Neuron Data Elements Environment Intelligent Rules Element

Version 4.0

C++ Reference Summary

Part Number: OI-6010-0796-00

Contents

NDNxAtom:: Class

Function	Returns	Arguments	Description
GetAtomInfo	NxAtomPtr	(NxAtomPtr *atom1*, NxAtomGAInfoEnum *code*, NxAtomPtr *atom2*, Int32 *optInt*);	Method to get various atom datatype information about `atom1'.
GetClientData	Long	(void);	Gets the user-defined client data value (stored as a long) associated with atom'.
GetDoubleInfo	Double	(NxAtomPtr *atom1*, NxAtomGAInfoEnum *code*, NxAtomPtr *atom2*, Int32 *optInt*);	Method to get various double datatype information about `atom1'.
GetIntInfo	Int32	(NxAtomPtr *atom1*, NxAtomGAInfoEnum *code*, NxAtomPtr *atom2*, Int32 *optInt*);	Method to get various integer datatype information about `atom1'.
GetLongInfo	Long	(NxAtomPtr *atom1*, NxAtomGAInfoEnum *code*, NxAtomPtr *atom2*, Int32 *optInt*);	Method to get various long datatype information about `atom1'.
GetName	Long	(void);	Returns the string name of `atom'.
GetStrInfo	Str	(NxAtomPtr *atom1*, NxAtomGAInfoEnum *code*, NxAtomPtr *atom2*, Int32 *optInt*, Int32 *len*);	Method to get various string datatype information about `atom1'.
GetType	NxAtomTypeEnum	(void);	Returns the atom type of `atom' (Class, Object, Rule, etc)
SetClientData	void	(Long *data*);	Sets a user-defined client data value (stored as a long) to be associated with `atom'.
SetInfo	Int32	NxAtomPtr *atom1*, NxAtomSAInfoEnum *code*, NxAtomPtr *atom2*, Int32 *optInt*);	Sets various types of information about this atom.

NDNxClass:: Tables

Function	Returns	Arguments	Description
`CreateObject`	NxObjectPtr	(CStr *name*);	Creates an object named `name' with the specified parent class.
`DeleteObject`	Int32	(NxObjectPtr *object*);	Removes the object `object' from the specified parent class. Returns an integer status.
`Find7`	NxClassPtr	(CStr *name*);	Returns the IRE Class specified by `name'. Returns NULL if not found.
`FindSlot`	NxSlotPtr	(CStr *name*);	Returns the IRE Slot specified with a property name of `name' and a parent class `parent'.
`FindSlotByProp`	NxSlotPt	(NxPropPtr *prop*);	Returns the IRE Slot specified with a property of `prop' and a parent class `parent'.
`GetChildClassCount`	Int32	(void);	Returns the number of child objects attached directly to this class.
`GetChildObjectCount`	Int32	(void);	Returns the number of child classes attached directly to this class.
`GetClientData`	Long	(void);	Gets a user-defined client data value associated with this class.
`GetCount`	Int32	(void);	Returns the number of IRE Classes currently loaded.
`GetFirst`	NxClassPtr	(void);	Returns the first IRE Class of the currently loaded set.
`GetIndexedChildClass`	void	(void);	Returns the Nth child class attached directly to this class.
`GetIndexedChildObject`	NxObjectPtr	(Int32 *index*);	Returns the Nth child object attached directly to this class.
`GetIndexedMethod`	NxMethodPtr	(Int32 *index*);	Returns the Nth method attached directly to this class.
`GetIndexedParentClass`	NxSlotPtr	(Int32 index);	Returns the number of parent classes attached directly to this class.

Function	Returns	Arguments	Description
`GetIndexedSlot`	NxSlotPt	(Int32 *index*);	Returns the Nth slot attached directly to this class.
`GetKB`	NxKBPtr	(void);	Returns the Knowledge Base associated with this class.
`GetLast`	void	(void);	Returns the last IRE Class of the currently loaded set.
`GetLinkType`	NxClassLinkEnum	(NxAtomPtr *child*);	Returns link information: none, permanent, temporary, etc.
`GetMethodCount`	Int32	(void);	Returns the number of user-defined methods attached directly to this class.
`GetName`	Str	(void);	Returns the string name of the specified class.
`GetNext`	NxClassPtr	void);	Returns the next IRE Class of the currently loaded set.
`GetParentClassCount`	Int32	(void);	Returns the number of parent classes attached directly to this class.
`GetPrevious`	NxClassPtr	(void);	Returns the previous IRE Class of the currently loaded set.
`GetPrivateMethod`	NxMethodPtr	(CStr *name*);	Returns the method pointer / object of the private method named `name' attached directly to the specified class.
`GetPublicMethod`	NxMethodPt	(void);	Returns the method pointer / object of the public method named `name' attached directly to the specified class.
`GetSlotCount`	Int32	(void);	Returns the number of slots attached directly to this class.
`MakeLinkPermanent`	Int32	(NxObjectPtr *object*);	Changes an object's temporary link(s) to permanent link(s).
`SetClientData`	void	(Long *data*);	Sets a user-defined client data value to be associated with this class.
`SetKB`	void	(NxKBCPtr *kb*);	Changes the Knowledge Base that contains the definition of this class.

NDNxCtx:: Class

Function	Returns	Arguments	Description
`~NDNxCtx`	void	(void);	Default destructor
`AllocateClientId`	NxCtxClientId	(void)	Allocates and returns a client id so that multiple clients can associate multiple pieces of data with the same context block.
`GetClientData`	ClientPtr	(NxCtxClientId *id*);	Allows a customer to retrieve the information previously set with NxCtx::SetClientData().
`GetCur`	NxCtxClientId	(void);	Returns the current context.
`IsClientIdValid`	BoolEnum	(void);	A debugging tool to ensure that the specified client id is valid.
`IsValid`	BoolEnum	(void);	A debugging tool to ensure that the specified context is valid.
`NDNxCtx`	void	(void);	Default constructor
`SetClientData`	void	(NxCtxClientId *id*, ClientPtr *ptr*);	Allows a customer to store information for their purposes along with the context block
`SetCur`	NxCtxPtr	(void);	Sets the current context.
`SetNfyProc`	void	((NxCtxNfyProc *proc*);	Associates a notification procedure with the context mechanism.
`UnsetNfyProc`	void	(NxCtxNfyProc *proc*);	This will remove the specified procedure from the list of procedures that will be notified when a context event occurs.

NDNxEdt:: Class

Function	Returns	Arguments	Description
`~NDNxEdt`	void	(void);	Destructor
`Create`	void	(NxAtomPtr *oldAtom*, NxAtomPtrPtr *newAtom*);	Creates a new Atom in the Knowledge Base using the specified information. newAtom' is the newly created Atom.
`Delete`	Int	(NxAtomPtr *atom*);	Attempts to delete the specified Atom from the Knowledge Base.
`Fill`	Int	(NxAtomPtr *atom*);	Fills the edit structure with the current definition of an existing Atom.
`FindIndex`	Int	(Int *code*, CStrPtr *value*, Int *occurrence*);	Retrieves the index number in `occurrence' of the specified Atom, with a field code of `code' and a string value of `value'.
`GetNthStr`	Int	(Int *code*, CStrPtr *value*, Int *occurrence*);	Retrieves the string into `value' for the specified Atom which matches the `occurrence'-th instance for the field code `code'.
`GetStr`	Int	(Int *code*, CStrPtr *value*);	Retrieves into `value' the string corresponding to field `code' for the atom.
`Modify`	Int	(NxAtomPtr *oldAtom*, NxAtomPtrPtr *newAtom*);	Retrieves the string into `value' for the specified Atom which matches the `occurrence'-th instance for the field code `code'.
`NDNxEdt`	void	(void);	Constructor
`RemoveNthSt`	Int	(Int *code*);	Removes the `occurrence'-th instance of the field specified by `code' from the edit record of this Atom.
`Reset`	void	(void);	Resets the fields of the edit record for reuse.

Function	Returns	Arguments	Description
`SetAtomType`	Int	(Int *type*);	Sets the AtomType field in the edit record to the type specified by `type'.
`SetNthStr`	Int	(Int *code*, CStr *value*, Int *occurrence*);	Sets the `occurrence'-th instance of the field specified by `code' to the string passed in `value'.
`SetStr`	Int	(Int *code*, CStr *value*);	Sets the value of the field specified by `code' to the string specified in `value'.

NDNxEngine:: Class

Function	Returns	Argument	Description
`Compile`	Int32	(void);:	Compiles a user-supplied string with the TKB parser/compiler into the current KB.
`Continue`	NxEngineCtrlRetEnum	(void);:	Resume a (temporarily) stopped session.
`Control`	NxEngineCtrlRetEnum	(NxEngineCtrlEnum *code*);	Advanced access control of the rule engine.
`Exit`	NxEngineCtrlRetEnum	(void);:	Terminate the rule engine library.
`GetCurrentStrategy`	Int32	(NxEngineStrategyEnum *code*);	Retrieves the current setting for the strategy option specified.
`GetDefaultResetStrategy`	NxEngineVolStratEnum	(NxEngineStrategyEnum *code*);	Returns the default strategy to be used when resetting a slot/hypo to UNKNOWN
`GetDefaultStrategy`	Int32	(NxEngineStrategyEnum *code*);	Retrieves the current setting for the strategy option specified.
`GetDefaultSuggestStrategy`	NxEngineSugPrioEnum	(void);:	Returns the default strategy to be used when suggesting a hypo. If unset, the default is NXENGINE_SUGPRIO_SUG
`GetDefaultVolunteerStrategy`	void	(NxEngineVolStratEnum *strat)*;	Returns the default strategy to be used when volunteering data.
`GetError`	NxEngineErrEnum	(void);:	In the case of an error accessing any API, this will retrieve a more detailed error code. (unless an exception is thrown, instead).

Function	Returns	Argument	Description
`GetExecuteHandler`	NxpIProc	(CStr *name*);	Returns any previously installed 3GL execute handler specified by `name'.
`GetExecuteHandler`	NxpIProc	(CStr *name*, LongPtr *arg*);	Returns any previously installed 3GL execute handler specified by `name'.
`GetHandler`	NxpIProc	(NxEngineProcEnum *code*);	Returns any previously installed 3GL handler specified by `code'.
`GetHandler`	NxpIProc	(NxEngineProcEnum *code*, LongPtr *arg*);	Returns any previously installed 3GL handler specified by `code'.
`GetState`	NxEngineStateEnum	(void);:	Return the current "state" of the rule engine.
`Init`	NxEngineCtrlRetEnum	(void);:	Initialize the rule engine library.
`Journal`	Int32	(NxEngineJrnlEnum *mode*, CStr f*ilename*);	Save the working state of the rule engine to a specified file for later reuse.
`Restart`	NxEngineCtrlRetEnum	(void);:	Perform a "restart" operation to reset the values in the rule engine.
`SetCurrentStrategy`	void	(NxEngineStrategyEnum *code*, Int32 *value*);	Sets the strategy option `code' to the specified value.
`SetDefaultResetStrategy`	void	(NxEngineVolStratEnum *strat*);	Sets the default strategy to be used when resetting a slot/hypo to UNKNOWN.
`SetDefaultStrategy`	void	(NxEngineStrategyEnum *code*, Int32 *value*);	Sets the strategy option `code' to the specified value.
`SetDefaultSuggestStrategy`	void	(NxEngineSugPrioEnum *strat*);	Sets the default strategy to be used when suggesting a hypo.

Function	Returns	Argument	Description
`SetDefaultVolunteerStrategy`	void	(NxEngineVolStratEnum *strat*);	Sets the default strategy to be used when volunteering data.
`SetExecuteHandler`	Int32	(CStr *name*, NxpIProc *proc*);	Allows a user to establish a 3GL execute handler specified by `name', that should call `proc' when required.
`SetExecuteHandler`	Int32	(CStr *name*, NxpIProc *proc*, Long *arg*);	Allows a user to establish a 3GL execute handler specified by `name', that should call `proc' when required. `arg' is user-supplied custom information eg: a value or pointer) that will not be interpreted by the rule engine.
`SetHandler`	Int32	(NxEngineProcEnum *code*, NxpIProc *proc*);	Allows a user to establish a 3GL handler specified by `code', that should call `proc' when required.
`SetHandler`	Int32	(NxEngineProcEnum *code*, NxpIProc *proc*,Long *arg*);	Allows a user to establish a 3GL handler specified by `code', that should call `proc' when required.
`Start`	NxEngineCtrlRetEnum	(void);:	Start the rule engine.
`Stop`	NxEngineCtrlRetEnum	(void);:	Stop the rule engine.

NDNxKB:: Class

Function	Returns	Arguments	Description
`ClearAll`	Int32	(CStr *name*)	Clears all the Knowledge Bases from memory. Returns an integer status.
`Create`	NxKBPtr	(void);	Creates a knowledge base specified by `name'.
`Find`	NxKBPtr	(CStr *name*)	Given `name', find the corresponding KB object.
`GetComments`	CStr	(void);	For a given Knowledge Base `kb', returns any user comments associated with it.
`GetCount`	Int32	(void);	Returns the number of Knowledge Bases currently loaded.
`GetCurrent`	NxKBPtr	(void);	Retrieves the currently active / set Knowledge Base, or NULL if an error.
`GetFirst`	NxKBPtr	(void);	Returns the first Knowledge Base of the currently loaded set.
`GetLast`	NxKBPtr	(void);	Returns the last Knowledge Base of the currently loaded set.
`GetName`	Str	(void);	For a given Knowledge Base `kb', returns the string name associated with it.
`GetNext`	NxKBPtr	(void);	Returns the next Knowledge Base of the currently loaded set, based from kb'.
`GetPrevious`	NxKBPtr	(void);	Returns the previous Knowledge Base of the currently loaded set, based from kb', This requires a valid KB (eg: from first / next).

Function	Returns	Arguments	Description
`Load`	NxKBPtr	(void);	Loads the Knowledge Base from the file specified by `name'. Returns the KB object.
`MakeLinksPermanent`	Int32	(void);	This changes all temporary links of objects / classes of the `kb' to permanent.
`Merge`	Int32	(NxKBPtr *kb1*, NxKBPtr *kb2*);	Merges all components from knowledge base `kb2' into knowledge base `kb1'.
`Save`	Int32	(void);	Saves the specified Knowledge Base to a file, with `flags' to control saving of comments, compiled, text forms, etc. R
`SetCurrent`	void	(NxKBCPtr *kb*);	Sets the currently active Knowledge Base to `kb'.

NDNxMethod:: Class

Function	Returns	Arguments	Description
GetComments	Str	(void);	Returns the user comments string associated with the method.
GetCount	Int32	(void);	Returns the number of IRE Methods currently loaded.
GetCurrent	NxMethodPtr	(void);	Retrieves the current method, which may be NULL if there is none.
GetElseActionCount	Int32	(void);	Returns the number of Else-Actions found in this method (from the Else or EHS part).
GetFirst	NxMethodPtr	(void);	Returns the first IRE Method of the currently loaded set.
GetIfConditionCount	Int32	(void);	Returns the number of Conditions found in this method (from the If or LHS part).
GetIndexedElseAction	Str	(Int32 *index)*	Returns the string representation of the Nth EHS of this method.
GetIndexedIfCondition	Str	(void);	Returns the string representation of the Nth LHS of this method.
GetIndexedThenAction	void	(void);	Returns the string representation of the Nth RHS of this method.
GetKB	NxKBPtr	(void);	Returns the Knowledge Base associated with this method.
GetLast	NxMethodPtr	(void);	Returns the last IRE Method of the currently loaded set.
GetName	Str	(void);	Returns the string name of the specified method.
GetNext	NxMethodPtr	(void);	Returns the next IRE Method of the currently loaded set.
GetPrevious	NxMethodPtr	(void);	Returns the previous IRE Method of the currently loaded set.

Function	Returns	Arguments	Description
`GetThenActionCount`	Int32	(void);	Returns the number of Then-Actions found in this method (from the Then or RHS part).
`IsPrivate`	Int32	(void);	Returns boolean value indicating if the method is private.
`SetKB`	void	(NxKBCPtr *kb*);	Changes the Knowledge Base that contains the definition of this method.

NDNxObject:: Class

Function	Returns	Arguments	Description
`Create`	NxObjectPtr	(CStr *name*);	Creates an object named `name' with no specific parent. Returns the object, or NULL if an error.
`CreateObject`	NxObjectPtr	(CStr *name*);	Creates an object named `name' with the specified parent object.
`Delete`	Int32	(void);	Deletes the object named `object'.
`DeleteObject`	Int32	(NxObjectPtr c*hildObj*);	Removes an object named `childObj' from the specified parent object.
`Find`	void	(CStr *name*);	Returns the IRE Object specified by `name'. Returns NULL if not found.
`FindSlot`	NxSlotPtr	(CStr *name*);	Returns the IRE Slot specified with a property name of `name' and a parent object `parent'.
`FindSlotByProp`	NxSlotPtr	NxPropPtr *prop*);	Returns the IRE Slot specified with a property of `prop' and a parent object `parent'.
`GetChildObjectCount`	Int32	(void);	Returns the number of child objects attached directly to this object.
`GetClientData`	Long	(void);	Gets a user-defined client data value associated with this object.
`GetCount9`	Int32	(void);	Returns the number of IRE Objects currently loaded.
`GetFirst9`	NxObjectPtr	(void);	Returns the first IRE Object of the currently loaded set.
`GetIndexedChildObject`	NxObjectPtr	(Int32 *index*);	Returns the Nth child object attached directly to this object.

Function	Returns	Arguments	Description
`GetIndexedMethod`	NxMethodPtr	(Int32 *index*);	Returns the Nth method attached directly to this object.
`GetIndexedParentClass`	NxClassPtr N	(Int32 *index*);	Returns the Nth parent class attached directly to this object.
`GetIndexedParentObject`	NxObjectPtr	(Int32 *index*);	Returns the Nth child object attached directly to this object.
`GetIndexedSlot`	NxSlotPtr	(void);	Returns the Nth slot attached directly to this object.
`GetKB`	NxKBPtr	(void);	Returns the Knowledge Base associated with this object.
`GetLast9`	NxObjectPtr	(void);	Returns the last IRE Object of the currently loaded set.
`GetLinkType`	NxObjectLinkEnum	(NxObjectPtr *child*);	Returns link information: none, permanent, temporary, etc.
`GetMethodCount`	Int32	(void);	Returns the number of user-defined method attached directly to this object.
`GetName`	Str	(void);	Returns the string name of the specified object.
`GetNext`	NxObjectPtr N	(void);	Returns the next IRE Object of the currently loaded set.
`GetParentClassCount`	Int32	(void);	Returns the number of parent classes attached directly to this object.
`GetParentObjectCount`	Int32	(void);	Returns the number of parent objects attached directly to this object.
`GetPrevious`	NxObjectPtr	(void);	Returns the previous IRE Object of the currently loaded set.
`GetPrivateMethod`	NxMethodPtr	(CStr *name*);	Returns the method pointer / object of the private method named `name' attached directly to the specified object.

Function	Returns	Arguments	Description
`GetPublicMethod`	NxMethodPtr	(CStr *name*);	Returns the method pointer / object of the public method named `name' attached directly to the specified object.
`GetSlotCount`	Int32	(void);	Returns the number of slots attached directly to this object.
`MakeLinkPermanent`	Int32	(NxAtomPtr *atom*);	Changes an object's temporary link(s); to permanent link(s);.
`SetClientData`	void	(Long *data*);	Sets a user-defined client data value to be associated with this object.
`SetKB`	void	(NxKBCPtr *kb*);	Changes the Knowledge Base that contains the definition of this object

NDNxProp:: Class

Function	Returns	Arguments	Description
`Find`	NxPropPtr	(CStr *name*);	Returns the IRE Property specified by `name'.
`GetCount`	Int32	(void);	Returns the number of IRE Properties currently loaded.
`GetDataType`	NxPropDataTypeEnum	(void);	Returns the datatype of the property.
`GetFirst`	NxPropPtr	(void);	Returns the first IRE Property of the currently loaded set.
`GetFormat`	Str	(void);	Returns the format string used with the property.
`GetIndexedMethod`	NxMethodPtr	(Int32 *index*);	Returns the Nth method attached directly to this property.
`GetKB`	NxKBPtr	(void);	Returns the Knowledge Base associated with this property.
`GetLast`	NxPropPtr	(void);	Returns the last IRE Property of the currently loaded set.
`GetMethodCount`	Int32	(void);	Returns the number of user-defined method attached directly to this property.
`GetName`	Str	(void);	Returns the string name of the specified property.
`GetNext`	NxPropPtr	(void);	Returns the next IRE Property of the currently loaded set.
`GetPrevious`	NxPropPtr	(void);	Returns the previous IRE Property of the currently loaded set.
`GetPrivateMethod`	NxMethodPtr	(CStr *name*);	Returns the method pointer / object of the private method named `name' attached directly to the specified property.
`GetPublicMethod`	NxMethodPtr	(CStr *name*);	Returns the method pointer / object of the public method named `name' attached directly to the specified property.
`SetFormat`	void	(CStr *format*);	Sets the format string used with the property.

Function	Returns	Arguments	Description
`SetKB`	void	(NxKBCPtr *kb*);	Changes the Knowledge Base that contains the definition of this property.

NDNxRule:: Class

Function	Returns	Arguments	Description
`Find`	NxRulePtr	(CStr *name*);	Returns the IRE Rule specified by `name'. Returns NULL if not found.
`GetComments`	Str	(void);	Returns the user comments string associated with the rule.
`GetCount`	Int32	(void);	Returns the number of IRE Rules currently loaded.
`GetCurrent`	Int32	(void);	Returns the number of IRE Rules currently loaded.
`GetElseActionCount`	Int32	(void);	Returns the number of Else-Actions found in this rule (from the Else or EHS part).
`GetFirst`	NxRulePtr	(void);	Returns the first IRE Rule of the currently loaded set.
`GetHypo`	NxSlotPtr	(void);	Return the hypothesis of this rule.
`GetIfConditionCount`	Int32	(void);	Return the hypothesis of this rule.
`GetIndexedElseAction`	Str	(Int32 *index*);	Returns the string representation of the Nth EHS of this rule.
`GetIndexedIfCondition`	Str	(Int32 *index*);	Returns the string representation of the Nth LHS of this rule.
`GetIndexedThenAction`	Str	(Int32 *index*);	Returns the string representation of the Nth RHS of this rule.
`GetInferencePriority`	Int32	(void);	Returns the inference priority number associated with this rule.
`GetInferenceSlot`	NxSlotPtr	(void);	Returns the inference priority slot associated with this rule.

Function	Returns	Arguments	Description
`GetKB`	NxKBPtr	(**void);**	Returns the Knowledge Base associated with this rule.
`GetLast`	NxRulePtr	(void);	Returns the last IRE Rule of the currently loaded set.
`GetName`	Str	(void);	Returns the string name of the specified rule.
`GetNext`	NxRulePtr	(void);	Returns the next IRE Rule of the currently loaded set.
`GetPrevious`	NxRulePtr N	(void);	Returns the previous IRE Rule of the currently loaded set.
`GetThenActionCount`	Int32	(void);	Returns the number of Then-Actions found in this rule (from the Then or RHS part).
`GetValue`	Int32	(void);	Retrieves the boolean value of this rule (true / false / not-known / unknown).
`GetWhy`	Str	(void);	Returns the Why information associated with this rule.
`IsHypoSuggested`	Int32	(void);	Return a boolean indicating whether the hypothesis of this rule is to be evaluated by the rule engine.
`IsKnown`	Int32	(void);	Returns information on whether the rule is KNOWN.
`IsNotknown`	Int32	(void);	Returns information on whether the rule is NOT-KNOWN.
`IsUnknown`	Int32	(void);	Returns information on whether the rule is UN-KNOWN.
`SetKB`	void	(NxKBCPtr *kb*);	Changes the Knowledge Base that contains the definition of this rule.

Function	Returns	Arguments	Description
`SuggestHypo`	Int32	(void);	Suggest the hypothesis of this rule to be evaluated by the rule engine.
`SuggestHypo2`	Int32	(NxRuleSugPrioEnum *strategy*);	Suggest the hypothesis of this rule to be evaluated by the rule engine using the Suggest strategy provided.
`UnsuggestHypo`	Int32	(void);	Removes the hypothesis of this rule from the agenda of the rule engine.

NDNxSlot:: Class

Function	Returns	Argument	Description
`Find`	NxSlotPtr	(void);	Returns the IRE Slot specified by `name'. Returns NULL if not found.
`GetChoiceCount`	Int32	void);	Returns the number of choices the rule engine will present when asking a slot question, based on possible values found within the loaded Knowledge Bases.
`GetClientData`	Int32	(void);	Gets a user-defined client data value associated with this slot.
`GetContextCount`	Int32	(void);	Returns the number of hypotheses that are in the context of this hypothesis.
`GetCountData`	Int32	(void);	Returns the number of IRE Data Slots currently loaded.
`GetCountHypo`	Int32	(void);	Returns the number of IRE Hypothesis Slots currently loaded.
`GetCurrent`	NxSlotPtr	(void);	Retrieves the current slot, which may be NULL if there is none.
`GetDataType`	NxSlotDataTypeEnum	(void);	Returns the datatype of the slot.
`GetFirstData3`	NxSlotPtr	(void);	Returns the first IRE Data Slot of the currently loaded set.
`GetFirstHypo`	NxSlotPtr	(void);	Returns the first IRE Hypothesis Slot of the currently loaded set.
`GetFormat`	Str	(void);	Returns the format string used with the slot.

Function	Returns	Argument	Description
GetIndexedChoice	Str	(Int32 index);	Returns the Nth choice the rule engine will present when asking a slot question, based on possible values found within the loaded Knowledge Bases.
GetIndexedContext	NxSlotPtr	(Int32 index);	Returns the Nth context hypothesis.
GetIndexedMethod	NxMethodPtr	(Int32 index);	Returns the Nth method attached directly to this slot.
GetIndexedSuggestList	NxSlotPtr	(void);	Returns the Nth slot in the suggest list.
GetIndexedVolunteerList	NxSlotPtr	(void);	Returns the Nth slot in the volunteer list.
GetInferencePriority	Int32	(void);	Returns the inference priority number associated with this slot.
GetInferenceSlot	NxSlotPtr	(void);	Returns the inference priority slot associated with this slot.
GetInheritancePriority	Int3 2	(void);	Returns the inheritance priority number associated with this slot.
GetInheritanceSlot	NxSlotPtr	(void);	Returns the inheritance priority slot associated with this slot.
GetKB	NxKBPtr	(void);	Returns the Knowledge Base associated with this slot.
GetLastData	NxSlotPtr	(void);	Returns the last IRE Data Slot of the currently loaded set.
GetLastHypo	NxSlotPtr	(void);	Returns the last IRE Hypothesis Slot of the currently loaded set.
GetMethodCount	Int32	(void);	Returns the number of user-defined method attached directly t
GetName	void	(void);	Returns the string name of the specified slot.

Function	Returns	Argument	Description
GetNextData	NxSlotPtr	(void);	Returns the next IRE Data Slot of the currently loaded set. This requires a valid data slot (eg: from first / next).
GetNextHypo	NxSlotPtr	(void);	Returns the next IRE Hypothesis Slot of the currently loaded set.
GetParent	NxAtomPtr	(void);	Returns the parent (object or class) referenced by the specified slot.
GetPreviousData	NxSlotPtr	(void);	Returns the previous IRE Data Slot of the currently loaded set. This requires a valid data slot (eg: from first / next).
GetPreviousHypo	NxSlotPtr	(void);	Returns the previous IRE Hypo Slot of the currently loaded set. This requires a valid Hypo slot (eg: from first / next).
GetPrivateInitValue	Str	(void);	Returns a string containing the private (not-inheritable) initial value for this slot.
GetPrivateMethod	NxMethodPtr	(CStr *name*);	Returns the method pointer / object of the private method named `name' attached directly to the specified slot.
GetPrompt	Str	(void);	Returns the prompt string to be used when asking a question about this slot.
GetProperty	NxPropPtr	(void);	Returns the property referenced by the specified slot.
GetPublicInitValue	VStr	(void);	Returns a string containg the public (inheritable) initial value for this slot.
GetPublicMethod	NxMethodPtr	(CStr *name*);	Returns the method pointer / object of the public method named `name' attached directly to the specified slot.

Function	Returns	Argument	Description
`GetQuestionWindow`	Str	(void);	Returns the name of an Open Interface question window to be used when asking a question.
`GetStrategy`	BoolEnum	NxSlotStratEnum *strategy*);	Returns a boolean indicating the setting for the specified strategy.
`GetStringValue`	Str	(void);	Retrieves the value of a slot as a formatted string (using any format specified with this slot.
`GetSuggestListCount`	Int32	(void);	Returns the number of slots in the suggest list.
`GetValidationExecute`	Str	(void);	Returns the name of the Execute to be invoked to provide additional user validation for this slot.
`GetValidationFunction`	Str	(void);	Returns a string representation of a validation function to be evaluated to determine whether this slot represents a valid response.
`GetValidationHelp`	Str	(void);	Returns the string that will be used to provide additional help when a validation violation is discovered on this slot.
`GetValue`	VarPtr	(void);	Retrieves the value of this slot, ignoring formats in a variant.
`GetVolunteerListCount`	Int32	(void);	Returns the number of slots in the volunteer list.
`GetWhy`	Str	(void);	Returns the Why information associated with this slot.
`IsDefaultStrategy`	BoolEnum	(NxSlotDefStratEnum *strategy*);	Returns a boolean indicating whether the specified strategy is the default strategy.
`IsHypo`	Int3 2	(void);	Returns a boolean indicating whether the slot is used as the hypothesis of a rule.
`IsKnown`	Int3 2	(void);	Returns information on whether the slot is KNOWN.

Function	Returns	Argument	Description
IsNotknown	Int32	(void);	Returns information on whether the slot is NOT-KNOWN.
IsPrivate	Int32	(void);	Returns a boolean indicating whether this slot is private or not.
IsSuggested	Int32	(void);	Returns a boolean indicating whether the slot has been suggested to be evaluated by the rule engine.
IsUnknown	Int32	(void);	Returns information on whether the slot is UNKNOWN.
SetClientData	void	(Long *data*);	Sets a user-defined client data value to be associated with this slot.
SetFormat	void	(CStr *format*);	Sets the format string used with the slot.
SetKB	void	(NxKBCPtr *kb*);	Changes the Knowledge Base that contains the definition of this slot.
SetValue	void	(VarCPtr *value*);	Volunteers the specified value into this slot, using the Default Volunteer Strategy
Suggest	Int32	(void);	Enters a slot to be evaluated by the rule engine when it starts / resumes.
Suggest2	Int32 N	NxSlotSugPrioEnum *strategy*);	Enters a slot to be evaluated by the rule engine when it starts / resumes.
Unsuggest	Int32	(void);	Removes a slot from the agenda of the rule engine. Returns an integer status.
Volunteer	Int32	(VarCPtr *value*);	Volunteers a value to the slot using the default volunteer strategy.
Volunteer2	Int32	(VarCPtr *value*, NxSlotVolStratEnum *strategy*);	Volunteers a value to the slot using the specified volunteer strategy.