action to calculate critical clearing time for faults on the lines that meet the specified filter.

[line] or filter	:	A single line can be specified in the format [BRANCH keyfield1 keyfield2 ckt] or [BRANCH label]. Multiple lines can be selected by specifying a filter. See the Using Filters in Script Commands section for information on specifying the filter. For the specified lines, this calculation will determine the first time a violation is reached (critical clearing time), where a violation is determined based on all enabled Transient Limit Monitors. For each line, results are saved as a new Transient Contingency (named CritCTG) on a line, with the fault duration equal to the critical clearing time.

TSCalculateSMIBEigenValues;

Calculate single machine infinite bus eigenvalues. Initialization to the start time is always done before calculating eigenvalues.

TSClearAllModels;

Clears all transient stability models. Load Model Groups are considered part of the case and will not be deleted.

TSClearResultsFromRAM(ALL/SELECTED/"ContingencyName", ClearSummary, ClearEvents, ClearStatistics, ClearTimeValues, ClearSolutionDetails);

Added in the March 15, 2021 patch of Simulator 21

Clears results from RAM of one or more specified Transient Contingencies.

ALL/SELECTED/"ContingencyName"	:	Specifies for which Transient Contingencies to clear results.
ClearSummary	:	(optional) YES to clear, NO to leave them alone.
ClearEvents	:	(optional) YES to clear, NO to leave them alone.
ClearStatistics	:	(optional) YES to clear, NO to leave them alone.
ClearTimeValues	:	(optional) YES to clear, NO to leave them alone.
ClearSolutionDetails	:	(optional) YES to clear, NO to leave them alone.

TSDisableMachineModelNonZeroDerivative(DerivativeThreshold);

Added in October 27, 2022 patch of Simulator 22

The transient stability solution is initialized and state derivatives are calculated. The state derivatives for all generator models are checked, and for any where the absolute value is greater than the DerivativeThreshold, the machine model for the generator is disabled. Some models may have state derivatives where it is valid for the derivatives to initialize to a non-zero value. These states will be excluded from the derivative check.

DerivativeThreshold	:	Optional parameter – default is 0.001 Absolute value of a state's derivative must be greater than this value to disable a model.

TSGetVCurveData("FileName", filter);

For a synchronous generator a curve of points for field current and field voltage is created from a fixed terminal voltage and MW (P) power output and varying Mvar (Q) output.

FileName : Name of file to create. If no file extension is specified this defaults to CSV.

filter : Specifies for which generators to create curves.

- SELECTED : only generators whose Selected field = YES will be included
- AREAZONE : only generators that meet the area/zone/owner filters will be included
- "FilterName" : only generators that meet the specified filter will be included. See the Using Filters in Script Commands section for more information on specifying the filtername.

TSGetResults("FileName", SINGLE/SEPARATE/JSIS, [Contingencies], [Plots, ObjectFields], StartTime, EndTime]);

Use this to save out results for specific variables from plots, subplots, and object/field pairs after a transient stability simulation has been run. If StartTime and StopTime are not specified, results for the entire simulation time are obtained.

FileName : Name of the CSV result file to write out

SINGLE/SEPARATE/JSIS : Determines whether the results are all saved in one file (SINGLE) with name "filename" or whether results for each transient contingency is saved in a separate file (SEPARATE) with name "filename_ctgname.csv." A separate header file is also saved out, with a name of "filename_Header.csv". If using the JSIS format, a single file with the name "filename" is written in the WECC JSIS format.

Contingencies : A list of contingency names for which to save out results

Plots, ObjectFields : A list of plots and object/field pairs to save out for the specified contingencies. For plots, write PLOT and then the name of the plot. For ObejctsFields you need the Obejct name followed by the key fields and a vertical line (|), then the field.

StartTime : Start of the window of simulation time from which the results are to be retrieved

EndTime : End of the window of simulation time from which the results are to be retrieved

```
TSGetResults("D:\Cases\Scripts\Test.CSV",SINGLE,["My Transient Contingency"], ["PLOT
Gen_Rotor Angle","Bus 4 | frequency","PLOT Bus_Frequency","GEN 1  '1'|
TSGenMachineState:2"],0,10);
```

TSInitialize(CheckInitialized);

Added in June 21, 2022 patch of Simulator 22

Initialize the transient stability solution. This is useful to calculate the initial state and state derivative values. It is not necessary to call this step manually as it will be done automatically with any other transient analysis, but it is useful if analysis of the initial states is desired without doing any additional calculations. Any validation errors must be corrected before initialization can be done.

CheckInitialized : Optional parameter – default is NO
Set to YES to check if the transient solution has already been initialized and not to initialize again if it has. Set to NO to initialize whether or not the transient solution has already been initialized.

TSLoadBPA("FileName");

Loads transient stability data stored in the BPA format.

FileName : Name of the BPA file to load

TSLoadGE("FileName", GENCCYN, EnableOutOfOrderModels);

Loads transient stability data stored in the GE DYD format.

FileName : Name of the DYD file to load
GENCCYN : YES to split combined cycle units, NO to leave them alone
EnableOutOfOrderModels
: (optional) Default is YES. If set to YES, models that are specified out of order in the file will be enabled. If set to NO, out of order models will be disabled.

TSLoadPTI("FileName", "MCREfilename", "MTRLOfilename", "GNETfilename", "BASEGENfilename", "MODREMOVEfilename);

Loads transient stability data in the PTI format.

FileName : Name of the DYR file to load
MCREfilename : (optional) If not loading a MCRE file, specify ""
MTRLOfilename : (optional) If not loading a MTRLO file, specify ""
GNETfilename : (optional) If not loading a GNET file, specify ""
BASEGENfilename : (optional) If not loading a BASEGEN file, specify ""
MODREMOVEfilename : (optional) if not loading a MODREMOVE file, specify ""

TSLoadRDB("filename", ModelType, filter);

Loads a SEL RDB file. The RDB file is a Schweitzer format for describing a relay. This command will load the file and translate the relay settings into a PowerWorld transient stability model. An attempt is made to match up the protected lines in the power flow model with the SID field in the RDB data. A SID in the format "FROM SUB/BREAKERS/TO SUB" is expected. If the SID does not match this format, or a match is not found in the case, the objects can be linked to the transient stability model in the user interface manually.

"filename" : The directory path or filename of the RDB file(s). If pointed to a directory, the script command will load every RDB file in that directory. If pointed to a file, the script command will load the single RDB file. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.
ModelType : One of the following:
DISTRELAY : create a Simulator DistRelay model from the RDB data
ZPOTT : create a Simulator ZPOTT model from the RDB data
filter : Optional parameter
Lines meeting this filter are searched to find ones matching the SID. See the Using Filters in Script Commands section for information on specifying the filter.

TSLoadRelayCSV("filename", ModelType, filter);

This is a quicker alternative to using the TSLoadRDB command for loading RDB files. Relevant relay data can be exported to a CSV file so that a single CSV file contains multiple relay models. This is much faster because it does not contain the unused data that an RDB file contains.

"filename"	:	Name of the CSV file to load
ModelType	:	One of the following: DISTRELAY : create a Simulator DistRelay model from the RDB data ZPOTT : create a Simulator ZPOTT model from the RDB data
filter	:	Optional parameter Lines meeting this filter are searched to find ones matching the SID. See the Using Filters in Script Commands section for information on specifying the filter.

TSResultStorageSetAll(objecttype, YES/NO);

This command will allow setting which object types are stored in memory during a transient stability run. This will affect all fields and states for the specified objecttype.

objecttype	:	Specifies which objects to set. The objecttype is the object name of supported objects such as GEN, BUS, BRANCH, etc. ALL can be used to set all supported object types.
YES/NO	:	Using this command will toggle all the "Save All" fields to YES/NO. It will also toggle all the "state" fields (such as exciter, machine, governor, etc.) to YES/NO.

TSRunUntilSpecifiedTime("ContingencyName", [StopTime, StepSize, StepsInCycles, ResetStartTime, NumberOfTimeStepsToDo]);

This command allows manual control of the transient stability run. The simulation can be run until a specified time or number of times steps and then paused for further evaluation.

ContingencyName	:	The name of the contingency to solve.
StopTime	:	(optional) This is the time to which the simulation will be run. This should be entered in seconds. If NumberOfTimeStepsToDo > 0, this field will be ignored. If not specified, the stop time specified with the contingency will be used.
StepSize	:	(optional) Simulation step size in either seconds or cycles. If StepsInCycles = YES this should be specified in cycles. If not specified, the step size specified with the contingency will be used.
StepsInCycles	:	(optional) Set to YES to specify StepSize in cycles. If not specified, the units of the step size specified with the contingency will be used.
ResetStartTime	:	(optional) Set to YES to reset the simulation start time. Default value is NO.
NumberOfTimeStepsToDo	:	(optional) Number of time steps to run. If NumberOfTimeStepsToDo > 0, StopTime is ignored. Default value is 0.

TSSaveBPA("FileName", DiffCaseModifiedOnly);

Save transient stability data stored in the BPA IPF format.

FileName	:	Name and path for the output file. Typically this will be an *.swi file extension.
DiffCaseModifiedOnly	:	(optional) Default is NO. When set to YES, it will only save models that are either new or models which have had a parameter modified as compared to the difference case tool base case.

TSSaveGE("FileName", DiffCaseModifiedOnly);

Save transient stability data stored in the GE DYD format.

FileName : Name and path for the output file. Typically this will be an *.dyd file extension.

DiffCaseModifiedOnly : (optional) Default is NO. When set to YES, it will only save models that are either new or models which have had a parameter modified as compared to the difference case tool base case.

TSSavePTI("FileName", DiffCaseModifiedOnly);

Save transient stability data stored in the PTI DYR format.

FileName : Name and path for the output file. Typically this will be an *.dyr file extension.

DiffCaseModifiedOnly : (optional) Default is NO. When set to YES, it will only save models that are either new or models which have had a parameter modified as compared to the difference case tool base case.

TSSaveTwoBusEquivalent ("AuxFileName", [BUS]);

Save the two bus equivalent model of a specified bus to a PWB file. Initialization to the start time is always done before saving the two bus equivalent.

AuxFileName : Name and path for the output AUX file

BUS : Bus can be specified in three ways:

- Number : [BUS busnum]
- Name/NomkV : [BUS "busname_nominalKV"]
- Label : [BUS "buslabel"]

TSSolve("ContingencyName", [StartTime, StopTime, StepSize, StepInCycles]);

Solves only the specified contingency.

ContingencyName : The name of the contingency to solve

Times : (optional) Pararmer is a comma delimited list of strings enclosed in square brackets. This string then consists of 4 optional parameters which override the corresponding property of the contingency.

- StartTime : (optional) Start time in seconds
- StopTime : (optional) Stop time in seconds
- StepSize : (optional) Step size (in seconds unless StepInCycles = YES)
- StepInCycles : (optional) Set to YES to mean that the StepSize is given in cycles

TSSolveAll(DoDistributed);

Solves all defined transient contingencies that are not set to skip.

DoDistributed : (optional) Set to YES to use Distributed Computing with the transient analysis. Default is NO.

TSTransferStateToPowerFlow;

Added in November 16, 2022 patch of Simulator 22

Transfer the transient stability state to the power flow.

TSValidate;

Added in June 21, 2022 patch of Simulator 22

Validate transient stability models and input values. This command is useful for examining model errors and warnings when preparing a case for analysis. Validation will be done automatically when running transient analysis, so this command does not need to be run manually prior to analysis.

TSWriteModels("FileName", DiffCaseModifiedOnly);

Save transient stability dynamic model records only the auxiliary file format.

FileName	:	Name and path for the output file. Typically this will be an *.aux file extension.
DiffCaseModifiedOnly	:	(optional) Default is NO. When set to YES, it will only save models that are either new or models which have had a parameter modified as compared to the difference case tool base case.

TSWriteOptions("FileName",[SaveDynamicModel, SaveStabilityOptions, SaveStabilityEvents, SaveResultsEvents, SavePlotDefinitions, SaveTransientLimitMonitors, SaveResultAnalyzerTimeWindow], KeyField);

Save the transient stability option settings to an auxiliary file.

FileName	:	Name and path of the file to save
SaveDynamicModel	:	(optional) NO doesn't save dynamic model (default YES)
SaveStabilityOptions	:	(optional) NO doesn't save stability options (default YES)
SaveStabilityEvents	:	(optional) NO doesn't save stability events (default YES)
SaveResultsSettings	:	(optional) NO doesn't save results settings (default YES)
SavePlotDefinitions	:	(optional) NO doesn't save plot definitions (default YES)
SaveTransientLimitMonitors	:	(optional) NO doesn't save transient limit monitors (default YES)
SaveResultAnalyzerTimeWindows	:	(optional) NO doesn't save result analyzer time windows (default YES)
KeyField	:	(optional) Specifies key: can be Primary, Secondary, or Label (default Primary)

TSSetSelectedForTransientReferences(SetWhat, SetHow, [ObjectType List],[ModelType List]);

Set the Custom Integer field to a corresponding integer or Selected field to Yes/No for objects referenced in a transient stability model with extra objects.

SetWhat	:	Either Selected or Custom Integer number (1, 2, 3…). Default is number 1 for the Custom Integer
SetHow	:	when SetWhat = Selected, will be either YES or NO. Otherwise it will be an integer to be used with Custom Integer. Default is 1.
[ObjectType List]	:	A comma-delimited list of objecttypes. These are the objects on which the custom field will be set.
[ModelType List]	:	A comma-delimited list of transient stability model types on which references will be queried.

TSSaveDynamicModels("FileName", FileType, ObjectType, Filter, Append);

Save dynamics models for specified object types to file.

FileName	:	Name and path of the file to save
FileType	:	AUX or DYD.
ObjectType	:	The object type to save - Gen, Load, SwitchedShunt, DCLine, VSCDCLine, Branch, among others available in Transient Stability..
Filter	:	See Using Filters in Script Commands section for more information on specifying the filter.
Append	:	YES or NO to whether replace an existing file or append to it.

Scheduled Actions Related Actions

```
IdentifyBreakersForScheduledActions(IdentifyFromNormalStatus);
SetScheduleView              (ViewTime, ApplyActions, UseNormalStatus, ApplyWindow);
SetScheduleWindow            (StartTime, EndTime, Resolution, ResolutionUnits);
```

IdentifyBreakersForScheduledActions(IdentifyFromNormalStatus);

For each Scheduled Outage, identifies breakers that are necessary to implement OpenBreakers and CloseBreakers actions. New Scheduled Actions are added to the Scheduled Outage if new breakers are identified that do not already exist as actions.

IdentifyFromNormalStatus : Set to YES to return all branches to their normal status before searching for breakers. If using this option, all branches will be returned to their current status at the end of the process. Set to NO to leave all branches at their current status before searching for breakers.

SetScheduleView(ViewTime, ApplyActions, UseNormalStatus, ApplyWindow);

Sets the View Time for Scheduled Actions.

ViewTime : The desired view time. Must be between the currently configured Start and End times, and fall on a valid view point given current resolution settings.

ApplyActions : (optional) Set to YES or NO to override current "Apply Actions" settings to either apply actions at this view time or suppress the application of actions at this time. If left blank, current "Apply Actions" settings will be used.

UseNormalStatus : (optional) Set to YES or NO to override current "Use Normal Status" settings to either restore devices to their Normal Status after they are restored, or to use whatever status they had before the action was applied. If left blank, current "Use Normal Status" settings will be used.

ApplyWindow : (optional) Set to YES or NO to override current "Apply Window" settings to either apply any actions active in the window starting at this View Time and extending forward one Resolution step forward in time, or only apply actions active at the specific View Time specified. If left blank, current "Apply Window" settings will be used.

SetScheduleWindow(StartTime, EndTime, Resolution, ResolutionUnits);

Defines the window of interest for Scheduled Actions.

StartTime : Defines the start of the window.

EndTime : Defines the end of the window. Must fall on a valid time relative to the Start Time as defined by the Resolution (e.g. if the Start Time is 5/20/17 8:00 and the resolution is 1 DAY, the End Time must be at 8:00 as well on a later date.)

Resolution : (optional) Decimal value defining the time step between the Start Time and any valid View Time.

ResolutionUnits : (optional but must be specified if Resolution is specified) May be MINUTES, HOURS, or DAYS

Trainer Related Actions

```
ResetStatusChangeCount;
SuppressCommandDialog;
```

ResetStatusChangeCount;

There is a field with each branch called Status Change Count that tracks the number of times a branch changes status. This script action sets this count to 0 for all branches. This field is only used with Trainer.

SuppressCommandDialog;

Tells trainer to only send status change commands instead of opening a dialog, which in the case of generators and other objects has additional options.

DATA Section

Concise Auxiliary File Header

PowerWorld defaults to using a concise header that starts with the `object_type` string followed by the `list_of_fields` as the argument between the parentheses.

```
object_type DataName(list_of_fields)
{
data_list_1
    .
    .
    .
data_list_n
}
```

Simulator Version 19 and later support reading both this syntax and the older Legacy header. Older versions of Simulator only support the Legacy Auxiliary File Header below. All versions of Simulator support reading the legacy header format. When writing out an auxiliary file there is an option starting in Simulator Version 19 called Use Concise Variable Names and Auxiliary File Headers that determines whether to write out the DATA keyword and other arguments or to write out the concise auxiliary file header. See the online help documentation for more details: http://www.powerworld.com/WebHelp/#MainDocumentation_HTML/Auxiliary_Files.htm

When using the concise format, the data between the curly braces is always written using space delimiter and the Create_if_not_found is always assumed to be YES.

Legacy Auxiliary File Header

```
DATA DataName(object_type, [list_of_fields], file_type_specifier, create_if_not_found)
{
data_list_1
    .
    .
    .
data_list_n
}
```

Immediately following the DATA keyword, you may optionally include a DataName. By including the DataName, you can make use of the script command LoadData("filename", DataName) to call this particular data section from another auxiliary file. Following the optional DataName is an argument list. The argument list is contained inside left and right parentheses "( )". There are 4 arguments in this list which will be described shortly: `object_type`, `[list_of_fields]`, `file_type_specifier`, and `create_if_not_found`.

ObjectType

The `object_type` parameter identifies the type of object or data element the information section describes or models. For example, if `object_type` equals `BUS`, then the data describes BUS objects.

There are some special object types that start with the keyword `REMOVED`. If these are loaded into Simulator while in Edit mode, the corresponding objects will be deleted. For example `REMOVEDBUS` will delete `BUS` objects, `REMOVEDBRANCH` will delete `BRANCH` objects, etc. Not all object types have a corresponding `REMOVED` object type, and simply prepending this keyword to the front of an `object_type` will not allow this functionality. The objects that exist of this with this functionality are the ones that allow comparison of topological changes through the Difference Flows tool.

The list of object types Simulator's auxiliary file parser can recognize will grow as new applications are developed. Within Simulator, you will always be able to obtain a list of the available object types by going to the main menu and choosing Window, Export Case Object Fields, and then exporting the objects to Excel or a text file.

File_Type_Specifier

When using the Legacy Auxiliary File Header, the `file_type_specifier` parameter distinguishes the information section as containing custom auxiliary data (as opposed to Simulator's native auxiliary formats), and indicates the format of the data. The parser recognizes two values for `file_type_specifier`:

`(blank)` or `AUXDEF` or `DEF`	Data fields are space delimited
`AUXCSV` or `CSV` or `CSVAUX`	Data fields are comma delimited

Required Fields and Create_if_not_found

Each ObjectType in PowerWorld has a set of Required Fields that must be included in the List_of_Fields when using an Auxiliary File to *create a new object*. These Required Fields are highlighted in green in the user interface. If these Required Fields are not included in the List_of_Fields, then new objects will never be created, and the Auxiliary file will only edit existing objects. When using the Concise Auxiliary File Header, if the required fields are given, then new objects will be created. There are also some restrictions for objects that defined the network topology and these objects may only be created while in Run Mode: Bus, Gen, Load, Shunt, Branch, LineShunt, DCTransmissionLine, VSCDCLine, MSLine, 3WXFormer, MTDCRecord, MTDCBus, MTDCLine, and MTDCConverter.

In addition to specifying these fields, when using the Legacy Auxiliary File Header, an optional field `Create_if_not_found` must be specified. In the Legacy header, objects will only be created if all Required Fields are given and the `Create_if_not_found=YES`. If the value is NO, objects will not be created. If `Create_if_not_found` is omitted in the Legacy header a prompt dialog will appear when loading the file asking the user what to do if an object is encountered that does not exist. If loading an auxiliary file using the `LoadAux` script command, the `Create_if_not_found` field for the data section will override the `Create_if_not_found` field with the script.
To repeat, regardless of the choice of `CreateIfNotFound`, a new object will only be created if a particular list of Required Fields is in the List_of_Fields.

List_of_Fields

The `list_of_fields` parameter lists the types of values the ensuing records in the data section contain. The order in which the fields are listed in `list_of_fields` dictates the order in which the fields will be read from the file. Simulator currently recognizes many different field types, each identified by a specific field variable name. Because the available fields for an object may grow as new applications are developed for the convenience of our customers, you will always be able to obtain a list of the available object types and fields by going to the main menu and choosing Window, Export Object Fields, and then choosing to export to Excel or a text file. Certainly, only a subset of these fields would be found in a typical custom auxiliary file. In crafting applications to export custom auxiliary files, developers need concern themselves only with the fields they need to communicate between their applications and Simulator. A few points of interest regarding the `list_of_fields` are:

- The `list_of_fields` may take up several lines of the text file.
- When using the older heading starting with the keyword DATA, the `list_of_fields` should be enclosed by square brackets [ ]. When using the concise heading these square brackets are not used.
- When encountering the PowerWorld comment string '//' in one of these lines of the text file, all text to the right is ignored.
- Blank lines, or lines whose first characters are '//' will be ignored as comments.
- Field variable names must be separated by commas.

Example: the following are equivalent representations

```
DATA (BUS, [NomKV, Number,  // comment here
   VAngleA,  VAngleB,  VpuA,  VpuB,
   // comments allowed here to

   // note that blank rows are ignored
    AreaNum,  VAngle,  ActB,  Equiv,  ActG,
    kV,  MargCostMW,  LoadMVA, // more comment
    LoadMW,  LongName])
```

```
BUS (NomKV, Number,  // comment here
   VAngleA,  VAngleB,  VpuA,  VpuB,
   // comments allowed here to

   // note that blank rows are ignored
    AreaNum,  VAngle,  ActB,  Equiv,  ActG,
    kV,  MargCostMW,  LoadMVA, // more comment
    LoadMW,  LongName)
```

Concise Field Variable Names

Variable names within Simulator were overhauled starting with Version 19. Most no longer utilize the special location integer and instead spell out such information in the field variable name. In general, the variable names have been made more concise or at least more understandable. Therefore what was once called for a BRANCH object LineMW:1 is now called MWTo. Similarly LineMW:2 is now called MWFromCalc (representing the MW flow at the from bus of branch calculated from the terminal voltages). The only fields that continue to use the location integer are those that represent fields for which a dynamic number of fields are available. Examples of this include the CustomInteger, CustomString, and CustomFloat fields which use the location integer to specify which value is used. Other examples include the multiple direction PTDF results fields PTDFMult:0, PTDFMult:1, and so on.

When writing out an auxiliary file there is an option within Simulator Version 19 called Use Concise Variable Names and Auxiliary File Headers that determines whether to write out the DATA keyword and other arguments or to write out the concise auxiliary file header. See the online help documentation for more details:
http://www.powerworld.com/WebHelp/#MainDocumentation_HTML/Auxiliary_Files.htm

Legacy Field Variable Naming

When listing fields, some field variable names may be augmented with a field location. These are in the format `variablename:location`. One example of this is the field LineMW. For a branch, there are two MW flows associated with the line: one MW flow at the from bus, and one MW flow at the to bus. So that the number of fields does not become huge, the same field variable name is used for both of these values. For the from bus flow, we write LineMW:0, and for the to bus flow, we write LineMW:1. Note that field variable names using a location of 0, such as LineMW:0, may simply leave off the :0.

These field variable names have been updated with Concise Field Variable Names starting in Simulator version 19. These are described in the next section. The Legacy field variable names can still be used to support existing auxiliary files or files needed for loading into earlier versions of Simulator.

Special Naming

There are several fields that can be referred to by the user-defined name for the field rather than using the location number. These are fields that might have their location numbers change when different auxiliary files are merged in the same case. Referring to these by name can eliminate this possible confusion. These fields can be defined in the format `variablename:location_by_name`. They can also be referred to by location number as well.

Fields that allow referring to the location by name are:

- Expressions – "CustomExpression:`my expression name`"
- String Expressions – "CustomExpressionStr:`my string expression name`"
- Custom fields (Floating Point, Integer, and String) – "CustomSingle:`my custom single name`". Using this format for custom fields requires that Custom Field Descriptions be created for the fields to be used.
- Calculated Fields – "BGCalcField:`my calculated field variable name`"

Key Fields

Simulator uses certain fields to identify the specific object being described. These fields are called key fields. For example, the key field for BUS objects is `BusNum`, because a bus can be identified uniquely by its number. The key fields for GEN objects are `BusNum` and `GenID`. To properly identify each object, the object's key fields must be present. They can appear in any order in the `list_of_fields` (i.e. they need not be the first fields listed in list_of_fields). As long as the key fields are present, Simulator can identify the specific object. By going to the main menu and choosing Window and then Export Case Object Fields you will obtain a list of fields available for each object type in either Excel or text format. In this output, the key fields will appear with asterisks *.

Data List

After the data argument list is completed, the Data list is given. The data section lists the values of the fields for each object in the order specified in `list_of_fields`. The data section begins with a left curly brace and ends with the a right curly brace. A few points of interest regarding the `value_list`:

- The `value_list` may take up several lines of the text file.
- Each new data object must start on its own line of text.
- When encountering the PowerWorld comment string '//' in one of these lines of the text file, all text to the right of this is ignored.
- Blank lines, or lines whose first characters are '//' will be ignored as comments.
- Remember that the right curly brace must appear on its own line at the end of the `data_list`.
- If the `file_type_specifier` is CSV, the values should be separated by commas. Otherwise, separate the field variable names using spaces.
- Strings can be enclosed in double quotes, but this is not required. You should however always inclose strings that contain spaces (or commas) in quotes. Otherwise, strings containing commas would cause errors for comma-delimited files, and spaces would cause errors for space-delimited formatted files.

Special Data List Entries

When specifying values in case information displays, AUX files, and script commands, there are some special formats that can be used. These formats will allow the value of a model or object field to be used instead of specifying an explicit value. Formats that are allowed include those described in the **Special Identifiers for Model Fields in Data** section as well as others mentioned below in this section.

A string of the format `"@Variablenamelegacy:location:digits:decimals"`
Or `"@concisename:location:digits:decimals"`
will be treated as though the value of the named variable is entered in the field, with `digits` total and `decimals` digits to the right of the decimal point. This format may be used in case information displays, AUX files, and script commands that set values. String fields that can be converted to a valid numeric value can be used to populate either floating point or integer fields. Floating point values that are used to populate integer fields are truncated before populating the integer field.

When parsing an AUX file while reading, the treatment of concise and legacy variable names is automatically handled by the parser.

There are specific fields for specific object types that allow referencing a model field definition as the value of the field. The model field definition is specified in this syntax: @MODELFIELD<objecttype 'key1' 'key2' variablename:digits:rod>. The model field definition and not the value of the model field itself is used as the value of the field until the field is used for its intended purpose. The fields that can be specified in this manner include those that specify directory paths and file locations. When an action is carried out that requires using the directory path or file location, the model field definition is converted to the value that it represents and that gets used as the location of the file or directory path.

The following object types and fields (given by variable name) can use this special model field syntax: Transient_Options (RSHD_Directory), PVCurve_Options (PVCOutFile, PVCStoreStatesWhere), QVCurve_Options (QVOutputDir, QVOutputFileName), CTG_Options (PostCTGSolAuxFile, PostCTGAuxFile, HardDriveFileName), Sim_Environment_Options (SEOSpecifiedAuxFile:0, SEOSpecifiedAuxFile:1, SEOSpecifiedAuxFile:2) and MessLog_Options (LogAutoFileName).

Special Identifiers for Model Fields in Data

The following special formats can be used in case information displays, AUX files, and script commands that set values. They are also used in some script commands as part of parameters that input text. These formats will allow the value of a model or object field to be used instead of specifying an explicit value.

A string in the format `"&ModelExpressionName:digits:decimals"` will be treated as though the value of the named model expression is entered in the field, with `digits` total and `decimals` digits to the right of the decimal point. If no digits or decimals are specified, 7 decimal places will be used. Trailing zeros will be removed if no decimals are specified.

A string of the format `"&Objecttype 'key fields' variablenamelegacy:location:digits:decimals"`
Or `"&Objecttype 'key fields' concisename:digits:decimals"`
will be treated as though the value of the named object and object field is entered in the field, with `digits` total and `decimals` digits to the right of the decimal point. If no digits or decimals are specified, 7 decimal places will be used. Trailing zeros will be removed if no decimals are specified.

Using Labels for Identification

Most data objects (such as buses, generators, loads, switched shunts, transmission lines, areas, zones, and interfaces) may have an alternative names assigned to them. These alternative names are called labels. Labels allow you to refer to equipment in the model in a way that may be unique to your organization. Labels may thus help clarify which elements are described by a particular set of data, especially when the short names employed by the power system model prove cryptic. Furthermore, since labels are likely to change less frequently than bus numbers, and since a label must, by definition, identify only one power system component, they may function as an immutable key for importing data from auxiliary files into different cases, even when bus numbering schemes change between the cases. Labels must be unique for devices of the same type, but the same label can be used for a device of a different type.

Information dialogs corresponding to buses, generators, loads, switched shunts, transmission lines, areas, zones, and interfaces feature a button called Labels. If you press this button, the device's Label Manager Dialog will appear. The Label Manager Dialog lists the labels associated with the device. You can delete a label from the list by selecting it and pressing the delete key on the keyboard or clicking the Delete button. You may add a label to the device by typing its name in the textbox and pressing the Add New button. You will not be allowed to add a Label that already exists for the same type of device. A single power system device may have multiple labels, but each label may be associated with only one device of a given type. For example, a bus could have the label Bus North while a generator could also have the same label, but there could not be another bus or generator with this same label.

You also may designate a particular label to be the primary label for the device by checking the box Primary before adding the label. Alternatively, you can select the device from the list and click the Make Primary button. A device's primary label is the one that is listed first in the Labels (All) field (`variablename = LabelsAll`) in a Case Information Display. This field lists all labels assigned to a device as a comma-delimited string. Any label can be used to import data from auxiliary data files.

Labels can be used to map data from an auxiliary data file to a power system device. Recall that auxiliary data files require you to include a device's key fields in each data record so that data may be mapped to the device. Labels provide an alternative key. Instead of supplying the bus number to identify a bus, for example, you can supply one of the bus's labels. The label will enable Simulator to associate the data with the device associated with that label. This mechanism performs most efficiently when the primary label is used, but other labels will also provide the mapping mechanism. The Label (for use in input from AUX or Paste) field (`variablename = Label`) is used for importing data using labels and is blank when viewing in a case information display. Keep in mind that all devices read via an auxiliary file using the label field should have a non-blank label. Otherwise, information for that device will not be read. Even if the primary or secondary key fields are provided with the device, as long as the label field is present, that is the only field that will be used to identify the device. New devices cannot be created by simply identifying them by label. Either the primary or secondary key fields must be present to create a new device and the label field should not be present.
Again, it is important to remember this: a single power system device may have multiple labels, but each label may be associated with only one device of a particular type. This is the key to enabling data to be imported from an auxiliary file using labels.

Saving Auxiliary Files Using Labels

All devices that can be identified by labels will have the Labels (All) and Label (for use in input from AUX or Paste) fields available in their case information displays. In order to save auxiliary files that identify devices by label, the two label fields should be added to the case information display prior to saving the data in an auxiliary file. Because the Label (for use in input from AUX or Paste) field will be blank when saved in the auxiliary file, this field must be populated with one of the labels in the Labels (All) field before loading the auxiliary file back in. Keep in mind that devices with blank labels cannot be identified when loading in an auxiliary file, so avoid saving auxiliary files by label if all devices do not have labels. Note that when saving out an entire case as an auxiliary file, the field "AllLabels" is included for each object type that allows labels and has some labels defined.

Many devices require SUBDATA sections. These sections have custom formats specific to the type of information that they contain. When saving auxiliary files with devices that require SUBDATA sections, the user can choose to use primary or secondary key fields or labels to identify devices in the SUBDATA sections. The user will either be prompted when saving the devices, or there is an option to change the key field to use when saving subdata sections on the PowerWorld Simulator Options dialog under the Case Information Displays category. When choosing to use labels, if a device has a label, it will be used. If it is a device that can be identified by buses and bus labels exist, bus labels will be used. Finally, if the device does not have a label and the buses do not have labels, the primary key for the device will be used for identification.

Devices that have SUBDATA sections that contain other devices that can be identified by labels include: contingencies, interfaces, injection groups, post power flow solution actions, and owners.

The setting to choose which identifier to use for the SUBDATA sections does not just apply to SUBDATA sections. Often when saving groups of options, this setting will apply to everything being saved with those options and not just the SUBDATA sections. This includes contingency options, ATC options, limit monitoring settings, and PVQV options. In these cases, there will be a prompt asking the user to decide which identifier to use in the auxiliary file.

Loading Auxiliary Files SUBDATA Sections Using Labels

The various SUBDATA sections that represent references to other objects can also be read using labels. Examples include contingencies, interfaces, injection groups, post power flow solution actions, and owners. When reading a SUBDATA section such as this, PowerWorld makes no assumption ahead of time about what identification was used to write this SUBDATA section. Instead, an order of precedence for the identification is as follows

	Identification	Explanation	Example
1	Key Fields	assumes that the strings represent Key Fields	`BRANCH 8 9 1`
2	Secondary Key Fields	assumes that the strings represent Secondary Key Fields	`BRANCH Eight_138 Nine_230 1`
3	Labels for component objects	the key/secondary key fields for some objects consist of references to other objects. An example of this is the BRANCH object that is described by the From Bus, To Bus, and Circuit ID. This assumes that labels of the component objects are used.	`BRANCH Label8 Label9 1`
4	Labels	Assumes that the string represents one of the Labels of the object	`BRANCH LabelForBranch`

Special Use of Labels in SUBDATA

There are a few special cases where objects have fields that identify other devices. These devices can be identified by label but not in the conventional means because the label field applies to the object that contains the device and a SUBDATA section is not necessary. These special cases include: (Note all fields given below are by variable name because the use of labels is most relevant with auxiliary files.)

ATC Scenarios

ATC Scenario change records usually contain primary key fields to identify the devices that should be adjusted during the scenario. If using labels, these primary key fields will be replaced with a single Label field. The use of this field is different because the Label field refers to the device in the change record and not to the change record itself. When labels are used with ATC scenarios, device labels only can be used. Bus labels cannot be used to identify devices for which no label exists but a bus label does.

ATC Extra Monitors

ATC Extra Monitors identify either branches or interfaces to monitor during the ATC analysis. These devices are identified in the WhoAmI field of ATC Extra Monitor records. Usually, the WhoAmI field is a special format that contains key field tags. Optionally, this field can use the label of the device for the extra monitor. If the device label is not available, the standard format will be used. There is no option to use bus labels if they exist and the device labels do not.

Model Conditions

Devices in Model Conditions are usually identified by the WhoAmI field which is in a special format that contains key field tags. Optionally, this field can use the label of the device. If the device label does not exist, the standard format will be used. There is no option to use bus labels if they exist and the device labels do not.

Model Expressions

Model Expressions contain Model Fields. Model Fields are identified by the WhoAmI fields in the Model Expressions. Usually, the WhoAmI fields are in a special format that contains key field tags. Optionally, these fields can use the label of the device associated with the Model Field. If the device does not exist, the standard format will be used. There is no option to use bus labels if they exist and the device labels do not.

Bus Load Throw Over Records

Bus Load Throw Over Records are used with contingency analysis. These records have an option to identify the bus to which the load will be transferred by either number or name_kV combination. If choosing to identify objects by label, the BusName_NomVolt:1 field will contain the label of the bus instead of the name_kV combination. Bus Load Throw Over Records will be saved in an auxiliary file if choosing to Save settings on the Contingency Analysis dialog.

Injection Group Participation Points

All participation points and the injection groups to which they belong can be listed on the Injection Group Display. Load, generator, bus, and shunt devices that can be assigned to a participation point must be identified by bus and ID. The bus can be identified by either the number or name. When identifying by name, the BusName_NomVolt field is used to provide the name_kV combination for the bus. If choosing to identify devices by label, this field instead will contain the label of the device. If the device does not have a label but the bus does, the bus label will be used instead in conjunction with the ID of the device. Even if the device does contain a label, the ID field must be included in any auxiliary file that is going to be loaded because it is a key field. Injection groups can be included in other injection groups. Injection groups can be identified by label, even though this is not a normal thing to do. If any injection groups have labels and these injection groups are included in other injection groups, their labels will also appear in the BusName_NomVolt field. If they do not have labels, they will be identified by the injection group name that appears in the PPntID field.

SubData Sections

The format described thus far works well for most kinds of data in Simulator. It does not work as well however for data that stores a list of objects. For example, a contingency stores some information about itself (such as its name), and then a list of contingency elements, and possible a list of limit violations as well. For data such as this, Simulator allows `<SubData>`, `</SubData>` tags that store lists of information about a particular object. This formatting looks like the following

```
object_type (list_of_fields)
{
value_list_1
    <SUBDATA subobject_type1>
      precise format describing an object_type1
      precise format describing an object_type1
      .
      .
      .
    </SUBDATA>
    <SUBDATA subobject_type2>
      precise format describing an object_type2
      precise format describing an object_type2
      .
      .
      .
    </SUBDATA>
value_list_2
   .
   .
   .
value_list_n
}
```

Note that the information contained inside the `<SubData>`, `</SubData>` tags may not be flexibly defined. It must be written in a precisely defined order that will be documented for each SubData type. The description of each of these SubData formats follows.

ATC_Options

RLScenarioName

GScenarioName

IScenarioName

These three sections contain the pretty names of the RL Scenarios, G Scenarios, and I Scenarios. Each line consists of two values: Scenario Number and a name string enclosed in quotes.

Scenario Number	:	The scenarios are number 0 through the number of scenarios minus 1.
Scenario Name	:	These represent the names of the various scenarios.

Example:

```
<SUBDATA RLScenarioName>
//Index   Name
   0     "Scenario Name 0"
   1     "Scenario Name 1"
</SUBDATA>
```

ATCMemo

This section contains the memo text for the ATC analysis.

Example:

```
<SUBDATA ATCMemo>
//Memo
"Comments for the ATC analysis"
</SUBDATA>
```

ATCExtraMonitor

ATCFlowValue

This subdata section contains a list of a flow values for specified transfer levels. Each line consists of two values: Flow Value (flow on the monitored element) and a Transfer Level (in MW).

Flow Value	:	Contains a string describing which monitor this belongs to.
Transfer Level	:	Contains the value for this extra monitor at the last linear iteration.

Example:

```
<SUBDATA ATCFlowValue>
//MWFlow TransferLevel
   94.05    55.30
  105.18    80.58
  109.02   107.76
</SUBDATA>
```

ATCScenario

TransferLimiter

This subdata section contains a list of the TransferLimiters for this scenario. Each line contains fields relating one of the Transferlimiters. The fields are written out in the following order:

Limiting Element	:	Contains a description of the limiting element. The possible values are: "PowerFlow Divergence" "AREA num" "SUPERAREA name" "ZONE num" "BRANCH num1 num2 ckt" "INJECTIONGROUP name" "INTERFACE name"
Limiting Contingency	:	The name of the limiting contingency. If blank, then this means it's a limitation in the base case.
MaxFlow	:	The transfer limitation in MW in per unit.
PTDF	:	The PTDF on the limiting element in the base case (not in percent).
OTDF	:	The OTDF on the limiting element under the limiting contingency.
LimitUsed	:	The limit which was used to determine the MaxFlow in per unit.
PreTransEst	:	The estimated flow on the line after the contingency but before the transfer in per unit.
MaxFlowAtLastIteration	:	The total transfer at the last iteration in per unit.
IterativelyFound	:	Either YES or NO depending on whether it was iteratively determined.

Example:

```
<SUBDATA TransferLimiter >
  "BRANCH 40767 42103  1" "contin"  2.84 -0.0771 -0.3883 -4.35 -4.35 -0.01 "-55.88"
YES
  "BRANCH 42100 42321  1" "Contin"  4.42  0.1078  0.5466  6.50  5.64  1.57 " 22.59" NO
  "BRANCH 42168 42174  1" "Contin"  7.45 -0.0131 -0.0651 -1.39 -1.09  4.60 "-33.31" NO
  "BRANCH 42168 42170  1" "Contin"  8.54  0.0131  0.0651  1.39  1.02  5.69 " 26.10" NO
  "BRANCH 41004 49963  1" "Contin"  9.17 -0.0500 -0.1940 -4.39 -3.16  6.32 " 68.73" NO
  "BRANCH 46403 49963  1" "Contin"  9.53  0.0500  0.1940  4.46  3.16  6.68 "-68.68" NO
  "BRANCH 42163 42170  1" "Contin" 10.14 -0.0131 -0.0651 -1.39 -0.92  7.29 "-15.58" NO
</SUBDATA>
```

ATCExtraMonitor

This subdata section contains a list of the ATCExtraMonitors for this scenario. Each line contains three fields relating one of the ATCExtraMonitors. The first field describes the ATCExtraMonitor which this subdata corresponds to. The second and third variables are the initial value and sensitivity for this extra monitor for the sceanario. An optional fourth field may be included if we are using one of the iterated ATC solution options. This field must be the String "ATCFlowValue".

Monitor Description	:	Contains a string describing which monitor this belongs to.
InitialValue	:	Contains the value for this extra monitor at the last linear iteration.
Sensitivity	:	Contains the senstivity of this monitor.
ATCFlowValue	:	A string which signifies that a block will follow which stores a list of flow values for specified transfer levels. Each line of the block consists of two values: Flow Value (flow on the monitored element) and a Transfer Level (in MW). The block is terminated when a line of text that starts with 'END' is encountered.

Example:

```
<SUBDATA ATCExtraMonitor>
  "Interface<KEY1>Left-Right</KEY1>"                 40.0735 0.633295
  "Branch<KEY1>2</KEY1><KEY2>5</KEY2><KEY3>1</KEY3>" 78.7410 0.266589
</SUBDATA>
```

AUXFileExportFormatData

DataBlockDescription

This subdata section is used to define the objects that should be included in an auxiliary file along with their fields, subdata sections, and any filter used to specify which objects should be included. Each line contains the following:

ObjectType	:	Name of the object to include in the auxiliary file.
[FieldList]	:	List of fields to include. Must be enclosed in brackets. This list can either be space-delimited or comma-delimited.
[SubdataList]	:	List of subdata sections to include. This list must be enclosed in brackets and can be either space-delimited or comma-delimited. Include empty brackets to not include subdata or for objects that do not have any subdata sections.
"Filter"	:	Description of the filter to use for determining which objects to include. This must be enclosed in double quotes. If no filter is to be used, empty double quotes should be included. Valid entries are: "", "filtername", "AREAZONE", and "SELECTED". See the Using Filters in Script Commands section for more information on specifying the filtername.

Example:

```
<SUBDATA DataBlockDescription>
  // ObjectType [FieldList] [SubdataList] "Filter"
  Area [AreaName, AreaNum] [] "SELECTED"
  Gen [BusNum, BusName, GenID] [BidCurve, ReactiveCapability] ""
</SUBDATA>
```

AUXFileExportFormatDisplay

DataBlockDescription

Same format as for the AUXFileExportFormatData subdata section.

Example:

```
<SUBDATA DataBlockDescription>
  // ObjectType [FieldList] [SubdataList] "Filter"
  DisplayArea [AreaName, AreaNum, SOAuxiliaryID] [] ""
  DisplayTransmissionLine [BusNum, BusNum:1, LineCircuit, SOAuxiliaryID]
     [Line] "Nominal Voltage > 138 kV"
</SUBDATA>
```

BGCalculatedField

Condition

Calculated Fields allow you to define a calculation over most network and aggregation objects along with a few other types of objects. The calculation can then be used to show an aggregation calculation on objects that link to these calculation objects in some manner. Part of the definition is a filter which specifies which objects to operate over. This subdata section is identical to the Condition subdata section of the Filter object type.

Bus

MWMarginalCostValues

MvarMarginalCostValues

LPOPFMarginalControls

These three sections contain specific values computed for an OPF solution. In MWMarginalCostValues or MvarMarginalCostValues these specific values are the MW or Mvar marginal prices for each constraint. In LPOPFMarginalControls the values are the sensitivities of the controls with respect to the cost of each bus.

Example:

```
<SUBDATA MWMarginalCostValues>
  //Value
    16.53
     0.00
    21.80
</SUBDATA>
```

BusViewFormOptions

BusViewBusField

BusViewFarBusField

BusViewGenField

BusViewLineField

BusViewLoadField

BusViewShuntField

The values represent specific fields on the custom defined bus view onelines. Each line contains two values:

- Location : The various locations on the customized bus view contain slots for fields. This is the slot number.
- FieldDescription : This is a string enclosed in double quotes. The string itself is delimited by the @ character. The string contains five values:
  - Name of Field : The name of the field. Special fields that appear on dialog by default have special names. Otherwise these are the same as the fieldnames of the AUX file format (for the "other fields" feature on the dialogs).
  - Total Digit : Number of total digits for a numeric field.
  - Decimal Points : Number of decimal points for a numeric field.
  - Color : This is the color of the field. It is not presently used.
  - Increment Value : This is the "delta per mouse" click for the field.

Example:

```
<SUBDATA BusViewLineField>
  0 "MW Flow@6@1@0@0"
  1 "MVar Flow@6@1@0@0"
  2 "MVA Flow@6@1@0@0"
  3 "BusAngle:1@6@2@0@0"
</SUBDATA>
```

ColorMap

ColorPoint

A colorpoint is simply described by a real number (between 0 and 100) indicating the percentage breakpoint, an integer describing the color, and a field indicating if the color should be used or the contour should be transparent. These three values are written on a single line of text. Each line contains two values:

cmvalue	:	Real number between 0 and 100 (minimum to maximum value).
cmcolor	:	Integer between 0 and 16,777,216. Value is determined by taking the red, green and blue components of the color and assigning them a value between 0 and 255. The color is then equal to red + 256*green + 256*256*blue.
cmalpha	:	Integer between 0 and 255, where only 0 and 255 are valid values. A value of 0 indicates that the color point is transparent, while a value of 255 indicates that the color point is opaque. If the alpha channel is omitted, a default value of 255 (opaque) will be assigned.

Example:

```
<SUBDATA ColorPoint>
  // Value Color Alpha
  100.0000 127 255
   62.5000 65535 255
   50.0000 8388479 0
   12.5000 16711680 0
    0.0000 8323072 255
</SUBDATA>
```

Contingency

CTGElementAppend

Normally when reading in contingency definitions, the CTGElement SubData section is used to define the list of elements. When reading a CTGElement SubData section, all existing elements of the contingency are deleted are replaced with the ones read from the file. Using the CTGElementAppend as the SubData section will modify this behavior so that the elements are appended to the existing ones instead of deleted.

CTGElement

A contingency element is described by up to the following entries. All entries must be on a single line of text:

Action	:	String describing the action associated with this element. See below for actions available.
ModelCriteria	:	This is the name of a ModelFilter or ModelCondition under which this action should be performed. This entry is optional. If it is not specified, then a blank (or no criteria) is assumed. If you want to enter a Status, then use must specify "" as the ModelCriteria.

Status : The following options are available:

- CHECK : perform action if ModelCriteria is true
- ALWAYS : perform action regardless of ModelCriteria
- NEVER : do not perform action
- TOPOLOGYCHECK: perform action if ModelCriteria is true following implementation of other actions and before solving the power flow
- POSTCHECK : perform action if ModelCriteria is true following implementation of other actions and solving the power flow
- SOLUTIONFAIL : perform the action if ModelCriteria is true or not defined following the failure of the power flow solution.

This entry is optional. If it is not specified, then CHECK is assumed.

InclusionFilter : This entry is optional and will only exist for elements of RemedialAction or GlobalContingencyActions objects. This is the name of an advanced filter or device filter that gets applied to each contingency. If the contingency meets the filter, that contingency will include this element. Otherwise, the element will be ignored.

TimeDelay : This entry is optional. If not specified, 0 is assumed. This entry will only exist for elements of Contingency, RemedialAction, or GlobalContingencyActions objects. This is the time delay in seconds to wait before the action takes place.

Persistent : This entry is optional. It not specified, NO is assumed. Normally after a contingency action has been implemented it will not be applied again. Setting this option to YES to mark an action as persistent will change this behavior. Any action marked as persistent that also has a Status of TOPOLOGYCHECK, POSTCHECK, or SOLUTIONFAIL will be applied in the appropriate section of the overall contingency process any time that its ModelCriteria is met. An exception is that a SOLUTIONFAIL element will only remain persistent until a solution is successfully achieved.

ArmingCriteria : This entry is optional and will only exist for elements of RemedialAction objects. This is the name of a ModelFilter or ModelCondition under which this action should be armed. If it is not specified, then a blank (or no criteria) is assumed. If you want to enter a ArmingStatus, then use must specify "" as the ArmingCriteria.

ArmingStatus : This entry is optional and will only exist for elements of RemedialAction objects. If not specified, CHECK is assumed. The following options are available:

- CHECK : action is armed if ArmingCriteria is true
- ALWAYS : action is considered armed regardless of ArmingCriteria
- NEVER : action is not armed

Comment : All text to the right of the comment symbol (//) will be saved with the CTGElement as a comment.

Possible Actions:

Many actions have a `value` field that can be specified. This value can be expressed in three ways:

1. A numerical value that will be used directly.
2. The variablename of a field for the object in the action preceded by the tag `<Field>`. This field will be evaluated and that value will be used. Including the keyword REF in the appropriate place in the action string will cause the field to be evaluated in the contingency reference case. Otherwise, the field will be evaluated at the moment the action is implemented.

3. The name of a Model Expression preceded by the tag `<Expression>`. Single quotes should enclose the entirety of the tag and the name if the name contains spaces. The model expression will be evaluated and the result will be used as the value. Including the keyword `REF` in the appropriate place in the action string will cause the model expression to be evaluated in the contingency reference case. Otherwise, the model expression will be evaluated at the moment the action is implemented.

Transmission Line or Transformer outage or insertion

```
BRANCH | bus1# bus2# ckt   | OPEN
       |                  | CLOSE
       |                  | OPENCBS
       |                  | CLOSECBS
       |                  | SET_TO     | value      | LimitMVA   | REF
```

Takes branch out of service, or puts it in service. The contingency rating of the branch can also be set for the duration of the contingency using the `SET_TO` action. Note: bus# values may be replaced by a string enclosed in single quotes where the string is the name of the bus followed by an underscore character and then the nominal voltage of the bus. These values may also be replaced by a string enclosed in single quotes which represents the label of the bus. Also, the entire sequence [bus1# bus2# ckt] may be replaced by the label of the branch.

Generator, Load, or Switched Shunt outage or insertion

```
GEN                | bus# id    | OPEN, CLOSE, OPENCBS, or CLOSECBS
LOAD               | bus# id    | OPEN, CLOSE, OPENCBS, or CLOSECBS
SHUNT              | bus# id    | OPEN, CLOSE, OPENCBS, or CLOSECBS
INJECTIONGROUP     | name       | OPEN, CLOSE, OPENCBS, or CLOSECBS
```

Takes a generator, load, or shunt out of service, or puts it in service. If specifying an injection group, the status of all devices in the injection group will be changed. Note: bus# values may be replaced by a string enclosed in single quotes where the string is the name of the bus followed by an underscore character and then the nominal voltage of the bus. These values may also be replaced by a string enclosed in single quotes which represents the label of the bus. Also, the sequence [bus1# ckt] or [name] may be replaced by the label of the device.

Generator, Load or Switched Shunt movement to another bus

For the following set of actions, all of the object types can use the same action keywords which are associated with the value keyword following the actual value. The move is based on specifying a bus:

```
GEN    | bus1#      | MOVE_P_TO  | bus2#      | value      | MW   | REF
LOAD   |            | MOVE_Q_TO  |            |            | MVR  |
SHUNT  |            |            |            |            |      |
```

For the following set of actions, all of the object types can use the same action keywords which are associated with the value keyword following the actual value. The move is based on specifying a particular device:

```
GEN    | bus1# id   | MOVE_P_TO  | bus2#      | value      | MW   | REF
LOAD   |            | MOVE_Q_TO  |            |            | MVR  |
SHUNT  |            |            |            |            |      |
```

Generator actions that move a generator by a percentage only apply to the generator MW:

```
GEN    | bus1#      | MOVE_P_TO  | bus2#      | value      | PERCENT    | REF
GEN    | bus1# id   | MOVE_P_TO  | bus2#      | value      | PERCENT    | REF
```

The following set of actions are used for specifying a load move by maintaining a constant power factor:

```
LOAD   | bus1#       | MOVE_PQ_TO  | bus2#       | value       | MW   | REF
LOAD   | bus1# id    | MOVE_PQ_TO  | bus2#       |             | MW   |
```

The following set of actions apply to loads and shunts and are used to move a percentage of the entire MW and Mvar output. The move is based on specifying a bus:

```
LOAD   | bus1#       | MOVE_PQ_TO  | bus2#       | value       | PERCENT     | REF
SHUNT  |             |             |             |             |             |
```

The following set of actions apply to loads and shunts and are used to move a percentage of the entire MW and Mvar output. The move is based on specifying a particular device:

```
LOAD   | bus1# id    | MOVE_PQ_TO  | bus2#       | value       | PERCENT     | REF
SHUNT  |             |             |             |             |             |
```

Use to move generation, load or shunt at a bus1 over to bus2. This can be used on a bus or specific device basis in specifying what to move. Note: bus# values may be replaced by a string enclosed in single quotes where the string is the name of the bus followed by an underscore character and then the nominal voltage of the bus. These values may also be replaced by a string enclosed in single quotes which represents the label of the bus. When identifying specific devices, the device label can replace the bus number and device id.

Generator, Load or Switched Shunt set or change a specific value

For the following set of actions, all of the object types can use the same action keywords which are associated with the value keyword following the actual value. These changes are based on specifying a bus:

```
GEN    | bus#        | SET_P_TO    | value       | MW   | REF
LOAD   |             | SET_Q_TO    |             | MVR  |
SHUNT  |             | CHANGE_P_BY |             | MW   |
       |             | CHANGE_Q_BY |             | MVR  |
```

For the following set of actions, all of the object types can use the same action keywords which are associated with the value keyword following the actual value. These changes are based on specifying a bus:

```
GEN    | bus# id     | SET_P_TO    | value       | MW   | REF
LOAD   |             | SET_Q_TO    |             | MVR  |
SHUNT  |             | CHANGE_P_BY |             | MW   |
       |             | CHANGE_Q_BY |             | MVR  |
```

The following set of actions are used to set or change the MW output of generation at a bus by a percentage:

```
GEN    | bus#        | SET_P_TO    | value       | PERCENT     | REF
GEN    |             | CHANGE_P_BY |             |             |
```

The following set of actions are used to set or change the MW output of a particular generator by a percentage:

```
GEN    | bus# id     | SET_P_TO    | value       | PERCENT     | REF
GEN    |             | CHANGE_P_BY |             |             |
```

The following set of actions apply to loads and shunts and are used to set or change a percentage of the entire MW and Mvar output. This based on specifying a bus:

```
LOAD   | bus#        | SET_PQ_TO         | value       | PERCENT     | REF
SHUNT  |             | CHANGE_PQ_BY      |             |             |
```

The following set of actions apply to loads and shunts and are used to set or change a percentage of the entire MW and Mvar output. This based on specifying a specific device:

```
LOAD   | bus# id     | SET_PQ_TO         | value       | PERCENT     | REF
SHUNT  |             | CHANGE_PQ_BY      |             |             |
```

The following set of actions are used to specify a load set or change by maintaining a constant power factor. This is based on specifying a bus:

```
LOAD   | bus# | SET_PQ_TO         | value       | MW   | REF
       |      | CHANGE_PQ_BY      |             | MW   |
```

The following set of actions are used to specify a load set or change by maintaining a constant power factor. This is based on specifying a specific load:

```
LOAD   | bus# id     | SET_PQ_TO          | value       | MW   | REF
       |             | CHANGE_PQ_BY       |             | MW   |
```

The following set of actions apply to generators and shunts and are used to set or change the setpoint voltage of the devices at the specified bus:

```
GEN    | bus#        | SET_VOLT_TO        | value       | PU   | REF
SHUNT  |             |CHANGE_VOLT_BY      |             |      |
```

The following set of actions apply to generators and shunts and are used to set or change the setpoint voltage of the specified device:

```
GEN    | bus# id     | SET_VOLT_TO        | value       | PU   | REF
SHUNT  |             |CHANGE_VOLT_BY      |             |      |
```

Note: bus# values may be replaced by a string enclosed in single quotes where the string is the name of the bus followed by an underscore character and then the nominal voltage of the bus. These values may also be replaced by a string enclosed in single quotes which represents the label of the bus. When identifying specific devices, the device label can replace the bus number and device id.

Bus outage causes all lines connected to the bus to be outage

```
BUS    | bus# | OPEN
             | OPENCBS
```

Takes all branches connected to the bus out of service. Also outages all generation, load, or shunts attached to the bus. Note: bus# values may be replaced by a string enclosed in single quotes where the string is the name of the bus followed by an underscore character and then the nominal voltage of the bus. These values may also be replaced by a string enclosed in single quotes which represents the label of the bus.

Interface outage or insertion

```
INTERFACE     | name | OPEN
                     | CLOSE
                     | OPENCBS
                     | CLOSECBS
```

Takes all monitored branches in the interface out of service, or puts them all in service. Open actions will also open all generators and loads contained in the interface including generators and loads inside any injection groups or other interfaces. Note: the [name] may be replaced by the label of the interface.

Interface change specific value

```
INTERFACE | name     | CHANGE_P_BY | value       | Option      | REF  | PPREF
                     | SET_P_TO    |             |             |      |
```

The following `Option` settings are allowed to set or change the MW flow of an interface by or to a particular value:

`MWMERITORDEROPEN`

: Value will be interpreted as the amount of MW flow change or new MW flow. The element in the interface with the highest participation factor will be opened, followed by the second generator and so on. This will continue until the amount of MW flow opened is as close to the desired amount as possible without exceeding the amount of MW flow open. If an element will cause a change in flow that is not in the desired direction, that element is not opened and the next element is examined.

`PERCENTMERITORDEROPEN` or `%MERITORDEROPEN`

: Same as `MWMERITORDEROPEN` except that the value will be interpreted as percentage of the contingency reference state MW flow.

`MWMERITORDEROPENEXCEED`

: Same as `MWMERITORDEROPEN` except that the amount of MW opened is allowed to exceed the desired amount of change. Interface elements will be opened in merit order until the desired amount is met or exceeded.

`PERCENTMERITORDEROPENEXCEED` or `%MERITORDEROPENEXCEED`

: Same as MWMERITORDEROPENEXCEED except that the value will be interpreted as percentage of the contingency reference state MW injection.

MWEFFECTOPEN

: Value that is specified with the action is the desired MW Effect that the action should have. The participation factors defined with the interface elements will be interpreted as effectiveness factors akin to transfer distribution factors. These factors are supplied as input by the user when defining the interface. The effectiveness factors are multiplied by the present MW flow of elements in the interface to determine how much effect they will have if dropped. The flow of an element is determined by its MW flow multiplied by the Weighting factor specified with the element. If the effect of a particular element is in the opposite direction of the desired effect, that element is skipped. The action will find the smallest number of elements to drop that results in a total MW Effect that is within 5% of the desired MW Effect, but does not exceed the desired MW Effect.
This option is not valid with SET_P_TO actions.

MWEFFECTOPENEXCEED

: Same as MWEFFECTOPEN but it will ensure that the total MW Effect is within 5% of the total desired MW Effect and also meets or exceeds the desired MW Effect.
This option is not valid with SET_P_TO actions.

MWBESTFITOPEN

: Value will be interpreted as the amount of MW flow change or new MW flow. When using this option the participation factors of interface elements do not impact which elements are opened. The flow on an element, which is determined by its MW flow multiplied by the weighting factor specified with the element, is used to determine which elements should be opened. If the follow on an element is in the appropriate direction to achieve the desired flow change on the interface, that element is eligible fo being opened. To determine if an element will actually be opened, the best fit algorithm attempts to determine the combination of elements that will achieve the desired flow change by opening the least amount of elements and achieving an actual flow change within 5% fo the desired flow change without exceeding the desired amount.

MWBESTFITOPENEXCEED

: Same as MWBESTFITOPEN except that the MW flow change is allowed to exceed the desired amount of change.

PERCENTBESTFITOPEN or %BESTFITOPEN

: Same as MWBESTFITOPEN except that the value will be interpreted as percentage of the contingency reference state MW flow.

PERCENTBESTFITOPENEXCEED or %BESTFITOPENEXCEED

: Same as PERCENTBESTFITOPEN except that the MW flow change is allowed to exceed the desired amount of change.

When using an action that requires participation factors, an optional parameter PPREF can be specified. This indicates that the participation factors should be determined in the contingency reference case.

Interfaces can contain other interfaces. The treatment of interfaces within interfaces is to open the entire contained interface when using the MWMERITORDEROPEN type actions.

Notes: The [name] may be replaced by the label of the interface.

Line Shunt outage or insertion

```
LINESHUNT     | bus1# bus2# bus# ckt     | OPEN
              | CLOSE
```

Takes a line shunt out of service, or puts it in service. bus1# and bus2# identify the line that the line shunt is on and bus# identifies the side of the line that the line shunt is on. bus# values may be replaced by a string enclosed in single quotes where the string is the name of the bus followed by an underscore character and then the nominal voltage of the bus. bus# values may also be replaced by a string enclosed in single quotes that represents the label of the bus. The sequence [bus1# bus2#] may be replaced by the label of the line to which the line shunt is attached.

Injection Group outage or insertion

```
INJECTIONGROUP      | name       | OPEN
                    | CLOSE      | value       | REF  | PPREF
                    | OPENCBS
                    | CLOSECBS
                    | OPEN       | value       | REF  | PPREF
```

Takes all devices in the injection group out of service, or puts them all in service.
The `OPEN` action will open all devices in the injection group if no value is specified. If a value is specified, only that number of devices will be opened in the order of highest to lowest participation factor. The CLOSE action will close all devices in the injection group if no value is specified. If a value is specified, only that number of devices will be closed in the order of highest to lowest participation factor. When using an action that requires participation factors, an optional parameter `PPREF` can be specified. This indicates that the participation factors should be determined in the contingency rerference case. This will only be done for participation points using an AutoCalcMethod that indicates the factor should be dynamically determined and the AutoCalc field is set to YES for the participation point.

The [name] may be replaced by the label of the injection group. Bus participation points will be completely ignored in this process.

Injection Group change specific value

```
INJECTIONGROUP      | name | CHANGE_P_BY | value       | Option      | REF  | PPREF
                           | SET_P_TO    |             |             |      |
```

The following `Option` settings are allowed to set or change the MW generation/load in an injection group by or to a particular value:

- `MW` : Value will be interpreted as the amount of MW injection change or new MW injection. Each participation point in the injection group will be changed in proportion to the participation factors of the group.
- `PERCENT` or `%` : Same as MW except that the value will be interpreted as percentage of the contingency reference state MW injection.
- `MWMERITORDER` : Value will be interpreted as the amount of MW injection change or new MW injection. Both generator and load points will be modified in the injection group. Elements will be adjusted in order of highest participation factor to lowest before moving to the next element. This process continues until the desired injection is met. Generators will not be opened in this process, which means all online generators will continue to provide Mvar support. Loads that have both their minimum and maximum MW limits set to zero will not be allowed to increase. They can only decrease towards 0.
- `PERCENTMERITORDER` or `%MERITORDER` : Same as `MWMERITORDER` except that the value will be interpreted as percentage of the contingency reference state MW injection.

MWMERITORDEROPEN

: Value will be interpreted as the amount of MW injection change or new MW injection. Both generator and load points can be modified in the injection group. If the MW injection change is negative, the generator in the injection group with the highest participation factor will have its status changed to Open, followed by the second generator and so on. This will continue until the amount of MW opened is as close to the desired amount as possible without exceeding the desired amount of drop. If the MW injection change is positive, loads will be opened in the same manner. If an element would cause the desired gen drop amount to be exceeded, that element is skipped and the next element in merit order is processed. If the change requested is positive and there are no loads in the injection group, generators will be increased toward their maximum MW output in the same manner as `MWMERITORDER` as though the `OPEN` option was not specified. If the change requested is negative and there are no generators in the injection group, loads will be increased toward their maximum MW output in the same manner as `MWMERITORDER` as though the `OPEN` option was not specified.

`PERCENTMERITORDEROPEN` or `%MERITORDEROPEN`

: Same as `MWMERITORDEROPEN` except that the value will be interpreted as percentage of the contingency reference state MW injection.

`MWMERITORDEROPENEXCEED`

: Same as `MWMERITORDEROPEN` except that the amount of MW opened is allowed to exceed the desired amount of change. Generators or loads will be opened in merit order until the desired amount is met or exceeded.

`PERCENTMERITORDEROPENEXCEED` or `%MERITORDEROPENEXCEED`

: Same as `MWMERITORDEROPENEXCEED` except that the value will be interpreted as percentage of the contingency reference state MW injection.

`MWEFFECTOPEN` : Value that is specified with the action is the desired MW Effect that the action should have. The participation factors defined with the Injection Group will be interpreted as effectiveness factors akin to transfer distribution factors. These factors are supplied as input by the user when defining the injection group. The effectiveness factors are multiplied by the present output of generators (or loads) in the injection group to determine how much effect they will have if dropped. The action will find the smallest number of generators (or loads) to drop which results in a total MW Effect that is within 5% of the desired MW Effect, but does not exceed the desired MW Effect.
This option is not valid with `SET_P_TO` actions.

`MWEFFECTOPENEXCEED`

: Same as `MWEFFECTOPEN` but it will ensure that the total MW Effect is within 5% of the total desired MW Effect and also meets or exceeds the desired MW Effect.
This option is not valid with `SET_P_TO` actions.

`MWBESTFITOPEN` : Value will be interpreted as the amount of MW injection change or new MW injection. When using this option the participation factors do not impact which elements are opened. All generators or loads defined with the injection group can participate if they are online. Specifially which generators or loads depends on an algorithm that attempts to get the actual injection change within 5% of the desired injection change by

opening the smallest number of generators or loads without exceeding the desired amount.

`MWBESTFITOPENEXCEED`
: Same as `MWBESTFITOPEN` except that the MW injection change is allowed to exceed the desired amount of change.

`PERCENTBESTFITOPEN` or `%BESTFITOPEN`
: Same as `MWBESTFITOPEN` except that the value will be interpreted as percentage of the contingency reference state MW injection.

`PERCENTBESTFITOPENEXCEED` or `%BESTFITOPENEXCEED`
: Same as `PERCENTBESTFITOPEN` except that the MW injection change is allowed to exceed the desired amount of change.

When using an action that requires participation factors, an optional parameter `PPREF` can be specified. This indicates that the participation factors should be determined in the contingency reference case. This will only be done for participation points using an AutoCalcMethod that indicates the factor should be dynamically determined and the AutoCalc field is set to YES for the participation point.

Injection Groups can contain participation points that reference another injection group. The treatment of injection groups within injection groups will be to drop the entire contained injection group when using the `MWMERITORDEROPEN` and `MWEFFECTOPEN` type actions.

Notes: The [name] may be replaced by the label of the injection group. Bus participation points will be completely ignored in this process.

Series Capacitor Bypass or Inservice

```
SERIESCAP     | bus1# bus2# ckt    | BYPASS
                                   | INSERVICE
```

Bypasses a series capacitor, or puts it in service. Note: bus# values may be replaced by a string enclosed in single quotes where the string is the name of the bus followed by and underscore character and then the nominal voltage of the bus. Note: bus# values may also be replaced by a string enclosed in single quotes which represents the label of the bus. Also, the entire sequence [bus1# bus2# ckt] may be replaced by the label of the branch. Keyword SERIESCAP may also be replaced with BRANCH, which will allow bypassing or not bypassing any branch and is not limited to series capacitors.

Series Capacitor set impedance

```
SERIESCAP     | bus1# bus2# ckt    | SET_X_TO    | value       | PERCENT     | REF
                                   |             |             | PU          |
```

Changes the impedance a series capacitor either specifying a new per unit value or specifying a percentage of the value in the contingency reference case. Note: bus# values may be replaced by a string enclosed in single quotes where the string is the name of the bus followed by and underscore character and then the nominal voltage of the bus. Note: bus# values may also be replaced by a string enclosed in single quotes which represents the label of the bus. Also, the entire sequence [bus1# bus2# ckt] may be replaced by the label of the branch. Keyword SERIESCAP may also be replaced with BRANCH, which will allow setting the impedance of any branch and is not limited to series capacitors.

DC Transmission or VSC DC Transmission Line outage

```
DCLINE        | bus1# bus2# ckt    | OPEN
              | OPENCBS
VSCDCLINE     | 'Name'             | OPEN
              | OPENCBS
```

Takes DC Line or VSC DC Line out of service. Note: bus# values may be replaced by a string enclosed in single quotes where the string is the name of the bus followed by an underscore character and then the nominal voltage of the bus. These values may also be replaced by a string enclosed in single quotes

which represents the label of the bus. Also, the entire sequence [bus1# bus2# ckt] may be replaced by the label of the dc transmission line. For the VSC DC Line, the identifiers are replaced simply with the name of the VSCDCLINE instead.

DC Line set a specific value or insertion

```
DCLINE        | bus1# bus2# ckt   | SET_P_TO    | value       | MW          | REF
                                  | CHANGE_P_BY |             | PERCENT
                                  | SET_I_TO    |             | AMPS
                                  | CHANGE_I_BY |
                                  | CLOSE       |
                                  | CLOSECBS    |
                                  | SET_TO      | value       | OHMS        | REF
VSCDCLINE     | 'Name'            | Same options as for the DC Line, except that
                                    the AMPS option are not avalailable for VSC
```

Use to set the DC Line setpoint to a particular value, or puts it in service. Note: bus# values may be replaced by a string enclosed in single quotes where the string is the name of the bus followed by an underscore character and then the nominal voltage of the bus. Note: bus# values may also be replaced by a string enclosed in single quotes which represents the label of the bus. Also, the entire sequence [bus1# bus2# ckt] may be replaced by the label of the dc transmission line. (Note: for the CLOSE and CLOSECBS choice, only the units of MW or AMPS may be used.) For the VSC DC Line, the identifiers are replaced simply with the name of the VSCDCLINE instead.

MTDC Converter outage

```
DCCONVERTER   | rec# bus#   | OPEN
                            | OPENCBS
```

Takes multi-terminal DC converter out of service. The rec# specifies the multi-terminal DC line record, while bus# specifies the AC bus to which the converter is connected. Note: bus# values may be replaced by a string enclosed in single quotes where the string is the name of the bus followed by an underscore character and then the nominal voltage of the bus. These values may also be replaced by a string enclosed in single quotes which represents the label of the bus.

MTDC Converter set a specific value or insertion

```
DCCONVERTER   | rec# bus#   | SET_P_TO    | value       | MW          | REF
                            | CHANGE_P_BY |             | PERCENT
                            | SET_I_TO    |             | AMPS
                            | CHANGE_I_BY |
                            | CLOSE       |
                            | CLOSECBS    |
```

Use to set the multi-terminal DC converter setpoint to a particular value, or puts it in service. The rec# specifies the multi-terminal DC line record, while bus# specifies the AC bus to which the converter is connected. Note: bus# values may be replaced by a string enclosed in single quotes where the string is the name of the bus followed by an underscore character and then the nominal voltage of the bus. Note: bus# values may also be replaced by a string enclosed in single quotes which represents the label of the bus. (Note: for the CLOSE and CLOSECBS choice, only the units of MW or AMPS may be used.)

Phase Shifter set a specific value

```
PHASESHIFTER  | bus1# bus2# ckt   | SET_P_TO    | value       | MW          | REF
                                  | CHANGE_P_BY |             | PERCENT
                                  | SET_TO      | value       | DEG         | REF
                                  | CHANGE_BY   |             |             |
```

Use the MW and PERCENT options to change or set the middle of the phase shifter MW regulation range to the specified value. Use the DEG option to change or set the phase shift angle in degrees to a particular value.

Note: bus# values may be replaced by a string enclosed in single quotes where the string is the name of the bus followed by an underscore character and then the nominal voltage of the bus. These values may

also be replaced by a string enclosed in single quotes which represents the label of the bus. Also, the entire sequence [bus1# bus2# ckt] may be replaced by the label of the branch.

Keyword PHASESHIFTER may also be replaced with BRANCH. If the branch is not a phase shifter, no change will be made.

3-Winding Transformer outage or insertion

```
3WXFORMER     | bus1# bus2# bus3# ckt    | OPEN
                                         | CLOSE
                                         | OPENCBS
                                         | CLOSECBS
```

Takes all three windings of a 3-winding transformer out of service, or puts them in service. Note: bus# values may be replaced by a string enclosed in single quotes where the string is the name of the bus followed by an underscore character and then the nominal voltage of the bus. Note: bus# values may also be replaced by a string enclosed in single quotes which represents the label of the bus. Also, the entire sequence [bus1# bus2# bus#3 ckt] may be replaced by the label of the three winding transformer.

Area Control Type Change

```
AREA   | area#       | SET_TO      | 'OFF'
                                   | 'PARTFAC'
                                   | 'AREASLACK  bus#'
                                   | 'IGSLACK injectiongroup name'
```

Specify to change the make-up power for an area so that it is different during a contingency than the area control settings used in the reference case. The area may be set to toggle the control setting to OFF, PARTFAC, AREASLACK, and IGSLACK. The Area Control topic provides more information about these control types. If selecting Area Slack is chosen, then a bus must be specified which will act as the area slack during the contingency action. If selecting IG Slack, then an injection group must be specified by name.
Note: bus# values may be replaced by a string where the string is the name of the bus followed by an underscore character and then the nominal voltage of the bus. Note: bus# values may also be replaced by a string which represents the label of the bus.
In order for the Area contingency action to work correctly, there are contingency and power flow solution options that must be set correctly. Simulator does not automatically set these options so the user must make sure they are set.

- Area control must be enabled in the contingency base case, i.e. the Power Flow Solution Option for Island-Based AGC must be set to Disable (Use the Area and Super Area Dispatch settings).
- The contingency Make-Up Power option must be set to Same as Power Flow case.
- The option to Disable Automatic Generation Control (AGC) found with the Power Flow Solution Options must NOT be selected.

Another suggestion, although not a strict requirement, is that the area should be on area control prior to contingency analysis if a control type other than Off AGC is going to be set during a contingency. If a large ACE exists in the base case with area control off, switching the area on control during the contingency will zero out the ACE in addition to compensating for required make-up power.

Substation outage

```
SUBSTATION    | sub# | OPEN
                     | OPENCBS
                     | SET_P_TO    | value       | MW          | REF
                     | CHANGE_P_BY |             | PERCENT
```

OPEN and OPENCBS will take a substation out of service. The set and change actions will set the MW output of online generators in the substation to the specified value. sub# is the number that identifies the

substation. sub# can be replaced by a string enclosed in single quotes where the string is the name or label of the substation.

Abort

```
ABORT
```

Include this action to cause the solution of the contingency to be aborted.

Execute a Power Flow Solution

```
SOLVEPOWERFLOW
```

Include this action to cause the solution of the contingency to be split into pieces. Actions that are listed before each SOLVEPOWERFLOW call will be performed as a group.

Calling of a name ContingencyBlock

```
CONTINGENCYBLOCK | name
```

Calls a ContingencyBlock and executes each of the actions in that block.

Make-Up Power Compensation

Only valid immediately following a SET, CHANGE, OPEN or CLOSE action on a Generator, Shunt or Load. This describes how the change in MW or MVAR are picked up by buses throughout the system. The values specify participation factors. Note: bus# values may be replaced by a string enclosed in single quotes where the string is the name of the bus followed by and underscore character and then the nominal voltage of the bus.

```
COMPENSATION
bus#1   value1
bus#2   value2
...
END
```

Example:

```
<SUBDATA CTGElement>
  // just some comments
  // action                     Model Criteria  Status TimeDelay  comment
  "BRANCH 40821 40869 1  OPEN" ""               ALWAYS 0 //Raver - Paul 500 kV
  "GEN 45041 1  OPEN"          ""               ALWAYS 0 //Trip Unit #2
  "BRANCH 42702 42727 1  OPEN" "Line X Limited" CHECK  0 //Open Fern Hill
  "GEN 40221 1  OPEN"          "Interface L1"   CHECK  0 //Drop ~600 MW
  "GEN 40227 1  OPEN"          "Interface L2"   CHECK  0 //Drop ~1200 MW
  "GEN 40221 1  OPEN"          "Interface L3"   CHECK  0 //Drop ~600 MW
</SUBDATA>
```

Note: ContingencyElement object types can also be directly created inside their own DATA section as well. One of the key fields of the object is then the name of the contingency to which the ContingencyElement belongs. The Action string will remain the same.

LimitViol

A LimitViol is used to describe the results of a contingency analysis run. Each Limit Violation lists nine possible values:

ViolType : One of six values describing the type of violation.
- BAMP : branch amp limit violation
- BMVA : branch MVA limit violation
- VLOW : bus low voltage limit violation
- VHIGH : bus high voltage limit violation
- INTER : interface MW limit violation
- CUSTOM : Custom Monitor value

ViolElement : This field depends on the ViolType.
- for VLOW, VHIGH : "bus1#" or "busname_buskv" or "buslabel"
- for INTER : "interfacename" or "interfacelabel"
- for BAMP, BMVA : "bus1# bus2# ckt violationbus# MWFlowDirection"

  violationbus# is the bus number for the end of the branch which is violated

  MWFlowDirection is the direction of the MW flow on the line. Potential values are "FROMTO" or "TOFROM".

  Note: each bus# may be replaced with the name underscore nominal kV string enclosed in single quotations. Or bus# values may be replaced by a string enclosed in single quotes representing the label of the bus. Also the entire sequence [bus1# bus2# ckt] may be replaced by the label of the branch.
- for CUSTOM : "custommonitorname deviceidentifier" where the deviceidentifier will use the key fields or label as specified by the option selected when saving

Limit : This is the numerical limit which was violated.

ViolValue : This is the numerical value of the violation.

PTDF : This field is optional. It only makes sense for interface or branch violations. It stores a sensitivity of the flow on the violating element during in the base case with respect to a transfer direction This must be calculated using the Contingency Analysis Other Actions related to Sensitivities.

OTDF : Same as for the PTDF.

InitialValue : This stores a number. This stores the base case value for the element which is being violated. This is used to compare against when looking at change violations.

Reason : This will say whether this was a pure violation, or is being reported as a violation because the change from the base case is higher than a specified threshold.
- LIMIT : means this is a violation of a line/interface/bus limit or simply a Custom Monitor
- CHANGE : means this is being reported as a limit because the change from the initial value is higher than allowed

CTG Specified Limit : This specifies if the Limit originated from a contingency action or from the rating specified with the line and Limit Monitoring Settings.
- NO : the Limit originated from the line and Limit Monitoring Settings
- YES : the Limit originated from a contingency action

Example:

```
<SUBDATA LimitViol>
  BAMP "1 3  1 1 FROMTO"  271.94031  398.48096  10.0  15.01  //Note OTDF/PTDF
  // values can also be specified with name underscore nominal kV string
  // enclosed inside a single quote as shown next
  BAMP "'One_138' 'Three_138' 1 1 FROMTO"  271.94031  398.48096  10.0  15.01
  INTER "Right-Top"   45.00000   85.84451  None   None  56.000  LIMIT NO
</SUBDATA>
```

ViolationCTG object types can also be directly created inside their own DATA section as well. One of the key fields of the object is then the name of the contingency to which the ViolationCTG belongs.

Sim_Solution_Options

These describe the power flow solution options which should be used under this particular contingency. The format of the subdata section is two lines of text. The first line is a list of the fieldtypes for Sim_Solution_Options which should be changed. The second line is a list of the values. Note that in general, power flow solution options are stored at three different locations in contingency analysis. When implementing a contingency, Simulator gives precendence to these three locations in the following order:

1. Contingency Record Options (stored with the particular contingency).
2. Contingency Tool Options (stored with CTG_Options).
3. The global solution options.

WhatOccurredDuringContingency

Each line of this subdata section is part of a text description of what actually ended up being implemented for this contingency. This will list which actions were executed and which actions ended up being skipped because of their model criteria. Each line of the subdata section must be enclosed in quotes.

Example:

```
<SUBDATA WhatOccurredDuringContingency>
  "Applied: "
  "  OPEN Branch Two      (2)  TO  Five     (5) CKT 1 |  | CHECK | | ELEMENT"
</SUBDATA>
```

ContingencyMonitoringException

Each line of this subdata section contains a string identifying a specially handled monitored element for this contingency followed by a string indicating how this monitored element should be handled with this contingency. The elements can be identified by their primary or secondary key fields or by label. The element descriptions should be enclosed in quotes because they contain spaces.

Example:

```
<SUBDATA ContingencyMonitoringException>
  "Branch '2' '3' '1'" "Exclude"
  "Branch 'Three_138.00' 'Four_138.00'" "Include"
  "Branch 'Line_2_5'" "Default"
</SUBDATA>
```

CTG_Options

Sim_Solution_Options

These describe the power flow solution options which should be used under this particular contingency. The format of the subdata section is two lines of text. The first line is a list of the fieldtypes for Sim_Solution_Options which should be changed. The second line is a list of the values. Note that in general, power flow solution options are stored at three different locations in contingency analysis. When implementing a contingency, Simulator gives precendence to these three locations in the following order:

1. Contingency Record Options (stored with the particular contingency).
2. Contingency Tool Options (stored with CTG_Options).

3. The global solution options.

CTGElementBlock

CTGElement

This format is the same as for the Contingency objecttype, however, you cannot call a ContingencyBlock from within a contingencyblock.

CTGElementAppend

When a subdata section is defined as CTGElementAppend rather than CTGElement, the actions of this subdata section will be appended to the contingency actions, instead of replacing them. This format is the same as for the Contingency objecttype, however, you cannot call a ContingencyBlock from within a contingencyblock.

Note: CTGElementBlockElement object types can also be directly created inside their own DATA section as well. One of the key fields of the object is then the name of the contingency block to which the CTGElementBlockElement belongs.

CustomColors

CustomColors

These describe the customized colors used in Simulator, which are specified by the user. A custom color is an integer describing a color. Each custom color is written on a single line of text and is an integer between 0 and 16,777,216. The value is determined by taking the red, green, and blue components of the color and assigning them a value between 0 and 255. The color is then equal to red + 256*green + 256*256*blue. Each line contains only one integer that corresponds to the color specified.

Example:

```
<SUBDATA CustomColors>
  9823301
  8613240
</SUBDATA>
```

CustomCaseInfo

ColumnInfo

Each line of this SUBDATA section can be used for specifying the column width of particular columns of the respective Custom Case Information Sheet. The line contains two values – the column and then a column width. This is shown in the following example.

Example:

```
<SUBDATA ColumnInfo>
  "SheetCol"    133
  "SheetCol:1"  150
  "SheetCol:2"  50
</SUBDATA>
```

DataGrid

ColumnInfo

Contains a description of the columns which are shown in the respective data grid. Each line of text contains at least four fields: VariableName, ColumnWidth, TotalDigits, DecimalPoints. The remaining fields are used when showing a Data View based on this DataGrid object. See help website for Data View or the OpenDataView script command for more information about this.

Variablename : Contains the variable which is shown in this column.

ColumnWidth	:	The column width.
TotalDigits	:	The total digits displayed for numerical values.
DecimalPoints	:	The decimal points shown for numerical values.
TabBreak	:	Optional. Default to NO. Set to YES to indicate that a new tab should be started immediately before this field.
TabCaption	:	Optional. Default to blank string. Specifies a caption for the tabbed control for fields occurring after the Tab Break.
RowBreak	:	Optional. Default to NO. Set to YES to indicate that a new row should be started immediately before this field.
RowCaption	:	Optional. Default to blank string. Specifies a caption for a group box for the fields occurring after the row break.
ColBreak	:	Optional. Default to NO. Set to YES to indicate that a new Column should be started immediately before this field. Also, a special feature for column breaks only is you may specify a number after YES to indicate multiple column breaks to skip over a column. For example "YES 2" to skip a column because there are 2 consecutive column breaks.
RowCaption	:	Optional. Default to blank string. Specifies a caption for a group box for the fields occurring after the column break.

Example:

```
DataGrid (DataGridName)
{
   BUS
   <SUBDATA COLUMNINFO>
      BusNomVolt   100   8   2
      AreaNum       50   8   2  "YES" "Tab Caption" "NO" "" "NO" ""
      ZoneNum       50   8   2
   </SUBDATA>
   BRANCHRUN
   <subdata COLUMNINFO >
      BusNomVolt:0  100   8   2
      BusNomVolt:1  100   8   2 "NO" "" "NO" "" "YES 2" "Col Caption"
      LineMW:0      100   9   3 "NO" "" "YES" "Row Caption" "NO" ""
   </SUBDATA>
}
```

ColumnContourInfo

Contains a description of the column contour settings

ColumnNumber	:	Contains the column index of the contoured column
UseAbsValue	:	Contour the absolute value if YES. If NO then contour the signed value.
IgnoreValuesAbove	:	If YES values above the maximum percentage are ignored.
IgnoreValuesBelow	:	if YES values below the minimum percentage are ignored.
AbsMin	:	The minimum value for the colormap
LimMin	:	The break low value for the colormap
Nominal	:	The nominal value for the colormap
LimMax	:	The break high value for the colormap
AbsMax	:	The maximum value for the colormap
ColorMapName	:	The name of the color map the contour is using. This must reference a color map that exists in the case.
ColorMapBrightness	:	The brightness or color saturation. The values range from -0.8 (darker) to 0.8 (brigher).
ColorMapReverseColors	:	If YES, the colors in the colormap will be reversed.

Example:

```
DATA (DataGrid,
[DataGridName,BGDisplayFilter,FilterName,NonDefaultFont,CaseInfoRowHeight,FontName,Fon
tStyles,SOFontSize,FontColor,VariableName,ConditionType,ViewZoomLevel,FrozenColumns])
{
"Bus" "YES" "" "YES"     13 "Segoe UI" ""      8      0 "" "High To Low" 100.00 -1
   <SUBDATA ColumnInfo>
     "BusNum"                75  8 2
     "BusName"               75  8 2
     "AreaName"              75  8 2
     "BusNomVolt"            75  8 2
     "BusPUVolt"             75  8 5
     "BusKVVolt"             75  8 3
     "BusAngle"              75  8 2
   </SUBDATA>
   <SUBDATA ColumnContourInfo>
     5  NO   NO   NO   0.996563732624054  1.01118695735931  1.02581024169922
1.03790521621704  1.05000007152557  "Blue=low, Red=High"  0  NO
     7  NO   NO   NO   -1.17415904998779  0.0616339445114136  1.29742693901062
3.84944224357605  6.4014573097229  "Discrete 20 Red/Blue"  0  NO
   </SUBDATA>
}
```

DynamicFormatting

DynamicFormattingContextObject

This subdata section contains a list of the display object types which are chosen to be selected. Each line of the section consists of the following:

```
DisplayObjectType  (WhichFields)  (ListOfFields)
```

DisplayObjectType	:	The object type of the display object. These are generally the same as the values seen in the subdata section SelectByCriteriaSetType of SelectByCriteriaSet object types. The only exception is the string CaseInfo, which is used for formatting applying to the case information displays.
(WhichFields)	:	For display objects that can reference different fields, this sets which of those fields it should select (e.g. select only Bus Name Fields). The value may be either ALL or SPECIFIED.
(ListOfFields)	:	If WhichFields is set to SPECIFIED, then a delimited list of fields follows.

Example:

```
<SUBDATA DynamicFormattingContextObject>
  // Note: CaseInfo applies to case information displays
  CaseInfo "SPECIFIED" BusName
  DisplayAreaField "ALL"
  DisplayBus
  DisplayBusField "SPECIFIED" BusName BusPUVolt BusNum
  DisplayCircuitBreaker
  DisplaySubstation
  DisplaySubstationField "SPECIFIED" SubName SubNum BusNomVolt BGLoadMVR
  DisplayTransmissionLine
  DisplayTransmissionLineField "ALL"
</SUBDATA>
```

LineThicknessLookupMap

LineColorLookupMap

FillColorLookupMap

FontColorLookupMap

FontSizeLookupMap

BlinkColorLookupMap

XoutColorLookupMap

FlowColorLookupMap

SecondaryFlowColorLookupMap

The values of the lookup table for the characteristics that can be modified by the dynamic formatting tool. The first line contains the two following fields:

fieldname	:	It is the field that the lookup table is going to look for.
usediscrete	:	Set to YES or NO. If set to YES, the characteristic values will be discrete, meaning that the characteristic value will correspond exactly to the one specified in the table. If set to NO, the characteristic values will be continuous, which means the characteristic value will be an interpolation of the high and low closest values specified in the table.

The following lines contain two fields:

fieldvalue	:	The value for the field.
characteristicvalue	:	The corresponding characteristic value for such field value.

Example:

```
<SUBDATA LineColorLookupMap>
   // FieldName UseDiscrete
     BusPUVolt YES
   // FieldValue Color
     1.02 16711808
     1.05 8454143
     1.1  16744703
</SUBDATA>
```

Filter

Condition

Conditions store the conditions of the filter. Each condition is described by one line of text which can contain up to five fields:

variablename : It is one of the fields for the object_type specified. It may optionally be followed by a colon and a non-negative integer. If not specified, 0 is assumed.

Example: on a LINE, 0 = from bus, 1 = to bus

sgLineMW:0 = the MW flow leaving the from bus

sgLineMW:1 = the MW flow leaving the to bus

Note: this value may also be the string "_UseAnotherfilter" which would then be followed by either meets or notmeets and then the name of another Filter.

Condition :

Possible Values	Alternate1	Alternate2	Requires othervalue
between	><		yes
notbetween	~><		yes
equal	=	==	
notequal	<>	~=	
greaterthan	>		
lessthan	<		
greaterthanorequal	>=		
lessthanorequal	<=		
about			yes
notabout			yes
contains			
notcontains			
startswith			
notstartswith			
inrange			
notinrange			
meets			
notmeets			
isblank			
notisblank			

value : The value used for comparison.
For fields associated with strings, this must be a string.
For fields associated with real numbers, this must be a number.
For fields associated with integers, this is normally an integer, except when the Condition is "inrange" or "notinrange". In this case, value is a comma/dash separated number string.

(othervalue) : If required, the other value used for comparison. For conditions "about" and "notabout" this is the tolerance with which the value should be equal or not equal.

(FieldOpt) : Optional string with following meanings. Unspecified means that strings are case insensitive, use number fields directly (older files may have had an integer 0 as well).

ABS : strings are case sensitive, take absolute value of field values (older files may have had an integer 1 as well)

Example:

```
FILTER (objecttype, filtername, filtertype, prefilter)
{
BUS "a bus filter" "AND" "no"
   <SUBDATA CONDITION>
      BusNomVolt  >  100
      AreaNum   inrange  "1 – 5 , 7 , 90-95"
      ZoneNum   between
   </SUBDATA>
BRANCH "a branch filter" "OR" "no"
   <subdata CONDITION>
      BusNomVolt:0  > 100    // Note location 0 means from bus
      BusNomVolt:1  > 100    // Note location 1 means to bus
      LineMW:0      > 100 1  // Note, final field 1 denotes absolute value
      _UseAnotherFilter meets
   </SUBDATA>
}
```

Gen

BidCurve

BidCurve subdata is used to define a piece-wise linear cost curve (or a bid curve). Each bid point consists of two real numbers on a single line of text: a MW output and then the respective bid (or marginal cost).

Example:

```
<SUBDATA BidCurve>
  // MW   Price[$/MWhr]
  100.00  10.6
  200.00  12.4
  400.00  15.7
  500.00  16.0
</SUBDATA>
```

ReactiveCapability

Reactive Capability subdata is used to the reactive capability curve of the generator. Each line of text consists of three real numbers: a MW output, and then the respective Minimum MVAR and Maximum MVAR output.

Example:

```
<SUBDATA ReactiveCapability>
  // MW   MinMVAR   MaxMVAR
  100.00  -60.00   60.00
  200.00  -50.00   50.00
  400.00  -30.00   20.00
  500.00  - 5.00    2.00
</SUBDATA>
```

Note: ReactiveCapability object types can also be directly created inside their own DATA section as well. Two of the key fields of the object are then the bus number and generator ID of the generator to which the ReactiveCapability point belongs.

GeoDataViewStyle

TotalAreaValueMap

This subdata section is used to define the lookup table for determining the total area size of geographic data view objects based on the value of a selected field. Two values are entered for each mapping:

FieldValue	:	Value of the field selected for the Total Area attribute.
TotalArea	:	The total area size of the object.

Example:

```
<SUBDATA TotalAreaValueMap>
// FieldValue TotalArea
1.000 0
4.000 23
7.000 46
</SUBDATA>
```

RotationRateValueMap

This subdata section is used to define the lookup table for determining the rotation rate of geographic data view objects based on the value of a selected field. Two values are entered for each mapping:

FieldValue	:	Value of the field selected for the Rotation Rate attribute.
RotationRate	:	The rotation rate of the object. Entered in Hz.

Example:

```
<SUBDATA RotationRateValueMap>
// FieldValue RotationRate
1.000 0.00
4.000 0.10
7.000 0.20
</SUBDATA>
```

RotationAngleValueMap

This subdata section is used to define the lookup table for determining the rotation angle of geographic data view objects based on the value of a selected field. Two values are entered for each mapping:

FieldValue	:	Value of the field selected for the Rotation Angle attribute.
RotationAngle	:	The rotation angle of the object. Entered in degrees.

Example:

```
<SUBDATA RotationAngleValueMap>
// FieldValue RotationAngle
1.000 -90.0
4.000 0.0
7.000 90.0
</SUBDATA>
```

LineThicknessValueMap

This subdata section is used to define the lookup table for determining the thickness of the border line around geographic data view objects based on the value of a selected field. Two values are entered for each mapping:

FieldValue	:	Value of the field selected for the Line Thickness attribute.
LineThickness	:	The line thickness of the border line around the object. This should be an integer value.

Example:

```
<SUBDATA LineThicknessValueMap>
// FieldValue LineThickness
1.000 1
4.000 2
7.000 3
</SUBDATA>
```

GlobalContingencyActions

CTGElementAppend

This format is the same as for the Contingency objecttype except that the SolvePowerFlow action is not allowed.

CTGElement

This format is the same as for the Contingency objecttype except that the SolvePowerFlow action is not allowed.

Note: GlobalContingencyActionsElement object types can also be directly created inside their own DATA section as well.

HintDefValues

HintObject

Stores the values for the custom hints. Each line has one value:

FieldDescription	:	This is a string enclosed in double quotes. The string itself is delimited by the @ character. The string contains five values:
		Name of Field : The name of the field. Special fields that appear on dialog by default have special names. Otherwise these are the same as the fieldnames of the AUX file format (for the "other fields" feature on the dialogs).
		Total Digit : Number of total digits for a numeric field.
		Decimal Points : Number of decimal points for a numeric field.
		Include Suffix : Set to 0 for not including the suffix, and set to 1 to include it.
		Field Preffix : The prefix text.

Example:

```
<SUBDATA HintObject>
     "BusPUVolt@4@1@1@PU Volt ="
     "BusAngle@4@1@1@Angle ="
</SUBDATA>
```

InjectionGroup

PartPoint

A participation point is used to describe the contents of an injection group. Each participation point lists six values:

- PointType : One of five values describing the type of point.
  - GEN : a generator
  - LOAD : a load
  - SHUNT : a switched shunt
  - BUS : a bus
  - INJECTIONGROUP: another injection group
- PointBusNum : The bus number of the partpoint if the type is a GEN, LOAD, SHUNT, or BUS. Value will be blank for an injection group type. Note: bus# values may be replaced by a string enclosed in double quotes where the string is the name of the bus followed by an underscore character and then the nominal voltage of the bus. These values may also be replaced by a string enclosed in double quotes that represents the label of the bus or a string representing the label of the generator, load, or switched shunt.
- PointID : For GEN, LOAD, or SHUNT type, this is the id for the partpoint. For an INJECTIONGROUP type, this is the name or label of the injection group. This is blank for a BUS type.
- PointParFac : The participation factor for the point.
- ParFacCalcType : How the participation factor is calculated. There are several options depending on the PointType.
  - Generators : SPECIFIED, MAX GEN INC, MAX GEN DEC, or MAX GEN MW
  - Loads : SPECIFIED or LOAD MW
  - Shunts : SPECIFIED, MAX SHUNT INC, MAX SHUNT DEC, or MAX SHUNT MVAR
  - Bus : SPECIFIED
  - Injection Groups : SPECIFIED

  All PointTypes can also set their participation factor based on a field associated with the device. To specify this, the tag `<Field>` should be followed by the variable name of the field: `<Field>`*variablename*. All PointTypes can also set their participation factor based on a Model Expression. To specify this, the tag `<Expression>` should be followed by the name of the Model Expression: `<Expression>`*ModelExpression*.
- ParFacNotDynamic : Should the participation factor be recalculated dynamically as the system changes.

Example:

```
<SUBDATA PartPoint>
  "GEN"  1 "1"   1.00 "SPECIFIED"   "NO"
  "GEN"  4 "1" 104.96 "MAX GEN INC" "NO"
  "GEN"  6 "1"  50.32 "MAX GEN DEC" "YES"
  "GEN"  7 "1" 600.00 "MAX GEN MW"  "NO"
  "LOAD" 2 "1"   5.00 "SPECIFIED"   "NO"
  "LOAD" 6 "1" 200.00 "LOAD MW"     "YES"
</SUBDATA>
```

Note: PartPoint object types can also be directly created inside their own DATA section as well. One of the key fields of the PartPoint object is then the name of the injection group to which the participation point belongs.

Interface

InterfaceElement

A interfaces's subdata contains a list of the elements in the interface. Each line contains a text descriptions of the interface element. Note that this text description must be encompassed by quotation marks. There are eleven kinds of elements allowed in an interface. Please note that the direction specified in the monitoring elements is important.

"BRANCH num1 num2 ckt"
: Monitor the MW flow on the branch starting from bus num1 going to bus num2 with circuit ckt. (order of bus numbers defines the direction)

"AREA num1 num2" : Monitor the sum of the AC branches that connect area1 and area2.

"ZONE num1 num2" : Monitor the sum of the AC branches that connect zone1 and zone2.

"BRANCHOPEN num1 num2 ckt"
: When monitoring the elements in this interface, monitor them under the contingency of opening this branch.

"BRANCHCLOSE num1 num2 ckt"
: When monitoring the elements in this interface, monitor them under the contingency of closing this branch.

"DCLINE num1 num2 ckt"
: Monitor the flow on a DC line.

"INJECTIONGROUP 'name'"
: Monitor the net injection from an injection group (generation contributes as a positive injection, loads as negative).

"GEN num1 id" : Monitor the net injection from a generator (output is positive injection)

"LOAD num1 id" : Monitor the net injection from a load (output is negative injection).

"MSLINE num1 num2 ckt"
: Monitor the MW flow on the multi-section line starting from bus num1 going to bus num2 with circuit ckt.

"INTERFACE 'name' " : Monitor the MW flow on the interface given by name.

"GENOPEN num1 id" : When monitoring the elements in this interface, monitor them under the contingency of opening this generator.

"LOADOPEN num1 id": When monitoring the elements in this interface, monitor them under the contingency of opening this load.

Note: bus# values may be replaced by a string enclosed in single quotes where the string is the name of the bus followed by an underscore character and then the nominal voltage of the bus. Labels may also be use as follows.

- bus# values for all element types may be replaced by a string enclosed in single quotes where the string is the label of the bus.
- for GEN or LOAD elements, the section `num1 id` may be replaced by the device's label.
- For MSLINE, DCLINE, or BRANCH elements, the `num1 num2 ckt` section may be replaced by the device's label.

For the interface element type `BRANCH num1 num2 ckt` and `DCLINE num1 num2 ckt`, an optional field can also be written specifying whether the flow should be measured at the far end. This field is either YES or NO.

Example:

```
<SUBDATA InterfaceElement
  "BRANCH  8   9 1" NO  // monitor the flow from bus 8 to bus 9 on this branch

  "BRANCH 12  33 1" YES // monitor the flow from bus 12 to bus 33 on branch
                        // measurefarend is set to true, therefore, we are
                        // monitoring the MW flow that arrives at bus 33
  // the following demonstrates the format when bus names and
  // nominal voltages are used.
  "BRANCH 'Twelve_230' 'name33_230' 1" YES

  "AREA    2   1"       // monitor tie line flow from area 2 to area 1
  "ZONE   66  53"       // monitor tie lines flows from zone 66 to zone 53
  "BRANCHOPEN  5  6 1"  // does monitoring after branch opens
  "BRANCHCLOSE 7 10 1"  // does monitoring after branch closes
  "GENOPEN" 55 1"       // does monitoring after generator opens
</SUBDATA>
```

Note: InterfaceElement object types can also be directly created inside their own DATA section as well. One of the key fields of the InterfaceElement object is then the name of the interface to which the interface element belongs.

KMLExportFormat

DataBlockDescription

This subdata section is used to describe the objects and fields that should be saved to a KML file. Same format as for the AUXFileExportFormatData subdata section.

LimitSet

LimitCost

LimitCost records describe the piece-wise unenforceable constraint cost records for use by unenforceable line/interface limits in the OPF or SCOPF. Each row contains two values

PercentLimit	:	Percent of the transmission line limit.
Cost	:	Cost used at this line loading percentage value.

Example:

```
<SUBDATA LimitCost>
  //Percent  Cost [$/MWhr]
    100.00    50.00
    105.00   100.00
    110.00   500.00
</SUBDATA>
```

Load

BidCurve

BidCurve subdata is used to define a piece-wise linear benefit curve (or a bid curve). Each bid point consists of two real numbers on a single line of text: a MW output and then the respective bid (or marginal cost). These costs must be increasing for loads.

Example:

```
<SUBDATA BidCurve>
  // MW   Price[$/MWhr]
  100.00  16.0
  200.00  15.7
  400.00  12.4
  500.00  10.6
</SUBDATA>
```

LPVariable

LPVariableCostSegment

Stores the cost segments for the LP variables. Each line contains four values:

Cost (Up)	:	Cost associated with increasing the LP variable.
Minimum value	:	Minimum limit of the LP variable.
Maximum value	:	Maximum limit of the LP variable.
Artificial	:	Whether the cost segment is artificial or not.

Example:

```
<SUBDATA LPVariableCostSegment>
   //Cost(Up)           Minimum         Maximum Artificial
  -20000.0000 -10000000000.5801         -0.6000 YES
      16.2343           -0.6000          0.0000 NO
      16.5526            0.0000          0.6000 NO
      16.8708            0.6000          1.2000 NO
      17.1890            1.2000          1.8000 NO
      17.5073            1.8000          2.4000 NO
   20000.0000            2.4000 9999999999.4199 YES
</SUBDATA>
```

ModelCondition

Condition

ModelConditions are the combination of an object and a Filter. They are used to return when the particular object meets the filter specified. As a result, the subdata section here mostly identical to the Condition subdata section of a Filter. See the description there. There is one exception however with the FieldOpt which has additional strings

(FieldOpt) : Optional string with following meanings:

- ABS : strings are case sensitive, take ABS of field values (older files may have had an integer 1 as well)
- REF : Means that the variablename is evaluated in the contingency reference state
- ABSREF : means that both ABS and REF are being used

Nothing specified means that strings are case insensitive, use number fields directly and values are evaluated as normal (older files may have had an integer 0 as well)

ModelExpression

LookupTable

LookupTables are used inside Model Expressions sometimes. These lookup table represent either one or two dimensional tables. If the first string in the SUBDATA section is "x1x2", this signals that it is a two dimensional lookup table. From that point on it will read the first row as "x2" lookup points, and the first column in the remainder of the rows as the x1 lookup values.

Example:

```
MODELEXPRESSION (CustomExpression,ObjectType,CustomExpressionStyle,
CustomExpressionString,WhoAmI,VariableName,WhoAmI:1,VariableName:1)
{
// The following demonstrated a one dimensional lookup table
22.0000, "oneD", "Lookup", "", "Gen<KEY1>1</KEY1><KEY2>1</KEY2>",
"Gen<KEY1>1</KEY1><KEY2>1</KEY2><VAR>GenMW</VAR>", "", ""
   <SUBDATA LookupTable>
     // because it does not start with the string x1x2 this will
     // represent a one dimensional lookup table
     x1        value
       0.000000    1.000000
      11.000000   22.000000
     111.000000  222.000000
   </SUBDATA>
0.0000, "twod", "Lookup", "",
"Gen<KEY1>1</KEY1><KEY2>1</KEY2>",
"Gen<KEY1>1</KEY1><KEY2>1</KEY2><VAR>GenMW</VAR>",
"Gen<KEY1>6</KEY1><KEY2>1</KEY2>",
"Gen<KEY1>6</KEY1><KEY2>1</KEY2><VAR>GenMW</VAR>"
   <SUBDATA LookupTable>
     // because this starts with x1x2 this represent a two dimensional
     // lookup table.  The first column represents lookup values for x1.
     // The first row represents lookup values for x2
     x1x2          0.100000    0.300000  // these are lookup heading for x2
       0.000000    1.000000    3.000000
      11.000000   22.000000   33.000000
     111.000000  222.000000  333.000000
   </SUBDATA>
}
```

ModelFilter

ModelCondition

A Model Filter's subdata contains a list of each ModelCondition in the filter. Because a list of Model Conditions is stored within Simulator, this subdata section only requires the name of each ModelCondition on each line and whether or not the condition is using the NOT operator as part of the Model Filter.

Example:

```
<SUBDATA ModelCondition>
// ModelConditionName NotCondition
  "Name of First Model Condition" "NO"
  "Name of Second Model Condition" "NO"
  "Name of Third Model Condition" "NO"
</SUBDATA>
```

MTDCRecord

An example of the entire multi-terminal DC transmission line record is given at the end of this record description. Each of the SUBDATA sections is discussed first.

MTDCBus

For this SUBDTA section, each DC Bus is described on a single line of text with exactly 8 fields specified.

DCBusNum	:	The number of the DC Bus. Note DC bus numbers are independent AC bus numbers.
DCBusName	:	The name of the DC bus enclosed in quotes.
ACTerminalBus	:	The AC terminal to which this DC bus is connected (via a MTDCConverter). If the DC bus is not connected to any AC buses, then specify as zero. You may also specify this as a string enclosed in double quotes with the bus name followed by an underscore character, following by the nominal voltage of the bus.
DCResistanceToground	:	The resistance of the DC bus to ground. Not used by Simulator.
DCBusVoltage	:	The DC bus voltage in kV.
DCArea	:	The area that this DC bus belongs to.
DCZone	:	The zone that this DC bus belongs to.
DCOwner	:	The owner that this DC bus belongs to.

MTDCBus object types can also be directly created inside their own DATA section as well. One of the key fields of the object is then the number of the MTDCRecord to which the MTDCBus belongs.

MTDCConverter

For this SUBDTA section, each AC/DC Converter is described by exactly 24 field which may be spread across several lines of text. Simulator will keep reading lines of text until it finds 24 fields. All text to the right of the 24th field (on the same line of text) will be ignored. The 24 fields are listed in the following order:

BusNum	:	AC terminal bus number.
MTDCNBridges	:	Number of bridges for the converter.
MTDCConvEBas	:	Converter AC base voltage.
MTDCConvAngMxMn	:	Converter firing angle.
MTDCConvAngMxMn:1	:	Converter firing angle max.
MTDCConvAngMxMn:2	:	Converter firing angle min.
MTDCConvComm	:	Converter commutating resistance.
MTDCConvComm:1	:	Converter commutating reactance.
MTDCConvXFRat	:	Converter transformer ratio.
MTDCFixedACTap	:	Fixed AC tap.
MTDCConvTapVals	:	Converter tap.
MTDCConvTapVals:1	:	Converter tap max.
MTDCConvTapVals:2	:	Converter tap min.
MTDCConvTapVals:3	:	Converter tap step size.
MTDCConvSetVL	:	Converter setpoint value (current or power).
MTDCConvDCPF	:	Converter DC participation factor.
MTDCConvMarg	:	Converter margin (power or current).
MTDCConvType	:	Converter type.
MTDCMaxConvCurrent	:	Converter Current Rating.
MTDCConvStatus	:	Converter Status.
MTDCConvSchedVolt	:	Converter scheduled DC voltage.

MTDCConvIDC : Converter DC current.
MTDCConvPQ : Converter real power.
MTDCConvPQ:1 : Converter reactive power.

MTDCConverter object types can also be directly created inside their own DATA section as well. One of the key fields of the object is then the number of the MTDCRecord to which the MTDCConverter belongs.

MTDCTransmissionLine

For this SUBDATA section, each DC Transmission Line is described on a single line of text with exactly 5 fields specified:

DCFromBusNum : From DC Bus Number.
DCToBusNum : To DC Bus Number.
CKTID : The DC Circuit ID.
Resistance : Resistance of the DC Line in Ohms.
Inductance : Inductance of the DC Line in mHenries (Not used by Simulator).

Example:

```
MTDCRECORD (Num,Mode,ControlBus)
{
//-------------------------------------------------------------------------
// The first Multi-Terminal DC Transmission Line Record
//-------------------------------------------------------------------------
1    "Current"   "SYLMAR3 (26098)"
   <SUBDATA Bus>
       //-----------------------------------------------------------------
       // DC Bus data must appear on a single line of text
       // The data consists of exactly 8 values
       // DC Bus Num, DC Bus Name, AC Terminal Bus, DC Resistance to ground,
       // DC Bus Voltage, DC Bus Area, DC Bus Zone, DC Bus Owner
         3  "CELILO3P"         0  9999.00   497.92    40   404     1
         4  "SYLMAR3P"         0  9999.00   439.02    26   404     1
         7  "DC7"          41311  9999.00   497.93    40   404     1
         8  "DC8"          41313  9999.00   497.94    40   404     1
         9  "DC9"          26097  9999.00   439.01    26   404     1
        10  "DC10"         26098  9999.00   439.00    26   404     1
   </SUBDATA>
   <SUBDATA Converter>
       //-----------------------------------------------------------------
       // convert subdata keeps reading lines of text until it has found
        // values specified for 24 fields.  This can span any number of lines
        // any values to the right of the 24th field found will be ignored
        // The next converter will continue on the next line.
       //-----------------------------------------------------------------
       41311   2   525.00   20.25   24.00    5.00     0.0000    16.3100
            0.391048  1.050000  1.000000  1.225000  0.950000  0.012500
            1100.0000  1650.0000   0.0000 "Rect"  1650.0000 "Closed"
            497.931  1100.0000   547.7241   295.3274
       41313   4   232.50   15.36   17.50    5.00     0.0000     7.5130
            0.457634  1.008700  1.030000  1.150000  0.990000  0.010000
            2000.0000  2160.0000   0.1550 "Rect"  2160.0000 "Closed"
            497.940  2000.0000   995.8800   561.8186
       26097   2   230.00   20.90   24.00    5.00     0.0000    16.3100
            0.892609  1.000000  1.100000  1.225000  0.950000  0.012500
           -1100.0000  1650.0000 ""       "Inv"   1650.0000 "Closed"
            439.009  1100.0000  -482.9099   274.5227
       26098   4   232.00   17.51   20.00    5.00     0.0000     7.5130
            0.458621  1.008700  1.100000  1.120000  0.960000  0.010000
            439.0000  2160.0000 ""       "Inv"   2160.0000 "Closed"
            439.000  1999.9999  -878.0000   544.2775
   </SUBDATA>
   <SUBDATA TransmissionLine>
       //-----------------------------------------------------------------
       // DC Transmission Segment information appears on a single line of
       // text.  It consists of exactly 5 value
```

```
       // From DCBus, To DCBus, Circuit ID, Line Resistance, Line Inductance
       //-------------------------------------------------------------------
       3      4    "1"        19.0000    1300.0000
       7      3    "1"         0.0100       0.0000
       8      3    "1"         0.0100       0.0000
       9      4    "1"         0.0100       0.0000
      10      4    "1"         0.0100       0.0000
   </SUBDATA>
//-----------------------------------------------------------------------
// A second Multi-Terminal DC Transmission Line Record
//-----------------------------------------------------------------------
2    "Current"   "SYLMAR4 (26100)"
   <SUBDATA Bus>
         5  "CELILO4P"         0  9999.00   497.92    40   404     1
         6  "SYLMAR4P"         0  9999.00   439.02    26   404     1
        11  "DC11"         41312  9999.00   497.93    40   404     1
        12  "DC12"         41314  9999.00   497.94    40   404     1
        13  "DC13"         26099  9999.00   439.01    26   404     1
        14  "DC14"         26100  9999.00   439.00    26   404     1
   </SUBDATA>
   <SUBDATA Converter>
       41312   2   525.00   20.26   24.00    5.00     0.0000    16.3100
            0.391048  1.050000  1.000000  1.225000  0.950000  0.012500
            1100.0000  1650.0000   0.0000 "Rect"  1650.0000 "Closed"
            497.931  1100.0000   547.7241   295.3969
       41314   4   232.50   15.45   17.50    5.00     0.0000     7.5130
            0.457634  1.008700  1.030000  1.150000  0.990000  0.010000
            2000.0000  2160.0000   0.1550 "Rect"  2160.0000 "Closed"
            497.940  2000.0000   995.8800   562.9448
       26099   2   230.00   20.90   24.00    5.00     0.0000    16.3100
            0.892609  1.000000  1.100000  1.225000  0.950000  0.012500
           -1100.0000  1650.0000 ""       "Inv"   1650.0000 "Closed"
            439.009  1100.0000  -482.9099   274.5227
       26100   4   232.00   17.51   20.00    5.00     0.0000     7.5130
            0.458621  1.008700  1.100000  1.120000  0.960000  0.010000
            439.0000  2160.0000 ""       "Inv"   2160.0000 "Closed"
            439.000  1999.9999  -878.0000   544.2775
   </SUBDATA>
   <SUBDATA TransmissionLine>
       5      6    "1"        19.0000    1300.0000
      11      5    "1"         0.0100       0.0000
      12      5    "1"         0.0100       0.0000
      13      6    "1"         0.0100       0.0000
      14      6    "1"         0.0100       0.0000
   </SUBDATA>
}
```

MTDCTransmissionLine object types can also be directly created inside their own DATA section as well. One of the key fields of the object is then the number of the MTDCRecord to which the MTDCTransmissionLine belongs.

MultiSectionLine

Bus

A multi section line's subdata contains a list of each dummy bus, starting with the one connected to the From Bus of the MultiSectionLine and proceeding in order to the bus connected to the To Bus of the Line. Note: bus# values may be replaced by a string enclosed in double quotes where the string is the name of the bus followed by an underscore character and then the nominal voltage of the bus, or the string may represent the label of the bus.

Example:

```
//-----------------------------------------------------------------------
//  The following describes a multi-section line that connnects bus
//    2 - 1 - 5 - 6 - 3
//-----------------------------------------------------------------------
MultiSectionLine (BusNum, BusName, BusNum:1, BusName:1,
                  LineCircuit, MSLineNSections, MSLineStatus)
{
2 "Two" 3 "Three" "&1" 2 "Closed"
   <SUBDATA Bus>
     1
     5
     6
   </SUBDATA>
}
```

BusRenumber

This subdata section allows renumbering of the dummy buses. The entries in the subdata section must be the new bus number that should be assigned to each dummy bus followed by the name of the new bus. The entries can be either space or comma delimited. The bus number must be specified, but the name is optional. If the name is not included and a new bus needs to be created, the name will be the same as the number. If an incorrect number of dummy buses is entered for a multi-section line, none of the dummy buses will be updated for that line. If a dummy bus number is specified that matches an existing bus that is another dummy bus, the other dummy bus will be assigned to a new bus number and the current dummy bus will be assigned to the number specified in the data.

Example:

```
MultiSectionLine (BusNum, BusNum:1, LineCircuit)
{
1 2 "1"
   <SUBDATA BusRenumber>
     3 "Bus 3"
     4 "Bus 4"
     5 "Bus 5"
   </SUBDATA>
22 33 "1"
   <SUBDATA BusRenumber>
     14 "Bus 14"
     15 "Bus 15"
   </SUBDATA>
}
```

Nomogram

InterfaceElementA

InterfaceElementB

InterfaceElementA values represent the interface elements for the first interface of the nomogram. InterfaceElementB values represent the interface elements for the second interface of the nomogram. The format of these SUBDATA sections is identical to the format of the InterfaceElement SUBDATA section of a normal Interface.

NomogramBreakPoint

This subdata section contains a list of the vertex points on the nomogram limit curve.

Example:

```
<SUBDATA NomogramBreakPoint>
 //  LimA   LimB
     -100    -20
     -100    100
       80     50
       60    -10
</SUBDATA>
```

NomogramInterface

InterfaceElement

This follows the same convention as the InterfaceElement SUBDATA section described with the Interface objecttype.

Owner

Bus

This subdata section contains a list of the buses which are owned by this owner. Each line of text contains the bus number. As an alternative to specifying the bus number, a string enclosed in double quotes may be used where the string represents the name of the bus followed by an underscore character and then the nominal voltage of the bus, or the string may represent the label of the bus.

Example:

```
<SUBDATA Bus>
   1
  35
  65
</SUBDATA>
```

Load

This subdata section contains a list of the loads which are owned by this owner. Each line of text contains the bus number followed by the load id. As an alternative to specifying the bus number, a string enclosed in double quotes may be used where the string represents the name of the bus followed by an underscore character and then the nominal voltage of the bus, or the string may represent the label of the bus. Also, instead of specifying the bus and load id, the label of the load enclosed in double quotes may be used.

Example:

```
<SUBDATA Load>
    5 1  // shows ownership of the load at bus 5 with id of 1
  423 1
</SUBDATA>
```

Gen

This subdata section contains a list of the generators which are owned by this owner and the fraction of ownership. Each line of text contains the bus number, followed by the gen id, followed by an integer showing the fraction of ownership. As an alternative to specifying the bus number, a string enclosed in double quotes may be used where the string represents the name of the bus followed by an underscore character and then the nominal voltage of the bus, or the string may represent the label of the bus. Also, instead of specifying the bus and generator id, the label of the generator enclosed in double quotes may be used.

Example:

```
<SUBDATA Gen>
  78 1  50  // shows 50% ownership of generator at bus 78 with id of 1
  23 3  70
</SUBDATA>
```

Branch

This subdata section contains a list of the branches which are owned by this owner and the fraction of ownership. Each line of text contains the from bus number, followed by the to bus number, followed by the circuit id, followed by an integer showing the fraction of ownership. As an alternative to specifying the bus numbers, strings enclosed in double quotes may be used where the string represents the name of the bus followed by an underscore character and then the nominal voltage of the bus, or the string may represent the label of the bus. Also instead of specifying the two numbers and a circuit id, the label of the branch enclosed in double quotes may be used.

Example:

```
<SUBDATA Branch>
  6  10  1  50  // shows 50% ownership of line from bus 6 to 10, circuit 1
</SUBDATA>
```

PostPowerFlowActions

CTGElementAppend

This format is the same as for the Contingency objecttype except that Abort, ContingencyBlock, and SolvePowerFlow actions are not allowed.

CTGElement

This format is the same as for the Contingency objecttype except that Abort, ContingencyBlock, and SolvePowerFlow actions are not allowed. PostPowerFlowActionsElement object types can also be directly created inside their own DATA section as well.

PWCaseInformation

PWCaseHeader

This subdata section contains the Case Description in free-formatted text. Note: as it is read back into Simulator all spaces from the start of each line are removed.

PWFormOptions

PieSizeColorOptions

There can actually be several PieSizeColorOptions subdata sections for each PWFormOptions object. The first line of each subdata section, the first line of text consist of exactly four values

ObjectName	:	The objectname of the type of object these settings apply to. Will be either be BRANCH or INTERFACE.
FieldName	:	The fieldname for the pie charts that these settings apply to.
UseDiscrete	:	Set to YES to use a discrete mapping of colors and size scalars instead of interpolating for intermediate values.
UseOtherSettings	:	Set to YES to default these settings to the BRANCH MVA values for BRANCH object. This allows you to apply the same settings to all pie charts.

After this first line of text, if the UseOtherSettings Value is NO, then another line of text will contain exactly three values:

ShowValue	:	This is the percentage at which the value should be drawn on the pie chart.

NormalSize : This is the scalar size multiplier which should be used for pie charts below the lowest percentage specified in the lookup table.

NormalColor : This is the color which should be used for pie charts below the lowest percentage specified in the lookup table.

Finally the remainder of the subdata section will contain a lookup table by percentage of scalar and color values. This lookup table will consist of consecutive lines of text with exactly three values

Percentage : This is the percentage at which the follow scalar and color should be applied.

Scalar : A scalar (multiplier) on the size of the pie charts.

Color : A color for the pie charts.

Example:

```
<SUBDATA PieSizeColorOptions>
   // ObjectName FieldName UseDiscrete UseOtherSettings
   Branch  MVA  YES  NO
   // ShowValue NormalSize NormalColor
    80.0000   1.0000 16776960
   // Percentage Scalar Color
    80.0000   1.5000 33023
   100.0000   2.0000 255
</SUBDATA>
<SUBDATA PieSizeColorOptions>
   // ObjectName FieldName UseDiscrete UseOtherSettings
   Branch  MW  YES  YES
</SUBDATA>
```

PWLPOPFCTGViol

OPFControlSense

OPFBusSenseP

OPFBusSenseQ

This stores the control sensitivities for each contingency violation during OPF/SCOPF analysis. Each line contains one value:

Sensitivity : The value of the sensitivity with respect to each control in OPFControlSense or with respect to each bus in OPFBusSenseP and OPFBusSenseQ.

Example:

```
<SUBDATA OPFControlSense>
 // Value
   1.000441679
   2.447185E-7
  -1.1109307E-6
   1.6427327E-7
   0
</SUBDATA>
```

PWLPTabRow

LPBasisMatrix

This subdata section stores the basis matrix associated with the final LP OPF solution. Each line contains two values:

Variable : The basic variable.

Value : The sensitivity of the constraint to the basic variable.

Example:

```
<SUBDATA LPBasisMatrix>
 // Var  Value
     1   1.00000
     2   1.00000
     5   1.00000
     6   1.00000
</SUBDATA>
```

PWPVResultListContainer

PWPVResultObject

This subdata section contains the results of a particular PV Curve scenario. The data consists of two general sections: the first three rows of text contain the "independent axis" of the PV Curve. The first row starts with the string INDNOM and is followed by a list of numbers representing the nominal shift, the second row starts with INDEXP and is followed by the export shift, and the third row starts with INDIMP and is followed by the import shift. Following after these rows is a list of all the tracked quantities. Each tracked quantity row consists of three parts which are separated by the strings ?f= and &v= . The first part of the string represents a description of the power system object being tracked, the second part represents the field variable name being tracked, and the third contains a list of all the values at the various shift levels.

Example:

```
<SUBDATA PWPVResultObject>
  INDNOM                   0.00  500.00 1000.00 1500.00 1750.00 1875.00 1975.00
  INDEXP                   0.00  500.00 1000.00 1500.00 1750.00 1875.00 1975.00
  INDIMP                   0.00 -417.23 -701.58 -890.58 -952.60 -975.35 -990.43
  Bus '3'?f=BusPUVolt&v=  0.993   0.983   0.964   0.939   0.926   0.919   0.914
  Bus '5'?f=BusPUVolt&v=  1.007   1.000   0.982   0.956   0.940   0.932   0.926
  Gen '4' '1'?f=GenMVR&v= 19.99  245.27  523.62  831.13  986.84  1060.6  1118.7
  Gen '6' '1'?f=GenMVR&v= -6.59 -120.84 -131.37  -39.53   48.35   103.8   154.5
</SUBDATA>
```

LimitViol

This subdata section contains the limit violations of a particular PV Curve scenario. This subdata section would only exist if using the option to monitor limit violations with the PV tool. Each row consists of an identifier, either VLOW or VHIGH, to indicate the type of limit violation followed by the bus identifier based on the key field identifier chosen. The bus can be identified by number, name and nominal kV combination, or label. The bus identifier is followed by the limit in use to identify a voltage violation and this is followed by the voltage at the bus.

Example:

```
<SUBDATA LimitViol>
  VLOW 3    1.00000    0.99017
  VLOW 5    1.00000    0.98245
</SUBDATA>
```

PVBusInadequateVoltages

This subdata section contains a list of buses that are considered to have inadequate voltages at each transfer level for a particular PV Curve scenario. This subdata section would only exist if using the option to store inadequate voltages. The data consists of two general sections: the first row starts with the string INDNOM and is followed by a list of numbers representing the nominal shift. The second and subsequent rows list the buses and inadequate voltages for any bus that has an inadequate voltage at any transfer level. Each row starts with the bus identifier followed by the voltages at that bus at the corresponding shift levels. If a voltage is not inadequate at a particular transfer level, a blank entry will appear instead of a voltage value. The bus identifier is based on the key field identifier chosen and can be number, name and nominal kV combination, or label.

Example:

```
<SUBDATA PVBusInadequateVoltages>
    // INDNOM ShiftLevel1 ShiftLevel2 ...
    // BUS Voltage1 Voltage2 ...
    INDNOM    0.000  100.000  200.000  300.000  400.000  500.000
    "Bus '3'"   0.99269   0.99278   0.99282   0.99280   0.99273   0.99262
    "Bus '4'" ""   1.00000 "" "" ""
</SUBDATA>
```

PWQVResultListContainer

PWPVResultObject

This subdata section contains the results of a particular QV Curve scenario. These results will exist when tracking quantities with the QV curve tool. The data consists of two general sections: the first three rows of text contain the "independent axis" of the QV Curve. The first three rows start with the strings INDNOM, INDEXP, and INDIMP and are followed by a list of numbers representing the setpoint voltage representing the V of the QV curve. Following after these rows is a list of all the tracked quantities. Each tracked quantity row consists of three parts which are separated by the strings ?f= and &v= . The first part of the string represents a description of the power system object being tracked, the second part represents the field variable name being tracked, and the third contains a list of all the values at the various setpoint voltage levels.

Example:

```
<SUBDATA PWPVResultObject>
    INDNOM    1.100    1.093    1.083    1.073    1.063
    INDEXP    1.100    1.093    1.083    1.073    1.063
    INDIMP    1.100    1.093    1.083    1.073    1.063
    Bus '1'?f=BusPUVolt&v=1.05000 1.05000 1.05000 1.05000 1.05000
    Bus '1'?f=BusKVVolt&v=144.89999 144.89999 144.89999 144.89999 144.89999 </SUBDATA>
```

QVCurve

QVPoints

This subdata section contains a list of the QV Curve points calculated for the respect QVCurve. Each line consists of exactly six values:

PerUnitVoltage	:	The per unit voltage of the bus for a QV point.
FictitiousMvar	:	The amount of Mvar injection from the fictitious generator at this QV point.
ShuntDeviceMvar	:	The Mvar injection from any switched shunts at the bus.
TotalMvar	:	The total Mvar injection from switched shunts and the fictitious generator.
ReservesMvar	:	Total amount of Mvar reserves available at the bus.
ReservesTotalMvar	:	Total Mvar injection from the switched shunts, fictitious generator, and available reserves.

Example:

```
QVCURVE (BusNum,CaseName,qv_VQ0,qv_Q0,qv_Vmax,qv_QVmax,qv_VQmin,qv_Qmin,
         qv_Vmin,qv_QVmin,Qinj_Vmax,Qinj_0,Qinj_min,Qinj_Vmin)
{
5 "BASECASE"  0.880   0.000   1.100 312.490    0.480 -221.072
              0.180 -86.334 191.490 -77.373 -244.075  -89.562
   <SUBDATA QVPoints>
     // NOTE: This bus has a constant impedance
     // switched shunt value of -100 Mvar at it.
     //V(PU), Q(MVR), Q_shunt(MVR), Q_tot(MVR), Q_res(MVR), Q_tot_res(MVR)
     1.1000,  312.4898, -121.0000,  191.4898,  0.0000,  191.4898
     0.9800,  124.6619,  -95.9656,   28.6963,  0.0000,   28.6963
     0.7800,  -96.6202,  -60.7808, -157.4010,  0.0000, -157.4010
     0.5800, -206.9895,  -33.5960, -240.5855,  0.0000, -240.5855
     0.3800, -207.4962,  -14.4113, -221.9075,  0.0000, -221.9075
   </SUBDATA>
}
```

QVCurve_Options

Sim_Solution_Options

This subdata section contains solution options that will be used when running QV Curves. See explanation under the CTG_Options object type for more information.

RemedialAction

CTGElementAppend

This format is the same as for the Contingency objecttype except that the SolvePowerFlow action is not allowed.

CTGElement

This format is the same as for the Contingency objecttype except that the SolvePowerFlow action is not allowed. RemedialActionElement object types can also be directly created inside their own DATA section as well.

SelectByCriteriaSet

SelectByCriteriaSetType

This subdata section contains a list of the display object types which are chosen to be selected. Each line of the section consists of the following:

DisplayObjectType	:	The object type of the display object.
(FilterName)	:	This field is optional, but must be given if either of the following fields is given. See the Using Filters in Script Commands section for more information on specifying the filtername.
(WhichFields)	:	For display objects that can reference different fields, this sets which of those fields it should select (e.g. select only Bus Name Fields). The value may be either ALL or SPECIFIED.
(ListOfFields)	:	If WhichFields is set to SPECIFIED, then a delimited list of fields follows.

Example:

```
<SUBDATA SelectByCriteriaSetType>
  DisplayAreaField "" "ALL"
  DisplayBus ""
  DisplayBusField "Name of Bus Filter" "SPECIFIED" BusName BusPUVolt BusNum
  DisplayCircuitBreaker ""
  DisplaySubstation ""
  DisplaySubstationField "" "SPECIFIED" SubName SubNum BusNomVolt BGLoadMVR
  DisplayTransmissionLine ""
  DisplayTransmissionLineField "" "ALL"
</SUBDATA>
```

Area

This subdata section contains a list of areas which were chosen to be selected. Each line of the section consists of either the number or the name. When generated automatically by PowerWorld we also include the other identifier as a comment.

Example:

```
<SUBDATA Area>
  18 // NEVADA
  22 // SANDIEGO
  30 // PG AND E
  52 // AQUILA
</SUBDATA>
```

Zone

This subdata section contains a list of zones which were chosen to be selected. Each line of the section consists of either the number or the name. When generated automatically by PowerWorld we also include the other identifier as a comment.

Example:

```
<SUBDATA Zone>
  680 // ID SOLUT
  682 // WY NE IN
</SUBDATA>
```

ScreenLayer

This subdata section contains a list of screen layers which were chosen to be selected. Each line of the section consists of either the name.

Example:

```
<SUBDATA ScreenLayer>
  "Border"
  "Transmission Line Objects"
</SUBDATA>
```

ShapefileExportDescription

This object uses the same subdata sections as **SelectByCriteriaSet**. The only distinction is that only buses and lines can be exported.

StudyMWTransactions

ImportExportBidCurve

This subdata section contains the piecewise linear transactions cost curves for areas involved in a MW transaction. Costs are only for areas that are not on OPF control. Curves must be monotonically increasing. Each line corresponds to a point in the cost curve, and it has two values:

MW	:	The MW value. Use negative values for imports (purchase) and positive values for exports (sales)
Price	:	The price in $/MWh.

Two different cost curves can be entered for each transaction. One is for the cost curve relative to the Export Area specified in the transaction, and the other is for the cost curve relative to the Import Area specified in the transaction. The first curve that is listed in the SUBDATA section is the curve relative to the Export Area. The curve relative to the Import Area is denoted by the keyword REVERSE. Either or both of the curves can be blank.

Example:

```
<SUBDATA ImportExportBidCurve>
  //MW   Price[$/MWh]
  -20.00   5.00
  -10.00  10.00
    0.00  15.00
   10.00  20.00
   20.00  45.00
   30.00  70.00
  REVERSE
  -25.00   7.00
  -15.00  12.00
    5.00  17.00
   15.00  22.00
   25.00  47.00
   35.00  72.00
</SUBDATA>
```

SuperArea

SuperAreaArea

This subdata section contains a list of areas within each super area. Each line of text contains two values, the area number followed by a participation factor for the area that can be optionally used.

Example:

```
<SUBDATA SuperAreaArea>
   1   48.9
   5   34.2
  25   11.2
</SUBDATA>
```

TSSchedule

SchedPoint

This section stores the schedule time points used in Time Step Simulation. Each line contains seven values:

Date	:	The date of the point.
Hour	:	The hour of the point.
Pointtype	:	An integer specifying the point type. 0 : Numeric 1 : Boolean (Yes/No, Closed/Open) 2 : Text
Numeric Value	:	The numeric value if point type is Numeric. Otherwise it is just zero.
Boolean Value	:	The boolean value if point type is Boolean. Otherwise it is just false.
Text value	:	The text value if point type is Text. Otherwise it is just an empty string.
Audiofilename	:	The audio filename associated to the point. If none, it is just an empty string.

Example:

```
<SUBDATA SchedPoint>
   //Date   Hour     PointType  NValue  BValue  TValue  AValue
   5/8/2006              0      1.00    NO
   5/8/2006 6:00:00 AM   0      1.10    NO
   5/8/2006 12:00:00 PM  0      1.25    NO
</SUBDATA>
```

UserDefinedDataGrid

ColumnInfo

This follows the same convention as the ColumnInfo SUBDATA section described with the DataGrid objecttype.

SCRIPT Section for Display Auxiliary File

The syntax for script commands in Display Auxiliary Files is the same as for Auxiliary Files. See the **SCRIPT Section** and its sub-sections for details on the proper syntax. Any differences for display auxiliary files will be discussed below.

AXD Actions

The following script commands are available for AXD files

```
AutoInsertBorders;
AutoInsertBuses                (LocationSource, MapProjection, AutoInsertBranches,
                               InsertIfNotAlreadyShown, "filename", InsertSelected);
AutoInsertInterfaces           (InsertPieCharts, PieChartSize);
AutoInsertLineFlowObjects      (MinkV, InsertOnlyIfNotAlreadyShown, LineLocation, Size,
                               FieldDigits, FieldDecimals, TextPosition, ShowMW, ShowMvar,
                               ShowMVA, ShowUnits, ShowComplex);
AutoInsertLineFlowPieCharts    (MinkV, InsertOnlyIfNotAlreadyShown, InsertMSLines, Size);
AutoInsertLines                (MinkV, InsertTextFields, InsertEquivObjects, InsertZBRPieCharts,
                               InsertMSLines, SBRImpedance, NoStubsZBRs, SingleCBZRs);
AutoInsertLoads                (MinkV, InsertTextFields, InsertEquivObjects);
AutoInsertGens                 (MinkV, InsertTextFields);
AutoInsertSwitchedShunts       (MinkV, InsertTextFields);
AutoInsertSubstations          (LocationSource, MapProjection, AutoInsertBranches,
                               InsertIfNotAlreadyShown, "FileName", InsertSelected);
AutoInsertAreas                (MapProjection, InsertIfNotAlreadyShown, InsertSelected);
AutoInsertZones                (MapProjection, InsertIfNotAlreadyShown, InsertSelected);
AutoInsertOwners               (MapProjection, InsertIfNotAlreadyShown, InsertSelected);
AutoInsertInjectionGroups      (MapProjection, InsertIfNotAlreadyShown, InsertSelected);
FixFlowArrowLineEnds           ("OnelineName", "LayerName");
FixFlowArrowPosition           ("OnelineName", "LayerName");
InsertConnectedBuses           ("BusIdentifier");
LoadAXDFromAXD                 ("filename", CreateIfNotFound);
PanAndZoomToObject             ("ObjectID", "DisplayObjectType", "DoZoom");
ResetStubLocations             (ZBRImpedance,NoStubsZBRs);
```

AutoInsertBorders;

Use this action to automatically insert borders according to the settings in the AutoInsertBordersOptions object

AutoInsertBuses(LocationSource, MapProjection, AutoInsertBranches, InsertIfNotAlreadyShown, "filename", InsertSelected);

Use this action to automatically insert buses based on specified location data.

LocationSource : "Bus", "Substation" or "File"
MapProjection : "Simple Conic", "Mercator", "Alaska" or "xy"
AutoInsertBranches : YES to insert transmission lines when finished, NO not to
InsertOnlyIfNotAlreadyShown
: YES if only buses that are not already shown should be inserted, NO to insert all buses.
"filename" : (optional) path to location source file (if LocationSource is "File")

Modified in April 1, 2020 patch of Simulator 21

FileCoordinates is no longer used. It should not be included when creating new auxiliary files, but if it is included in existing auxiliary files, it will be read and ignored. If MapProjection is set to "xy" and using a file, the file coordinates are assumed to be in x,y, otherwise, file coordinates are assumed to be lon, lat.

FileCoordinates : (optional) format of coordinates in file "xy" or "lonlat" (if LocationSource is "File")
InsertSelected : (optional) Default is NO. YES is only insert buses that are selected (SELECTED = YES).

AutoInsertInterfaces(InsertPieCharts, PieChartSize);

Use this action to automatically insert line flow objects.

InsertPieCharts : (optional) Insert pie charts as well (default=YES)
PieChartSize : (optional) default size of interface pie charts (default=50.0)

AutoInsertLineFlowObjects(MinkV, InsertOnlyIfNotAlreadyShown, LineLocation, Size, FieldDigits, FieldDecimals, TextPosition, ShowMW, ShowMvar, ShowMVA, ShowUnits, ShowComplex);

Use this action to automatically insert line flow objects.

MinkV : Minimum kV level to insert (default=0)
InsertOnlyIfNotAlreadyShown: (optional) if existing line flow objects are ignored (default=YES)
LineLocation : (optional) where to insert flow objects (default=0)
0 : middle
1 : 10%/90%
2 : after stubs
Size : (optional) size (default=5.0)
FieldDigits : (optional) total digits in field (default=6)
FieldDecimals : (optional) digits to the right of the decimal (default=2)
TextPosition : (optional) position of fields relative to flow object (default=YES)
YES : above
NO : below
ShowMW : (optional) show MW field (default=YES)
ShowMvar : (optional) show Mvar field (default=YES)
ShowMVA : (optional) show MVA field (default=YES)
ShowSuffix : (optional) show field units (default=YES)
ShowComplex : (optional) show complex form (MW+jMvar) (default=NO)

AutoInsertLineFlowPieCharts(MinkV, InsertOnlyIfNotAlreadyShown, InsertMSLines, Size);

Use this action to automatically insert line flow pie charts.

MinkV : Minimum kV level to insert (default=0)
InsertOnlyIfNotAlreadyShown
: (optional) if existing line flow objects are ignored (default=YES)
InsertMSLines : (optional) insert pie charts for Multi-Section Lines (default=YES)
Size : (optional) size (default=5.0)

AutoInsertLines(MinkV, InsertTextFields, InsertEquivObjects, InsertZBRPieCharts, InsertMSLines, ZBRImpedance, NoStubsZBRs, SingleCBZRs);

Use this action to automatically insert lines.

MinkV : (optional) minimum kV level to insert (default=0)
InsertTextFields : (optional) insert text fields (default=YES)
InsertEquivObjects : (optional) insert Equivalenced Objects (default=YES)
InsertZBRPieCharts : (optional) insert pie charts for lines with no limit and bus ties (default=NO)
InsertMSLines : (optional) insert MultiSecton Lines (default=YES)
ZBRImpedance : (optional) maximum PU impedance for bus ties (default =0.0001)
NoStubsZBRs : (optional) ignore stubs for bus ties (default=YES)
SingleCBZBRs : (optional) only insert a single circuit breaker (default=YES)

AutoInsertLoads(MinkV, InsertTextFields);

Use this action to automatically insert loads.

MinkV : Minimum kV level to insert (default=0)
InsertTextFields : (optional) insert text fields (default=YES)

AutoInsertSwitchedShunts(MinkV, InsertTextFields);

Use this action to automatically insert switched shunts.

MinkV : Minimum kV level to insert (default=0)
InsertTextFields : (optional) insert text fields (default=YES)

AutoInsertSubStations(LocationSource, MapProjection, AutoInsertBranches, InsertIfNotAlreadyShown, "filename", InsertSelected);

Use this action to automatically insert substations based on specified location data.

LocationSource : "Bus", "Substation" or "File"
MapProjection : "Simple Conic", "Mercator", "Alaska" or "x,y"
AutoInsertBranches : YES to insert transmission lines when finished, NO not to
InsertOnlyIfNotAlreadyShown
: YES if only buses that are not already shown should be inserted, NO to insert all buses.
"filename" : (optional) path to location source file (if LocationSource is "File")

Modified in July 31, 2020 patch of Simulator 21

FileCoordinates is no longer used. It should not be included when creating new auxiliary files, but if it is included in existing auxiliary files, it will be read and ignored. If MapProjection is set to "xy" and using a file, the file coordinates are assumed to be in x,y, otherwise, file coordinates are assumed to be lon, lat.

FileCoordinates : (optional) format of coordinates in file "xy" or "lonlat" (if LocationSource is "File")
InsertSelected : (optional) Default is NO. YES is only insert buses that are selected (SELECTED = YES).

AutoInsertAreas(MapProjection, InsertIfNotAlreadyShown, InsertSelected);

Added in the May 18, 2022 patch of Simulator 22

Use this action to automatically insert areas based on the latitude and longitude of the area (which is calculated as the average lat/long of buses in the area).

MapProjection : "Simple Conic", "Mercator", "Alaska" or "x,y"
InsertOnlyIfNotAlreadyShown
: (optional) Default is NO. YES if only areas that are not already shown should be inserted, NO to insert all areas.
InsertSelected : (optional) Default is NO. YES is only insert areas that are selected (SELECTED = YES).

AutoInsertInjectionGroups(MapProjection, InsertIfNotAlreadyShown, InsertSelected);

Added in the May 18, 2022 patch of Simulator 22

Use this action to automatically insert injection groups based on the latitude and longitude of the injection group (which is calculated as the average lat/long of objects in the injection group).

MapProjection : "Simple Conic", "Mercator", "Alaska" or "x,y"
InsertOnlyIfNotAlreadyShown
: (optional) Default is NO. YES if only injection groups that are not already shown should be inserted, NO to insert all injection groups.
InsertSelected : (optional) Default is NO. YES is only insert injection groups that are selected (SELECTED = YES).

AutoInsertOwners(MapProjection, InsertIfNotAlreadyShown, InsertSelected);

Added in the May 18, 2022 patch of Simulator 22

Use this action to automatically insert owners based on the latitude and longitude of the owner (which is calculated as the average lat/long of objects in the owner).

MapProjection : "Simple Conic", "Mercator", "Alaska" or "x,y"
InsertOnlyIfNotAlreadyShown
: (optional) Default is NO. YES if only owners that are not already shown should be inserted, NO to insert all owners.
InsertSelected : (optional) Default is NO. YES is only insert owners that are selected (SELECTED = YES).

AutoInsertZones(MapProjection, InsertIfNotAlreadyShown, InsertSelected);

Added in the May 18, 2022 patch of Simulator 22

Use this action to automatically insert zones based on the latitude and longitude of the zone (which is calculated as the average lat/long of buses in the zone).

MapProjection : "Simple Conic", "Mercator", "Alaska" or "x,y"

InsertOnlyIfNotAlreadyShown : (optional) Default is NO. YES if only zones that are not already shown should be inserted, NO to insert all zones.

InsertSelected : (optional) Default is NO. YES is only insert zones that are selected (SELECTED = YES).

FixFlowArrowLineEnds("OnelineName", "LayerName");

The unmoving line flow arrow indicators that can be displayed on lines may not always be setup correctly. This script corrects the flow arrows so that they look at the end of the line to which they are the closest to determine the direction of flow.

"OnelineName" : Optional – default is blank
The name of the oneline to which this action should be applied. The oneline must already be open. When not specified this action is applied to the oneline to which the display auxiliary file has been applied.

"LayerName" : Optional – default is blank
Name of the screen layer in which this action should be applied. When this is left blank the action is applied to all flow arrow objects regardless of the layer.

FixFlowArrowPosition("OnelineName", "LayerName");

The unmoving line flow arrow indicators that can be displayed on lines are difficult to position properly. This script is intended to automate as much of the work as possible for positioning these objects.

"OnelineName" : Optional – default is blank
The name of the oneline to which this action should be applied. The oneline must already be open. When not specified this action is applied to the oneline to which the display auxiliary file has been applied.

"LayerName" : Optional – default is blank
Name of the screen layer in which this action should be applied. When this is left blank the action is applied to all flow arrow objects regardless of the layer.

InsertConnectedBuses("BusIdentifier");

Insert oneline display buses that are connected to the identified bus.

"BusIdentifier" : Identifier for the bus for which connected display buses should be added. Bus can be identified as a data object "BUS Number", "BUS NameNomkV", or "BUS Label". When identified as a data object all display buses that are linked to this data bus will have display buses inserted.
Bus can also be identified as a display object "DISPLAYBUS BusNum SOAuxiliaryID" or "DISPLAYBUS BusName_NomVolt". When identified as a display object only display buses connected to this bus will be inserted.

LoadAXDFromAXD("filename", CreateIfNotFound)

Added in the July 9, 2021 patch of Simulator 22

Use this action to apply a display auxiliary file via this script command that is part of another display auxiliary file.

"filename"	:	The file name of the display auxiliary file to load. See the Specifying File Names in Script Commands section for special keywords that can be used when specifying the file name.
CreateIfNotFound	:	Optional – default is NO Set to YES or NO. YES means that objects that cannot be found will be created while reading in DATA sections from the specified file. Legacy format data sections have a parameter that indicates how to handle objects that cannot be found. If this parameter is specified in the data section, this parameter with the script command is ignored. Concise format data sections will always create objects that cannot be found, which means that is script command parameter is always ignored. All required key fields must exist with a data section in order for new objects to be created.

PanAndZoomToObject("ObjectID", "DisplayObjectType", "DoZoom");

Use this action to pan to and optionally zoom in on a display object. This action will find the first matching display object linked to the model object identified by ObjectID or the exact display object identified by ObjectID.

ObjectID	:	The object identifier that uniquely identifies the power system model object or display object. For example, to identify bus 123354, the ObjectID takes the form "BUS 123354".
DisplayObjectType	:	(optional) The display object type to find. When a power system model object is passed in for ObjectID, this parameter helps this action narrow down to what kind of display object to pan. For example, to find the first bus display object, use "DISPLAYBUS". (default="")
DoZoom	:	(optional) Whether or not to zoom after panning. (default=YES)

ResetStubLocations(ZBRImpedance, NoStubsZBRs);

Use this action to reset stub locations.

ZBRImpedance	:	(optional) max P.U. impedance for bus ties (default=0.0001)
NoStubsZBRs	:	(optional) Ignore stubs for bus ties (default=YES)

General Script Commands

The following script commands defined above in the general SCRIPT section are available for display auxiliary files as well:

ExitProgram
LoadScript
LoadData
SelectAll
UnSelectAll
SetData
SaveData
SaveDataWithExtra
CreateData
DeleteFile
RenameFile
CopyFile
SetCurrentDirectory
SaveObjectFields

DATA Section for Display Auxiliary File

The syntax for Display Auxiliary Files is the same as for Auxiliary Files. See the **DATA Section** and its sub-sections for details on the proper syntax. Any differences for display auxiliary files will be discussed below.

Key Fields

See the **Key Fields** topic in the **DATA Section** for general details.

Display objects have an additional key field used for identification because multiple objects can be present on the same one-line diagram that represent the same power system element. This extra key field is `SOAuxiliaryID`. This is a field that is unique for each type of display object and other key field combination. If there are two display buses that represent bus one in the power system, the `SOAuxiliaryID` field will be different for both. Simulator will automatically create unique identifiers when these objects are created graphically. They can also be user specified but are forced to be unique. This field does not need to be present when reading in a display auxiliary file, but if it is missing, Simulator assumes that the ID is "1". This field is the only key field identifier for objects that do not link to power system elements such as background lines and pictures, and therefore, should always be included when reading in these objects or the expected results may not be achieved.

By going to the main menu and choosing Help, Export Display Object Fields you will obtain a list of fields available for each display object type. In this output, the key fields will appear with asterisks *.

Special Data Sections

There are several object types that should be noted here because they can impact the reading of an entire display auxiliary file, overall look of the resulting one-line diagram, or require special input to properly import/export the object.

GeographyDisplayOptions

Most objects supported in the display auxiliary file have coordinates that can be specified in the appropriate data sections. What these coordinates specify can be controlled by the GEOGRAPHYDISPLAYOPTIONS object. This object has only two fields available: MapProjection and ShowLonLat. There are four possible settings for MapProjection: `"x,y"`, `"Simple Conic"`, `"Alaska"`, and `"Mercator"`. The choice of projection will determine how the x,y values for display objects are interpreted. ShowLonLat can be either `"YES"` or `"NO"`. If ShowLonLat is `"YES"`, the setting specified for the MapProjection will be the longitude,latitude projection used when reading/writing the object x,y values. If ShowLonLat is `"NO"`, the x,y values will always be interpreted as x,y regardless of the MapProjection setting. This object should be placed in the display auxiliary file before any other objects containing coordinates are read. If this object is not included in the auxiliary file, the coordinates will be interpreted based on the current settings of map projection and whether or not coordinates are showing longitude,latitude.

Picture

PICTURE objects represent background images that cannot be stored in a text file format. To properly include a PICTURE object in a display auxiliary file, the file containing the image must be saved and read along with the auxiliary file. The `FileName` field indicates the name and location of the image file. If the image file cannot be found when reading in a display auxiliary file and attempting to create a new object, no PICTURE object will be created. If attempting to update an existing object and the image file cannot be found, the object will not be updated with a new image, but the `FileName` field will be updated with the specified file name.

PWFormOptions

One-line display options that affect the current display settings can be changed by using the PWFORMOPTIONS object. Usually, this object specifies named sets of options that can be selected and used to change the various one-line display options through the GUI. By including a specially named object, the current options can be changed through a display

auxiliary file. PWFORMOPTIONS are named using the `OOName` field. Setting this field to "THESE_OPTIONS_ARE_APPLIED_TO_THE_CURRENT_DISPLAY" will apply the specified set of options to the current one-line when the file is read. When saving the entire one-line to a display auxiliary file, a PWFORMOPTIONS object with this name is added to the file by default.

View

Different views can be specified in the display auxiliary file using the VIEW object. Usually, this object is used to specify named sets of options used to select and change the view through the GUI. By including a specially named object, the current view can be changed through a display auxiliary file. VIEW objects are named using the `ViewName` field. Setting this field to "THIS_VIEW_IS_APPLIED_TO_THE_CURRENT_DISPLAY" will apply the specified set of view options to the current one-line when the file is read. When saving the entire one-line to a display auxiliary file, a VIEW object with this name is added to the file by default.

SubData Sections

The format described thus far works well for most kinds of data in Simulator. It does not work as well however for data that stores a list of objects. For example, a contingency stores some information about itself (such as its name), and then a list of contingency elements, and possible a list of limit violations as well. For data such as this, Simulator allows `<SubData>`, `</SubData>` tags that store lists of information about a particular object. This formatting looks like the following

```
object_type (list_of_fields)
{
value_list_1
    <SUBDATA subobject_type1>
      precise format describing an object_type1
      precise format describing an object_type1
      .
      .
      .
    </SUBDATA>
    <SUBDATA subobject_type2>
      precise format describing an object_type2
      precise format describing an object_type2
      .
      .
      .
    </SUBDATA>
value_list_2
   .
   .
   .
value_list_n
}
```

Note that the information contained inside the `<SubData>`, `</SubData>` tags may not be flexibly defined. It must be written in a precisely defined order that will be documented for each SubData type. The description of each of these SubData formats follows.

ColorMap

Same format as in data auxiliary files.

CustomColors

Same format as in data auxiliary files.

DisplayDCTramisssionLine

DisplayInterface

DisplayMultiSectionLine

DisplaySeriesCapacitor

DisplayTransformer

DisplayTransmissionLine

Line

Line

This is a list of points defining the graphical line used to represent the object. Each set of coordinates can be enclosed in square brackets, [ ], or the brackets can be eliminated. The brackets will be included when Simulator generates an auxiliary file. The individual coordinates are separated by the specified delimiter, either a space or a comma, and if the brackets are included, the same delimiter should be used to separate sets of coordinates. The list of points is in a somewhat free form and sets of coordinates can span multiple lines. Each point should either be in x,y coordinates or longitude,latitude coordinates. Which coordinates should be used depends on the current option settings for map projection and whether or not coordinates should be shown in longitude,latitude. If the display auxiliary file is automatically generated by Simulator, a comment will be included in the subdata section indicating the coordinate system in use during file creation.

Example using brackets and a comma delimiter:

```
<SUBDATA Line>
//Coordinates are x,y
  [14.00000000, 63.00000000], [14.00000000, 60.00000000],
  [20.00000000, 45.00000000], [20.00000000, 42.00000000]
</SUBDATA>
```

Example with no brackets and a space delimiter:

```
<SUBDATA Line>
//Coordinates are x,y
  14.00000000 63.00000000 14.00000000 60.00000000
  20.00000000 45.00000000 20.00000000 42.00000000
</SUBDATA>
```

DynamicFormatting

Same format as in data auxiliary files.

Filter

Same format as in data auxiliary files.

GeoDataViewStyle

Same format as in data auxiliary files.

PieChartGaugeStyle

ColorMap

This is a lookup table by percentage of scalar and color values. This lookup table will consist of consecutive lines of text with exactly three values:

Percentage	:	This is the percentage at which the following scalar and color should be applied.
Scalar	:	A scalar (multiplier) on the size of the pie chart/gauge.
Color	:	A color for the pie chart/gauge.

Example:

```
<SUBDATA ColorMap>
//Percentage  Scalar  Color
   85.0000    1.5000  33023
  100.0000    2.0000    255
</SUBDATA>
```

PWFormOptions

Same format as in data auxiliary files.

SelectByCriteriaSet

Same format as in data auxiliary files.

UserDefinedDataGrid

Same format as in data auxiliary files.

View

ScreenLayer

This is a list of screen layer names that are hidden in the current view. Each screen layer name is on a separate line of text.

Example:

```
<SUBDATA ScreenLayer>
//These are hidden screen layers
  "pie layer"
</SUBDATA>
```