CORE® 5

COREscript Construct Reference Guide

Vitech

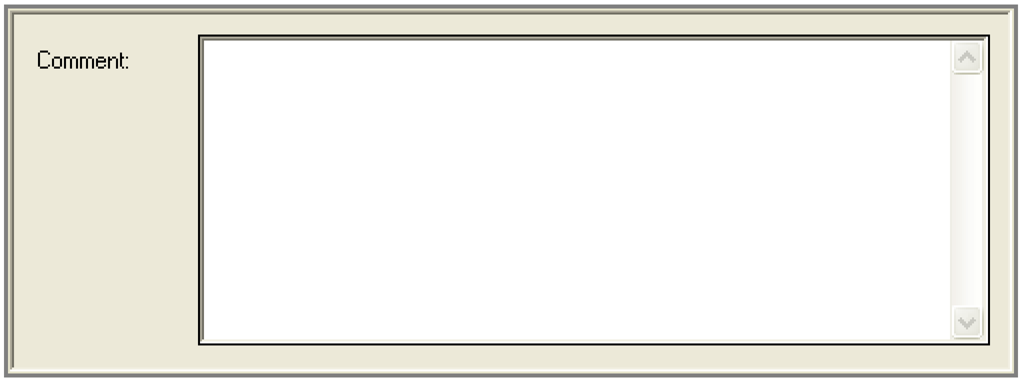
Vitech Corporation
2070 Chain Bridge Road, Suite 100
Vienna, Virginia 22182-2536
703.883.2270 FAX: 703.883.1860
Customer Support: support@vitechcorp.com
www.vitechcorp.com

Revision Date: September 2007

Table of Contents

Control Constructs

Block Construct



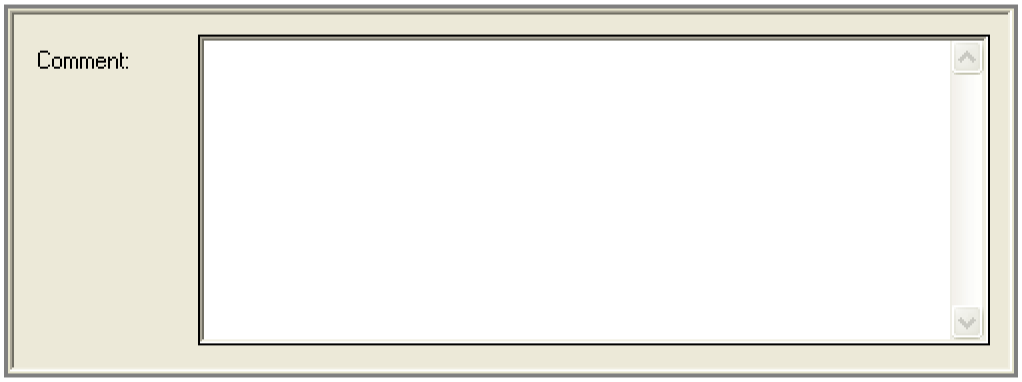
Description

This construct consists of two parts on different lines of the Report Editor’s Script pane that may enclose other constructs indented between them. A block is simply an artificial, non-functional construct for the purpose of promoting code decomposability, composability, reusability and documentation. It encloses code that can then be copied and moved as a unit, and it breaks up the flow of the script into more manageable units.

Comment

An explanation of what is accomplished by the constructs enclosed within the block, block assumptions, etc.

Disabled Code Construct



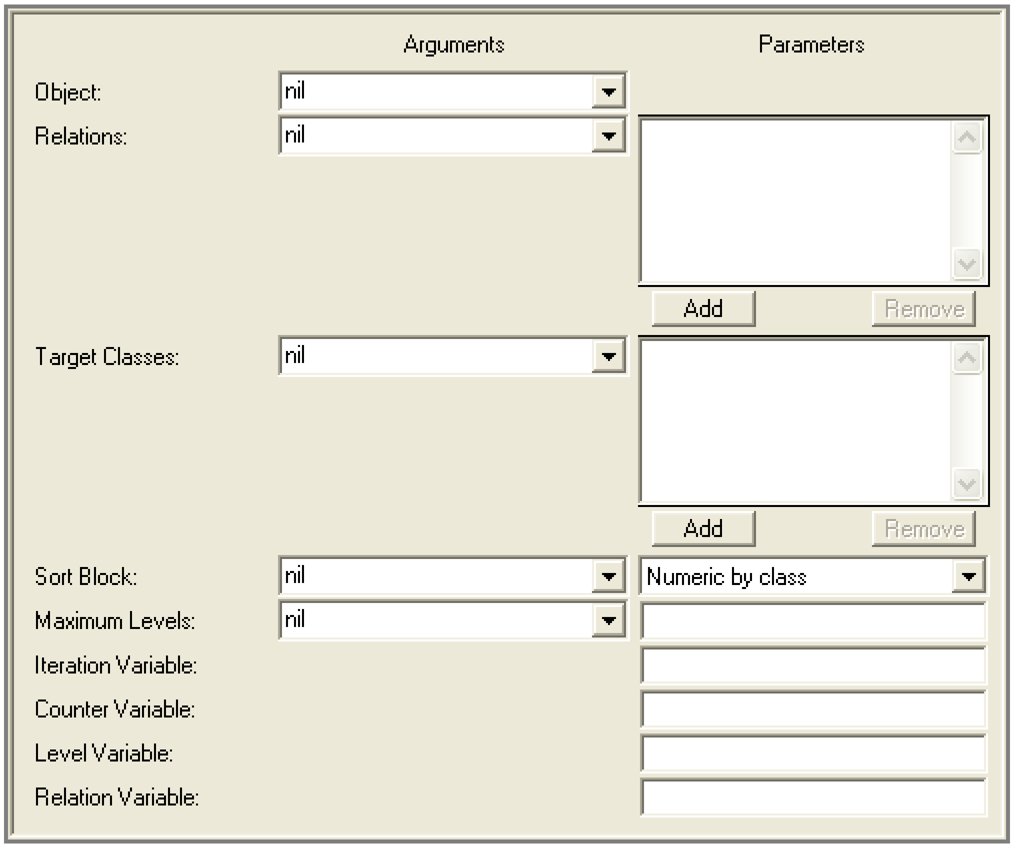
Description

This construct consists of two parts on different lines of the Report Editor’s Script pane that may enclose other constructs indented between them. The enclosed constructs are not executed.

Comment

An explanation of why the enclosed constructs have been disabled.

Hierarchy Iterate Construct



	Arguments	Parameters
Object:	nil	
Relations:	nil	Add Remove
Target Classes:	nil	Add Remove
Sort Block:	nil	Numeric by class
Maximum Levels:	nil	
Iteration Variable:		
Counter Variable:		
Level Variable:		
Relation Variable:		

Description

Given a starting point (the root) and a collection of valid relations and target classes, the Hierarchy Iterate Construct generates a tree of elements and iterates across the tree. The tree is expanded depth first (left to right) and uses a pre-ordered traversal. This construct is recursion-safe – the second time an element is encountered, it is reported but its child nodes are not traversed.

Object

An element specifying the root for the tree. This is the starting point from which the hierarchy will be built based upon the specified relations and target classes.

Relations

The collection of valid relations to use when building the hierarchy for iteration. This is equivalent to the set of relations you would specify if building a custom hierarchy view.

Target Classes

The collection of valid target classes to use when building the hierarchy for iteration. Only those targets within the classes specified will appear in the hierarchy. For example, if you wished to iterate through a Requirement hierarchy and all Functions that are specified by the Requirement hierarchy, specifying Requirement and Function as the valid target classes would prevent any other classes (e.g., Components) from appearing within the iteration.

Sort Block

Sort block definition of the sort block to be used to order the children of a node within the hierarchy. Child nodes are ordered first by relation and then by the criteria in the specified sort block.

Maximum Levels

An integer specifying the maximum depth of the tree. Nodes will be expanded until either the leaf-level nodes have no children (as defined by the set of valid relations and target classes) or the leaf nodes are at the maximum depth. The root of the tree is defined as
level 1.

Iteration Variable

The variable which will hold the current element during the iteration.

Counter Variable

The variable, which will hold the total number of elements, iterated over so far.

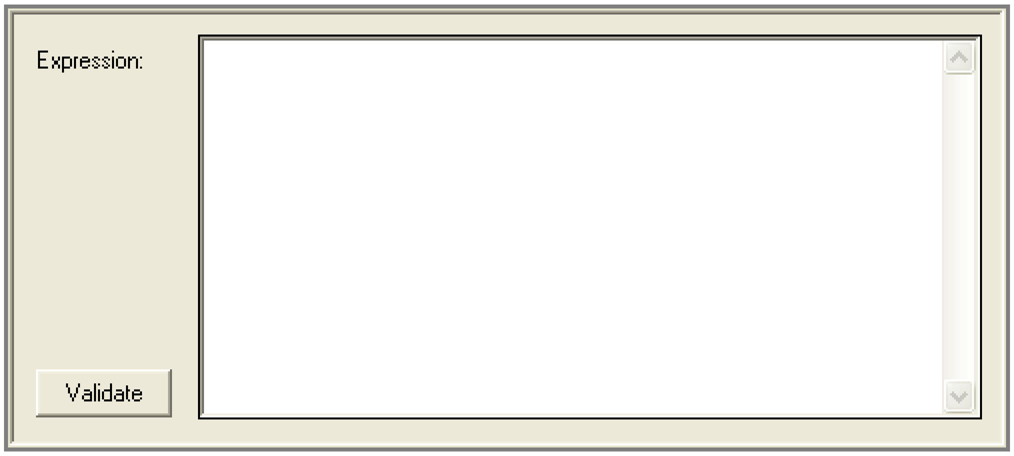
Level Variable

The variable which will hold the level of the current element within the hierarchy (where the root of the tree is defined as level 1).

Relation Variable

The variable, which will hold the relation, used to reach the current element (the relation from the parent node to the child node).

If Then Construct



Description

This construct consists of two parts on different lines of the Report Editor’s Script pane that may enclose other constructs indented between them. The enclosed constructs are executed if and only if *expression* evaluates to *true*.

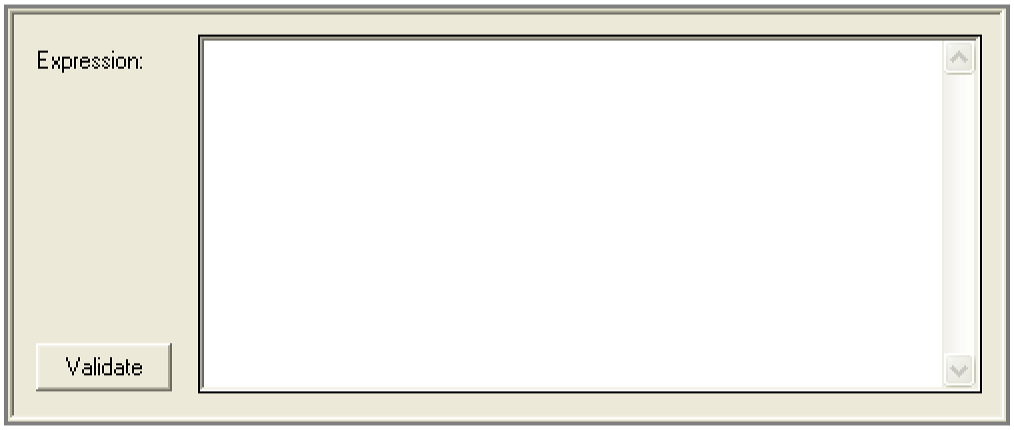
Expression

An expression consisting of operators in the CORE expression language, variables defined using the Edit Variables command under the Properties menu of the Report Editor, and constants that can be coded in some data types. The result of this expression must be a Boolean.

Validate

Validates the syntax of the *expression* to ensure that neither operators nor operands appear in unexpected locations, all operators are recognized, and all variables have been defined. No type or bounds checking can be performed.

If Then Else Construct



Description

This construct consists of three parts on different lines of the Report Editor’s Script pane that may enclose two groups of other constructs indented between them. This construct implements branch logic. The first enclosed group of constructs (the *then* branch) is executed if *expression* evaluates to *true*. The second enclosed group of constructs (the *else* branch) is executed if *expression* evaluates to *false*.

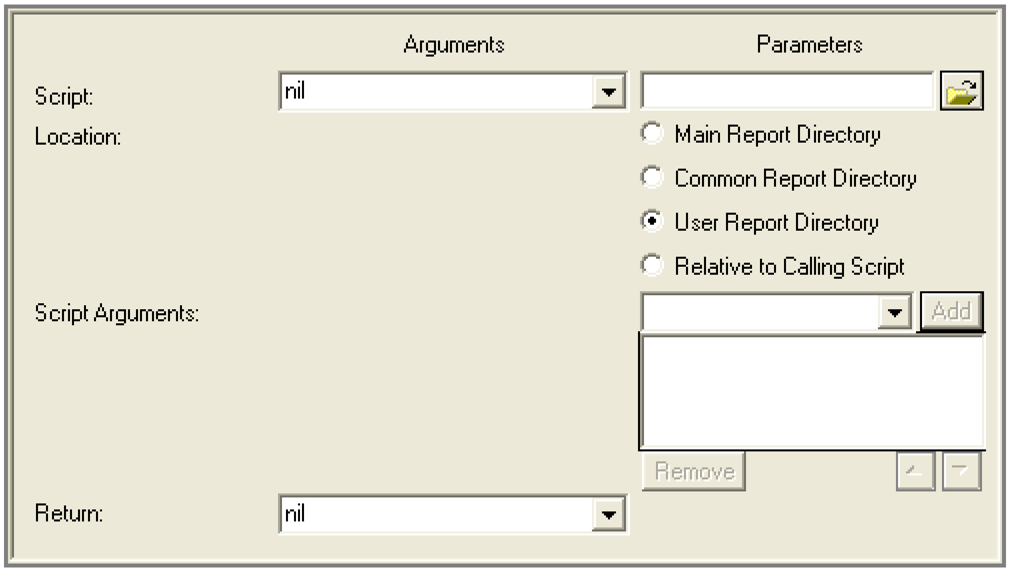
Expression

An expression consisting of operators in the CORE expression language, variables defined using the Edit Variables command under the Properties menu of the Report Editor, and constants that can be coded in some data types. The result of this expression must be a Boolean.

Validate

Validates the syntax of the *expression* to ensure that neither operators nor operands appear in unexpected locations, all operators are recognized, and all variables have been defined. No type or bounds checking can be performed.

Include Script



	Arguments	Parameters
Script:	nil	
Location:		( ) Main Report Directory ( ) Common Report Directory (●) User Report Directory ( ) Relative to Calling Script
Script Arguments:		
Return:	nil	

Description

Transfers control to the specified script file, which is executed in its entirety before control returns to the next construct in the current script (much like a subroutine or procedure call). The invoked script and any scripts it includes have access to global variables declared in the current report execution. Any duplicate variable declarations in the included script reference variables already defined in the current report execution. Note that double-clicking the left mouse button with the pointer over an Include Script construct in the Script pane of a Report Editor opens a new Report Editor containing the script file included by the construct, if any.

Script

The name of the script file to be executed. Can be specified either by selecting a variable from the drop-down on the left, manually entering as a parameter, or selecting the file using a standard Windows file dialog (via the Script Browse Button). If a manually entered file name does not currently exist the parameter field background will change to yellow.

Script Browse Button

The Script browse button opens a standard Windows file dialog. The user must specify a script file in the *location* directory as described below or in a subdirectory thereof. Canceling the dialog leaves the previous selection unchanged.

Location

The CORE directory where the script is located. When set to main, common, or user, the construct uses the settings established in the Report Directories portion of the CORE user preferences to locate the included script file in the appropriate directory.

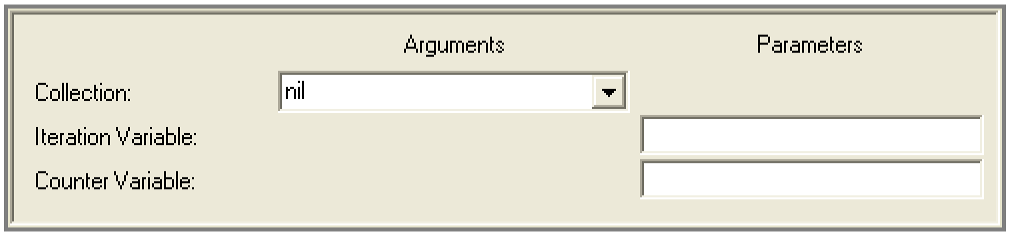
Script Arguments

The ordered collection of variables whose values will be passed to the invoked script as arguments. If the number of variables does match exactly with the number of parameters declared in the script properties portion of the invoked script, an error will result. Any values assigned to the parameters from within the invoked script will not affect the corresponding arguments in the current script.

Return

Variable to hold the result, if any, from executing the include script. In order for an include script to return a value, it must complete with a Return construct that has an argument.

Iterate Construct



Collection: nil
Iteration Variable:
Counter Variable:

Description

This construct consists of two parts on different lines of the Report Editor's Script pane that may enclose other constructs indented between them. The enclosed constructs are executed once for each object in the *collection*.

On each iteration, a different object in the *collection* is assigned to the *iteration variable* if one is defined. If the *collection* is an array or ordered collection, this construct iterates over the objects in their ordered defined by the *collection*. If the *collection* is a string or symbol, this construct iterates over its characters in order.

On the first iteration, the *counter variable* is assigned the value 1. On every subsequent iteration, its value is incremented by 1.

Collection

Name of a variable containing the collection over which to iterate, which can be of one of the following data types:

- Array
- Bag
- Dictionary
- Ordered Collection
- Set
- String
- Symbol.

If the collection is a dictionary, this construct iterates over the values stored in the dictionary, not the keys whereby the values are accessed.

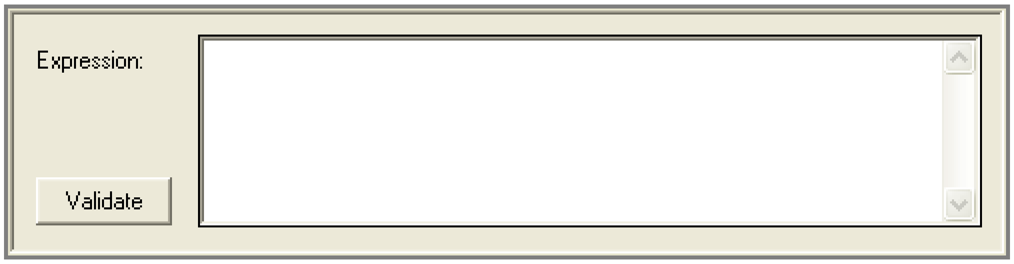
Iteration Variable

Name of the iteration variable. This variable is declared within the scope of this construct but will not be accessible after this construct terminates.

Counter Variable

Name of the counter variable. This variable is declared within the scope of this construct but will not be accessible after this construct terminates.

Repeat Until Construct



Description

This construct consists of two parts on different lines of the Report Editor’s Script pane that may enclose other constructs indented between them. This construct implements loop logic. After each time the group of enclosed constructs is executed, the *expression* is tested and, if it does not evaluate to *true*, control flows back to the beginning. Note that the enclosed constructs are guaranteed to be executed at least once, since the test is performed at the end.

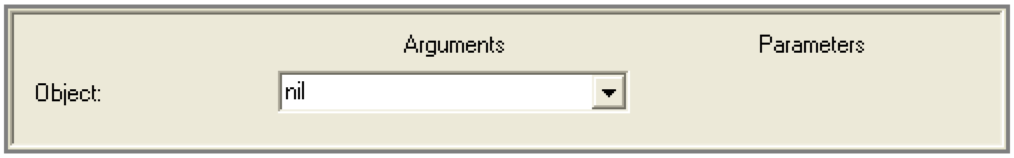
Expression

An expression consisting of operators in the CORE expression language, variables defined using the Edit Variables command under the Properties menu of the Report Editor, and constants that can be coded in some data types. The result of this expression must be a Boolean.

Validate

Validates the syntax of the *expression* to ensure that neither operators nor operands appear in unexpected locations, all operators are recognized, and all variables have been defined. No type or bounds checking can be performed.

Return



Object: nil

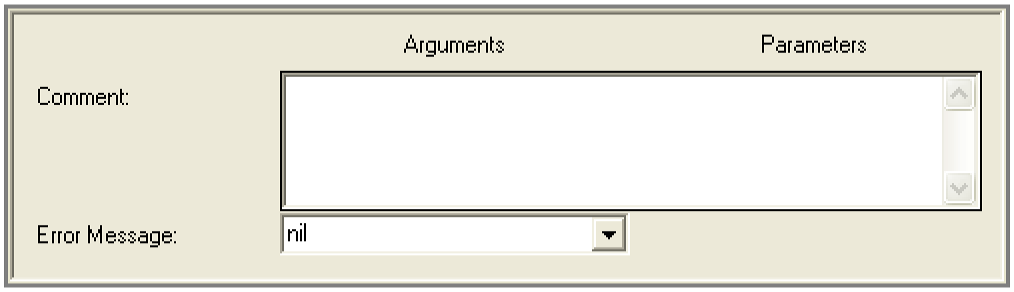
Description

Terminates execution of an include script returning control to the calling script. If the Return Construct is encountered in the top-level script, the report is exited.

Object

Variable that contains the object to be returned from the script.

Update Block Construct



nil

Description

This construct consists of three parts on different lines of the Report Editor's Script pane that may enclose two groups of other constructs indented between them. This construct wraps database updates with an error handler to process errors encountered due to invalid update attempts. The first enclosed group is the block of constructs to be executed. If a database update error is encountered, the second enclosed group of constructs (the *error handler*) is executed.

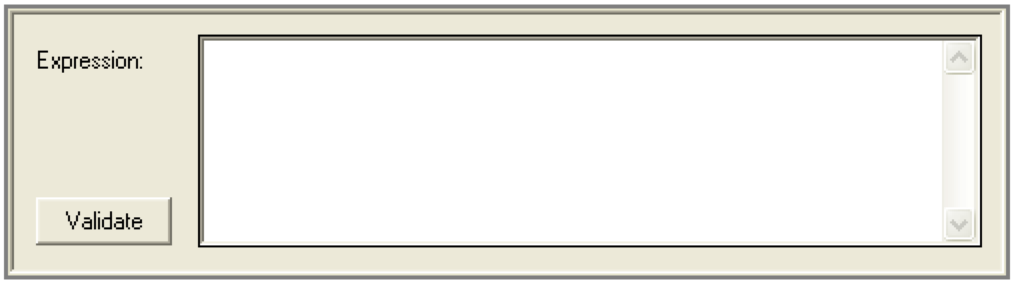
Comment

An explanation of what is accomplished by the constructs enclosed within the block, block assumptions, etc.

Error Message

Variable to hold the error message if an error is encountered within the protected code of this construct

While Do Construct



Description

This construct consists of two parts on different lines of the Report Editor's Script pane that may enclose other constructs indented between them. This construct implements loop logic. If the *expression* evaluates to *true*, the enclosed group of constructs is executed repeatedly until the *expression* evaluates to *false*. Note that the enclosed constructs are not guaranteed to be executed, since the *expression* is tested before any of them are executed.

Expression

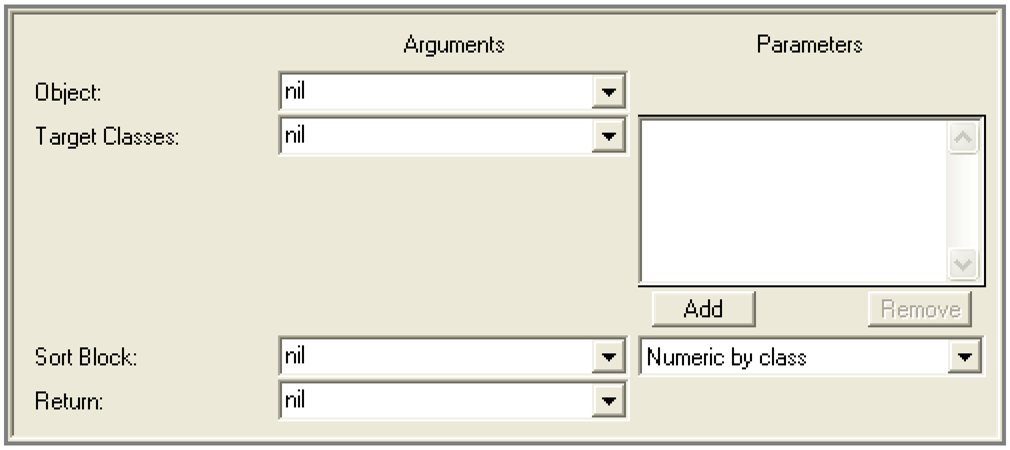
An expression consisting of operators in the CORE expression language, variables defined using the Edit Variables command under the Properties menu of the Report Editor, and constants that can be coded in some data types. The result of this expression must be a Boolean.

Validate

Validates the syntax of the *expression* to ensure that neither operators nor operands appear in unexpected locations, all operators are recognized, and all variables have been defined. No type or bounds checking can be performed.

Database Query Constructs

Children Query



Object: nil
Target Classes: nil
Sort Block: nil | Numeric by class
Return: nil

Description

Assigns to the *return* variable the collection of all the elements in the CORE database that are targets of parent-child relationships with the *receiver* element. The parent-child relationships are defined by the parent-child relation established in the CORE schema for the class of the *receiver* element. There may be no parent-child relation for the class of the *receiver*. The parent-child relation is used to do attribute value roll-ups, where an attribute value in one element is computed from the corresponding attribute values in its children.

The children elements may not all be of the same class. To filter out all children except those in one or more particular classes, specify the *target classes* collection. The returned collection will be ordered according to the specified *sort block*.

Object

Variable containing the element to query.

Target Classes

Variable containing a collection of filter classes. Can be specified either by a variable from the drop-down on the left or manual additions into the collection on the right. Can be of any of the following data types:

- Array
- Bag
- Dictionary
- Ordered Collection
- Set
- Sorted Collection.

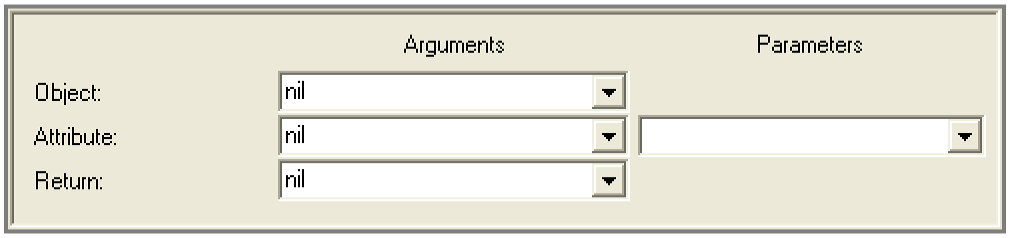
Sort Block

Sort block definition of a sort block defined in the CORE project. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right. This sort criteria continues to govern the behavior of the sorted collection that results from this construct.

Return

Variable to hold the sorted collection of elements that results from this construct.

Element Attribute Query



	Arguments	Parameters
Object:	nil	
Attribute:	nil	
Return:	nil	

Description

Assigns to the *return* variable the value of the specified *attribute* of the element in the CORE database specified by *object*. Returns *nil* if either the attribute has not been assigned a value or the user does not have sufficient permissions to view the value.

Object

Variable containing the element from the CORE database to query.

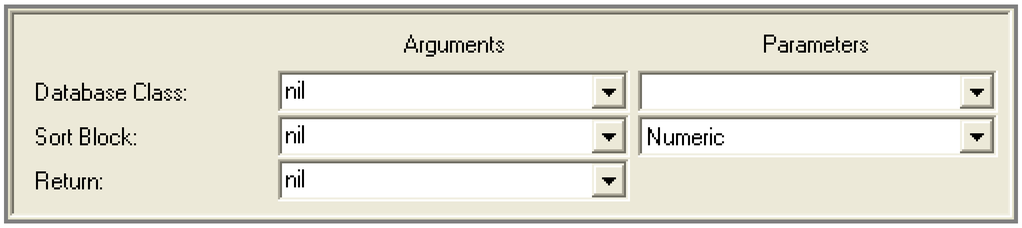
Attribute

Symbol naming the attribute which value is desired. The attribute should be defined in the CORE schema for the class of the *object*. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Return

Variable to hold the result of this construct.

Elements Query



	Arguments	Parameters
Database Class:	nil	
Sort Block:	nil	Numeric
Return:	nil	

Description

Assigns to the *return* variable a collection of all the elements in the CORE database that are of the specified *class*. The returned collection will be ordered according to the specified *sort block*.

Database Class

Symbol naming a class defined in the CORE schema. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

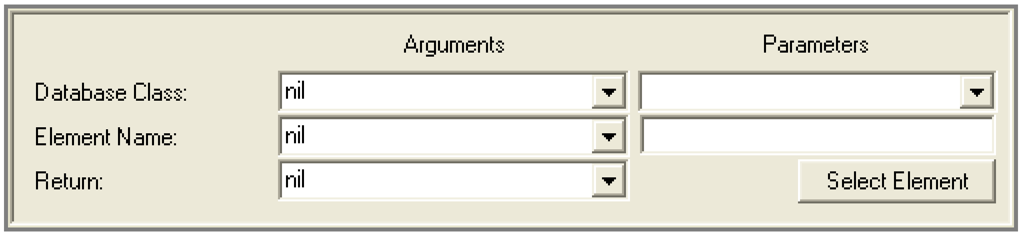
Sort Block

Sort block definition of a sort block defined in the CORE project. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right. This sort criteria continues to govern the behavior of the sorted collection that results from this construct.

Return

Variable to hold the sorted collection of elements that is the result of this construct.

Named Element Query



Description

Assigns to the *return* variable the element in the CORE database that has the specified name and is of the specified *class*. The construct will return nil if either the requested element is not in the database or the user does not have sufficient permissions to access the element.

Database Class

A class object defined in the CORE schema. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

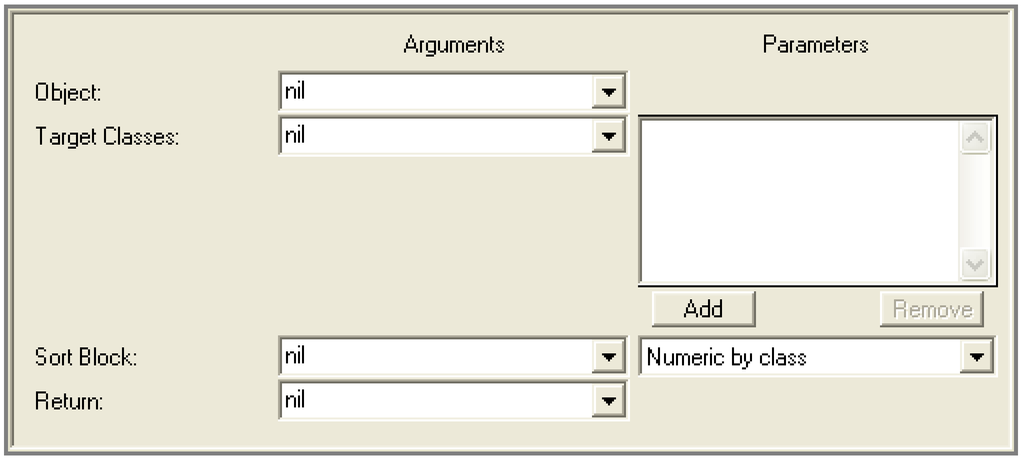
Element Name

Symbol naming the element to be returned. Can be specified either by a variable from the drop-down on the left or a manual selection using an element selection window opened by pressing the Select Element button.

Return

Variable to hold the result of the construct.

Parents Query



Object: nil
Target Classes: nil
Sort Block: nil — Numeric by class
Return: nil

Description

Assigns to the *return* variable the collection of all the elements in the CORE database that are sources (vice targets) of parent-child relationships with the *receiver* element (i.e., the *receiver* is the target of the relationships). The parent-child relationships are defined by parent-child relations established in the CORE schema with the class of the *receiver* element as the target class. The parent-child relation is used to do attribute value roll-ups, where attribute values of a child are used to compute the corresponding attribute values of its parents.

There is at most a single parent-child relation established for each class; however, since this operator views the relations in the opposite direction (i.e., their complements from child to parent), the class of the *receiver* may be a target class of multiple parent-child relations, each with a different source class. To filter out all parents except those in one or more particular classes, specify the *target classes* collection. The returned collection will be ordered according to the specified *sort block*.

Object

Variable containing the element to query.

Target Classes

Variable containing a collection of filter classes. Can be specified either by a variable from the drop-down on the left or manual additions into the collection on the right. Can be of any of the following data types:

Array
Bag
Dictionary
Ordered Collection
Set
Sorted Collection.

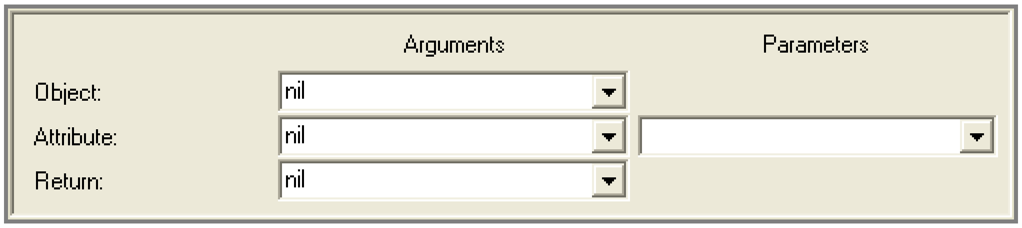
Sort Block

Sort block definition of a sort block defined in the CORE project. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right. This sort criteria continues to govern the behavior of the sorted collection that results from this construct.

Return

Variable to hold the sorted collection of elements that results from this construct.

Relationship Attribute Query



	Arguments	Parameters
Object:	nil	
Attribute:	nil	
Return:	nil	

Description

Assigns to the *return* variable the value of the specified *attribute* of the relationship in the CORE database specified by *object*.

Object

Variable containing the relationship from the CORE database to query.

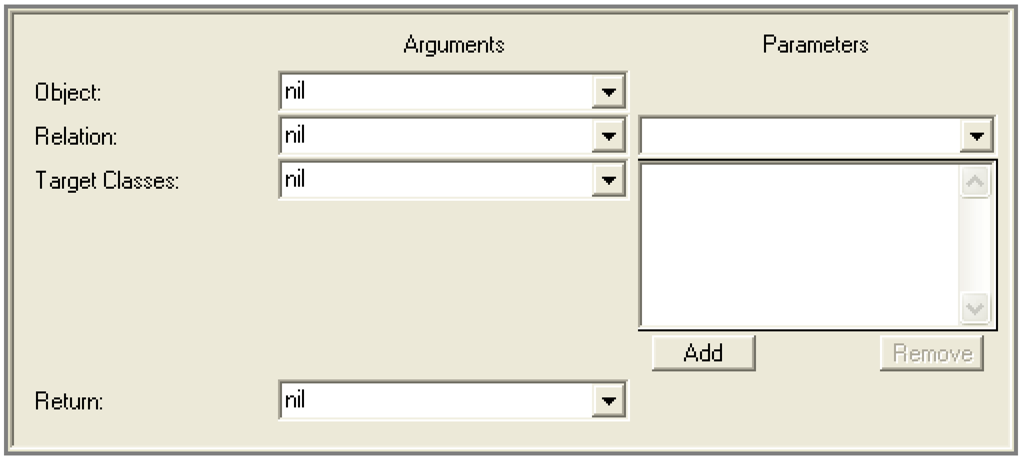
Attribute

Symbol naming the attribute which value is desired. The attribute should be defined in the CORE schema for the relation of the *object*. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Return

Variable to hold the result of this construct.

Relationships Query



Description

Assigns to the *return* variable a collection of all the relationships in the CORE database that are of the specified *relation* and have the element specified by *object* as their source. To filter out all relationships except those, which targets are in one or more particular classes, specify the *target classes* collection.

Object

Variable containing the element from the CORE database to query.

Relation

Symbol naming a relation defined in the CORE schema. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Target Classes

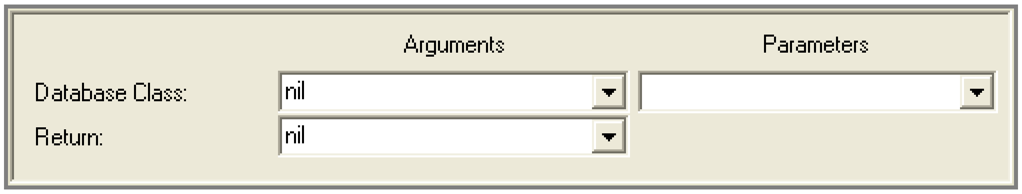
Variable containing a collection of filter classes. Can be specified either by a variable from the drop-down on the left or manual additions into the collection on the right. Can be of any of the following data types:

Array
Bag
Dictionary
Ordered Collection
Set
Sorted Collection.

Return

Variable to hold the set of relationships that results from this construct.

Root Folder Query



Database Class:	nil	
Return:	nil	

Description

Assigns to the *return* variable the element folder object for the specified *class*. Each class has a root-level folder whose name is the name of the class.

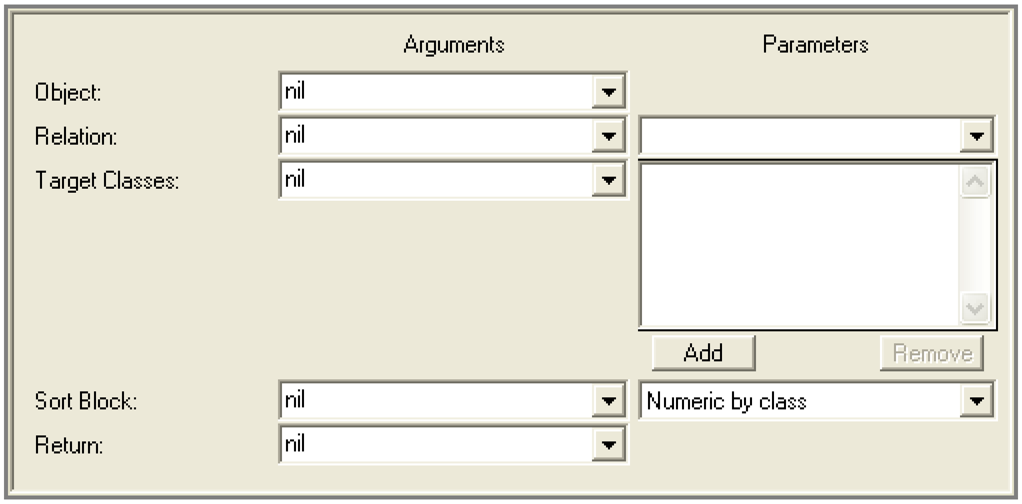
Database Class

A class object defined in the CORE schema. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Return

Variable to hold the result from this construct.

Targets Query



	Arguments	Parameters
Object:	nil	
Relation:	nil	
Target Classes:	nil	
Sort Block:	nil	Numeric by class
Return:	nil	

Description

Assigns to the *return* variable a collection of all the elements in the CORE database that are of the specified *relation* and have the element specified by *object* as their source. The returned collection will be ordered according to the specified *sort block*. To filter out all targets except those in one or more particular classes, specify the *target classes* collection.

Object

Variable containing the element from the CORE database to query.

Relation

Symbol naming a relation defined in the CORE schema. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Target Classes

Variable containing a collection of filter classes. Can be specified either by a variable from the drop-down on the left or manual additions into the collection on the right. Can be of any of the following data types:

- Array
- Bag
- Dictionary
- Ordered Collection
- Set
- Sorted Collection.

Sort Block

Sort block definition of a sort block defined in the CORE project. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right. This sort criteria continues to govern the behavior of the sorted collection that results from this construct.

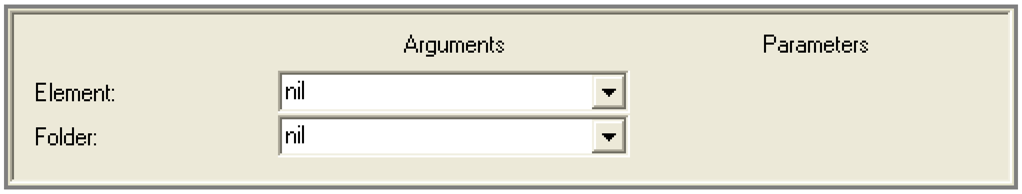
Return

Variable to hold the sorted collection of elements that results from this construct.

THIS PAGE INTENTIONALLY BLANK

Database Update Constructs

Add Element To Folder



	Arguments
Element:	nil
Folder:	nil

Description

Adds the specified *element* to the *folder* without removing the *element* from any other folder. If the user does not have write permission for the *element*, an error will be signaled.

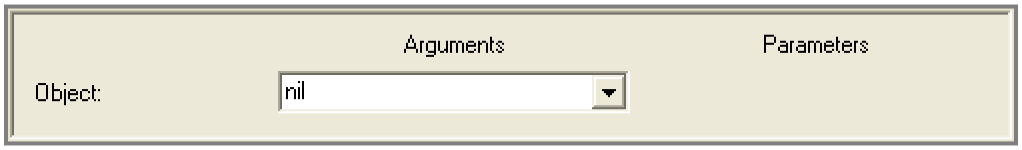
Element

Variable containing the element to add to the folder.

Folder

Variable containing the folder to which the element should be added.

Delete Object



Object: nil

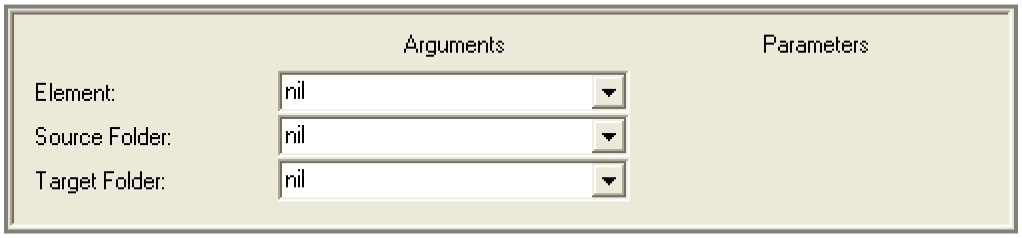
Description

Deletes the specified *object* (either an element, relationship, or folder) from the database. If the user does not have permission to delete the *object*, an error will be signaled.

Object

Variable containing the object (an element, relationship or folder) to be deleted.

Move Element To Folder



	Arguments	Parameters
Element:	nil	
Source Folder:	nil	
Target Folder:	nil	

Description

Removes the specified *element* from the *source folder* and adds it to the *target folder*. Any other folders containing the *element* are unaffected. Note that if the source folder does not contain the specified *element*, no error will be signaled and the element will be added to the *target folder*.

If the user does not have write permission for the element, an error will be signaled.

Element

Variable containing the element to move.

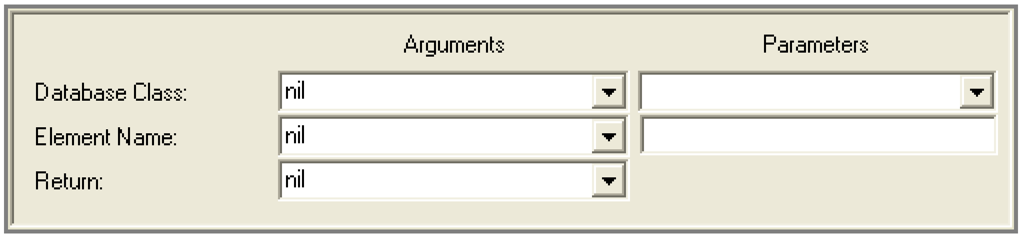
Source Folder

Variable containing the folder from which to remove the *element*.

Target Folder

Variable containing the folder to which the *element* should be added.

New Element



	Arguments	Parameters
Database Class:	nil	
Element Name:	nil	
Return:	nil	

Description

Creates a new element of the specified *database class* with the specified *element name* and assigns it to the *return* variable. An error is signaled and *nil* is returned if the element cannot be created due to a name conflict or insufficient user permissions to create elements.

Database Class

Symbol naming a class defined in the CORE schema. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

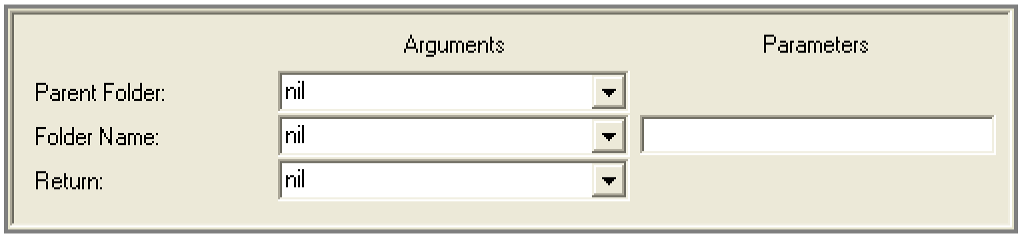
Element Name

Symbol naming the element to be created. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Return

Variable to hold the result of this construct – the new element if the operation is successful or *nil* if it is not.

New Folder



	Arguments	Parameters
Parent Folder:	nil	
Folder Name:	nil	
Return:	nil	

Description

Creates a new folder with the specified *folder name* as a subfolder of *parent folder* and assigns it to the *return* variable. An error is signaled and *nil* is returned if the folder cannot be created due to a name conflict or insufficient user permissions to create folders.

Parent Folder

Variable containing the folder in which the new subfolder is to be created.

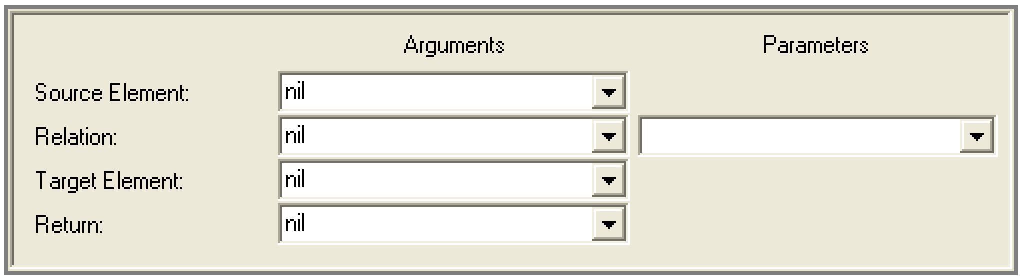
Folder Name

Symbol naming the folder to be created. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Return

Variable to hold the result of this construct – the new folder if the operation is successful or *nil* if it is not.

New Relationship



	Arguments	Parameters
Source Element:	nil	
Relation:	nil	
Target Element:	nil	
Return:	nil	

Description

Creates a new relationship of type *relation* between the specified elements and assigns it to the *return* variable. An error is signaled and *nil* is returned if the relationship cannot be created due to a schema conflict, a database constraint, or insufficient user permissions to create the relationship.

Source Element

Variable containing the source element of the new relationship. This is equivalent to the element selected in CORE when selecting a relation name and issuing the “Add Targets” command.

Relation

Relation defined in the CORE schema. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

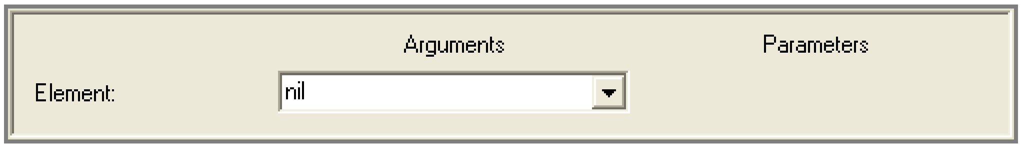
Target Element

Variable containing the target of the new relationship. This is equivalent to the element specified in the “Add Targets” dialog in CORE.

Return

Variable to hold the result of this construct – the new relationship if the operation is successful or *nil* if it is not.

Release Edit Lock



Element: nil

Description

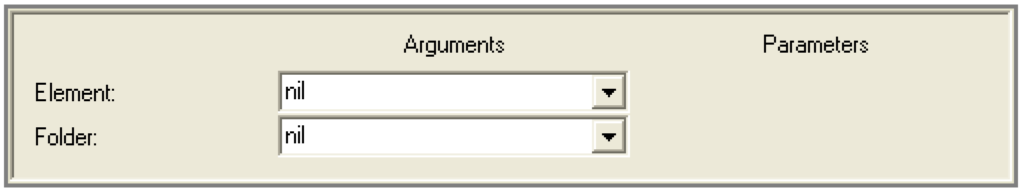
If the user executing the script has an edit lock for the specified *element*, removes the edit lock so that another user can lock and edit the element. If another user currently has an edit lock on the element, an error will be signaled.

Note that this construct is ignored in CORE Workstation. CORE Workstation does not provide concurrent access to elements and therefore does not require edit locks.

Element

Variable containing the CORE element for which the edit lock should be released.

Remove Element From Folder



	Arguments
Element:	nil
Folder:	nil

Description

Removes the specified *element* from the *folder*. If no other folder contains the specified *element*, it will automatically be added to the root folder for the database class.

If the user does not have write permission for the element, an error will be signaled.

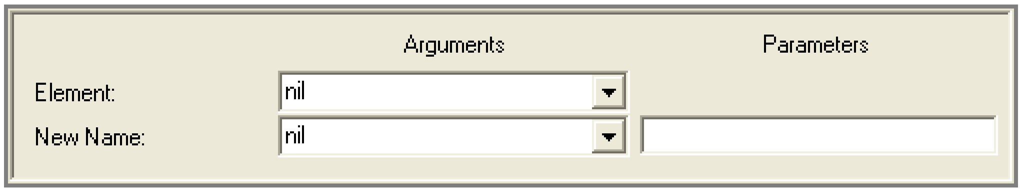
Element

Variable containing the element to remove from the *folder*.

Folder

Variable containing the folder from which the *element* should be removed.

Rename Element



	Arguments	Parameters
Element:	nil	
New Name:	nil	

Description

Changes the name of *element* from its current value to *new name*. An error is signaled if the user does not have permission to change the element name or if the new element name conflicts with an existing name (element names must be unique within a database class).

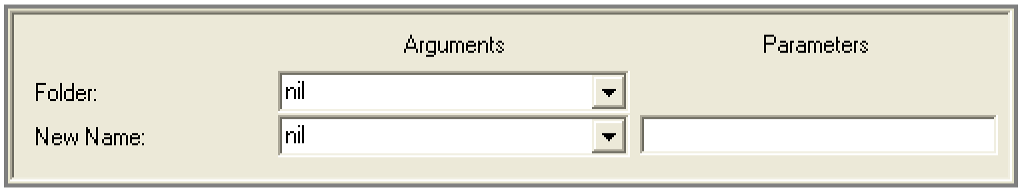
Element

Variable containing the element to be renamed.

New Name

Symbol specifying the new name for the *element*. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Rename Folder



	Arguments	Parameters
Folder:	nil	
New Name:	nil	

Description

Changes the name of *folder* from its current value to *new name*. An error is signaled if the user does not have permission to change the folder name or if the new folder name conflicts with an existing name (folder names must be unique within a parent folder just as directory names are unique in Microsoft Windows).

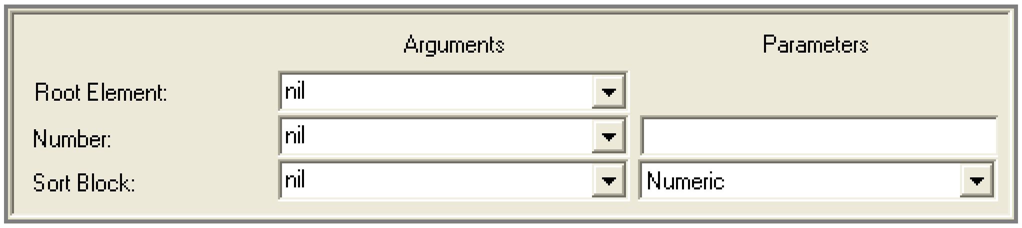
Folder

Variable containing the folder to be renamed.

New Name

Symbol specifying the new name for the *folder*. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Renumber Elements



	Arguments	Parameters
Root Element:	nil	
Number:	nil	
Sort Block:	nil	Numeric

Description

Renumbers *root element* and all of its descendents. The number attribute for *root element* is set to *number* and new hierarchical numbers are generated for the entire descendent tree. This is equivalent to the Renumber Element command in CORE.

Root Element

Variable containing the starting element for the renumber command.

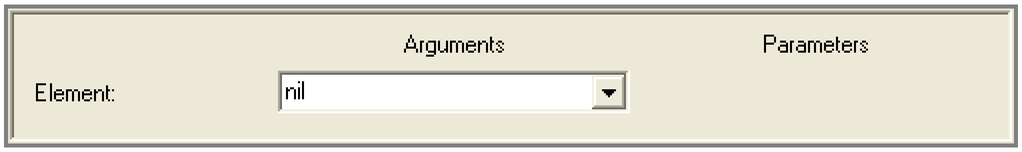
Number

Hierarchical number specifying the new number for the *root element* and starting value for renumbering all descendents. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Sort Block

Sort block definition of a sort block defined in the CORE project. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right. For elements without structures such as FFBDs, this sort block governs the order in which descendents are renumbered.

Set Edit Lock



Element: nil

Description

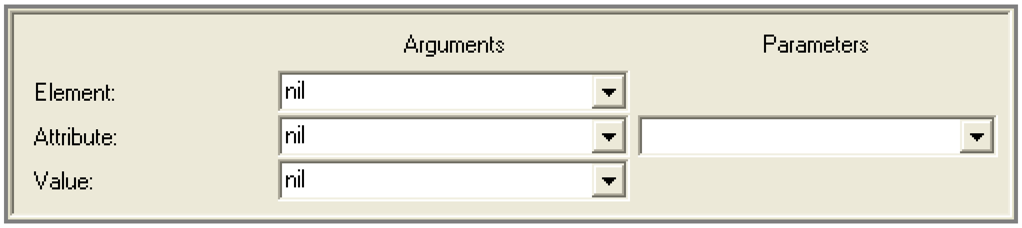
Attempts to acquire an edit lock for the specified *element*. If the user's account does not have write permission for the element or if another user currently has an edit lock on the *element*, an error will be signaled.

Note that this construct is ignored in CORE Workstation. CORE Workstation does not provide concurrent access to elements and therefore does not require edit locks.

Element

Variable containing the CORE element to edit lock.

Set Element Attribute



	Arguments	Parameters
Element:	nil	
Attribute:	nil	
Value:	nil	

Description

Sets the value of the specified *attribute* of the specified *element* to *value*. An error is signaled if the user does not have permission to change the attribute or if a type mismatch occurs (e.g., attempting to assign a string value to a numeric attribute).

Element

Variable containing the element from the CORE database to access.

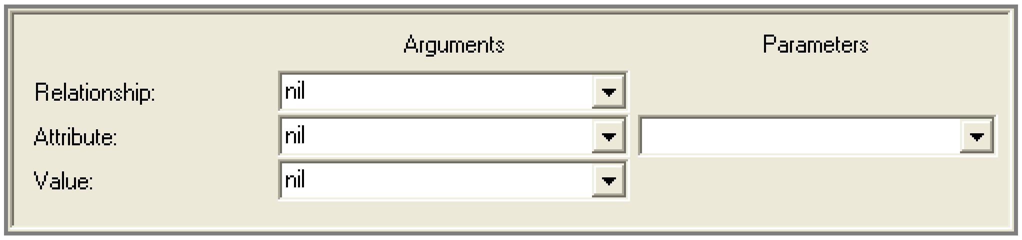
Attribute

Symbol naming the attribute to be set. The attribute should be defined in the CORE schema for the class of the *element*. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Value

Variable containing the value to be assigned to the specified *attribute*.

Set Relationship Attribute



	Arguments	Parameters
Relationship:	nil	
Attribute:	nil	
Value:	nil	

Description

Sets the value of the specified *attribute* of the specified *relationship* to *value*. An error is signaled if the user does not have permission to change the attribute or if a type mismatch occurs (e.g., attempting to assign a string value to a numeric attribute).

Relationship

Variable containing the relationship from the CORE database to access.

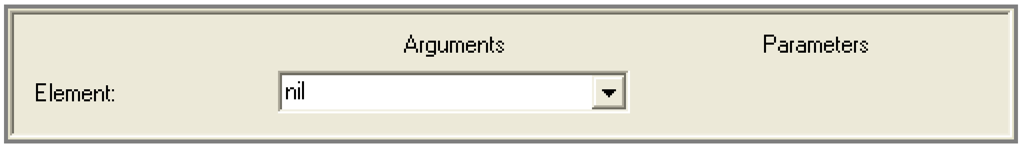
Attribute

Symbol naming the attribute to be set. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Value

Variable containing the value to be assigned to the specified *attribute*.

Terminate Edit Lock



	Arguments	Parameters
Element:	nil	

Description

Terminates any edit lock for the specified *element*. This command requires that the user's account have administrator permission for the specified *element*. If the account does not have the administrator permission, an error will be signaled.

Note that this construct is ignored in CORE Workstation. CORE Workstation does not provide concurrent access to elements and therefore does not require edit locks.

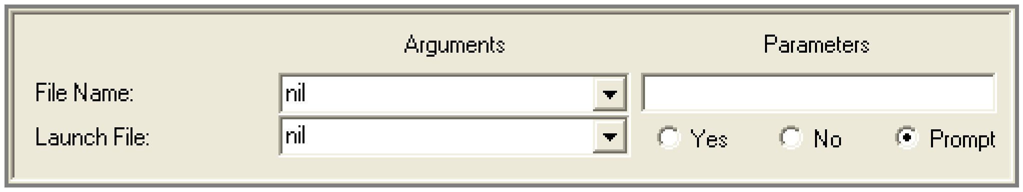
Element

Variable containing the CORE element for which the edit lock should be terminated.

THIS PAGE INTENTIONALLY BLANK

File Management Constructs

Close File



	Arguments	Parameters
File Name:	nil	
Launch File:	nil	Yes No [✓ Prompt]

Description

Closes the specified file. If an Open Output File construct is used or if a Set Default Output File construct is used in conjunction with one of the data output constructs, then a corresponding Close File construct should subsequently be included. If it is missing, the file is closed automatically, but an error message is generated.

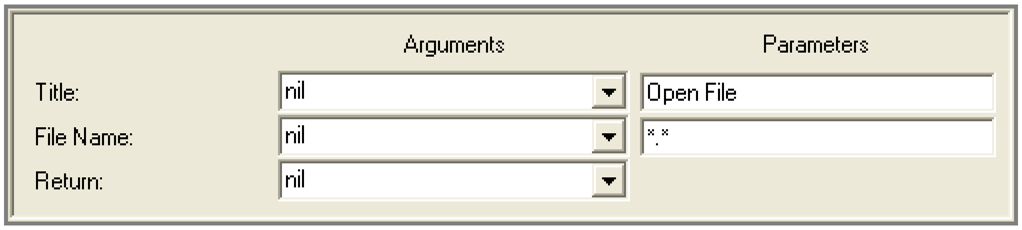
File Name

String naming the file to close. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Launch File

Boolean designating whether or not to open the file in the associated application. Can also be nil in which case, the user will be prompted to launch the file or not. Can be specified either by a variable from the drop-down on the left or by selecting the desired option on the right.

Open File Dialog



	Arguments	Parameters
Title:	nil	Open File
File Name:	nil	*.*
Return:	nil	

Description

Opens the common file dialog provided by the host operating system, allowing the user to navigate all accessible storage volumes and directory structures and to select a file to be opened for reading.

Title

Descriptive name appearing in the title bar of the common dialog. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right. The type of this parameter should be a string.

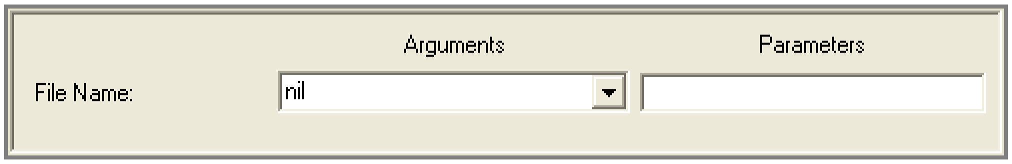
File Name

String specifying the default file name or filter, such as *.csv, to appear in the selection field of the common dialog. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right. The common dialog will automatically open on the Source directory specified in the File Locations portion of User Preferences. This can be overridden by specifying a variable that contains a full path name.

Return

Variable to hold the result of this construct, which is a string specifying the full path to the selected file, or *nil* if the user selects the Cancel button.

Open Input File



File Name: nil

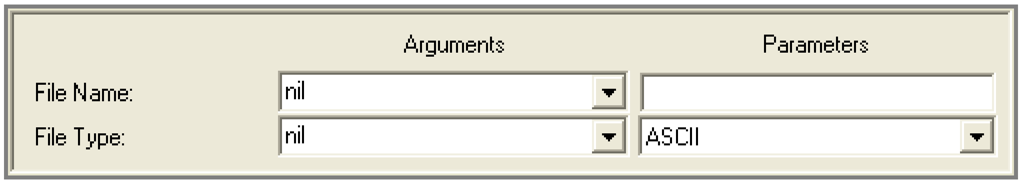
Description

Opens the specified file for reading. It does no harm to open a file multiple times for input, but an error results if the file is already open for output. A file must be opened via this construct before it can be referenced in parsing constructs.

File Name

String naming the file to be opened. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right. If the full path is not specified, it is assumed to be relative to the working directory in which CORE was launched.

Open Output File



	Arguments	Parameters
File Name:	nil	
File Type:	nil	ASCII

Description

Opens the file specified by the *file name*, which can then be used as the destination of subsequent output constructs and should be closed afterward.

File Name

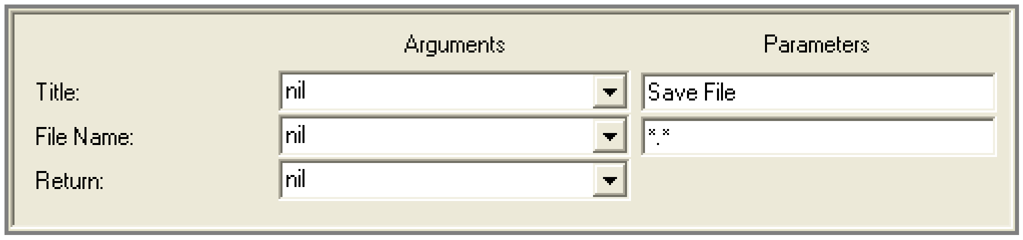
String naming the file to open. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

File Type

Symbol specifying the desired file type. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

If RTF (Rich Text Format) is specified, all output will be in RTF format, which can include multiple font styles as well as embedded graphics and can be imported into word processors. If HTML is specified, appropriate text translations will be made automatically to conform to HTML requirements. You can also generate plain ASCII files to output the results of simple queries, to generate input for other parsers or tools, etc.

Save File Dialog



	Arguments	Parameters
Title:	nil	Save File
File Name:	nil	*.*
Return:	nil	

Description

Displays the standard Windows Save File dialog, prompting the user to select a file to open (for writing), and assigns the result to the *return* variable.

Title

String to use as the dialog window title. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

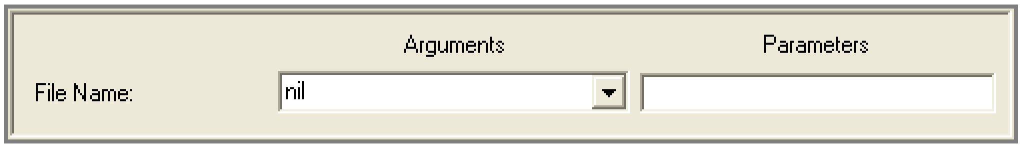
File Name

String specifying the default file name or filter, such as *.rtf, to appear in the selection field of the standard dialog. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right. The standard dialog will automatically open on the Output directory specified in the File Locations portion of User Preferences. This can be overridden by specifying a variable that contains a full path name.

Return

Variable to hold the result of this construct, which is a string specifying the full path for the selected file, or nil if the user cancels the dialog.

Set Default Input File



File Name: nil

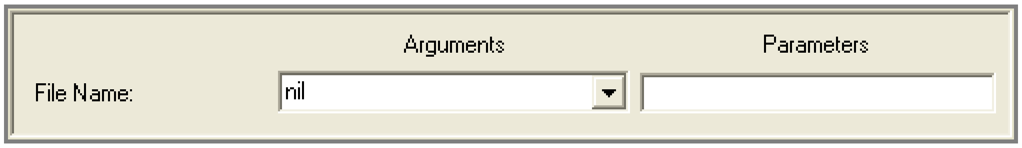
Description

Sets the default file used by parsing constructs when their *file* parameters are not specified and multiple files are open for reading. Before an input file can be set as the default, it must first be opened.

File Name

String naming the input file to be used as the default. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right. The value of this parameter should be identical to the file name with which the file was opened.

Set Default Output File



Description

Sets the default file used by output constructs when their *file* parameters are not specified and multiple files are open for writing. This file must still be opened before it can be used.

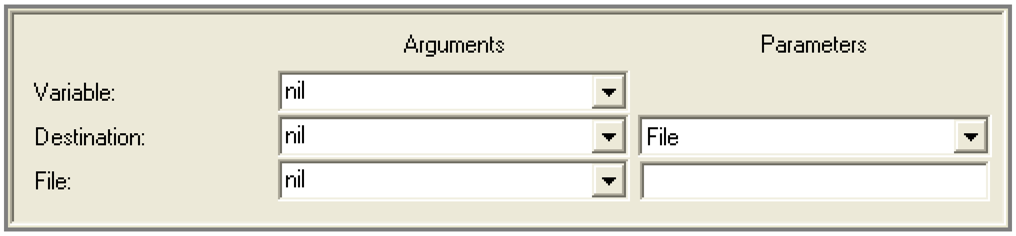
File Name

String naming the default output file. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

THIS PAGE INTENTIONALLY BLANK

Output Constructs

Comma-Delimited Output



	Arguments	Parameters
Variable:	nil	
Destination:	nil	File
File:	nil	

Description

Outputs the contents of a *variable* formatted as a comma-delimited field. If the variable being output could contain a comma or other special character, the field is delimited by quotation marks and any embedded quotation marks are doubled (dates, time stamps, strings, and text blocks are output in this manner). Otherwise, the variable is output in standard format.

Variable

Variable whose contents is to be output in comma-delimited format.

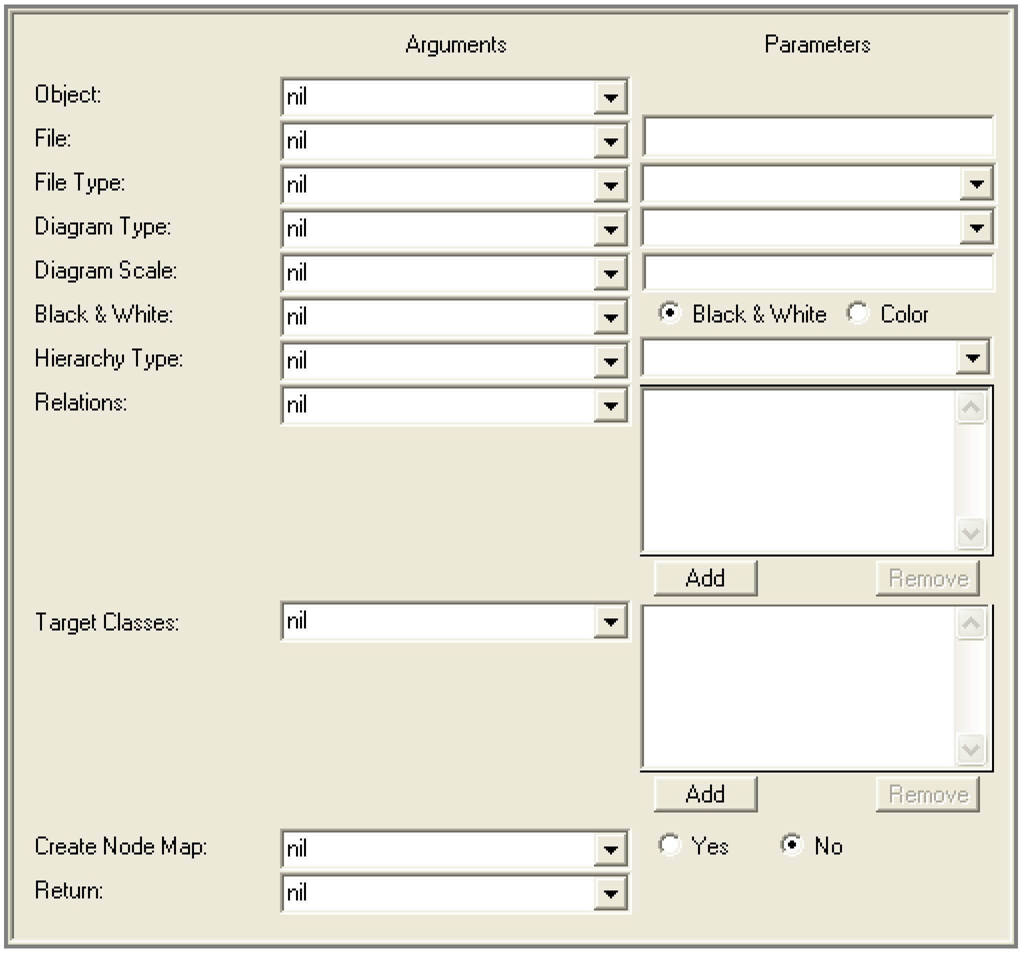
Destination

Symbol specifying whether the output should be directed to a file, the on-screen transcript, both a file and the on-screen transcript, or a CORE2net stream.

File

String specifying the name of the destination file to which the variable is to be output.

Diagram File Output



	Arguments	Parameters
Object:	nil	
File:	nil	
File Type:	nil	
Diagram Type:	nil	
Diagram Scale:	nil	
Black & White:	nil	[✓ Black & White] Color
Hierarchy Type:	nil	
Relations:	nil	
Target Classes:	nil	
Create Node Map:	nil	Yes [✓ No]
Return:	nil	

Description

Outputs any CORE database or schema diagram as a graphics file. A diagram can be output in either WMF, JPG, or PNG format.

Optionally, for JPG and PNG formats, assigns to the *return* variable an ordered collection of diagram entity locators each of which corresponds to a hotspot on the output diagram.

Object

Variable containing an object of one of the following data types:

Class
Element
Facility

File

String naming the *destination* of file output. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

File Type

Symbol specifying the desired file type. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Diagram Type

Symbol specifying the diagram type to output. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

If the *object* is a class, can be “Class ER Diagram”. If the *object* is an element, can be “Enhanced FFBD”, “ER Diagram”, “FFBD”, “Hierarchy Diagram”, “IDEF0 A-0 Context Diagram”, “IDEF0 Diagram”, “N2 Diagram, or “Physical Block Diagram”. If the *object* is a facility, can be “Attribute Matrix”, “Attributed-Relation Matrix”, or “Relation Matrix”.

Diagram Scale

Scale (as a percentage of the original diagram size) at which the diagram is to be output.

Black & White

Boolean indicating whether the diagram should be forced to black and white (as opposed to color). Can be specified either by a variable from the drop-down on the left or a manual selection from the buttons on the right.

Hierarchy Type

Symbol specifying the type of hierarchy diagram to be produced. Only meaningful if *diagram type* is set to “Hierarchy Diagram” and *object* is an Element. Can be any hierarchy type defined in the CORE project. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Relations

Collection of symbols naming relations to traverse in computing the closure of the specified element *object*. Used if the specified diagram is a custom hierarchy (i.e., *diagram type* is set to “Hierarchy Diagram” and *hierarchy type* is empty). Can be specified either by a variable from the drop-down on the left or manual additions into the collection on the right. The best kind of collection to use is a set.

Target Classes

Variable containing a collection of filter classes. Used if the specified diagram is a custom hierarchy (i.e., *diagram type* is set to “Hierarchy Diagram” and *hierarchy type* is empty). Can be specified either by a variable from the drop-down on the left or manual additions into the collection on the right. Can be of any of the following data types:

Array
Bag
Dictionary
Ordered Collection
Set
Sorted Collection.

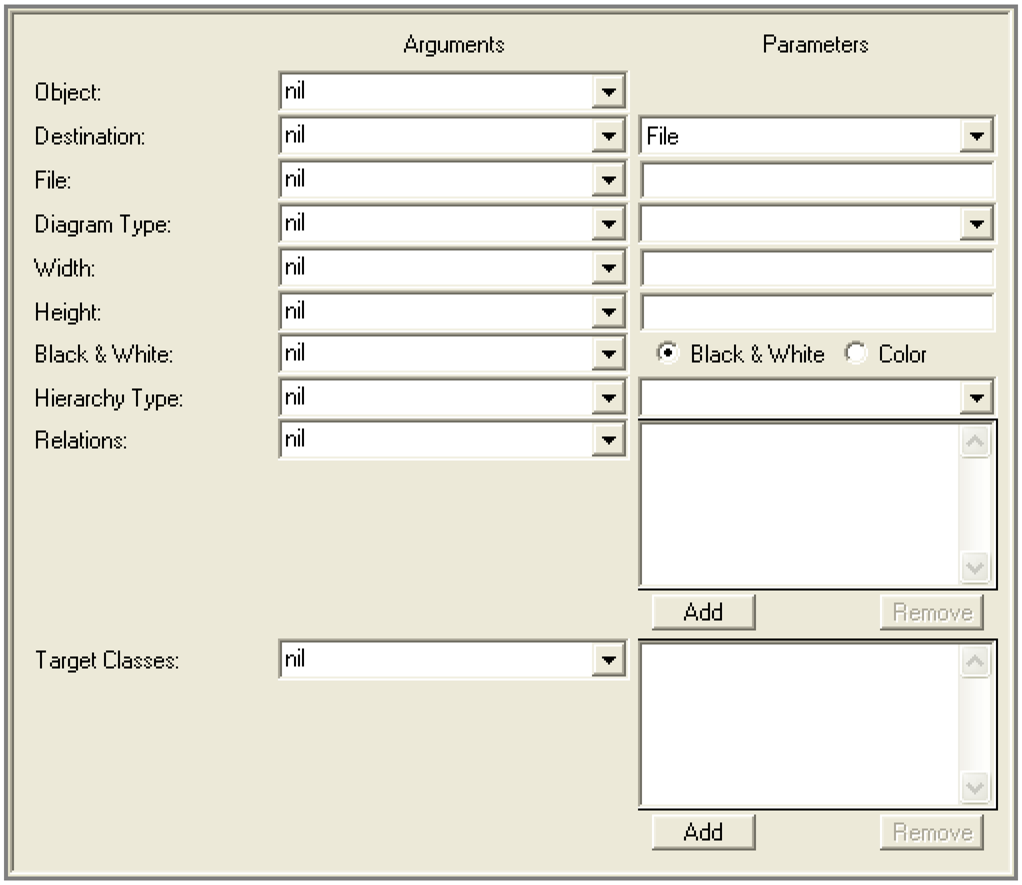
Create Node Map

Boolean indicating whether a node map for the diagram should be generated. Can be specified either by a variable from the drop-down on the left or a manual selection from the buttons on the right.

Return

Variable to hold the result of this construct. It is either an ordered collection of diagram entity locators or *nil* depending on the setting of *create node map*.

Diagram Output



	Arguments	Parameters
Object:	nil	
Destination:	nil	File
File:	nil	
Diagram Type:	nil	
Width:	nil	
Height:	nil	
Black & White:	nil	[✓ Black & White] Color
Hierarchy Type:	nil	
Relations:	nil	
Target Classes:	nil	

Description

Encodes any CORE database or schema diagram as a graphic in an RTF file.

The diagram is reduced to the scale at which its height is less than or equal to the specified maximum *height* and its width is less than or equal to the specified maximum *width* (e.g., to ensure that it fits on a page). Whichever scaling factor is more restrictive is used to reduce both dimensions proportionally. This construct is not useful if the transcript is selected as the *destination* or if file output is chosen and the specified *file* was opened in ASCII mode.

Object

Variable containing an object of one of the following data types:

Class
Element
Facility.

Destination

Symbol specifying that output be directed either to a *file*, to an on-screen text transcript, to both a file and the on-screen transcript, or to a CORE2net stream. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

File

String naming the *destination* of file output. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Diagram Type

Symbol specifying the diagram type to output. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

If the *object* is a class, can be “Class ER Diagram”. If the *object* is an element, can be “Enhanced FFBD”, “ER Diagram”, “FFBD”, “Hierarchy Diagram”, “IDEF0 A-0 Context Diagram”, “IDEF0 Diagram”, “N2 Diagram, or “Physical Block Diagram”. If the *object* is a facility, can be “Attribute Matrix”, “Attributed-Relation Matrix”, or “Relation Matrix”.

Width

Float specifying the maximum width of the diagram (used in conjunction with the *height* to determine diagram scale). Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Height

Float specifying the maximum width of the diagram (used in conjunction with the *width* to determine diagram scale). Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Black & White

Boolean indicating whether the diagram should be forced to black and white (as opposed to color). Can be specified either by a variable from the drop-down on the left or a manual selection from the buttons on the right.

Hierarchy Type

Symbol specifying the type of hierarchy diagram to be produced. Only meaningful if *diagram type* is set to “Hierarchy Diagram” and *object* is an element. Can be any hierarchy type defined in the CORE project. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Relations

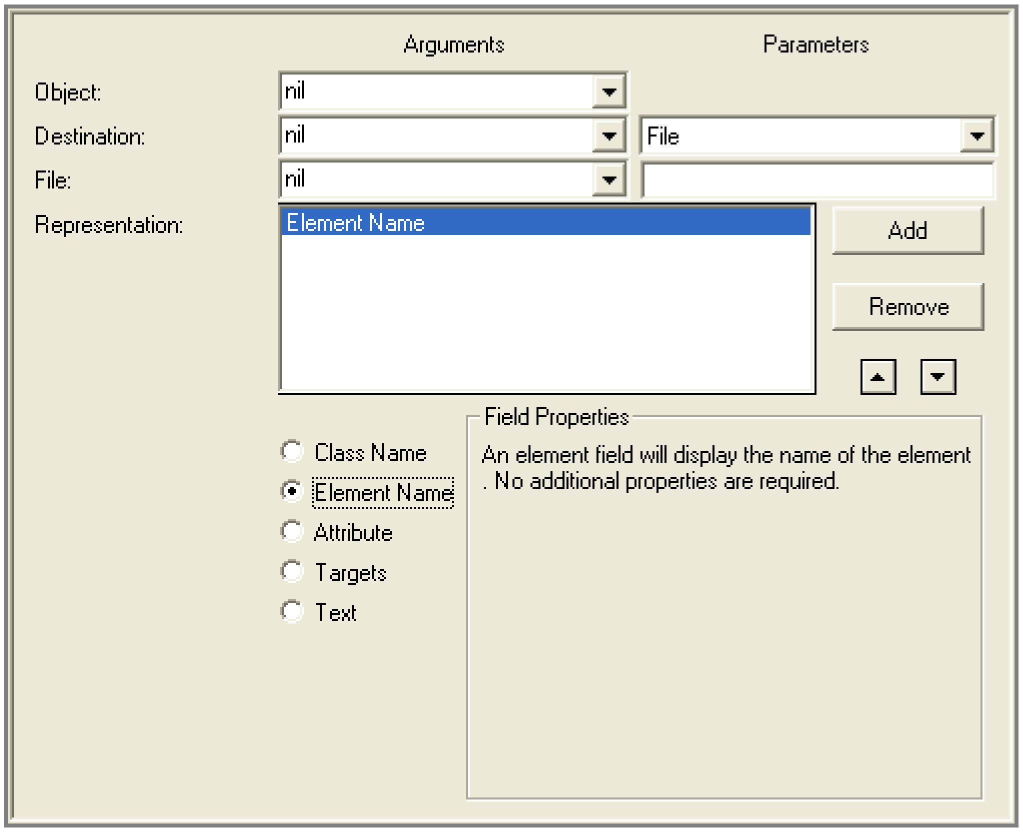
Collection of symbols naming relations to traverse in computing the closure of the specified element *object*. Used if the specified diagram is a custom hierarchy (i.e., *diagram type* is set to “Hierarchy Diagram” and *hierarchy type* is empty.) Can be specified either by a variable from the drop-down on the left or manual additions into the collection on the right. The best kind of collection to use is a set.

Target Classes

Variable containing a collection of filter classes. Used if the specified diagram is a custom hierarchy (i.e., *diagram type* is set to “Hierarchy Diagram” and *hierarchy type* is empty). Can be specified either by a variable from the drop-down on the left or manual additions into the collection on the right. Can be of any of the following data types:

Array
Bag
Dictionary
Ordered Collection
Set
Sorted Collection.

Element Output



Arguments | Parameters

Object: nil

Destination: nil | File

File: nil |

Representation: Element Name

( ) Class Name
(●) Element Name
( ) Attribute
( ) Targets
( ) Text

Field Properties

An element field will display the name of the element . No additional properties are required.

Description

Represents an element *object* from the CORE database in the report output. The amount and format of the information included from the element is determined by its *representation*.

Object

Variable containing an element to be referenced in or from which information is to be extracted for report output.

Destination

Symbol specifying that output be directed to either a *file*, an on-screen text transcript, both a file and the on-screen transcript, or a CORE2net stream. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

File

String naming the *destination* of file output. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Representation

Ordered collection of representation fields, each of which can be the alias of the element's class, the name of the element itself, the value of an attribute defined for the class, the targets of a relation, or hard-coded text.

Class Name

Outputs the alias of the element's class, if defined, or the class name if the alias is undefined.

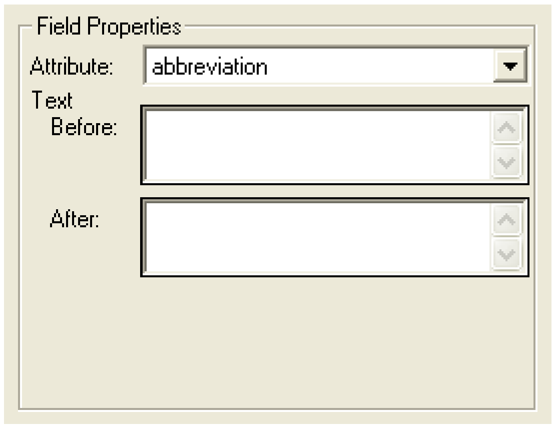
Element Name

Outputs the name of the element.

Attribute

Outputs the attribute value for the specified *attribute* with boilerplate text before and after the value. For each **Attribute** field, the desired *attribute* must be selected from the drop-down list.

If the attribute value is *nil*, nothing is output. Otherwise, the *before* text is output first, followed by the attribute value, and finally the *after* text is output. If boilerplate text is desired regardless of whether or not a value exists, the boilerplate text should be specified in a **Text** representation field instead of an **Attribute** representation field.

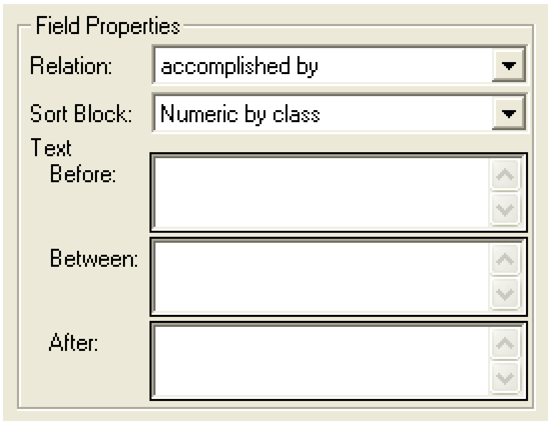


Attribute: abbreviation
Text Before:

Targets

Outputs the targets for the specified *relation* with boilerplate text before and after the target list as well as text to separate individual targets in the list. For each **Targets** field, the desired *relation* and *sort block* must be selected from the drop-down lists. The *sort block* specifies not only in what order to output the targets but also what form the representation should take.

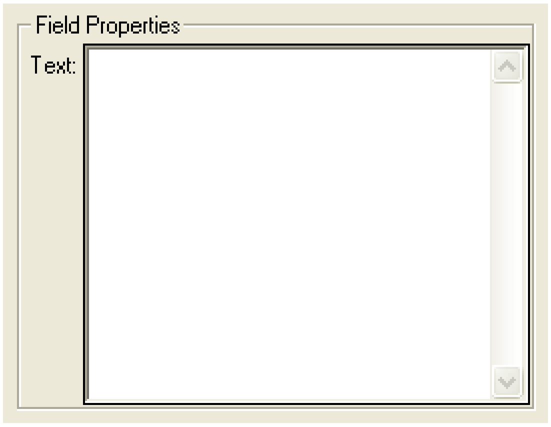
If no targets exist for the specified relation, nothing is output. Otherwise, the *before* text is output first, followed by the list of targets separated by the *between* text, and finally the *after* text is output. If boilerplate text is desired regardless of whether or not targets exist, the boilerplate text should be specified in a **Text** representation field instead of a **Targets** representation field.



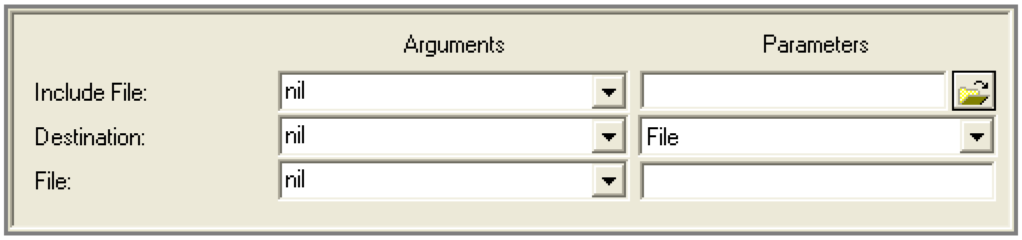
Relation: accomplished by
Sort Block: Numeric by class

Text

Outputs the boilerplate text specified in the *text* field.



Include External Boilerplate



	Arguments	Parameters
Include File:	nil	
Destination:	nil	File
File:	nil	

Description

Imports an external *file* into a report. This is only meaningful if the specified *destination* is a file and the file was opened in RTF mode. Any file type is supported that can be read and understood by the word processor into which the RTF file will ultimately be imported.

Include File

String naming the file to be imported. Can be specified either by a variable from the drop-down on the left or by browsing using the button on the right. The file browse button opens a standard Windows file dialog prompting the user to select a file for importation.

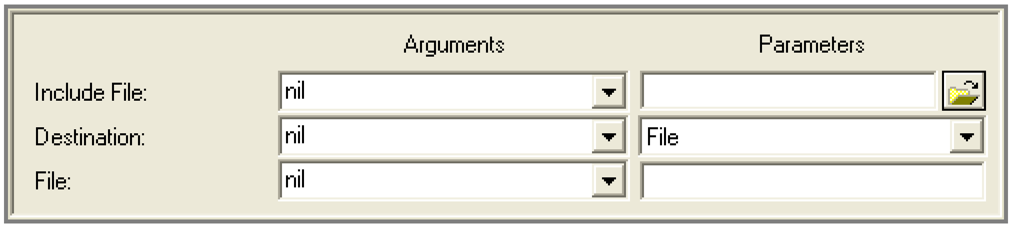
Destination

Symbol specifying that output be directed to either a *file*, an on-screen text transcript, both a file and the on-screen transcript, or a CORE2net stream. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

File

String naming the *destination* of file output. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Include External Graphic



	Arguments	Parameters
Include File:	nil	
Destination:	nil	File
File:	nil	

Description

Imports an external graphics *file* into a report. This is only meaningful if the specified *destination* is a file and the file was opened in RTF mode. Any picture type is supported that can be read and understood by the word processor into which the RTF file will ultimately be imported.

Include File

String naming the graphics file to be imported. Can be specified either by a variable from the drop-down on the left or by browsing using the button on the right. The file browse button opens a standard Windows file dialog prompting the user to select a file for importation.

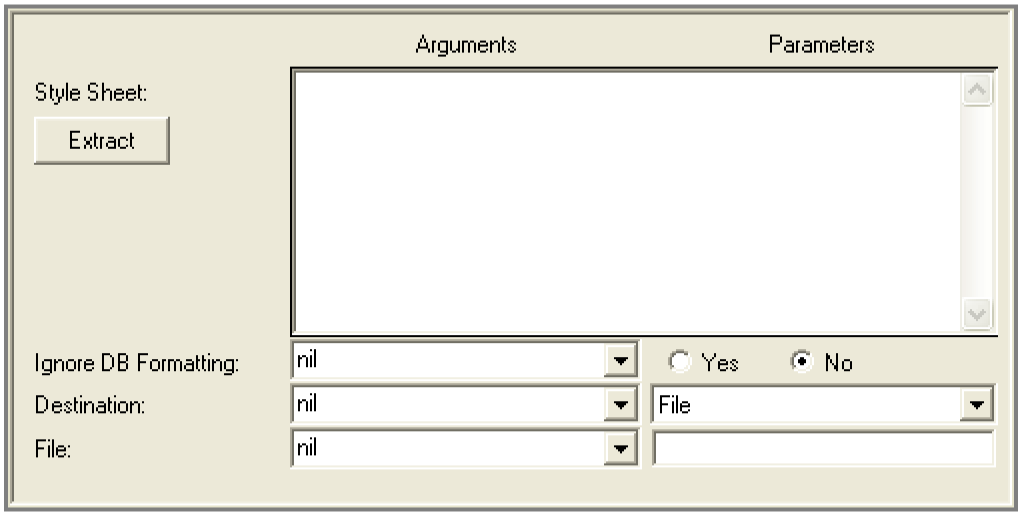
Destination

A symbol specifying that output be directed to either a *file*, an on-screen text transcript, both a file and the on-screen transcript, or a CORE2net stream. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

File

String naming the *destination* of file output. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Include Style Sheet



Style Sheet:

Ignore DB Formatting: nil   Yes [✓ No]

Destination: nil   File

File: nil

Description

Places a style sheet in a report *file*, on the report transcript, or both. This is only meaningful if the *destination* is a file, and the *file* is opened in RTF mode. Only one style sheet can be included in a script.

Style Sheet

A block of text containing style commands in the RTF language to parse and output. The easiest way to create this is to define your desired styles in a word processor, save the document in RTF format, and use the Extract button to have CORE copy the style header from the RTF file into this field. Word processors can also pull styles from multiple documents into a single document.

Ignore DB Formatting

Boolean indicating whether any styles associated with element text attributes output by the script should be removed upon output. Can be specified either by a variable from the drop-down on the left or manual selection from the buttons on the right.

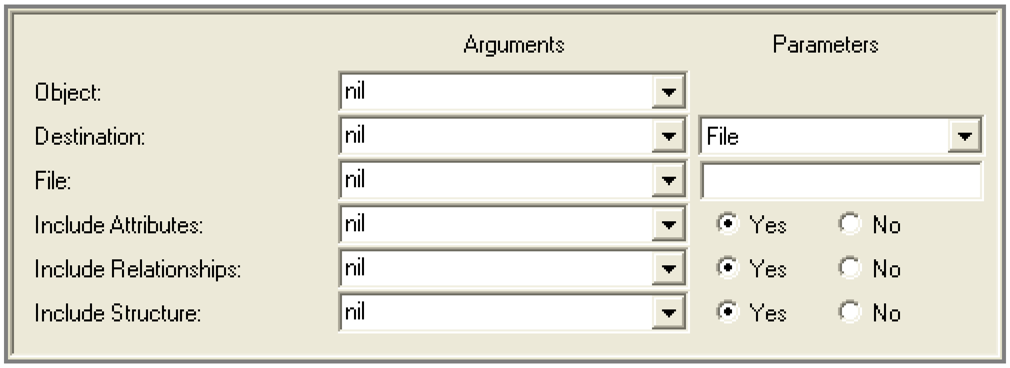
Destination

Symbol specifying that output be directed to either a *file*, an on-screen text *transcript*, both a file and the on-screen transcript, or a CORE2net stream. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

File

String naming the *destination* of file output. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

RDD Output



	Arguments	Parameters
Object:	nil	
Destination:	nil	File
File:	nil	
Include Attributes:	nil	[✓ Yes] No
Include Relationships:	nil	[✓ Yes] No
Include Structure:	nil	[✓ Yes] No

Description

Outputs information from an element *object* in the CORE database in a format that can be imported into RDD-100. The amount of information included from the element is determined by the include toggles. Element information is translated to account for schema terminology differences based on the .cbt file found in the CORE>bridges folder.

Object

Variable containing an element from which information is to be extracted for output.

Destination

A symbol specifying that output be directed to either a *file*, an on-screen text transcript, both a file and the on-screen transcript, or a CORE2net stream. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

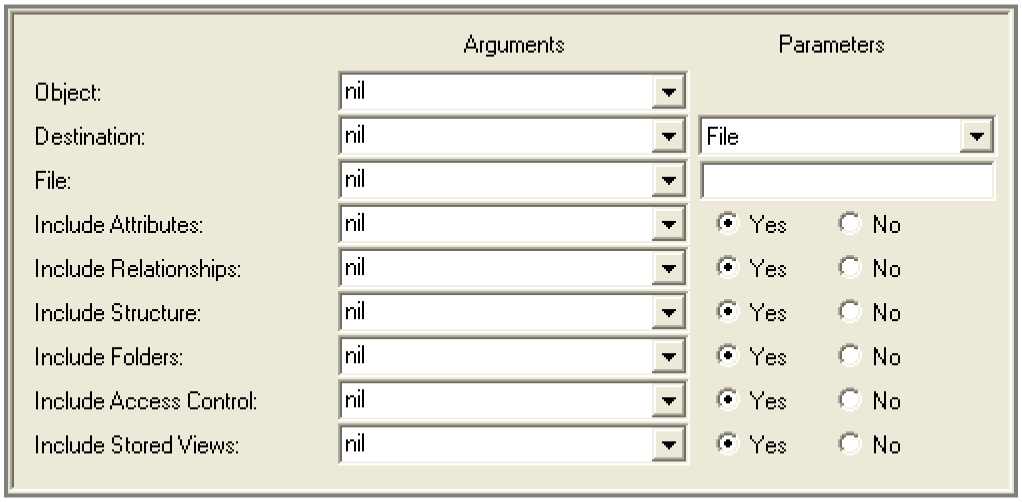
File

String naming the *destination* of file output. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Include Toggles

A series of toggles indicating whether or not the element attributes, relationships, and structure is to be included in the output. Can be specified either by a variable from the drop-down on the left or a manual selection from the Yes or No buttons on the right. If a variable is used, it must contain a Boolean value. The value of *true* corresponds to the button choice of Yes. The value of *false* corresponds to the button choice of No.

RDT Output



	Arguments	Parameters	
Object:	nil		
Destination:	nil	File	
File:	nil		
Include Attributes:	nil	[✓ Yes]	No
Include Relationships:	nil	[✓ Yes]	No
Include Structure:	nil	[✓ Yes]	No
Include Folders:	nil	[✓ Yes]	No
Include Access Control:	nil	[✓ Yes]	No
Include Stored Views:	nil	[✓ Yes]	No

Description

Outputs information from an element *object* in the CORE database in .rdt format that can be imported into CORE. The amount of information included from the element is determined by the include toggles.

Object

Variable containing an element from which information is to be extracted for output.

Destination

Symbol specifying that output be directed to either a *file*, an on-screen text transcript, both a file and the on-screen transcript, or a CORE2net stream. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

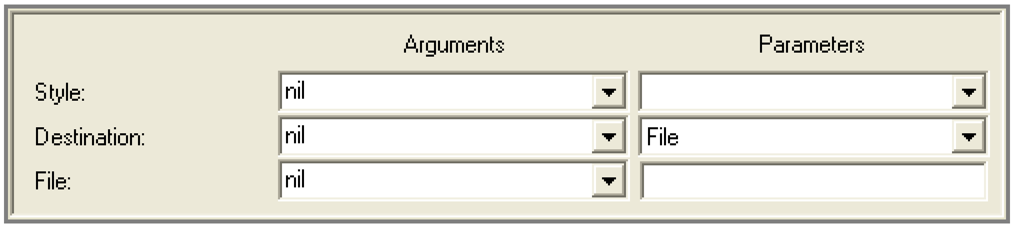
File

String naming the *destination* of file output. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Include Toggles

A series of toggles indicating whether or not the element attributes, relationships, structure, class folders, access control information, and stored view descriptions are to be included in the output. Can be specified either by a variable from the drop-down on the left or a manual selection from the Yes or No buttons on the right. If a variable is used, it must contain a Boolean value. The value of *true* corresponds to the button choice of Yes. The value of *false* corresponds to the button choice of No.

Set Style



	Arguments	Parameters
Style:	nil	
Destination:	nil	File
File:	nil	

Description

Sets the formatting *style* for subsequent output to a *file*. This is only meaningful if the file is opened in RTF mode. The *style* must previously have been included in the report.

Style

String naming a style. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

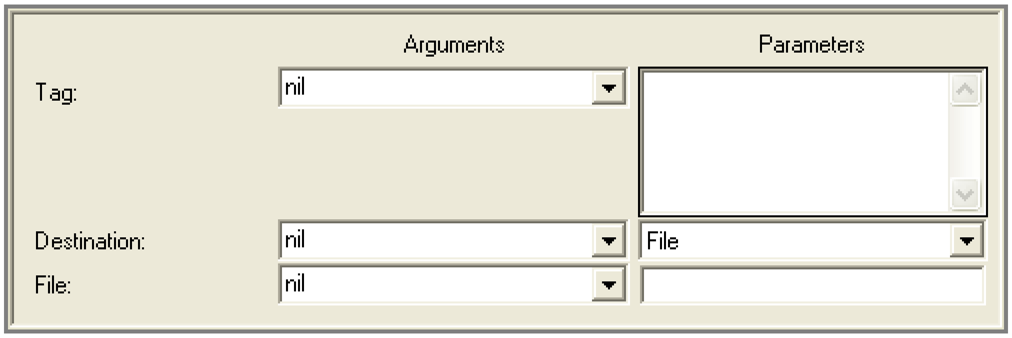
Destination

A symbol specifying that output be directed to either a *file*, an on-screen text transcript, both a file and the on-screen transcript, or a CORE2net stream. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

File

String naming the *destination* of file output. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Tag Output



	Arguments	Parameters
Tag:	nil	
Destination:	nil	File
File:	nil	

Description

Places the *tag* text in the script output. Like the Text Output construct except that no translation is performed on the *tag*. This allows the direct output of RTF and HTML commands.

Tag

A text string. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

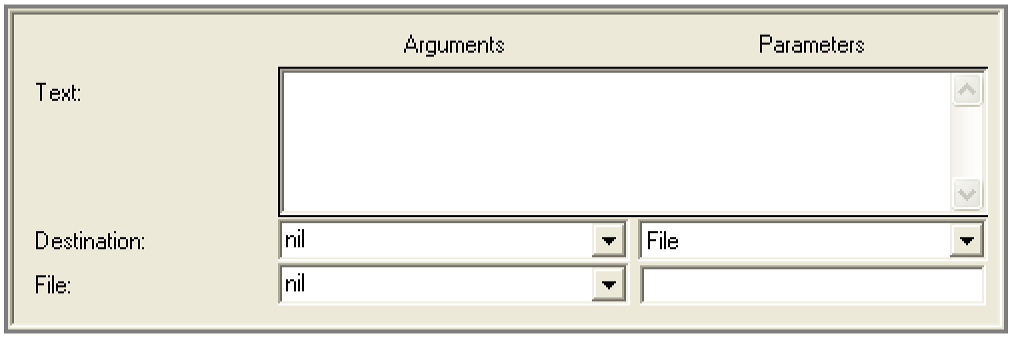
Destination

A symbol specifying that output be directed to either a *file*, an on-screen text transcript, both a file and the on-screen transcript, or a CORE2net stream. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

File

String naming the *destination* of file output. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Text Output



	Arguments	Parameters
Text:		
Destination:	nil	File
File:	nil	

Description

Places fixed *text* in the output.

Text

A text string to output. Both carriage return-line feeds and tab spacing are allowed in this field.

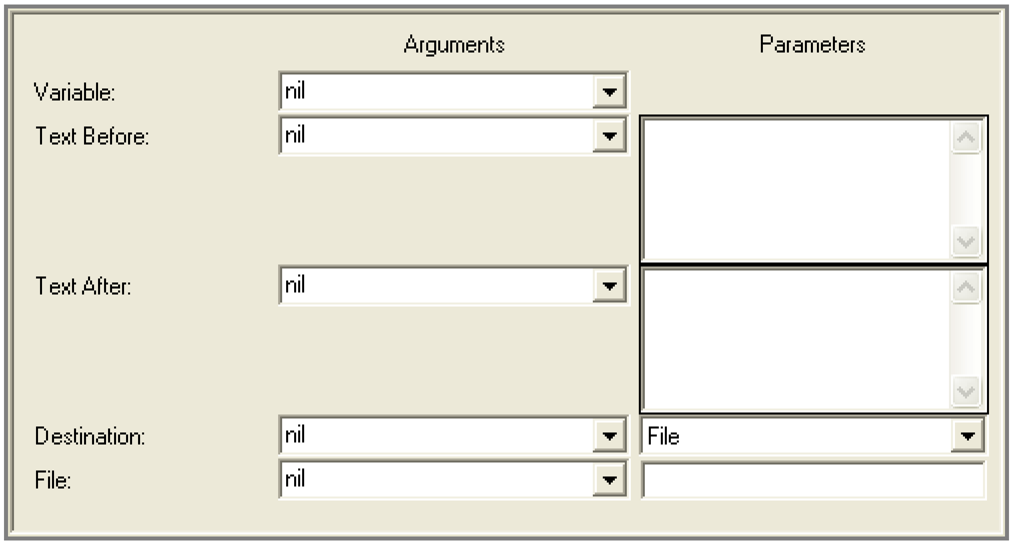
Destination

A symbol specifying that output be directed to either a *file*, an on-screen text transcript, both a file and the on-screen transcript, or a CORE2net stream. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

File

String naming the *destination* of file output. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Variable Output



	Arguments	Parameters
Variable:	nil	
Text Before:	nil	
Text After:	nil	
Destination:	nil	File
File:	nil	

Description

Places a string representation of the object contained in the specified *variable* in a report output with boilerplate text before and after the string.

If the variable is *nil*, nothing is output. Otherwise the *text before* is output first, followed by the *variable* value, and finally the *text after* is output. If boilerplate text is desired regardless of whether or not a value exists, the boilerplate text should be specified in a Text Output Construct rather than in a Variable Output Construct.

Variable

Variable whose value is to be output.

Text Before

Text to be output before outputting the *variable* value if it exists.

Text After

Text to be output after outputting the *variable* value if it exists.

Destination

A symbol specifying that output be directed to either a *file*, an on-screen text transcript, both a file and the on-screen transcript, or a CORE2net stream. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

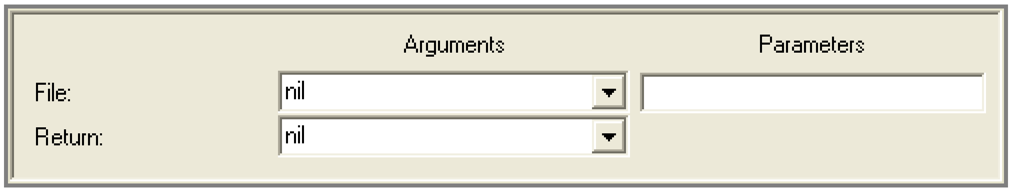
File

String naming the *destination* of file output. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

THIS PAGE INTENTIONALLY BLANK

Parsing Constructs

At End



	Arguments	Parameters
File:	nil	
Return:	nil	

Description

Assigns to the *return* variable a Boolean value indicating whether or not the current position pointer in the specified *file* is at the end of the *file*. An error results if the *file* has not previously been opened for input.

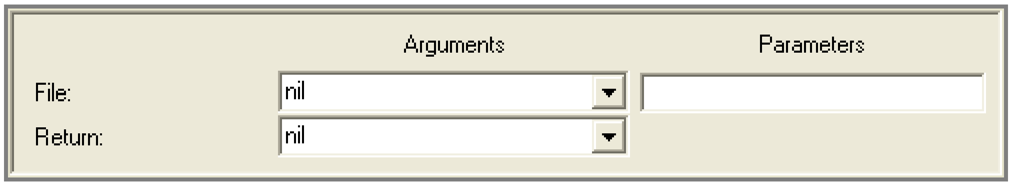
File

String naming the input file to be tested. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right. The value of this parameter should be identical to the *file name* with which the file was opened. If this parameter is not specified, the default input file is tested.

Return

Variable to hold the result of this construct.

Get Position



	Arguments	Parameters
File:	nil	
Return:	nil	

Description

Assigns to the *return* variable the integer index of the current position pointer in the input stream specified by the *file* parameter. An error results if the *file* has not previously been opened for input. This construct can be used to temporarily record a point in the *file* in order to return to it later using the Set Position construct.

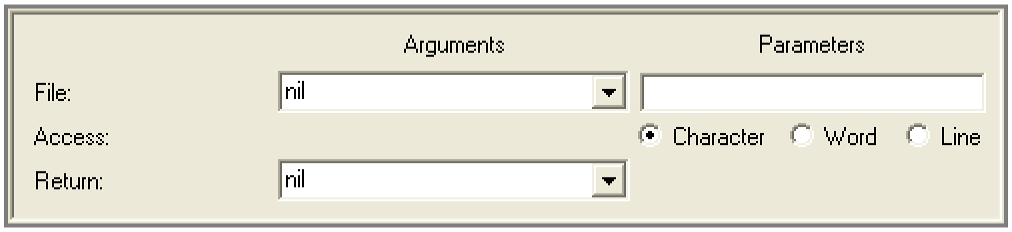
File

A string naming the input file to be queried. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right. The value of this parameter should be identical to the *file name* with which the file was opened. If this parameter is not specified, the default input file is queried.

Return

Variable to hold the result of this construct.

Next



File: nil

Access: [✓ Character] Word Line

Return: nil

Description

Parses a string from the specified *file*, beginning at the current position in the *file*, advances the current position pointer past what has been read, and assigns the string to the *return* variable. An error results if the *file* has not previously been opened for input.

File

A string naming the input file to be read. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right. The value of this parameter should be identical to the *file name* with which the file was opened. If this parameter is not specified, the default input file is read.

Access

Determines how much is read from the specified *file*. If this parameter is set to *Character*, the next byte is read and converted to a string. An error results if the current position pointer in the *file* is at the end of the *file*.

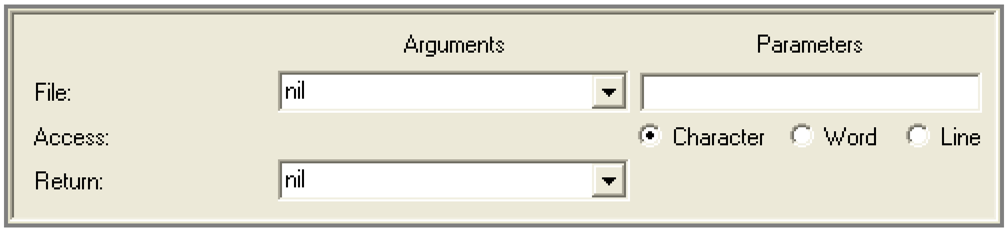
If this parameter is set to *Word* and the current position in the *file* is the end of the *file*, then this construct returns *nil*. Otherwise, it skips forward from the current position to the first alphabetic or numeric character it encounters or to the end of the file. It then reads any contiguous sequence of letters and digits beginning at that point, stopping, if necessary, at the end of the *file*. This construct may return an empty string if there are no letters or digits remaining in the *file*.

If this parameter is set to *Line*, then this construct reads all characters up to the next line feed, the next carriage return and line feed combination, or the end of the *file*. If the current position in the *file* is at the end of the *file*, then this construct returns an empty string.

Return

Variable to hold the result of this construct.

Peek



	Arguments	Parameters
File:	nil	
Access:		[✓ Character] Word Line
Return:	nil	

Description

Parses a string from the specified *file*, beginning at the current position in the *file*, and assigns the string to the *return* variable. An error results if the *file* has not previously been opened for input. This construct differs from the Next construct in that the current position pointer in the *file* is **not** advanced, such that the next parsing construct to be invoked begins at the same point in the input stream.

File

A string naming the input file to be read. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right. The value of this parameter should be identical to the *file name* with which the file was opened. If this parameter is not specified, the default input file is read.

Access

Determines how much is read from the specified *file*. If this parameter is set to *Character*, the next byte is read and converted to a string. Unlike the *Next* construct, an error does **not** result if the current position pointer in the *file* is at the end of the *file*. Instead, this construct returns *nil*.

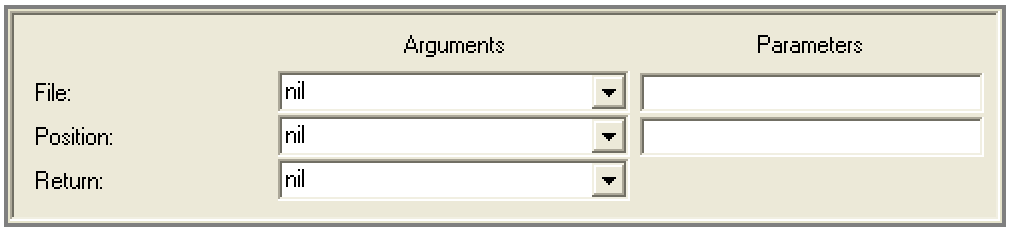
If this parameter is set to *Word* and the current position in the *file* is the end of the *file*, then this construct returns *nil*. Otherwise, it skips forward from the current position to the first alphabetic or numeric character it encounters or to the end of the file. It then reads any contiguous sequence of letters and digits beginning at that point, stopping, if necessary, at the end of the *file*. This construct may return an empty string if there are no letters or digits remaining in the *file*.

If this parameter is set to *Line*, then this construct reads all characters up to the next line feed, the next carriage return and line feed combination, or the end of the *file*. Unlike the *Next* construct, if the current position in the *file* is at the end of the *file*, then this construct returns *nil*.

Return

Variable to hold the result of this construct.

Set Position



	Arguments	Parameters
File:	nil	
Position:	nil	
Return:	nil	

Description

Moves the current position pointer in the specified *file* to the specified *position*. An error results if the *file* has not previously been opened for input.

File

A string naming the input file which position pointer is to be moved. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right. The value of this parameter should be identical to the *file name* with which the file was opened. If this parameter is not specified, the position pointer of the default input file is set.

Position

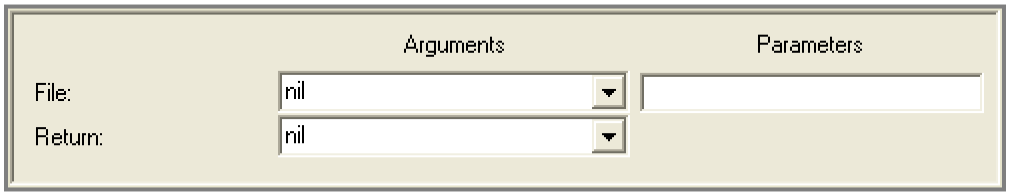
An integer specifying the index of the position in the specified *file* to which its position pointer is to be set. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Note that file positions are zero-based – that is, the beginning of a file is index 0, and the position at the index equal to the size of the file (where the At End construct returns *true*) is **after** the last byte of the file. An error results if the specified position is negative or greater than the size of the file.

Return

Variable to hold the result of this construct, which is an integer specifying the previous position pointer in the specified *file*, the position at the time this construct was invoked.

Size



	Arguments	Parameters
File:	nil	
Return:	nil	

Description

Assigns to the *return* variable an integer, which is the number of bytes in the specified *file*. An error results if the *file* has not previously been opened for input.

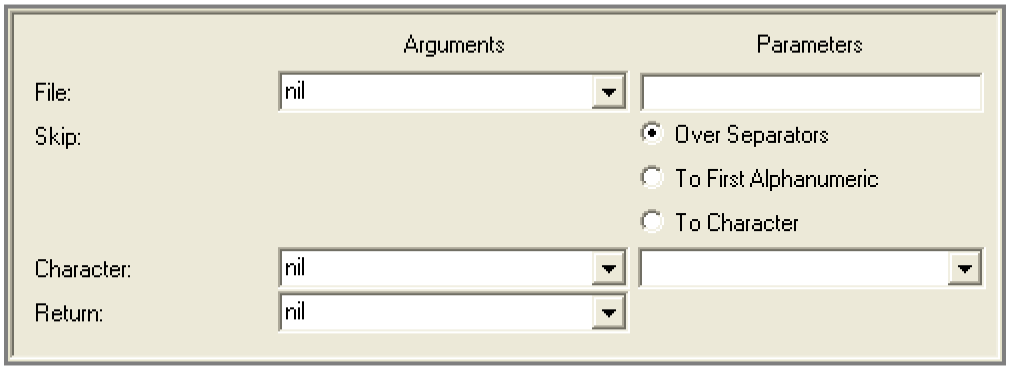
File

A string naming the input file to be queried. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right. The value of this parameter should be identical to the *file name* with which the file was opened. If this parameter is not specified, the default input file is queried.

Return

Variable to hold the result of this construct.

Skip



	Arguments	Parameters
File:	nil	
Skip:		[✓ Over Separators] To First Alphanumeric To Character
Character:	nil	
Return:	nil	

Description

Advances the current position pointer in the specified *file* to the specified place in its input stream. Note that the position might not have to change in order for the conditions implied by the *skip* parameter to be met (i.e., the input stream might already be positioned at the desired type of object). An error results if the *file* has not previously been opened for input.

File

A string naming the input file which position pointer is to be moved. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right. The value of this parameter should be identical to the *file name* with which the file was opened. If this parameter is not specified, the position pointer of the default input file is set.

Skip

Specifies how far the current position pointer advances through the specified *file* by informing the report writer what kind of object to skip over or to skip to in the input stream. If this parameter is set to *Over Separators*, then the parser skips up to the first non-white space character or up to the end of the file. White space is defined as spaces, tabs, carriage returns, line feeds, and form feeds.

If this parameter is set to *To First Alphanumeric*, then the parser skips all characters until it encounters the first letter or digit after the point in the input stream at which this construct began or until it reaches the end of the file.

If this parameter is set *To Character*, then the report author can select a variable containing a printable character or can manually type a character in the field corresponding to this choice, and, when this construct is invoked, the parser will advance to the next occurrence of the specified character in the input stream or to the end of the file.

If this parameter is set to *Special Character*, then the report author can choose a non-printable character from the drop-down list corresponding to this choice, and, when this construct is invoked, the parser will advance to the next occurrence of the specified character in the input stream or to the end of the file.

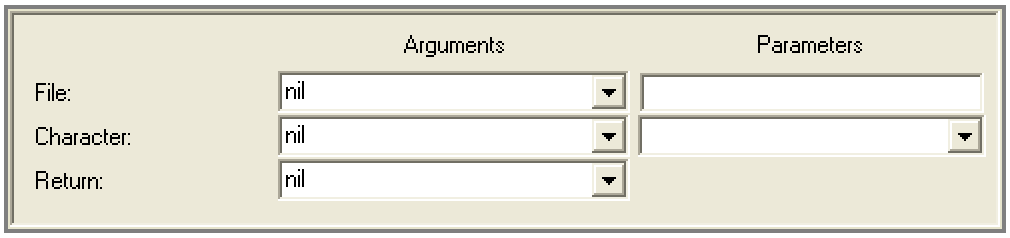
Character

If a variable is specified in this field and the *To Standard Character* radio button is selected, then the parser attempts to match in the input stream the value of this variable instead of any character that has been manually entered in the field next to the radio button.

Return

Variable to hold the result of this construct, which is an integer specifying the previous position pointer in the specified *file*, the position at the time this construct was invoked.

Up To



File:	nil	
Character:	nil	
Return:	nil	

Description

Advances the current position pointer in the specified *file* to the specified place in its input stream and assigns to the *return* variable the string of all characters (if any) over which the pointer was advanced. Note that the position might not have to change in order for the conditions implied by the *up to* parameter to be met (i.e., the input stream might already be positioned at the desired type of object). An error results if the *file* has not previously been opened for input.

File

A string naming the input file to be read. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right. The value of this parameter should be identical to the *file name* with which the file was opened. If this parameter is not specified, the default input file is read.

Character

If a variable is specified in this field, then the parser reads from the input stream until it encounters the value of this variable. If a special character is required (such as a carriage return, line feed, tab, or space) it can be specified using the drop-down list in the argument pane.

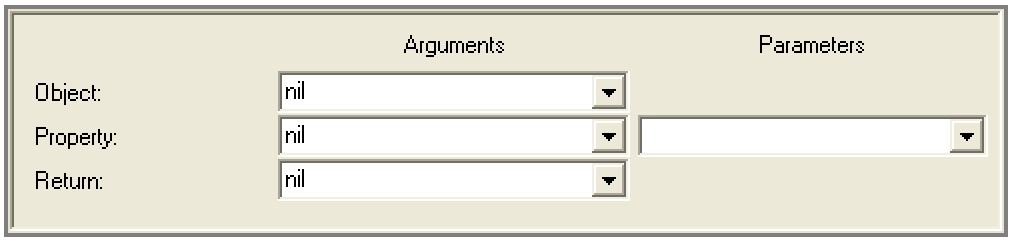
Return

Variable to hold the result of this construct.

THIS PAGE INTENTIONALLY BLANK

Project Query Constructs

Project Property



	Arguments	Parameters
Object:	nil	
Property:	nil	
Return:	nil	

Description

The Project Property Query construct assigns to the *return* variable the value of the specified *property* for the given project *object*.

Object

Project in the CORE database for which the property value is desired. Note that this argument is a project object itself, not the name of a project.

Property

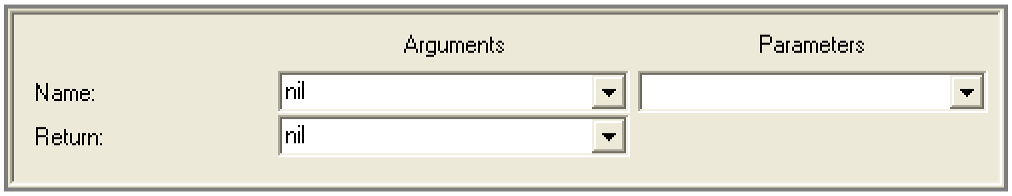
Symbol naming the property of which the value is desired. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right. The available properties are the following:

basePath	a string that is the full path set in the Base Path property for the project. Returns *nil* if this path has not been set.
creationStamp	the time stamp (combination of date and time) at which the project was created
creator	a symbol that is the name of the user that created the project
description	formatted text that is the contents of the project Description property. Returns *nil* if this property is empty.
modificationStamp	the time stamp (combination of date and time) at which a project property was last modified
name	a symbol that is the name of the project
schemaBaselineName	a string that is the name of the schema selected as the base schema when the project was created
schemaVersion	a hierarchical number that is the version (e.g., either 4.0 or 5.0) of the schema being used in the project

Return

Variable to hold the result of this construct.

Project Query



	Arguments	Parameters
Name:	nil	
Return:	nil	

Description

The Project Query construct assigns to the *return* variable the project object specified by its *name*.

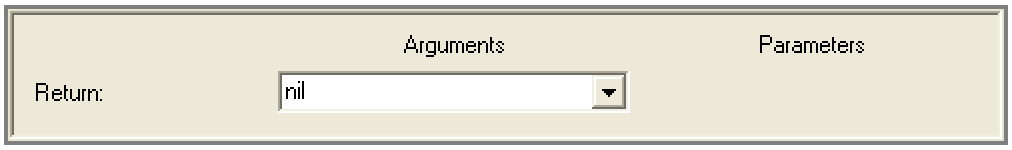
Name

Symbol naming the desired project. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Return

Variable to hold the result of this construct, which is a project object in the CORE database.

Projects



Return: nil

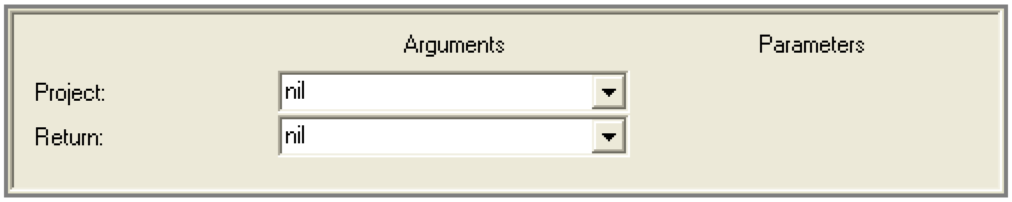
Description

The Projects construct returns a collection of all projects defined in the CORE database.

Return

Variable to hold the result of this construct, which is a set of projects. Note that the objects in the set are project objects themselves, not the names of projects.

Set Project



Project:	nil
Return:	nil

Description

Sets the project for subsequent database and schema queries. If not explicitly set using a Set Project construct, project is defaulted to the project associated with the window from which a script is run.

Project

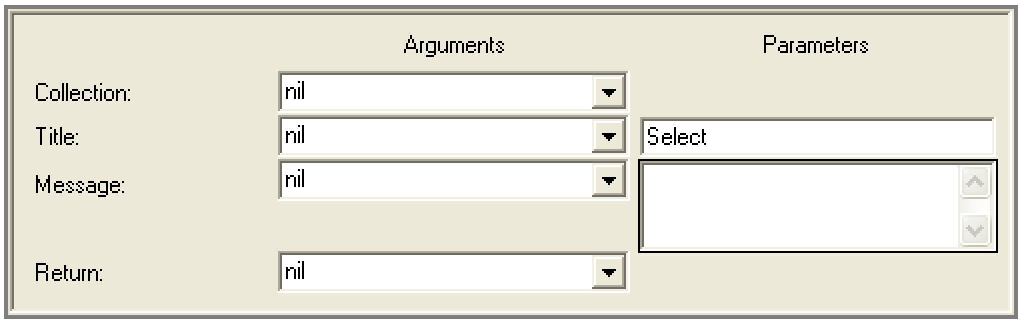
Project in the CORE database from which subsequent retrievals are to be made. Note that this argument is a project object, not the name of a project.

Return

Project in the CORE database that is the previously set project. Note that this argument is a project object, not the name of a project.

Prompting Constructs

Drop-Down Selection Prompt



Collection:	nil	
Title:	nil	Select
Message:	nil	
Return:	nil	

Description

Selection prompts allow the report to prompt the user to select one or more items from a list. The Drop-Down Selection Prompt construct prompts the user to select one string from a drop-down list representing a *collection* of strings and assigns the selected string to the *return* variable. Note that if the Cancel button is pressed, *nil* is returned.

Collection

Collection of strings from which to select. Can be of one of the following data types:

- Array
- Bag
- Dictionary
- Ordered Collection
- Set
- Sorted Collection

If the collection is a dictionary, the strings should be the values stored in the dictionary, not the keys whereby other values are accessed.

Title

String to use as the dialog window title. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

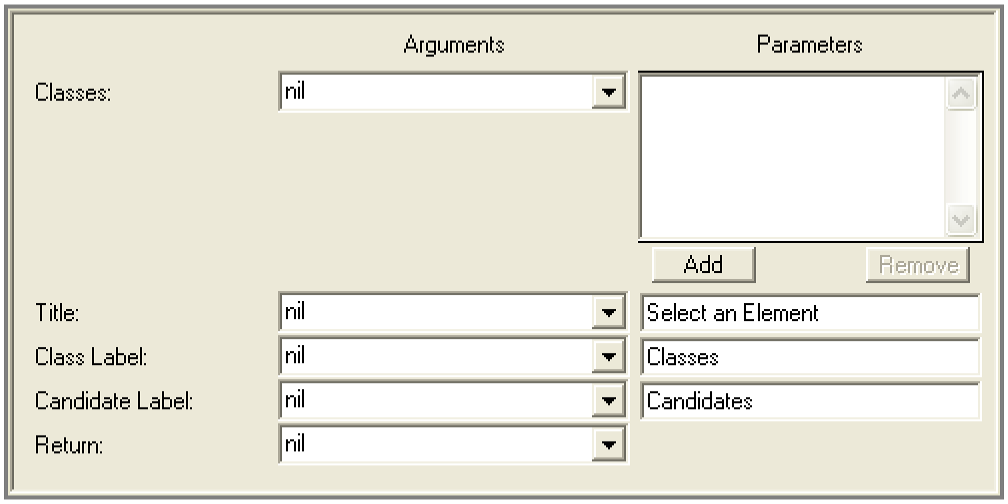
Message

String to use as the notification message displayed in the dialog. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Return

Variable to hold the result of this construct, which is a string or *nil*.

Element Selection Prompt



	Arguments	Parameters
Classes:	nil	
Title:	nil	Select an Element
Class Label:	nil	Classes
Candidate Label:	nil	Candidates
Return:	nil	

Description

The Element Selection Prompt construct displays an Element Selection dialog allowing the user to select one element from any class in the collection of *classes* and assigning the selection to the *return* variable. If no element is selected or if the Cancel button is pressed, then this construct returns *nil*.

Classes

Collection of valid classes from which the user may select an element. Can be of any of the following data types:

Array
Bag
Dictionary
Ordered Collection
Set
Sorted Collection

Title

String to be used as the dialog window title. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Class Label

String to be used as the label for the class pane. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

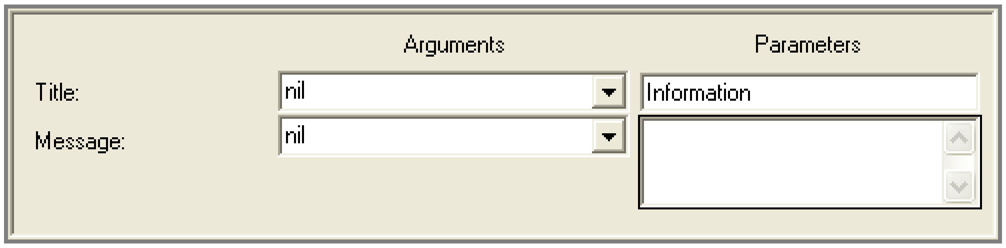
Candidate Label

String to be used as the label for the pane listing all elements in the selected class. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Return

Variable to hold the result of this construct, which is an element or *nil*.

Message Prompt



Title:	nil	Information
Message:	nil	

Description

The Message Prompt construct displays a notification dialog for the user. The user must close the dialog before proceeding.

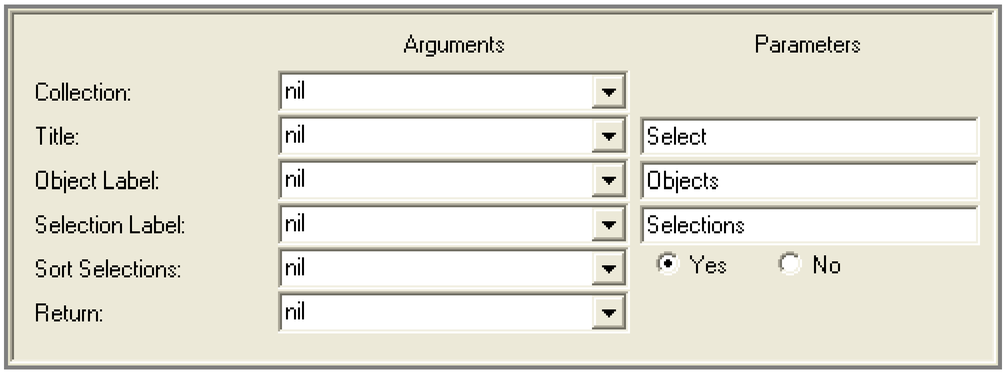
Title

String to use as the dialog window title. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Message

String to use as the notification message displayed in the dialog. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Multiple Selection Prompt



	Arguments	Parameters
Collection:	nil	
Title:	nil	Select
Object Label:	nil	Objects
Selection Label:	nil	Selections
Sort Selections:	nil	[✓ Yes] No
Return:	nil	

Description

Selection Prompts allow the report to prompt the user to select one or more items from a list. The Multiple Selection Prompt construct prompts the user to select one or more objects from a list and assigns the collection of selected objects to the *return* variable. Each object is represented as a string in the list presented to the user. Note that if the Cancel button is pressed, *nil* is returned. If the OK button is pressed without selecting an item from the list, an empty collection is returned.

Collection

Collection of objects from which to select. Can be of one of the following data types:

- Array
- Bag
- Dictionary
- Ordered Collection
- Set
- Sorted Collection

If the collection is a dictionary, the user-selectable objects should be the values stored in the dictionary, not the keys whereby other values are accessed.

Title

String to be used as the dialog window title. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Object Label

Indicates the string with which to label the candidate list (the list from which selections are being made). Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Selection Label

Indicates the string with which to label the list of selected items. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Sort Selections

Boolean object indicating whether or not the selections should be. If the argument value is *true*, this construct returns the user selections in a sorted collection using a simple comparison on the objects in the collection as the sort criterion (but each type of object handles the comparison in its own way). If the argument value is *false*, the selections are returned in the *collection* in the order selected by the user. Can be specified either by a variable from the drop-down on the left or a manual selection from the buttons on the right.

Return

Variable to hold the selected subset of the *collection* described above. It may be *nil* or a sorted collection.

Report Settings Prompt

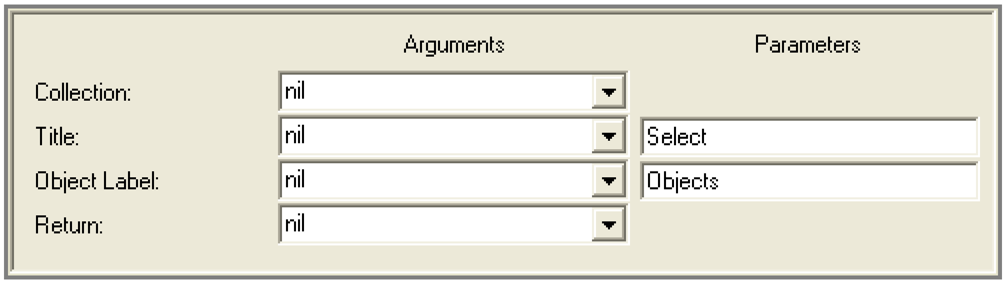
(No settings required)

Description

The Settings Prompt construct opens the Report Settings dialog to allow the user to modify the "prepared by", "prepared for", creator, and date information.

Note: Not available in Simulation scripts.

Single Selection Prompt



	Arguments	Parameters
Collection:	nil	
Title:	nil	Select
Object Label:	nil	Objects
Return:	nil	

Description

Selection Prompts allow the report to prompt the user to select one or more items from a list. The Single Selection Prompt construct prompts the user to select a single object from a list and assigns the selection to the *return* variable. Each object is represented as a string in the list presented to the user. Note that if the Cancel button is pressed, *nil* is returned.

Collection

Collection of objects from which to select. Can be of one of the following data types:

Array
Bag
Dictionary
Ordered Collection
Set
Sorted Collection

If the collection is a dictionary, the user-selectable objects should be the values stored in the dictionary, not the keys whereby other values are accessed.

Title

String to use as the dialog window title. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

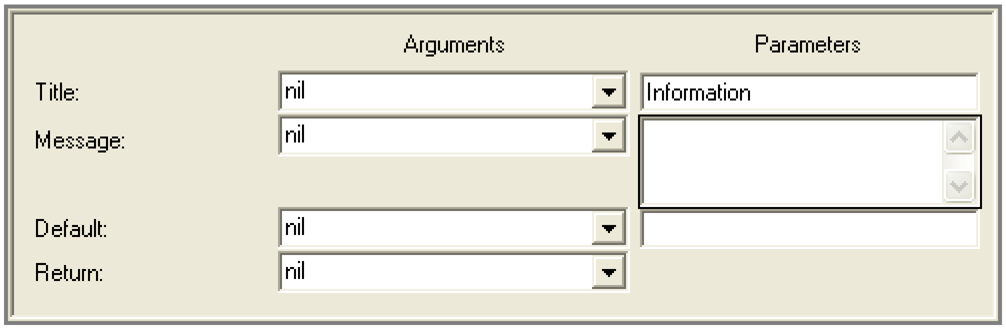
Object Label

Indicates the string with which to label the pane displaying the currently selected object, if any. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Return

Variable to hold the selected object or *nil*.

Value Prompt



	Arguments	Parameters
Title:	nil	Information
Message:	nil	
Default:	nil	
Return:	nil	

Description

The Value Prompt construct prompts the user to enter a string before continuing and assigns it to the *return* variable. Note that if the Cancel button is pressed, an empty string is returned.

Title

String to be used as the dialog window title. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Message

String to be used as the notification message displayed in the dialog. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

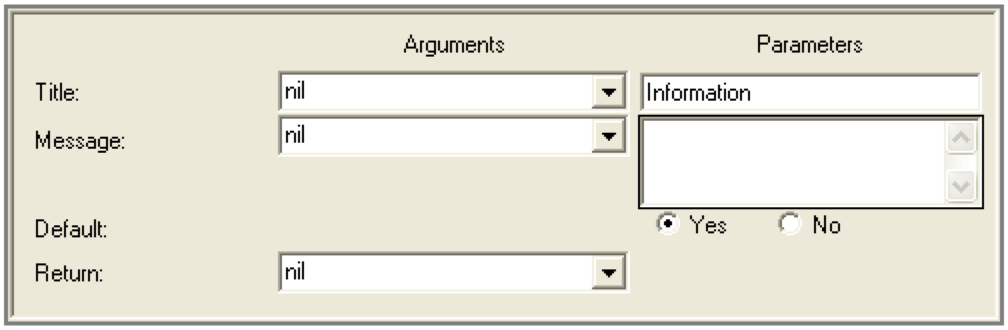
Default

Initial string to be displayed and selected in the entry field when the prompt is opened. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Return

Variable to hold the string entered by the user.

YesOrNo Prompt



	Arguments	Parameters
Title:	nil	Information
Message:	nil	
Default:		[✓ Yes] No
Return:	nil	

Description

The YesOrNo Prompt construct asks the user a yes/no question and assigns to the *return* variable a Boolean *true* if the user clicks on the Yes button or a Boolean *false* if the user clicks on the No button.

Title

String to be used as the dialog window title. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Message

String to be used as the notification message displayed in the dialog. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Default

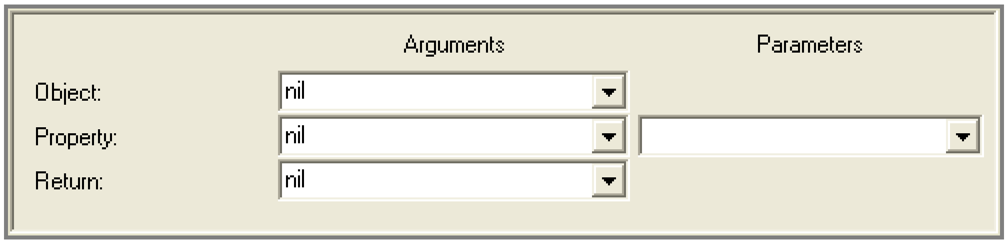
A Boolean indicating which button is the default button. If the object is *true*, then the default is the Yes button. If the object is *false*, then the default is the No button. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Return

Variable to hold the result of this construct, which is a Boolean.

Schema Query Constructs

Attribute Definition Property



	Arguments	Parameters
Object:	nil	
Property:	nil	
Return:	nil	

Description

Assigns to the *return* variable the value of the specified *property* for the given attribute definition.

Object

Attribute definition from the CORE schema for which the property value is desired.

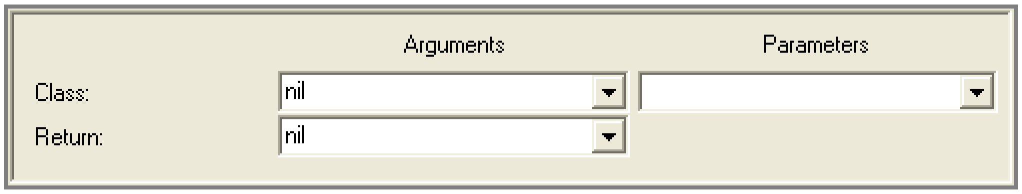
Property

Symbol naming the property which value is desired. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Return

Variable to hold the result of this construct.

Class Attributes



Class: nil
Return: nil

Description

Assigns to the *return* variable a collection of all attribute definitions for the specified *class*. These definitions can then be queried for their property values (e.g., name, type) as necessary.

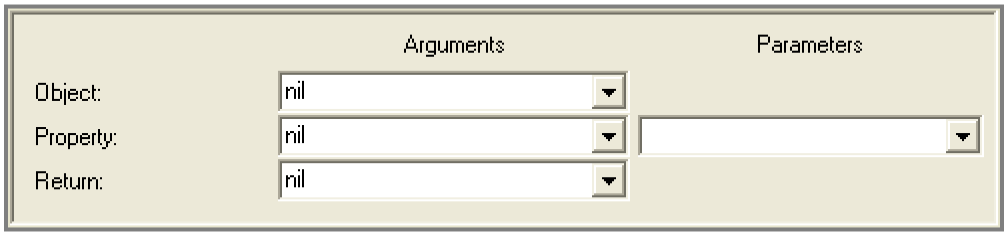
Class

A class defined in the CORE schema for which the attribute definitions are desired. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right. Note that this argument is a class object itself, not just the name of a class.

Return

Variable to hold the result of this construct, which is a sorted collection of attribute definitions ordered alphabetically by display name.

Class Property



	Arguments	Parameters
Object:	nil	
Property:	nil	
Return:	nil	

Description

Assigns to the *return* variable the value of the specified *property* for the given class *object*.

Object

Class in the CORE schema for which the property value is desired. Note that this argument is a class object itself, not just the name of a class.

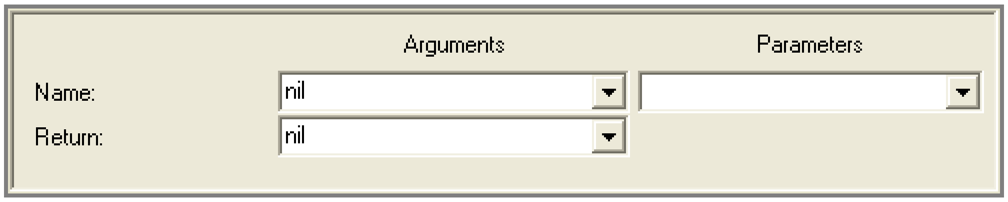
Property

Symbol naming the property which value is desired. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Return

Variable to hold the result of this construct.

Class Query



	Arguments	Parameters
Name:	nil	
Return:	nil	

Description

Assigns to the *return* variable the class object specified by its *name*.

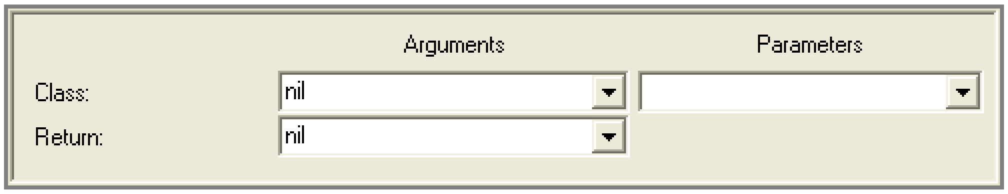
Name

Symbol naming the desired class. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Return

Variable to hold the result of this construct, which is a class object in the CORE schema.

Class Relations



Class: nil

Return: nil

Description

Assigns to the *return* variable a collection of all relations from the specified *class*. These definitions can then be queried for their property values (e.g., name, complement) as necessary.

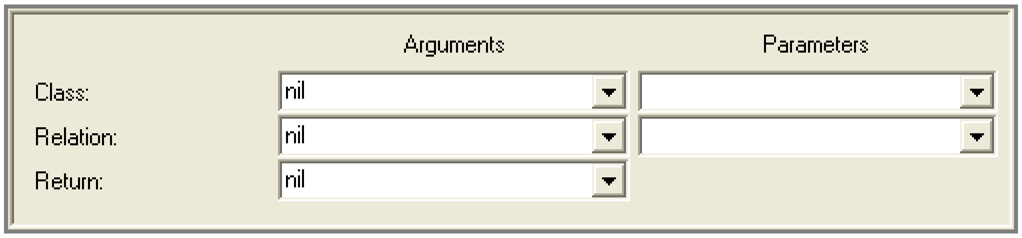
Class

A class defined in the CORE schema for which the relations are desired. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right. Note that this argument is a class object itself, not just the name of a class.

Return

Variable to hold the result of this construct, which is a sorted collection of relations ordered alphabetically by name.

Class Target Classes



	Arguments	Parameters
Class:	nil	
Relation:	nil	
Return:	nil	

Description

Assigns to the *return* variable a collection of all classes defined in the CORE schema as targets of the specified *relation* from the specified *class*. Relationships defined by the *relation* can be created in the CORE database with elements of the specified *class* as the source and elements of the target classes as targets, subject to the limitations imposed by the user in the schema.

Class

Class defined in the CORE schema for which the target classes of the specified *relation* are desired. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right. Note that this argument is a class object itself, not just the name of a class.

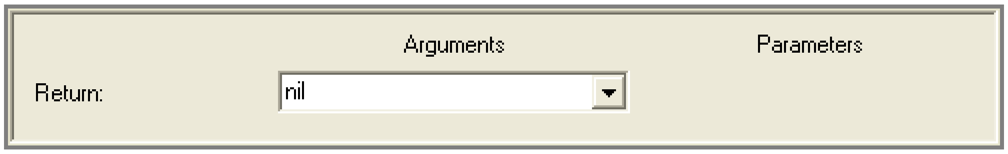
Relation

Relation defined in the CORE schema for which the target classes are desired. Should be a valid relation from the specified *class*. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right. Note that this argument is a relation object itself, not just the name of a relation.

Return

Variable to hold the result of this construct, which is a sorted collection of class objects ordered alphabetically by display name.

Database Classes


Arguments Parameters

Return: nil

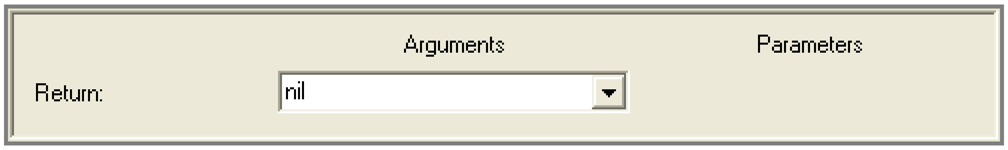
Description

Assigns to the *return* variable a collection of all concrete (i.e., non-abstract) classes in the schema.

Return

Variable to hold the result of this construct, which is an ordered collection of classes ordered alphabetically by display name. Note that the objects in the collection are class objects themselves, not just the names of classes.

Facilities



Arguments | Parameters

Return: nil

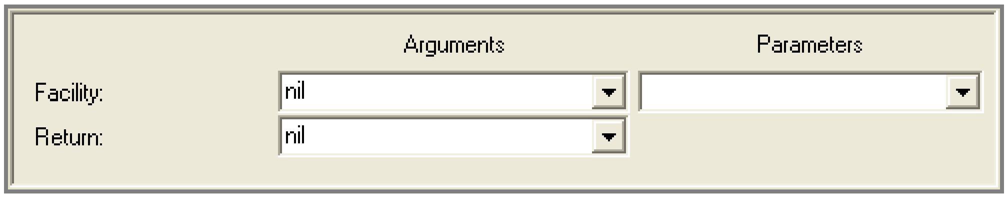
Description

Assigns to the *return* variable a collection of all facilities defined in the CORE schema.

Return

Variable to hold the result of this construct, which is a sorted collection of facilities ordered alphabetically by name. Note that the objects in the collection are facility objects themselves, not just the names of facilities.

Facility Database Classes



	Arguments	Parameters
Facility:	nil	
Return:	nil	

Description

Assigns to the *return* variable a collection of all concrete (i.e., non-abstract) classes defined in the CORE schema as belonging to the specified *facility*.

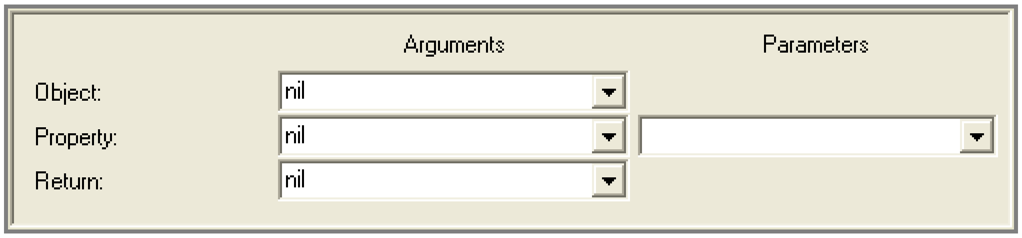
Facility

Facility in the CORE schema to be queried. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right. Note that this argument is a facility object itself and not just the name of a facility.

Return

Variable to hold the result of this construct, which is an ordered collection of classes ordered alphabetically by display name. Note that the objects in the collection are class objects themselves, not just the names of classes.

Facility Property



	Arguments	Parameters
Object:	nil	
Property:	nil	
Return:	nil	

Description

Assigns to the *return* variable the value of the specified *property* for the given facility *object*.

Object

Facility in the CORE schema for which the property value is desired. Note that this argument is a facility object, not just the name of a facility.

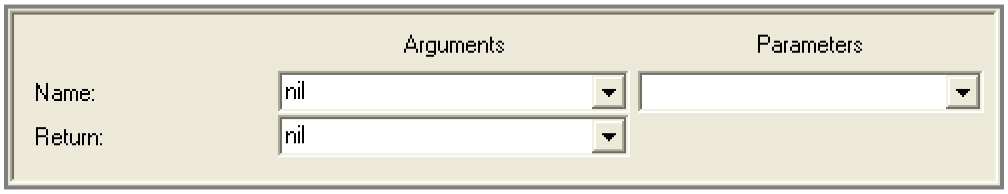
Property

Symbol naming the property which value is desired. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Return

Variable to hold the result of this construct.

Facility Query



	Arguments	Parameters
Name:	nil	
Return:	nil	

Description

Assigns to the *return* variable the facility object specified by its *name*.

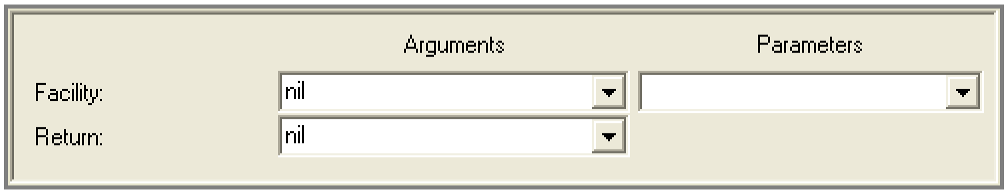
Name

Symbol naming the desired facility. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Return

Variable to hold the result of this construct, which is a facility object in the CORE schema.

Facility Schema Classes



Facility:	nil	
Return:	nil	

Description

Assigns to the *return* variable a collection of all abstract and concrete classes used to identify the classes in the CORE schema that belong to the specified *facility*.

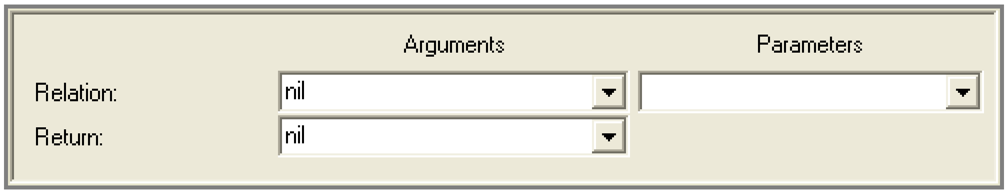
Facility

Facility in the CORE schema to be queried. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right. Note that this argument is a facility object itself and not just the name of a facility.

Return

Variable to hold the result of this construct, which is a sorted collection of classes ordered alphabetically by display name. Note that the objects in the collection are class objects themselves, not just the names of classes.

Relation Attributes



	Arguments	Parameters
Relation:	nil	
Return:	nil	

Description

Assigns to the *return* variable a collection of all attribute definitions for the specified *relation*. These definitions can then be queried for their property values (e.g., name, type) as necessary.

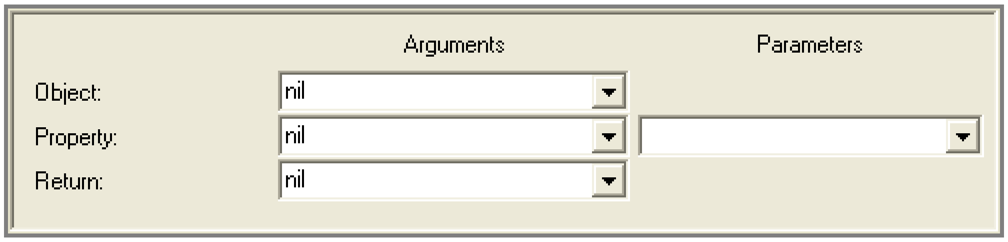
Relation

A relation defined in the CORE schema for which the attribute definitions are desired. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right. Note that this argument is a relation object itself, not just the name of a class.

Return

Variable to hold the result of this construct, which is a sorted collection of attribute definitions ordered alphabetically by display name.

Relation Property



	Arguments	Parameters
Object:	nil	
Property:	nil	
Return:	nil	

Description

Assigns to the *return* variable the value of the specified *property* for the given relation *object*.

Object

Relation in the CORE schema for which the property value is desired. Note that this argument is a relation object itself, not just the name of a relation.

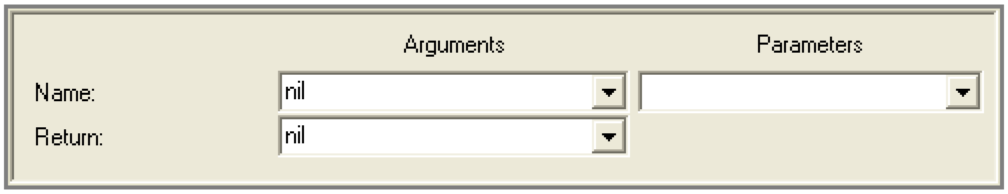
Property

Symbol naming the property which value is desired. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Return

Variable to hold the result of this construct.

Relation Query



Name: nil
Return: nil

Description

Assigns to the *return* variable the relation object specified by its *name*.

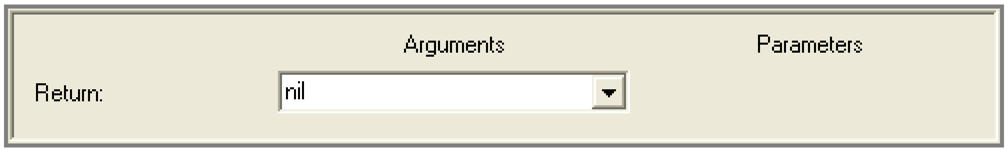
Name

Symbol naming the desired relation. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Return

Variable to hold the result of this construct, which is a relation object in the CORE schema.

Relations



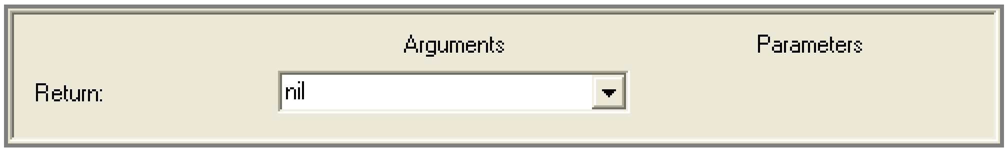
Description

Assigns to the *return* variable a collection of all relations in the schema.

Return

Variable to hold the result of this construct, which a sorted collection of relations ordered alphabetically by name. Note that the objects in the collection are relation objects themselves, not just the names of relations.

Schema Classes



Return: nil

Description

Assigns to the *return* variable a collection of all classes in the schema.

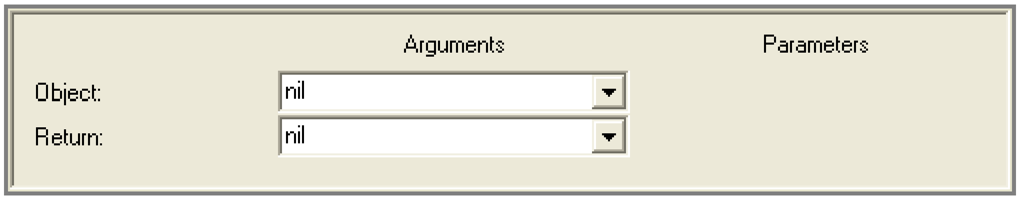
Return

Variable to hold the result of this construct, which a sorted collection of classes ordered alphabetically by name. Note that the objects in the collection are class objects themselves, not just the names of classes.

THIS PAGE INTENTIONALLY BLANK

Simulation Constructs

Resource Level Query



	Arguments	Parameters
Object:	nil	
Return:	nil	

Description

Assigns to the *return* variable the currently available amount of the specified resource if the resource is used in the scope of the current model being simulated. If the resource is defined but not utilized by the model, the construct returns the initial value of the resource. If the resource is undefined, the construct returns zero.

Object

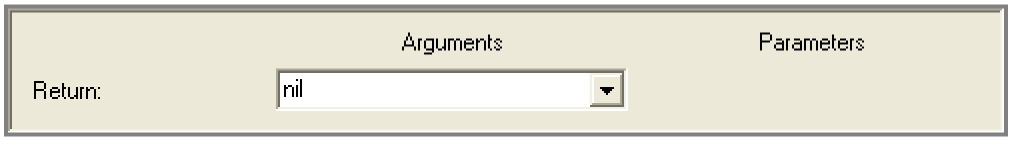
Variable containing the element in the Resource class from the CORE database to query.

Return

Variable to hold the result of this construct, which is either a float or integer depending on the Amount Type attribute of the Resource element.

Note: Only available to Simulation scripts.

Script Context Query



Return: nil

Description

Assigns to the *return* variable the context in which the script is executing (i.e., the Function element in the case of a Begin Logic, Duration, End Logic, Exit Logic, or Timeout attribute; the Resource element in the case of a Maximum Amount attribute; and the relationship object in the case of an Amount attribute).

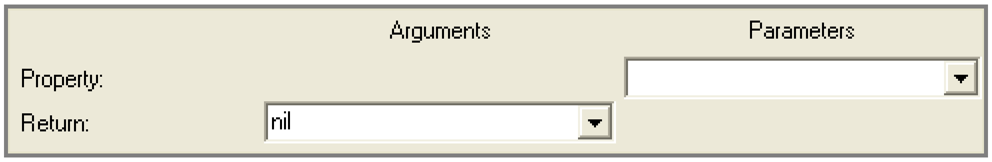
Return

Variable to hold the result of this construct, which is either a Function, Resource, or relationship object.

Notes: Only available to Simulation scripts.

The Begin Logic and End Logic attributes are available as extensions. Begin Logic is executed at the very start of Function execution, after enablement and triggering but before Resources are acquired. End Logic is executed at the very end of Function execution, after Resources are produced and Items are output. These attributes can be added by importing the appropriate schema extension file (i.e., either Base40SimulationExtensions.xml, C4ISR40SimulationExtensions.xml, Base50SimulationSchemaExtensions.xml, or DoDAF50SimulationSchemaExtensions.xml) located in the Samples subfolders.

Simulator Property Query



Property:		
Return:	nil	

Description

Assigns to the *return* variable the selected simulator *property* during simulation execution.

Property

The simulator property to be returned. This field is a drop-down list with the following selections:

areLinksConstrained	a Boolean that is true if the Link constraints are being applied, false otherwise.
time	the current simulation clock time
timeLimit	user set simulation clock time at which the simulation will automatically be terminated if it has not already completed. Returns *nil* if it has not been set.

Return

Variable to hold the result of this construct.

Note: Only available to Simulation scripts.

THIS PAGE INTENTIONALLY BLANK

Utilities Constructs

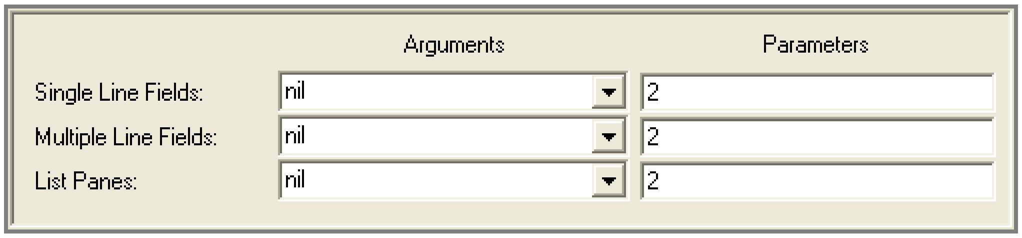
Beep

(No settings required)

Description

Beeps the system speaker.

Breakpoint



	Arguments	Parameters
Single Line Fields:	nil	2
Multiple Line Fields:	nil	2
List Panes:	nil	2

Description

Stops script execution and opens a debugger on the current script.

Single Line Fields

An integer specifying the number of entry fields to display values in the debugger (good for dates, strings, integers, elements, etc.).

Multiple Line Fields

An integer specifying the number of formatted text fields to display large blocks of information (good for text fields such as descriptions).

List Panes

An integer specifying the number of list panes to display collections (intended for collections such as arrays, ordered collections, and dictionaries).

Although any variable can be displayed in any pane, it is most valuable to use the pane best suited for that variable.

Cancel Report

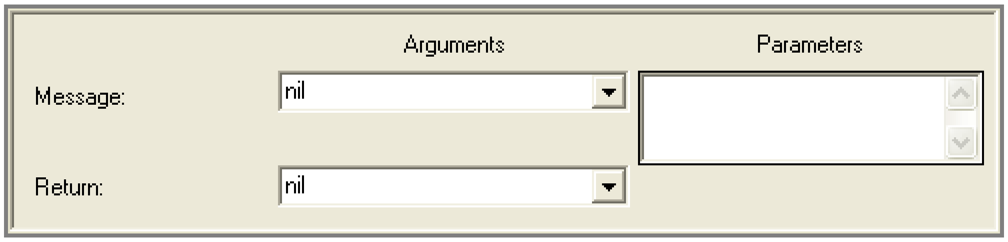
(No settings required)

Description

Aborts the execution of the script. This is the same as if the Cancel button on the progress indicator is pressed.

Note: Not available to Simulation scripts.

Change Indicator



Message: nil

Return: nil

Description

Allows the report to change the message currently displayed in the progress indicator. This is a useful technique to keep the individual running a report informed of the status. Note that the prior message is assigned to the *return* variable.

Message

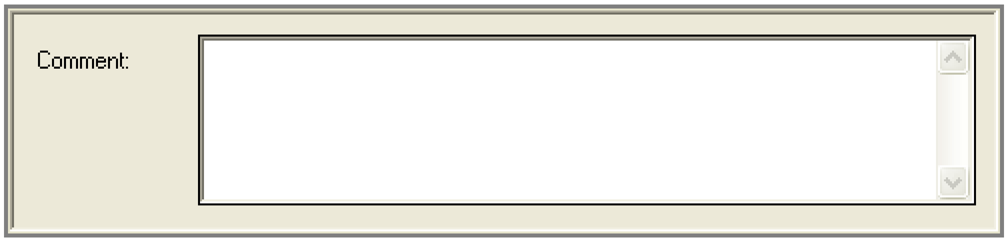
String to be displayed. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Return

Variable to hold the result of this construct, which is the previously displayed string.

Note: Not available to Simulation scripts.

Comment



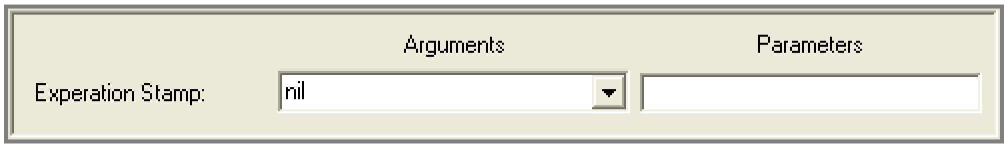
Description

Comment allows the script author to add internal comments to document the script.

Comment

An explanation of what is accomplished by following constructs, assumptions made, etc.

Expire Outputs



Experation Stamp: nil

Description

Expire Output is used in a CORE2net script to tell a browser when to ‘expire’ the output page. Expire Output sets the ‘Expires’ flag within the HTTP header of the output document**.**

Expiration Stamp

A time stamp that ensures the latest database updates are shown in CORE2net output.

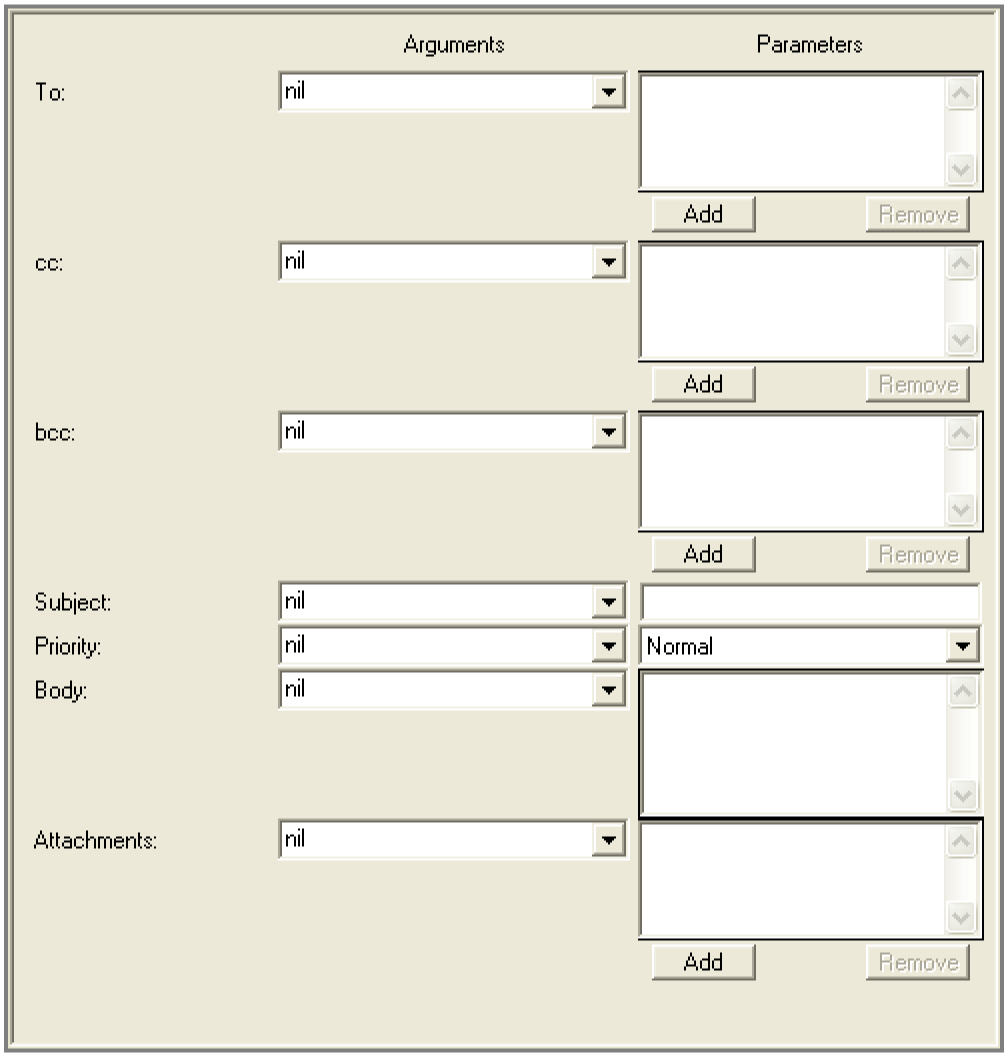
Open Transcript

(No settings required)

Description

Opens the report transcript. Note that this happens automatically if “Transcript” or “Both [File and Transcript]” is specified as the destination of an output construct or if an error is encountered.

Send Mail Message



Description

Allows a script to send an e-mail message. This could be used in a script to notify users of any database modifications after a specific data and time.

Requires that MAPI32.dll be enabled. If Microsoft mail options have been installed and configured, this .dll should be enabled. If CORE does not allow you to utilize the send feature contact your network system administrator to verify that this .dll is enabled.

Addresses

Ordered Collections of the To, cc, and bcc distribution addresses. Each collection can be specified by a variable from the drop-down on the left or a manual entry using the buttons on the right.

Subject

String containing the subject line of the message. Can be specified either by a variable from the drop-down on the left or a manual entry of the text on the right.

Priority

Symbol designating the delivery priority of the message. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

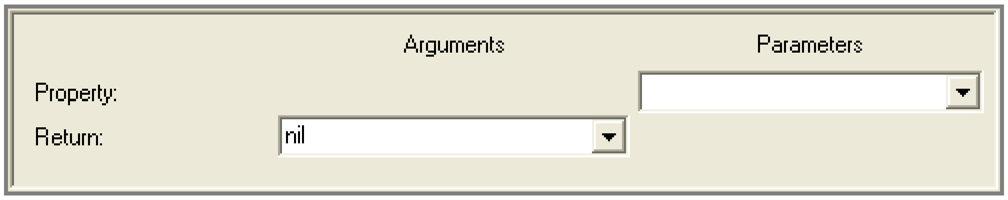
Body

String containing the text of the message. Can be specified either by a variable from the drop-down on the left or a manual entry of the text on the right.

Attachments

Ordered collection of full path names for files to be attached to the message. Can be specified either by a variable from the drop-down on the left or a manual identification of each file on the right.

System Property Query



Description

Assigns to the *return* variable the value of the *property*.

Property

The system property to be returned. This field is a drop-down list with the following selections:

allReportScripts	a dictionary of all reports scripts available to the CORE project. Each value is a string that is the full path for the scripts with a key that is a string containing the name of the script.
baselineSchemaName	the name of the schema selected as the base schema when the current project was created
commonScriptDirectory	a string that is the full path for the directory of Common scripts. The value is context sensitive to whether it is executed from a report or CORE2net script.
commonScriptURI	a string that is the Uniform Resource Identifier (URI) for the directory of Common CORE2net scripts.
currentScriptDirectory	a string that is the full path for the directory of the script that is being executed
date	the current date
imageName	a string that is the full path for the current image file (returns nil in CORE Enterprise)
isEnterprise	a Boolean indicating whether the currently executing product is CORE Enterprise
mainScriptDirectory	a string that is the full path for the directory of Main scripts. The value is context sensitive to whether it is executed from a report or CORE2net script.
mainScriptURI	a string that is the Uniform Resource Identifier (URI) for the directory of Main CORE2net scripts.
outputDirectory	the full path of the Output directory as set in User Preferences
project	the current project
schemaVersion	a hierarchical number which is the version (e.g., either 4.0 or 5.0) of the schema being used in the project
time	the current time
timeStamp	the current time stamp (combination of the date and time)
user	the current system user
userAgent	a user agent object that represents the browser in use in a CORE2net session. Note that this object responds to name (returns a string), isInternetExplorer (returns a Boolean), and isNetscape (returns a Boolean).
userScriptDirectory	a string that is the full path for the directory of User scripts. The value is context sensitive to whether it is executed from a report or CORE2net script.

userScriptURI	a string that is the Uniform Resource Identifier (URI) for the directory of User CORE2net scripts.

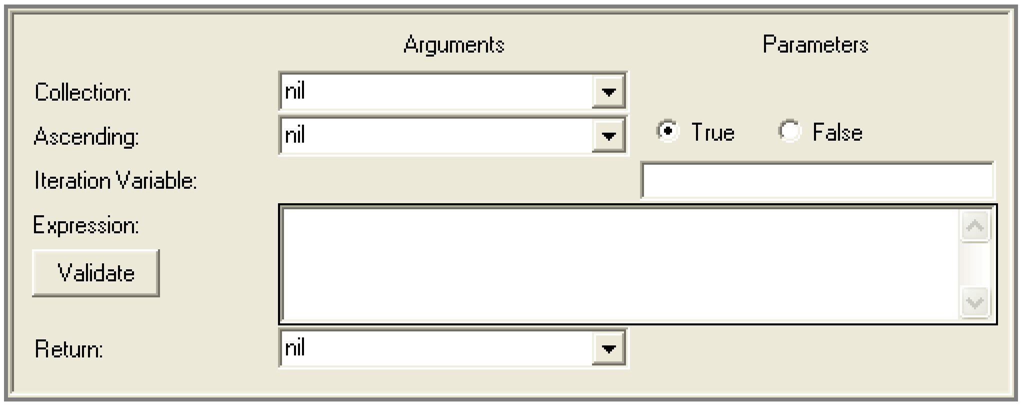
Return

Variable to hold the specified system property. Unless otherwise stated, a system property is returned as an object.

THIS PAGE INTENTIONALLY BLANK

Variable Manipulation Constructs

Database Object Sort



	Arguments	Parameters
Collection:	nil	
Ascending:	nil	[✓ True] False
Iteration Variable:		
Expression:		
Return:	nil	

Description

Assigns to the *return* variable a new ordered collection containing all the same objects as are in the *collection* to be sorted. The objects in the new collection are ordered so that, when substituted for the *iteration variable* in the specified *expression*, the value that results for each one is less than or equal to the value that results when the object in the next higher ordinal position within the new collection is substituted for the *iteration variable* in the specified *expression*. Note that the original *collection* is not modified.

Collection

Variable containing a collection of objects to be sorted. Can be of one of the following data types:

- Array
- Bag
- Dictionary
- Ordered Collection
- Set
- Sorted Collection
- String
- Symbol

If the collection is a dictionary, the objects in the returned collection and the objects that are assigned to the *iteration variable* as described above are the values stored in the dictionary, not the keys whereby the values are accessed. If the collection is a string or symbol, this construct returns a collection of its characters in a new order.

Ascending

Boolean object indicating whether the collection should be sorted in ascending or descending order. If the argument value is *false*, this construct returns the objects in the *collection* in descending order. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Iteration Variable

String specifying the name of a variable referenced in the *expression*. This variable is declared within the scope of this construct but will not be accessible after this construct terminates.

Expression

Indicates the criterion to be used in sorting. An expression consists of operators in the CORE expression language, variables defined using the Edit Variables command under the Properties menu of the Report Editor, constants that can be coded in some data types, and the *iteration variable* described above.

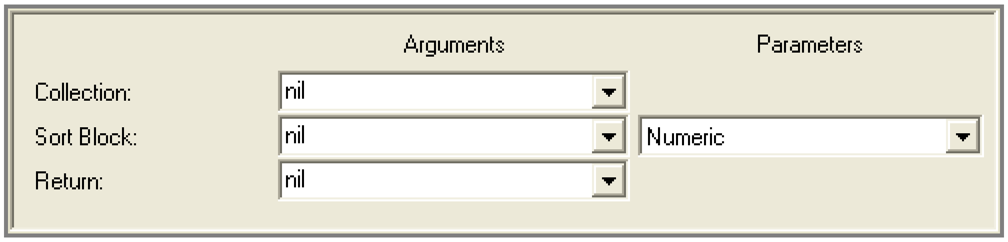
Validate

Validates the syntax of the *expression* to ensure that neither operators nor operands appear in unexpected locations, all operators are recognized, and all variables have been defined. No type or bounds checking can be performed.

Return

Variable to hold the result of this construct, which is an ordered collection.

Element Sort



	Arguments	Parameters
Collection:	nil	
Sort Block:	nil	Numeric
Return:	nil	

Description

Assigns to the *return* variable a new sorted collection containing all the same elements as are in the *collection* to be sorted. Elements in the new collection are ordered based upon the specified *sort block*. This construct differs from the Database Object Sort Construct in that the Element Sort Construct uses a pre-defined CORE sort block to sort the elements whereas the Database Object Sort Construct sorts the elements based upon an expression entered as a construct parameter.

Collection

Variable containing the collection of elements to be sorted. This collection can contain either elements from a single class or elements from multiple classes.

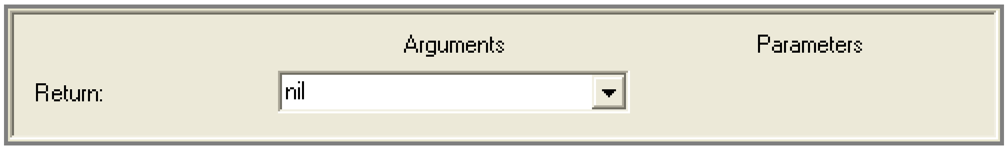
Sort Block

Sort block definition of a sort block defined in the CORE project. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right. This sort criteria continues to govern the behavior of the sorted collection that results from this construct.

Return

Variable to hold the sorted collection of elements.

Element Sort Blocks



	Arguments	Parameters
Return:	nil	

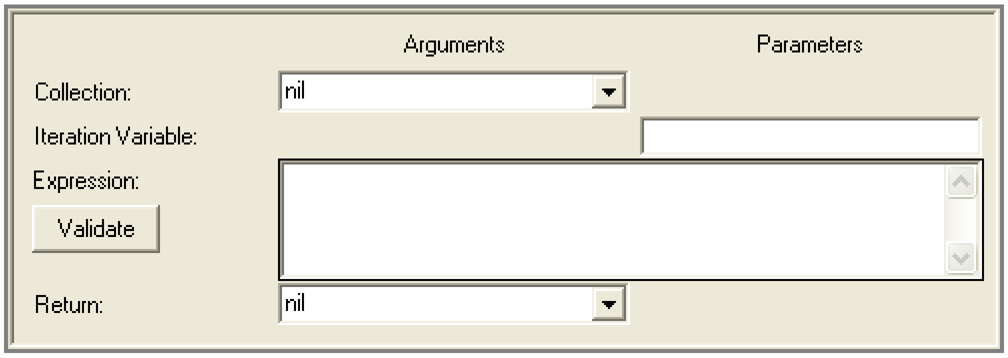
Description

Assigns to the *return* variable an alphabetically sorted collection containing all sort block definitions in the CORE project. This list contains both single class sort blocks and multi-class sort blocks. Therefore, it is very important that a standard naming convention be used when defining sort blocks in CORE. For example, if the names of all multi-class sort blocks have “by class” appended (e.g., Numeric by class), it is very easy to distinguish single class sort blocks from multi-class sort blocks.

Return

The variable to hold the sorted collection of all sort block definitions.

Filter



Collection: nil

Return: nil

Description

Assigns to the *return* variable a new collection of the same data type as the *collection* to be filtered (except that if the *collection* is a symbol, then a string is returned). Each object in the *collection* to be sorted is assigned, in turn, to the *iteration variable*, and, if the specified *expression* then evaluates to a Boolean *true*, the object is added to the new collection.

Collection

Variable containing the collection of objects to be filtered. Can be of one of the following data types:

- Array
- Bag
- Dictionary
- Ordered Collection
- Set
- Sorted Collection
- String
- Symbol

Iteration Variable

String specifying the name of a variable referenced in the *expression*. This variable is declared within the scope of this construct but will not be accessible after this construct terminates.

If the *collection* to be filtered is a dictionary, then only the values stored in the dictionary will be assigned to the iteration variable, not the keys whereby the values are accessed. If the *collection* is a string or symbol, individual characters are assigned to the iteration variable.

Expression

Indicates the criterion to be used in filtering. An expression consists of operators in the CORE expression language, variables defined using the Edit Variables command under the Properties menu of the Script Editor, constants that can be coded in some data types, and the *iteration variable* described above. This expression must evaluate to a Boolean.

Validate

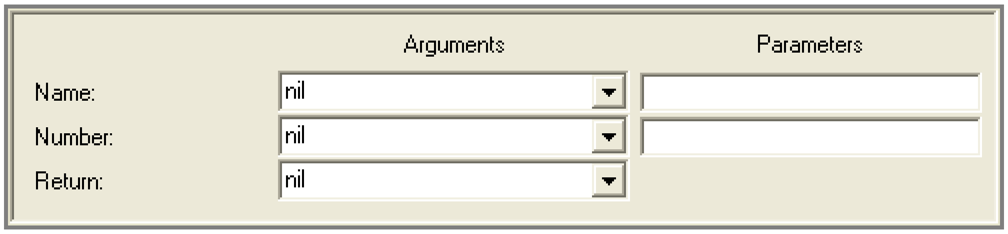
Validates the syntax of the *expression* to ensure that neither operators nor operands appear in unexpected locations, all operators are recognized, and all variables have been defined. No type or bounds checking can be performed.

Return

Variable to hold the result of this construct. It is of the same data type as the *collection* to be filtered (except that if the *collection* is a symbol, then a string is returned) and contains a subset of the objects therein. If the collections are ordered or sorted, the objects they contain maintain their relative positions.

If the *collection* is a sorted collection, then the same sort criterion will be maintained by the returned collection. If the *collection* is a dictionary, each filtered value will be associated with the same key.

Report Section



	Arguments	Parameters
Name:	nil	
Number:	nil	
Return:	nil	

Description

Assigns to the *return* variable a new object of the report section data type. One possible use of this construct is as follows:

This construct is executed multiple times at the beginning of a report script to create different report sections, and each resultant object is added to a collection. A Multiple Selection Prompt is then opened on the collection, allowing the user to choose a subset of the report sections. An iterate construct iterates over the selected report sections and builds a dictionary from them using a simple data type such as a symbol for keys. The script language statements corresponding to each section of the report can be place in “if-then” constructs, using the presence of the associate key in the dictionary as the condition for execution.

Name

A string naming the new report section. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

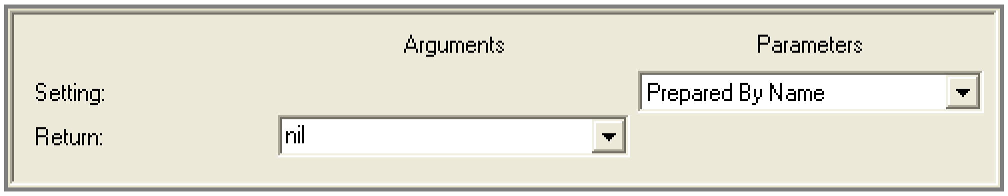
Number

A hierarchical number according to which report sections can be sorted. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Return

Variable to hold the result of this construct, which is a report section.

Report Setting



	Arguments	Parameters
Setting:		Prepared By Name
Return:	nil	

Description

Assigns to the *return* variable the value of the selected *setting* property from the report settings dialog (e.g., Prepared By Name).

Setting

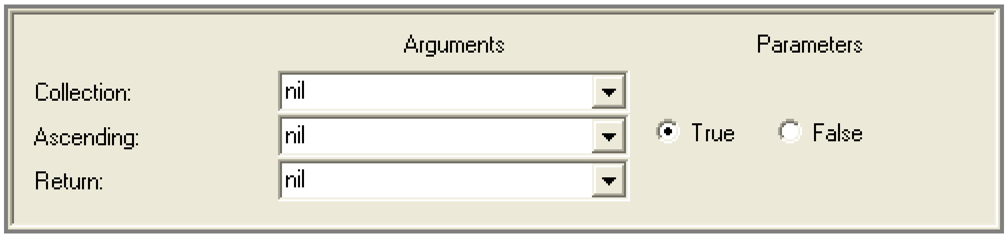
Name of the property which value is to be returned. It is one of the following strings: “Creator Override Value”, “Date Override Value”, “Override Creator”, “Override Date”, “Prepared By Address”, “Prepared By Name”, “Prepared For Address”, or “Prepared For Name”.

Return

Variable to hold the result of this construct.

Note: Not available to Simulation scripts.

Sort



	Arguments	Parameters
Collection:	nil	
Ascending:	nil	● True ○ False
Return:	nil	

Description

Assigns to the *return* variable a new, sorted ordered collection containing all the same objects as are in the *collection* to be sorted. Note that the original collection is not modified.

Collection

Variable containing a collection of objects to be sorted. Can be of one of the following data types:

Array
Bag
Dictionary
Ordered Collection
Set
Sorted Collection
String
Symbol

If the collection is a dictionary, the objects in the returned collection are the values stored in the dictionary, not the keys whereby the values are accessed. If the collection is a string or symbol, this construct returns a collection of its characters in a new order.

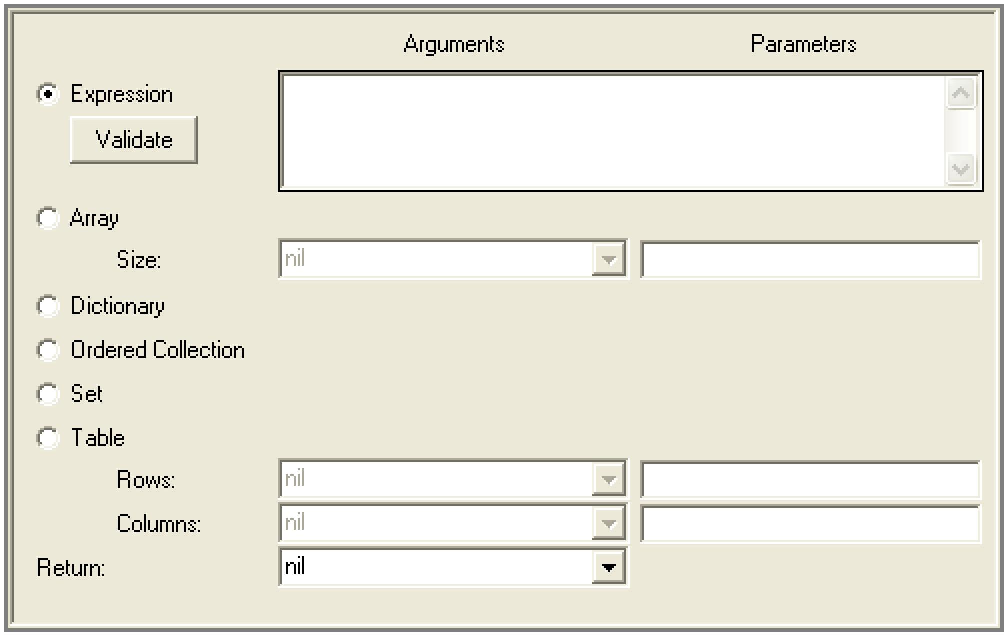
Ascending

Boolean object indicating whether the collection should be sorted in ascending or descending order. If the argument value is *false*, this construct returns the objects in the *collection* in descending order. Can be specified either by a variable from the drop-down on the left or a manual selection from the drop-down on the right.

Return

Variable to hold the result of this construct, which is an ordered collection.

Variable Assignment



Selected: Expression

Size: nil
Rows: nil
Columns: nil
Return: nil

Description

Assigns an object to the *return* variable. The object assigned can be a new and empty collection of one of the available data types or it can be the result of a CORE *expression*.

Expression

An expression consisting of operators in the CORE expression language, variables defined using the Edit Variables command under the Properties menu of the Script Editor, and constants that can be coded in some data types.

Validate

Validates the syntax of the *expression* to ensure that neither operators nor operands appear in unexpected locations, all operators are recognized, and all variables have been defined. No type or bounds checking can be performed.

Data Type

This construct can create new objects of the following types:

- Array
- Dictionary
- Ordered Collection
- Set
- Table

Arrays and tables have fixed dimensions that are also defined by this construct.

Size

The number of indices allocated to the returned array if the Array *data type* is selected. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Rows

The number of rows allocated in the returned table if the Table *data type* is selected. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Columns

The number of columns allocated in the returned table if the Table *data type* is selected. Can be specified either by a variable from the drop-down on the left or manual entry in the field on the right.

Return

Variable to hold the result of this construct.

THIS PAGE INTENTIONALLY BLANK

Construct Index

Vitech Corporation
2070 Chain Bridge Road, Suite 100
Vienna, Virginia 22182-2536
703.883.2270 FAX: 703.883.1860
Customer Support: support@vitechcorp.com
www.vitechcorp.com