COREscript Expression Language Reference Guide

Vitech

Restricted Rights Legend

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7013.

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

Publication Date: September 2007

Table of Contents

THIS PAGE INTENTIONALLY BLANK

General Information

COREscript is a procedural/object-oriented language hybrid. The control of the language is procedural in nature, but the data handled are objects. This guide provides a reference to the messages (also known as methods or functions) that objects, termed receivers, will respond to. It is organized by type of the receiver object.

In general, if a method is expecting an argument, its name is followed by a ':' character. These are referred to as keyword methods. There are single and multiple keyword methods. Keyword methods require as many arguments as they have keywords. In this reference guide, the arguments are designated as argument1, argument2, etc. indicating the order of the arguments and, thereby, which keyword the argument follows. For example, consider the common array. To store an object in an array, the at:put: method is used. The following expression:

TheArray at: 5 put: 7

will store the value 7 at the 5th position in the array TheArray. The 5 is argument1 and 7 is argument2. In the documentation, the method would be referred to as the **at:put:** message.

All objects respond to the messages identified below.

displayString	Answers a string representation of the object. The exact form of the representation depends on the type of object.
isNil	Returns Boolean false for all defined objects; otherwise, returns true.

THIS PAGE INTENTIONALLY BLANK

General Object Types

Array

An array is a collection of objects accessed using an integer, called an index, which is the position of the desired item in the collection. The size of an array is fixed at its creation.

<	Answers a Boolean indicating whether or not the receiver is less than argument1. Arrays are ordered by the values in each index position, with index position decreasing in significance as the index increases. If one array is equal to the lower index positions of another, the smaller array is less than the larger.
<=	Answers a Boolean indicating whether or not the receiver is less than or equal to argument1. Arrays are ordered by the values in each index position, with index position decreasing in significance as the index increases. If one array is equal to the lower index positions of another, the smaller array is less than the larger.
<>	Answers a Boolean indicating whether or not the receiver is not equal to argument1.
=	Answers a Boolean indicating whether or not the receiver is equal to argument1.
>	Answers a Boolean indicating whether or not the receiver is greater than argument1. Arrays are ordered by the values in each index position, with index position decreasing in significance as the index increases. If one array is equal to the lower index positions of another, the smaller array is less than the larger.
>=	Answers a Boolean indicating whether or not the receiver is greater than or equal to argument1. Arrays are ordered by the values in each index position, with index position decreasing in significance as the index increases. If one array is equal to the lower index positions of another, the smaller array is less than the larger.
asOrderedCollection	Answers a new ordered collection that is a copy of the receiver array converted to an ordered collection. The order is the same as in the array.
asSet	Answers a new collection that is a copy of the receiver converted to a set. The order of the objects in the set is arbitrary. A set only maintains a single object with any given value, so if the array contains multiple objects with the same value or multiple references to the same object, the multiple objects or references are lost in the conversion process.
asSortedCollection	Answers a new collection that is a copy of the array converted to a sorted collection. The objects in the new collection are sorted 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).
at:	Answers the object stored in the ordinal position given by argument1. Argument1 must be an integer. An error results if the argument is outside the bounds of the receiver array.
at:ifAbsent:	Answers the object stored in the ordinal position given by argument1. Argument1 must be an integer. If argument1 is outside the bounds of the receiver array, then and only then is the expression in argument2 evaluated and the result returned. Argument2 must be a block, i.e., an expression enclosed in square brackets.

at:put:	Replaces the object at ordinal position argument1 in the receiver array with argument2. Argument1 must be an integer. An error results if argument1 is outside the bounds of the array. Answers argument2.
concat:	Answers a new array containing all of the receiver's objects in order followed by all argument1's objects. Argument1 must be an array, ordered collection, sorted collection, string, or symbol.
copyFrom:to:	Answers a new array containing every object from ordinal position argument1 through ordinal position argument2 in the receiver array. Argument1 and argument2 must be integers. An error results if either argument1 or argument2 represents a position outside the bounds of the receiver array.
displayString	Answers a string containing the contents of the array as a sequence of printable characters. The string representation of each entry in the array appears in the result string in the same order as it occurs in the array and is separated from other entries by spaces. The entire array content is delimited by parentheses.
first	Answers the object in the first ordinal position within the array. An error results if the array is empty. MyArray at: 1 and MyArray first are equivalent statements.
includes:	Answers a Boolean (true or false) to indicate whether or not an object equal to argument1 is present within the receiver array. The test employed is equality--that is, whether the two compared objects have the same value. It is not absolute equality--that is, whether the two references point to exactly the same object in memory.
indexOf:	Answers the integer index into the array where the first object equal to argument1 appears. The value 0 is returned if the receiver does not include an object equal in value to argument1.
isEmpty	Answers a Boolean (true or false) indicating whether or not at least one object is stored within the receiver array.
isNil	Answers Boolean false.
last	Answers the object in the last ordinal position within the receiver array.
notEmpty	Answers a Boolean (true or false) indicating whether or not the array contains no objects. The notEmpty method always answers the opposite of the isEmpty method.
occurrencesOf:	Answers the number of times the object argument1 occurs in the receiver array.
size	Answers the integral number of cells in the receiver array when it was created regardless of the number of objects stored in the array.

Attribute Definition

An attribute definition is the particular schema information used to pattern an attribute whenever an element or relationship is created. It contains information from the schema regarding a specific attribute of a particular class or relation, including its name, alias, description, type, initial value, etc. You can avoid hard coding this information into your scripts and create report templates with additional flexibility.

<	Answers a Boolean indicating whether or not the receiver is less than argument1. Attribute definitions are ordered alphabetically by alias, if any, and by name otherwise.
<=	Answers a Boolean indicating whether or not the receiver is less than or equal to argument1. Attribute definitions are ordered alphabetically by alias, if any, and by name otherwise.
<>	Answers a Boolean indicating whether or not the receiver is not equal to argument1.
=	Answers a Boolean indicating whether or not the receiver is equal to argument1.
>	Answers a Boolean indicating whether or not the receiver is greater than argument1. Attribute definitions are ordered alphabetically by alias, if any, and by name otherwise.
>=	Answers a Boolean indicating whether or not the receiver is greater than or equal to argument1. Attribute definitions are ordered alphabetically by alias, if any, and by name otherwise.
displayString	Answers a string that contains the alias for the attribute. If no alias has been defined for the attribute, the attribute definition name is returned as a string.
isNil	Answers Boolean false.
name	Answers a symbol that is the unique identifier for the receiver attribute definition.

Bag

A bag is an unordered collection of objects like a set except that duplicate objects may exist and are maintained. Only the first object with a particular value is actually stored; thereafter, a count is associated with each object having a distinct value. If you iterate over the objects in the bag, each object will appear the number of times indicated by the object and its counter.

add:	Adds the object argument1 to the receiver bag. If another object of equal value is already present in the bag, argument1 will not be physically added. A bag, however, will remember that another object of equal value has been added, and when iterating over the collection that value will be represented the same number of times as it was added to the bag. Answers argument1.
addAll:	Iterates through the objects in argument1 and adds each, in turn, to the bag. Argument1 must be either an array, bag, dictionary, ordered collection, set, sorted collection, string, or symbol. If argument1 is a dictionary, then this operator iterates over the values stored in the dictionary, not the keys. If argument1 is a string or symbol, then the added objects will all be characters. If argument1 is a bag, set or dictionary, then the order of iteration is unspecified. Otherwise, this operator iterates over the objects in argument1 in increasing index order. The position of the added objects is unspecified. Answers argument1.
asArray	Answers an array containing all the objects in the bag. The same object may be represented multiple times in the answer, depending on how many times objects of equal value were added to the bag.
asOrderedCollection	Answers an ordered collection containing all the objects in the bag. The order of the objects in the ordered collection is arbitrary. The same object may appear multiple times in the ordered collection, depending on the number of times objects with equal value were added to the receiver.
asSet	Answers a set that is a copy of the receiver converted to a set. The order of the objects in the set is arbitrary. A set only maintains a single object with any given value, so if the bag contains multiple objects with the same value or multiple references to the same object, the multiple objects or references are lost in the conversion process.
asSortedCollection	Answers a sorted collection containing all the objects in the bag. The objects in the collection are sorted 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). The same object may appear multiple times in the sorted collection, depending on the number of times objects with equal value were added to the bag.
displayString	Answers a string containing the contents of the bag as a sequence of printable characters. The string representations of individual bag entries appear in an arbitrary order in the result and are separated by spaces. The string representation for an object will appear as many times in the result as it occurs in the bag. The bag representation is delimited by parentheses and preceded by the word Bag.
includes:	Answers a Boolean (true or false) to indicate whether or not an object equal to argument1 is present within the bag.
isNil	Answers Boolean false.
occurrencesOf:	Answers the number of times the object argument1 occurs in the bag.

remove:	Decrements the count associated with the value of argument1 by one and reduces the size of the receiver bag by one. If the count reaches zero, then the object is removed from the receiver bag. An error results if the receiver does not include argument1. Answers argument1.
remove:ifAbsent:	Decrements the count associated with the value of argument1 by one, reduces the size of the receiver bag by one, and answers argument1. If the count reaches zero, then the object is removed from the receiver bag. If the receiver does not contain an object equal to argument1, then and only then is the expression contained in argument2 evaluated and the result returned. Argument2 must be a block, i.e., an expression enclosed in square brackets.

Boolean

A Boolean can have only one of two possible values representing either logical truth or falsehood. Comparison operations, such as equality (=), result in a Boolean condition. A Boolean condition test determines which of several possible courses of action to follow in a script. Logical operations, such as alternation (or:), combine multiple Boolean objects into a single Boolean object representing a complex condition.

and:	Answers a Boolean true if both the receiver and argument1 are true; otherwise, answers Boolean false. Argument1 must be either a Boolean.
displayString	Answers a string representing the Boolean receiver. Answers either the string ‘true’ (without the quotation marks) or the string ‘false’ as appropriate.
isNil	Answers Boolean false.
not	Answers the inverse of the receiver Boolean. If the receiver is true, answers Boolean false; otherwise, answers Boolean true.
or:	Answers a Boolean true if either the receiver or argument1 is true; otherwise, answers false. Argument1 must be a Boolean.
xor:	Answers Boolean true if exactly one of the receiver and argument1 is true; otherwise, answers Boolean false. Argument1 must be a Boolean.

Character

The character class is the representation of a single keyboard character, or anything stored in a single byte, including special symbols and non-printing characters.

A shorthand syntax exists to represent a character object in the COREscript expression language. Simply type a dollar sign ($) followed by the desired character, which can be anything, even a space or another dollar sign. For example, an object representing the character C in a COREscript expression, is created using $C.

<	Answers a Boolean indicating whether or not the receiver is less than argument1. Characters are ordered by their ASCII values.
<=	Answers a Boolean indicating whether or not the receiver is less than or equal to argument1. Characters are ordered by their ASCII values.
<>	Answers a Boolean indicating whether or not the receiver is not equal to argument1.
=	Answers a Boolean indicating whether or not the receiver is equal to argument1.
>	Answers a Boolean indicating whether or not the receiver is greater than argument1. Characters are ordered by their ASCII values.
>=	Answers a Boolean indicating whether or not the receiver is greater than or equal to argument1. Characters are ordered by their ASCII values
asciiValue	Answers the integer value that is the ASCII representation of the character.
asLowercase	Answers the lower case for an alphabetic character. If the receiver character is not alphabetic, answers the receiver.
asUppercase	Answers the upper case for an alphabetic character. If the receiver character is not alphabetic, answers the receiver.
decrement	Answers the character with the next lower ASCII value than that of the receiver. If the receiver is the character a, the answer is also a. The case (upper or lower) of the answer matches the case of the receiver.
displayString	Answers a string containing the single receiver character.
increment	Answers the character with the next higher ASCII value than that of the receiver. If the receiver is the character z, the answer is also z. The case (upper or lower) of the answer matches the case of the receiver.
isAlphaNumeric	Answers a Boolean indicating whether or not the receiver is either an alphabetic or numeric character.
isDigit	Answers a Boolean indicating whether or not the receiver is a numeric character.
isLetter	Answers a Boolean indicating whether or not the receiver is an alphabetic character.
isLowercase	Answers a Boolean indicating whether or not the receiver is a lower case alphabetic character.
isNil	Answers Boolean false.
isUppercase	Answers a Boolean indicating whether or not the receiver is an upper case alphabetic character.

Class

Class defines or categorizes a particular type or kind of object. Every element in a CORE database is an instance of some class. All the elements in a given class have the same qualities, parameters, or attributes, though each element may possess a unique state, or set of values for those attributes. If objects were cookies, then classes would be the cookie-cutters, i.e., templates for cookies. The element class also determines the way the element behaves and relates to other elements. A class may have children, called subclasses, which may have additional attributes or further refine the behavior of instances of the parent class

A class is itself an object containing information from the schema, including its own name, alias, description, subclasses, and information common to all instances of the class, including the available attributes, relations, and target classes. The advantages of accessing classes are that you avoid hard coding this information into your scripts and create report templates that offer additional flexibility.

<	Answers a Boolean indicating whether or not the receiver is less than argument1. Classes are ordered alphabetically by the values of their alias properties, if any, and their names otherwise.
<=	Answers a Boolean indicating whether or not the receiver is less than or equal to argument1. Classes are ordered alphabetically by the values of their alias properties, if any, and their names otherwise.
<>	Answers a Boolean indicating whether or not the receiver is not equal to argument1.
=	Answers a Boolean indicating whether or not the receiver is equal to argument1.
>	Answers a Boolean indicating whether or not the receiver is greater than argument1. Classes are ordered alphabetically by the values of their alias properties, if any, and their names otherwise.
>=	Answers a Boolean indicating whether or not the receiver is greater than or equal to argument1. Classes are ordered alphabetically by the values of their alias properties, if any, and their names otherwise.
displayString	Answers a string that contains the alias for the class. If no alias has been defined for the class, the class name is returned as a string.
elementAt:ifAbsent:	Answer the element in the receiver class with the name given by symbol argument1. If no element exists with the specified name, then the result of evaluating argument2 is answered. Argument2 must be a block, i.e., an expression enclosed in square brackets.
isNil	Answers Boolean false.
name	Answers a symbol that is the unique identifier for the receiver Class.

Date

A date is a value that represents a calendar day.

<	Answers a Boolean indicating whether or not the receiver is less than argument1. Argument1 must be a date.
<=	Answers a Boolean indicating whether or not the receiver is less than or equal to argument1. Argument1 must be a date.
<>	Answers a Boolean indicating whether or not the receiver is not equal to argument1.
=	Answers a Boolean indicating whether or not the receiver is equal to argument1.
>	Answers a Boolean indicating whether or not the receiver is greater than to argument1. Argument1 must be a date.
>=	Answers a Boolean indicating whether or not the receiver is greater than or equal to argument1. Argument1 must be a date.
addDays:	Answers a new date that is later than the receiver by the number of days designated by argument1. Argument1 must be an integer.
dayName	Answers a symbol that is the day of the week for the receiver date.
dayOfMonth	Answers an integer that is the day of the month portion of the receiver.
dayOfYear	Answers an integer that is the number of days into the calendar year that corresponds to the receiver.
displayString	Answers a string containing the date as a sequence of printable characters. CORE Preferences control whether a short or long format is returned. The order of the date parts and, for the short format, the delimiter is specified in the Windows system settings.
isNil	Answers Boolean false.
longDisplayString	Answers a string containing the date in long format. The order of the date is specified in the Windows system settings.
longDMYDisplayString	Answers a string containing the date in long format in the order of day month year.
longMDYDisplayString	Answers a string containing the date in long format in the order of month day year.
longYMDDisplayString	Answers a string containing the date in long format in the order of year month day.
monthIndex	Answers an integer containing the number of the month in the receiver.
monthName	Answers a symbol that is the name of the month in the receiver.
year	Answers an integer that is the year in the receiver date.
shortDisplayString	Answers a string containing the date in short format. The order and style of the date fields and the delimiter are specified in the Windows system settings.
shortDMYDisplayString	Answers a string containing the date in short format in the order day month year. The treatment of one digit days and months, the delimiter used, and whether the year is two or four digits are specified in the Windows system settings.
shortMDYDisplayString	Answers a string containing the date in short format in the order month day year. The treatment of one digit days and months, the delimiter used, and whether the year is two or four digits are specified in the Windows system settings.
shortYMDDisplayString	Answers a string containing the date in short format in the order year month day. The treatment of one digit days and months, the delimiter used, and whether the year is two or four digits are specified in the Windows system settings.

subtractDate:	Answers an integer that is the number of days determined by subtracting argument1 from the receiver. Argument1 must be a date.
subtractDays:	Answers a new date that differs from the receiver by the number of days designated by argument1. Argument1 must be an integer.

Dictionary

A dictionary is a collection of objects, each of which is accessed or stored using a unique key. This is similar to finding a term definition in a standard language dictionary by looking up the term; however, only one value at a time may be associated with a given key.

asArray	Answers an array containing all the objects in the receiver dictionary. The array contains the values stored in the dictionary, not the keys whereby the values were accessed.
asOrderedCollection	Answers an ordered collection that contains the receiver dictionary values. The order of the objects in the ordered collection is arbitrary.
asSet	Answers a set that consists of the objects (values) stored in the receiver dictionary. A set only maintains a single object with any given value, so if the dictionary contains multiple objects with the same value, the duplicate objects are lost in the conversion process.
asSortedCollection	Answers a sorted collection that contains all objects stored in the receiver dictionary. The objects in the new collection are sorted using a simple comparison of the objects in the collection as the sort criterion (but each type of object handles the comparison in its own way). The objects in the answered collection consist only of the values stored in the dictionary, not the keys whereby the values were accessed.
at:	Answers the object stored in the receiver dictionary using the key argument1. Argument1 can be any type of object. Argument1 is not a key in the dictionary, an error results.
at:ifAbsent:	Answers the object stored in the receiver dictionary using the key argument1. If argument1 is not a key for the receiver dictionary, then and only then is the expression contained in argument2 evaluated and the resulting value returned. Argument1 can be any type of object. Argument2 must be a block, i.e., an expression enclosed in square brackets.
at:put:	Stores argument2 at the key argument1in the receiver dictionary. Both argument1 and argument2 can be any type object. Answers argument2.
displayString	Answers a string containing the contents of the receiver dictionary as a sequence of printable characters. The string representations of individual dictionary entries appear in an arbitrary order in the result and are separated by spaces. The dictionary contents in string representation form is delimited by parentheses and preceded by the word Dictionary.
includes:	Answers a Boolean indicating whether or not the receiver dictionary contains an object that is equal to argument1. The values stored in the dictionary are searched, not the keys whereby the values are accessed. Argument1 can be any type of object.
includesKey:	Answers a Boolean indicating whether or not a key equal to argument1 is present within the receiver dictionary. Only the keys are searched, not the values stored in the dictionary. Argument1 can be any type of object.
isEmpty	Answers a Boolean (true or false) indicating whether or not at least one object is stored within the receiver dictionary.
isNil	Answers Boolean false.
keys	Answers a set containing all keys in the receiver dictionary.

keyAtValue:	Searches the receiver dictionary for the first occurrence of argument1 and returns the associated key. The order of search is arbitrary. If argument1 is not present in the dictionary, nil is returned. Argument1 can be any type of object.
notEmpty	Answers a Boolean (true or false) indicating whether or not the receiver dictionary contains no objects. The notEmpty method always answers the opposite of the isEmpty method.
occurrencesOf:	Answers the number of times the object argument1 occurs in the receiver dictionary. Argument1 can be any type of object.
removeKey:	Removes the key argument1 from the receiver dictionary along with its associated definition or value. Argument1 need not be absolutely equal to (that is, exactly the same object as) the key to be removed. It need only be of equal value. An error results if no key equal to argument1 is found within the dictionary. Argument1 can be any type of object. Answers argument1.
removeKey:ifAbsent:	Removes the key equal to argument1 from the receiver dictionary along with its associated definition or value. Argument1 need not be absolutely equal to (that is, exactly the same object as) the key to be removed. It need only be of equal value. If argument1 does not exist in the receiver dictionary, then and only then is the expression contained in argument2 evaluated. Argument2 must be a block, i.e., an expression enclosed in square brackets. Argument1 can be any type of object. Answers argument1 if present in the dictionary; otherwise, answers the result of evaluating argument2.
size	Answers the number of keys in the receiver dictionary with which values are currently associated.
values	Answers a bag containing all values in the receiver dictionary.

Element

An element is an object containing information pertaining to a specific database element, including its name and other attribute values such as abbreviation, description, etc., as specified by its class definition.

<	Answers a Boolean indicating whether or not the receiver is less than argument1. Elements are ordered by class and then alphabetically by name.
<=	Answers a Boolean indicating whether or not the receiver is less than or equal to argument1. Elements are ordered by class and then alphabetically by name.
<>	Answers a Boolean indicating whether or not the receiver is not equal to argument1.
=	Answers a Boolean indicating whether or not the receiver is equal to argument1.
>	Answers a Boolean indicating whether or not the receiver is greater than argument1. Elements are ordered by class and then alphabetically by name.
>=	Answers a Boolean indicating whether or not the receiver is greater than or equal to argument1. Elements are ordered by class and then alphabetically by name.
attributeValueAt:	Answers the value of the attribute specified by argument1 for the receiver element. If the attribute type is collection, then the answer is an ordered collection. If the attribute type is computed or enumerated, the data type of the answer is the value type of the attribute. If the attribute type is text, the answer is a string. Otherwise, the data type of the answer is the same as the attribute type. Returns nil if either the attribute does not exist or the user has insufficient permissions to read the attribute value.
attributeValueAt:ifAbsent:	Answers the value of the receiver element attribute specified by argument1. If the user has insufficient permissions to read the attribute value, nil is returned. If the attribute does not exist in the element, then and only then is the expression in argument2 evaluated and the result returned. Argument 2 must be a block, i.e., an expression enclosed in square brackets.
children	Answers a set containing all the elements 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 schema for the class of the receiver element. There may be no parent-child relation for the class of the receiver. The children elements may be of multiple classes.
class	Answers the class object that defines the receiver element and of which the receiver is an instance. Note: The actual class object is answered, not the class name. Use the name message to obtain the name of the class.
currentStructure	Answers a network that is the structure for the receiver element. The receiver must be a processing unit such as a function or activity.
displayString	Answers the name of the receiver element as a string.
folders	Answers the set of folders that contain this element.
isNil	Answers Boolean false.

name	Answers a symbol that is the unique identifier for the receiver element. Returns the symbol ‘Insufficient privileges to access’ if the user has insufficient permissions to read the element.
number	Answers the hierarchical number that is the value of the number attribute of the receiver element. Returns nil if either the number is unassigned or the user has insufficient permissions to read the attribute value.
parents	Answers a set containing all the elements that are sources (vice targets) of parent-child relationships with the receiver element; i.e., the receiver is the target of the relationships. Parent-child relations established in the schema with the class of the receiver element as the target class define the parent-child relationships.
relationships:	Answers a set containing all relationship objects of the relation specified by argument1 that have been defined with the receiver element as the source. This set can subsequently be iterated over to obtain from each relationship its target and the values of any attributes defined for it. Argument1 must be a symbol that is the name of a valid relation defined on the element’s class.
relationships:targetClass:	Answers a set containing all relationship objects of the relation specified by argument1 that have been defined with the receiver element as the source and an element of the class specified by argument2 as the target. Argument1 must be a symbol that is the name of a valid relation defined on the element’s class. Argument2 must be a class, not a class name.
relationshipsAt:ifAbsent:	Answers a set containing all relationship objects of the relation type specified by argument1 that have been defined with the receiver element as their source. If no such relationships have been established (or if argument1 does not specify a valid relation for the receiver’s class), then and only then is the expression in argument2 evaluated and the result returned. Argument1 must be a symbol that is the name of a valid relation defined on the element’s class. Argument2 must be a block, i.e., an expression enclosed in square brackets.
targets:	Answers a set containing all elements that are targets of relationships defined by relation argument1 and having the receiver element as the source. The targets can be of multiple classes. Argument1 must be a symbol that is the name of a valid relation defined for the class of the receiver.
targets:targetClass:	Answers a set containing all elements that are of class argument2 and that are targets of relationships defined by relation argument1 and having the receiver element as the source. Argument1 must be a symbol that is the name of a valid relation defined for the class of the receiver. Argument2 must be a valid target class defined for the relation named by argument1 in the class of the receiver element. Note that argument2 is not the name of a class but the class itself.

Facility

A facility is a specialized subset of all available classes that are closely related or that collaboratively support a particular system engineering process (e.g., verification phase). Facilities are used by the CORE user interface to refine the amount of information viewed at one time.

An object representing a facility contains information as specified by the CORE schema, including the name, alias, and description of the facility, as well as its relevant classes.

<	Answers a Boolean indicating whether or not the receiver is less than argument1. Facilities are ordered alphabetically by alias, if any, and by name otherwise.
<=	Answers a Boolean indicating whether or not the receiver is less than or equal to argument1. Facilities are ordered alphabetically by alias, if any, and by name otherwise.
<>	Answers a Boolean indicating whether or not the receiver is not equal to argument1.
=	Answers a Boolean indicating whether or not the receiver is equal to argument1.
>	Answers a Boolean indicating whether or not the receiver is greater than argument1. Facilities are ordered alphabetically by alias, if any, and by name otherwise.
>=	Answers a Boolean indicating whether or not the receiver is greater than or equal to argument1. Facilities are ordered alphabetically by alias, if any, and by name otherwise.
allDatabaseClasses	Answers an ordered collection containing all of the concrete class objects in the receiver facility. The collection is ordered alphabetically by class alias, if any, and name otherwise.
displayString	Answers the alias, if any, or the name of the receiver as a string.
isNil	Answers Boolean false.
name	Answers a symbol that is the unique identifier for the Facility.

Float

A float is an object that represents the value of a floating-point number.

Any numeric value with a decimal point that is preceded by at least one digit and followed by at least one digit (e.g., 0.0) is automatically instantiated as a float in the COREscript expression language.

+	Answers the sum of the receiver and argument1. Argument1 must be an integer, float, or fraction. The answer is a float.
-	Answers the difference between the receiver and argument1. Argument1 must be an integer, float, or fraction. The answer is a float.
*	Answers the product of the receiver and argument1. Argument1 must be an integer, float or fraction. The answer is a float.
/	Answers the quotient of the receiver and argument1. Argument1 must be an integer, float, or fraction. The answer is a float.
<	Answers a Boolean indicating whether or not the receiver is less than argument1.
<=	Answers a Boolean indicating whether or not the receiver is less than or equal to argument1.
<>	Answers a Boolean indicating whether or not the receiver is not equal to argument1.
=	Answers a Boolean indicating whether or not the receiver is equal to argument1.
>	Answers a Boolean indicating whether or not the receiver is greater than argument1.
>=	Answers a Boolean indicating whether or not the receiver is greater than or equal to argument1.
abs	Answers the absolute value of the receiver.
displayString	Answers a string containing the receiver value as a sequence of printable characters.
div:	Answers an integer that is the quotient of the receiver and argument1 rounded toward negative infinity. Argument1 must be an integer, float, or fraction.
isNil	Answers Boolean false.
max:	Answers the greater of the receiver and argument1. Argument1 must be an integer, float, or fraction.
min:	Answers the lesser of the receiver and argument1. Argument1 must be an integer, float, or fraction.
mod:	Answers an integer that is the remainder of the receiver divided by argument1. Argument1 must be an integer, float, or fraction.
negated	Answers a new float whose sign is the opposite of the receiver.
raisedTo:	Answers a new float that is the value of the receiver raised to the power of argument1. Argument1 must be an integer, float, or fraction.
rounded	Answers an integer that is the receiver rounded up or down depending on the fractional part of the number. Fractions 0.5 and higher are rounded up. All others are rounded down.
sqrt	Answers the square root of the receiver. The result is always a float.
truncated	Answers an integer that is the integral part of the receiver.

Folder

Folders organize elements in smaller groupings under a class. Folders are hierarchical in nature.

allElements	Answers a set of the elements that belong to the receiver and its subfolders.
allParentFolders	Answers an ordered collection of the ancestors of the receiver folder. The folders are ordered hierarchically starting with the root folder.
allSubfolders	Answers an ordered collection of all folders below the receiver. Unlike the subfolders message, which only answers one level down in the hierarchy, the allSubfolders answers all folders in the hierarchy. The order in the collection is hierarchical depth first, with siblings ordered alphabetically.
displayString	Answers the name of the folder as a string.
localElements	Answers a set of the elements that belong to the folder.
name	Answers a symbol that is the name of the folder.
numberOfLocalElements	Answers the integral number of elements in the receiver folder.
pathName	Answers a symbol made up of the folder names of the receiver folder's ancestors plus the receiver folder name. The names appear in hierarchical order starting with the root folder and are each separated by a single slash.
subfolders	Answers a set containing the folders that are directly below the receiver.
totalNumberOfElements	Answers the integral number of elements in the entire hierarchy of the receiver folder.

Formatted Text

Formatted text is an object that stores text along with any formatting. The methods for formatted text objects operate on the ASCII representation of the formatted text (i.e., a string). The string contains characters in the normal text font; any symbol or wingding characters are converted to their normal text equivalent.

<	Answers a Boolean indicating whether or not the receiver is less than argument1. Formatted text objects are ordered by the ASCII values of their index position characters, with index position decreasing in significance from left to right. If one formatted text object is equal in the leftmost positions of another, the shorter is less than the longer.
<=	Answers a Boolean indicating whether or not the receiver is less than or equal to argument1. Formatted text objects are ordered by the ASCII values of their index position characters, with index position decreasing in significance from left to right. If one formatted text object is equal in the leftmost positions of another, the shorter is less than the longer.
<>	Answers a Boolean indicating whether or not the receiver is not equal to argument1.
=	Answers a Boolean indicating whether or not the receiver is equal to argument1.
>	Answers a Boolean indicating whether or not the receiver is greater than argument1. Formatted text objects are ordered by the ASCII values of their index position characters, with index position decreasing in significance from left to right. If one formatted text object is equal in the leftmost positions of another, the shorter is less than the longer.
>=	Answers a Boolean indicating whether or not the receiver is greater than or equal to argument1. Formatted text objects are ordered by the ASCII values of their index position characters, with index position decreasing in significance from left to right. If one formatted text object is equal in the leftmost positions of another, the shorter is less than the longer.
asArray	Answers an array containing all the characters in the receiver formatted text in the same order that they occur in the formatted text.
asArrayOfSubstrings	Parses the receiver into substrings breaking the formatted text at a blank or sequence of blanks. Answers an array containing the substrings. The substrings contain no blanks and are in the order that they occur in the receiver.
asArrayOfSubstringsSeparatedBy:	Parses the receiver into substrings breaking the formatted text at each occurrence of the argument1 character. Answers an array containing the substrings. The substrings do not contain argument1 and are in the order that they occur in the receiver. The new array will include empty substrings if argument1 occurs in a sequence in the receiver.
asASCII	Answers a string containing the ASCII representation of the receiver. Note that any symbol or wingding font characters are converted to their normal text equivalent.
asBoolean	Answers Boolean true if the receiver consists of the character sequence ‘true’. Answers Boolean false otherwise.

asDate	Answers a date that is parsed and converted from the character sequence of the formatted text. All preliminary white space is ignored. If the first character of the string is alphabetic, the date is assumed to be in month-day-year format. Otherwise, this operator expects the date format specified in the Preferences dialog. Any non-alphanumeric characters may separate the components of the date. If the month name or abbreviation is used, it need not be separated from the day or year. An error results if any characters or white space remain after the date has been parsed. The year may be specified as an integer. If the year is not specified, the current year is supplied by default. The month may be specified as a full name or a 3-letter abbreviation in mixed case, or an integer. An error results if the month value falls outside the range of valid months in a year. The day may be specified as an integer. An error results if the day value falls outside the range of valid days in the specified month for the specified year.
asFloat	Answers a float the value of which is parsed and converted from the receiver formatted text. The formatted text character sequence may consist of any or all of: an initial sign (+ or -), a sequence of digits comprising the magnitude of the integral portion of the number, a decimal point (.) followed by a sequence of digits comprising the magnitude of the fractional part of the number, an uppercase E or lowercase e, another sign, and a sequence of digits comprising the magnitude of the multiplicative factor (as a power of 10). If a sign is not specified, a positive magnitude is assumed. If digits are not specified for any portion, that portion defaults to 0. An error results if the string is empty or if no exponent is specified after an E or an e.
asHierarchicalNumber	Answers a hierarchical number whose value is parsed and converted from the character sequence of the receiver string. The hierarchical number consists of one or more cells separated by periods (.). Each cell can consist either of one or more letters, or one or more digits.
asHTML	Answers a string containing the HTML representation of the receiver including all HTML font and styling commands.
asInteger	Answers an integer whose value is parsed and converted from the character sequence of the receiver formatted text. The integer may consist of an optional sign (+ or -) and/or a sequence of digits. If no digits are specified, 0 is assumed. Returns 0 if the string is empty or begins with a non-numeric character.
asLowercase	Answers a string that contains the sequence of characters in the formatted text of the receiver with all uppercase alphabetic characters converted to lowercase.
asOrderedCollection	Answers an ordered collection containing all the characters in the receiver formatted text in the same order that they occur in the formatted text.
asRTF	Answers a string containing the RTF representation of the receiver including all RTF font and styling tags.
asSet	Answers a set containing the unique characters in the receiver.
asSortedCollection	Answers a sorted collection containing the characters in the receiver sorted by ASCII number.
asSymbol	Answers a symbol containing the same characters as the string.

asTime	Answers a time with a value that is parsed and converted from the character sequence of the receiver string. The string may be specified in either of the following formats: hh:mm, hh:mm:ss. Each component may be either 1 or 2 digits. If the seconds are not specified, 0 is assumed. If the hours are less than 12, the string may optionally be suffixed by AM or PM in mixed case. An error results if any component is out of range.
asTimeStamp	Answers a timestamp the value of which is parsed and converted from the character sequence of the receiver. The receiver can consist of a date (as specified above for asDate) followed by white space, the word at, more white space, and a time (as specified above for asTime). If the time is omitted, 00:00:00 is assumed.
asUppercase	Answers a string that is a copy of the receiver with all lowercase alphabetic characters converted to uppercase.
at:	Answers the character at the ordinal position in the receiver indicated by argument1. Argument1 must be an integer. An error results if the index is outside the bounds of the string.
copyFrom:to:	Answers a new string containing every character in the receiver string from ordinal position argument1 through ordinal position argument2. Argument1 and argument2 must be integers. An error results if either argument1 or argument2 represents a position outside the bounds of the string.
decrement	Answers a new string that is a copy of the receiver string with the last character decremented. If the string ends in a substring of one or more ‘a’s, then the final 'a's are changed to ‘z’s and the last non-a is decremented. If the string consists entirely of ‘a’s, then the last ‘a’ is omitted and the rest are changed to ‘z’s. If the string consists of a single a, then the answer is also a. The case (upper or lower) of each character in the answer matches the case of the corresponding character of the receiver.
displayString	Answers a string that is a copy of the receiver text with all formatting removed.
doubleEmbedded:	Answers a new string that is a copy of the receiver in which all occurrences of the character specified by argument1 have been doubled. For example, ‘50% growth’ doubleEmbedded: $% answers ‘50%% growth’.
first	Answers the first character in the receiver formatted text. An error results if the receiver is empty.
includes:	Answers a Boolean (true or false) indicating whether or not the character argument1 is contained in the receiver string. An error results if the receiver is empty.
includesString:	Answers a Boolean (true or false) indicating whether or not the sequence of characters in argument1 was found in the receiver. An asterisk (*) within argument1 can be matched by any substring of 1 or more characters in the string or even by an empty string. For example, includesString: will find re*d within red, reed, and received.
increment	Answers a new string that is a copy of the receiver string with the last character incremented. If the string ends in a substring of one or more ‘z’s, then the final 'z's are changed to ‘a’s and the last non-z is incremented. If the string consists entirely of ‘z’s, an ‘a’ appended to the end of the string in addition to changing the ‘z’s to ‘a’s. The case (upper or lower) of each character in the answer matches the case of the corresponding character in the receiver.

indexOf:	Answers the integer index of the first occurrence of character argument1 in the receiver formatted text. Returns 0 if the receiver does not include the character. An error results if the receiver is empty.
isEmpty	Answers a Boolean (true or false) indicating whether or not at least one character is stored within the receiver formatted text. That is, the equivalent string is at least one character long.
isNil	Answers Boolean true if the formatted text is empty. Otherwise, answers Boolean false
last	Answers the last character of the receiver string. An error results if the string is empty.
match:	Searches the receiver for the first occurrence of the sequence of characters in argument1. The search begins with the first character in the receiver. If there is a match, answers an array of size 2 where the first index position is the index of the first character of argument1 and the second index position contains the index of the last character of argument1 in the receiver. Answers the array (0 0) if there is no match. Wildcards are supported.
match:startingIndex:	Searches the receiver for the first occurrence of the sequence of characters in argument1. The search begins at the index into the receiver specified by argument2 rather than at the beginning of the receiver. Answers an array of size 2 where the first index position is the index of the first character of argument1 and the second index position is the index of the last character of argument1 in the receiver. Answers the array (0 0) if there is no match. Wildcards are supported.
notEmpty	Answers a Boolean (true or false) indicating whether or not the receiver string contains no characters. The notEmpty method always answers the opposite of the isEmpty method.
occurrencesOf:	Answers an integer that is the number of times argument1 occurs in the receiver string. Argument1 must be a character.
size	Answers an integer that is the number of characters in the receiver.
trimBlanks	Answers a new string that is a copy of the receiver string with any leading and trailing blanks, carriage returns, and tabs removed.

Fraction

A fraction is a ratio between two integers. The numerator and denominator values are stored separately so that all combinations with other fractions and integers using mathematical operators result in fractions and integers, eliminating floating-point error.

+	Answers the sum of the receiver and argument1. Argument1 can be an integer, float, or fraction. The answer will be a float if argument1 is a float. Otherwise, the answer is a fraction or integer.
-	Answers the difference between the receiver and argument1. Argument1 must be an integer, float, or fraction. The answer will be a fraction or integer if argument1 is an integer or fraction. Otherwise, the answer is a float.
*	Answers the product of the receiver and argument1. Argument1 can be an integer, float, or fraction. The answer will be a float if argument1 is a float. Otherwise, the answer is a fraction or integer.
/	Answers the quotient of the receiver and argument1. If argument1 is a float, then the answer is a float. Otherwise, the answer is a fraction or integer.
<	Answers a Boolean indicating whether or not the receiver is less than argument1.
<=	Answers a Boolean indicating whether or not the receiver is less than or equal to argument1.
<>	Answers a Boolean indicating whether or not the receiver is not equal to argument1.
=	Answers a Boolean indicating whether or not the receiver is equal to argument1.
>	Answers a Boolean indicating whether or not the receiver is greater than argument1.
>=	Answers a Boolean indicating whether or not the receiver is greater than or equal to argument1.
abs	Answers a new fraction that is the absolute value of the receiver.
asFloat	Answers the receiver value concerted to a float.
denominator	Answers the fraction denominator as an integer.
displayString	Answers a string consisting of the numerator value followed by a slash and the denominator value.
div:	Answers an integer that is the quotient of the receiver and argument1 rounded toward negative infinity.
isNil	Answers Boolean false.
max:	Answers the greater of the receiver and argument1. Argument1 must be an integer, float, or fraction.
min:	Answers the lesser of the receiver and argument1. Argument1 must be an integer, float, or fraction.
mod:	Answers an integer that is the remainder of the receiver divided by argument1.
negated	Answers a new fraction whose sign is the opposite of the receiver.
numerator	Answers the fraction numerator as an integer.
raisedTo:	Answers a float that is the value of the receiver raised to the power of argument1. Argument1 may be an integer, float, or fraction.
rounded	Answers an integer that is the receiver rounded up or down depending on the fractional part of the number. Fractions 0.5 and higher are rounded up. All others are rounded down.
sqrt	Answers a float that is the square root of the receiver.
truncated	Answers an integer that is the integral part of the receiver.

Hierarchical Number

A hierarchical number is a concatenation of one or more integers and/or alphabetic character sequences separated by periods and used to tag another object. Each period-delimited segment, which is termed a cell, from left to right indicates that the associated object is of an additional level of specificity or detail. Alphabetic characters and digits cannot be mixed at the same level in the same hierarchical number. By incrementing the value at one level of a hierarchical number you indicate sequencing at that level. Thus, the name of this data type is derived from the fact that it imparts hierarchical structure like that of a traditional outline.

<	Answers a Boolean indicating whether or not the receiver is less than argument1. Hierarchical numbers are ordered numerically or alphabetically by the values of their cells within a given level, with cells decreasing in significance from left to right and numeric cells preceding alphabetic cells. If one hierarchical number is equal to the leftmost cells of another, the shorter is less than the longer.
<=	Answers a Boolean indicating whether or not the receiver is less than or equal to argument1. Hierarchical numbers are ordered numerically or alphabetically by the values of their cells within a given level, with cells decreasing in significance from left to right and numeric cells preceding alphabetic cells. If one hierarchical number is equal to the leftmost cells of another, the shorter is less than the longer.
<>	Answers a Boolean indicating whether or not the receiver is not equal to argument1.
=	Answers a Boolean indicating whether or not the receiver is equal to argument1.
>	Answers a Boolean indicating whether or not the receiver is greater than argument1. Hierarchical numbers are ordered numerically or alphabetically by the values of their cells within a given level, with cells decreasing in significance from left to right and numeric cells preceding alphabetic cells. If one hierarchical number is equal to the leftmost cells of another, the shorter is less than the longer.
>=	Answers a Boolean indicating whether or not the receiver is greater than or equal to argument1. Hierarchical numbers are ordered numerically or alphabetically by the values of their cells within a given level, with cells decreasing in significance from left to right and numeric cells preceding alphabetic cells. If one hierarchical number is equal to the leftmost cells of another, the shorter is less than the longer.
cells	Answers an ordered collection containing the contents of each cell of the receiver in level order. Within the collection, alphabetic cells are converted to strings and numeric cells are converted to integers.
decrement	Answers a new hierarchical number that is a copy of the receiver with the value of the least significant (rightmost) cell decremented by 1 according to the rules for decrementing integers and strings.
decrementLevel	Answers a new hierarchical number that is a copy of the receiver with the least significant (rightmost) cell omitted.
displayString	Answers a string representation of the hierarchical number.
increment	Answers a new hierarchical number that is a copy of the receiver with the value of the least significant (rightmost) cell incremented by 1 according to the rules for incrementing integers and strings.
incrementAlphabeticLevel	Answers a new hierarchical number that is a copy of the receiver with an additional (least significant) cell of value a.
incrementLevel	Answers a new hierarchical number that is a copy of the receiver with

	an additional (least significant) cell of value 0. If the receiver only has a single cell and that cell is of value 0, then the answer is a copy of the receiver.
includesString:	Answers a Boolean indicating whether or not the sequence of characters in argument1 occurs in the receiver. An asterisk (*) within argument1 can be matched by any substring of 1 or more characters in the string or even by an empty string. For example, includesString: will find re*d within red, reed, and received.
isNil	Answers Boolean false.
level	Answers an integer that is the number of cells in the receiver hierarchical number. For example, the hierarchical number A.1.B.12 has four levels.
match:	Searches the receiver for the first occurrence of the sequence of characters in argument1. The search begins with the first character in the receiver. If there is a match, answers an array of size 2 where the first index position is the index of the first character of argument1 and the second index position contains the index of the last character of argument1 in the receiver. Answers the array (0 0) if there is no match. Wildcards are supported.
match:startingIndex:	Searches the receiver for the first occurrence of the sequence of characters in argument1. The search begins at the index into the receiver specified by argument2 rather than at the beginning of the receiver. Answers an array of size 2 where the first index position is the index of the first character of argument1 and the second index position is the index of the last character of argument1 in the receiver. Answers the array (0 0) if there is no match. Wildcards are supported.

Integer

An integer is an object that represents the value of an integral number.

Any sequence of digits with no decimal point is automatically instantiated as an integer object in the COREscript expression language.

+	Answers the sum of the receiver and argument1. Argument 1 must be an integer, fraction, or float. The answer will be the same type as argument1.
-	Answers the difference between the receiver and argument1. Argument 1 must be an integer, fraction, or float. The answer will be the same type as argument1.
*	Answers the product of the receiver and argument1. Argument1 must be an integer, float, or fraction. The answer will be the same type as argument1.
/	Answers the quotient of the receiver and argument1. If argument1 is a float, then the answer is a float. Otherwise the answer is a fraction or integer.
<	Answers a Boolean indicating whether or not the receiver is less than argument1.
<=	Answers a Boolean indicating whether or not the receiver is less than or equal to argument1.
<>	Answers a Boolean indicating whether or not the receiver is not equal to argument1.
=	Answers a Boolean indicating whether or not the receiver is equal to argument1.
>	Answers a Boolean indicating whether or not the receiver is greater than argument1.
>=	Answers a Boolean indicating whether or not the receiver is greater than or equal to argument1.
abs	Answers a new integer that is the absolute value of the receiver.
asCharacter	Answers a character that is the character for the receiver interpreted as an ASCII value. The receiver must >= 1 and <= 255.
decrement	Answers a new integer of value one less than the receiver value. If the receiver is <= 0, the receiver is returned.
displayString	Answers the receiver value as a string of characters.
div:	Answers an integer that is the quotient of the receiver and argument1 rounded toward negative infinity.
isNil	Answers Boolean false.
max:	Answers the greater of the receiver and argument1. Argument1 must be an integer, float, or fraction.
min:	Answers the lesser of the receiver and argument1. Argument1 must be an integer, float, or fraction.
mod:	Answers an integer that is the remainder of the receiver divided by argument1.
negated	Answers a new integer whose sign is the opposite of the receiver.
raisedTo:	Answers a float that is the value of the receiver raised to the power of argument1. Argument1 must be a float, fraction, or integer.
rounded	Answers the receiver.
sqrt	Answers a float that is the square root of the receiver.
truncated	Answers the receiver.

Ordered Collection

An ordered collection is a collection of objects in which the position of each object relative to the others matters, but not its absolute position.

add:	Adds the object argument1 to the receiver collection. The object is added at the end of the collection. Returns argument1.
addAll:	Iterates through the objects in collection argument1 and adds each, in turn, to the ordered collection. If argument1 is a dictionary, then this operator iterates over the values stored in the dictionary, not the keys. If argument1 is a bag, set or dictionary, then the order of iteration is unspecified. Otherwise, this operator iterates over the objects in collection argument1 in increasing index order. The added objects are appended in the existing order. Returns argument1.
addFirst:	Adds argument1 to the receiver ordered collection ahead of any objects already existing in the ordered collection. Returns argument1.
addLast:	Adds argument1 to the receiver ordered collection following any objects already existing in the ordered collection. Returns argument1.
asArray	Answers an array containing all the objects in the ordered collection. The objects in the new array are in the same order as they are in the ordered collection.
asSet	Answers a set that consists of the objects in the receiver. A set only maintains a single object with any given value, so if the ordered collection contains multiple objects with the same value, the duplicate objects are lost in the conversion process.
asSortedCollection	Answers a sorted collection that contains all objects in the receiver collection. The objects in the new collection are sorted 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).
at:	Answers the object stored in the receiver in the ordinal position given by argument1. Argument1 must be an integer. An error results if argument1 is outside the bounds of the receiver.
at:ifAbsent:	Answers the object stored in the receiver in the ordinal position given by argument1. If argument1 is outside the bounds of the receiver, then and only then is the expression contained in argument2 evaluated and the resulting value returned. Argument2 must be a block, i.e., an expression enclosed in square brackets.
at:put:	Stores argument2 in the receiver at the ordinal position indicated by argument1. Argument1 must be an integer. Answers argument2.
concat:	Answers a new ordered collection containing all objects contained in the receiver collection followed by all the objects in argument1. Argument1 must be an array, dictionary, ordered collection, set, sorted collection, string or a symbol.
copyFrom:to:	Answers a new ordered collection containing every object from the argument1 through the argument2 ordinal positions in the receiver ordered collection. Argument1 and argument2 must be integers. An error results if either argument1 or argument2 represents a position outside the bounds of the receiver collection.
displayString	Answers a string consisting of the string representation for each object in the ordered collection separated by a comma and space. Braces enclose the entire collection representation.
first	Answers the object in the first ordinal position within the receiver ordered collection. An error results if the ordered collection is empty.

includes:	Answers a Boolean (true or false) to indicate whether or not an object equal to argument1 is present within the receiver ordered collection.
isEmpty	Answer a Boolean (true or false) indicating whether or not at least one object is stored within the receiver ordered collection.
isNil	Answers Boolean false.
last	Answers the object in the last ordinal position within the receiver ordered collection. An error results if the ordered collection is empty.
notEmpty	Answers a Boolean (true or false) indicating whether or not the receiver ordered collection contains no objects. The notEmpty method always answers the opposite of the isEmpty method.
occurrencesOf:	Answers the number of times the object argument1 occurs in the receiver ordered collection.
remove:	Removes the first occurrence of argument1 from the receiver ordered collection and reduces the size of the collection by one. An error results if the ordered collection does not include argument1. Answers argument1.
remove:ifAbsent:	Removes an object equal to argument1 from the receiver collection, reduces its size by one, decrements the ordinal position of each object remaining in the collection by one, and returns argument1. If the collection contains more than one instance of the object, only the first occurrence of argument1 is removed. If the object is not contained in the collection, then and only then is the expression contained in argument2 evaluated and the result returned. Argument2 must be a block, i.e., an expression enclosed in square brackets.
removeFirst	Removes the first object in the receiver ordered collection, reduces its size by one, and decrements the ordinal position of each object remaining in the collection by one. An error results if the collection is empty. Answers the first object.
removeLast	Removes the last object in the receiver ordered collection and reduces its size by one. An error results if the collection is empty. Answers the last object.
size	Answers an integer that is the number of objects contained in the collection.

ReferenceSpec

A referenceSpec is a data type that represents a resource (file, website, email address, etc) outside of CORE. The specification consists of a resource reference – either relative to the base path specified in the project or absolute – and an optional display name to describe the resource being referenced.

asHTML	Answers a hyperlink for the reference. If the display name is not nil, the hyperlink is in the form *absolute reference*. Otherwise, the hyperlink is in the form *absolute reference*.
displayName	Answers the descriptive name (either a string or nil) provided for the external resource.
displayString	Answers a string that contains an absolute external reference. This absolute reference is suitable for conversion to a hyperlink within a document.
isNil	Answers Boolean false.
reference	Answers a string containing the reference provided for the external resource. This reference can be either relative to the base path specified in the project or absolute, depending upon what has been provided by the user. To obtain an absolute reference in all cases, use the displayString message.

Relation

A relation is the schema information that serves as the pattern or template for a relationship. The association of a relation to a relationship is analogous to the association that a class has to an element. Just as a class defines the attributes common to all elements of that class, so a relation determines the attributes common to all relationships that are instances of that relation.

<	Answers a Boolean indicating whether or not the receiver is less than argument1. Relations are ordered alphabetically by alias, if any, and by name otherwise.
<=	Answers a Boolean indicating whether or not the receiver is less than or equal to argument1. Relations are ordered alphabetically by alias, if any, and by name otherwise.
<>	Answers a Boolean indicating whether or not the receiver is not equal to argument1.
=	Answers a Boolean indicating whether or not the receiver is equal to argument1.
>	Answers a Boolean indicating whether or not the receiver is greater than argument1. Relations are ordered alphabetically by alias, if any, and by name otherwise.
>=	Answers a Boolean indicating whether or not the receiver is greater than or equal to argument1. Relations are ordered alphabetically by alias, if any, and by name otherwise.
displayString	Answers a string that contains the alias for the relation. If no alias has been defined for the relation, the relation name is returned as a string.
isNil	Answers Boolean false.
name	Answers a symbol that is the unique identifier for the Relation.

Relationship

A relationship links two elements just like a verb links a subject noun with an object noun. An object representing a relationship contains information about the relationship such as the subject and target elements, the relationship name and any other attribute values defined by its relation.

attributeValueAt:	Answers the value of the attribute specified by argument1 for the receiver relationship. If the attribute type is collection, then the answer is an ordered collection. If the attribute type is computed or enumerated, the data type of the answer is the value type of the attribute. If the attribute type is text, the answer is a string. Otherwise, the data type of the answer is the same as the attribute type. Returns nil if the attribute has not been assigned a value. Argument1 must be a symbol that is the name of a valid attribute defined for the receiver's class.
attributeValueAt:ifAbsent:	Answers the value of the attribute specified by argument1 for the receiver relationship. If the attribute type is collection, then the answer is an ordered collection. If the attribute type is computed or enumerated, the data type of the answer is the value type of the attribute. If the attribute type is text, the answer is a string. Otherwise, the data type of the answer is the same as the attribute type. If the attribute does not exist for the relationship, then and only then is the expression in argument2 evaluated and the results returned. Argument2 must be a block, i.e., an expression enclosed in square brackets.
complementRelativeTo:	For the receiver relationship, answers the complement relation definition for the relationship direction in which argument1 is the subject element.
definitionRelativeTo:	For the receiver relationship, answers the relation definition for the relationship direction in which argument1 is the subject element.
displayString	Answers a string that consists of the subject; the relation alias, if any, or name; and the target separated by spaces. The subject and target representations consist of the class alias, if any, or name and the element name. The set of names is enclosed in parentheses and preceded by the word Relationship.
isNil	Answers Boolean false.
source	Answers the element that is on the source side of the receiver relationship where the source element is assigned when the relationship is established. For example, if a Requirement is linked to a Function, the Requirement is the source element. Conversely, if a Function is linked to a Requirement, the Function is the source element.
target	Answers the element that is on the target side of the receiver relationship where the target element is assigned when the relationship is established. For example, if a Requirement is linked to a Function, the Function is the source element. Conversely, if a Function is linked to a Requirement, the Requirement is the source element.
traverse:	Answers the element that is on the other side of the receiver relationship looking from the point of view of the argument1 element; i.e., if argument1 is the source of the receiver relationship, the target is answered. Argument1 must be an element on one side of the receiver relationship.

Report Section

Standard CORE reports are subdivided into sections at both the script language level and the user interface level to make them more manageable to use and write. Sectioning the report allows the user to select which parts of a document she or he wishes to generate and, at the script language level, which parts of the script to execute. The report section data type merely formalizes this arrangement and makes it a little easier and more standardized.

Report sections should be created and collected at the beginning of a report script, the user prompted for which sections to generate, the resultant collection queried at the beginning of each major section of the script to determine whether to interpret or skip that section, and, if a section is to be output, its number and name printed as a heading.

Report sections are used in conjunction with include files.

<	Answers a Boolean indicating whether or not the receiver is less than argument1. Report sections are ordered by number and then by name if they have the same number.
<=	Answers a Boolean indicating whether or not the receiver is less than or equal to argument1. Report sections are ordered by number and then by name if they have the same number.
<>	Answers a Boolean indicating whether or not the receiver is not equal to argument1.
=	Answers a Boolean indicating whether or not the receiver is equal to argument1.
>	Answers a Boolean indicating whether or not the receiver is greater than argument1. Report sections are ordered by number and then by name if they have the same number.
>=	Answers a Boolean indicating whether or not the receiver is greater than or equal to argument1. Report sections are ordered by number and then by name if they have the same number.
displayString	Answers a string that consists of the receiver number and name separated by two spaces.
isNil	Answers Boolean false.
name	Answers a symbol that is the unique identifier for the receiver report section.
number	Answers the number of the receiver report section.

Set

A set is an unordered collection of distinct objects. Duplicate objects added to the set are not maintained. You can iterate over all the objects in the set or access a specific object by value, but there are no keys by which the values can be found as in a dictionary.

add:	Adds the object argument1 to the receiver set. If another object of equal value is already present in the set, argument1 will not be added to the set. Argument1 can be any type of object. Returns argument1.
addAll:	Adds each object in collection argument1 to the receiver set. If argument1 contains an object equal in value to one already present in the set, another object will not be physically added to the set. If argument1 is a dictionary, then this operator iterates over the values stored in the dictionary, not the keys whereby the values are accessed. If argument1 is a string or symbol, then the added objects will all be characters. Returns argument1.
asArray	Answers an array containing all the objects in the receiver set.
asOrderedCollection	Answers an ordered collection that contains all the objects in the receiver set. The order of the objects is arbitrary.
asSortedCollection	Answers a sorted collection that contains all the objects in the receiver set. The objects in the new collection are sorted 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).
displayString	Answers a string consisting of the string representation for each member of the set, separated by a comma and space. The entire representation is enclosed in parentheses.
includes:	Answers a Boolean (true or false) indicating whether or not an object equal to argument1 is present within the receiver set.
isEmpty	Answers a Boolean (true or false) indicating whether or not at least one object is stored within the receiver set.
isNil	Answers Boolean false.
notEmpty	Answers a Boolean (true or false) indicating whether or not the receiver set contains no objects. The notEmpty method always answers the opposite of the isEmpty method.
remove:	Removes argument1 from the receiver set and reduces the size of the set by one. An error results if the set does not include argument1. Answers argument1.
remove:ifAbsent:	Removes an object equal to argument1 from the receiver set and reduces the size of the set by one. Answers argument1 if present. If the object is not contained in the set, then and only then is the expression in argument2 evaluated and the result returned. Argument2 must be a block, i.e., an expression enclosed in square brackets.
size	Answers an integer that is the number of objects contained in the receiver set.

Sorted Collection

A sorted collection is a type of ordered collection used by various report script constructs and that has built in criteria for ordering the objects it contains. New objects added to a sorted collection are automatically inserted at their proper locations among the existing objects.

add:	Adds the object argument1 to the receiver sorted collection. Argument1 will automatically be inserted at the proper location within the sorted collection based on the ordering criteria built into the receiver when it was created. Returns argument1.
addAll:	Iterates through the objects in collection argument1 and adds each, in turn, to the sorted collection. If argument1 is a dictionary, then this operator iterates over the values stored in the dictionary, not the keys whereby the values are accessed. If argument1 is a string or symbol, then the added objects will all be characters. Each object in argument1 will automatically be inserted in its proper location within the receiver based on the ordering criteria built into the receiver when it was created. Returns argument1.
asArray	Answers an array containing all the objects in the receiver sorted collection. The order of the objects in the array is the same as the order in the receiver.
asOrderedCollection	Answers an ordered collection which contains the objects stored in the sorted collection. The order of the objects in the ordered collection is the same as the order in the receiver.
asSet	Answers a set that consists of the objects in the receiver. A set only maintains a single object with any given value, so if the sorted collection contains multiple objects with the same value, the duplicate objects are lost in the conversion process.
at:	Answers the object stored in the receiver at the ordinal position given by argument1. Argument1 must be an integer. An error results if the argument is outside the bounds of the collection.
copyFrom:to:	Answers an ordered collection containing every object in the receiver from ordinal position argument1 through ordinal position. Argument1 and argument2 must be integers. An error results if either argument1 or argument2 represents a position outside the bounds of the receiver collection.
displayString	Answers a string consisting of the string representation for each object in the sorted collection, separated by comma and space. Braces enclose the entire collection representation.
first	Answers the object in the first ordinal position within the receiver sorted collection. An error results if the collection is empty.
includes:	Answers a Boolean (true or false) indicating whether or not an object equal to argument1 is present within the receiver.
isEmpty	Answers a Boolean (true or false) indicating whether or not at least one object is stored within the receiver.
isNil	Answers Boolean false.
last	Answers the object in the last ordinal position within the receiver. An error results if the receiver is empty.
notEmpty	Answers a Boolean (true or false) indicating whether or not the receiver contains no objects. The notEmpty method always answers the opposite of the isEmpty method.
occurrencesOf:	Answers the integral number of times the object argument1 occurs in the receiver collection.

remove:	Removes the first occurrence of argument1 from the receiver sorted collection and reduces the size of the collection by one. Returns argument1. An error results if the receiver does not include argument1.
remove:ifAbsent:	Removes the first occurrence of argument1 from the receiver sorted collection and reduces the size of the collection by one. Answers argument1 if present in the collection. If argument1 is not contained in the sorted collection, the expression in argument2 is evaluated and the resultant value returned. Argument2 must be a block, i.e., an expression enclosed in square brackets.
removeFirst	Removes the first object in the receiver sorted collection, reduces its size by one, and decrements the ordinal position of each object remaining in the collection by one. An error results if the receiver is empty. Answers the first object.
removeLast	Removes the last object in the receiver sorted collection and reduces its size by one. An error results if the receiver is empty. Answers the last object.
size	Answers an integer that is the number of objects contained in the receiver collection.

String

A string is an object that stores text. It is essentially an array of characters. Strings can be compared, and many other data types can be converted to and from strings.

A shorthand syntax exists to represent a string object in the COREscript expression language. Simply enclose the desired text in single quotation marks (i.e., apostrophes). For example, to create a string containing the word text, simply type (including the single quotes): ‘text’.

<	Answers a Boolean indicating whether or not the receiver is less than argument1. Strings are ordered by the ASCII values of their index position characters, with index position decreasing in significance from left to right. If one string is equal in the leftmost positions of another, the shorter is less than the greater.
<=	Answers a Boolean indicating whether or not the receiver is less than or equal to argument1. Strings are ordered by the ASCII values of their index position characters, with index position decreasing in significance from left to right. If one string is equal in the leftmost positions of another, the shorter is less than the greater.
<>	Answers a Boolean indicating whether or not the receiver is not equal to argument1.
=	Answers a Boolean indicating whether or not the receiver is equal to argument1.
>	Answers a Boolean indicating whether or not the receiver is greater than argument1. Strings are ordered by the ASCII values of their index position characters, with index position decreasing in significance from left to right. If one string is equal in the leftmost positions of another, the shorter is less than the greater.
>=	Answers a Boolean indicating whether or not the receiver is greater than or equal to argument1. Strings are ordered by the ASCII values of their index position characters, with index position decreasing in significance from left to right. If one string is equal in the leftmost positions of another, the shorter is less than the greater.
asArray	Answers an array containing all the characters in the receiver string in the same order as stored in the string.
asArrayOfSubstrings	Parses the receiver into substrings breaking the string at a blank or sequence of blanks. Answers an array containing the substrings. The substrings contain no blanks and are in the order that they occur in the receiver.
asArrayOfSubstringsSeparatedBy:	Parses the receiver into substrings breaking the string at each occurrence of the argument1 character. Answers an array containing the substrings. The substrings do not contain argument1 and are in the order that they occur in the receiver. The new array will include empty substrings if argument1 occurs in a sequence in the receiver.
asBoolean	Answers Boolean true if the receiver consists of the character sequence ‘true’, answers Boolean false otherwise.

asDate	Answers a date that is parsed and converted from the character sequence of the string. All preliminary white space is ignored. If the first character of the string is alphabetic, the date is assumed to be in month-day-year format. Otherwise, this operator expects the date format specified in the Preferences dialog. Any non-alphanumeric characters may separate the components of the date. If the month name or abbreviation is used, it need not be separated from the day or year. An error results if any characters or white space remain after the date has been parsed. The year may be specified as an integer. If the year is not specified, the current year is supplied by default. The month may be specified as a full name or a 3-letter abbreviation in mixed case, or an integer. An error results if the month value falls outside the range of valid months in a year. The day may be specified as an integer. An error results if the day value falls outside the range of valid days in the specified month for the specified year.
asFloat	Answers a float the value of which is parsed and converted from the receiver string. The strings character sequence may consist of any or all of: an initial sign (+ or -), a sequence of digits comprising the magnitude of the integral portion of the number, a decimal point (.) followed by a sequence of digits comprising the magnitude of the fractional part of the number, an uppercase E or lowercase e, another sign, and a sequence of digits comprising the magnitude of the multiplicative factor (as a power of 10). If a sign is not specified, a positive magnitude is assumed. If digits are not specified for any portion, that portion defaults to 0. The float 0.0 is returned if the string is empty. If no exponent is specified after an E or an e, the exponent is ignored.
asHierarchicalNumber	Answers a hierarchical number whose value is parsed and converted from the character sequence of the receiver string. The hierarchical number consists of one or more cells separated by periods (.). Each cell can consist either of one or more letters, or one or more digits.
asInteger	Answers an integer whose value is parsed and converted from the character sequence of the receiver string. The integer may consist of an optional sign (+ or -) and/or a sequence of digits. If no digits are specified, 0 is assumed. Returns 0 if the string is empty or begins with a non-numeric character.
asLowercase	Answers a string that is a copy of the receiver with all uppercase alphabetic characters converted to lowercase.
asOrderedCollection	Answers an ordered collection containing the characters stored in the receiver string.
asSet	Answers a set containing the unique characters in the receiver string.
asSortedCollection	Answers a sorted collection containing the characters in the receiver string.
asSymbol	Answers a symbol containing the same characters as the string.
asTime	Answers a time with a value that is parsed and converted from the character sequence of the receiver string. The string may be specified in either of the following formats: hh:mm, hh:mm:ss. Each component may be either 1 or 2 digits. If the seconds are not specified, 0 is assumed. If the hours are less than 12, the string may optionally be suffixed by AM or PM in mixed case. An error results if any component is out of range.

asTimeStamp	Answers a timestamp the value of which is parsed and converted from the character sequence of the receiver. The receiver can consist of a date (as specified above for asDate) followed by white space, the word at, more white space, and a time (as specified above for asTime). If the time is omitted, 00:00:00 is assumed.
asUppercase	Answers a string that is a copy of the receiver with all lowercase alphabetic characters converted to uppercase.
at:	Answers the character at the ordinal position in the receiver indicated by argument1. Argument1 must be an integer. An error results if the index is outside the bounds of the string.
at:put:	Replaces the character in the receiver in ordinal position argument1 with argument2. Argument1 must be an integer and argument2 must be a character.
concat:	Answers a new string consisting of the characters of the receiver string followed by the characters in argument1. Argument1 must be an array, ordered collection, sorted collection, string, or symbol and contain only objects of type character.
copyFrom:to:	Answers a new string containing every character in the receiver string from ordinal position argument1 through ordinal position argument2. Argument1 and argument2 must be integers. An error results if either argument1 or argument2 represents a position outside the bounds of the string.
decrement	Answers a new string that is a copy of the receiver string with the last character decremented. If the string ends in a substring of one or more ‘a’s, then the final 'a's are changed to ‘z’s and the last non-a is decremented. If the string consists entirely of ‘a’s, then the last ‘a’ is omitted and the rest are changed to ‘z’s. If the string consists of a single a, then the answer is also a. The case (upper or lower) of each character in the answer matches the case of the corresponding character of the receiver.
doubleEmbedded:	Answers a new string that is a copy of the receiver in which all occurrences of the character specified by argument1 have been doubled. For example, ‘50% growth’ doubleEmbedded: $% answers ‘50%% growth’.
first	Answers the first character in the receiver string. An error results if the receiver is empty.
includes:	Answers a Boolean (true or false) indicating whether or not the character argument1 is contained in the receiver string.
includesString:	Answers a Boolean (true or false) indicating whether or not the sequence of characters in argument1 was found in the receiver. An asterisk (*) within argument1 can be matched by any substring of 1 or more characters in the string or even by an empty string. For example, includesString: will find re*d within red, reed, and received.
increment	Answers a new string that is a copy of the receiver string with the last character incremented. If the string ends in a substring of one or more ‘z’s, then the final 'z's are changed to ‘a’s and the last non-z is incremented. If the string consists entirely of ‘z’s, an ‘a’ appended to the end of the string in addition to changing the ‘z’s to ‘a’s. The case (upper or lower) of each character in the answer matches the case of the corresponding character in the receiver.
indexOf:	Answers the integer index of the first occurrence of character argument1 in the receiver string. An error results if the receiver does not include the character.

isEmpty	Answers a Boolean (true or false) indicating whether or not at least one character is stored within the receiver string. That is, the string is at least one character long.
isNil	Answers Boolean false.
last	Answers the last character of the receiver string. An error results if the string is empty.
match:	Searches the receiver for the first occurrence of the sequence of characters in argument1. The search begins with the first character in the receiver. If there is a match, answers an array of size 2 where the first index position is the index of the first character of argument1 and the second index position contains the index of the last character of argument1 in the receiver. Answers the array (0 0) if there is no match. Wildcards are supported.
match:startingIndex:	Searches the receiver for the first occurrence of the sequence of characters in argument1. The search begins at the index into the receiver specified by argument2 rather than at the beginning of the receiver. Answers an array of size 2 where the first index position is the index of the first character of argument1 and the second index position is the index of the last character of argument1 in the receiver. Answers the array (0 0) if there is no match. Wildcards are supported.
notEmpty	Answers a Boolean (true or false) indicating whether or not the receiver string contains no characters. The notEmpty method always answers the opposite of the isEmpty method.
occurrencesOf:	Answers an integer that is the number of times argument1 occurs in the receiver string. Argument1 must be a character.
size	Answers an integer that is the number of characters in the receiver.
trimBlanks	Answers a new string that is a copy of the receiver string with any leading and trailing blanks, carriage returns, and tabs removed.

Symbol

A symbol is a special kind of string that is unique system-wide and unchangeable. The names of objects are frequently represented as symbols, and symbols are often used as dictionary keys.

A shorthand syntax exists to represent some symbol objects in the COREscript expression language. Any symbol beginning with an alphabetic character and consisting only of alphanumeric characters, possibly interrupted by single colons, can be automatically created by preceding the character sequence with a pound sign (#). For example, to represent the name of one of the COREscript expression language operators as a symbol, you could simply type '#at:put:' (without the quotation marks).

<	Answers a Boolean indicating whether or not the receiver is less than argument1. Symbols are compared as strings and ordered by the ASCII values of their index position characters, with index position decreasing in significance from left to right. If one symbol is equal in the leftmost positions of another, the shorter is less than the greater.
<=	Answers a Boolean indicating whether or not the receiver is less than or equal to argument1. Symbols are compared as strings and ordered by the ASCII values of their index position characters, with index position decreasing in significance from left to right. If one symbol is equal in the leftmost positions of another, the shorter is less than the greater.
<>	Answers a Boolean indicating whether or not the receiver is not equal to argument1.
=	Answers a Boolean indicating whether or not the receiver is equal to argument1.
>	Answers a Boolean indicating whether or not the receiver is greater than argument1. Symbols are compared as strings and ordered by the ASCII values of their index position characters, with index position decreasing in significance from left to right. If one symbol is equal in the leftmost positions of another, the shorter is less than the greater.
>=	Answers a Boolean indicating whether or not the receiver is greater than or equal to argument1. Symbols are compared as strings and ordered by the ASCII values of their index position characters, with index position decreasing in significance from left to right. If one symbol is equal in the leftmost positions of another, the shorter is less than the greater.
asArray	Answers an array containing all the characters in the symbol in the same order as they occur in the symbol.
asArrayOfSubstrings	Parses the receiver into substrings breaking the receiver character string at a blank or sequence of blanks. Answers a new array containing the substrings. The substrings contain no blanks and are in the order that they occur in the receiver.
asArrayOfSubstringsSeparatedBy:	Parses the receiver into substrings breaking the string at each occurrence of the argument1 character. Answers an array containing the substrings. The substrings do not contain argument1 and are in the order that they occur in the receiver. The new array will include empty substrings if argument1 occurs in a sequence in the receiver.
asBoolean	Answers Boolean true if the receiver symbol consists of the character sequence 'true', answers Boolean false otherwise.

asDate	Answers a date the value of which is parsed and converted from the character sequence of the receiver symbol. All preliminary white space is ignored. If the first character of the symbol is alphabetic, the date is assumed to be in month-day-year format. Otherwise, this operator expects the date format specified in the Preferences dialog. Any non-alphanumeric characters may separate the components of the date. If the month name or abbreviation is used, it need not be separated from the day or year. An error results if any characters or white space remain after the date has been parsed. The year may be specified as an integer. If the year is not specified, the current year is supplied by default. If less than three digits are specified for the year, it is assumed to fall between 1950 and 2049 inclusive. The month may be specified as a full name or a 3-letter abbreviation in mixed case, or an integer. An error results if the month value falls outside the range of valid months in a year. The day may be specified as an integer. An error results if the day value falls outside the range of valid days in the specified month for the specified year.
asFloat	Answers a float the value of which is parsed and converted from the receiver symbol. The symbols character sequence may consist of any or all of: an initial sign (+ or -), a sequence of digits comprising the magnitude of the integral portion of the number, a decimal point (.) followed by a sequence of digits comprising the magnitude of the fractional part of the number, an uppercase E or lowercase e, another sign, and a sequence of digits comprising the magnitude of the multiplicative factor (as a power of 10). If a sign is not specified, a positive magnitude is assumed. If digits are not specified for any portion, that portion defaults to 0. The float 0.0 is returned if the string is empty. If no exponent is specified after an E or an e, the exponent is ignored.
asHierarchicalNumber	Answers a hierarchical number whose value is parsed and converted from the character sequence of the receiver symbol. The hierarchical number consists of one or more cells separated by periods (.). Each cell can consist either of one or more letters, or one or more digits.
asInteger	Answers an integer whose value is parsed and converted from the character sequence of the receiver symbol. The integer may consist of an optional sign (+ or -) and/or a sequence of digits. If no digits are specified, 0 is assumed. Returns 0 if the string is empty or begins with a non-numeric character.
asLowercase	Answers a string containing the receiver character string with all uppercase alphabetic characters converted to lowercase.
asOrderedCollection	Answers an ordered collection which contains the characters in the receiver symbol in the order that they occur in the receiver.
asSet	Answers a set containing the unique characters in the symbol.
asSortedCollection	Answers a sorted collection of the characters in the symbol.
asUppercase	Answers a new string that is a copy of the receiver with all lowercase alphabetic characters converted to uppercase.
at:	Answers the character at the position indicated by argument1 in the receiver symbol string. Argument1 must be an integer. An error results if the index is outside the bounds of the symbol.
concat:	Answers a new symbol with the characters of the receiver symbol followed by the characters in argument1. Argument1 must be an array, ordered collection, sorted collection, string, or symbol and must contain only objects of type character.

copyFrom:to:	Answers a new string containing every character in the receiver string from ordinal position argument1 through ordinal position argument2. Argument1 and argument2 must be integers. An error results if either argument1 or argument2 represents a position outside the bounds of the symbol.
displayString	Answers the receiver symbol as a string.
doubleEmbedded:	Answers a new string that is a copy of the receiver symbol string in which all occurrences of the character specified by argument1 have been doubled. For example, 10% growth doubleEmbedded: $% answers 10%% growth.
first	Answers the first character of the receiver symbol.
includes:	Answers a Boolean (true or false) indicating whether or not the character argument1 is contained in the receiver symbol.
includesString:	Answers a Boolean (true or false) indicating whether or not the sequence of characters in argument1 is found in the receiver. An asterisk (*) within argument1 can be matched by any substring of 1 or more characters in the receiver or by an empty string. For example, includesString: will find re*d within red, reed, and received.
indexOf:	Answers the integer index of the first occurrence of character argument1 in the symbol. An error results if the symbol does not include the character.
isEmpty	A symbol is never empty. Therefore, the isEmpty method will always answer Boolean false.
isNil	Answers Boolean false.
last	Answers the last character of the receiver symbol.
match:	Searches the receiver for the first occurrence of the sequence of characters in argument1. The search begins with the first character in the receiver. If there is a match, answers an array of size 2 where the first index position is the index of the first character of argument1 and the second index position contains the index of the last character of argument1 in the receiver. Answers the array (0 0) if there is no match. Wildcards are supported.
match:startingIndex:	The match:startingIndex: message searches the receiver for the first occurrence of the sequence of characters in argument1. The search begins at the index into the receiver specified by argument2 rather than at the beginning of the receiver. Answers an array of size 2 where the first index position is the index of the first character of argument1 and the second index position is the index of the last character of argument1 in the receiver. Answers the array (0 0) if there is no match. Wildcards are supported.
notEmpty	A symbol is never empty. Therefore, the notEmpty method will always answer Boolean true.
occurrencesOf:	Answers an integer that is the number of times argument1 occurs in the receiver symbol string. Argument1 must be a character.
size	Answers an integer that is the number of characters in the receiver.
trimBlanks	Answers a string that contains the sequence of characters in the receiver with any leading and trailing blanks, carriage returns, and tabs removed.

Table

A table is a collection of objects forming a matrix or two-dimensional array, essentially an array of arrays, where the subsidiary arrays are all of the same size. The size of the table is fixed at its creation. You may think of a table as having rows and columns. At each intersection of a particular row with a particular column is a cell containing an object that may be individually stored or accessed.

cells	Answers an ordered collection containing the contents of each cell obtained by iterating across the rows.
displayString	Answers a string consisting of the string representations of the contents of each table cell. The cell representations are included in column order by row. The column entries are separated by spaces. Each row representation is separated from other rows by a carriage return - line feed.
first	Answers the contents of the first cell in the table.
isNil	Answers Boolean false.
last	Answers the contents of the last cell in the table.
numberOfColumns	Answers an integer that is the number of columns defined for the table.
numberOfRows	Answers an integer that is the number of rows defined for the table.
row:column:	Answers the object stored at the specified ordered pair of coordinates, the intersection of row argument1 and column argument2 in the two-dimensional matrix represented by the receiver table. This operator answers a nil (undefined) value if no object has previously been stored at the specified location, but no error results. Argument1 and Argument2 must be integers. An error results if either row argument1 or column argument2 fall outside the bounds of the table.
row:column:put:	Stores object argument3 at the specified ordered pair of coordinates, the intersection of row argument1 and column argument2 in the two-dimensional matrix represented by the receiver table. Any object previously stored at that location is overwritten. Argument1 and Argument2 must be integers. Argument3 can be any object. An error results if either row argument1 or column argument2 fall outside the bounds of the table. Argument1 and Argument2 must be integers. Argument3 can be any object.
size	Answers an integer that is the number of cells in the table regardless of whether or not a cell is occupied. The size of the table is the number of rows times the number of columns.

Time

A value that represents a time of day to the nearest second. Time is based on a twelve hour clock and, therefore, also includes an AM/PM designation.

<	Answers a Boolean indicating whether or not the receiver is less than argument1.
<=	Answers a Boolean indicating whether or not the receiver is less than or equal to argument1.
<>	Answers a Boolean indicating whether or not the receiver is not equal to argument1.
=	Answers a Boolean indicating whether or not the receiver is equal to argument1.
>	Answers a Boolean indicating whether or not the receiver is greater than to argument1.
>=	Answers a Boolean indicating whether or not the receiver is greater than or equal to argument1.
addTime:	Answers a new time that is the receiver time plus the argument1. Argument1 must be a time and, thus, the receiver time can be incremented by one second less than twenty four hours.
displayString	Answers a string containing the time as a sequence of printable characters. The format of the string is specified in the Windows system settings.
hours	Answers an integer that is the hours component of the receiver based on a 24 hour clock.
isNil	Answers Boolean false.
minutes	Answers an integer that is the minutes component of the receiver.
seconds	Answers an integer that is the seconds component of the receiver.
subtractTime:	Answers a new time that is the receiver time plus the argument1. Argument1 must be a time and, thus, the receiver time can be decremented by one second less than twenty-four hours.

TimeStamp

An object that consisting of a date and time. Timestamps are automatically incorporated into other objects to mark their creation or modification and, in this usage, cannot be changed.

<	Answers a Boolean indicating whether or not the receiver is less than argument1.
<=	Answers a Boolean indicating whether or not the receiver is less than or equal to argument1.
<>	Answers a Boolean indicating whether or not the receiver is not equal to argument1.
=	Answers a Boolean indicating whether or not the receiver is equal to argument1.
>	Answers a Boolean indicating whether or not the receiver is greater than to argument1.
>=	Answers a Boolean indicating whether or not the receiver is greater than or equal to argument1.
asDate	Answers a date object that is equal to the date portion of the receiver timestamp.
asTime	Answers a time object that is equal to the time portion of the receiver timestamp.
displayString	Answers a string containing the timestamp contents as a sequence of printable characters. The format of the string is ‘date at time’ (without the quotes) where the format for date and time are specified by the Windows system settings.
isNil	Answers Boolean false.

HTML Diagram Object Types

The HTML diagram object types are used when outputting diagrams in a JPG or PNG file and interfacing the hotspots in the diagram to other HTML information output from CORE.

Diagram Entity Locator

A diagram entity locator contains information that associates a database entity (usually an element or class) with a rectangle in a diagram output in a JPG or PNG file. The rectangle information consists of the coordinates of the diagram object selection region (i.e., HTML hotspot). Diagram entity locators are returned only by the COREscript Diagram File Output construct.

coordinates	Answers a rectangle containing the diagram coordinates for the HTML hotspot.
entities	Answers an ordered collection that contains the database entity.

Point

A point contains the coordinates of either the upper-left corner or lower-right corner of an object selection region in a CORE generated JPG or PNG file. Points are the constituents of the data type rectangle.

x	Answers an integer that is the x coordinate of the point.
y	Answers an integer that is the y coordinate of the point.

Rectangle

A rectangle consists of the coordinate information for an object selection region (i.e., an HTML hotspot) in a CORE generated JPG or PNG file. A rectangle contains two points: the upper-left corner and the lower-right corner. A rectangle is a constituent of a diagram entity locator.

leftTop	Answers a point that is the upper-left corner of the region specified by the rectangle.
rightBottom	Answers a point that is the lower-right corner of the region specified by the rectangle.

THIS PAGE INTENTIONALLY BLANK

Structure Object Types

The structure object types are encountered when traversing a processing unit's structure.

Exit Construct

An exit construct is the representation of a structure exit node.

constructType	Returns the symbol #exit.
element	Answers the element to which the construct corresponds.

Function Construct

A function construct is the structure representation of a processing element reference.

branches	Answers an ordered collection containing the branches which emanate from the construct. The branches are ordered in the order that they would appear (top to bottom) in an FFBD or EFFBD opened on the parent element.
constructType	Returns the symbol #function.
element	Answers the element to which the construct corresponds.

Iterate Construct

An iterate construct is the representation of a structure iterate.

branch	Answers the main network branch for the iterate (i.e., the branch enclosed by the iterate).
constructType	Returns the symbol #iterate.
domainSet	Answers the element associated with the iterate.

Loop Construct

A loop construct is the representation of a structure loop.

branch	Answers the main network branch for the loop (i.e., the branch enclosed by the loop).
constructType	Returns the symbol #loop.
loopCondition	Answers the string which is the annotation on the loop. Answers nil if there is no loop annotation.

Loop Exit Construct

A loop exit construct is the representation of a structure loop exit.

constructType	Returns the symbol #loopExit.

Network

A network defines the structure for a processing element.

structure	Answers a network branch that is the main branch of the receiver.

Network Branch

A network branch is a single, simple branch within a processing element structure.

annotation	Answers a string which is the annotation for the branch. Answers nil if there is no branch annotation.
constructs	Answers an ordered collection of constructs on the branch. The constructs are ordered by the order of their occurrence on the branch. The type of construct (i.e., either ExitConstruct, FunctionConstruct, IterateConstruct, LoopConstruct, LoopExitConstruct, ParallelConstruct, ReplicateConstruct, SelectConstruct) corresponds to the node type.

Network Exit Branch

A network exit branch is the structure representation of an exit branch emanating from a function.

completionCriterion	Answers the element (in the Exit class) to which this exit branch corresponds.
constructs	Answers an ordered collection of constructs on the branch. The constructs are ordered by the order of their occurrence on the branch. The type of construct (i.e., either ExitConstruct, FunctionConstruct, IterateConstruct, LoopConstruct, LoopExitConstruct, ParallelConstruct, ReplicateConstruct, SelectConstruct) corresponds to the node or reference type.
selectionProbability	Answers the value of the attribute selectionProbability for the exits by relation between the Function from which the branch emanates and the Exit represented by the branch. Returns either a float or nil.

Network Parallel Branch

A network parallel branch is the structure representation of a parallel branch emanating from an AND node.

annotation	Answers a string which is the annotation for the branch. Answers nil if there is no branch annotation.
constructs	Answers an ordered collection of constructs on the branch. The constructs are ordered by the order of their occurrence on the branch. The type of construct (i.e., either ExitConstruct, FunctionConstruct, IterateConstruct, LoopConstruct, LoopExitConstruct, ParallelConstruct, ReplicateConstruct, or SelectConstruct) corresponds to the node or reference type.
killStatus	Answers a Boolean indicating whether or not the branch is a kill branch.

Network Selection Branch

A network selection branch is the structure representation of a branch emanating from an OR node.

annotation	Answers a string which is the annotation for the branch. Answers nil if there is no branch annotation.
constructs	Answers an ordered collection of constructs on the branch. The constructs are ordered by the order of their occurrence on the branch. The type of construct (i.e., either ExitConstruct, FunctionConstruct, IterateConstruct, LoopConstruct, LoopExitConstruct, ParallelConstruct, ReplicateConstruct, or SelectConstruct) corresponds to the node or reference type.
selectionProbability	Answers the value of the selection probability for the branch. Returns either a float or nil.

Parallel Construct

A parallel construct is the representation of a structure parallel.

branches	Answers an ordered collection containing the branches which emanate from the construct. The branches are ordered in the order that they would appear (top to bottom) in an FFBD or EFFBD opened on the parent element.
constructType	Returns the symbol #parallel.

Replicate Construct

A replicate construct is the representation of a structure replicate.

branch	Returns a parallel network branch which is the main branch (i.e., the replicated branch) of the replicate.
branches	Returns an ordered collection of parallel network branches where the first is the coordination branch and the second is the replicated branch.
constructType	Returns the symbol #replicate.
coordinationBranch	Returns a parallel network branch which the is the coordination branch of the replicate.

Select Construct

A select construct is the representation of a structure select.

branches	Answers an ordered collection containing the branches which emanate from the construct. The branches are ordered in the order that they would appear (top to bottom) in an FFBD or EFFBD opened on the parent element.
constructType	Returns the symbol #replicate.
coordinationBranch	Returns a parallel network branch which the is the coordination branch of the replicate.

THIS PAGE INTENTIONALLY BLANK

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