COSMOSMotion API

Programming Reference

Mechanical Dynamics, Inc.
Design Technologies division
Copyright 2002
**IDDMAddin**

**Description:** This is the root object providing access to the Assemblies and their Mechanisms. Use GetObject() to obtain this object.

[Solid Edge] This object is only available after the Motion Environment has been entered at least once and before the Solid Edge session is closed.

[SolidWorks] This object is available when the DDM addin supporting automation is loaded.

**Properties**

**ActiveAssembly**

**Syntax:** Assem = Addin.ActiveAssembly

**Value:** IDDMAssembly object

**Description:** DDM Assembly corresponding to the active assembly document.

**ProductID**

**Syntax:** Id = Addin.ProductID

**Value:** see DDMProductId

**Description:** Current DDM product (Simply Motion, Motion, Motion Professional).

**IDDMAssembly**

**Description:** Corresponds to an assembly document in the current modeler. Used to access the Mechanism object.

**Properties**

**MechanismIsActive**

**Syntax:** Assembly.MechanismIsActive

**Value:** Boolean

**Description:**

[Solid Edge] If the assembly document is open in the Motion environment, the Mechanism is active and can be modified and exercised.

[SolidWorks] The mechanism of an open Assembly is always active.

**Mechanism**

**Syntax:** Assembly.Mechanism

**Value:** IDDMMechanism object

**Description:** Accesses Mechanism object. This will fail if there is no active Mechanism.

**IDDMMechanism**

**Description:** Contains all Mechanism Elements (Parts, Joints, Forces, Couplers, etc.), settings (Gravity, Integrator settings, etc.), and the current Simulation if one has been run.

**Properties**

**Elements**

**Get Syntax:** Mechanism.Elements

**Value:** IDDMElements

**Description:** Provides access to all Elements.
Plots
Syntax: Mechanism.Plots
Value: IDDMPlots
Description: Provides access to all plots.

Simulation
Syntax: Mechanism.Simulation
Value: IDDMSimulation
Description: Provides access to the current Simulation, if one exists. The property Simulated is True if there is a Simulation.

Integrator
Syntax: Mechanism.Integrator
Value: IDDMIntegrator
Description: Provides access to the integration parameters used to solve a simulation.

Gravity
Syntax: Mechanism.Gravity
Value: IDDMGravity object
Description: Accesses Gravity object.

ForceUnits
Syntax: force = Mechanism.ForceUnits
Value: see DDMForceUnits
Description: Get and put the force units used for simulation results.

TimeUnits
Syntax: force = Mechanism.TimeUnits
Value: see DDMTimeUnits
Description: Get and put the time units used for simulation creation and results.

Simulated
Syntax: Mechanism.Simulated
Value: Boolean
Description: True if a Simulation has been run.

UseStoredMasses
Syntax: useStored = Mechanism.UseStoredMasses
Value: Boolean
Description: If True, moving Part masses are not recalculated before a Simulation is run.

Methods
GetElementByID
Syntax: Mechanism.GetElementByID(id, retval)
Input: (Long) id Identifier of element
Output: (Object)retval Element found
Description: Gets element by its identifier.
GetElementByName

Syntax: Mechanism.GetElementByName(name, retval)

Input: (String) name Name of element

Output: (Object) retval Element found

Description: Gets element by name.

GetElementsOfType

Syntax: Mechanism.GetElementsOfType(type, retval)

Input: (Long) type Type of elements (DDMElementType_contants)

Output: (IDDMElements) retval Elements found

Description: Gets elements of type.

GetPartBySource

Syntax: Mechanism.GetPartBySource(source, part)

Input: (String) name Source object of element.

[Solid Edge] Use Occurrence.

[SolidWorks] Use Component.

Output: (DDMPart) part Part found

Description: Gets Part corresponding to a part object from the modeler.

AddJoint

Syntax: Mechanism.AddJoint(name, type, partI, partJ, retval)

Input: (String) name Name of element

(Long) type Type of joint (DDMJointType)

(Object) partI First of two parts connected by the new element

(Object) partJ Second of two parts connected by the new element

Output: (Object) retval New joint object

Description: Adds a new element to the Mechanism. The element must be finished by setting its position and other parameters.

AddForceActionOnly

Syntax: Mechanism.AddForceActionOnly(name, type, part, relPart, retval)

Input: (String) name Name of element

(Long) type Type of force – Linear or Torsion

(Object) partI First of two parts connected by the new element

(Object) partJ Second of two parts connected by the new element

Output: (Object) retval New Action-Only Force element

Description: Adds a new element to the Mechanism. The element must be finished by setting its position and other parameters.

AddForceActionReaction

Syntax: Mechanism.AddForceActionReaction(name, type, partI, partJ, retval)

Input: (String) name Name of element

(Long) type Type of force – Linear or Torsion

(Object) partI First of two parts connected by the new element

(Object) partJ Second of two parts connected by the new element

Output: (Object) retval New Action-Reaction Force element

Description: Adds a new element to the Mechanism. The element must be finished by setting its position and other parameters.

AddBushing

Syntax: Mechanism.AddBushing(name, partI, partJ, retval)
Input: (String) name Name of element
(Object) partI First of two parts connected by the new element
(Object) partJ Second of two parts connected by the new element
Output: (Object) retval New Bushing element
Description: Adds a new element to the Mechanism. The element must be finished by setting its position and other parameters. By default only the translational properties are enabled and the torsional properties are disabled. The user can enable the torsional properties after creating this new bushing element.

AddSpring
Syntax: Mechanism.AddSpring(name, type, partI, partJ, retval)
Input: (String) name Name of element
(Long) type Type of spring – Linear or Torsion
(Object) partI First of two parts connected by the new element
(Object) partJ Second of two parts connected by the new element
Output: (Object) retval New Spring element
Description: Adds a new element to the Mechanism. The element must be finished by setting its position and other parameters.

AddDamper
Syntax: Mechanism.AddDamper(name, type, partI, partJ, retval)
Input: (String) name Name of element
(Long) type Type of damper – Linear or Torsion
(Object) partI First of two parts connected by the new element
(Object) partJ Second of two parts connected by the new element
Output: (Object) retval New Damper element
Description: Adds a new element to the Mechanism. The element must be finished by setting its position and other parameters.

AddImpact
Syntax: Mechanism.AddImpact(name, partI, partJ, retval)
Input: (String) name Name of element
(Object) partI First of two parts connected by the new element
(Object) partJ Second of two parts connected by the new element
Output: (Object) retval New Impact element
Description: Adds a new element to the Mechanism. The element must be finished by setting its position and other parameters.

AddPointCurve
Syntax: retval = Mechanism.AddPointCurve(name, partI, partJ, retval)
Input: (String) name Name of element
(Object) partI First of two parts connected by the new element
(Object) partJ Second of two parts connected by the new element
Return: (Object) retval New Point-Curve element
Description: Adds a new element to the Mechanism. The element must be finished by setting its position and other parameters.

AddCurveCurve
Syntax: Mechanism.AddCurveCurve(name, partI, partJ, retval)
Input: (String) name Name of element
(Object) partI First of two parts connected by the new element
(Object) partJ Second of two parts connected by the new element
Output: (Object) retval New Curve-Curve element
Description: Adds a new element to the Mechanism. The element must be finished by setting its position and other parameters.
Description: Adds a new element to the Mechanism. The element must be finished by setting its position and other parameters.

AddCoupler
Syntax: Mechanism.AddCoupler(name, joint1, joint2, retval)
Input: (String) name Name of element
(Object) joint1 First of two joints (Revolute, Translational, Cylindrical) connected by the new element
(Object) joint2 Second of two joints (Revolute, Translational, Cylindrical) connected by the new element
Output: (Object) retval New Coupler element
Description: Adds a new Coupler to the Mechanism.

AddTracePath
Syntax: Mechanism.AddTracePath(name, part, relPart, retval)
Input: (String) name Name of element
(Object) part Part containing the point to trace
(Object) partRel Part determining the relative motion of the point
Output: (Object) retval New Trace Path element
Description: Adds a new element to the Mechanism. The element must be finished by setting its point.

AddMotionOnPart
Syntax: retval = Mechanism.AddMotionOnPart(name, partI, partJ, retval)
Input: (String) name Name of element
(Object) partI First of two parts connected by the new element
(Object) partJ Second of two parts connected by the new element
Output: (Object) retval New Part Motion element
Description: Adds a new element to the Mechanism. The element must be finished by setting its position and motion parameters.

AddPlot
Syntax: retval = Mechanism.AddPlot(result, component, elemID, retval)
Input: (DDMPlotResult) result Type of result required. See DDMPlotResult.
(DDMResults Component) component Type of component required. X, Y, Z or Magnitude.
(Long) elemID ID of the element whose plot has to be plotted.
Output: (Object) retval New Plot element
Description: Adds a plot to the Mechanism with time as the x-axis and the result as the single y-axis. The x-axis change be changed using DMPlot SetXAxis().

DeleteElement
Syntax: Mechanism.DeleteElement(elem)
Input: (Object) elem Element to be deleted.
Description: Removes the element from the Mechanism.

Simulate
Syntax: Mechanism.Simulate(time, frames)
Input: (Double) time Duration of simulation in Mechanism’s time units
(Long) frames Number of simulation frames
Description: Simulates the Mechanism.
CalculateDegreeOfFreedom
Syntax: Mechanism.CalculateDegreeOfFreedom
Return: (int) dof  Degrees of freedom remaining in Mechanism.
Description: Calculates the Mechanism’s degree of freedom, removing redundant constraints. The
result will be zero if the Mechanism is fixed or higher.

DeleteSimulation
Syntax: Mechanism.DeleteSimulation
Description: Deletes the current Simulation, enabling Mechanism edit and simulate operations.

ExportADAMSModelFile
Syntax: Mechanism.ExportADAMSModelFile(fileName)
Value: (String)fileName  Give the full path along with the file name
Description: Exports the ADAMS file (.adm) to the given location.

ExportADAMSCommandFile
Syntax: Mechanism.ExportADAMSCommandFile(fileName)
Value: (String)fileName  Give the full path along with the file name
Description: Exports the ADAMS file (.adm), Results file (.res) and the Command file (.cmd) to the
given location.

IDDMIntegrator
Description: Provides access to the solver integration parameters.

Properties
Type
Get Syntax: type = Integrator.Type
Put Syntax: Integrator.Type = DDMIntegrator_GSTIFF
Value: see DDMIntegratorType
Description: GSTIFF is the default.

MinimumTimeStep
Get Syntax: step = Integrator.MinimumTimeStep
Put Syntax: Integrator.MinimumTimeStep = 0.0000001
Value: Double
Description: Specifies the minimum step used by the variable-step Integrator.

MaximumTimeStep
Get Syntax: step = Integrator.MaximumTimeStep
Put Syntax: Integrator.MaximumTimeStep = 0.0000001
Value: Double
Description: Specifies the maximum step used by the variable-step Integrator.

MaximumIterations
Get Syntax: maxIterations = Integrator.MaximumIterations
Put Syntax: Integrator.MaximumIterations = 25
Value: Long
Description: Specifies the maximum number of times the Integrator iterates in the search for the
solution to a given time step.
JacobianPattern
Get Syntax: jacobianPattern = Integrator.JacobianPattern
Put Syntax: Integrator.JacobianPattern = 30
Value: Long
Description: Controls how often the Integrator’s jacobian engine is updated. The value is a percent. At 100%, the jacobian is updated at every integration time step. At 50%, the jacobian is updated at every other integration time step. This value impacts models with intermittent curve/curve contacts. The higher the value, the more accurate the simulation and the more time required to complete.

Adaptivity
Get Syntax: adaptivity = Integrator.Adaptivity
Put Syntax: Integrator.Adaptivity = 10e-6
Value: Double
Description: Used primarily for intermittent curve/curve contact problems. If problems occur where contacts are not detected, this value can be progressively decreased. A good starting point is 10e-6 with reductions of 10e-1.

InitialTimeStep
Get Syntax: step = Integrator.InitialTimeStep
Put Syntax: Integrator.InitialTimeStep = 1e-4
Value: Double
Description: This should be small and is usually of little concern unless the model is locked in a position that the solver is needlessly trying to locate with very high precision. Setting the value to a higher number will cause the solver to stop its search sooner.

IDDMGravity
Accessors: IDDMMechanism.Gravity
Description: Implements gravity in the model, allowing it to be turned off or changed.

Properties
Enabled
Get Syntax: enabled = Gravity.Enabled
Put Syntax: Gravity.Enabled = True
Value: Boolean
Description: Turns gravity on and off.

Acceleration
Get Syntax: accel = Gravity.Acceleration
Put Syntax: Gravity.Acceleration = 9810#
Value: Double
Description: Sets the acceleration due to gravity along gravity vector. The units are length / time^2, where length and time are in the units of the Mechanism.

Methods
SetVector
Syntax: Gravity.SetVector(x, y, z)
Input: (Double) x X component of vector
       (Double) y Y component of vector
(Double) \( z \)  
**Description:** Sets the gravity vector.

**GetVector**

**Syntax:**  
Gravity.GetVector(x, y, z)

**Input:**  
(Double) \( x \)  \( X \) component of vector  
(Double) \( y \)  \( Y \) component of vector  
(Double) \( z \)  \( Z \) component of vector

**Description:** Gets the gravity vector.

**IDDMSimulation**

**Accessors:** IDDMMechanism.Simulation

**Description:** Provides results of a simulation and gives control over replay and export of results.

**Properties**

**Frames**

**Get Syntax:** frames = Simulation.Frames

**Value:** Long

**Description:** Gets the number of frames in the simulation.

**Time**

**Get Syntax:** time = Simulation.Time

**Value:** Double

**Description:** Gets the duration time of the simulation. This is equal to the time parameter of Mechanism.Simulate, unless the simulation was aborted by the user or by an error.

**CurrentFrame**

**Get Syntax:** frame = Simulation.CurrentFrame

**Put Syntax:** Simulation.CurrentFrame = frame

**Value:** Double

**Description:** Gets the required current frame after the simulation. The user can also Put a particular frame as the current frame. The value –1 signifies that there is no current frame and the model is not positioned by results. Frame 1 is the first frame at time 0.

**CurrentTime**

**Get Syntax:** time = Simulation.CurrentTime

**Put Syntax:** Simulation.CurrentTime = time

**Value:** Double

**Description:** Gets the required current time. And also the user can enter the current time.

**Methods**

**Play**

**Syntax:** Play(fwd, incr)

**Input:**  
(BOOL) \( fwd \) True to animate in forward direction else False  
(Long) \( incr \) 1 to fast forward the animation. 0 to animate at default speed

**Description:** Play animation.
ExportAVIMovie

**Syntax:** ExportAVIMovie(from, to, inc, fileName)

**Input:**
- (Long) from: frame number to start from
- (Long) to: frame number to stop at
- (Long) inc: increment between frames
- (String) fileName: name of the file

**Description:** Exports an animation (.avi) file.

ExportVRML2File

**Syntax:** ExportVRML2File(useTime, time, fileName)

**Input:**
- (Long) useTime: 1 if user wants to use time other than simulation time, 0 for default simulation time.
- (Long) time: Used when useTime = 1. User can specify time other than simulation time. Thus the user can data of simulation which is faster/slower than default.
- (String) fileName: name of the file

**Description:** Exports an VRML (.wrl) file.

ExportADAMSResultsFile

**Syntax:** ExportADAMSResultsFile(fileName)

**Input:**
- (String) fileName: name of the file

**Description:** Exports a Results (.res) file.

GetTranslationalDisplacement

**Syntax:** GetTranslationalDisplacement (component, frame, elemID, csysID, refID, value)

**Input:**
- (Long) component: Component to measure. See DDMResultsComponent.
- (Long) frame: Simulation frame of requested results. (1 to number of simulated frames)
- (Long) elemID: ID of part/marker to measure.
- (Long) csysID: Coordinate system in which vector components are expressed. By default, the coordinate system is the global coordinate system. This must be a part or marker ID.
- (Long) refID: Motion reference frame relative to which time-derivatives are performed. This must be a part or marker ID.

**Output:** (Double) Translational displacement component.

**Description:** Returns translational displacement component of a part or marker.

GetTranslationalVelocity

**Syntax:** GetTranslationalDisplacement (component, frame, elemID, csysID, refID, value)

**Input:**
- (Long) component: Component to measure. See DDMResultsComponent.
- (Long) frame: Simulation frame of requested results. (1 to number of simulated frames)
- (Long) elemID: ID of part/marker to measure.
- (Long) csysID: Coordinate system in which vector components are expressed. By default, the coordinate system is the global coordinate system. This must be a part or marker ID.
- (Long) refID: Motion reference frame relative to which time-derivatives are performed. This must be a part or marker ID.

**Output:** (Double) Translational velocity component.

**Description:** Returns translational velocity component of a part or marker.
GetTranslationalAcceleration
Syntax: GetTranslationalDisplacement (component, frame, elemID, csysID, refID, value)
Input: (Long) component Component to measure. See DDMResultsComponent.
(Long) frame Simulation frame of requested results. (1 to number of simulated frames)
(Long) elemID ID of part/marker to measure.
(Long) csysID Coordinate system in which vector components are expressed.
By default, the coordinate system is the global coordinate system. This must be a part or marker ID.
(Long) refID Motion reference frame relative to which time-derivatives are performed. This must be a part or marker ID.
Output: (Double) Translational acceleration component.
Description: Returns translational acceleration component of a part or marker.

GetAngularVelocity
Syntax: GetAngularVelocity (component, frame, elemID, csysID, refID, value)
Input: (Long) component Component to measure. See DDMResultsComponent.
(Long) frame Simulation frame of requested results. (1 to number of simulated frames)
(Long) elemID ID of part/marker to measure.
(Long) csysID Coordinate system in which vector components are expressed.
By default, the coordinate system is the global coordinate system. This must be a part or marker ID.
(Long) refID Motion reference frame relative to which time-derivatives are performed. This must be a part or marker ID.
Output: (Double) Angular velocity.
Description: Returns angular velocity component of a part or marker.

GetAngularAcceleration
Syntax: GetAngularAcceleration (component, frame, elemID, csysID, refID, value)
Input: (Long) component Component to measure. See DDMResultsComponent.
(Long) frame Simulation frame of requested results. (1 to number of simulated frames)
(Long) elemID ID of element to measure.
(Long) csysID Coordinate system in which vector components are expressed.
By default, the coordinate system is the global coordinate system. This must be a part or marker ID.
(Long) refID Motion reference frame relative to which time-derivatives are performed. This must be a part or marker ID.
Output: (Double) Angular acceleration.
Description: Returns angular acceleration component.

GetTotalForceOnPoint
Syntax: GetTotalForceOnPart (component, frame, markerID, csysID, value)
Input: (Long) component Component to measure. See DDMResultsComponent.
(Long) frame Simulation frame of requested results. (1 to number of simulated frames)
(Long) markerID ID of marker to measure.
(Long) csysID Coordinate system in which vector components are expressed.
By default, the coordinate system is the global coordinate system. This must be a part or marker ID.
Output: (Double) Force component.
Description: Returns the force component of a marker on the marker’s part.
GetTotalForceAtLocation

Syntax: GetTotalForceAtLocation (component, frame, markerID, csysID, value)

Input:
- (Long) component: Component to measure. See DDMResultsComponent.
- (Long) frame: Simulation frame of requested results. (1 to number of simulated frames)
- (Long) markerID: ID of marker to measure.
- (Long) csysID: Coordinate system in which vector components are expressed. By default, the coordinate system is the global coordinate system. This must be a part or marker ID.

Output: (Double) Force component.

Description: Returns the sum of all forces of all elements sharing a point specified by a marker on that marker’s part.

GetTotalTorqueOnPoint

Syntax: GetTotalTorqueOnPoint (component, frame, markerID, csysID, value)

Input:
- (Long) component: Component to measure. See DDMResultsComponent.
- (Long) frame: Simulation frame of requested results. (1 to number of simulated frames)
- (Long) markerID: ID of marker to measure.
- (Long) csysID: Coordinate system in which vector components are expressed. By default, the coordinate system is the global coordinate system. This must be a part or marker ID.

Output: (Double) Torque component.

Description: Returns the torque component of a marker on the marker’s part.

GetTotalTorqueAtLocation

Syntax: GetTotalTorqueAtLocation (component, frame, markerID, csysID, value)

Input:
- (Long) component: Component to measure. See DDMResultsComponent.
- (Long) frame: Simulation frame of requested results. (1 to number of simulated frames)
- (Long) markerID: ID of marker to measure.
- (Long) csysID: Coordinate system in which vector components are expressed. By default, the coordinate system is the global coordinate system. This must be a part or marker ID.

Output: (Double) Torque component.

Description: Returns the sum of all torques of all elements sharing a point specified by a marker on that marker’s part.

GetPowerConsumption

Syntax: GetPowerConsumption (frame, elemID, motionIndex, value)

Input:
- (Long) frame: Simulation frame of requested results. (1 to number of simulated frames)
- (Long) elemID: ID of element with motions.
- (Long) motionIndex: X translation = 0
  Y translation = 1
  Z translation = 2
  X rotation = 3
  Y rotation = 4
  Z rotation = 5

Output: (Double) Power consumption.

Description: Returns the power consumption of a motion applied to Joint or PartMotion element’s degree of freedom. The x, y, z of the motionIndex is in the Joint or PartMotion coordinate system.
GetPotentialEnergyDelta

Syntax: GetPotentialEnergyDelta (frame, partID, value)
Input: (Long) frame Simulation frame of requested results. (1 to number of simulated frames)
       (Long) partID Part ID.
Output: (Double) Potential energy.
Description: Measures the potential energy of a part.

GetTranslationalKineticEnergy

Syntax: GetTranslationalKineticEnergy (frame, partID, value)
Input: (Long) frame Simulation frame of requested results. (1 to number of simulated frames)
       (Long) partID Part ID.
Output: (Double) Translational energy.
Description: Measures the translational energy of a part.

GetAngularKineticEnergy

Syntax: GetAngularKineticEnergy (frame, partID, value)
Input: (Long) frame Simulation frame of requested results. (1 to number of simulated frames)
       (Long) partID Part ID.
Output: (Double) Angular kinetic energy.
Description: Measures the angular kinetic energy of a part.

GetKineticEnergy

Syntax: GetKineticEnergy (frame, partID, value)
Input: (Long) frame Simulation frame of requested results. (1 to number of simulated frames)
       (Long) partID Part ID.
Output: (Double) Kinetic energy.
Description: Measures the kinetic energy of a part.

GetTranslationalMomentum

Syntax: GetTranslationalMomentum (component, frame, partID, csysID, refID, value)
Input: (Long) component Component to measure. See DDMResultsComponent.
       (Long) frame Simulation frame of requested results. (1 to number of simulated frames)
       (Long) partID ID of part to measure.
       (Long) csysID Coordinate system in which vector components are expressed. By default, the coordinate system is the global coordinate system. This must be a part or marker ID.
       (Long) refID Motion reference frame relative to which time-derivatives are performed. This must be a part or marker ID.
Output: (Double) Translational momentum.
Description: Returns translational momentum component of a part.

GetAngularMomentumAboutCM

Syntax: GetAngularMomentumAboutCM (component, frame, partID, csysID, refID, value)
Input: (Long) component Component to measure. See DDMResultsComponent.
       (Long) frame Simulation frame of requested results. (1 to number of simulated frames)
       (Long) partID ID of part to measure.
LONG csysID Coordinate system in which vector components are expressed. By default, the coordinate system is the global coordinate system. This must be a part or marker ID.

LONG refID Motion reference frame relative to which time-derivatives are performed. This must be a part or marker ID.

Output: DOUBLE Translational momentum about center of mass of part.

Description: Returns translational momentum component of a part about its center of mass.

GetCMPosition
Syntax: GetCMPosition (component, frame, partID, csysID, refID, value)
Input: (LONG) component Component to measure. See DDMResultsComponent.
(Long) frame Simulation frame of requested results. (1 to number of simulated frames)
(Long) partID ID of part to measure.
(Long) csysID Coordinate system in which vector components are expressed. By default, the coordinate system is the global coordinate system. This must be a part or marker ID.
(Long) refID Motion reference frame relative to which time-derivatives are performed. This must be a part or marker ID.

Output: DOUBLE Component position.

Description: Returns component of part’s center of mass position.

GetCMVelocity
Syntax: GetCMVelocity (component, frame, partID, csysID, refID, value)
Input: (LONG) component Component to measure. See DDMResultsComponent.
(Long) frame Simulation frame of requested results. (1 to number of simulated frames)
(Long) partID ID of part to measure.
(Long) csysID Coordinate system in which vector components are expressed. By default, the coordinate system is the global coordinate system. This must be a part or marker ID.
(Long) refID Motion reference frame relative to which time-derivatives are performed. This must be a part or marker ID.

Output: DOUBLE Velocity component.

Description: Returns component of part’s center of mass velocity.

GetCMAcceleration
Syntax: GetCMAcceleration (component, frame, partID, csysID, refID, value)
Input: (LONG) component Component to measure. See DDMResultsComponent.
(Long) frame Simulation frame of requested results. (1 to number of simulated frames)
(Long) partID ID of part to measure.
(Long) csysID Coordinate system in which vector components are expressed. By default, the coordinate system is the global coordinate system. This must be a part or marker ID.
(Long) refID Motion reference frame relative to which time-derivatives are performed. This must be a part or marker ID.

Output: DOUBLE Acceleration component.

Description: Returns component of part’s center of mass acceleration.

GetCMAngularVelocity
Syntax: GetCMAngularVelocity (component, frame, partID, csysID, refID, value)
Input: (LONG) component Component to measure. See DDMResultsComponent.
(Long) frame  Simulation frame of requested results. (1 to number of simulated frames)
(Long) partID  ID of part to measure.
(Long) csysID  Coordinate system in which vector components are expressed. By default, the coordinate system is the global coordinate system. This must be a part or marker ID.
(Long) refID  Motion reference frame relative to which time-derivatives are performed. This must be a part or marker ID.

Output:  (Double)  Angular velocity component.
Description:  Returns component of part’s center of mass angular velocity.

GetCMAngularAcceleration
Syntax:  GetCMAngularAcceleration (component, frame, partID, csysID, refID, value)
Input:  (Long) component  Component to measure. See DDMResultsComponent.
(Long) frame  Simulation frame of requested results. (1 to number of simulated frames)
(Long) partID  ID of part to measure.
(Long) csysID  Coordinate system in which vector components are expressed. By default, the coordinate system is the global coordinate system. This must be a part or marker ID.
(Long) refID  Motion reference frame relative to which time-derivatives are performed. This must be a part or marker ID.

Output:  (Double)  Angular acceleration component.
Description:  Returns component of part’s center of mass angular acceleration.

GetPressureAngle
Syntax:  GetCMAngularAcceleration (component, frame, curveCurveID, csysID, refID, value)
Input:  (Long) component  Component to measure. See DDMResultsComponent.
(Long) frame  Simulation frame of requested results. (1 to number of simulated frames)
(Long) curveCurveID  ID of curve-curve element.
(Long) csysID  Coordinate system in which vector components are expressed. By default, the coordinate system is the global coordinate system. This must be a part or marker ID.
(Long) refID  Motion reference frame relative to which time-derivatives are performed. This must be a part or marker ID.

Output:  (Double)  Angle component.
Description:  Returns component of the pressure angle for a DDMCurveCurve element.

GetElementForce
Syntax:  GetElementForce (component, frame, elemID, csysID, refID, fromFirst, value)
Input:  (Long) component  Component to measure. See DDMResultsComponent.
(Long) frame  Simulation frame of requested results. (1 to number of simulated frames)
(Long) elemID  ID of element (joints, forces)
(Long) csysID  Coordinate system in which vector components are expressed. By default, the coordinate system is the global coordinate system. This must be a part or marker ID.
(Long) refID  Motion reference frame relative to which time-derivatives are performed. This must be a part or marker ID.
(Bool) fromFirst  True if measured from I marker to J marker, False if reverse.

Output:  (Double)  Force component.
Description:  Returns component of the force on a element connecting two parts.
GetElementTorque

Syntax: GetElementForce (component, frame, elemID, csysID, refID, fromFirst, value)
Input:
(Long) component Component to measure. See DDMResultsComponent.
(Long) frame Simulation frame of requested results. (1 to number of simulated frames)
(Long) elemID ID of element (joints, forces)
(Long) csysID Coordinate system in which vector components are expressed. By default, the coordinate system is the global coordinate system. This must be a part or marker ID.
(Long) refID Motion reference frame relative to which time-derivatives are performed. This must be a part or marker ID.
(Boolean) fromFirst True if measured from I marker to J marker, False if reverse.
Output: (Double) Torque component.
Description: Returns component of the torque on a element connecting two parts.

IDDMElement

Properties

Name
Get Syntax: name = element.Name
Put Syntax: element.Name = name
Value: String
Description: Accesses the element’s name. This name must be unique among all elements. If a name is set that is not unique, a number is appended.

Type
Get Syntax: type = element.Type
Value: Long
Description: Gets the type of the element. For possible values, see DDMElementType_contants.

ID
Get Syntax: type = element.ID
Value: Long
Description: The ID is a unique number assigned to an element. Element IDs are also unique among Motion and Connector IDs. These IDs are used in the Simulation results functions.

Suppressed
Get Syntax: suppressed = element.Suppressed
Put Syntax: element. Suppressed = suppressed
Value: Boolean
Description: The Get function returns whether the element is suppressed or not. The user can also suppress or unsuppress the element through the Put function.

Valid
Get Syntax: valid = element.Valid
Value: Boolean
Description: Checks whether the element is valid or not.
IDDMPart

**Properties**

**MaterialID**
Get Syntax:  
Put Syntax:  
Value:  
Description:  

**Connectors**

**AttachedParts**

**DensitySource**

**Density**

**CustomMass**

**MotionType**

**Attached**
Value: Boolean
Description: True if part is attached to another DDMPart.

ParentPart
Syntax: Part.ParentPart
Value: IDDMPart object
Description: Returns the parent DDMPart if the part is attached.

SourceObject
Get Syntax: obj = Part.SourceObject
Value: IDispatch
Description: Retrieves the CAD system’s part object used to define DDMPart.

Methods

SetInertia
Syntax: Part.SetInertia(inertia)
Value: (Double)inertia SAFEARRAY of type double.
Description: Sets the mass properties of the part.

GetInertia
Syntax: Part.GetInertia(inertia)
Value: (Double)inertia SAFEARRAY of type double.
Description: Gets the mass properties of the part.

SetInitVelocities
Syntax: Part.SetInitVelocities(vel)
Value: (Double)vel SAFEARRAY of type double.
Description: Sets the initial linear and angular velocities for the part.

GetInitVelocities
Syntax: Part.SetInitVelocities(vel)
Value: (Double)vel SAFEARRAY of type double.
Description: Gets the initial linear and angular velocities for the part.

GetPosition
Syntax: Part.GetPosition(frame, pos)
Value: (Long)frame Frame at which position of the part in needed. (Double)pos SAFEARRAY of type double.
Description: Gets the position of the part at the given frame.

AttachToPart
Syntax: Part.AttachToPart( part)
Input: (Object) part Part that has to be attached to the parent
Description: Attaches the part passed in by the user to a parent part.

GetInertialStateForFea
Syntax: Part.GetInertialStateForFea (graAccn, transVel, transAccn, rotVel, rotAccn, frame)
Value: (Double) graAccn Gravitational acceleration at the given frame. SAFEARRAY of doubles.
(Double) transVel  Translational velocity at the given frame. SAFEARRAY of doubles.
(Double) transAccn  Translational acceleration at the given frame. SAFEARRAY of doubles.
(Double) rotVel  Rotational velocity at the given frame. SAFEARRAY of doubles.
(Double) rotAccn  Rotational acceleration at the given frame. SAFEARRAY of doubles.
(Long)frame  The frame at which the Fea data is needed. The frame should be greater than zero but less than or equal to the number of frames in the simulation.

Description:  Outputs the inertial forces at the given frame for all the free parts in the mechanism.

IDDMConnector

Description:  Elements connecting two parts are Connectors, including joints, forces, and others. Connectors must connect at least one moving Part to a Part that is included in the Mechanism.

Properties

PartI
   Get Syntax:  Part = Connector.PartI
   Put Syntax:  Connector.PartI = Part
   Value:  IDDMPart object
   Description:  Gets and puts first of the two Parts.

PartJ
   Get Syntax:  Part = Connector.PartJ
   Put Syntax:  Connector.PartJ = Part
   Value:  IDDMPart object
   Description:  Gets and puts second of the two Parts.

IDI
   Get Syntax:  id = Connector.IDI
   Value:  Long
   Description:  Gets the ID of the position of the Connector on the i Part. This can be used in Function expressions and for retrieving results.

IDJ
   Get Syntax:  id = Connector.IDJ
   Value:  Long
   Description:  Gets the ID of the position of the Connector on the j Part. This can be used in Function expressions and for retrieving results.

Methods

GetDataForFea
   Syntax:  Connector.GetDataForFea (position1, load1, torque1, name1, position2, load2, torque2, name2, result, frame)
   Value:  (Double)position1  Position of the first part at the given frame. SAFEARRAY of doubles.
IDLMMotion API Programming Reference

(Double) load1 Load on the first part at the given frame. SAFEARRAY of doubles.
(Double) torque1 Torque of the first part at the given frame. SAFEARRAY of doubles.
(String) name1 Name of the first part whose data is given above.
(Double) position2 Position of the second part at the given frame. SAFEARRAY of doubles.
(Double) load2 Load of the second part at the given frame. SAFEARRAY of doubles.
(Double) torque2 Torque of the second part at the given frame. SAFEARRAY of doubles.
(String) name2 Name of the second part whose data is given above.
(Long) result In order to get the Fea data the user has to preselect the bearing surfaces. The method returns result = 0 if the user has not selected the bearing surfaces and no Fea data will be given out in that case. If bearing surfaces are already selected the result = 1 and Fea data will be given to the user.
(Long) frame The frame at which the Fea data is needed. The ‘frame’ should be greater than zero but less than or equal to the number of frames in the simulation.

Description: Outputs the Fea data (load, torque and position) for the given frame and for the two parts connected to the connector that is under consideration.

IDLMMConstrainedJoint

Properties

Friction

Syntax: friction = Joint.Friction
Value: IDDMJointFriction object
Description: Provides access to friction parameters of Joint.

Methods

Explode

Syntax: Joint.Explode()
Description: If the ConstrainedJoint is defined by a combination of constraints, the joint is replaced by multiple ConstrainedJoints, one for each constraint.

IDDMJoint

Properties

Location

Syntax: Location = Joint.Location
Value: IDDMLocation object
Description: Provides access to joint’s location.

OrientationI

Syntax: OrientI = Joint.OrientationI
Value: IDDMOrientation object
**Description:** Provides access to joint’s orientation. If the Joint is a Universal Joint, this property is the orientation of the shaft fixed in PartI.

### OrientationJ

**Syntax:**

\[ \text{OrientJ} = \text{Joint.OrientationJ} \]

**Value:** IDDMOrientation object

**Description:** Provides access to the Universal Joint’s orientation of the shaft fixed in PartJ. This property is not used by other Joint types.

### JointType

**Get Syntax:**

\[ \text{type} = \text{Joint.JointType} \]

**Put Syntax:**

\[ \text{Joint.JointType} = \text{type} \]

**Value:** DDMJointType

**Description:** Sets and gets the type of joint. See DDMJointType for values.

### ScrewPitch

**Get Syntax:**

\[ \text{pitch} = \text{Joint.ScrewPitch} \]

**Put Syntax:**

\[ \text{Joint.ScrewPitch} = \text{pitch} \]

**Value:** Double

**Description:** Puts and gets the pitch for a screw joint.

### ScrewCylindrical

**Get Syntax:**

\[ \text{include} = \text{Joint.ScrewCylindrical} \]

**Put Syntax:**

\[ \text{Joint.ScrewCylindrical} = \text{include} \]

**Value:** Boolean

**Description:** Gets whether the screw joint is cylindrical. The user can also set the screw joint as cylindrical.

### Motions

**Syntax:**

\[ \text{Motions} = \text{Joint.Motions} \]

**Value:** IDDMMotions object

**Description:** Collection of six motions for each degree of freedom.

### Friction

**Syntax:**

\[ \text{friction} = \text{Joint.Friction} \]

**Value:** IDDMJointFriction object

**Description:** Provides access to friction parameters of Joint.

### IDDMJointFriction

#### Properties

**Supported**

**Get Syntax:**

\[ \text{supported} = \text{JointFriction.Supported} \]

**Value:** Boolean

**Description:** True if the joint supports friction or else returns False.

**Enabled**

**Get Syntax:**

\[ \text{enabled} = \text{JointFriction.Enabled} \]

**Put Syntax:**

\[ \text{JointFriction.Enabled} = \text{enabled} \]
Value: Boolean
Description: The get function returns True if friction is set for the joint or else False. The user can also set the friction by passing True.

**Coefficient**

Get Syntax: `coeff = JointFriction.Coefficient`
Put Syntax: `JointFriction.Coefficient = coeff`
Value: Double
Description: Gets and Sets the Dynamic friction coefficient.

**Source**

Get Syntax: `coeff = JointFriction.Coefficient`
Put Syntax: `JointFriction.Coefficient = coeff`
Value: Double
Description: Gets and Sets the Dynamic friction coefficient.

**Methods**

**GetDimensions**

Syntax: `JointFriction.GetDimensions(d0, d1, d2)`
Output: *(Double) d0* Radius of the joint
*(Double) d1* Length of the joint
*(Double) d2* Height of the joint
Description: This gives the dimensions of the joint. All the joints do not have all the three dimensions listed above. In such cases 0 is returned for that dimension.

**SetDimensions**

Syntax: `JointFriction.SetDimensions(d0, d1, d2)`
Output: *(Double) d0* Radius of the joint
*(Double) d1* Length of the joint
*(Double) d2* Height of the joint
Description: Sets the dimensions of the joint. All the joints do not have all the three dimensions listed above. In such cases pass in 0 for that dimension.

**IDDMPartMotion**

**Properties**

**Location**

Syntax: `Location = Joint.Location`
Value: IDDMLocation object
Description: Provides access to origin’s location.

**OrientationX**

Syntax: `OrientX = Joint.OrientationX`
Value: IDDMOrientation object
Description: Provides access to orientation of X Axis.

**OrientationZ**

Syntax: `OrientZ = Joint.OrientationZ`
Value: IDDMOrientation object
Description: Provides access to orientation of Z Axis.

Motions
Syntax: Motions = Joint.Motions
Value: IDDMMotions object
Description: Collection of six motions for each degree of freedom.

**IDDMMotion**

**Properties**

**MotionType**
Syntax: Type = Motion.MotionType
        Motion.MotionType = DDMMotion_velocity
Value: DDMMotionType
Description: If the Motion is fixed by a Joint’s type, this property cannot be set. If the Motion is not free, the movement is determined by a function of displacement, velocity, or acceleration.

**Function**
Syntax: Function = Motion.Function
Value: IDDMFunction object
Description: Accesses Function object. This property cannot be accessed if the MotionType is fixed or free.

**InitialDisplacement**
Get Syntax: disps = Motion.InitialDisplacement
Put Syntax: Motion.InitialDisplacement = disps
Value: Double
Description: The user can get and set the initial displacement of the body if the motion type for the joint is velocity or acceleration.

**InitialVelocity**
Get Syntax: vels = Motion.InitialVelocity
Put Syntax: Motion.InitialVelocity = vels
Value: Double
Description: The user can get and set the initial velocity of the body if the motion type for the joint is acceleration.

**IDDMBushing**

**Properties**

**Location**
Syntax: Location = Bushing.Location
Value: IDDMLocation object
Description: Provides access to Bushing’s location on PartI or PartJ.

**TranslationalStiffness**
Get Syntax: transStiff = Bushing.TranslationalStiffness
Put Syntax: Bushing. TranslationalStiffness = transStiff
Value: Double
Description: The user can get and put the translational stiffness value of the bushing.

TranslationalDamping
Get Syntax: transDamp = Bushing. TranslationalDamping
Put Syntax: Bushing. TranslationalDamping = transDamp
Value: Double
Description: The user can get and put the translational damping value of the bushing.

Force
Get Syntax: force = Bushing. Force
Put Syntax: Bushing. Force = force
Value: Double
Description: The user can get and put the preload (force) value of the bushing.

RotationalStiffness
Get Syntax: rotStiff = Bushing. RotationalStiffness
Put Syntax: Bushing. RotationalStiffness = rotStiff
Value: Double
Description: The user can get and put the rotational stiffness value of the bushing, if torsional properties are enabled.

RotationalDamping
Get Syntax: rotDamp = Bushing. RotationalDamping
Put Syntax: Bushing. RotationalDamping = rotDamp
Value: Double
Description: The user can get and put the rotational damping value of the bushing, if torsional properties are enabled.

Torque
Get Syntax: torque = Bushing. Torque
Put Syntax: Bushing. Torque = torque
Value: Double
Description: The user can get and put the preload (torque) value of the bushing, if torsional properties are enabled.

Torsion
Get Syntax: tor = Bushing. Torsion
Put Syntax: Bushing. Torsion = tor
Value: Boolean
Description: The user can enable or disable the torsional properties of the bushing.

IDDMSpring

Properties

LocationI
Syntax: Location = Spring.LocationI
Value: IDDMLocation object
**Description:** Provides access to Spring’s location on PartI or the location in both parts if the Spring is rotational.

### LocationJ

**Syntax:** Location = Spring.LocationJ
**Value:** IDDMLocation object
**Description:** Provides access to Spring’s location on PartJ. This is ignored if the Spring is rotational.

### Orientation

**Syntax:** Orient = Spring.Orientation
**Value:** IDDMOrientation object
**Description:** Provides access a rotational Spring’s orientation. This property is used only by rotational Springs.

### LengthIsDerived

**Get Syntax:** derived = Spring.LengthIsDerived
**Put Syntax:** Spring.LengthIsDerived = derived
**Value:** Boolean
**Description:** The get function returns a True if the length of the spring is design length else returns a False. The user can also set the spring length to be design length by passing in True in the put function. The user can pass in False to the put function if he wants to set the spring length to some other value. This property is only used by non-rotational Springs.

### Length

**Get Syntax:** len = Spring.Length
**Put Syntax:** Spring.Length = len
**Value:** Double
**Description:** The user can get and put the length of the spring. This property is only used by non-rotational Springs.

### Angle

**Get Syntax:** ang = Spring.Angle
**Put Syntax:** Spring.Angle = ang
**Value:** Double
**Description:** The user can get and put the angle of the spring. This property is only used by rotational Springs.

### Force

**Get Syntax:** force = Spring.Force
**Put Syntax:** Spring.Force = force
**Value:** Double
**Description:** The user can get and put the force in the non-rotational springs. In case of rotational springs the value will be set as torque.

### Rotational

**Get Syntax:** rot = Spring.Rotational
**Put Syntax:** Spring.Rotational = rot
**Value:** Boolean
**Description:** The get property returns a True if the spring is rotational else returns False. The user can also set the spring to be rotational by passing in True and as non-rotational by passing in False in the put property.
Stiffness

**Get Syntax:**  
\texttt{stiff = Spring.Stiffness}

**Put Syntax:**  
\texttt{Spring.Stiffness = stiff}

**Value:**  
Double

**Description:**  
Gets and puts the stiffness of the spring. This property is used by both rotational and non-rotational Springs.

### IDDMDamper

#### Properties

**LocationI**

**Syntax:**  
\texttt{Location = Damper.LocationI}

**Value:**  
IDDMLocation object

**Description:**  
Provides access to Damper’s location on PartI or the location in both parts if the Damper is rotational.

**LocationJ**

**Syntax:**  
\texttt{Location = Damper.LocationJ}

**Value:**  
IDDMLocation object

**Description:**  
Provides access to Damper’s location on PartJ. This is ignored if the Damper is rotational.

**Orientation**

**Syntax:**  
\texttt{Orient = Damper.Orientation}

**Value:**  
IDDMOrientation object

**Description:**  
Provides access a rotational Damper’s orientation. This property is used only by rotational Dampers.

**Damping**

**Get Syntax:**  
\texttt{damp = Damper.Damping}

**Put Syntax:**  
\texttt{Damper.Damping = damp}

**Value:**  
Double

**Description:**  
Gets and sets the damping values for the damper. This property is used by both rotational and non-rotational dampers.

**Rotational**

**Get Syntax:**  
\texttt{rot = Damper.Rotational}

**Put Syntax:**  
\texttt{Damper.Rotational = rot}

**Value:**  
Boolean

**Description:**  
The get property returns a True if the damper is rotational else returns False. The user can also set the damper to be rotational by passing in True and as non-rotational by passing in False in the put property.
IDDMImpact

Properties

Location I
Syntax: Location = Impact.LocationI
Value: IDDMLocation object
Description: Provides access to Impact Force’s location on Part I.

Location J
Syntax: Location = Impact.LocationJ
Value: IDDMLocation object
Description: Provides access to Impact Force’s location on Part J.

Length Is Derived
Get Syntax: derived = Impact.LengthIsDerived
Put Syntax: Impact.LengthIsDerived = derived
Value: Boolean
Description: The get function returns a True if the impact length is the design length else returns a False. The user can also set the impact length to be design length by passing in True in the put function. The user can pass in False to the put function if he wants to set the impact length to some other value.

Length
Get Syntax: len = Impact.Length
Put Syntax: Impact.Length = len
Value: Double
Description: The user can get and put the impact length.

Contact Properties
Syntax: props = Impact.ContactProperties
Value: IDDMContactProperties object
Description: Provides access to contact properties object which can be used to set the properties of the impact force.

IDDMAc tion Only Force

Properties

Location
Syntax: Location = Force.Location
Value: IDDMLocation object
Description: Provides access to force’s location on the Part I of the Force.

Orientation
Syntax: Orient = Force.Orientation
Value: IDDMOrientation object
Description: Provides access to force’s orientation. This orientation changes with the position of the Part J of the Force.
Function
Syntax: Function = Force.Function
Value: IDDMFunction object
Description: Accesses the Force’s Function object which defines the force applied to the I Part.

Rotational
Syntax: Force.Rotational
Value: Boolean
Description: True if the Force produces torque about the orientation vector. False if the Force is linear in the direction of the orientation vector.

IDDMActionReactionForce

Properties

LocationI
Syntax: Location = Force.LocationI
Value: IDDMLocation object
Description: Provides access to force’s location on the first part (PartI) of the Force.

LocationJ
Syntax: Location = Force.LocationJ
Value: IDDMLocation object
Description: Provides access to force’s location on the second part (PartJ) of the Force. This is not used if the Force is rotational.

Orientation
Syntax: Orient = Force.Orientation
Value: IDDMOrientation object
Description: Provides access to force’s orientation. This applies only if the Force is rotational.

Function
Syntax: Function = Force.Function
Value: IDDMFunction object
Description: Accesses the Force’s Function object which defines the force applied to the I Part and the reactive force applied to the J Part.

Rotational
Syntax: Force.Rotational
Value: Boolean get/put
Description: True if the Force produces torque about the orientation vector. False if the Force is linear in the direction of the orientation vector.

IDDMFunction

Properties

Type
Get Syntax: Function.Type
**Value:**
DDMFunctionType

**Description:**
Gives the type of function. See DDMFunctionType for the values.

### Methods

#### SetStepParams

**Syntax:**
Function.SetStepParams(params)

**Input:**
(Double) params[0] Initial time  
(Double) params[1] Initial angle  
(Double) params[2] Final time  
(Double) params[3] Final angle

**Description:**
The property sets the initial time and angle from where the simulation will start and also the final time and angle where the simulation should stop in the step motion function during the simulation.

#### GetStepParams

**Syntax:**
Function.SetStepParams(params)

**Output:**
(Double) params[0] Initial time  
(Double) params[1] Initial angle  
(Double) params[2] Final time  
(Double) params[3] Final angle

**Description:**
The property gets the initial time and angle from where the simulation will start and also the final time and angle where the simulation has stopped in the step motion function.

#### SetHarmonicParams

**Syntax:**
Function.SetHarmonicParams(params)

**Input:**
(Double) params[0] Time offset  
(Double) params[1] Frequency  
(Double) params[2] Amplitude  
(Double) params[3] Phase shift  
(Double) params[4] Average

**Description:**
The property sets the parameters for the harmonic motion function used during simulation.

#### GetHarmonicParams

**Syntax:**
Function.GetHarmonicParams(params)

**Output:**
(Double) params[0] Time offset  
(Double) params[1] Frequency  
(Double) params[2] Amplitude  
(Double) params[3] Phase shift  
(Double) params[4] Average

**Description:**
The method gets the parameters for the harmonic motion function used during simulation.

#### SetDataPoints

**Syntax:**
Function.SetDataPoints(type, numPoints, xs, ys)

**Input:**
(Long) type Type of spline (Cubic or )  
(Long) numPoints Number of points on the pline  
(Double) xs x coordinates. SAFEARRAY of doubles.  
(Double) ys y coordinates. SAFEARRAY of doubles.

**Description:**
The method sets the parameters for the given set of data points.
GetDataPoints
Syntax: Function.GetDataPoints(type, numPoints, xs, ys)
Output:
(Long) type Type of spline (Cubic or )
(Long) numPoints Number of points on the spline
(Double) xs x coordinates. SAFEARRAY of doubles.
(Double) ys y coordinates. SAFEARRAY of doubles.
Description The method gets the parameters for the given set of data points.

LoadDataPoints
Syntax: Function.LoadDataPoints(type, fileName)
Input: (String) fileName Name of the file containing the data points
Output: (Long) type Type of spline (Cubic or )
Description This method loads the data points given by the user by reading a file.

SetExpression
Syntax: Function.LoadDataPoints(expr)
Input: (String) expr Expression for the motion
Description The property sets the motion by using the expression given by the user.

GetExpression
Syntax: Function.LoadDataPoints(expr)
Output: (String) expr Expression for the motion
Description The property gets the expression given by the user for the motion during the simulation.

SetConstant
Syntax: Function.LoadDataPoints(const)
Input: (Double)const Constant angular displacement degrees
Description The property sets the angular displacement value for the constant motion during the simulation.

GetConstant
Syntax: Function.LoadDataPoints(const)
Output: (Double)const Constant angular displacement degrees
Description The property gets the angular displacement value for the constant motion during the simulation.

IDDMPntCurve
Properties
Point
Syntax: point = PointCurve.Point
Value: IDDMLocation object
Description: Accesses the location for the point-curve.
Curve
Syntax: curve = PointCurve.Curve
Value: IDDMCurve object
Description: Accesses the properties that can define the curve.

IDDMCurve

Properties

Intermittent
Get Syntax: intermittent = CurveCurve.Intermittent
Put Syntax: CurveCurve.Intermittent = intermittent
Value: Boolean
Description: Set to True to enable the intermittent contact using the ContactProperties for impact and friction parameters.

CurveI
Syntax: curve = CurveCurve.CurveI
Value: IDDMCurve object
Description: Accesses the properties that define CurveI.

CurveJ
Syntax: curve = CurveCurve.CurveJ
Value: IDDMCurve object
Description: Accesses the properties that define CurveJ.

ContactProperties
Syntax: contact = CurveCurve.ContactProperties
Value: IDDM ContactProperties object
Description: Accesses the contact properties for calculating impact force and friction. Used if Intermittent is True.

IDDMContactProperties

Properties

MaterialID1
Get Syntax: mat = ContactProperties.MaterialID1
Put Syntax: ContactProperties.MaterialID1 = mat
Value: Long
Description: Overrides the material associated with the owning connector’s PartI. The material combination is used to reference impact parameters in the DDM Material Database if Source is True. Set to –1 to not override the connector’s PartI material.
MaterialID2

**Get Syntax:**
mat = ContactProperties.MaterialID2

**Put Syntax:**
ContactProperties.MaterialID2 = mat

**Value:** Long

**Description:** Overrides the material associated with the owning connector’s PartJ. The material combination is used to reference impact parameters in the DDM Material Database if Source is True. Set to –1 to not override the connector’s PartJ material.

Source

**Get Syntax:**
sourse = ContactProperties.Source

**Put Syntax:**
ContactProperties.Source = source

**Value:** Long

**Description:** Set to True to use the materials for contact properties, disabling the impact and friction settings. Set to False to use the impact and friction settings, ignoring the materials.

UseCoefficientOfRestitution

**Get Syntax:**
use = ContactProperties.UseCoefficientOfRestitution

**Put Syntax:**
ContactProperties.UseCoefficientOfRestitution = True

**Value:** Boolean

**Description:** Set to True to use coefficient of restitution parameter in place of the impact parameters. False by default.

CoefficientOfRestitution

**Get Syntax:**
val = ContactProperties.UseCoefficientOfRestitution

**Put Syntax:**
ContactProperties.UseCoefficientOfRestitution = val

**Value:** Double

**Description:** Used if UseCoefficientOfRestitution is True.

Damping

**Get Syntax:**
val = ContactProperties.Damping

**Put Syntax:**
ContactProperties.Damping = val

**Value:** Double

**Description:** Sets and gets the damping value used to calculate impact force. Used if ContactProperties.UseCoefficientOfRestitution is False.

DynamicFrictionCoefficient

**Get Syntax:**
val = ContactProperties.DynamicFrictionCoefficient

**Put Syntax:**
ContactProperties.DynamicFrictionCoefficient = val

**Value:** Double

**Description:** Sets and gets the damping value used to calculate impact force. Used if Source is False.

DynamicFrictionVelocity

**Get Syntax:**
val = ContactProperties.DynamicFrictionVelocity

**Put Syntax:**
ContactProperties.DynamicFrictionVelocity = val

**Value:** Double

**Description:** Sets and gets the dynamic friction velocity value. Used if Source is False.

StaticFrictionCoefficient

**Get Syntax:**
val = ContactProperties.StaticFrictionCoefficient

**Put Syntax:**
ContactProperties.StaticFrictionCoefficient = val
Value: Double
Description: Sets and gets the static friction value. Used if Source is False.

**StaticFrictionVelocity**
Get Syntax: `val = ContactProperties.StaticFrictionVelocity`
Put Syntax: `ContactProperties.StaticFrictionVelocity = val`
Value: Double
Description: Sets and gets the static friction velocity value. Used if Source is False.

**Exponent**
Get Syntax: `val = ContactProperties.Exponent`
Put Syntax: `ContactProperties.Exponent = val`
Value: Double
Description: Sets and gets the exponent value used to calculate the impact force. Settable if Source is False. Used if UseCoefficientOfRestitution is False.

**Penetration**
Get Syntax: `val = ContactProperties.Penetration`
Put Syntax: `ContactProperties.Penetration = val`
Value: Double
Description: Sets and gets the penetration value used to calculate the impact force. Settable if Source is False. Used if UseCoefficientOfRestitution is False. Value is in mechanism’s length unit.

**Stiffness**
Get Syntax: `val = ContactProperties.Stiffness`
Put Syntax: `ContactProperties.Stiffness = val`
Value: Double
Description: Sets and gets the stiffness value used to calculate the impact force. Settable if Source is False. Used if UseCoefficientOfRestitution is False.

**IDDMCoupler**

**Properties**

**Joint1**
Get Syntax: `joint = Coupler.Joint1`
Put Syntax: `Coupler.Joint1 = joint`
Value: IDDMJoint or IDDMConstrainedJoint object
Description: Accesses the properties of DDMJoint or DDMConstrainedJoint. The set function sets the first joint selected by the user.

**Joint2**
Get Syntax: `joint = Coupler.Joint2`
Put Syntax: `Coupler.Joint2 = joint`
Value: IDDMJoint or IDDMConstrainedJoint object
Description: Accesses the properties of DDMJoint or DDMConstrainedJoint. The set function sets the second joint selected by the user.
Rotational1

Get Syntax:    rot = Coupler.Rotational1
Put Syntax:    Coupler. Rotational1 = rot
Value:         Boolean
Description:   The user can set the rotational property by passing in True and translational property by passing in False. The user can also know which property is set from the get property.

Rotational2

Get Syntax:    rot = Coupler.Rotational2
Put Syntax:    Coupler. Rotational2 = rot
Value:         Boolean
Description:   The user can set the rotational property by passing in True and translational property by passing in False. The user can also know which property is set from the get property.

Scale1

Get Syntax:    scale = Coupler.Scale1
Put Syntax:    Coupler. Scale1 = scale
Value:         Double
Description:   Sets and gets the scale of the first joint.

Scale2

Get Syntax:    scale = Coupler.Scale2
Put Syntax:    Coupler. Scale2 = scale
Value:         Double
Description:   Sets and gets the scale of the second joint.

IDDMTracePath

Properties

Point

Syntax:    loc = TracePath.Point
Value:     IDDMLocation
Description: Accesses the property to set the location of the point on the relative part whose motion has to be traced.

PathPointNumber

Get Syntax:    num = TracePath.PathPointNumber
Value:         Long
Description:   Gets the number of points in the trace path.

Method

GetPathPoints

Syntax:    TracePath.GetPathPoints(points)
Value:     (Double) points SAFEARRAY of doubles.
Description: Gets the points on the trace path.

ExportCSVFile

Syntax:    TracePath .ExportCSVFile(fileName)
Value: (String).fileName  Name of the exported file
Description: Exports the data of the Trace path namely the number of points and the actual points on the trace path.

**IDDMLocation**

**Properties**

**SourceObject**

*Get Syntax:* source = Location.SourceObject
*Value:* IDispatch
*Description:* Gets the location directly from the CAD system.

**Methods**

**SetPoint**

*Set Syntax:* Location.SetPoint(x, y, z)
*Value:* (Double)x  X coordinate of the point  
(Double)y  Y coordinate of the point  
(Double)z  Z coordinate of the point
*Description:* Sets the point/location given by the user.

**GetPoint**

*Set Syntax:* Location.GetPoint(x, y, z)
*Value:* (Double)x  X coordinate of the point  
(Double)y  Y coordinate of the point  
(Double)z  Z coordinate of the point
*Description:* Gets the point/location given to the user.

**IDDMOrientation**

**Properties**

**Flipped**

*Get Syntax:* flipped = Orientation.Flipped
*Put Syntax:* Orientation.Flipped = flipped
*Value:* Boolean
*Description:* By passing True in the Put property the direction of the orientation vector can be changed while passing False it remains at the already set direction.

**SourceObject**

*Get Syntax:* source = Orientation.SourceObject
*Value:* IDispatch
*Description:* Gets the orientation directly from the CAD system.
**Methods**

**SetVector**

*Syntax:* `Orientation.SetVector(x, y, z)`

*Value:*  
(Double)x  X coordinate of the point  
(Double)y  Y coordinate of the point  
(Double)z  Z coordinate of the point

*Description:* Sets the vector/direction given by the user.

**GetVector**

*Syntax:* `Orientation.GetVector(x, y, z)`

*Value:*  
(Double)x  X coordinate of the point  
(Double)y  Y coordinate of the point  
(Double)z  Z coordinate of the point

*Description:* Gets the vector/direction given to the user.

**IDDMCurve**

**Properties**

**PointNumber**

*Get Syntax:* `num = Curve.PointNumber`

*Value:* Long

*Description:* Gets the number of points if curve is set by number of points.

**FitType**

*Get Syntax:* `type = Curve.FitType`

*Set Syntax:* `Curve.FitType = type`

*Value:* Long

*Description:* The user can set and get the type of fit required. The type of fits are PointsPerCurve fit and the Tolerance fit.

**FitPointNumber**

*Get Syntax:* `num = Curve.FitPointNumber`

*Set Syntax:* `Curve. FitPointNumber = num`

*Value:* Long

*Description:* Gets and puts the number of points for that curve.

**FitTolerance**

*Get Syntax:* `tol = Curve.FitTolerance`

*Set Syntax:* `Curve. FitTolerance = tol`

*Value:* Long

*Description:* Gets and puts the tolerance value to fit the curve.
MaterialSideFlipped

**Get Syntax:** flipped = Curve.MaterialSideFlipped

**Set Syntax:** Curve.MaterialSideFlipped = flipped

**Value:** Boolean

**Description:** The user can change the material side of the curve by passing in True in the set property or False is not to change the material side. The user can also know whether the material side has been flipped from the get property.

SourceObjects

**Get Syntax:** flipped = Curve. SourceObjects

**Set Syntax:** Curve. SourceObjects

**Value:** Object array.

**Description:** Source objects are arrays of edges that compose a curve. If the curve is defined by the outer boundary of a face, then the array has a single face object.

### Methods

**ClearSourceObjects**

**Syntax:** Curve.ClearSourceObjects()

**Value:** Void

**Description:** This clears all the curves.

**SetPoints**

**Syntax:** Curve.SetPoints(points)

**Value:** (Double)points SAFEARRAY of doubles

**Description:** Sets the points in the curve.

**GetPoints**

**Syntax:** Curve.GetPoints(points)

**Value:** (Double)points SAFEARRAY of doubles

**Description:** Gets the points in the curve.

**AddSourceObject**

**Syntax:** Curve.AddSourceObject(Object)

**Value:** Void

**Description:** Adds an edge to the curve’s edge list. When the list is complete, the edges must connect end-to-end to form and open or closed curve. If the object is a face, the outer boundary of the face is used to define the curve.

### IDDMPlot

### Properties

**AgainstTime**

**Get Syntax:** if (Plot.AgainstTime) then

**Value:** Boolean

**Description:** If the x axis is Time, this returns true. Otherwise, plot x axis is a result.
Methods

AddYAxis
Syntax: Plot.AddYAxis(elemId,result,component)
Value: (Long)elemId  ID of the element whose plot is being plotted
(DDMPlotResult)result  Type of result desired.
(DDMResultsComponent)component  X, Y, Z or Magnitude of the result
Description: Adds another curve of the desired result to the existing plot with respect to the same X axis.

ExportCSVFile
Syntax: Plot.ExportCSVFile(fileName)
Value: (String)fileName  Name of the file in which the exported data is stored.
Description: Exports the plot data to the file. Exports the X and the corresponding Y values of the result to the file.

SetXAxis
Syntax: Plot.SetXAxis(type,result,component,elemID)
Value: (DDMPlotXAxis)type  Plotting the graph with respect to either time.
(DDMPlotResult)result  Type of result desired.
(DDMResultsComponent)component  X, Y, Z or Magnitude of the result.
(Long)elemID  ID of the element whose plot is being plotted.
Description: Sets the X Axis to be time or frame or can be any of the results type. Please see the DDMPlotResult for the results type.

GetXAxis
Syntax: Plot.GetXAxis(type,result,component,elemID)
Value: (Long)type  Plotting the graph with respect to either time.
(DDMPlotResult)result  Type of result desired. See DDMPlotResult.
(DDMResultsComponent)component  X, Y, Z or Magnitude of the result.
(Long)elemID  ID of the element whose plot is being plotted.
Description: Gets the X Axis that can be time or frame or can be any of the results type. Please see the DDMPlotResult for the results type.

IDDMContact3D

Properties

PartSources
Get Syntax: sources = Contact3D.PartSources(set)
Value: DDMSourceObjects  List of CAD references.
Inputs: (Long)set  1 for first set of parts, 2 for second set.
Description: Gets the list of CAD object references to parts that are used to form body pairs for contacts.
TwoPartSets

Get Syntax: if (contact3D.TwoPartSets) then …
Value: Bool
Description: True if two sets of parts are used to create body pairs.

ContactProperties

Get Syntax: body = Contact3D.ContactProperties
Value: IDDMContactProperties Object
Description: Gives access to the contact properties required for the 3D contact.

Methods

AddPartSource
Syntax: AddPartSource(set, object)
Inputs: (Long)set 1 for first set of parts, 2 for second set.
(Object)object Component part or subassembly containing one or more solids.
Description: Adds a reference to a component in the CAD system to the set. If the component is not
under a moving or grounded DDM part, the addition is not done.

RemovePartSource
Syntax: RemovePartSource(object)
Inputs: (Object)object Component part or subassembly to be removed.
Description: The component must be in either of the contact part sets or it must contain components in
either set. The matching components are removed. If the the Contact no longer defines
body pairs, the element will be deleted on the next update of the mechanism (e.g. before a
solve).

RemovePartSet
Syntax: RemovePartSet(set)
Inputs: (Long)set 1 for first set of parts, 2 for second set.
Description: If two sets are used, one is removed and new body pairs are created using a single set.

IDDMotions

Properties

TranslateX
Get Syntax: transX = Motions.TranslateX
Value: IDDMotion object
Description: Translation along X direction. This property creates an instance on IDDMotion.

TranslateY
Get Syntax: transY = Motions.TranslateY
Value: IDDMotion object
Description: Translation along Y direction. This property creates an instance on IDDMotion.

TranslateZ
Get Syntax: transZ = Motions.TranslateZ
Value: IDDMMotion object
Description: Translation along Z direction. This property creates an instance on DDMMotion.

RotateX
Get Syntax: rotX = Motions.RotateX
Value: IDDMMotion object
Description: Rotation about X direction. This property creates an instance on DDMMotion.

RotateY
Get Syntax: rotY = Motions.RotateY
Value: IDDMMotion object
Description: Rotation about Y direction. This property creates an instance on DDMMotion.

RotateZ
Get Syntax: rotZ = Motions.RotateZ
Value: IDDMMotion object
Description: Rotation about Z direction. This property creates an instance on DDMMotion.

IDDMElements

Properties

Count
Get Syntax: count = Elements.Count
Value: Long
Description: Gives the total number of elements in the assembly.

Methods

Item
Syntax: Elements.Item(index,retval)
Value: (Long)index   The number of the entity that is under consideration
        (Object)retval   IDDMEElement Object
Description: Gives access to the element properties. The user can select a particular element from the
list of all the elements in the assembly by using this method.

IDDMConnectors

Properties

Count
Get Syntax: count = Connectors.Count
Value: Long
Description: Gives the total number of connectors in the assembly.
Methods

Item
Syntax: Connectors.Item(index,retval)
Value: (Long)index The number of the connector that is under consideration
(Object)retval IDDMConnector Object
Description: Gives access to the connector properties. The user can select a particular connector from
the list of all the connectors in the assembly by using this method.

IDDMParts

Properties

Count
Get Syntax: count = Parts.Count
Value: Long
Description: Gives the total number of parts in the assembly.

Methods

Item
Syntax: Parts.Item(index,retval)
Value: (Long)index The number of the part that is under consideration
(Object)retval IDDMPart Object
Description: Gives access to the part properties. The user can select a particular part from the list of all
the elements in the parts by using this method.

IDDMPlots

Properties

Count
Get Syntax: count = Plots.Count
Value: Long
Description: Gives the total number of plots in the assembly.

Methods

Item
Syntax: Plots.Item(index,retval)
Value: (Long)index The number of the plot that is under consideration
(Object)retval IDDMPlot Object
Description: Gives access to the plot properties. The user can select a particular plot from the list of all
the plots in the assembly by using this method.
Constants

**DDMProductId**
- DDMProduct_SimplyMotion = 1,
- DDMProduct_Motion = 2,
- DDMProduct_MotionProfessional = 3

**DDMForceUnits**
- DDMForce_kilogram = 1,
- DDMForce_newton = 2,
- DDMForce_kilonewton = 3,
- DDMForce_pound = 4,
- DDMForce_kilopound = 5,
- DDMForce_dyne = 6,
- DDMForce_ounce = 7,
- DDMForce_millinewton = 8

**DDMTimeUnits**
- DDMTime_hour = 1,
- DDMTime_minute = 2,
- DDMTime_second = 3,
- DDMTime_millisecond = 4

**DDMIntegratorType**
- DDMIntegrator_GSTIFF = 1,
- DDMIntegrator_WSTIFF = 2

**DDMResultsComponent**
- DDMComponent_x,
- DDMComponent_y,
- DDMComponent_z,
- DDMComponent_magnitude

**DDMPartParameterSource**
- DDMPart_useUserSettings = 1,
- DDMPart_useMaterial = 2,
- DDMPart_usePart = 3

**DDMPartMotionType**
- DDMPart_excluded = 1,
- DDMPart_moving = 2,
- DDMPart_grounded = 3

**DDMElemntType**
- DDMElement_part = 1,
- DDMElement_joint = 2,
- DDMElement_constrainedJoint = 3,
- DDMElement_pointCurve = 4,
- DDMElement_curveCurve = 5,
- DDMElement_forceActionOnly = 6,
- DDMElement_forceActionReaction = 7,
DDMElement_spring = 8,
DDMElement_damper = 9,
DDMElement_impact = 10,
DDMElement_coupler = 11,
DDMElement_tracePath = 12

DDMJointType
DDMJoint_revolute = 1,
DDMJoint_cylindrical = 2,
DDMJoint_translational = 3,
DDMJoint_spherical = 4,
DDMJoint_planar = 5,
DDMJoint_universal = 6,
DDMJoint_screw = 7,
DDMJoint_fixed = 8,
DDMJoint_inline = 9,
DDMJoint_inplane = 10,
DDMJoint_parallel = 11,
DDMJoint_perpendicular = 12,
DDMJoint_orientation = 13

DDMMotionType
DDMMotion_fixed = 1,
DDMMotion_free = 2,
DDMMotion_displacement = 3,
DDMMotion_velocity = 4,
DDMMotion_acceleration = 5

DDMFunctionType
DDMFunction_constant = 1,
DDMFunction_step = 2,
DDMFunction_harmonic = 3,
DDMFunction_dataPoints = 4,
DDMFunction_expression = 5

DDMContactPropertySource
DDMContact_useUserSettings = 1,
DDMContact_usePartMaterials = 2,
DDMContact_useContactMaterials = 3

DDMPlotResult
DDMPlot_TranslationalDisplacement = 1,
DDMPlot_TranslationalVelocity = 2,
DDMPlot_TranslationalAcceleration = 3,
DDMPlot_AngularVelocity = 4,
DDMPlot_AngularAcceleration = 5,
DDMPlot_TotalForceOnPoint = 6,
DDMPlot_TotalForceAtLocation = 7,
DDMPlot_TotalTorqueOnPoint = 8,
DDMPlot_TotalTorqueAtLocation = 9,
DDMPlot_PowerConsumption = 10,
DDMPlot_PotentialEnergyDelta = 11,
DDMPlot_StrainKineticEnergy = 12,
DDMPlot_TranslationalKineticEnergy = 13,
DDMPlot_AngularKineticEnergy = 14,
DDMPlot_KineticEnergy = 15,
DDMPlot_TranslationalMomentum = 16,
DDMPlot_AngularMomentumAboutCM = 17,
DDMPlot_CMPosition = 18,
DDMPlot_CMVelocity = 19,
DDMPlot_CMacceleration = 20,
DDMPlot_CMAngularVelocity = 21,
DDMPlot_CMAngularAcceleration = 22,
DDMPlot_PressureAngle = 23,
DDMPlot_ElementForce = 24,
DDMPlot_ElementTorque = 25

**DDMPlotXAxis**

DDMPlot_vsTime=1,
DDMPlot_vsFrame=2,
DDMPlot_vsResult=3