Solaris Inspector Library

A Guide to the BigFix® Solaris Inspectors

BigFix, Inc.
Emeryville, CA

Last Modified: October 3, 2007
Compatible with
BES 7.0

All inquiries regarding the foregoing should be addressed to:
BigFix, Inc.
1480 64th Street, Suite 200
Emeryville, CA 94608

Contents

Preface

The *Solaris Inspector Library* is a complete guide to the ordinary phrases (known as Inspectors) of the **BigFix Relevance Language**™ for the Solaris operating system. With this manual and the ***Authoring Fixlet messages*** guide, you will be able to write **Fixlet**® messages and post them to **Fixlet Sites**.

Audience

This guide is for IT managers, product support groups and other people who want to write Fixlet messages for Solaris computers. IT managers, in particular, will use the BigFix Enterprise Suite (BES) to keep a network of computers up to date and running smoothly without interruption.

Organization of this manual

This guide contains a library of the ordinary phrases, or Inspectors, in the Relevance Language. For each Inspector, there is a list of corresponding properties. The Inspectors are organized by category as follows:

- **Primitive Objects.** This chapter covers the basic data types supported by the language and describes the operations that can be applied to them.
- **World Objects.** This chapter covers the keywords used to create all the ‘top’ level objects of the world. The properties of these objects provide access to all levels of the machine state that can be inspected.
- **File System Objects.** This chapter covers the keywords for extracting information from the file system, like files, drives, pathnames, folders, etc. It also includes the keywords needed to identify and compare version information of files.
- **System Objects.** This chapter covers the keywords available for querying aspects of the operating system. This chapter also covers the keywords used to describe the vendors and types of the various processors that coexist in a typical computer system.
- **Site Objects.** This chapter covers the keywords that query the properties of Fixlet sites to which the client is subscribed.
- **Client Objects.** This chapter covers properties of the client application hosting the relevance evaluation.
- **Environment Objects.** Environment objects are provided to access environment variables. These are the same variables you’re used to seeing in a shell like BASH when you type the 'printenv' command. Note that you’re inspecting the environment of the application executing the relevance clause, which may or may not match the environment of other applications on the computer.

- **User Objects.** This chapter covers the local and current user keywords. A Local User object is provided to access the user data of the local machine. Note that domain users are not available through this inspector.
- **Action Objects.** These are the keywords associated with properties available for inspection during the execution of BigFix Actions.
- **Networking Objects.** These are the keywords used to query the local network configuration.
- **Introspectors.** These are objects that talk about the language itself, letting you interrogate the types and objects of the relevance language.
- **Key Phrases (Inspectors).** This chapter provides an alphabetical list of all the Inspector key phrases along with the form, context object type, and resulting object type.

Conventions Used in this manual

This document makes use of the following conventions and nomenclature:

Convention	Use
Bold Sans	A bold sans-serif font is used for Inspector headers.
`Mono-space`	A mono-spaced font is used to indicate expressions in the Relevance Language.
{curly braces}	Braces are used to indicate the comparison {=, !=}or arithmetic operators {+, -} that are available for a binary operation.
<angle bracket>	Angle brackets are used to indicate an object type. For instance to indicate the creation and usage of a particular object, you might see “absolute value of <integer>” which indicates that an integer is to follow the “absolute value of” keyphrase.
Italics	An inspector form. Some inspectors are simple keywords. Others are a keyword in combination with another inspector. Still other forms allow iteration through object lists. Each form is defined below
Small print	The small print beneath the description of each Inspector lists the first implementation for every relevant operating system.

Examples

Square bullets and a mono-spaced font denote examples of Inspectors as used in a Relevance Expression:

- concatenation of "light" & "year"
- Returns "lightyear"

Versions

Most Inspectors have equivalent implementations on other operating systems, allowing you to write cross-platform relevance expressions. There are exceptions, of course. To keep track of them for each Inspector and operating system, the debut BigFix version is listed at the end of the description, e.g.:

```
Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
```

These are the abbreviations for some of the current operating systems:

Win: the Windows version of the BigFix Enterprise Suite (BES).

Lin: the SUSE and Red Hat Linux versions of BES.

Sol: the SUN Solaris operating system version of BES.

HPUX: the Hewlett-Packard Unix version of BES.

AIX: the AIX version of BES.

Mac: the Macintosh version of BES.

Introduction

This manual details the properties and operators of the BigFix Inspector keywords. Inspectors are the basis of the Relevance Language. They can be thought of as object-oriented representations of the underlying computer system. With Inspectors, you can write Relevance expressions that query all aspects of the computer. Inspectors are also used to produce substituted variables in action buttons. They can also be used to create human-readable descriptions of any given computer system.

You will notice that many of the keywords of the language are not unique; they get their meaning from their context. Accordingly, their definitions often include a phrase to define the context of each Inspector, such as "character of <string>" or "character <integer> of <string>."

This document describes inspectors for the Solaris Operating System. Contact your BigFix sales representative for information about Inspector Guides for other operating systems, including Windows, HP, AIX, Red Hat Linux and Mac.

In the following pages, you will find tables defining the inspectors of the relevance language. The inspectors come in several **forms** depending upon their context:

Form	Syntax required
Cast	<object> as keyword
Global	keyword
Index	keyword *index* of <object>
Named	keyword "*name*" of <object>
NamedGlobal	keyword "*name*"
Numbered	keyword *number* of <object>
NumberedGlobal	keyword *number*
Plain	keyword of <object>

These differ from one another in format and in the syntax they require. Except for Cast, these forms can be used to access both single objects and *lists* of objects by using the plural form of the keyword. The plurals are listed in the Keyword section later in this document.

Creation Methods are used to create objects of the specified type, and various **Properties** are defined for each object.

Operators list the binary and unary operations that can be performed with the given object type. Binary operators take two inputs and generate one output. The integer '+' (addition) operator is an example of a binary operation. Unary operators take a single input and generate a single output. The boolean 'Not' operation is an example of a unary operation.

Primitive Objects

The relevance language is based upon a comprehensive set of primitive objects. These primitives are the basic building blocks of the more complex objects to follow.

Boolean

Creation Methods

These boolean creation methods are in addition to the other properties that return the boolean type.

Key Phrase	*Form*	Description
<string> as boolean	*Cast*	Returns a boolean TRUE or FALSE from a string. The string must contain values of "TRUE" or "FALSE". Case is ignored. For example, "FalSe" as boolean = FALSE. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
bit <integer> of <integer>	*Numbered*	Return TRUE if the bit referenced by the integer is on. Bits are numbered starting with zero being the least significant. For example, bit 0 of 5 and bit 2 of 5 and not bit 1 of 5 = TRUE. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
boolean <string>	*NamedGlobal*	Creates the boolean value of the <string>, e.g., • boolean "False" = FALSE. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
conjunction of <boolean>	*Plain*	This inspector performs a serial AND on all its boolean arguments: • conjunction of (true; true; true) -> TRUE • conjunction of (true; true; false) -> FALSE. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
disjunction of <boolean>	*Plain*	This inspector performs a serial OR on all its boolean arguments: • disjunction of (false; false; false) -> FALSE • disjunction of (false; false; true) -> TRUE. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
false	*PlainGlobal*	Creates a boolean with value FALSE. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Description
true	*PlainGlobal*	Creates a boolean with value TRUE. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Properties

Key Phrase	*Form*	Return Type	Description
<boolean> as string	*Cast*	<string>	Converts the boolean value to a string. The possible values returned are "True" and "False" with this exact case, e.g., • TRUE as string = "True". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
conjunction of <boolean>	*Plain*	<boolean>	This inspector performs a serial AND on all its boolean arguments: • conjunction of (true; true; true) -> TRUE • conjunction of (true; true; false) -> FALSE. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
disjunction of <boolean>	*Plain*	<boolean>	This inspector performs a serial OR on all its boolean arguments: • disjunction of (false; false; false) -> FALSE • disjunction of (false; false; true) -> TRUE. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Operators

Key phrase	Return Type	Description
<boolean> * <time range>	*<timed(time range, boolean)>*	Returns a time interval labeled with a boolean TRUE or FALSE. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
<boolean> {cmp} <boolean>	*<boolean>*	Compare two boolean expressions. Returns another boolean, depending on the evaluation of the comparison: • {cmp} is one of: =, != . Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key phrase	Return Type	Description
<boolean> {op} <boolean>	*<boolean>*	Operates on two boolean expressions. Returns another boolean, depending on the evaluation of the operation, e.g., (True And True) = True. • {op} is one of: And, Or . Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<time interval> {cmp} <time interval>	*<boolean>*	Compare two time intervals, where: • {cmp} is one of: =, !=, <, <=, >, >= . Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<time range> * <boolean>	*<timed(time range, boolean)>*	Returns a time interval labeled with the specified boolean, in the form of: • (<date> to <date>), <boolean>. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Examples

- `bit 0 of (least integer + 1)`
- Returns the least significant bit of the smallest possible integer, plus one.

Integer

Integers are represented internally as 64-bit signed values.

Creation Methods

These integer creation methods are in addition to the other properties that return the integer type.

Key Phrase	*Form*	Description
<integer> as integer	*Cast*	Integer casting for completeness. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> as integer	*Cast*	Converts from a string to an integer. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
absolute value of <integer>	*Plain*	Creates the positive value of the <integer> object. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
greatest integer	*PlainGlobal*	Creates the value 9,223,372,036,854,775,807. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	***Form***	**Description**
hexadecimal integer <string>	*NamedGlobal*	Creates an integer from the provided hexadecimal value. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
integer <integer>	*NumberedGlobal*	Creates a global object with the given integer value, e.g., Integer 123. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
integer <string>	*NamedGlobal*	Creates a global object with the integer value given by a string, e.g., Integer "123" creates the value 123. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
least integer	*PlainGlobal*	Creates the value -9,223,372,036,854,775,808. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
length of <rope>	*Plain*	Creates an integer object corresponding to the number of bytes in the rope. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
length of <string>	*Plain*	Creates an integer object corresponding to the number of bytes in the string. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
lower bound of <integer range>	*Plain*	The low end of the integer range. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
maximum of <integer>	*Plain*	Returns the maximum of a list of integers. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
minimum of <integer>	*Plain*	Returns the minimum of a list of integers. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
numeric value of <string>	*Plain*	Creates an integer object containing the value of the first number contained in a string. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
product of <integer>	*Plain*	Multiplies a list of integers, returning the product. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
significant digits <integer> of <integer>	*Numbered*	Creates a number with <integer> significant digits (e.g.. significant digits 3 of 1235569 = 1240000). Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
sum of <integer>	*Plain*	Returns the sum of a list of integers. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Description
upper bound of <integer range>	*Plain*	The high end of the integer range. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1

Properties

Integers are represented internally as 64-bit signed values.

Key Phrase	*Form*	Return Type	Description
<integer> as bit set	*Cast*	<bit set>	Returns the bits of the binary representation of the integer; bit zero is the least-significant bit. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
<integer> as bits	*Cast*	<bit set>	Returns the bits of the binary representation of the integer; bit zero is the least-significant bit. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
<integer> as day_of_month	*Cast*	<day of month>	Cast an integer as a day of the month type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<integer> as hexadecimal	*Cast*	<string>	Converts an integer into a hexadecimal string. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
<integer> as integer	*Cast*	<integer>	Reflective cast for completeness. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<integer> as month	*Cast*	<month>	Returns the name of the nth month of the year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<integer> as string	*Cast*	<string>	Converts an integer to a string. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<integer> as year	*Cast*	<year>	Casts an integer as a year type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
absolute value of <integer>	*Plain*	<integer>	Returns the positive value of the integer. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
bit <integer> of <integer>	*Numbered*	<boolean>	Returns TRUE if the numbered bit is on. Bits are numbered starting at zero. Bit 0 is the least significant bit. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
maximum of <integer>	*Plain*	<integer>	Returns the maximum of a list of integers. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Return Type	Description
minimum of <integer>	*Plain*	<integer>	Returns the minimum of a list of integers. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
product of <integer>	*Plain*	<integer>	Multiplies a list of integers, returning the product. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
set of <integer>	*Plain*	<integer set>	Creates a set from the given list of semicolon-separated integers. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
significant digits <integer> of <integer>	*Numbered*	<integer>	Returns a number with <integer> significant digits (e.g., significant digits 3 of 1235569 = 1240000). Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
sum of <integer>	*Plain*	<integer>	Returns the sum of a list of integers. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
unique value of <integer>	*Plain*	<integer with multiplicity>	Given a set of integers, returns the number of instances of each integer. Given (1,2,2,2,3), returns (1,3,1). Earlier versions of this Inspector returned the unique set of integers. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Operators

Key phrase	Return Type	Description
<hertz> {op} <integer>	<*hertz*>	Returns a hertz object operated on by the given integer, where: • {op} is one of: *, / . Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<integer set> contains <integer>	<*boolean*>	Returns TRUE if the specified set contains the given integer. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
<integer> * <number of months>	<*number of months*>	Multiply a number of months by an integer, producing a new number of months. This is a typical technique to create a value of this type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<integer> * <time range>	<*timed(time range,*	Returns a tuple of a time interval and an integer. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Key phrase	Return Type	Description
	integer)>	
<integer> {cmp} <integer>	*<boolean>*	Returns boolean TRUE or FALSE, depending on the comparison operator, where: • {cmp} is one of: =, !=, <, <=, >, >= . Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<integer> {op} <integer>	*<integer>*	Returns the integer solution to the equation, depending on the operator, where: • {op} is one of: +, -, *, /, mod . Win:2.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<time range> * <integer>	*<timed(time range, integer)>*	Returns a time interval labeled with the specified integer, in the form of: • (<date> to <date>), <integer>. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Examples

- `hexadecimal integer "A0"`
  - Returns 160.
- `numeric value of "string 123 xyz 45" = 123`
  - Returns TRUE.
- `255 as hexadecimal`
  - Returns the string "ff".
- `maximum of (7;2;4;5)`
  - Returns 7.
- `significant digits 3 of 1235569`
  - Returns 1240000.
- `set of (1;2;3) contains 3`
  - Returns TRUE.
- `21 mod 5`
  - Returns 1.

Integer Range

Specifies a range between two 64-bit signed integers.

Creation Methods

Key Phrase	*Form*	Description
distance of <selected server>	*Plain*	The distance, in IP gateway hops, to the server. Among servers with the same priority, closer servers are preferred. Returns an integer range, since the exact distance may not be known. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1

Properties

Key Phrase	*Form*	Return Type	Description
lower bound of <integer range>	*Plain*	<integer>	The low end of the integer range. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
upper bound of <integer range>	*Plain*	<integer>	The high end of the integer range. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1

Integer With Multiplicity

These Inspectors deal with arrays of integers, allowing you to pluck out unique numbers and count them. These objects are derived from integer types.

Creation Methods

Key Phrase	*Form*	Description
unique value of <integer>	*Plain*	Given a set of integers, returns the number of instances of each integer. Given (1,2,2,2,3), returns (1,3,1). Earlier versions of this Inspector returned the unique set of integers. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Properties

Key Phrase	*Form*	Return Type	Description
multiplicity of <integer with multiplicity>	*Plain*	<integer>	Returns the multiplicity (quantity) of each element in a multiple integer list. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Integer Set

These Inspectors deal with sets of integers, which are essentially lists or arrays with integer elements. Think of them as mathematical sets: you can compare them, subtract them from other sets and form the union and intersection of multiple sets.

Creation Methods

Key Phrase	*Form*	Description
set of <integer>	*Plain*	Creates a set from the given integers (usually plural). Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
union of <integer set>	*Plain*	Returns a set of integers equal to the union of the specified sets. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Properties

Key Phrase	*Form*	Return Type	Description
element of <integer set>	*Plain*	<integer>	Typically used in the plural, returns the individual elements of the specified set of integers. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
intersection of <integer set>	*Plain*	<integer set>	Returns a set of integers equal to the intersection of the specified sets, in numeric order and with redundant elements stripped out. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
size of <integer set>	*Plain*	<integer>	Returns the number of elements in the specified set. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Key Phrase	*Form*	Return Type	Description
union of <integer set>	*Plain*	<integer set>	Returns a set of integers equal to the union of the specified sets, in numeric order and with redundant elements stripped out. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Operators

Key phrase	Return Type	Description
<integer set> - <integer set>	<*integer set*>	Subtracts the elements in the second set from the elements in the first. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
<integer set> * <integer set>	<*integer set*>	Returns the intersection of the two specified sets. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
<integer set> + <integer set>	<*integer set*>	Returns the union of the specified sets. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
<integer set> = <integer set>	<*boolean*>	Returns TRUE if the specified sets have identical contents. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
<integer set> contains <integer set>	<*boolean*>	Returns TRUE if the first set contains all the elements of the second set. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
<integer set> contains <integer>	<*boolean*>	Returns TRUE if the specified set contains the given integer. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Examples

- `elements of union of (set of (1;2;3); set of (2;3;4); set of (5;6))`
  - Returns the list: 1,2,3,4,5,6.

- `elements of union of (set of (1;2;3); set of (2;3;4))`
  - Returns the list: 1,2,3,4.

- `elements of (set of (1;2;3) - set of (2;3;4))`
  - Returns 1.

- `elements of (set of (1;2;3) * set of (2;3;4))`
  - Returns the list: 2,3.
- `elements of (set of (1;2;3) * set of (2;3;4) * set of (3;4;5))`
  - Returns 3.
- `elements of (set of (1;2;3) + set of (2;3;4))`
  - Returns the list: 1,2,3,4.
- `set of (1;2;3) contains set of (2;3)`
  - Returns TRUE.
- `set of (1;2;3) contains 2`
  - Returns TRUE.

String

A string literal is written within double quotes. Special characters must be inserted by using the percent sign followed by 2 hex digits. Special characters include those characters with ASCII codes less than the 'space' character (hex 20) or greater than 'tilde' character (hex 7f) as well as the percent character itself (25 hex). For example, to create a string containing a null character and a percent character use "a null is %00, the percent itself is %25". Conversion to upper and lower case is also provided. String works in combination with the string position and substring data types. A string position is a point within a string. It can be compared to an integer, but it also acts as a pointer within a string so that the preceding and following text can be extracted. A substring is a part of a larger string. All operations allowed on a string can be performed on a substring. There are two substrings "be" in the string "To be or not to be". The substrings only differ in their positions within the string.

Creation Methods

These string creation methods are in addition to the other properties that return the string type.

Key Phrase	*Form*	Description
<boolean> as string	*Cast*	Operates on a boolean to return a string. Possible values are "True" and "False". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<date> as string	*Cast*	Cast a date type as a string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<hertz> as string	*Cast*	Creates a string containing the number of hertz and the word hertz, e.g., (3 * hz) as string = "3 hertz". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Description
<integer> as hexadecimal	*Cast*	Converts an integer into a hexadecimal string. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
<integer> as string	*Cast*	Creates a string formatted with the integer provided. (-22) as string = "-22". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> as hexadecimal	*Cast*	Converts a string to a hexadecimal number. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
<string> as left trimmed string	*Cast*	Trims the leading spaces from a string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<string> as lowercase	*Cast*	Creates a lowercase version of the string provided. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> as right trimmed string	*Cast*	Trims the trailing spaces from a string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<string> as string	*Cast*	Reflexive cast of string to string. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> as trimmed string	*Cast*	Trims the leading and trailing spaces off of the specified string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<string> as uppercase	*Cast*	Creates an uppercase version of the string provided. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<time interval> as string	*Cast*	Returns a string formatted as • ddd days, HH:MM:SS.mmmmmm • For example, millisecond as string = " 00:00:00.001". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<time zone> as string	*Cast*	Creates a string containing a time zone. See <time zone>. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<time> as local string	*Cast*	Creates a string containing a time. See <time>. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<time> as string	*Cast*	Creates a string containing a time. See <time>. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Description
<time> as universal string	*Cast*	Creates a string containing a time. See <time>. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
character <integer>	*NumberedGlobal*	Creates a string containing the single ASCII character for the decimal number provided. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
concatenation <string> of <string>	*Named*	Combines the second set of strings into a single string, separated by the first string. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
concatenation of <string>	*Plain*	Combines the supplied strings into a single string, end-to-end. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
hexadecimal string <string>	*NamedGlobal*	Creates a string from the given hexadecimal value. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
parameter <string>	*NamedGlobal*	This Inspector is a synonym for the parameter <string> of <action>. It looks up the value of the action parameter specified by <string>. This is used in conjunction with the parameter set command. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
string <string>	*NamedGlobal*	Creates a string matching the name provided. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
two digit hour of <time of day with time zone>	*Plain*	Returns the hour of the zoned time of day as text, with values less than 10 having a leading zero. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
two digit minute of <time of day with time zone>	*Plain*	Returns the minute of the zoned time of day as text, with values less than 10 having a leading zero. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
two digit second of <time of day with time zone>	*Plain*	Returns the second of the zoned time of day as text, with values less than 10 having a leading zero. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Properties

Key Phrase	*Form*	Return Type	Description
<string> as boolean	*Cast*	<boolean>	Returns a boolean value for the string. All possible capitalization's of "TRUE" and "FALSE" will convert successfully. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> as date	*Cast*	<date>	Casts a string as a date type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<string> as day_of_month	*Cast*	<day of month>	Casts a string as a day of the month (eg. 28). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<string> as day_of_week	*Cast*	<day of week>	Casts a string as a day of the week. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<string> as hexadecimal	*Cast*	<string>	Converts a string to a hexadecimal number. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
<string> as integer	*Cast*	<integer>	Returns an integer value for the string provided. If the string contains anything but ASCII digits, the conversion will fail. Use numeric value for more liberal parsing rules. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> as left trimmed string	*Cast*	<string>	Trims the leading spaces from a string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<string> as local time	*Cast*	<time>	Returns a local time object from a properly formatted string. See <time>. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> as local zoned time_of_day	*Cast*	<time of day with time zone>	Converts a string to a time of day with local time zone. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<string> as lowercase	*Cast*	<string>	Returns a lowercase version of the string provided. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> as month	*Cast*	<month>	Converts a string into a month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<string> as right trimmed string	*Cast*	<string>	Trims the trailing spaces from a string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Return Type	Description
<string> as string	*Cast*	<string>	Returns the string provided. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> as time	*Cast*	<time>	Returns a time object from a properly formatted string. See <time>. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> as time interval	*Cast*	<time interval>	Returns a time interval object from a properly formatted string. Expects strings formatted as • ddd days, HH:MM:SS.mmmmmm. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> as time zone	*Cast*	<time zone>	Returns a time zone object from a properly formatted string. See <time zone>. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> as time_of_day	*Cast*	<time of day>	Converts a string to a time_of_day type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<string> as trimmed string	*Cast*	<string>	Trims the leading and trailing spaces off of the specified string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<string> as universal time	*Cast*	<time>	Returns a universal time object from a properly formatted string. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> as universal zoned time_of_day	*Cast*	<time of day with time zone>	Converts a string into a universal zoned time of day. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<string> as uppercase	*Cast*	<string>	Returns an uppercase version of the string provided. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> as version	*Cast*	<version>	Returns a version if the string can be parsed as a version. The first numeric set of characters delimited with period, comma or comma-space is returned. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
<string> as year	*Cast*	<year>	Converts a string into a year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Return Type	Description
<string> as zoned time_of_day	*Cast*	<time of day with time zone>	Converts a string into a zoned time of day. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
character <integer> of <string>	*Numbered*	<substring>	Returns a string of length 1 made by taking the character identified by <integer> from the string. Numbering begins at zero. Example, Character 1 of "HI" is "I". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
character of <string>	*Plain*	<substring>	Returns the characters from the string. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
concatenation <html> of <string>	*Index<html>*	<html>	This is an additional overload of the existing html concatenation operators with the same conventions (string arguments are escaped). Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
concatenation <string> of <string>	*Named*	<string>	Concatenation <string1> of <string2> concatenates a list of strings indicated by string2, placing string1 between each. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
concatenation of <string>	*Plain*	<string>	Combines the supplied strings into a single string, end-to-end. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
first <integer> of <string>	*Numbered*	<substring>	Returns a substring containing the number of characters specified from the given string. For example, First 5 of "To be or not to be" is "To be". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
first <string> of <string>	*Named*	<substring>	Returns a substring containing the first occurrence of the name provided. See substring. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
last <integer> of <string>	*Numbered*	<substring>	Returns a substring containing the number of characters specified. For example, Last 5 of "To be or not to be" is "to be". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Return Type	Description
last <string> of <string>	*Named*	<substring>	Returns a substring containing the last occurrence of the name provided. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
length of <string>	*Plain*	<integer>	Returns the number of characters in the string. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
numeric value of <string>	*Plain*	<integer>	Returns an integer for the first numeric value in the string. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
position <integer> of <string>	*Numbered*	<string position>	Returns a string position pointing to the character position specified. The first character is at position 0. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
position of <string>	*Plain*	<string position>	Returns the positions of the string. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
set of <string>	*Plain*	<string set>	Creates a set from the given list of semicolon-separated strings. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
substring <string> of <string>	*Named*	<substring>	Iterates through the string returning all the substrings matching the name given. For example, number of substrings "be" of "to be or not to be" = 2. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
substring after <string> of <string>	*Named*	<substring>	Returns the substrings that come after the first string delimiter. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
substring before <string> of <string>	*Named*	<substring>	Returns the substrings that come before the first string delimiter. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
substring between <string> of <string>	*Named*	<substring>	Returns the substring in the second string found between two instances of the first string. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
substring separated by <string> of <string>	*Named*	<substring>	Returns a substring (or set of substrings) delimited by the first string. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1

Key Phrase	*Form*	Return Type	Description
unique value of <string>	*Plain*	<string with multiplicity>	Given a list of strings, returns the count of each unique string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Operators

Key phrase	Return Type	Description
<rope> & <string>	<*rope*>	Concatenates a rope and a string, producing a rope. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<rope> contains <string>	<*boolean*>	Returns boolean TRUE if the rope contains the string. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string set> contains <string>	<*boolean*>	Returns TRUE if the specified set of strings contains the given string. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
<string> & <rope>	<*rope*>	Concatenates a rope and a string, returning a new rope. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> & <string>	<*string*>	Concatenates two strings, producing a new string. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> {cmp} <string>	<*boolean*>	Returns a boolean TRUE/FALSE depending on the result of the comparison, where: • {cmp} is one of: =, !=, <, <=, >, >= . Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Note

Many Inspectors return string values from the operating system using a variety of APIs. For the most part, these strings are encoded as single-byte character sets (SBCSs) or multi-byte character sets (MBCSs) depending on the active code page. You can use the code page Inspectors to determine which page is currently active on the client.

Examples

- `255 as hexadecimal`
  - Returns the string "ff".

- `concatenation of "light" & "year"`
  - Returns "lightyear".

- `exists character whose (it is "z") of "Paul Cezanne"`
  - Returns True.

- `concatenation "/" of ("a" ; "b" ; "c" )`
  - Returns "a/b/c".

- `preceding text of last "ab" of "abracadabra" is "abracad"`
  - Returns True.

- `substrings after ":" of "definition: after the colon"`
  - Returns " after the colon".

- `substrings before "<--" of "the item pointed to <--"`
  - Returns "the item pointed to".

- `substrings between "*" of "the item *between* asterisks"`
  - Returns "between".

- `substrings separated by "," of "1,2,3"`
  - Returns the list of numbers separated by commas in the specified string.

- `multiplicities of unique values of ("steak"; "chop"; "rib"; "rib"; "rib")`
  - Returns the multiplicity of (the number of times) each string in the list is used, namely, 1,3,1.

• Note that the multiplicities are based on the alphabetic order of the strings (chop, rib, steak), not their position in the list.

- `set of ("foo";"bar") contains "foo"`
  - Returns TRUE.

String Position

String position works in combination with the string and substring data types. A string position is a point within a string. It can be compared to an integer (which it is derived from), but it also acts as a pointer within a string so that the preceding and following text can be extracted. A substring (a part of a larger string) is derived from a string object.

Creation Methods

Key Phrase	*Form*	Description
end of <substring>	*Plain*	Creates an object corresponding to the position in the string of the end of the substring. For example, end of first "be" of "To be or not to be" = 5. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
position <integer> of <string>	*Numbered*	Creates an index (zero based) into the string. For example, position 5 of "to be or not to be" = 5. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
position of <string>	*Plain*	Iterates through the string returning values for all possible positions within it. For example, number of positions of "hi" = 3. Note that the positions being counted here are 0, 1, and 2. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
start of <substring>	*Plain*	Creates the position of the substring within its containing string. For example, Start of substring "or" of "to be or not to be" = 6. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Properties

Key Phrase	*Form*	Return Type	Description
following text of <string position>	*Plain*	<substring>	Returns the substring following the position in the string. For example, following text of position 5 of "0123456789" = "567890". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
preceding text of <string position>	*Plain*	<substring>	Returns the substring preceding the position in the string. For example, preceding text of position 5 of "0123456789" = "01234". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Note

String positions can be compared and combined with integers using the same operations that are available for integers.

Examples

- `preceding text of position 5 of "Four score and seven.."`
- Returns "Four".

Substring

A substring object is derived from a string object, so it has all the properties of a string. Substrings also have these additional properties:

Creation Methods

Key Phrase	*Form*	Description
character <integer> of <string>	*Numbered*	Creates the single character substring at the position given within the string. For example, character 2 of "abc" = "c". Note that numbering begins at zero. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
character of <string>	*Plain*	Iterates through the string (or substring) returning substrings that contain the individual characters of the string. For example, number of characters of string "abc" = 3. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
first <integer> of <string>	*Numbered*	Creates a substring for the given number of characters at the start of the string. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
first <string> of <string>	*Named*	Creates an object containing the first match of the given string. For example, first "be" of "to be or not to be" = "be". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
following text of <string position>	*Plain*	Creates an object containing the substring following the position in the string. For example, following text of position 5 of "0123456789" = "567890". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Description
following text of <substring>	*Plain*	Creates an object containing the string following the substring. For example, following text of last "." of "log.txt" = "txt". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
last <integer> of <string>	*Numbered*	Creates an object containing a substring from the last part of the string containing the number of characters specified. For example, Last 5 of "To be or not to be" is "to be". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
last <string> of <string>	*Named*	Creates a substring containing the last occurrence of the name provided. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
preceding text of <string position>	*Plain*	Creates the substring preceding the position in the string. For example, preceding text of position 5 of "0123456789" = "01234". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
preceding text of <substring>	*Plain*	Creates an object containing the string preceding the substring. For example, preceding text of last "." of "log.txt" = "log". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
substring <string> of <string>	*Named*	Iterates through the string returning all the substrings matching the name given. For example, number of substrings "be" of "to be or not to be" = 2. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
substring after <string> of <string>	*Named*	Returns the substrings that come after the first string delimiter. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
substring before <string> of <string>	*Named*	Returns the substrings that come before the first string delimiter. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
substring between <string> of <string>	*Named*	Returns the substring in the second string found between two instances of the first string. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
substring separated by <string> of <string>	*Named*	Returns a substring (or set of substrings) delimited by the first string. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1

Properties

Key Phrase	*Form*	Return Type	Description
end of <substring>	*Plain*	<string position>	Returns the position of the substring within its containing string. For example, end of first "be" of "to be or not to be" = 5. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
following text of <substring>	*Plain*	<substring>	Returns the string following the substring. For example, following text of last "." of "log.txt" = "txt". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
preceding text of <substring>	*Plain*	<substring>	Returns the string preceding the substring. For example, preceding text of last "." of "log.txt" = "log". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
start of <substring>	*Plain*	<string position>	Returns the position within the string of the substring. For example, start of substring "or" of "to be or not to be" = 6. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Note

All the string operators can also be applied to substrings.

Examples

- `substrings after ":" of "definition: after the colon"`
  - Returns " after the colon".
- `substrings before "<--" of "the item pointed to <--"`
  - Returns "the item pointed to".
- `substrings between "*" of "the item *between* asterisks"`
  - Returns "between".
- `substrings separated by "," of "1,2,3"`
  - Returns the list of numbers separated by commas in the specified string.

String With Multiplicity

These Inspectors deal with arrays of strings, allowing you to pluck out unique strings and count them.

Creation Methods

Key Phrase	*Form*	Description
unique value of <string>	*Plain*	Given a list of strings, returns the count of each unique string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Properties

Key Phrase	*Form*	Return Type	Description
multiplicity of <string with multiplicity>	*Plain*	<integer>	Returns the multiplicity (quantity) of each element in a multiple string list. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

String Set

These Inspectors deal with sets of strings, which are essentially lists or arrays with string elements. Think of them as mathematical sets: you can compare them, subtract them from other sets and form the union and intersection of multiple sets.

Creation Methods

Key Phrase	*Form*	Description
union of <string set>	*Plain*	Returns a set of strings equal to the union of the specified sets. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Properties

Key Phrase	*Form*	Return Type	Description
element of <string set>	*Plain*	<string>	Typically used in the plural, returns the individual elements of the specified set of strings. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Key Phrase	*Form*	Return Type	Description
intersection of <string set>	*Plain*	<string set>	Returns a set of strings equal to the intersection of the specified sets, aphabetized and with redundant elements stripped out. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
size of <string set>	*Plain*	<integer>	Returns the number of elements in the specified set. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
union of <string set>	*Plain*	<string set>	Returns a set of strings equal to the union of the specified sets, alphabetized and with redundant elements stripped out. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Operators

Key phrase	Return Type	Description
<string set> - <string set>	<*string set*>	Subtracts the elements in the second set from the elements in the first. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
<string set> * <string set>	<*string set*>	Returns the intersection of the specified sets. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
<string set> + <string set>	<*string set*>	Returns the union of the specified sets. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
<string set> = <string set>	<*boolean*>	Returns TRUE if the specified sets have identical contents. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
<string set> contains <string set>	<*boolean*>	Returns TRUE if the first set contains all the elements of the second set. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
<string set> contains <string>	<*boolean*>	Returns TRUE if the specified set of strings contains the given string. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Examples

- `elements of union of (set of ("to";"be"); set of ("or";"not";"to";"be"))`
  - Returns the list: be,not,or,to.

- `elements of union of (set of ("to";"be"); set of ("or";"not";"to";"be"))`
  - Returns the list: be,not,or,to.

- `elements of (set of ("to";"be";"or") - set of ("not";"to";"be"))`
  - Returns or.

- `elements of (set of ("fee";"fie";"foe") - set of ("fee") - set of("foe"))`
  - Return "fie.".

- `elements of (set of ("to";"be";"or") * set of ("not";"to";"be"))`
  - Returns the list: be,to.

- `elements of (set of ("lime";"pie") * set of ("pie";"face") * set of("pie";"in";"sky"))`
  - Returns "pie.".

- `elements of (set of ("to";"be";"or") + set of ("not";"to";"be"))`
  - Returns the list: be,not,or,to.

- `set of ("to";"be";"or";"not") contains set of ("to";"be")`
  - Returns TRUE.

- `set of ("foo";"bar") contains "foo"`
  - Returns TRUE.

Rope

The rope object is a way to efficiently concatenate long strings. String literals in the Relevence language are limited to 512 characters, but internally, they can be any length. Ropes provide a technique for concatenating string literals that is memory-efficient. In general, the Fixlet author will not need to worry about ropes, but they are useful for increasing efficiency.

Creation Methods

Key Phrase	*Form*	Description
rope <string>	*NamedGlobal*	Creates a rope object from the given string. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Properties

Key Phrase	*Form*	Return Type	Description
<rope> as string	*Cast*	<string>	Converts a rope into a string object. Once converted, all the other string properties are available. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
length of <rope>	*Plain*	<integer>	Returns the number of bytes in the rope. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Operators

Key phrase	Return Type	Description
<rope> & <rope>	<*rope*>	Concatenates two ropes into a new rope. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<rope> & <string>	<*rope*>	Concatenates a rope and a string, producing a rope. Win:1.2, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:5.1
<rope> contains <string>	<*boolean*>	Returns TRUE if the rope contains the string. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> & <rope>	<*rope*>	Concatenates a rope and a string, returning a new rope. Win:1.2, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:5.1

Examples

- `rope "Tom" & rope " and " & rope "Jerry"`
  - Returns "Tom and Jerry".

Bit Set

A small, numbered collection of bits that can be examined and manipulated.

Creation Methods

Key Phrase	*Form*	Description
<integer> as bit set	*Cast*	Returns the bits of the binary representation of the integer; bit zero is the least-significant bit. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
<integer> as bits	*Cast*	Returns the bits of the binary representation of the integer; bit zero is the least-significant bit. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
bit <integer>	*NumberedGlobal*	Creates a <bit set> object representing the nth bit position as specified by the integer. The integer value must be between 0 and 63 corresponding to the bit position of interest. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
bit set <string>	*NamedGlobal*	Returns the bits of the binary number given by the string. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
left shift <integer> of <bit set>	*Numbered*	A bit set which, at each position n >= delta, holds bit n-delta of the original bit set, where delta is the given integer. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
right shift <integer> of <bit set>	*Numbered*	A bit set which, at each position n, holds bit n+delta of the original bit set, where delta is the given shift integer. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1

Properties

Key Phrase	*Form*	Return Type	Description
<bit set> as integer	*Cast*	<integer>	Returns the integer whose binary representation matches the bit set. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
<bit set> as string	*Cast*	<string>	Returns the bits (0s and 1s) in a string format. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1

Key Phrase	*Form*	Return Type	Description
bit <integer> of <bit set>	*Numbered*	<boolean>	Returns the value of the bit at the given <integer> position in the set. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
least significant one bit of <bit set>	*Plain*	<integer>	Returns the least n such that bit n of the set is true. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
left shift <integer> of <bit set>	*Numbered*	<bit set>	A bit set which, at each position n >= delta, holds bit n-delta of the original bit set, where delta is the given integer. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
most significant one bit of <bit set>	*Plain*	<integer>	Returns the greatest n such that bit n of the set is true. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
one bit of <bit set>	*Plain*	<integer>	Returns the numbers n for which bit n of the set is true. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
right shift <integer> of <bit set>	*Numbered*	<bit set>	A bit set which, at each position n, holds bit n+delta of the original bit set, where delta is the given shift integer. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1

Operators

Key phrase	Return Type	Description
<bit set> - <bit set>	<*bit set*>	Returns the bits that are true in the left bit set and false in the right bit set. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
<bit set> * <bit set>	<*bit set*>	Returns the intersection of the two bit sets. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
<bit set> + <bit set>	<*bit set*>	Returns the union of the two sets. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
<bit set> = <bit set>	<*boolean*>	Returns true if the corresponding bits of the two sets are equal. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1

Key phrase	Return Type	Description
<bit set> contains <bit set>	<*boolean*>	Returns false if -- for any n -- bit n of the left set is false, but bit n of the right set is true. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1

Examples

- `bit 0 of 5`
- Returns TRUE.

- `bit 3 of bit 3`
- Returns TRUE.

Regular Expression

These Inspectors let you use regular expressions (or regexes) in relevance statements. They use the boost library implementation of the 'POSIX-Extended' regular expression syntax, as documented at:
• http://www.boost.org/libs/regex/doc/syntax_extended.html.

Creation Methods

Key Phrase	*Form*	Description
case insensitive regex <string>	*NamedGlobal*	Creates a case-insensitive regular expression (regex) from the specified string. Win:6.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
case insensitive regular expression <string>	*NamedGlobal*	Same as case insensitive regex <string>. Win:6.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
regex <string>	*NamedGlobal*	Creates a regex object from the given string. Win:6.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
regular expression <string>	*NamedGlobal*	Same as regex <string>. Win:6.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Operators

Key phrase	Return Type	Description
<regular expression> = <string>	<*boolean*>	Returns TRUE if the regular expression is equal to the specified string. Win:6.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> = <regular expression>	<*boolean*>	Returns TRUE if the regular expression is equal to the specified string. Win:6.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> contains <regular expression>	<*boolean*>	Returns TRUE if the specified string contains the contents of the regular expression. Win:6.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> ends with <regular expression>	<*boolean*>	Returns TRUE if the string ends with the contents of the regular expression. Win:6.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> starts with <regular expression>	<*boolean*>	Returns TRUE if the string starts with the contents of the regular expression. Win:6.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Examples

- `regex ".+match.+" = "We will win the match tonight"`
  - Returns TRUE.

- `regex ".*PATH.*" = line 1 of file "/etc/profile"`
  - Returns TRUE if the word "PATH" exists in line 1 of the given file.

Regular Expression Match

These Inspectors let you match regular expressions (or regexes) in relevance statements. They use the boost library implementation of the 'POSIX-Extended' regular expression syntax, as documented at:
• http://www.boost.org/libs/regex/doc/syntax_extended.html. These objects are derived from substring objects.

Creation Methods

Key Phrase	*Form*	Description
first match <regular expression> of <string>	*Index<regular expression>*	Creates an object containing the first match to the regular expression in the given string. Win:6.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
match <regular expression> of <string>	*Index<regular expression>*	Creates an object containing all the matches to the regular expression in the given string. Win:6.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Properties

Key Phrase	*Form*	Return Type	Description
parenthesized part <integer> of <regular expression match>	*Numbered*	<substring>	Returns the nth parenthetical (given by <integer>) in the specified regular expression match. Win:6.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
parenthesized part of <regular expression match>	*Plain*	<substring>	Returns the parenthetical part of the specified regular expression match. Win:6.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Examples

- `first match (regex "to.+") of "just too hot to handle"`
- Returns "too hot to handle".

Undefined

The "undefined" type is used as the result type of Inspectors that never return a value.

Creation Methods

Key Phrase	*Form*	Description
error <string>	*NamedGlobal*	Always fails; if an error message is generated, it is based on the given string. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1

Examples

- `if FALSE then 1 else error "my error message"`
- Returns the string: User-defined error: my error message.

Hertz

The hertz object is useful to measure clock cycles. It is used primarily to measure clock frequency by the speed of the processor Inspector. Hertz objects have a resolution of 1 hertz and are stored internally as a 64 bit signed integer.

Creation Methods

Key Phrase	*Form*	Description
absolute value of <hertz>	*Plain*	Creates a hertz object with a positive value. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
ghz	*PlainGlobal*	Creates a hertz object corresponding to 1 giga-hertz. For example, ghz = 1000*mhz. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
greatest hz	*PlainGlobal*	Creates the largest hertz object that can be represented on the current machine. It returns the value 9,223,372,036,854,775,807 hertz. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
hz	*PlainGlobal*	Creates a hertz object corresponding to 1 hertz. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
khz	*PlainGlobal*	Creates a hertz object corresponding to 1 kilohertz. For example, khz = 1000*hz. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Description
least hz	*PlainGlobal*	Creates the largest negative hertz object that can be represented on the current machine. It returns the value -9,223,372,036,854,775,808 hertz. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
mhz	*PlainGlobal*	Creates a hertz object corresponding to 1 megahertz. For example, mhz = 1000*khz. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
significant digits <integer> of <hertz>	*Numbered*	Rounds up the value of a hertz object with <integer> significant digits. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Properties

Key Phrase	*Form*	Return Type	Description
<hertz> as string	*Cast*	<string>	Returns a string formatted "##### hertz". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
absolute value of <hertz>	*Plain*	<hertz>	Returns the positive value of the hertz object. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
significant digits <integer> of <hertz>	*Numbered*	<hertz>	Returns the value of a hertz object with <integer> significant digits (e.g.. significant digits 3 of 1235569 = 1240000). Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Operators

Key phrase	Return Type	Description
- <hertz>	<*hertz*>	Returns the negative of the <hertz> value. Win:2.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:5.1
<hertz> {cmp} <hertz>	<*boolean*>	Returns a boolean TRUE/FALSE depending on the result of the comparison, where: • {cmp} is one of: =, !=, <, <=, >, >= . Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<hertz> {op} <hertz>	<*hertz*>	Returns a hertz object equal to the result of the operation, where: • {op} is one of: +, -, mod . Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key phrase	Return Type	Description
<hertz> {op} <integer>	<*hertz*>	Returns a hertz object equal to the result of the operation, where: • {op} is one of: *, / . Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Examples

- `speed of processor > 3*ghz`
  - Returns TRUE on machines faster than 3Ghz.

- `greatest hz`
  - Returns a large positive value, such as 9223372036854775807 hertz.

- `least hz`
  - Returns a large negative value, such as -9223372036854775808 hertz.

- `significant digits 3 of 1235569`
  - Returns 1240000.

- `speed of processor`
  - Returns the speed of the processor in hz, such as 3394000000 hertz for a 3.4 GHz computer.

Time

A time object is used to identify a point in time. Time objects are used to represent important properties of objects such as the modification time of a file. You can create time objects from literal strings. The format of the string is defined by the MIME standard. The difference between two Time objects may be calculated by subtracting them and yields time intervals. Time intervals may be added or subtracted from time objects to obtain time objects.

Creation Methods

Key Phrase	*Form*	Description
<string> as local time	*Cast*	Local time creates a time object by parsing the string literal provided. The time zone is optional. If not present, the local time zone is assumed. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<string> as time	*Cast*	Parses the string. Time zone information must be provided. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Description
<string> as universal time	*Cast*	Parses the string. If time zone is not provided in the string, the universal time zone is assumed. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
maximum of <time>	*Plain*	Returns the maximum time from a list of times. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
minimum of <time>	*Plain*	Returns the minimum time from a list of times. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
now	*PlainGlobal*	Creates an object for the current time. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
time <string>	*NamedGlobal*	The time inspector creates a time object by parsing the string literal provided. The zone info is required. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
universal time <string>	*NamedGlobal*	The universal time inspector returns a time object by parsing the string literal provided. The time zone is optional. If not present, universal time is assumed. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Properties

Key Phrase	*Form*	Return Type	Description
<time> as local string	*Cast*	<string>	Returns a string in MIME format of the given time object. The format is: ddd, DD mmm YYYY HH:MM:SS sZZZZ. The string is formatted using the local time zone. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<time> as string	*Cast*	<string>	Same as above. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<time> as universal string	*Cast*	<string>	Returns a string in MIME format of the given time object. The format is: • ddd, DD mmm YYYY HH:MM:SS +0000 • The string is formatted using the universal time zone. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
date <time zone> of <time>	*Index<time zone>*	<date>	Returns the date adjusted for the specified time zone. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Return Type	Description
maximum of <time>	*Plain*	<time>	Returns the maximum time from a list of times. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
minimum of <time>	*Plain*	<time>	Returns the minimum time from a list of times. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
time <time zone> of <time>	*Index<time zone>*	<time of day with time zone>	Adjusts the specified time to the given time zone. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Operators

Key phrase	Return Type	Description
<time interval> & <time>	*<time range>*	Concatenates a time interval with a time, returning a time range of the form time1 to time2. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
<time range> & <time>	*<time range>*	Concatenates a time with a time range, producing a new time range, in the form of: • <date> to <date>. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
<time> & <time interval>	*<time range>*	Concatenates a time and a time interval, producing a time range object. Win:1.2, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
<time> & <time range>	*<time range>*	Concatenates a time and a time range, producing a new time range. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
<time> & <time>	*<time range>*	Concatenates two times into a time range, with the earliest date first and the latest date last. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
<time> {cmp} <time>	*<boolean>*	Returns a boolean TRUE/FALSE depending on the result of the comparison, where: • {cmp} is one of: =, !=, <, <=, >, >= . Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<time> {op} <time interval>	*<time>*	Returns a <time> corresponding to the operator, where: • {op} is one of: +, -. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Note

The string format for a time object is given by the MIME standard. When output as a string, the format used is:
ddd, DD mmm YYYY HH:MM:SS sZZZZ
where:

ddd	The day of the week. Abbreviations are Mon, Tue, Wed, Thu, Fri, Sat, Sun.
DD	The day of the month. A leading zero will be applied to make it two characters wide.
mmm	The Month. Abbreviations are Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec.
YYYY	The year.
HH	The hour of the day. It is always output at two digits. Possible values run from 0 to 23. The digits 00 are used to designate midnight.
MM	The minutes of the hour. It is always output as two digits. Possible values run from 0 to 59.
SS	The seconds of the minute.
s	A single character representing whether the time is east or west of Greenwich. The value of + means east of Greenwich while the value - means west of Greenwich.
ZZZZ	The number of minutes east or west of Greenwich.

To create a string from a literal, use the format:**ddd,DD mmm YYYY HH:MM:SS zoneinfo**
where:

ddd	The optional day of the week. Abbreviations are Mon, Tue, Wed, Thu, Fri, Sat, Sun. Case is not important in these names. If provided it must be correct. For example, time "Sat, 19 jun 1998 00:00:00 +0000" will fail since June 19, 1998 was a Friday.
DD	The day of the month. One or two digits are allowed.
mmm	The Month. Abbreviations are Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec. Case is not important.
YYYY	The year. A two, three or four digit year. If two digits are given a base of 1900 is assumed.
HH	The hour of the day. It is always input at two digits. Possible values run from 0 to 23. The digits 00 are used to designate midnight.
MM	The minutes of the hour. It is always output as two digits. Possible values run from 0 to 59.
SS	The seconds of the minute. Range from 0 to 59. This is optional. If seconds are not present, the preceding colon should also not be present
Zoneinfo	The time zone information. It is provided in one of these formats: Single character + or - followed by 4 digits. The 4 digits are interpreted as HHMM two digits of hours and two digits of minutes. Plus designates east of universal time while minus designates west of universal time. Three letters for the civilian name of the time zone. cdt, edt, mdt, pdt are the designations for central, eastern, mountain and pacific daylight savings time while cst, est, mst, pst are the designations for central, eastern, mountain and pacific standard time. gmt designates Greenwich mean time. A single letter military name of the time zone. Military time zones use single letters from a to z, excepting j. a-m represent offsets from universal time of -1 to -12 hours respectively. z represents 0 offset.

Examples

- `now`
- Returns the current time.

Time Of Day

These Inspectors provide tools for dealing and calculating with time-of-day types, which are of the form HH:MM:SS, as in 12:59:59.

Creation Methods

Key Phrase	*Form*	Description
<string> as time_of_day	*Cast*	Converts a string to a time_of_day type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
midnight	*PlainGlobal*	Returns 00:00:00 as a time of day object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
noon	*PlainGlobal*	Returns 12:00:00 as a time of day object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
time of <time of day with time zone>	*Plain*	Returns the time of day, without the time zone information. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
time_of_day <string>	*NamedGlobal*	Creates a time of day object out of the given string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Properties

Key Phrase	*Form*	Return Type	Description
<time of day> as string	*Cast*	<string>	Casts the time of day as a string type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
hour_of_day of <time of day>	*Plain*	<integer>	Returns the hour section of the 'time of day' object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
minute_of_hour of <time of day>	*Plain*	<integer>	Returns the 'minutes after the hour' section of the 'time of day' object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Return Type	Description
second_of_minute of <time of day>	*Plain*	<integer>	Extracts the 'seconds after the minute' section of the 'tim of day' object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
two digit hour of <time of day>	*Plain*	<string>	Extracts the 2-digit hour from the time of day. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
two digit minute of <time of day>	*Plain*	<string>	Extracts the 2-digit minute from the time of day. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
two digit second of <time of day>	*Plain*	<string>	Extracts the 2-digit second from the time of day. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Operators

Key phrase	Return Type	Description
<time of day> - <time of day>	*<time interval>*	Subtracts two times of day, returning a time interval. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<time of day> & <time zone>	*<time of day with time zone>*	Concatenates a time of day with a time zone, returning a time of day with time zone type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<time zone> & <time of day>	*<time of day with time zone>*	Concatenates a time of day with a time zone, returning a time of day with time zone type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Time Zone

Time zones are used in conjunction with the time object. Time zones have a resolution of 1 minute.

Creation Methods

Key Phrase	*Form*	Description
<string> as time zone	*Cast*	Creates a time zone object corresponding to the string provided. For example, "pst" as time zone. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
local time zone	*PlainGlobal*	Creates a time zone object corresponding to the local time zone. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Description
time zone <string>	*NamedGlobal*	Creates a time zone object corresponding to the string provided. For example, time zone "edt" as string = "-0400". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
universal time zone	*PlainGlobal*	Creates a time zone object corresponding to the universal time zone. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Properties

Key Phrase	*Form*	Return Type	Description
<time zone> as string	*Cast*	<string>	Returns a string corresponding to the time zone object provided. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Operators

Key phrase	Return Type	Description
<time of day> & <time zone>	*<time of day with time zone>*	Concatenates a time of day with a time zone, returning a time of day with time zone type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<time zone> & <time of day with time zone>	*<time of day with time zone>*	Converts a 'time of day with time zone' to the time in the specified time zone. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<time zone> & <time of day>	*<time of day with time zone>*	Concatenates a time of day with a time zone, returning a time of day with time zone type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<time zone> {op} <time interval>	*<time zone>*	Returns a time zone object offset by a time interval, where: • {op} is one of: +, - . Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Examples

- `local time zone - 2 * hour`
- Returns the time zone two hours away.

Time Of Day With Time Zone

These Inspectors provide tools for dealing and calculating with time-of-day-with-time-zone types, which are of the form HH:MM:SS +ZZZZ, as in 12:59:59 -0400.

Creation Methods

Key Phrase	*Form*	Description
<string> as local zoned time_of_day	*Cast*	Converts a string to a time of day with local time zone. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<string> as universal zoned time_of_day	*Cast*	Converts a string into a universal zoned time of day. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<string> as zoned time_of_day	*Cast*	Converts a string into a zoned time of day. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
current time_of_day	*PlainGlobal*	Returns the current time of day in the local time zone. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
current time_of_day <time zone>	*Index<time zone>Global*	Returns the current time of day in the specified time zone. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
time <time zone> of <time>	*Index<time zone>*	Converts the specified time to the given time zone. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
zoned time_of_day <string>	*NamedGlobal*	Creates a 'zoned time of day' out of a string object in the form of HH:MM:SS +ZZZZ. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Properties

Key Phrase	*Form*	Return Type	Description
<time of day with time zone> as string	*Cast*	<string>	Converts a 'time of day with time zone' object into a string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
hour_of_day of <time of day with time zone>	*Plain*	<integer>	Returns the hour section of the 'time of day with time zone' object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Return Type	Description
minute_of_hour of <time of day with time zone>	*Plain*	<integer>	Returns the 'minutes after the hour' section of the 'time of day with time zone' object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
second_of_minute of <time of day with time zone>	*Plain*	<integer>	Returns the 'seconds after the minute' section of the 'time of day with time zone' object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
time of <time of day with time zone>	*Plain*	<time of day>	Returns the time of day, without the time zone information. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
two digit hour of <time of day with time zone>	*Plain*	<string>	Returns the hour of the zoned time of day as text, with values less than 10 having a leading zero. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
two digit minute of <time of day with time zone>	*Plain*	<string>	Returns the minute of the zoned time of day as text, with values less than 10 having a leading zero. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
two digit second of <time of day with time zone>	*Plain*	<string>	Returns the second of the zoned time of day as text, with values less than 10 having a leading zero. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
zone of <time of day with time zone>	*Plain*	<time zone>	Returns the zone associated with the specified time. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Operators

Key phrase	Return Type	Description
<date> & <time of day with time zone>	*<time>*	Concatenates a date with a time and a time zone for a complete time stamp. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<time of day with time zone> - <time of day with time zone>	*<time interval>*	Subtracts two times of day (including time zones), returning a time interval. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key phrase	Return Type	Description
<time of day with time zone> & <time zone>	*<time of day with time zone>*	Concatenates a 'time of day with a time zone' and another time zone. The 'time of day with time zone' object that is produced is adjusted to fit the appended time zone. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<time of day with time zone> = <time of day with time zone>	*<boolean>*	Compares two times of day with time zone. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<time zone> & <time of day with time zone>	*<time of day with time zone>*	Converts a 'time of day with time zone' to the time in the specified time zone. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Time Range

These Inspectors provide tools for dealing and calculating with time-range types, which are of the form <time> to <time>, such as Tue, 18 Apr 2006 16:46:07 -0400 to Wed, 19 Apr 2006 16:46:07 -0400

Creation Methods

Key Phrase	*Form*	Description
final part <time interval> of <time range>	*Index<time interval>*	Returns a time range with the specified interval, but ending on the final date of the time range. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
initial part <time interval> of <time range>	*Index<time interval>*	Returns a time range starting with the first date of the time range and lasting for the specified interval. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
range after <time> of <time range>	*Index<time>*	Returns a new time range, starting from the specified time and continuing through the end of the original range. The time must be within the range, or an error will result. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
range before <time> of <time range>	*Index<time>*	Returns a new time range, starting from the original time in the specified range and continuting to the specified time. The time must be within the range, or an error will result. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Properties

Key Phrase	*Form*	Return Type	Description
<time range> as string	*Cast*	<string>	Casts a time range as a string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
end of <time range>	*Plain*	<time>	Returns the end date of a time range. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
final part <time interval> of <time range>	*Index<time interval>*	<time range>	Returns a time range with the specified interval, but ending on the final date of the time range. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
initial part <time interval> of <time range>	*Index<time interval>*	<time range>	Returns a time range starting with the first date of the time range and lasting for the specified interval. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
length of <time range>	*Plain*	<time interval>	Returns the time interval (in days, hours, minutes, seconds) between the start and end date of a time range. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
range after <time> of <time range>	*Index<time>*	<time range>	Returns a new time range, starting from the specified time and continuing through the end of the original range. The time must be within the range, or an error will result. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
range before <time> of <time range>	*Index<time>*	<time range>	Returns a new time range, starting from the original time in the specified range and continuting to the specified time. The time must be within the range, or an error will result. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
start of <time range>	*Plain*	<time>	Returns the starting date of a time range. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Operators

Key phrase	Return Type	Description
<time range> & <time range>	*<time range>*	Returns the smallest range that contains both ot the specified ranges (same as <time range> + <time range>). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
<time range> & <time>	*<time range>*	Concatenates a time with a time range, producing a new time range, in the form of: • <date> to <date>. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
<time range> * <time range>	*<time range>*	Returns the intersection of the two specified time ranges, if one exists. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
<time range> + <time range>	*<time range>*	Returns the smallest range that contains both ot the specified ranges (same as <time range> & <time range>). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
<time> & <time range>	*<time range>*	Concatenates a time and a time range, producing a new time range. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Time Interval

Time intervals are used in conjunction with the time object. Time intervals have a resolution of 1 microsecond.

Creation Methods

Key Phrase	*Form*	Description
<string> as time interval	*Cast*	Returns a time interval object from a properly formatted string. Expects strings formatted as • ddd days, HH:MM:SS.mmmmmm. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
absolute value of <time interval>	*Plain*	Creates the positive value of a time interval. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Description
day	*PlainGlobal*	Creates a time interval corresponding to 1 day. For example, 2 * day = 48 * hour. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
greatest time interval	*PlainGlobal*	Creates the largest time interval that can be represented on the current machine. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
hour	*PlainGlobal*	Creates a time interval corresponding to 1 hour. For example, day = 24 * hour. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
least time interval	*PlainGlobal*	Creates the largest negative time interval that can be represented on the current machine. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
maximum of <time interval>	*Plain*	Returns the maximum interval from a list of time intervals. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
microsecond	*PlainGlobal*	Creates a time interval corresponding to 1 microsecond. For example, 1000 * microsecond = 1 * millisecond. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
millisecond	*PlainGlobal*	Creates a time interval corresponding to 1 millisecond. For example, 1000 * millisecond = 1 * second. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
minimum of <time interval>	*Plain*	Returns the minimum interval from a list of time intervals. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
minute	*PlainGlobal*	Creates a time interval corresponding to 1 minute. For example, minute = 60 * second. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
second	*PlainGlobal*	Creates a time interval corresponding to 1 second. For example, 1000000 * microsecond = second. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
time interval <string>	*NamedGlobal*	Creates a time interval from the string. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Description
week	*PlainGlobal*	Creates a time interval corresponding to 1 week. For example, 7*day = 1*week. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Properties

Key Phrase	*Form*	Return Type	Description
<time interval> as string	*Cast*	<string>	Returns a string formatted as • ddd days, HH:MM:SS.mmmmmm • For example, millisecond as string = " 00:00:00.001". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
absolute value of <time interval>	*Plain*	<time interval>	Returns positive value of the time interval. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
maximum of <time interval>	*Plain*	<time interval>	Returns the maximum interval from a list of time intervals. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
minimum of <time interval>	*Plain*	<time interval>	Returns the minimum interval from a list of time intervals. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Operators

Key phrase	Return Type	Description
- <time interval>	*<time interval>*	The negative of a time interval. Win:2.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:5.1
<time interval> & <time>	*<time range>*	Concatenates a time interval with a time, returning a time range of the form time1 to time2. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
<time interval> {op} <integer>	*<time interval>*	Creates a time interval calculated as an integer operation on another time interval, where: • {op} is one of: *, / . Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<time interval> {op} <time interval>	*<time interval>*	Returns a calculated time interval, where: • {op} is one of: +, -, mod, / . Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key phrase	Return Type	Description
<time interval> {op} <time zone>	*<time interval>*	Returns a calculated time interval, where: • {op} is one of: +, - . Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<time interval> + <time of day with time zone>	*<time of day with time zone>*	Adds a time interval (days, hours, minutes, seconds) to a time of the day with time zone to create a new time of the day with time zone. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<time interval> + <time of day>	*<time of day>*	Adds a time interval (days, hours, minutes, seconds) to a time of the day to create a new time of the day. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<time> & <time interval>	*<time range>*	Concatenates a time and a time interval, producing a time range object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Date

These are the various Inspectors that access the date types.

Creation Methods

Key Phrase	*Form*	Description
<string> as date	*Cast*	Casts a string as a date type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
april <integer> of <integer>	*Numbered*	Returns the nth day of april and the specified year as a date (day of week, month day year). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
august <integer> of <integer>	*Numbered*	Returns the nth day of August and the specified year as a date (day of week, month day year). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
current date	*PlainGlobal*	Returns the current date in the format: • Day of week, Day Month Year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
date <string>	*NamedGlobal*	Converts the given string into a date. The string should be of the form 'Day Month Year' and the returned date will be of the form 'Day of week, Day Month Year'. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Description
date <time zone> of <time>	*Index<time zone>*	Returns the date adjusted for the specified time zone. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
december <integer> of <integer>	*Numbered*	Returns the nth day of December and the specified year as a date (day of week, month day year). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
february <integer> of <integer>	*Numbered*	Returns the nth day of February and the specified year as a date (day of week, month day year). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
january <integer> of <integer>	*Numbered*	Returns the nth day of January and the specified year as a date (day of week, month day year). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
july <integer> of <integer>	*Numbered*	Returns the nth day of July and the specified year as a date (day of week, month day year). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
june <integer> of <integer>	*Numbered*	Returns the nth day of June and the specified year as a date (day of week, month day year). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
march <integer> of <integer>	*Numbered*	Returns the nth day of March and the specified year as a date (day of week, month day year). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
may <integer> of <integer>	*Numbered*	Returns the nth day of May and the specified year as a date (day of week, month day year). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
november <integer> of <integer>	*Numbered*	Returns the nth day of November and the specified year as a date (day of week, month day year). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
october <integer> of <integer>	*Numbered*	Returns the nth day of October and the specified year as a date (day of week, month day year). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
september <integer> of <integer>	*Numbered*	Returns the nth day of September and the specified year as a date (day of week, month day year). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Properties

Key Phrase	*Form*	Return Type	Description
<date> as string	*Cast*	<string>	Cast a date type as a string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
day_of_month of <date>	*Plain*	<day of month>	Extracts the day of the month from the specified date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
day_of_week of <date>	*Plain*	<day of week>	Extracts the day of the week (Monday, Tuesday, etc.) from the specified date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
day_of_year of <date>	*Plain*	<day of year>	Extracts the day of year from the specified date, in the 'Month Day' format. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
month of <date>	*Plain*	<month>	Returns the month derived from the given date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
month_and_year of <date>	*Plain*	<month and year>	Formats the specified date in month year format, eg. March 2012. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
year of <date>	*Plain*	<year>	Returns the year, extracted from the given date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Operators

Key phrase	Return Type	Description
<date> - <date>	*<time interval>*	Subtracts two dates to produce a time interval. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<date> & <time of day with time zone>	*<time>*	Concatenates a date with a time and a time zone for a complete time stamp. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<number of months> + <date>	*<date>*	Adds a number of months to a date, returning a new date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<time interval> + <date>	*<date>*	Adds a time interval (days, hours, minutes, seconds) to a date to create a new date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key phrase	Return Type	Description
<time of day with time zone> & <date>	*<time>*	Concatenates a 'time of day with time zone' object with a date object to produce a time object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Day Of Week

These Inspectors provide tools for dealing and calculating with day-of-week types, which include Monday, Tuesday, etc. This set of Inspectors includes each day as a self-named object.

Creation Methods

Key Phrase	*Form*	Description
<string> as day_of_week	*Cast*	Casts a string as a day of the week. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
current day_of_week	*PlainGlobal*	Retruns the current day of the week, eg. Monday, Tuesday, etc. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
day_of_week <string>	*NamedGlobal*	Converts the given string value to a day of week type, eg. Monday, Tuesday, etc. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
day_of_week of <date>	*Plain*	Extracts the day of the week from the specified date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
friday	*PlainGlobal*	Returns Friday as a day of week object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
monday	*PlainGlobal*	Returns the day of week object for Monday. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
saturday	*PlainGlobal*	Returns Saturday as a day of week object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
sunday	*PlainGlobal*	Returns Sunday as a day of week object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
thursday	*PlainGlobal*	Returns Thursday as a day of week object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
tuesday	*PlainGlobal*	Returns Tuesday as a day of week object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Description
wednesday	*PlainGlobal*	Returns Wednesday as a day of week object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Properties

Key Phrase	*Form*	Return Type	Description
<day of week> as string	*Cast*	<string>	Casts the day of week as a string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<day of week> as three letters	*Cast*	<string>	Casts the day of week as a three-letter abbreviation (Mon, Tue, etc.). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Operators

Key phrase	Return Type	Description
<day of week> - <day of week>	*<time interval>*	Subtract two day of week types (Monday, Tuesday, etc.) to produce a time interval. The answer cannot exceed 6 days. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<day of week> = <day of week>	*<boolean>*	Compares two days of the week and returns a boolean TRUE or FALSE. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<time interval> + <day of week>	*<day of week>*	Adds a time interval (days, hours, minutes, seconds) to a day of the week to create a new day of the week. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Day Of Month

These Inspectors provide tools for dealing and calculating with day-of-month types, which are numbers from 1-31.

Creation Methods

Key Phrase	*Form*	Description
<integer> as day_of_month	*Cast*	Cast an integer as a day of the month type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<string> as day_of_month	*Cast*	Casts a string as a day of month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
current day_of_month	*PlainGlobal*	Returns the current day of the month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
day of <day of year>	*Plain*	Returns the day of the month of the specified date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
day_of_month <integer>	*NumberedGlobal*	Converts the given integer to a day of month type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
day_of_month <string>	*NamedGlobal*	Converts the given string value (must be an integer from 1-31) to a day of month type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
day_of_month of <date>	*Plain*	Extracts the day of the month from the specified date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Properties

Key Phrase	*Form*	Return Type	Description
<day of month> as integer	*Cast*	<integer>	Cast a day of month type as an integer. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<day of month> as string	*Cast*	<string>	Cast a day of month type as a string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<day of month> as two digits	*Cast*	<string>	Cast a day of month type as a two-digit number. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Operators

Key phrase	Return Type	Description
<day of month> - <day of month>	*<time interval>*	Subtract two day of month types, producing a time interval. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<day of month> & <month and year>	*<date>*	Concatenate a day of month with a month and year type to produce a complete date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<day of month> & <month>	*<day of year>*	Concatenate a day of month with a month type to produce a day of year (eg. April 20). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<month and year> & <day of month>	*<date>*	Concatenates a month and year with a day of month to produce a complete date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<month> & <day of month>	*<day of year>*	Concatenates a month and a day of the month to produce a day of year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<time interval> + <day of month>	*<day of month>*	Adds a time interval (days, hours, minutes, seconds) to a day of the month to create a new day of the month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Day Of Year

These Inspectors provide tools for dealing and calculating with day-of-month types, which are of the form Sun, 01 Apr 2007.

Creation Methods

Key Phrase	*Form*	Description
april <integer>	*NumberedGlobal*	Returns the nth day of april as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
august <integer>	*NumberedGlobal*	Returns the nth day of August as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Description
current day_of_year	*PlainGlobal*	Retruns the current day of the year, in a Month Day format. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
day_of_year of <date>	*Plain*	Extracts the day of year from the specified date, in the 'Month Day' format. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
december <integer>	*NumberedGlobal*	Returns the nth day of December as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
february <integer>	*NumberedGlobal*	Returns the nth day of February as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
january <integer>	*NumberedGlobal*	Returns the nth day of January as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
july <integer>	*NumberedGlobal*	Returns the nth day of July as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
june <integer>	*NumberedGlobal*	Returns the nth day of June as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
march <integer>	*NumberedGlobal*	Returns the nth day of March as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
may <integer>	*NumberedGlobal*	Returns the nth day of May as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
november <integer>	*NumberedGlobal*	Returns the nth day of November as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
october <integer>	*NumberedGlobal*	Returns the nth day of October as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Description
september <integer>	*NumberedGlobal*	Returns the nth day of September as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Properties

Key Phrase	*Form*	Return Type	Description
<day of year> as string	*Cast*	<string>	Casts a day of the year as a string type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
day of <day of year>	*Plain*	<day of month>	Returns the day of the month of the specified date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
month of <day of year>	*Plain*	<month>	Returns the month portion of the given date (in month day format). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Operators

Key phrase	Return Type	Description
<day of year> - <day of year>	*<time interval>*	Subtracts two days of the year to produce a time interval. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<day of year> & <month and year>	*<date>*	Concatenates a day of the year with a month and year to create a complete date type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<day of year> & <year>	*<date>*	Concatenates a day of the year with a year to create a complete date type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<month and year> & <day of year>	*<date>*	Concatenates a month and year with a day of year to produce a complete date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<number of months> + <day of year>	*<day of year>*	Adds a number of months to a day of the year (July 4, say) to produce another day of the year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key phrase	Return Type	Description
<time interval> + <day of year>	*<day of year>*	Adds a time interval (days, hours, minutes, seconds) to a day of the year to create a new day of the year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<year> & <day of year>	*<date>*	Concatenates a year with the day of the year, returning a full date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Month

These Inspectors provide tools for dealing and calculating with month types, which are of the form January, February, etc. This set of Inspectors includes each month as a self-named object.

Creation Methods

Key Phrase	*Form*	Description
<integer> as month	*Cast*	Returns the name of the nth month of the year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<string> as month	*Cast*	Converts a string into a month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
april	*PlainGlobal*	Returns april as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
august	*PlainGlobal*	Returns August as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
current month	*PlainGlobal*	Returns the current month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
december	*PlainGlobal*	Returns December as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
february	*PlainGlobal*	Returns February as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
january	*PlainGlobal*	Returns January as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
july	*PlainGlobal*	Returns July as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Description
june	*PlainGlobal*	Returns June as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
march	*PlainGlobal*	Returns March as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
may	*PlainGlobal*	Returns May as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
month <integer>	*NumberedGlobal*	Returns the month type corresponding to the given <integer>. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
month <string>	*NamedGlobal*	Returns a month type corresponding to the given <string>. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
month of <date>	*Plain*	Returns the month of the given date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
month of <day of year>	*Plain*	Returns the month portion of the given date (in month day format). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
month of <month and year>	*Plain*	Returns the month portion of the given date (in month year format). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
november	*PlainGlobal*	Returns November as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
october	*PlainGlobal*	Returns October as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
september	*PlainGlobal*	Returns September as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Properties

Key Phrase	*Form*	Return Type	Description
<month> as integer	*Cast*	<integer>	Converts the given month into an integer (1-12). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<month> as string	*Cast*	<string>	Converts the given month into a string value. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<month> as three letters	*Cast*	<string>	Converts the given month into a 3-letter string (Jan, Feb, etc.). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<month> as two digits	*Cast*	<string>	Converts the month into a two digit number (01 - 12). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Operators

Key phrase	Return Type	Description
<day of month> & <month>	*<day of year>*	Concatenate a day of month with a month type to produce a day of year (eg. April 20). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<month> - <month>	*<number of months>*	Subtracts two months, returning a positive number of months. If the first month is earlier than the second, it assumes the year has rolled over. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<month> & <day of month>	*<day of year>*	Concatenates a month and a day of the month to produce a day of year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<month> & <year>	*<month and year>*	Returns a date (in month year format) from the concatenation of a month and a year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<number of months> + <month>	*<month>*	Adds a number of months to the given month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<year> & <month>	*<month and year>*	Returns a date (in month year format) from the concatenation of a month and a year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Month And Year

These Inspectors provide tools for dealing and calculating with month-and-year types, which are of the form month of year, eg., January of 2007.

Creation Methods

Key Phrase	*Form*	Description
april of <integer>	*Plain*	Creates a date (in month year format) corresponding to april of the specified year (as an <integer>). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
august of <integer>	*Plain*	Creates a date (in month year format) corresponding to August of the specified year (as an <integer>). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
current month_and_year	*PlainGlobal*	Returns the current date in month year format, eg. January 2012. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
december of <integer>	*Plain*	Creates a date (in month year format) corresponding to December of the specified year (as an <integer>). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
february of <integer>	*Plain*	Creates a date (in month year format) corresponding to February of the specified year (as an <integer>). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
january of <integer>	*Plain*	Creates a date (in month year format) corresponding to January of the specified year (as an <integer>). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
july of <integer>	*Plain*	Creates a date (in month year format) corresponding to July of the specified year (as an <integer>). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
june of <integer>	*Plain*	Creates a date (in month year format) corresponding to June of the specified year (as an <integer>). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
march of <integer>	*Plain*	Creates a date (in month year format) corresponding to March of the specified year (as an <integer>). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
may of <integer>	*Plain*	Creates a date (in month year format) corresponding to May of the specified year (as an <integer>). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Description
month_and_year of <date>	*Plain*	Formats the specified date in month year format, eg. March 2012. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
november of <integer>	*Plain*	Creates a date (in month year format) corresponding to November of the specified year (as an <integer>). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
october of <integer>	*Plain*	Creates a date (in month year format) corresponding to October of the specified year (as an <integer>). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
september of <integer>	*Plain*	Creates a date (in month year format) corresponding to September of the specified year (as an <integer>). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Properties

Key Phrase	*Form*	Return Type	Description
<month and year> as string	*Cast*	<string>	Casts a date (in month year format) as a string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
first <day of week> of <month and year>	*Index<day of week>*	<date>	Finds the specific date corresponding to the first day of the week (eg. Friday) for a given month and year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
first friday of <month and year>	*Plain*	<date>	Finds the date corresponding to the first Friday of any given month and year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
first monday of <month and year>	*Plain*	<date>	Finds the date corresponding to the first Monday of any given month and year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
first saturday of <month and year>	*Plain*	<date>	Finds the date corresponding to the first Saturday of any given month and year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
first sunday of <month and year>	*Plain*	<date>	Finds the date corresponding to the first Sunday of any given month and year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Return Type	Description
first thursday of <month and year>	*Plain*	<date>	Finds the date corresponding to the first Thursday of any given month and year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
first tuesday of <month and year>	*Plain*	<date>	Finds the date corresponding to the first Tuesday of any given month and year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
first wednesday of <month and year>	*Plain*	<date>	Finds the date corresponding to the first Wednesday of any given month and year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
length of <month and year>	*Plain*	<time interval>	Returns the number of days in the specified month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
month of <month and year>	*Plain*	<month>	Returns the name of the month corresponding to the given date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
year of <month and year>	*Plain*	<year>	Returns the year portion of the specified date (in month year format). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Operators

Key phrase	Return Type	Description
<day of month> & <month and year>	*<date>*	Concatenate a day of month with a month and year type to produce a complete date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<day of year> & <month and year>	*<date>*	Concatenates a day of the year with a month and year to create a complete date type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<month and year> - <month and year>	*<number of months>*	Subtracts two dates (in month year format), returning a number of months. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<number of months> + <month and year>	*<month and year>*	Adds a number of months to a given date (in month year format) producing a new date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Number Of Months

These Inspectors provide tools for dealing and calculating with number-of-month types, which are similar to integers, but wih yearly roll-over.

Creation Methods

Key Phrase	*Form*	Description
month	*PlainGlobal*	Returns the specified number of months. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
year	*PlainGlobal*	Returns the specified number of years as a <number of months> type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Properties

Key Phrase	*Form*	Return Type	Description
<number of months> as string	*Cast*	<string>	Converts a number of months type into a string type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Operators

Key phrase	Return Type	Description
- <number of months>	*<number of months>*	Creates the negative of the specified number of months. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<number of months> + <year>	*<year>*	Returns the year after adding the specified number of months. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Year

These Inspectors provide tools for dealing and calculating with year types, which are of the form YYYY, as in 2008.

Creation Methods

Key Phrase	*Form*	Description
<integer> as year	*Cast*	Casts an integer as a year type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<string> as year	*Cast*	Converts a string into a year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
current year	*PlainGlobal*	Returns the current year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
year <integer>	*NumberedGlobal*	Creates a year object from the specified integer. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
year <string>	*NamedGlobal*	Creates a year object from the specified string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
year of <date>	*Plain*	The year derived from the given date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
year of <month and year>	*Plain*	Returns the year portion of the specified date (in month year format). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Properties

Key Phrase	*Form*	Return Type	Description
<year> as integer	*Cast*	<integer>	Casts a year as an integer. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<year> as string	*Cast*	<string>	Casts a year as a string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
leap of <year>	*Plain*	<boolean>	Returns a flag indicating whether or not the specified year is a leap year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
length of <year>	*Plain*	<time interval>	Returns the number of day in the specified year. Leap years have 366 days. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Operators

Key phrase	Return Type	Description
<day of year> & <year>	*<date>*	Concatenates a day of the year with a year to create a complete date type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<month> & <year>	*<month and year>*	Returns a date (in month year format) from the concatenation of a month and a year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<year> - <year>	*<number of months>*	Subtracts two years and produces a time interval marked in months and years. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<year> & <day of year>	*<date>*	Concatenates a year with the day of the year, returning a full date. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
<year> & <month>	*<month and year>*	Returns a date (in month year format) from the concatenation of a month and a year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

World Objects

World

All objects created without context are known as 'properties of the world' in the relevance language. Below is a list of these global properties, sorted by key phrase.

Properties

Key Phrase	*Form*	Return Type	Description
action	*PlainGlobal*	<action>	Returns the action currently being parsed. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
action <integer>	*NumberedGlobal*	<action>	Returns the action matching the <integer> id. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
action lock state	*PlainGlobal*	<action lock state>	Returns the client action lock state. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
active action	*PlainGlobal*	<action>	Returns the action currently executing. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
apparent registration server time	*PlainGlobal*	<time>	Shorthand for 'now of registration server'. When the client registers with the server, the server passes its current time back to the client. The client starts a stop watch at that time. The apparent registration server time is the time the server passed back to the client, plus the elapsed time on the stop watch. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
application <string>	*NamedGlobal*	<application>	Creates an application object for the name provided. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
application usage summary	*PlainGlobal*	<application usage summary>	Returns an application usage summary containing information including the start time, duration and other statistics on client applications. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Key Phrase	*Form*	Return Type	Description
application usage summary <string>	*NamedGlobal*	<application usage summary>	Returns the usage summary for the application specified in <string>. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
april	*PlainGlobal*	<month>	Returns april as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
april <integer>	*NumberedGlobal*	<day of year>	Returns the nth day of april as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
august	*PlainGlobal*	<month>	Returns August as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
august <integer>	*NumberedGlobal*	<day of year>	Returns the nth day of August as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
bes license	*PlainGlobal*	<license>	Synonym for 'client license'. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
binary operator <string>	*NamedGlobal*	<binary operator>	Typically used in the plural, returns the various possible binary inspectors that use the specified operators. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
binary operator returning <type>	*Index<type>Global*	<binary operator>	Returns a list of binary operators that return the specified type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
bit <integer>	*NumberedGlobal*	<bit set>	Returns TRUE or FALSE, corresponding to value of the bit specified by <integer>. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
bit set <string>	*NamedGlobal*	<bit set>	Returns the bits of the binary number given by the string. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
boolean <string>	*NamedGlobal*	<boolean>	Returns a boolean. For example, boolean "TRUE". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Return Type	Description
case insensitive regex <string>	*NamedGlobal*	<regular expression>	Returns a case-insensitive regular expression from the supplied string. Win:6.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
case insensitive regular expression <string>	*NamedGlobal*	<regular expression>	Same as case insensitive regex <string>. Win:6.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
cast <string>	*NamedGlobal*	<cast>	Returns a list of the objects that can be cast into the type specified by <string>. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
cast returning <type>	*Index<type>Global*	<cast>	Returns a list of the objects that can be cast into the specified type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
character <integer>	*NumberedGlobal*	<string>	Returns a string containing a single ASCII character. For example, character 90 = "Z". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
client	*PlainGlobal*	<client>	Returns the client object corresponding to the BigFix application evaluating the current relevance expression. Win:3.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
client license	*PlainGlobal*	<license>	Global object containing client licensing information. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
computer id	*PlainGlobal*	<integer>	This is a unique integer assigned to the computer by the BES system. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
computer name	*PlainGlobal*	<string>	Returns a string corresponding to the name of the computer as it appears on the network. Win:3.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
current date	*PlainGlobal*	<date>	Returns the current date in the format: • Day of week, Day Month Year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
current day_of_month	*PlainGlobal*	<day of month>	Returns the current day of the month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Return Type	Description
current day_of_week	*PlainGlobal*	<day of week>	Retruns the current day of the week, eg. Monday, Tuesday, etc. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
current day_of_year	*PlainGlobal*	<day of year>	Retruns the current day of the year, in a Month Day format. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
current month	*PlainGlobal*	<month>	Returns the current month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
current month_and_year	*PlainGlobal*	<month and year>	Returns the current date in month year format, eg. January 2012. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
current relay	*PlainGlobal*	<current relay>	Returns an object corresponding to the server or relay that the client last registered with. This may be a BES Relay or the BES root server. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
current site	*PlainGlobal*	<site>	Returns the current site object. See site. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
current time_of_day	*PlainGlobal*	<time of day with time zone>	Returns the current time of day in the local time zone. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
current time_of_day <time zone>	*Index<time zone>Global*	<time of day with time zone>	Returns the current time of day in the specified time zone. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
current user	*PlainGlobal*	<current user>	Returns the current user if one is logged in to the desktop. • Note: For Unix, this returns a <user> type as of BES version 6.0. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
current user	*PlainGlobal*	<user>	Returns the current <user> if one is logged in to the desktop. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
current year	*PlainGlobal*	<year>	Returns the current year. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Return Type	Description
custom site subscription effective date <string>	*NamedGlobal*	<time>	Returns the date the custom site (specified by <string>) was last subscribed or unsubscribed. It is used internally by BES to manage custom site subscriptions. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
date <string>	*NamedGlobal*	<date>	Converts the given string into a date. The string should be of the form 'Day Month Year' and the returned date will be of the form 'Day of week, Day Month Year'. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
day	*PlainGlobal*	<time interval>	Returns a time interval corresponding to 1 day. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
day_of_month <integer>	*NumberedGlobal*	<day of month>	Converts the given integer to a day of month type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
day_of_month <string>	*NamedGlobal*	<day of month>	Converts the given string value (must be an integer from 1-31) to a day of month type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
day_of_week <string>	*NamedGlobal*	<day of week>	Converts the given string value to a day of week type, eg. Monday, Tuesday, etc. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
december	*PlainGlobal*	<month>	Returns December as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
december <integer>	*NumberedGlobal*	<day of year>	Returns the nth day of December as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
default web browser	*PlainGlobal*	<file>	Creates an object corresponding to the default web browser. This is a Windows and Macintosh inspector; it will fail gracefully on other operating systems instead of generating an error. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Return Type	Description
dns name	*PlainGlobal*	<string>	Returns the DNS name of the computer. Win:3.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
domain name	*PlainGlobal*	<string>	Returns the fully qualified domain name of the machine. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
domainname	*PlainGlobal*	<string>	Same as domain name. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
drive	*PlainGlobal*	<filesystem>	Iterates through all valid drives on the system. Included for compatibility with Windows machines, this Inspector is the same as filesystem. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
drive <string>	*NamedGlobal*	<drive>	Returns a drive object for the name provided. • Note: For Unix, this Inspector returns a <filesystem> object as of version 6.0 of BES. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
drive <string>	*NamedGlobal*	<filesystem>	Returns the drive associated with the pathname specified by <string>. Included for compatibility with Windows machines, this Inspector is the same as filesystem <string>. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
environment	*PlainGlobal*	<environment>	Returns an object corresponding to the currently defined set of environment variables. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
error <string>	*NamedGlobal*	<undefined>	Always fails; if an error message is generated, it is based on the given string. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
false	*PlainGlobal*	<boolean>	Returns the boolean FALSE. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
february	*PlainGlobal*	<month>	Returns February as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Return Type	Description
february <integer>	*NumberedGlobal*	<day of year>	Returns the nth day of February as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
file <string>	*NamedGlobal*	<file>	Returns a filesystem object corresponding to the full pathname provided in <string>. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
file <symlink>	*Index<symlink>Global*	<file>	Returns the file pointed to by the specified symlink. If the file doesn't exist, this Inspector will throw a 'non-existent object' error. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
filesystem	*PlainGlobal*	<filesystem>	Returns <filesystem> objects for all currently mounted file systems. Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1
filesystem <string>	*NamedGlobal*	<filesystem>	Returns the filesystem corresponding to the specified name. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
folder <string>	*NamedGlobal*	<folder>	Returns a folder object for the name provided. See drive. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
folder <symlink>	*Index<symlink>Global*	<folder>	Returns the folder pointed to by the specified symlink. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
friday	*PlainGlobal*	<day of week>	Returns Friday as a day of week object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
ghz	*PlainGlobal*	<hertz>	Returns a Hertz object corresponding to 1 giga-hertz. See hertz. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
greatest hz	*PlainGlobal*	<hertz>	Returns the largest hertz object that can be represented on this machine. See hertz. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
greatest integer	*PlainGlobal*	<integer>	Returns the largest integer that can be represented on this machine. See integer. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Return Type	Description
greatest time interval	*PlainGlobal*	<time interval>	Returns the greatest time interval representable. The value corresponds to 106751991 days, 04:00:54.775807. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
hexadecimal integer <string>	*NamedGlobal*	<integer>	Creates an integer from the provided hexadecimal value. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
hexadecimal string <string>	*NamedGlobal*	<string>	Creates a string from the given hexadecimal value. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
host name	*PlainGlobal*	<string>	Returns the machine name (the same as the computer name or hostname on Unix machines). Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
hostname	*PlainGlobal*	<string>	Returns the standard host name, usually for the computer’s network. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
hour	*PlainGlobal*	<time interval>	Returns a time interval corresponding to 1 hour. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
hz	*PlainGlobal*	<hertz>	Returns a hertz object corresponding to 1 hertz. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
integer <integer>	*NumberedGlobal*	<integer>	Returns an integer. The keyword is optional. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
integer <string>	*NamedGlobal*	<integer>	Returns integer for name provided. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
ipv4 address <string>	*NamedGlobal*	<ipv4 address>	Returns an ip address for the string provided. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
january	*PlainGlobal*	<month>	Returns January as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Return Type	Description
january <integer>	*NumberedGlobal*	<day of year>	Returns the nth day of January as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
july	*PlainGlobal*	<month>	Returns July as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
july <integer>	*NumberedGlobal*	<day of year>	Returns the nth day of July as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
june	*PlainGlobal*	<month>	Returns June as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
june <integer>	*NumberedGlobal*	<day of year>	Returns the nth day of June as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
khz	*PlainGlobal*	<hertz>	Returns a hertz object corresponding to 1 kilohertz. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
least hz	*PlainGlobal*	<hertz>	Returns the least hertz value that can be represented on this machine. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
least integer	*PlainGlobal*	<integer>	Returns the least integer value that can be represented on this machine. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
least time interval	*PlainGlobal*	<time interval>	Returns the least time interval that can be represented on this machine. The value corresponds to -106751991 days, 04:00:54.775808. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
local time <string>	*NamedGlobal*	<time>	Returns a time object for the name provided. See time. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
local time zone	*PlainGlobal*	<time zone>	Returns a time zone object corresponding to the local time zone. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Return Type	Description
main gather service	*PlainGlobal*	<service>	Returns a service object for the main gathering service, typically located on the main server. Win:3.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
main processor	*PlainGlobal*	<processor>	Returns the processor object corresponding to the main processor. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
march	*PlainGlobal*	<month>	Returns March as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
march <integer>	*NumberedGlobal*	<day of year>	Returns the nth day of March as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
may	*PlainGlobal*	<month>	Returns May as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
may <integer>	*NumberedGlobal*	<day of year>	Returns the nth day of May as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
mhz	*PlainGlobal*	<hertz>	Returns a hertz object corresponding to 1 megahertz. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
microsecond	*PlainGlobal*	<time interval>	Returns a time interval corresponding to .000001 seconds. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
midnight	*PlainGlobal*	<time of day>	Returns 00:00:00 as a time of day object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
millisecond	*PlainGlobal*	<time interval>	Returns a time interval corresponding to .001 seconds. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
minute	*PlainGlobal*	<time interval>	Returns a time interval corresponding to 1 minute. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
module <string>	*NamedGlobal*	<module>	For BigFix internal use only. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1

Key Phrase	*Form*	Return Type	Description
monday	*PlainGlobal*	<day of week>	Returns the day of week object for Monday. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
month	*PlainGlobal*	<number of months>	Returns the specified number of months. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
month <integer>	*NumberedGlobal*	<month>	Returns the month type corresponding to the given <integer>. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
month <string>	*NamedGlobal*	<month>	Returns a month type corresponding to the given <string>. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
network	*PlainGlobal*	<network>	Returns an object containing properties of the network. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
noon	*PlainGlobal*	<time of day>	Returns 12:00:00 as a time of day object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
november	*PlainGlobal*	<month>	Returns November as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
november <integer>	*NumberedGlobal*	<day of year>	Returns the nth day of November as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
now	*PlainGlobal*	<time>	Returns the current time as a time object. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
october	*PlainGlobal*	<month>	Returns October as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
october <integer>	*NumberedGlobal*	<day of year>	Returns the nth day of October as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
operating system	*PlainGlobal*	<operating system>	Returns the operating system object. See operating system. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Return Type	Description
parameter <string>	*NamedGlobal*	<string>	This Inspector is a synonym for the parameter <string> of <action>. It looks up the value of the action parameter specified by <string>. This is used in conjunction with the parameter set command. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
pending login	*PlainGlobal*	<boolean>	Installers may leave values in the registry that the operating system will execute when the next user logs in. Pending login can detect these registry entries. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:6.0
pending restart	*PlainGlobal*	<boolean>	Returns TRUE if the operating system indicates that a restart needs to occur. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
pending restart <string>	*NamedGlobal*	<boolean>	Immediately after issuing a command like 'Action requires restart "PatchGroupX"', the expression 'Pending restart "PatchGroupX"' will be true until the next restart. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
pkgdb	*PlainGlobal*	<pkgdb>	A top-level object containing a list of all the installed packages, in the form of pkginfos. Sol:3.1
process	*PlainGlobal*	<process>	Returns all process objects currently running. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
process <integer>	*NumberedGlobal*	<process>	Returns the process object corresponding to the given integer pid. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
process <string>	*NamedGlobal*	<process>	Returns the process object corresponding to the name specified by <string>. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1

Key Phrase	*Form*	Return Type	Description
processor	*PlainGlobal*	<processor>	Returns all the processor objects defined on the machine. See processor. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
processor <integer>	*NumberedGlobal*	<processor>	Returns a processor object for the numbered processor. Processors are numbered from 1. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
property <string>	*NamedGlobal*	<property>	Typically used in the plural, returns the "line" Inspector properties. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
property returning <type>	*Index<type>Global*	<property>	Produces a list of the Inspector properties that return the specified <type>. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
ram	*PlainGlobal*	<ram>	Returns a ram object for inspecting the properties of Random Access Memory installed on the machine. See ram. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
random access memory	*PlainGlobal*	<ram>	Same as above. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1
regex <string>	*NamedGlobal*	<regular expression>	Creates a regex object from the given string. Win:6.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
regular expression <string>	*NamedGlobal*	<regular expression>	Same as regex <string>. Win:6.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
relay service	*PlainGlobal*	<service>	Returns a service object for the relay component of BES. Win:3.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
root folder	*PlainGlobal*	<folder>	Returns the folder corresponding to ‘/’. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
root server	*PlainGlobal*	<root server>	Returns an object representing the root BES Server to which the client last registered. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Key Phrase	*Form*	Return Type	Description
rope <string>	*NamedGlobal*	<rope>	Creates a rope object from the given string. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
runlevel	*PlainGlobal*	<runlevel>	Returns the current runlevel of the local machine. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
saturday	*PlainGlobal*	<day of week>	Returns Saturday as a day of week object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
second	*PlainGlobal*	<time interval>	Returns a time interval corresponding to 1 second. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
selected server	*PlainGlobal*	<selected server>	The BES Server or BES Relay to which the agent reports. Returned as the "selected server" type. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
september	*PlainGlobal*	<month>	Returns September as an object of type month. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
september <integer>	*NumberedGlobal*	<day of year>	Returns the nth day of September as a 'day of year' type (month day). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
service <string>	*NamedGlobal*	<service>	Returns the service object matching the name provided regardless of its running state. Win:1.2, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
site	*PlainGlobal*	<site>	Returns all the site objects that are currently loaded into memory. See site. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
site <string>	*NamedGlobal*	<site>	Returns a site object for the name provided. The name is the URL of the site location. See site. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Return Type	Description
string <string>	*NamedGlobal*	<string>	Returns a string for the name provided. The keyword string is optional. For example, string "hi" = "hi". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
sunday	*PlainGlobal*	<day of week>	Returns Sunday as a day of week object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
symlink <filesystem object>	*Index<filesystem object>Global*	<symlink>	Returns a symlink from the specified filesystem object. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
symlink <string>	*NamedGlobal*	<symlink>	Returns a symlink from the specified string. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
symlink <symlink>	*Index<symlink>Global*	<symlink>	Returns a symlink from the specified symlink path, even if the symlink is broken. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
system language	*PlainGlobal*	<string>	Returns the language of the system as a string. It is identified using the GetSystemDefaultLangID() system call. See the language keyword of the application object for a list of possible language value. Win:1.2, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1
system locale	*PlainGlobal*	<language>	Determines which bitmap fonts, and OEM, ANSI, and MAC code pages are defaults for the system. This only affects applications that are not fully Unicode. Win:4.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1
system ui language	*PlainGlobal*	<language>	Determines the default language of menus and dialogs, messages and help files. Win:4.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1
thursday	*PlainGlobal*	<day of week>	Returns Thursday as a day of week object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Return Type	Description
time <string>	*NamedGlobal*	<time>	Returns a time object for the name provided. See time. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
time interval <string>	*NamedGlobal*	<time interval>	Creates a time interval from the string. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
time zone <string>	*NamedGlobal*	<time zone>	Returns a time zone object for the name provided. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
time_of_day <string>	*NamedGlobal*	<time of day>	Creates a time of day object out of the given string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
true	*PlainGlobal*	<boolean>	Returns the boolean TRUE. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
tuesday	*PlainGlobal*	<day of week>	Returns Tuesday as a day of week object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
unary operator <string>	*NamedGlobal*	<unary operator>	Typically used in the plural, this inspector returns a list of objects that use the specified operator. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
unary operator returning <type>	*Index<type>Global*	<unary operator>	Returns a list of the unary operator inspectors (such as negative) that return the specified type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
universal time <string>	*NamedGlobal*	<time>	Returns a time object for the name provided. See time. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
universal time zone	*PlainGlobal*	<time zone>	Returns a time zone object corresponding to the universal time zone. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
user	*PlainGlobal*	<user>	Returns objects for all logged-on users of the computer. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1

Key Phrase	*Form*	Return Type	Description
user <string>	*NamedGlobal*	<user>	Returns the user specified by <string>. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
version <string>	*NamedGlobal*	<version>	Short hand for 'file version'. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
wednesday	*PlainGlobal*	<day of week>	Returns Wednesday as a day of week object. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
week	*PlainGlobal*	<time interval>	Returns a time interval corresponding to 1 week. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
year	*PlainGlobal*	<number of months>	Returns the specified number of years as a <number of months> type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
year <integer>	*NumberedGlobal*	<year>	Creates a year object from the specified integer. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
year <string>	*NamedGlobal*	<year>	Creates a year object from the specified string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
zoned time_of_day <string>	*NamedGlobal*	<time of day with time zone>	Returns a 'time of day with time zone' object from the specified string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Examples

- `bit 0 of 5`
  - Returns TRUE.
- `bit set "101" as integer`
  - Returns 5.
- `if FALSE then 1 else error "my error message"`
  - Returns the string: User-defined error: my error message.
- `names of filesystems`
  - Returns the names of the mounted file systems.
- `hexadecimal integer "A0"`
  - Returns 160.
- `host name`
  - Returns a string like "localhost.localdomain" or "user.bigcorp".

Filesystem Objects

This chapter covers the keywords for extracting information from the file system, like files, drives, pathnames, folders, etc. It also includes the keywords needed to identify and compare version information of files and patches.

Filesystem

The filesystem object can be used to inspect various aspects of the directory structure and file types. Here are some of the possible types:

- affs
- ext, ext2, ext2_old
- hpfs
- iso
- minix, minix_30, minix2, minix2_30
- msdos
- ncp
- nfs
- proc
- smb
- xenix
- sysv4, sysv2
- coh
- ufs
- xia

Creation Methods

Key Phrase	*Form*	Description
drive	*PlainGlobal*	For relevance language compatibility with Windows, you can iterate the mounted file systems using this inspector. Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1
drive <string>	*NamedGlobal*	Returns the drive associated with the pathname specified by <string>. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
drive of <symlink>	*Plain*	Returns the drive associated with the specified symlink as a <filesystem> object. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Key Phrase	*Form*	Description
filesystem	*PlainGlobal*	Returns <filesystem> objects for all currently mounted file systems. Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1
filesystem <string>	*NamedGlobal*	Returns the filesystem object for the name provided. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
filesystem of <file>	*Plain*	Returns the filesystem on which the file resides. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
filesystem of <folder>	*Plain*	Returns the filesystem on which the folder resides. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
filesystem of <symlink>	*Plain*	Returns the filesystem on which the symlink resides. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Properties

Key Phrase	*Form*	Return Type	Description
file count of <filesystem>	*Plain*	<integer>	Returns the total number of files that may be saved on this filesystem. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
free file count of <filesystem>	*Plain*	<integer>	Returns the number of files available on this filesystem. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
free percent of <filesystem>	*Plain*	<integer>	Returns the percentage of the file system currently available. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
free space of <filesystem>	*Plain*	<integer>	Returns the number of bytes on this filesystem. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
name of <filesystem>	*Plain*	<string>	Returns the mount point of the filesystem object. Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1
size of <filesystem>	*Plain*	<integer>	Returns the total number of bytes on this file system (same as total space). Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
total space of <filesystem>	*Plain*	<integer>	Returns the total number of bytes on this file system (same as size). Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1

Key Phrase	*Form*	Return Type	Description
type of <filesystem>	*Plain*	<string>	Returns the filesystem type. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
used file count of <filesystem>	*Plain*	<integer>	The number of files in use on this filesystem. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
used percent of <filesystem>	*Plain*	<integer>	Returns the percentage of the file system currently in use. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
used space of <filesystem>	*Plain*	<integer>	Returns the number of bytes on this filesystem currently in use. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1

Examples

- `names of drives`
  - Returns the names of the mounted drives.

- `file count of drive "/etc/passwd"`
  - Returns the total number of files on the drive containing the specified file.

- `names of filesystems`
  - Returns the mount points of the file systems.

Filesystem Object

Properties

Key Phrase	*Form*	Return Type	Description
<filesystem object> as string	*Cast*	<string>	Casts a filesystem object as a string. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:5.1
<filesystem object> as symlink	*Cast*	<symlink>	Casts a link in the form of a file into a symlink. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
accessed time of <filesystem object>	*Plain*	<time>	When the filesystem object (file or folder) was last accessed. Some file systems maintain this property. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Return Type	Description
ancestor of <filesystem object>	*Plain*	<folder>	Returns all ancestor folders (recursive parent folders) of the given filesystem object (file or folder). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
gid of <filesystem object>	*Plain*	<integer>	Returns the group ID of the given filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
group execute of <filesystem object>	*Plain*	<boolean>	Returns TRUE if the group execute flag is set for the given filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
group mask of <filesystem object>	*Plain*	<integer>	Returns the group permission mask of the given filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
group name of <filesystem object>	*Plain*	<string>	Returns the group name of the given filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
group read of <filesystem object>	*Plain*	<boolean>	Returns TRUE if the group read flag is set for the given filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
group write of <filesystem object>	*Plain*	<boolean>	Returns TRUE if the group write flag is set for the given filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
link count of <filesystem object>	*Plain*	<integer>	Returns an integer corresponding to the number of hard links attached to the specified filesystem object. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
location of <filesystem object>	*Plain*	<string>	Returns the name of the directory in which the file or folder (filesystem object) is located. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
mode of <filesystem object>	*Plain*	<mode>	Returns the permissions mode for the given filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Return Type	Description
modification time of <filesystem object>	*Plain*	<time>	The date and time of latest modification of the file. This corresponds to what is shown in the "Get Info" box. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:4.1
name of <filesystem object>	*Plain*	<string>	This returns the name of the file or folder. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:4.1
other execute of <filesystem object>	*Plain*	<boolean>	Returns TRUE if others (not in the group) have execute permissions on the given filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
other mask of <filesystem object>	*Plain*	<integer>	Returns the other (not in the group) mask as a 3-bit integer corresponding to rwx permissions for the specified filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
other read of <filesystem object>	*Plain*	<boolean>	Returns TRUE if others (not in the group) have read permissions on the given filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
other write of <filesystem object>	*Plain*	<boolean>	Returns TRUE if others (not in the group) have write permissions on the given filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
parent folder of <filesystem object>	*Plain*	<folder>	The folder containing the specified file or folder. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:4.1
pathname of <filesystem object>	*Plain*	<string>	Returns the full pathname of the specified file or folder (filesystem object) as a string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:5.1
setgid of <filesystem object>	*Plain*	<boolean>	Returns TRUE if the setgid (group ID) bit is set for the specified filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
setuid of <filesystem object>	*Plain*	<boolean>	Returns TRUE if the setuid (user ID) bit is set for the specified filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
uid of <filesystem object>	*Plain*	<integer>	The user ID of the user who owns this filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Key Phrase	*Form*	Return Type	Description
user execute of <filesystem object>	*Plain*	<boolean>	Returns TRUE if the owner (user) has execute permissions on the given filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
user mask of <filesystem object>	*Plain*	<integer>	Integer representing user permissions (3 bit mask, RWX) on the specified filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
user name of <filesystem object>	*Plain*	<string>	Returns the owner (user) name of the specified filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
user read of <filesystem object>	*Plain*	<boolean>	Returns the owner (user) read permissions for the specified filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
user write of <filesystem object>	*Plain*	<boolean>	Returns the owner (user) write permissions for the specified filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Examples

- `file "/example/link" as symlink`
  - Returns the link (in file format) as a symlink object ("/example/link").

- `name of object "iChat.app" of applications folder`
  - Returns iChat.app.

File

For each file in the file system, you can create a corresponding file object and inspect its properties. Inspectors are also provided to look at version data of executable files. File objects are derived from filesystem objects.

Creation Methods

See application objects for additional creation methods

Key Phrase	*Form*	Description
<symlink> as file	*Cast*	Converts a symlink object into the file it points to. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
default web browser	*PlainGlobal*	Creates an object corresponding to the default web browser. This is a Windows and Macintosh inspector; it will fail gracefully on other operating systems instead of generating an error. • Note: This Inspector returns a <file> object on Unix, an <application> on Windows and a <filesystem object> on the Mac. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
descendant of <folder>	*Plain*	Returns a list of all the descendant files of the specified folder. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:6.0
file <string>	*NamedGlobal*	Returns a filesystem object corresponding to the full pathname provided in <string>. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
file <string> of <folder>	*Named*	Creates the file objects corresponding to the named file within the folder. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
file <symlink>	*Index<symlink>Global*	Returns the file pointed to by the specified symlink. If the file doesn't exist, this Inspector will throw a 'non-existent object' error. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
file of <folder>	*Plain*	Iterates through the files of a folder. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Description
find file <string> of <folder>	*Named*	Creates an object corresponding to the files of the folder that that match the wildcard <string> provided. Win:1.2, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
masthead of <site>	*Plain*	A copy of the masthead is maintained with the site data. This inspector returns a file object for the copy. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Note

File systems that do not maintain the creation or last accessed times will often return the last modification time when queried for the creation or last accessed times or files. Modification times are preserved when files are copied. Thus, it is not uncommon to see a file that appears to have been modified before it was created.

Properties

Key Phrase	*Form*	Return Type	Description
<file> as string	*Cast*	<string>	Creates a string containing the full pathname of the specified file. See <file>. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
byte <integer> of <file>	*Numbered*	<integer>	Returns the numeric value of the byte located at the offset specified by number within the file. Byte 0 of the file is the first byte. Win:1.2, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
content of <file>	*Plain*	<file content>	Returns an object that can be used to search for a string in the file. See content. Win:1.2, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
filesystem of <file>	*Plain*	<filesystem>	Returns the Unix filesystem flag for the given file. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
key <string> of <file>	*Named*	<string>	Returns a key and its value from the given structured text file. It iterates over lines that start with the key name (as specified by <string>) followed by an = or : character. When searching, white space is ignored. Win:4.1, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:6.0

Key Phrase	*Form*	Return Type	Description
line <integer> of <file>	*Numbered*	<file line>	Returns the nth line (specified by <integer>) from the given file. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:5.1
line containing <string> of <file>	*Named*	<file line>	Returns all lines from the given file that contain the specified string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:5.1
line of <file>	*Plain*	<file line>	Iterates over all the lines of the specified file. NOTE: lines are truncated to 1023 characters. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:5.1
line starting with <string> of <file>	*Named*	<file line>	Same as line <string> of <file>, returns the lines of the given file that start with the specified string. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:5.1
section <string> of <file>	*Named*	<file section>	Returns a named section of a file. Useful for locating sections of 'ini' files. Section names are delimited by square bracket characters '[section name]'. See examples below. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:6.0
sha1 of <file>	*Plain*	<string>	Returns the sha1 checksum of the file hex encoded as a 40 character long string. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
size of <file>	*Plain*	<integer>	Returns the size in bytes of a file. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
variable of <file>	*Plain*	<string>	Returns the names of variables contained in an INF style file, in the format [section].name=value. Win:4.1, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Note

Folder and file names may be case sensitive. Use “as uppercase” or “as lowercase” if you don’t know the actual case when making comparisons. Iterating through folders with many files can be time consuming.

Note

The format of the string returned when casting a file using 'as string' is:
"<fileName>" "<version>" "<fileDesc>" "<fileVersion>" "<companyName>"
Where:

<fileName>	The name of the file
<version>	The 'Product Version' of the file.
<fileDesc>	The value 'FileDescription' of version block 1 of the file.
<fileVersion>	The value 'FileVersion' of version block 1 of the file.
<companyName>	The value 'CompanyName' of version block 1 of the file.

Examples

- `Number of find files "siteico*.bmp" of client folder of current site = 3`
  - Returns TRUE if there are 3 files matching the wildcard pattern siteico*.bmp.

- `names of find files "*.exe" of windows folder`
  - Returns a list of the names of all the executable programs in the Windows folder.

- `modification time of masthead of current site < time "4 Aug 1997 01:00 pdt"`
  - TRUE if the masthead is older than the specified date.

Application

Application objects derive from file objects. Therefore, application objects inherit all of the properties of the file object. This means that you can inspect properties such as 'modification time' or 'Product Version' of an application just as you would a file. See the properties of a file object for a complete list of these.

• The real power and primary purpose of the application object is their creation. The creation methods are optimized in anticipation of the importance of this object.

Creation Methods

Key Phrase	*Form*	Description
application <string>	*NamedGlobal*	Creates an application object for the name provided. Win:1.2, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
application <string> of <folder>	*Named*	As with the file object, you can create an application object by naming it relative to its parent folder. Win:1.2, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Note

See ‘File’ for a list of the Application properties. Folder and file names may be case sensitive. Use “as uppercase” or “as lowercase” if you don’t know the actual case when making comparisons. For Windows versions of these Inspectors, a properly installed application will register itself with the operating system. It does this by creating a registry sub-key usually named after the executable. The regapp Inspector uses the default value of this sub-key as a string that is the full pathname of the executable.

Examples

- `exists application "notepad.exe" of the windows folder`

Using the application of folder creation method, this example locates the notepad application provided it exists in the Windows folder.

Folder

For every folder that exists in the file system, you can create a folder object. The properties of this object allow you to examine many aspects of the system. Folder objects are derived from filesystem objects. On the Macintosh, there are dozens of specialized folders; access to them depends on the domain. If the domain is not specified, it defaults to the system domain.

Creation Methods

Key Phrase	*Form*	Description
<symlink> as folder	*Cast*	Converts a symlink object into the folder it points to. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
ancestor of <filesystem object>	*Plain*	Returns all ancestor folders (recursive parent folders) of the given filesystem object (file or folder). Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
ancestor of <symlink>	*Plain*	Returns all ancestor folders (recursive parent folders) of the given symlink. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
client folder of <site>	*Plain*	Creates an object corresponding to the folder on the client where site data is gathered. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
descendant folder of <folder>	*Plain*	Returns the descendant folders, recursively, of the given folder. The folder equivalent of "descendants of <folder>". Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
folder <string>	*NamedGlobal*	Creates a folder object for the named folder. This is a global property. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
folder <string> of <folder>	*Named*	Creates a folder object for the named sub-folder. Trailing slashes should be omitted from the name. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
folder <symlink>	*Index<symlink>Global*	Returns the folder pointed to by the specified symlink. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Key Phrase	*Form*	Description
folder of <folder>	*Plain*	Iterates through the sub-folders of the folder object. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
parent folder of <filesystem object>	*Plain*	The folder containing the specified file or folder. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:4.1
parent folder of <symlink>	*Plain*	Creates a folder object corresponding to the parent folder of the given symlink. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
root folder	*PlainGlobal*	Returns the folder corresponding to '/'. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1

Properties

Key Phrase	*Form*	Return Type	Description
application <string> of <folder>	*Named*	<application>	Returns an application object for the named file located in the folder. See application. Win:1.2, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
descendant folder of <folder>	*Plain*	<folder>	Returns the descendant folders, recursively, of the given folder. The folder equivalent of "descendants of <folder>". Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
descendant of <folder>	*Plain*	<file>	Returns a list of all the descendant files of the specified folder. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:6.0
drive of <folder>	*Plain*	<filesystem>	Creates the drive object corresponding to the folder location. Included for compatibility with Windows machines, this Inspector is the same as filesystem of <folder>. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
file <string> of <folder>	*Named*	<file>	Returns a file object for the named file located in the folder. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
file of <folder>	*Plain*	<file>	Iterates through the files of a folder returning file objects. When combined with a whose clause you can select files with specific properties. See file. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Return Type	Description
filesystem of <folder>	*Plain*	<filesystem>	Returns the filesystem on which the folder resides. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
find file <string> of <folder>	*Named*	<file>	Iterates through the files of a folder returning file objects whose name matches the search string provided in the name parameter. See example below. Win:1.2, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
folder <string> of <folder>	*Named*	<folder>	Returns a folder object for the named sub-folder. Trailing slashes should be omitted from the name. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
folder of <folder>	*Plain*	<folder>	Iterates through the folders of a folder returning folder objects. When combined with a whose clause, you can select folders with specific properties. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
symlink <string> of <folder>	*Named*	<symlink>	Returns the named symlink from the specified folder. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
symlink of <folder>	*Plain*	<symlink>	Returns the symlink(s) in the specified folder, whether or not they are broken. • NOTE: This behavior differs from looking for files in a folder. Although that returns links along with the files, it doesn't return broken links. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Note

Folder and file names may be case sensitive. Use “as uppercase” or “as lowercase” if you don’t know the actual case when making comparisons. Be careful not to iterate through folders that contain a large number of files. Counting files in such a folder can be slow. Always try to use the most efficient techniques to minimize the client overhead. Consider using the “find file” Inspector which allows you to filter sets of files by using the wildcard.

Examples

- `exists folder "/usr/lib"`
- Checks for the existence of the usr/lib folder.

- `exists folder "tmp" of root folder`
- Returns TRUE if tmp is a subdirectory of the root directory.

File Section

Many programs and utilities store their settings in 'ini' files. This object is designed to access these settings. An 'ini' file is composed of zero or more named sections, each with zero or more keys. Each key is identified by name and has a string value.

Creation Methods

Key Phrase	*Form*	Description
section <string> of <file>	*Named*	Creates a file section for the name given. A case-insensitive search is performed to locate the named section in the file. Searching through files for configuration data can be a slow process. Particularly for large ini files. In this case you may want to find another method that requires less computation. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:6.0

Properties

Key Phrase	*Form*	Return Type	Description
key <string> of <file section>	*Named*	<string>	Returns a string containing the value for the name provided. A case-insensitive search is performed through the section of the file. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:6.0

Note

Files with an extension of .ini are common in Windows systems, but rare in unix systems. However they are a handy cross-platform way of maintaining a collection of named variables.

Examples

- `key "Manufacturer" of section "General" of file "/etc/oeminfo"`
- Returns the value of the "Manufacturer" key of the section named "General" of the specified file.

File Content

Content objects can be constructed from file objects to inspect their contents.

Creation Methods

Key Phrase	*Form*	Description
<file content> as lowercase	*Cast*	Returns the contents of the file as lower case characters. Win:1.2, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
<file content> as uppercase	*Cast*	Returns the contents of the file as upper case characters. Win:1.2, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
content of <file>	*Plain*	Creates a content object for a file. Win:1.2, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Properties

Key Phrase	*Form*	Return Type	Description
<file content> as lowercase	*Cast*	<file content>	Returns a lowercase version of the content provided. Win:1.2, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
<file content> as uppercase	*Cast*	<file content>	Returns an uppercase version of the content provided. Win:1.2, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Operators

Key phrase	Return Type	Description
<file content> contains <string>	*<boolean>*	Returns TRUE if the string is located in the content provided. Win:1.2, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Note

See "file section" for a more detailed inspection of .ini files.

Examples

- `content of file "oeminfo.ini" of system folder contains "Manufacturer=Dell"`
  - Returns TRUE if the exact character sequence "Dell" is located in the file.

- `content of file "oeminfo.ini" of system folder as lowercase contains "emachines"`
  - Returns TRUE if either of the strings "emachines" or "eMachines" is found in the file.

Version

This is the numeric method of indicating the file version, which is compact, convenient and fast. It makes use of a short string to define the version number.

Creation Methods

Key Phrase	*Form*	Description
<string> as version	*Cast*	Turns a string into a version object. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
version <string>	*NamedGlobal*	Creates a version object corresponding to the name provided. Syntax: version "1.2". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
version of <client>	*Plain*	The product version of the BES application (BESClient or QnA). Lin:4.1, Sol:4.1, HPUX:4.0, AIX:4.1
version of <current relay>	*Plain*	Returns a version object that is the version of the server or relay that the client last registered with. This may be a BES Relay or the BES root server. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Properties

Key Phrase	*Form*	Return Type	Description
<version> as string	*Cast*	<string>	Turns a version type into a string of the form "1.2.3.4". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Operators

Key phrase	Return Type	Description
<version> {cmp} <string>	<*boolean*>	Returns a boolean TRUE or FALSE, depending on the comparison operator, where: • {cmp} is one of: =, !=, <, <=, >, >= . Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
<version> {cmp} <version>	<*boolean*>	Returns a boolean TRUE or FALSE, depending on the comparison operator, where: • {cmp} is one of: =, !=, <, <=, >, >= . Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Note

Using the numeric version data is better than identifying an application based on version block strings. If you know the numeric version information and that the developer has identified each release of his application uniquely, then this is the way to proceed. It requires far less overhead than the other method. Furthermore, if you know that the numeric version data is monotonically increasing then you can compare their values using the special comparison operators.

Examples

- `"MyApp 1.2" as version = "MyApp 1.2" as version`

On unix, the version object contains a string. The comparison is expecting the string parts to match. If one of the operands comes from a property of some other object, this provides a way to compare a version against that property, provided they match in the rest of the string.

- `version of client as string`

Returns a string like "4.0.3.7".

- `version "09" < version "1"`

Returns TRUE. Uses the Unix version compare, strverscmp().

Mode

The mode inpector returns file type information and permissions. These are the possible values of mode:

- S_IFMT 170000 bitmask for the file type bitfields
- S_IFSOCK 140000 socket
- S_IFLNK 120000 symbolic link
- S_IFREG 100000 regular file
- S_IFBLK 060000 block device
- S_IFDIR 040000 directory
- S_IFCHR 020000 character device
- S_IFIFO 010000 fifo
- S_ISUID 004000 set UID bit
- S_ISGID 002000 set GID bit
- S_ISVTX 001000 sticky bit
- S_IRWXU 000700 mask for file owner permissions
- S_IRUSR 000400 owner has read permission
- S_IWUSR 000200 owner has write permission
- S_IXUSR 000100 owner has execute permission
- S_IRWXG 000070 mask for group permissions
- S_IRGRP 000040 group has read permission
- S_IWGRP 000020 group has write permission
- S_IXGRP 000010 group has execute permission
- S_IRWXO 000007 mask for permissions for others (not in group)
- S_IROTH 000004 others have read permission
- S_IWOTH 000002 others have write permission
- S_IXOTH 000001 others have execute permission

Creation Methods

Key Phrase	*Form*	Description
mode of <filesystem object>	*Plain*	Returns the permissions mode for the given filesystem object. Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Properties

Key Phrase	*Form*	Return Type	Description
<mode> as octal string	*Cast*	<string>	Converts the mode to a string of octal numbers. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
<mode> as string	*Cast*	<string>	Converts the mode to a string. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
group mask of <mode>	*Plain*	<mode_mask>	Returns the mask for group permissions for the given mode. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
other mask of <mode>	*Plain*	<mode_mask>	Returns the mask for permissions for others (those not in the group) for the given mode. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
setgid of <mode>	*Plain*	<boolean>	Returns TRUE if setgid (the group ID flag) is set. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
setuid of <mode>	*Plain*	<boolean>	Returns TRUE if setuid (the user ID flag) is set. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
sticky of <mode>	*Plain*	<boolean>	The `sticky' bit (S_ISVTX) on a directory means that a file in that directory can be renamed or deleted only by the owner of the file, by the owner of the directory, and by root. This inspector returns TRUE if the sticky bit of the given mode is on. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
user mask of <mode>	*Plain*	<mode_mask>	Returns the mask for the user (file owner) permissions for the given mode. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1

Mode_mask

The mode_mask Inspector is a differently formatted version of the mode, created by shifting the key information down to the low three bits.

Creation Methods

Key Phrase	*Form*	Description
group mask of <mode>	*Plain*	The rwx mask (shifted to the lower 3 bits) for group permissions for the given mode. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
other mask of <mode>	*Plain*	The rwx mask (shifted to the lower 3 bits) for permissions for others (those not in the group) for the given mode. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
user mask of <mode>	*Plain*	The rwx mask (shifted to the lower 3 bits) for permissions for the user (the file owner) for the given mode. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1

Properties

Key Phrase	*Form*	Return Type	Description
<mode_mask> as integer	*Cast*	<integer>	Converts the mode mask to an integer, 1-4. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
<mode_mask> as string	*Cast*	<string>	Converts the mode mask to a string, e.g. "rwx". Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
execute of <mode_mask>	*Plain*	<boolean>	Returns TRUE if the execute flag (x) of the rwx mode mask is on. (binary 001 = 1). Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
read of <mode_mask>	*Plain*	<boolean>	Returns TRUE if the read flag of the rwx mode mask is on. (binary 100 = 4). Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
write of <mode_mask>	*Plain*	<boolean>	Returns TRUE if the write flag (w) of the rwx mode mask is on. (binary 010 = 2). Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1

Pkgdb

This is a database object created by BigFix to keep track of all the Solaris packages (pkginfos) that have been installed on this computer. The pkginfo object, in turn, contains Solaris patches, each with an ID and a revision number. The database essentially contains a comprehensive list of patches, allowing you to directly access a patch without going through pkginfo. The syntax is "patch of pkgdb", which can greatly simplify your relevance expressions.

Creation Methods

Key Phrase	*Form*	Description
pkgdb	*PlainGlobal*	Creates a top-level object containing a list of all the installed packages, in the form of pkginfos. Sol:3.1

Properties

Key Phrase	*Form*	Return Type	Description
patch <string> of <pkgdb>	*Named*	<patch>	Returns the requested patch from the package database. Typically used for existence tests. Sol:3.1
patch id <string> of <pkgdb>	*Named*	<string>	Returns the specified patch id string from the package database. Typically used for existence tests. Sol:3.1
patch id of <pkgdb>	*Plain*	<string>	Returns a space-delimited list of patch ids from the specified package database. Sol:3.1
patch of <pkgdb>	*Plain*	<patch>	Returns all the patches listed in the given pkgdb (package database). Sol:3.1
pkginfo <string> of <pkgdb>	*Named*	<pkginfo>	Returns the specified pkginfo from the package database. Typically used for existence tests. Sol:3.1
pkginfo of <pkgdb>	*Plain*	<pkginfo>	Returns all the packages (as pkginfos) in the given package database. Sol:3.1

Examples

- `exists patch "123456" of pkgdb`
  - Returns TRUE if any revision of the patch with this ID exists in the package database.
- `exists patch "123456-12" of pkgdb`
  - Returns TRUE if this exact patch ID (including the revision) exists in the package database.
- `exists patch id "123456-01 345678-23" of pkgdb`
  - Returns TRUE if these patch ids exist in the package database.
- `exists patch id whose (it contains "123456-01") of pkgdb`
  - Returns TRUE if the specified patch id exists in the package database.
- `patch id of pkgdb contains "123456"`
  - Returns TRUE if the designated patch ID exists in the package database.
- `exists pkginfo "SUNWrdm" of pkgdb`
  - Returns TRUE if the specified pkginfo exists in the package database.

Pkginfo

Pkginfo is an object containing information about Solaris packages, which are used to deliver patches. Pkginfo contains information about the category, the intended architecture, vendors, patch IDs and revision numbers. Other package information can be inspected using the "param" property which lets you access other parameters by name.

Creation Methods

Key Phrase	*Form*	Description
pkginfo <string> of <pkgdb>	*Named*	Returns the specified pkginfo from the package database. Typically used for existence tests. Sol:3.1
pkginfo of <pkgdb>	*Plain*	Creates an object containing all the packages (as pkginfos) in the given package database. Sol:3.1

Properties

Key Phrase	*Form*	Return Type	Description
<pkginfo> as string	*Cast*	<string>	Converts a pkginfo object into a string. Sol:3.1
arch of <pkginfo>	*Plain*	<string>	Returns a string representing the architecture of the package specified by the pkginfo object. Sol:3.1
category of <pkginfo>	*Plain*	<string>	Returns a string representing the category of the package specified by pkginfo. Sol:3.1
name of <pkginfo>	*Plain*	<string>	Returns the name of the package from the specified pkginfo object. Sol:3.1
param <string> of <pkginfo>	*Named*	<string>	Returns the value of the named parameter (specified by <string>) from the pkginfo object. Sol:3.1
patch <string> of <pkginfo>	*Named*	<patch>	Returns a named patch from the specified pkginfo. Sol:3.1
patch id <string> of <pkginfo>	*Named*	<string>	Returns the specified patch id string from the specified pkginfo of the package database. Typically used for existence tests. Sol:3.1
patch id of <pkginfo>	*Plain*	<string>	Returns a space delimited list of patch ids, e.g.: "123456-01 123456-02 200213-23 501234-02". This is the same list available from the PATCHLIST parameter of the pkginfo. Sol:3.1
patch of <pkginfo>	*Plain*	<patch>	Returns the patch object referenced in the pkginfo object. Sol:3.1
pkginst of <pkginfo>	*Plain*	<string>	Returns a short string used as an abbreviation for the package name given by the pkginfo object. Typically contains a vendor symbol and an ID, like "CSCOh007". Sol:3.1

Key Phrase	*Form*	Return Type	Description
vendor of <pkginfo>	*Plain*	<string>	Returns a string containing the name of the package vendor. This is the same as the VENDOR parameter of the pkginfo. Sol:3.1
version of <pkginfo>	*Plain*	<string>	Returns a version number for the specified pkginfo object. Not to be confused with revision numbers of patches. This is the same as the VERSION parameter of the pkginfo. Sol:3.1

Examples

- `param "INSTDATE" of pkginfo "SPROhtws" of pkgdb`
  - Returns the install date as a string like "Jul 24 2003 09:45".

- `exists patch id "123456-01" of pkginfo "SUNWrdm" of pkgdb`
  - Returns TRUE if the designated patch ID exists in the specified pkginfo of the package database.

- `version of pkginfo "SPROhtws" of pkgdb`
  - Returns a version string like "8.0,REV=2003.04.02".

Patch

An object that exposes the properties of a Solaris patch, each of which has a base ID followed by a revision number, such as 123456-12. Here the base is 123456, which identifies the patch, and 12, which represents the revision of this particular patch.

Creation Methods

Key Phrase	*Form*	Description
patch <string> of <pkgdb>	*Named*	Returns the patch specified by <string> from the given Solaris package database. Sol:3.1
patch <string> of <pkginfo>	*Named*	Returns the requested patch object from the specified pkginfo. Typically used for existence tests. Sol:3.1
patch of <pkgdb>	*Plain*	Creates objects for all the patches in the specified package database. Sol:3.1

Key Phrase	*Form*	Description
patch of <pkginfo>	*Plain*	Returns the patch(es) named in the specified pkginfo object. Sol:3.1

Properties

Key Phrase	*Form*	Return Type	Description
base of <patch>	*Plain*	<string>	Returns the base number of the given patch, which is a unique identifier for the file, as a string. Sol:3.1
greatest revision of <patch>	*Plain*	<string>	Returns the highest revision of the patch. Sol:4.1
least revision of <patch>	*Plain*	<string>	Returns the lowest revision of the patch. Sol:4.1
revision <string> of <patch>	*Named*	<string>	Returns the specified revision string from the given patch. Typically used for existence tests. Sol:3.1
revision of <patch>	*Plain*	<string>	Returns an additional revision number which modifies the base of the patch. Sol:3.1

Examples

- `exists patch "123456" of pkginfo "SUNWrdm" of pkgdb`
  - Returns TRUE if this patch exists in the specified pkginfo of the package database.

- `greatest revision of patch "112233" of pkgdb = "12"`
  - Returns TRUE if the highest revision of the specified patch is equal to 12.

- `least revision of patch "112233" of pkgdb = "05"`
  - Returns TRUE if the lowest revision of the specified patch is equal to 05.

- `exists revision "02" of patch "123456" of pkginfo "SUNWrdm" of pkgdb`
  - Returns TRUE if the specified revision of the given patch in the given pkginfo exists in the package database.

- `exists patch whose (revision of it < "3") of pkgdb`
  - Returns TRUE if there is a patch in the package database with a revision number less than 3.

File Line

A file line is a string from a text file.

Creation Methods

Key Phrase	*Form*	Description
line <integer> of <file>	*Numbered*	Returns the nth line in a file. A file line is just a string, except that you can use the additional properties "next line" and "previous line". Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
line containing <string> of <file>	*Named*	Returns the line with the specified search string in the given file. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
line of <file>	*Plain*	Returns the lines of a specified file. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
line starting with <string> of <file>	*Named*	Returns a line from the given file beginning with the specified phrase. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1

Key Phrase	*Form*	Description
next line of <file line>	*Plain*	Returns the line after the specified line in a file (provided that it is not the last line). This Inspector can be chained indefinitely, eg., next line of next line of …. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
previous line of <file line>	*Plain*	Returns the line before the nth line in a file, provided n>1. You may repeat this command up to three times. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1

Properties

Key Phrase	*Form*	Return Type	Description
line number of <file line>	*Plain*	<integer>	Returns the line number of a given line. Can be used to locate specific lines in a file. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
next line of <file line>	*Plain*	<file line>	Returns the line after the specified line in a file (provided that it is not the last line). This Inspector can be chained indefinitely, eg., next line of next line of …. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1
previous line of <file line>	*Plain*	<file line>	Returns the line before the nth line in a file, provided n>1. You may repeat this command up to three times. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:5.1

Application Usage Summary

To enable these Inspectors, you first need to create the client setting _BESClient_UsageManager_EnableAppUsageSummary and initialize it to 1. You can also limit the summary to a subset of applications by creating _BESClient_UsageManager_EnableAppUsageSummaryApps and initializing it to a list of apps to include (or exclude). The value of this setting should look like +:app1:app2:app3: to add apps to the scope, and -:app1:app2: to exclude apps. The case is ignored. For instance, to enable summaries on the Word application, use the value +:winword.exe:.

Creation Methods

Key Phrase	*Form*	Description
application usage summary	*PlainGlobal*	Returns an application usage summary containing information including the start time, duration and other statistics on client applications. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
application usage summary <string>	*NamedGlobal*	Returns the usage summary for the application specified in <string>. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Properties

Key Phrase	*Form*	Return Type	Description
first start time of <application usage summary>	*Plain*	<time>	Returns the start time of the specified application since the computer was configured to track it, regardless of reboots. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
last start time of <application usage summary>	*Plain*	<time>	Returns the last time this specified application was started. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
last time seen of <application usage summary>	*Plain*	<time>	Returns the last time this specified application was seen running. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
name of <application usage summary>	*Plain*	<string>	Returns the names of the applications that are currently enabled for usage summaries. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
running of <application usage summary>	*Plain*	<boolean>	Returns TRUE if the specified application is currently running. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Key Phrase	*Form*	Return Type	Description
total duration of <application usage summary>	*Plain*	<time interval>	Returns the total elapsed time that the specified application has been running. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
total run count of <application usage summary>	*Plain*	<integer>	Returns the number of times that the specified application has been run since the client was configured to track it. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Symlink

Inspectors can analyze the file objects (files and folders) that symlinks point to by using the standard file Inspectors. The symlink Inspectors, on the other hand, allow you to analyze the properties of a symlink itself, not just the underlying file. There are many properties that symlinks have in common with files, such as name, pathname, parent folder, etc. In addition, a symlink has a value corresponding to the file object it points to. You can also determine if the file is available or not.

Creation Methods

Key Phrase	*Form*	Description
<filesystem object> as symlink	*Cast*	Casts a link in the form of a file into a symlink. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
<symlink> as symlink	*Cast*	Casts a symlink, provided for completeness. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
symlink <filesystem object>	*Index<filesystem object>Global*	Creates a symlink from the specified filesystem object. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
symlink <string>	*NamedGlobal*	Creates a symlink from the specified string. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
symlink <string> of <folder>	*Named*	Returns the named symlink from the specified folder. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
symlink <symlink>	*Index<symlink>Global*	Creates a symlink from the specified symlink path, even if the symlink is broken. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Key Phrase	*Form*	Description
symlink of <folder>	*Plain*	Returns the symlink(s) in the specified folder, whether or not they are broken. • NOTE: This behavior differs from looking for files in a folder. Although that returns links along with the files, it doesn't return broken links. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Properties

Key Phrase	*Form*	Return Type	Description
<symlink> as file	*Cast*	<file>	Converts a symlink object into the file it points to. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
<symlink> as folder	*Cast*	<folder>	Converts a symlink object into the folder it points to. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
<symlink> as string	*Cast*	<string>	Casts a symlink object as a string. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
<symlink> as symlink	*Cast*	<symlink>	Casts a symlink, provided for completeness. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
accessed time of <symlink>	*Plain*	<time>	Returns the last accessed time of the specified symlink. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
ancestor of <symlink>	*Plain*	<folder>	Returns all ancestor folders (recursive parent folders) of the given symlink. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
change time of <symlink>	*Plain*	<time>	Returns the last time the specified symlink was ‘changed’ by either writing it or setting its inode information. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
drive of <symlink>	*Plain*	<filesystem>	Returns the drive associated with the specified symlink as a <filesystem> object. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
filesystem of <symlink>	*Plain*	<filesystem>	Returns the filesystem on which the symlink resides. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Key Phrase	*Form*	Return Type	Description
gid of <symlink>	*Plain*	<integer>	Returns the group ID of the given symlink. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
group name of <symlink>	*Plain*	<string>	Returns the group name of the specified symlink. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
link count of <symlink>	*Plain*	<integer>	Returns an integer corresponding to the number of hard links attached to the specified symlink. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
location of <symlink>	*Plain*	<string>	Returns a string corresponding to the directory in which the symlink is located. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
modification time of <symlink>	*Plain*	<time>	Returns the time corresponding to the modification time of the specified symlink, not the file it points to. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
name of <symlink>	*Plain*	<string>	Returns a string that is the full path name of the specified symlink. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
parent folder of <symlink>	*Plain*	<folder>	Creates a folder object corresponding to the parent folder of the given symlink. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
pathname of <symlink>	*Plain*	<string>	Returns a string that contains the full path name of the specified symlink. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
uid of <symlink>	*Plain*	<integer>	The user ID of the user who owns this symlink. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
user name of <symlink>	*Plain*	<string>	Returns the symlink owner's (user's) name. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
value accessible of <symlink>	*Plain*	<boolean>	Returns TRUE if the pathname pointed to by the specified symlink is available. Returns FALSE if the file object is missing or unavailable with the current permissions. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
value of <symlink>	*Plain*	<string>	Returns the pathname that the symlink points to. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Examples

- `file "/example/link" as symlink`

  Returns the link (in file format) as a symlink object ("/example/link").

- `symlinks of folder "/example"`

  Returns a list of the symlink in the specified folder, whether or not they are broken. Note that this behavior is different from looking for files in a folder -- that will return links as well as files, but won't return broken links.

- `modification time of symlink "/example/link"`

  Returns the time corresponding to the modification time of the given symlink, not the file it points to.

- `modification time of file "/example/link"`

  Returns the time corresponding to the file object that the symlink is pointing to.

System Objects

These are the keywords available for querying various aspects of the system, including the name and version of the operating sytem. This chapter also covers the keywords used to describe the vendors and types of the various processors that coexist in a typical computer system. Some of these Inspectors are system-specific, but are included to provide cross-platform compatiblity.

Bios

On Windows computers, this object returns strings that identify the version of the BIOS. On other computers, all bios expressions will fail gracefully, rather than generating an error.

Creation Methods

Key Phrase	*Form*	Description
bios	*PlainGlobal*	Returns the date of the bios if it exists, or <unknown> if it does not exist. This is a Windows-only command. On a non-Windows system, bios returns False. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1

Properties

Key Phrase	*Form*	Return Type	Description
<bios> as string	*Cast*	<string>	This Windows-only Inspector returns a string that is the concatenation of the BIOS name and date. On a non-Windows operating system, it returns FALSE. Win:3.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
date of <bios>	*Plain*	<string>	This Windows-only Inspector returns the date string stored in the bios. This string is formatted as MM/DD/YY. On a non-Windows operating system, it returns FALSE. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
version of <bios>	*Plain*	<string>	This Windows-only Inspector returns the first string of the multi-string version stored in the bios. This string may not exist. The format depends upon your BIOS manufacturer. On a non-Windows operating system, it returns FALSE. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1

Operating System

The operating system object provides access to several important properties of the system.

Creation Methods

Key Phrase	*Form*	Description
operating system	*PlainGlobal*	Creates the global operating system object. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Properties

Key Phrase	*Form*	Return Type	Description
<operating system> as string	*Cast*	<string>	Returns a string containing the name of the operating system concatenated with the release. Win:3.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
architecture of <operating system>	*Plain*	<string>	Returns the architecture of the operating system. This is the value of the 'machine' element of the utsname structure obtained by calling uname. Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:6.0
boot time of <operating system>	*Plain*	<time>	Returns the time of the last restart. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
build of <operating system>	*Plain*	<string>	Returns a string corresponding to the build number of the OS. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:6.0
name of <operating system>	*Plain*	<string>	Returns the name of the operating system as a string. Names might include Win98, WinNT, etc. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
release of <operating system>	*Plain*	<string>	Information about the release of the operating system, typically formatted as <Major version>.<Minor version>. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
uptime of <operating system>	*Plain*	<time interval>	Returns a time interval that represents the elapsed time since the operating system was last booted. • Note: Depending on the Laptop, this interval may not include time spent in hibernation. Win:5.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1, Mac:6.0

Examples

- `now - boot time of operating system > week`
- Returns TRUE if the computer hasn't been rebooted for over a week.

- `name of operating system contains "Sun"`
- Returns TRUE on a typical Solaris system.

Processor

The processor object is used to identify the number and properties of processors in the system. You can identify the manufacturer of the CPU as well as the speed and other features. Many operating systems provide for multiple processors. You can inspect any one of them by their ordinal number.

Creation Methods

Key Phrase	*Form*	Description
main processor	*PlainGlobal*	Creates the object associated with the 'Primary' processor. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
processor	*PlainGlobal*	Iterates through the processors in the system. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
processor <integer>	*NumberedGlobal*	Creates the processor object for the number specified. The first processor is processor number 1. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1

Properties

Key Phrase	*Form*	Return Type	Description
family name of <processor>	*Plain*	<string>	Returns the family name of the CPU, dependent on the type of client computer, for instance Pentium, Sparc, PowerPC G4, etc. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
family of <processor>	*Plain*	<string>	A string representing the family of the CPU. Sol:6.0, AIX:6.0
fputype of <processor>	*Plain*	<string>	Returns pi_fputypes, a string containing the comma-separated types of floating-point units (FPUs) attached to the processor. This string will be empty if no FPU is attached. Sol:3.1

Key Phrase	*Form*	Return Type	Description
id of <processor>	*Plain*	<integer>	Returns an integer corresponding to the ID of the specified processor. Sol:6.0, AIX:6.0
index of <processor>	*Plain*	<integer>	Returns the ordinal number of the processor on a multi processor machine. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
model of <processor>	*Plain*	<string>	Returns the model number of the CPU as a string. Sol:6.0, HPUX:6.0, AIX:6.0
speed of <processor>	*Plain*	<hertz>	Returns the speed of the processor in Hertz. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
state of <processor>	*Plain*	<string>	Returns the current processor state, which can be "online", "offline", "poweroff", or "unknown". Sol:3.1, HPUX:4.0
type of <processor>	*Plain*	<integer>	Numeric type of the CPU. Values include: • 0 - standard • 1 - overdrive • 2 - dual CPU capable • 3 - reserved • Note: this Inspector returns a <string> type as of BES version 6.0 on Unix machines and version 5.1 on the Macintosh. Win:1.2, Sol:3.1, AIX:4.1, Mac:4.1
type of <processor>	*Plain*	<string>	Numeric type of the CPU, represented as a string. Values include: • 0 - standard • 1 - overdrive • 2 - dual CPU capable • 3 - reserved. Sol:6.0, AIX:6.0, Mac:5.1

Examples

- `number of processors > 1`
  Returns TRUE if the computer is a multi-processor system.

- `family name of main processor = "sparcv9"`
  Returns TRUE if the computer has the specified main processor family name.

- `speed of main processor < 2000 * MHz`
  - Returns TRUE is the cpu is slower than 2Ghz.

Ram

The ram object is used to inspect properties of the computer's random access memory.

Creation Methods

Key Phrase	*Form*	Description
ram	*PlainGlobal*	Creates the object that can be accessed to inspect the amount of ram on the machine. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
random access memory	*PlainGlobal*	Same as 'ram'. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1

Properties

Key Phrase	*Form*	Return Type	Description
free amount of <ram>	*Plain*	<integer>	Returns the amount of system RAM currently unused, in bytes. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
size of <ram>	*Plain*	<integer>	Returns the number of bytes of random access memory on the current machine. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
total amount of <ram>	*Plain*	<integer>	Same as size of <ram>. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
used amount of <ram>	*Plain*	<integer>	Returns the amount of system RAM currently used, in bytes. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1

Examples

- `size of ram / (1024 * 1024)`
  - Returns the size of RAM in megabytes.

License

These Inspectors are available only through the BigFix Enterprise System. They inspect the properties of the client's BigFix license.

Creation Methods

Key Phrase	*Form*	Description
bes license	*PlainGlobal*	Synonym for 'client license'. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
client license	*PlainGlobal*	Creates the global object containing client licensing information. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Properties

Key Phrase	*Form*	Return Type	Description
common name of <license>	*Plain*	<string>	Returns the name of the person (such as John Smith) who requested the action site license. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1
email address of <license>	*Plain*	<string>	Returns the email address of the person (such as John_Smith@bigcorp.com) who requested the action site license. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1
evaluation of <license>	*Plain*	<boolean>	Returns TRUE if client is running an evaluation license. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
expiration date of <license>	*Plain*	<time>	Returns date when license will expire. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
expiration state of <license>	*Plain*	<string>	Returns a string, one of "Unrestricted", "Grace" or "Restricted". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
gather url of <license>	*Plain*	<string>	Returns the gather URL for the deployment's main Action site as specified in the deployment masthead. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Key Phrase	*Form*	Return Type	Description
maximum seat count of <license>	*Plain*	<integer>	Returns maximum seat count allowed by the license. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
organization of <license>	*Plain*	<string>	Returns the organization of the person (such as Bigcorp, Inc.) who requested the action site license. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1
registrar number of <license>	*Plain*	<integer>	A unique number assigned to the issuer of the Action Site certificate. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1
seat count state of <license>	*Plain*	<string>	Returns one of "Unrestricted", "Grace" or "Restricted". Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
seat of <license>	*Plain*	<integer>	The license number assigned to the client. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
site number of <license>	*Plain*	<integer>	A unique number assigned to the Action Site certificate. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1
start date of <license>	*Plain*	<time>	The starting date specified for the BigFix license. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1

Service

Provides access to all services configured on Windows NT, 2K and XP systems. On a non-Windows system, expressions using these objects will fail gracefully instead of generating an error.

Creation Methods

Key Phrase	*Form*	Description
main gather service	*PlainGlobal*	Returns FALSE. Included for compatibility with Windows Inspectors. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
relay service	*PlainGlobal*	Returns a service object for the relay component of BES. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
service <string>	*NamedGlobal*	Creates the service object matching the specified name, regardless of its running state. Win:1.2, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Properties

Key Phrase	*Form*	Return Type	Description
state of <service>	*Plain*	<string>	Returns one of Continuing, Pausing, Paused, Running, Starting, Stopping, Stopped, Unknown. Win:1.2, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0

Process

When Unix launches a process, it assigns it a unique number called the process ID, or pid. The process object lets you inspect the properties of any running process, including its name and pid.

Creation Methods

Key Phrase	*Form*	Description
process	*PlainGlobal*	Returns all process objects currently running. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
process <integer>	*NumberedGlobal*	Returns the process object corresponding to the given integer pid. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
process <string>	*NamedGlobal*	Returns the process object corresponding to the name specified by <string>. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1

Properties

Key Phrase	*Form*	Return Type	Description
id of <process>	*Plain*	<integer>	Returns the integer ID of the specified process. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
name of <process>	*Plain*	<string>	Returns the name (as a string) of the specified process. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
pid of <process>	*Plain*	<integer>	Returns the integer process ID for the specified process. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
process id of <process>	*Plain*	<integer>	Returns the integer process ID for the specified process. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1

Examples

- `names of processes whose (pid of it < 20)`
  Returns a list of all process with an ID less than 20.

Language

A language is composed of a primary language (for example, Swiss) and a sub-language (for example, Swiss German).

Creation Methods

Key Phrase	*Form*	Description
system locale	*PlainGlobal*	Determines which bitmap fonts, and OEM, ANSI, and MAC code pages are defaults for the system. This only affects applications that are not fully Unicode. Win:4.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1
system ui language	*PlainGlobal*	Determines the default language of menus and dialogs, messages, INF files, and help files. Win:4.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1

Properties

Key Phrase	*Form*	Return Type	Description
<language> as string	*Cast*	<string>	Returns the language of the system locale. Win:4.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1
platform id of <language>	*Plain*	<string>	Returns the string resulting from a call to setlocale(LC_TYPE, ""). This call examines the system environment and returns a string representing the language and character set for any text-related system function. The string is of the form "en_US.UTF-8". Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1
primary language of <language>	*Plain*	<primary language>	Extracts the primary language identifier from a language. Win:4.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1

Examples

- `system locale as string`
  - Returns English (United States) if the system locale is US English.

Primary Language

A primary language identifier indicates the written/spoken language that is used by the system. However, to identify the language that is used in a country or region you must combine the primary language with a sub-language identifier to form language identifiers.

Creation Methods

Key Phrase	*Form*	Description
primary language of <language>	*Plain*	Extracts the primary language identifier from a language. Win:4.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1

Properties

Key Phrase	*Form*	Return Type	Description
<primary language> as string	*Cast*	<string>	Returns the primary language. Win:4.1, Lin:5.1, Sol:5.1, HPUX:5.1, AIX:5.1

Examples

- `primary language of system locale`
- Returns "English" for an English-language system.

Runlevel

The runlevel Inspectors refer to a mode of operation in various Unix systems. Typically, when a computer enters runlevel zero, it halts and when it enters runlevel six, it reboots. The intermediate runlevels differ widely among operating systems.

Creation Methods

Key Phrase	*Form*	Description
runlevel	*PlainGlobal*	Returns the current runlevel of the local machine. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Properties

Key Phrase	*Form*	Return Type	Description
<runlevel> as string	*Cast*	<string>	Casts a runlevel object as a string. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
effective time of <runlevel>	*Plain*	<time>	Returns the time at which the runlevel was set to its current value. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
value of <runlevel>	*Plain*	<string>	Returns the current runlevel. It evaluates to a platform-dependent string indicating the current runlevel. For instance, on Linux the value '3' indicates runlevel 3 and 'S' indicates single user mode. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Site Objects

These keywords query the properties of Fixlet sites to which the client is subscribed.

Site

A Site object is provided to access properties of Fixlet sites.

Creation Methods

Key Phrase	*Form*	Description
current site	*PlainGlobal*	Creates the site object corresponding to the site that provided the current Fixlet. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
site	*PlainGlobal*	Iterates through all the sites. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
site <string>	*NamedGlobal*	Creates the site object that corresponds to the name provided. The name is interpreted as a site locator and is therefore a URL. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Properties

Key Phrase	*Form*	Return Type	Description
client folder of <site>	*Plain*	<folder>	The folder containing the site content on the client machine. Site content is gathered into this location. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
fixlet of <site>	*Plain*	<fixlet>	Iterates through the Fixlet messages of the specified site. Win:5.0, Lin:5.0, Sol:5.0, HPUX:5.0, AIX:5.0, Mac:6.0
gather schedule authority of <site>	*Plain*	<string>	Returns a string corresponding to the authority of the site schedule, e.g.: Publisher, Custom, Manual or Disabled. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
gather schedule time interval of <site>	*Plain*	<time interval>	Returns the time interval between automatic gathering of site content. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Return Type	Description
group <integer> of <site>	*Numbered*	<site group>	Returns an object corresponding to the numbered group of the specified site. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
last gather time of <site>	*Plain*	<time>	Returns the time of last successful gathering from the site. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
masthead of <site>	*Plain*	<file>	Each site has a masthead, and the masthead is saved into the site data folder upon successful creation. This property returns a file object that corresponds to the copy in the site data folder. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
name of <site>	*Plain*	<string>	The name of the site. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
relevant fixlet of <site>	*Plain*	<fixlet>	Iterates through the Relevant Fixlet messages for the specified site. Win:5.0, Lin:5.0, Sol:5.0, HPUX:5.0, AIX:5.0, Mac:6.0
setting <string> of <site>	*Named*	<setting>	Returns the setting whose name matches the string provided from the Fixlet site settings. Win:2.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
setting of <site>	*Plain*	<setting>	Returns one or more settings from the site settings. Win:2.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
site tag of <site>	*Plain*	<string>	Returns the last component of the specified site's url, eg. ‘actionsite’, ‘enteprisesecurity’, etc. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
site version list of <site>	*Plain*	<__undefined__>	Returns the last gathered site version list (manyversion) of the specified site. Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
subscribe time of <site>	*Plain*	<time>	Returns the time that the current machine began subscribing to the site. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Return Type	Description
type of <site>	*Plain*	<string>	Returns one of the following 4 literal strings: • Master Action Site • Operator Site • Custom Site • Fixlet Site. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
url of <site>	*Plain*	<string>	Returns the Locator found in the masthead. A site locator is used to synchronize with the site. It normally contains the URL of a remote file system folder, or the URL of a cgi-bin program that provides a remote directory listing of the site. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
version of <site>	*Plain*	<integer>	Returns the version number of the site content. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Note

The 'as string' property yields a string formatted with the site name.

Examples

- `exists site "actionsite"`
  - TRUE when the action site exists on the target machine.

- `exists file "siteicon.bmp" of client folder of current site`
  - TRUE if the specified file exists in the client folder.

- `last gather time of current site > now - 30 * day`
  - Return TRUE if it has been over 30 days since last gathering, or synchronizing, with the site.

- `last gather time of current site < time "4 Aug 1997 01:00 pdt"`
  - Returns TRUE if the site was last synchronized before the specified date.

- `modification time of masthead of current site < time "4 Aug 1997 01:00 pdt"`
  - Returns TRUE if the masthead of the current site is older than the specified date.

Site Group

These Inspectors return information on the automatic groups defined for a given site.

Creation Methods

Key Phrase	*Form*	Description
group <integer> of <site>	*Numbered*	Returns an object corresponding to the numbered group of the specified site. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Properties

Key Phrase	*Form*	Return Type	Description
id of <site group>	*Plain*	<integer>	Returns the numeric ID of the specified site group. This is the number assigned to an automatic group when it is first defined. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
member of <site group>	*Plain*	<boolean>	Returns TRUE if the current computer is a member of the specified group. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Fixlet

These Inspectors can provide important information about the Fixlet messages at any site. These Inspectors only work in the context of property evaluation, not Fixlet evaluation.

Creation Methods

Key Phrase	*Form*	Description
fixlet of <site>	*Plain*	This Inspector iterates over all the Fixlet messages in the given site. Win:5.0, Lin:5.0, Sol:5.0, HPUX:5.0, AIX:5.0, Mac:6.0
relevant fixlet of <site>	*Plain*	Iterates over all the relevant Fixlet messages in the specified site. Win:5.0, Lin:5.0, Sol:5.0, HPUX:5.0, AIX:5.0, Mac:6.0

Properties

Key Phrase	*Form*	Return Type	Description
header <string> of <fixlet>	*Named*	<fixlet_header>	Returns the named header (case insensitive) of the specified Fixlet message. Fixlet headers are name:value pairs. Win:5.0, Lin:5.0, Sol:5.0, HPUX:5.0, AIX:5.0, Mac:6.0
header of <fixlet>	*Plain*	<fixlet_header>	Iterates over all the headers of the Fixlet message. Win:5.0, Lin:5.0, Sol:5.0, HPUX:5.0, AIX:5.0, Mac:6.0
id of <fixlet>	*Plain*	<integer>	Returns the numeric ID number of the specified Fixlet message. Win:5.0, Lin:5.0, Sol:5.0, HPUX:5.0, AIX:5.0, Mac:6.0
relevance of <fixlet>	*Plain*	<boolean>	Returns a boolean TRUE or False, depending on the Relevance of the specified Fixlet message. Win:5.0, Lin:5.0, Sol:5.0, HPUX:5.0, AIX:5.0, Mac:6.0

Fixlet_header

Fixlet headers are name:value pairs that can provide important information about the Fixlet messages at any site. These Inspectors only work in the context of property evaluation, not Fixlet evaluation.

Creation Methods

Key Phrase	*Form*	Description
header <string> of <fixlet>	*Named*	Returns the named header (case insensitive) of the specified Fixlet message. Fixlet headers are name:value pairs. Win:5.0, Lin:5.0, Sol:5.0, HPUX:5.0, AIX:5.0, Mac:6.0
header of <fixlet>	*Plain*	Iterates over all the headers of the Fixlet message. Win:5.0, Lin:5.0, Sol:5.0, HPUX:5.0, AIX:5.0, Mac:6.0

Properties

Key Phrase	*Form*	Return Type	Description
name of <fixlet_header>	*Plain*	<string>	Headers are name:value pairs, separated by a colon. This Inspector returns the name on the left hand side of the pair. Win:5.0, Lin:5.0, Sol:5.0, HPUX:5.0, AIX:5.0, Mac:6.0
value of <fixlet_header>	*Plain*	<string>	Headers are name:value pairs, separated by a colon. This Inspector returns the value on the right hand side of the pair. Win:5.0, Lin:5.0, Sol:5.0, HPUX:5.0, AIX:5.0, Mac:6.0

Examples

- `number of relevant fixlets whose (value of header "x-fixlet-source-severity" of it as lowercase = "critical") of site "enterprise security".`
  - Returns the number of critical fixlets in the Enterprise Security site.

Client Objects

These Inspectors retrieve information about the organization of the BES Client computers.

Client

The client object allows access to properties of the client application hosting the relevance evaluation, typically a BigFix program. In addition, the client maintains a collection of settings with both name and value properties that are inspectable using the client object. Client objects are derived from file objects, so they share all the file properties.

Creation Methods

Key Phrase	*Form*	Description
client	*PlainGlobal*	Returns the client object corresponding to the BigFix application evaluating the current relevance expression. Win:2.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Properties

Key Phrase	*Form*	Return Type	Description
administrator <string> of <client>	*Named*	<setting>	If the administrator named in the <string> is enabled on the given <client> computer, this property returns a setting with the given name and the value 'allow.' For instance, if the name of the administrator is joe_admin, then the client would return a setting object with the name 'joe_admin' and a value of 'allow'. Casting this as a string would return 'joe_admin=allow'. Win:3.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:6.0
administrator of <client>	*Plain*	<setting>	Creates a setting object corresponding to the administrators of the given <client>. Win:3.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:6.0
setting <string> of <client>	*Named*	<setting>	Returns a client setting whose name matches the string provided from the client settings. Win:2.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Return Type	Description
setting of <client>	*Plain*	<setting>	Returns one or more settings from the client settings. Win:2.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
version of <client>	*Plain*	<version>	The product version of the BES application (BESClient or QnA). • Note: On the Macintosh only, this Inspector returns a <string>. Lin:4.1, Sol:4.1, HPUX:4.0, AIX:4.1

Examples

- `version of client as string`
- Returns a string like "4.0.3.7".

Setting

A setting is a simple object with name and value properties. It is a property of a client, or a property of a site. Settings of a site are assigned by site authors. Settings of the client are assigned by the BES console operator.

Creation Methods

Key Phrase	*Form*	Description
administrator <string> of <client>	*Named*	Creates a setting with the given name on the given <client> computer. Win:3.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:6.0
administrator of <client>	*Plain*	Creates a setting object consisting of the administrator for the given <client> computer. Win:3.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:6.0
setting <string> of <client>	*Named*	Returns the setting whose name matches the string provided from the client settings. Win:2.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
setting <string> of <site>	*Named*	Returns the setting whose name matches the string provided from the site settings. Win:2.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
setting of <client>	*Plain*	Returns one or more settings from the client settings. Win:2.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Description
setting of <site>	*Plain*	Returns one or more settings from the site settings. Win:2.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Properties

Key Phrase	*Form*	Return Type	Description
<setting> as string	*Cast*	<string>	Returns a string formatted as <name>=<value> for the setting. Win:2.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
effective date of <setting>	*Plain*	<time>	Returns the date when the setting was last modified. Win:2.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
enabled of <setting>	*Plain*	<boolean>	Returns TRUE if the specified setting is enabled. Win:7.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:6.0, Mac:4.1
name of <setting>	*Plain*	<string>	Returns the name of the setting. Win:2.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
value of <setting>	*Plain*	<string>	Returns the value of the setting. Win:2.0, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Examples

- `names of settings of site "actionsite"`

  Returns the names of all the settings of the site named "actionsite".

Selected Server

The BES Server or BES Relay to which the BigFix agent or client reports.

Creation Methods

Key Phrase	*Form*	Description
selected server	*PlainGlobal*	The BES Server or BES Relay to which the agent reports. Returned as the "selected server" type. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1

Properties

Key Phrase	*Form*	Return Type	Description
competition size of <selected server>	*Plain*	<integer>	The number of servers in the competition from which this server was selected. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
competition weight of <selected server>	*Plain*	<integer>	The total of the weights of the servers in the competition from which this server was selected. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
distance of <selected server>	*Plain*	<integer range>	The distance, in IP gateway hops, to the server. Among servers with the same priority, closer servers are preferred. Returns an integer range, since the exact distance may not be known. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
gateway address <integer> of <selected server>	*Numbered*	<ipv4 address>	The ip address of a gateway between the agent and the selected server at the given distance from the agent, if known. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
gateway address of <selected server>	*Plain*	<ipv4 address>	All known ip addresses of gateways between the agent and the selected server. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
ip address of <selected server>	*Plain*	<ipv4 address>	The ip address to which reports are sent. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
name of <selected server>	*Plain*	<string>	The DNS name of the server, if known. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Return Type	Description
port number of <selected server>	*Plain*	<integer>	The port number to which reports are sent. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
priority of <selected server>	*Plain*	<integer>	The priority assigned to the server by the BES console. Servers with low priorities are preferred to servers with high priority. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
weight of <selected server>	*Plain*	<integer>	The weight assigned to the server by the BES console. Servers with the same priority and approximate distance compete to be chosen; servers with higher weights are more likely to be chosen. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1

Current Relay

These Inspectors refer to the BES Server or Relay that the client last registered with.

Creation Methods

Key Phrase	*Form*	Description
current relay	*PlainGlobal*	Returns an object corresponding to the server or relay that the client last registered with. This may be a BES Relay or the BES root server. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Properties

Key Phrase	*Form*	Return Type	Description
version of <current relay>	*Plain*	<version>	Returns a version object that is the version of the server that the client last registered with. This may be a BES Relay or the BES root server. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Root Server

These Inspectors refer to the root server that the Bes Client is currently connected to.

Creation Methods

Key Phrase	*Form*	Description
root server	*PlainGlobal*	Returns an object representing the root BES Server to which the client last registered. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Properties

Key Phrase	*Form*	Return Type	Description
host name of <root server>	*Plain*	<string>	The host (DNS) name of the BES root server that the BES Client last registered with. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
id of <root server>	*Plain*	<integer>	The DSA Server ID of the BES root server that the BES Client last registered with. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0

Environment Objects

The environment objects are provided to access environment variables. Note that you are inspecting the environment of the application executing the relevance clause (typically the BigFix Agent/Client), which may or may not match the environment of other applications on the computer.

Environment

Environment variables define a particular set of paths and variables for a computer or an application. These Inspectors let you examine this set.

Creation Methods

Key Phrase	*Form*	Description
environment	*PlainGlobal*	Creates the one and only environment object. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1

Properties

Key Phrase	*Form*	Return Type	Description
variable <string> of <environment>	*Named*	<environment variable>	Returns an environment variable that matches the given name. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
variable of <environment>	*Plain*	<environment variable>	Iterates through all the environment variables defined. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1

Examples

- `exists environment`
  - TRUE if the computer has an environment object.

- `value of variable "PATH" of environment contains "/sbin"`
  - TRUE if there is an environment variable named "PATH" and its value contains "/sbin".

- `value of variable "PATH" of environment contains "/sbin"`
  - TRUE if there is an environment variable named "PATH" and its value contains "/sbin".

Environment Variable

Every variable defined by the environment has both a name and a value. Both names and values are treated as strings.

Creation Methods

Key Phrase	*Form*	Description
variable <string> of <environment>	*Named*	Creates the variable of the environment matching the name provided. The capitalization of the name is ignored. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
variable of <environment>	*Plain*	Iterates through all the environment variables defined. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1

Properties

Key Phrase	*Form*	Return Type	Description
<environment variable> as string	*Cast*	<string>	Casting the variable as a string yields a string containing the variable name and the value of the variable separated by ' = '. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
name of <environment variable>	*Plain*	<string>	Returns the name of the variable. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1
value of <environment variable>	*Plain*	<string>	Returns the value of the variable. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:5.1

Examples

- `exists variable "PATH" of environment`
  - TRUE if a path variable has been defined in this environment.

- `number of variables of environment`
  - Returns the total number of variables in this environment.

User Objects

User

The user object allows you to inspect properties of all logged-on users.

Creation Methods

Key Phrase	*Form*	Description
current user	*PlainGlobal*	Creates an object corresponding to the currently logged-on user. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
user	*PlainGlobal*	Creates objects for all logged-on users. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
user <string>	*NamedGlobal*	Returns an object representing the user specified by <string>. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1

Properties

Key Phrase	*Form*	Return Type	Description
name of <user>	*Plain*	<string>	Returns the name of the user. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1
tty of <user>	*Plain*	<string>	Returns the tty of the user. Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1

Examples

- `names of users`
- Returns a list of all the logged on users.

Action Objects

These are the keywords associated with properties that can be inspected while BigFix Actions are being executed.

Action

These are the keywords associated with properties available for inspection during the execution of BigFix Actions.

Creation Methods

Key Phrase	*Form*	Description
action	*PlainGlobal*	Creates an action object corresponding to the BigFix Action currently being parsed. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:5.1
action <integer>	*NumberedGlobal*	Creates an action object matching the <integer> id. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
active action	*PlainGlobal*	Creates an action object corresponding to the currently executing action. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Properties

Key Phrase	*Form*	Return Type	Description
active of <action>	*Plain*	<boolean>	Returns TRUE if the action is currently running (active). Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
active start time of <action>	*Plain*	<time>	Returns the time the action started. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
complete time of <action>	*Plain*	<time>	Returns the time the action completed. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
constrained of <action>	*Plain*	<boolean>	Returns TRUE if action is unable to run yet. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Return Type	Description
group leader of <action>	*Plain*	<boolean>	Returns TRUE if the action is a group action and the action component is the group leader. When you deploy a mult-action from the BES Console, it constructs a group action with a group leader to control the overall behavior of the action. This inspector is used internally to manage the progress of the group action. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0
id of <action>	*Plain*	<integer>	Returns the numeric ID associated with the specified Action. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
last change time of <action>	*Plain*	<time>	Returns the time when the action state last changed. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
offer accepted of <action>	*Plain*	<boolean>	Returns TRUE when users indicated they want to run the action by accepting the offer presented by the BES Client UI. When an offer has been accepted, the Client evaluates its constraints and runs as soon as conditions allow. Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
offer of <action>	*Plain*	<boolean>	Returns TRUE when the Action is presented as an offer (as indicated by the header "x-offer: 1"). Win:7.0, Lin:7.0, Sol:7.0, HPUX:7.0, AIX:7.0
origin fixlet id of <action>	*Plain*	<integer>	Returns the Fixlet id that contained the action. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
parameter <string> of <action>	*Named*	<string>	Returns the value of parameter <string> for the active action. Parameters only live as long as the action is active. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
pending login of <action>	*Plain*	<boolean>	Returns TRUE if the specified action included an 'action requires login' command, and a login has not yet occurred since the action has run. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
pending of <action>	*Plain*	<boolean>	Returns TRUE if action is available to run. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Return Type	Description
pending restart of <action>	*Plain*	<boolean>	Returns TRUE if the specified action included an 'action requires restart' command and a restart has not occurred since the action has run. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
pending time of <action>	*Plain*	<time>	Returns the time the action became pending. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
status of <action>	*Plain*	<string>	Returns one of the following strings: • Running = when the action is currently active. • Executed = no longer relevant and action has completed. • Not Relevant = action was not relevant. • Waiting = action is relevant, but waiting to run. • Not Executed = action is relevant, unconstrained, but has not yet started. • Failed = action is relevant, unconstrained, has completed, but is still relevant. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1
waiting for download of <action>	*Plain*	<boolean>	Returns TRUE if client is waiting for mirroring server to have downloads required by the action. Win:1.2, Lin:3.1, Sol:3.1, HPUX:4.0, AIX:4.1, Mac:4.1

Networking Objects

This chapter includes the various networking Inspectors.

Network

These are the keywords used to query the local network configuration.

Creation Methods

Key Phrase	*Form*	Description
network	*PlainGlobal*	Creates an object containing properties of the network. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1

Properties

Key Phrase	*Form*	Return Type	Description
interface <integer> of <network>	*Numbered*	<network interface>	Returns the particular interface of the network. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
interface of <network>	*Plain*	<network interface>	Returns all the interfaces of the network. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
ip interface <integer> of <network>	*Numbered*	<network ip interface>	Returns the particular ip interface of the network. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
ip interface of <network>	*Plain*	<network ip interface>	Returns all the ip interfaces of the network. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1

Network Interface

The network interface object describes a generic network interface, and has information about the name and family of that interface. On the Mac these are commonly of type AF_INET, AF_LINK and AF_INET6.

Creation Methods

Key Phrase	*Form*	Description
interface <integer> of <network>	*Numbered*	Creates an object with the specified network interface. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
interface of <network>	*Plain*	Creates an object with all the interfaces of the network. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1

Properties

Key Phrase	*Form*	Return Type	Description
family of <network interface>	*Plain*	<integer>	Returns an family designator of the address family (i.e., 2=AFI_NET). Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1

Examples

- `names of interfaces of network`
- Returns a list of the network interface names, e.g., lo0, gif0, stf0, en0.

Network Ip Interface

In general, the network ip interface object holds locally determined properties of logical network devices configured on the computer. On the Mac, these correspond to interfaces of type AF_INET. The properties that are available depend on the socket support installed on the computer. For Windows computers with winsock 2 support installed, for instance, the information is obtained by an ioctl call and includes Interface address, Interface broadcast address, Interface network mask, Broadcast support flag, Multicast support flag, Loopback interface flag and Point to point interface flag.

Creation Methods

Key Phrase	*Form*	Description
ip interface <integer> of <network>	*Numbered*	Creates an object with the specified ip interface of the network. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
ip interface of <network>	*Plain*	Creates an object or an object list (using the plural keyword) with all the ip interfaces of the network. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1

Properties

Key Phrase	*Form*	Return Type	Description
address of <network ip interface>	*Plain*	<ipv4 address>	Returns the ip address of the ip interface. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
alias of <network ip interface>	*Plain*	<boolean>	Returns TRUE if the network ip interface has an alias defined for it (a virtual device, rather than a physical device). Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1
broadcast address of <network ip interface>	*Plain*	<ipv4 address>	Returns the broadcast address of the interface. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
broadcast support of <network ip interface>	*Plain*	<boolean>	Indicates that broadcast messages are supported by the ip interface. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
loopback of <network ip interface>	*Plain*	<boolean>	Indicates that the particular network ip interface is a loopback interface. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1

Key Phrase	*Form*	Return Type	Description
mac address of <network ip interface>	*Plain*	<string>	Returns the mac address (AKA hardware address) of the network ip interface object. The mac address is formatted as a string of lower case hex digits separated by '-'. Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1
multicast support of <network ip interface>	*Plain*	<boolean>	Indicates that multicast messages are supported by the ip interface. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
name of <network ip interface>	*Plain*	<string>	Returns the name of the network ip interface object. Typical names are lan0, lo0. Virtual interfaces are usually of the form lan0:2. Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1
point to point of <network ip interface>	*Plain*	<boolean>	Indicates that the interface is a point-to-point interface. Usually TRUE for dialup connections. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
subnet address of <network ip interface>	*Plain*	<ipv4 address>	The subnet to which the interface belongs. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
subnet mask of <network ip interface>	*Plain*	<ipv4 address>	The subnet mask of the interface. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1

Examples

- `names of ip interfaces of network`
  - Returns a list of the names of the network IP interfaces, e.g., lo0, en0.

- `addresses of ip interfaces of network`
  - Returns a list of the IP addresses of the network IP interfaces, e.g., 127.0.0.1, 192.168.1.100, etc.

- `address of ip interface whose (loopback of it = false) of network = "192.168.127.127"`
  - Returns TRUE if the given IP address doesn't have loopback.

- `mac address whose (it = "00-61-b1-d1-7d-29") of ip interfaces of network`
  - Returns the mac address of the specified network ip interface object.

Ipv4 Address

This is an Internet Protocol address, version 4. IP addresses are composed of four single-byte integers separated by periods, like "192.5.0.7".

Creation Methods

Key Phrase	*Form*	Description
address of <network ip interface>	*Plain*	Creates an object with the ip address of the interface. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
broadcast address of <network ip interface>	*Plain*	Creates an object with the broadcast address of the interface. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
gateway address <integer> of <selected server>	*Numbered*	The ip address of a gateway between the agent and the selected server at the given distance from the agent, if known. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
gateway address of <selected server>	*Plain*	All known ip addresses of gateways between the agent and the selected server. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
ip address of <selected server>	*Plain*	The ip address to which reports are sent. Win:4.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
ipv4 address <string>	*NamedGlobal*	Creates an object with an ip address for the string provided. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
subnet address of <network ip interface>	*Plain*	Creates an object with the subnet address of the network interface. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
subnet mask of <network ip interface>	*Plain*	Creates an object with the address bitwise ANDed with the subnet mask. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1

Properties

Key Phrase	*Form*	Return Type	Description
<ipv4 address> as string	*Cast*	<string>	Converts the ipv4 address to a string. Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1

Operators

Key phrase	Return Type	Description
<ipv4 address> {cmp} <ipv4 address>	<*boolean*>	Returns a boolean TRUE/FALSE depending on the result of the comparison, where: • {cmp} is one of: =, !=, <, <=, >, >= . Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1
<ipv4 address> {cmp} <string>	<*boolean*>	Returns a boolean TRUE/FALSE depending on the result of the comparison, where: • {cmp} is one of: =, !=, <, <=, >, >= . Win:1.2, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:5.1

Examples

- `exists ip interface whose (address of it = "127.0.0.1" and loopback of it) of network`
  - Returns TRUE if the specified ip interface (with loopback) exists on this computer.

- `addresses of ip interfaces of network`
  - Returns a list of IP addresses configured on the machine.

- `ipv4 address "192.168.100.1"`
  - Returns the four-byte ip address 192.168.100.1.

Introspectors

These Inspectors look into the currently installed relevance engine to retrieve information about specific Inspectors.

Type

Some Inspectors look at the Relevance language itself, inspecting the Inspectors, so to speak. There are several aspects to view, including the types, properties, casts and operators. This group of Inspectors looks at the various type options available from the Relevance language.

Creation Methods

Key Phrase	*Form*	Description
direct object type of <property>	*Plain*	The type (if any) required after the keyword "of" in an expression using the property. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
index type of <property>	*Plain*	The type (if any) required before or without the keyword "of" in an expression using the property. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
left operand type of <binary operator>	*Plain*	The type required before the operator in an expression. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
operand type of <cast>	*Plain*	The type required before the keyword "as" in an expression using the cast. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
operand type of <unary operator>	*Plain*	The type required in an expression using the operator. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
parent of <type>	*Plain*	The types (if any) whose properties are inherited by this type. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
result type of <binary operator>	*Plain*	The type that the operator produces. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
result type of <property>	*Plain*	The type that the property produces. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
result type of <unary operator>	*Plain*	The type that the operator produces. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Description
right operand type of <binary operator>	*Plain*	The type required after the operator in an expression. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
type	*PlainGlobal*	The inspector types. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
type <string>	*NamedGlobal*	The type with the given name. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1

Properties

Key Phrase	*Form*	Return Type	Description
<type> as string	*Cast*	<string>	A string indicating the type. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
cast from of <type>	*Plain*	<cast>	Returns the casts that can be created from the specified <type>. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
name of <type>	*Plain*	<string>	A string naming the type. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
parent of <type>	*Plain*	<type>	The types (if any) whose properties are inherited by this type. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
property <string> of <type>	*Named*	<property>	Returns the Inspector property of the specified string and type. Typically there is more than one property, so this is often used in the plural. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
property of <type>	*Plain*	<property>	Returns the Inspector property of the specified type. Typically there is more than one property, so this is often used in the plural. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
property returning <type> of <type>	*Index<type>*	<property>	Returns Inspectors of the form <type> of <type>. Typically there is more than one property, so this is often used in the plural. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
size of <type>	*Plain*	<integer>	The number of bytes used in the internal representation of an object of the given type. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1

Operators

Key phrase	Return Type	Description
<type> = <type>	<*boolean*>	Returns TRUE if both expressions denote the same type. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1

Property

Some Inspectors look at the Relevance language itself, inspecting the Inspectors, so to speak. There are several aspects to view, including the types, properties, casts and operators. This group of Inspectors looks at the properties available from the Relevance language.

Creation Methods

Key Phrase	*Form*	Description
property	*PlainGlobal*	The inspectors invoked with phrases, but without the keyword "as". Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
property <string>	*NamedGlobal*	Returns the first inspector property whose name matches the given string. Note that there may be more than one property with a given name. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
property <string> of <type>	*Named*	Returns the Inspector property of the specified string and type. Typically there is more than one property, so this is often used in the plural. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
property of <type>	*Plain*	Returns the Inspector property of the specified type. Typically there is more than one property, so this is typically used in the plural. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
property returning <type>	*Index<type>Global*	Produces a list of the Inspector properties that return the "file" type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
property returning <type> of <type>	*Index<type>*	Returns Inspectors of the form <type> of <type>. Typically there is more than one property, so this is often used in the plural. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Properties

Key Phrase	*Form*	Return Type	Description
<property> as string	*Cast*	<string>	A short description of the use of the property. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
direct object type of <property>	*Plain*	<type>	The type (if any) required after the keyword "of" in an expression using the property. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
index type of <property>	*Plain*	<type>	The type (if any) required before or without the keyword "of" in an expression using the property. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
multivalued of <property>	*Plain*	<boolean>	Can the property have more than one value for a single input?. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
plural name of <property>	*Plain*	<string>	The name of the property, in the plural. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
result type of <property>	*Plain*	<type>	The type that the property produces. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
singular name of <property>	*Plain*	<string>	The name of the property, in the singular. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
usual name of <property>	*Plain*	<string>	Returns the usual name of the specified property. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Binary Operator

Some Inspectors look at the Relevance language itself, inspecting the Inspectors, so to speak. There are several aspects to view, including the types, properties, casts and operators. This group of Inspectors looks at the various binary operators available from the Relevance language.

Creation Methods

Key Phrase	*Form*	Description
binary operator	*PlainGlobal*	The inspectors that have two parameters, and are invoked with punctuation marks or reserved phrases. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
binary operator <string>	*NamedGlobal*	Typically used in the plural, returns the various possible binary inspectors that use the specified operators. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
binary operator returning <type>	*Index<type>Global*	Returns a list of binary operators that return the specified type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Properties

Key Phrase	*Form*	Return Type	Description
<binary operator> as string	*Cast*	<string>	A short description of the use of the operator. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
left operand type of <binary operator>	*Plain*	<type>	The type required before the operator in an expression. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
name of <binary operator>	*Plain*	<string>	A phrase naming the operator. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
result type of <binary operator>	*Plain*	<type>	The type that the operator produces. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
right operand type of <binary operator>	*Plain*	<type>	The type required after the operator in an expression. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Return Type	Description
symbol of <binary operator>	*Plain*	<string>	A phrase or punctuation mark used to invoke the operator. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1

Unary Operator

Some Inspectors look at the Relevance language itself, inspecting the Inspectors, so to speak. There are several aspects to view, including the types, properties, casts and operators. This group of Inspectors looks at the various unary operators available from the Relevance language.

Creation Methods

Key Phrase	*Form*	Description
unary operator	*PlainGlobal*	The inspectors that have one parameter, and are invoked with punctuation marks or reserved phrases. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
unary operator <string>	*NamedGlobal*	Typically used in the plural, this inspector returns a list of objects that use the specified operator. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
unary operator returning <type>	*Index<type>Global*	Returns a list of the unary operator inspectors (such as negative) that return the specified type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Properties

Key Phrase	*Form*	Return Type	Description
<unary operator> as string	*Cast*	<string>	A short description of the use of the operator. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
name of <unary operator>	*Plain*	<string>	A phrase naming the operator. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
operand type of <unary operator>	*Plain*	<type>	The type required in an expression using the operator. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
result type of <unary operator>	*Plain*	<type>	The type that the operator produces. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1

Key Phrase	*Form*	Return Type	Description
symbol of <unary operator>	*Plain*	<string>	A phrase or punctuation mark used to invoke the operator. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1

Cast

Some Inspectors look at the Relevance language itself, inspecting the Inspectors, so to speak. There are several aspects to view, including the types, properties, casts and operators. This group of Inspectors looks at the various casting operations available from the Relevance language.

Creation Methods

Key Phrase	*Form*	Description
cast	*PlainGlobal*	The inspectors invoked using the keyword "as.". Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
cast <string>	*NamedGlobal*	Returns a list of the objects that can be cast into the type specified by <string>. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
cast from of <type>	*Plain*	Returns the casts that can be created from the specified <type>. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0
cast returning <type>	*Index<type>Global*	Returns a list of the objects that can be cast into the specified type. Win:6.0, Lin:6.0, Sol:6.0, HPUX:6.0, AIX:6.0, Mac:6.0

Properties

Key Phrase	*Form*	Return Type	Description
<cast> as string	*Cast*	<string>	A short description of the use of the cast. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
name of <cast>	*Plain*	<string>	The phrase used after the keyword "as" in an expression using the cast. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1
operand type of <cast>	*Plain*	<type>	The type required before the keyword "as" in an expression using the cast. Win:5.1, Lin:4.1, Sol:4.1, HPUX:4.1, AIX:4.1, Mac:4.1

Key Phrases (Inspectors)

This chapter provides an alphabetical list of the Inspector keywords and their casting operators. Both lists include the context object type (From an object), and the resulting object type (Creates an object). These lists are not all-inclusive; they only include those Inspectors that are relevant to the context of the current Guide.

Key phrases

This is a list of the key phrases relevant to this document, sorted alphabetically.

Key Phrase	Plural	Creates a	From a	*Form*
absolute value of <hertz>	absolute values	<hertz>	<hertz>	*Plain*
absolute value of <integer>	absolute values	<integer>	<integer>	*Plain*
absolute value of <time interval>	absolute values	<time interval>	<time interval>	*Plain*
accessed time of <filesystem object>	accessed times	<time>	<filesystem object>	*Plain*
action	actions	<action>	<world>	*PlainGlobal*
action <integer>	actions	<action>	<world>	*NumberedGlobal*
action lock state	action lock states	<action lock state>	<world>	*PlainGlobal*
active action	active actions	<action>	<world>	*PlainGlobal*
active of <action>	actives	<boolean>	<action>	*Plain*
active start time of <action>	active start times	<time>	<action>	*Plain*
address of <network ip interface>	addresses	<ipv4 address>	<network ip interface>	*Plain*
administrator <string> of <client>	administrators	<setting>	<client>	*Named*
administrator of <client>	administrators	<setting>	<client>	*Plain*
alias of <network ip interface>	aliases	<boolean>	<network ip interface>	*Plain*
ancestor of <filesystem object>	ancestors	<folder>	<filesystem object>	*Plain*
apparent registration server time	apparent registration server times	<time>	<world>	*PlainGlobal*
application <string>	applications	<application>	<world>	*NamedGlobal*
application <string> of <folder>	applications	<application>	<folder>	*Named*

Key Phrase	Plural	Creates a	From a	*Form*
application usage summary	application usage summaries	<application usage summary>	<world>	*PlainGlobal*
application usage summary <string>	application usage summaries	<application usage summary>	<world>	*NamedGlobal*
april	aprils	<month>	<world>	*PlainGlobal*
april <integer>	aprils	<day of year>	<world>	*NumberedGlobal*
april <integer> of <integer>	aprils	<date>	<integer>	*Numbered*
april of <integer>	aprils	<month and year>	<integer>	*Plain*
arch of <pkginfo>	archs	<string>	<pkginfo>	*Plain*
architecture of <operating system>	architectures	<string>	<operating system>	*Plain*
august	augusts	<month>	<world>	*PlainGlobal*
august <integer>	augusts	<day of year>	<world>	*NumberedGlobal*
august <integer> of <integer>	augusts	<date>	<integer>	*Numbered*
august of <integer>	augusts	<month and year>	<integer>	*Plain*
base of <patch>	bases	<string>	<patch>	*Plain*
bes license	bes licenses	<license>	<world>	*PlainGlobal*
binary operator <string>	binary operators	<binary operator>	<world>	*NamedGlobal*
binary operator returning <type>	binary operators returning	<binary operator>	<world>	*Index<type>Global*
bit <integer>	bits	<bit set>	<world>	*NumberedGlobal*
bit <integer> of <bit set>	bits	<boolean>	<bit set>	*Numbered*
bit <integer> of <integer>	bits	<boolean>	<integer>	*Numbered*
bit set <string>	bit sets	<bit set>	<world>	*NamedGlobal*
boolean <string>	booleans	<boolean>	<world>	*NamedGlobal*
boot time of <operating system>	boot times	<time>	<operating system>	*Plain*
broadcast address of <network ip interface>	broadcast addresses	<ipv4 address>	<network ip interface>	*Plain*
broadcast support of <network ip interface>	broadcast supports	<boolean>	<network ip interface>	*Plain*
build of <operating system>	builds	<string>	<operating system>	*Plain*

Key Phrase	Plural	Creates a	From a	*Form*
byte <integer> of <file>	bytes	<integer>	<file>	*Numbered*
case insensitive regex <string>	case insensitive regexes	<regular expression>	<world>	*NamedGlobal*
case insensitive regular expression <string>	case insensitive regular expressions	<regular expression>	<world>	*NamedGlobal*
cast <string>	casts	<cast>	<world>	*NamedGlobal*
cast from of <type>	casts from	<cast>	<type>	*Plain*
cast returning <type>	casts returning	<cast>	<world>	*Index<type>Global*
category of <pkginfo>	categories	<string>	<pkginfo>	*Plain*
character <integer>	characters	<string>	<world>	*NumberedGlobal*
character <integer> of <string>	characters	<substring>	<string>	*Numbered*
character of <string>	characters	<substring>	<string>	*Plain*
client	clients	<client>	<world>	*PlainGlobal*
client folder of <site>	client folders	<folder>	<site>	*Plain*
client license	client licenses	<license>	<world>	*PlainGlobal*
common name of <license>	common names	<string>	<license>	*Plain*
competition size of <selected server>	competition sizes	<integer>	<selected server>	*Plain*
competition weight of <selected server>	competition weights	<integer>	<selected server>	*Plain*
complete time of <action>	complete times	<time>	<action>	*Plain*
computer id	computer ids	<integer>	<world>	*PlainGlobal*
computer name	computer names	<string>	<world>	*PlainGlobal*
concatenation <html> of <html>	concatenations	<html>	<html>	*Index<html>*
concatenation <html> of <string>	concatenations	<html>	<string>	*Index<html>*
concatenation <string> of <string>	concatenations	<string>	<string>	*Named*
concatenation of <string>	concatenations	<string>	<string>	*Plain*
conjunction of <boolean>	conjunctions	<boolean>	<boolean>	*Plain*
constrained of <action>	constraineds	<boolean>	<action>	*Plain*
content of <file>	contents	<file content>	<file>	*Plain*
controller of <action lock state>	controllers	<string>	<action lock state>	*Plain*
current date	current dates	<date>	<world>	*PlainGlobal*
current day_of_month	current days_of_month	<day of month>	<world>	*PlainGlobal*

Key Phrase	Plural	Creates a	From a	*Form*
current day_of_week	current days_of_week	<day of week>	<world>	*PlainGlobal*
current day_of_year	current days_of_year	<day of year>	<world>	*PlainGlobal*
current month	current months	<month>	<world>	*PlainGlobal*
current month_and_year	current months_and_years	<month and year>	<world>	*PlainGlobal*
current relay	current relays	<current relay>	<world>	*PlainGlobal*
current site	current sites	<site>	<world>	*PlainGlobal*
current time_of_day	current times_of_day	<time of day with time zone>	<world>	*PlainGlobal*
current time_of_day <time zone>	current times_of_day	<time of day with time zone>	<world>	*Index<time zone>Global*
current user	current users	<current user>	<world>	*PlainGlobal*
current user	current users	<user>	<world>	*PlainGlobal*
current year	current years	<year>	<world>	*PlainGlobal*
custom site subscription effective date <string>	custom site subscription effective dates	<time>	<world>	*NamedGlobal*
date <string>	dates	<date>	<world>	*NamedGlobal*
date <time zone> of <time>	dates	<date>	<time>	*Index<time zone>*
date of <bios>	dates	<string>	<bios>	*Plain*
day	days	<time interval>	<world>	*PlainGlobal*
day of <day of year>	days	<day of month>	<day of year>	*Plain*
day_of_month <integer>	days_of_month	<day of month>	<world>	*NumberedGlobal*
day_of_month <string>	days_of_month	<day of month>	<world>	*NamedGlobal*
day_of_month of <date>	days_of_month	<day of month>	<date>	*Plain*
day_of_week <string>	days_of_week	<day of week>	<world>	*NamedGlobal*
day_of_week of <date>	days_of_week	<day of week>	<date>	*Plain*
day_of_year of <date>	days_of_year	<day of year>	<date>	*Plain*
december	decembers	<month>	<world>	*PlainGlobal*
december <integer>	decembers	<day of year>	<world>	*NumberedGlobal*
december <integer> of <integer>	decembers	<date>	<integer>	*Numbered*

Key Phrase	Plural	Creates a	From a	*Form*
december of <integer>	decembers	<month and year>	<integer>	*Plain*
default web browser	default web browsers	<file>	<world>	*PlainGlobal*
descendant folder of <folder>	descendant folders	<folder>	<folder>	*Plain*
descendant of <folder>	descendants	<file>	<folder>	*Plain*
direct object type of <property>	direct object types	<type>	<property>	*Plain*
disjunction of <boolean>	disjunctions	<boolean>	<boolean>	*Plain*
distance of <selected server>	distances	<integer range>	<selected server>	*Plain*
dns name	dns names	<string>	<world>	*PlainGlobal*
domain name	domain names	<string>	<world>	*PlainGlobal*
domainname	domainnames	<string>	<world>	*PlainGlobal*
drive	drives	<filesystem>	<world>	*PlainGlobal*
drive <string>	drives	<drive>	<world>	*NamedGlobal*
drive <string>	drives	<filesystem>	<world>	*NamedGlobal*
drive of <folder>	drives	<filesystem>	<folder>	*Plain*
effective date of <action lock state>	effective dates	<time>	<action lock state>	*Plain*
effective date of <setting>	effective dates	<time>	<setting>	*Plain*
element of <integer set>	elements	<integer>	<integer set>	*Plain*
element of <string set>	elements	<string>	<string set>	*Plain*
email address of <license>	email addresses	<string>	<license>	*Plain*
enabled of <setting>	enableds	<boolean>	<setting>	*Plain*
end of <substring>	ends	<string position>	<substring>	*Plain*
end of <time range>	ends	<time>	<time range>	*Plain*
environment	environments	<environment>	<world>	*PlainGlobal*
error <string>	errors	<undefined>	<world>	*NamedGlobal*
evaluation of <license>	evaluations	<boolean>	<license>	*Plain*
execute of <mode_mask>	executes	<boolean>	<mode_mask>	*Plain*
expiration date of <action lock state>	expiration dates	<time>	<action lock state>	*Plain*
expiration date of <license>	expiration dates	<time>	<license>	*Plain*
expiration state of <license>	expiration states	<string>	<license>	*Plain*

Key Phrase	Plural	Creates a	From a	*Form*
false	falses	<boolean>	<world>	*PlainGlobal*
family name of <processor>	family names	<string>	<processor>	*Plain*
family of <network interface>	families	<integer>	<network interface>	*Plain*
family of <processor>	families	<string>	<processor>	*Plain*
february	februarys	<month>	<world>	*PlainGlobal*
february <integer>	februarys	<day of year>	<world>	*NumberedGlobal*
february <integer> of <integer>	februarys	<date>	<integer>	*Numbered*
february of <integer>	februarys	<month and year>	<integer>	*Plain*
file <string>	files	<file>	<world>	*NamedGlobal*
file <string> of <folder>	files	<file>	<folder>	*Named*
file count of <filesystem>	file counts	<integer>	<filesystem>	*Plain*
file of <folder>	files	<file>	<folder>	*Plain*
filesystem	filesystems	<filesystem>	<world>	*PlainGlobal*
filesystem <string>	filesystems	<filesystem>	<world>	*NamedGlobal*
filesystem of <file>	filesystems	<filesystem>	<file>	*Plain*
final part <time interval> of <time range>	final parts	<time range>	<time range>	*Index<time interval>*
find file <string> of <folder>	find files	<file>	<folder>	*Named*
first <day of week> of <month and year>	firsts	<date>	<month and year>	*Index<day of week>*
first <integer> of <string>	firsts	<substring>	<string>	*Numbered*
first <string> of <string>	firsts	<substring>	<string>	*Named*
first friday of <month and year>	first fridays	<date>	<month and year>	*Plain*
first match <regular expression> of <string>	first matches	<regular expression match>	<string>	*Index<regular expression>*
first monday of <month and year>	first mondays	<date>	<month and year>	*Plain*
first saturday of <month and year>	first saturdays	<date>	<month and year>	*Plain*
first start time of <application usage summary>	first start times	<time>	<application usage summary>	*Plain*

Key Phrase	Plural	Creates a	From a	*Form*
first sunday of <month and year>	first sundays	<date>	<month and year>	*Plain*
first thursday of <month and year>	first thursdays	<date>	<month and year>	*Plain*
first tuesday of <month and year>	first tuesdays	<date>	<month and year>	*Plain*
first wednesday of <month and year>	first wednesdays	<date>	<month and year>	*Plain*
fixlet of <site>	fixlets	<fixlet>	<site>	*Plain*
folder <string>	folders	<folder>	<world>	*NamedGlobal*
folder <string> of <folder>	folders	<folder>	<folder>	*Named*
folder of <folder>	folders	<folder>	<folder>	*Plain*
following text of <string position>	following texts	<substring>	<string position>	*Plain*
following text of <substring>	following texts	<substring>	<substring>	*Plain*
fputype of <processor>	fputypes	<string>	<processor>	*Plain*
free amount of <ram>	free amounts	<integer>	<ram>	*Plain*
free file count of <filesystem>	free file counts	<integer>	<filesystem>	*Plain*
free percent of <filesystem>	free percents	<integer>	<filesystem>	*Plain*
free space of <filesystem>	free spaces	<integer>	<filesystem>	*Plain*
friday	fridays	<day of week>	<world>	*PlainGlobal*
gateway address <integer> of <selected server>	gateway addresses	<ipv4 address>	<selected server>	*Numbered*
gateway address of <selected server>	gateway addresses	<ipv4 address>	<selected server>	*Plain*
gather schedule authority of <site>	gather schedule authoritys	<string>	<site>	*Plain*
gather schedule time interval of <site>	gather schedule time intervals	<time interval>	<site>	*Plain*
gather url of <license>	gather urls	<string>	<license>	*Plain*
ghz	ghzs	<hertz>	<world>	*PlainGlobal*
gid of <filesystem object>	gids	<integer>	<filesystem object>	*Plain*
greatest hz	greatest hzs	<hertz>	<world>	*PlainGlobal*
greatest integer	greatest integers	<integer>	<world>	*PlainGlobal*
greatest revision of <patch>	greatest revisions	<string>	<patch>	*Plain*

Key Phrase	Plural	Creates a	From a	*Form*
greatest time interval	greatest time intervals	<time interval>	<world>	*PlainGlobal*
group <integer> of <site>	groups	<site group>	<site>	*Numbered*
group execute of <filesystem object>	group executes	<boolean>	<filesystem object>	*Plain*
group leader of <action>	group leaders	<boolean>	<action>	*Plain*
group mask of <filesystem object>	group masks	<integer>	<filesystem object>	*Plain*
group mask of <mode>	group masks	<mode_mask>	<mode>	*Plain*
group name of <filesystem object>	group names	<string>	<filesystem object>	*Plain*
group read of <filesystem object>	group reads	<boolean>	<filesystem object>	*Plain*
group write of <filesystem object>	group writes	<boolean>	<filesystem object>	*Plain*
header <string> of <fixlet>	headers	<fixlet_header>	<fixlet>	*Named*
header of <fixlet>	headers	<fixlet_header>	<fixlet>	*Plain*
hexadecimal integer <string>	hexadecimal integers	<integer>	<world>	*NamedGlobal*
hexadecimal string <string>	hexadecimal strings	<string>	<world>	*NamedGlobal*
host name	host names	<string>	<world>	*PlainGlobal*
host name of <root server>	host names	<string>	<root server>	*Plain*
hostname	hostnames	<string>	<world>	*PlainGlobal*
hour	hours	<time interval>	<world>	*PlainGlobal*
hour_of_day of <time of day with time zone>	hours_of_day	<integer>	<time of day with time zone>	*Plain*
hour_of_day of <time of day>	hours_of_day	<integer>	<time of day>	*Plain*
hz	hzs	<hertz>	<world>	*PlainGlobal*
id of <action>	ids	<integer>	<action>	*Plain*
id of <fixlet>	ids	<integer>	<fixlet>	*Plain*
id of <process>	ids	<integer>	<process>	*Plain*
id of <processor>	ids	<integer>	<processor>	*Plain*
id of <root server>	ids	<integer>	<root server>	*Plain*
id of <site group>	ids	<integer>	<site group>	*Plain*
index of <processor>	indexes	<integer>	<processor>	*Plain*
index type of <property>	index types	<type>	<property>	*Plain*

Key Phrase	Plural	Creates a	From a	*Form*
initial part <time interval> of <time range>	initial parts	<time range>	<time range>	*Index<time interval>*
integer <integer>	integers	<integer>	<world>	*NumberedGlobal*
integer <string>	integers	<integer>	<world>	*NamedGlobal*
interface <integer> of <network>	interfaces	<network interface>	<network>	*Numbered*
interface of <network>	interfaces	<network interface>	<network>	*Plain*
intersection of <integer set>	intersections	<integer set>	<integer set>	*Plain*
intersection of <string set>	intersections	<string set>	<string set>	*Plain*
ip address of <selected server>	ip addresses	<ipv4 address>	<selected server>	*Plain*
ip interface <integer> of <network>	ip interfaces	<network ip interface>	<network>	*Numbered*
ip interface of <network>	ip interfaces	<network ip interface>	<network>	*Plain*
ipv4 address <string>	ipv4 addresses	<ipv4 address>	<world>	*NamedGlobal*
january	januarys	<month>	<world>	*PlainGlobal*
january <integer>	januarys	<day of year>	<world>	*NumberedGlobal*
january <integer> of <integer>	januarys	<date>	<integer>	*Numbered*
january of <integer>	januarys	<month and year>	<integer>	*Plain*
july	julys	<month>	<world>	*PlainGlobal*
july <integer>	julys	<day of year>	<world>	*NumberedGlobal*
july <integer> of <integer>	julys	<date>	<integer>	*Numbered*
july of <integer>	julys	<month and year>	<integer>	*Plain*
june	junes	<month>	<world>	*PlainGlobal*
june <integer>	junes	<day of year>	<world>	*NumberedGlobal*
june <integer> of <integer>	junes	<date>	<integer>	*Numbered*
june of <integer>	junes	<month and year>	<integer>	*Plain*
key <string> of <file section>	keys	<string>	<file section>	*Named*
key <string> of <file>	keys	<string>	<file>	*Named*
khz	khzs	<hertz>	<world>	*PlainGlobal*
last <integer> of <string>	lasts	<substring>	<string>	*Numbered*

Key Phrase	Plural	Creates a	From a	*Form*
last <string> of <string>	lasts	<substring>	<string>	*Named*
last change time of <action>	last change times	<time>	<action>	*Plain*
last gather time of <site>	last gather times	<time>	<site>	*Plain*
last start time of <application usage summary>	last start times	<time>	<application usage summary>	*Plain*
last time seen of <application usage summary>	last times seen	<time>	<application usage summary>	*Plain*
leap of <year>	leaps	<boolean>	<year>	*Plain*
least hz	least hzs	<hertz>	<world>	*PlainGlobal*
least integer	least integers	<integer>	<world>	*PlainGlobal*
least revision of <patch>	least revisions	<string>	<patch>	*Plain*
least significant one bit of <bit set>	least significant one bits	<integer>	<bit set>	*Plain*
least time interval	least time intervals	<time interval>	<world>	*PlainGlobal*
left operand type of <binary operator>	left operand types	<type>	<binary operator>	*Plain*
left shift <integer> of <bit set>	left shifts	<bit set>	<bit set>	*Numbered*
length of <month and year>	lengths	<time interval>	<month and year>	*Plain*
length of <rope>	lengths	<integer>	<rope>	*Plain*
length of <string>	lengths	<integer>	<string>	*Plain*
length of <time range>	lengths	<time interval>	<time range>	*Plain*
length of <year>	lengths	<time interval>	<year>	*Plain*
line <integer> of <file>	lines	<file line>	<file>	*Numbered*
line containing <string> of <file>	lines containing	<file line>	<file>	*Named*
line number of <file line>	line numbers	<integer>	<file line>	*Plain*
line of <file>	lines	<file line>	<file>	*Plain*
line starting with <string> of <file>	lines starting with	<file line>	<file>	*Named*
local time <string>	local times	<time>	<world>	*NamedGlobal*
local time zone	local time zones	<time zone>	<world>	*PlainGlobal*
location of <filesystem object>	locations	<string>	<filesystem object>	*Plain*
lock string of <action lock state>	lock strings	<string>	<action lock state>	*Plain*

Key Phrase	Plural	Creates a	From a	*Form*
locked of <action lock state>	lockeds	<boolean>	<action lock state>	*Plain*
loopback of <network ip interface>	loopbacks	<boolean>	<network ip interface>	*Plain*
lower bound of <integer range>	lower bounds	<integer>	<integer range>	*Plain*
mac address of <network ip interface>	mac addresses	<string>	<network ip interface>	*Plain*
main gather service	main gather services	<service>	<world>	*PlainGlobal*
main processor	main processors	<processor>	<world>	*PlainGlobal*
march	marchs	<month>	<world>	*PlainGlobal*
march <integer>	marchs	<day of year>	<world>	*NumberedGlobal*
march <integer> of <integer>	marchs	<date>	<integer>	*Numbered*
march of <integer>	marchs	<month and year>	<integer>	*Plain*
masthead of <site>	mastheads	<file>	<site>	*Plain*
match <regular expression> of <string>	matches	<regular expression match>	<string>	*Index<regular expression>*
maximum of <integer>	maxima	<integer>	<integer>	*Plain*
maximum of <time interval>	maxima	<time interval>	<time interval>	*Plain*
maximum of <time>	maxima	<time>	<time>	*Plain*
maximum seat count of <license>	maximum seat counts	<integer>	<license>	*Plain*
may	mays	<month>	<world>	*PlainGlobal*
may <integer>	mays	<day of year>	<world>	*NumberedGlobal*
may <integer> of <integer>	mays	<date>	<integer>	*Numbered*
may of <integer>	mays	<month and year>	<integer>	*Plain*
member of <site group>	members	<boolean>	<site group>	*Plain*
mhz	mhzs	<hertz>	<world>	*PlainGlobal*
microsecond	microseconds	<time interval>	<world>	*PlainGlobal*
midnight	midnights	<time of day>	<world>	*PlainGlobal*
millisecond	milliseconds	<time interval>	<world>	*PlainGlobal*
minimum of <integer>	minima	<integer>	<integer>	*Plain*
minimum of <time interval>	minima	<time interval>	<time interval>	*Plain*
minimum of <time>	minima	<time>	<time>	*Plain*

Key Phrase	Plural	Creates a	From a	*Form*
minute	minutes	<time interval>	<world>	*PlainGlobal*
minute_of_hour of <time of day with time zone>	minutes_of_hour	<integer>	<time of day with time zone>	*Plain*
minute_of_hour of <time of day>	minutes_of_hour	<integer>	<time of day>	*Plain*
mode of <filesystem object>	modes	<mode>	<filesystem object>	*Plain*
model of <processor>	models	<string>	<processor>	*Plain*
modification time of <filesystem object>	modification times	<time>	<filesystem object>	*Plain*
module <string>	modules	<module>	<world>	*NamedGlobal*
monday	mondays	<day of week>	<world>	*PlainGlobal*
month	months	<number of months>	<world>	*PlainGlobal*
month <integer>	months	<month>	<world>	*NumberedGlobal*
month <string>	months	<month>	<world>	*NamedGlobal*
month of <date>	months	<month>	<date>	*Plain*
month of <day of year>	months	<month>	<day of year>	*Plain*
month of <month and year>	months	<month>	<month and year>	*Plain*
month_and_year of <date>	months_and_years	<month and year>	<date>	*Plain*
most significant one bit of <bit set>	most significant one bits	<integer>	<bit set>	*Plain*
multicast support of <network ip interface>	multicast supports	<boolean>	<network ip interface>	*Plain*
multiplicity of <integer with multiplicity>	multiplicities	<integer>	<integer with multiplicity>	*Plain*
multiplicity of <string with multiplicity>	multiplicities	<integer>	<string with multiplicity>	*Plain*
multivalued of <property>	multivalueds	<boolean>	<property>	*Plain*
name of <application usage summary>	names	<string>	<application usage summary>	*Plain*
name of <binary operator>	names	<string>	<binary operator>	*Plain*
name of <cast>	names	<string>	<cast>	*Plain*
name of <environment variable>	names	<string>	<environment variable>	*Plain*

Key Phrase	Plural	Creates a	From a	*Form*
name of <filesystem object>	names	<string>	<filesystem object>	*Plain*
name of <filesystem>	names	<string>	<filesystem>	*Plain*
name of <fixlet_header>	names	<string>	<fixlet_header>	*Plain*
name of <network ip interface>	names	<string>	<network ip interface>	*Plain*
name of <operating system>	names	<string>	<operating system>	*Plain*
name of <pkginfo>	names	<string>	<pkginfo>	*Plain*
name of <process>	names	<string>	<process>	*Plain*
name of <selected server>	names	<string>	<selected server>	*Plain*
name of <setting>	names	<string>	<setting>	*Plain*
name of <site>	names	<string>	<site>	*Plain*
name of <type>	names	<string>	<type>	*Plain*
name of <unary operator>	names	<string>	<unary operator>	*Plain*
name of <user>	names	<string>	<user>	*Plain*
network	networks	<network>	<world>	*PlainGlobal*
next line of <file line>	next lines	<file line>	<file line>	*Plain*
noon	noons	<time of day>	<world>	*PlainGlobal*
november	novembers	<month>	<world>	*PlainGlobal*
november <integer>	novembers	<day of year>	<world>	*NumberedGlobal*
november <integer> of <integer>	novembers	<date>	<integer>	*Numbered*
november of <integer>	novembers	<month and year>	<integer>	*Plain*
now	nows	<time>	<world>	*PlainGlobal*
numeric value of <string>	numeric values	<integer>	<string>	*Plain*
october	octobers	<month>	<world>	*PlainGlobal*
october <integer>	octobers	<day of year>	<world>	*NumberedGlobal*
october <integer> of <integer>	octobers	<date>	<integer>	*Numbered*
october of <integer>	octobers	<month and year>	<integer>	*Plain*
offer accepted of <action>	offer accepteds	<boolean>	<action>	*Plain*
offer of <action>	offers	<boolean>	<action>	*Plain*

Key Phrase	Plural	Creates a	From a	*Form*
one bit of <bit set>	one bits	<integer>	<bit set>	*Plain*
operand type of <cast>	operand types	<type>	<cast>	*Plain*
operand type of <unary operator>	operand types	<type>	<unary operator>	*Plain*
operating system	operating systems	<operating system>	<world>	*PlainGlobal*
organization of <license>	organizations	<string>	<license>	*Plain*
origin fixlet id of <action>	origin fixlet ids	<integer>	<action>	*Plain*
other execute of <filesystem object>	other executes	<boolean>	<filesystem object>	*Plain*
other mask of <filesystem object>	other masks	<integer>	<filesystem object>	*Plain*
other mask of <mode>	other masks	<mode_mask>	<mode>	*Plain*
other read of <filesystem object>	other reads	<boolean>	<filesystem object>	*Plain*
other write of <filesystem object>	other writes	<boolean>	<filesystem object>	*Plain*
param <string> of <pkginfo>	params	<string>	<pkginfo>	*Named*
parameter <string>	parameters	<string>	<world>	*NamedGlobal*
parameter <string> of <action>	parameters	<string>	<action>	*Named*
parent folder of <filesystem object>	parent folders	<folder>	<filesystem object>	*Plain*
parent of <type>	parents	<type>	<type>	*Plain*
parenthesized part <integer> of <regular expression match>	parenthesized parts	<substring>	<regular expression match>	*Numbered*
parenthesized part of <regular expression match>	parenthesized parts	<substring>	<regular expression match>	*Plain*
patch <string> of <pkgdb>	patches	<patch>	<pkgdb>	*Named*
patch <string> of <pkginfo>	patches	<patch>	<pkginfo>	*Named*
patch id <string> of <pkgdb>	patch ids	<string>	<pkgdb>	*Named*
patch id <string> of <pkginfo>	patch ids	<string>	<pkginfo>	*Named*
patch id of <pkgdb>	patch ids	<string>	<pkgdb>	*Plain*
patch id of <pkginfo>	patch ids	<string>	<pkginfo>	*Plain*
patch of <pkgdb>	patches	<patch>	<pkgdb>	*Plain*
patch of <pkginfo>	patches	<patch>	<pkginfo>	*Plain*

Key Phrase	Plural	Creates a	From a	*Form*
pathname of <filesystem object>	pathnames	<string>	<filesystem object>	*Plain*
pending login	pending logins	<boolean>	<world>	*PlainGlobal*
pending login of <action>	pending logins	<boolean>	<action>	*Plain*
pending of <action>	pendings	<boolean>	<action>	*Plain*
pending restart	pending restarts	<boolean>	<world>	*PlainGlobal*
pending restart <string>	pending restarts	<boolean>	<world>	*NamedGlobal*
pending restart of <action>	pending restarts	<boolean>	<action>	*Plain*
pending time of <action>	pending times	<time>	<action>	*Plain*
pid of <process>	pids	<integer>	<process>	*Plain*
pkgdb	pkgdbs	<pkgdb>	<world>	*PlainGlobal*
pkginfo <string> of <pkgdb>	pkginfos	<pkginfo>	<pkgdb>	*Named*
pkginfo of <pkgdb>	pkginfos	<pkginfo>	<pkgdb>	*Plain*
pkginst of <pkginfo>	pkginsts	<string>	<pkginfo>	*Plain*
platform id of <language>	platform ids	<string>	<language>	*Plain*
plural name of <property>	plural names	<string>	<property>	*Plain*
point to point of <network ip interface>	point to points	<boolean>	<network ip interface>	*Plain*
port number of <selected server>	port numbers	<integer>	<selected server>	*Plain*
position <integer> of <string>	positions	<string position>	<string>	*Numbered*
position of <string>	positions	<string position>	<string>	*Plain*
preceding text of <string position>	preceding texts	<substring>	<string position>	*Plain*
preceding text of <substring>	preceding texts	<substring>	<substring>	*Plain*
previous line of <file line>	previous lines	<file line>	<file line>	*Plain*
primary language of <language>	primary languages	<primary language>	<language>	*Plain*
priority of <selected server>	priorities	<integer>	<selected server>	*Plain*
process	processes	<process>	<world>	*PlainGlobal*
process <integer>	processes	<process>	<world>	*NumberedGlobal*
process <string>	processes	<process>	<world>	*NamedGlobal*
process id of <process>	process ids	<integer>	<process>	*Plain*

Key Phrase	Plural	Creates a	From a	*Form*
processor	processors	<processor>	<world>	*PlainGlobal*
processor <integer>	processors	<processor>	<world>	*NumberedGlobal*
product of <integer>	products	<integer>	<integer>	*Plain*
property <string>	properties	<property>	<world>	*NamedGlobal*
property <string> of <type>	properties	<property>	<type>	*Named*
property of <type>	properties	<property>	<type>	*Plain*
property returning <type>	properties returning	<property>	<world>	*Index<type>Global*
property returning <type> of <type>	properties returning	<property>	<type>	*Index<type>*
ram	rams	<ram>	<world>	*PlainGlobal*
random access memory	random access memories	<ram>	<world>	*PlainGlobal*
range after <time> of <time range>	ranges after	<time range>	<time range>	*Index<time>*
range before <time> of <time range>	ranges before	<time range>	<time range>	*Index<time>*
read of <mode_mask>	reads	<boolean>	<mode_mask>	*Plain*
regex <string>	regexes	<regular expression>	<world>	*NamedGlobal*
registrar number of <license>	registrar numbers	<integer>	<license>	*Plain*
regular expression <string>	regular expressions	<regular expression>	<world>	*NamedGlobal*
relay service	relay services	<service>	<world>	*PlainGlobal*
release of <operating system>	releases	<string>	<operating system>	*Plain*
relevance of <fixlet>	relevances	<boolean>	<fixlet>	*Plain*
relevant fixlet of <site>	relevant fixlets	<fixlet>	<site>	*Plain*
result type of <binary operator>	result types	<type>	<binary operator>	*Plain*
result type of <property>	result types	<type>	<property>	*Plain*
result type of <unary operator>	result types	<type>	<unary operator>	*Plain*
revision <string> of <patch>	revisions	<string>	<patch>	*Named*
revision of <patch>	revisions	<string>	<patch>	*Plain*
right operand type of <binary operator>	right operand types	<type>	<binary operator>	*Plain*
right shift <integer> of <bit set>	right shifts	<bit set>	<bit set>	*Numbered*
root folder	root folders	<folder>	<world>	*PlainGlobal*

Key Phrase	Plural	Creates a	From a	*Form*
root server	root servers	<root server>	<world>	*PlainGlobal*
rope <string>	ropes	<rope>	<world>	*NamedGlobal*
running of <application usage summary>	runnings	<boolean>	<application usage summary>	*Plain*
saturday	saturdays	<day of week>	<world>	*PlainGlobal*
seat count state of <license>	seat count states	<string>	<license>	*Plain*
seat of <license>	seats	<integer>	<license>	*Plain*
second	seconds	<time interval>	<world>	*PlainGlobal*
second_of_minute of <time of day with time zone>	seconds_of_minute	<integer>	<time of day with time zone>	*Plain*
second_of_minute of <time of day>	seconds_of_minute	<integer>	<time of day>	*Plain*
section <string> of <file>	sections	<file section>	<file>	*Named*
selected server	selected servers	<selected server>	<world>	*PlainGlobal*
september	septembers	<month>	<world>	*PlainGlobal*
september <integer>	septembers	<day of year>	<world>	*NumberedGlobal*
september <integer> of <integer>	septembers	<date>	<integer>	*Numbered*
september of <integer>	septembers	<month and year>	<integer>	*Plain*
service <string>	services	<service>	<world>	*NamedGlobal*
set of <integer>	sets	<integer set>	<integer>	*Plain*
set of <string>	sets	<string set>	<string>	*Plain*
setgid of <filesystem object>	setgids	<boolean>	<filesystem object>	*Plain*
setgid of <mode>	setgids	<boolean>	<mode>	*Plain*
setting <string> of <client>	settings	<setting>	<client>	*Named*
setting <string> of <site>	settings	<setting>	<site>	*Named*
setting of <client>	settings	<setting>	<client>	*Plain*
setting of <site>	settings	<setting>	<site>	*Plain*
setuid of <filesystem object>	setuids	<boolean>	<filesystem object>	*Plain*
setuid of <mode>	setuids	<boolean>	<mode>	*Plain*
sha1 of <file>	sha1s	<string>	<file>	*Plain*

Key Phrase	Plural	Creates a	From a	*Form*
significant digits <integer> of <hertz>	significant digitss	<hertz>	<hertz>	*Numbered*
significant digits <integer> of <integer>	significant digitss	<integer>	<integer>	*Numbered*
singular name of <property>	singular names	<string>	<property>	*Plain*
site	sites	<site>	<world>	*PlainGlobal*
site <string>	sites	<site>	<world>	*NamedGlobal*
site number of <license>	site numbers	<integer>	<license>	*Plain*
site tag of <site>	site tags	<string>	<site>	*Plain*
size of <file>	sizes	<integer>	<file>	*Plain*
size of <filesystem>	sizes	<integer>	<filesystem>	*Plain*
size of <integer set>	sizes	<integer>	<integer set>	*Plain*
size of <ram>	sizes	<integer>	<ram>	*Plain*
size of <string set>	sizes	<integer>	<string set>	*Plain*
size of <type>	sizes	<integer>	<type>	*Plain*
speed of <processor>	speeds	<hertz>	<processor>	*Plain*
start date of <license>	start dates	<time>	<license>	*Plain*
start of <substring>	starts	<string position>	<substring>	*Plain*
start of <time range>	starts	<time>	<time range>	*Plain*
state of <processor>	states	<string>	<processor>	*Plain*
state of <service>	states	<string>	<service>	*Plain*
status of <action>	statuss	<string>	<action>	*Plain*
sticky of <mode>	stickies	<boolean>	<mode>	*Plain*
string <string>	strings	<string>	<world>	*NamedGlobal*
subnet address of <network ip interface>	subnet addresses	<ipv4 address>	<network ip interface>	*Plain*
subnet mask of <network ip interface>	subnet masks	<ipv4 address>	<network ip interface>	*Plain*
subscribe time of <site>	subscribe times	<time>	<site>	*Plain*
substring <string> of <string>	substrings	<substring>	<string>	*Named*
substring after <string> of <string>	substrings after	<substring>	<string>	*Named*
substring before <string> of <string>	substrings before	<substring>	<string>	*Named*
substring between <string> of <string>	substrings between	<substring>	<string>	*Named*

Key Phrase	Plural	Creates a	From a	*Form*
substring separated by <string> of <string>	substrings separated by	<substring>	<string>	*Named*
sum of <integer>	sums	<integer>	<integer>	*Plain*
sunday	sundays	<day of week>	<world>	*PlainGlobal*
symbol of <binary operator>	symbols	<string>	<binary operator>	*Plain*
symbol of <unary operator>	symbols	<string>	<unary operator>	*Plain*
system language	system languages	<string>	<world>	*PlainGlobal*
system locale	system locales	<language>	<world>	*PlainGlobal*
system ui language	system ui languages	<language>	<world>	*PlainGlobal*
thursday	thursdays	<day of week>	<world>	*PlainGlobal*
time <string>	times	<time>	<world>	*NamedGlobal*
time <time zone> of <time>	times	<time of day with time zone>	<time>	*Index<time zone>*
time interval <string>	time intervals	<time interval>	<world>	*NamedGlobal*
time of <time of day with time zone>	times	<time of day>	<time of day with time zone>	*Plain*
time zone <string>	time zones	<time zone>	<world>	*NamedGlobal*
time_of_day <string>	times_of_day	<time of day>	<world>	*NamedGlobal*
total amount of <ram>	total amounts	<integer>	<ram>	*Plain*
total duration of <application usage summary>	total durations	<time interval>	<application usage summary>	*Plain*
total run count of <application usage summary>	total run counts	<integer>	<application usage summary>	*Plain*
total space of <filesystem>	total spaces	<integer>	<filesystem>	*Plain*
true	trues	<boolean>	<world>	*PlainGlobal*
tty of <user>	ttys	<string>	<user>	*Plain*
tuesday	tuesdays	<day of week>	<world>	*PlainGlobal*
two digit hour of <time of day with time zone>	two digit hours	<string>	<time of day with time zone>	*Plain*
two digit hour of <time of day>	two digit hours	<string>	<time of day>	*Plain*

Key Phrase	Plural	Creates a	From a	*Form*
two digit minute of <time of day with time zone>	two digit minutes	<string>	<time of day with time zone>	*Plain*
two digit minute of <time of day>	two digit minutes	<string>	<time of day>	*Plain*
two digit second of <time of day with time zone>	two digit seconds	<string>	<time of day with time zone>	*Plain*
two digit second of <time of day>	two digit seconds	<string>	<time of day>	*Plain*
type of <filesystem>	types	<string>	<filesystem>	*Plain*
type of <processor>	types	<integer>	<processor>	*Plain*
type of <processor>	types	<string>	<processor>	*Plain*
type of <site>	types	<string>	<site>	*Plain*
uid of <filesystem object>	uids	<integer>	<filesystem object>	*Plain*
unary operator <string>	unary operators	<unary operator>	<world>	*NamedGlobal*
unary operator returning <type>	unary operators returning	<unary operator>	<world>	*Index<type>Global*
union of <integer set>	unions	<integer set>	<integer set>	*Plain*
union of <string set>	unions	<string set>	<string set>	*Plain*
unique value of <integer>	unique values	<integer with multiplicity>	<integer>	*Plain*
unique value of <string>	unique values	<string with multiplicity>	<string>	*Plain*
universal time <string>	universal times	<time>	<world>	*NamedGlobal*
universal time zone	universal time zones	<time zone>	<world>	*PlainGlobal*
upper bound of <integer range>	upper bounds	<integer>	<integer range>	*Plain*
uptime of <operating system>	uptimes	<time interval>	<operating system>	*Plain*
url of <site>	urls	<string>	<site>	*Plain*
used amount of <ram>	used amounts	<integer>	<ram>	*Plain*
used file count of <filesystem>	used file counts	<integer>	<filesystem>	*Plain*
used percent of <filesystem>	used percents	<integer>	<filesystem>	*Plain*
used space of <filesystem>	used spaces	<integer>	<filesystem>	*Plain*
user	users	<user>	<world>	*PlainGlobal*
user <string>	users	<user>	<world>	*NamedGlobal*

Key Phrase	Plural	Creates a	From a	*Form*
user execute of <filesystem object>	user executes	<boolean>	<filesystem object>	*Plain*
user mask of <filesystem object>	user masks	<integer>	<filesystem object>	*Plain*
user mask of <mode>	user masks	<mode_mask>	<mode>	*Plain*
user name of <filesystem object>	user names	<string>	<filesystem object>	*Plain*
user read of <filesystem object>	user reads	<boolean>	<filesystem object>	*Plain*
user write of <filesystem object>	user writes	<boolean>	<filesystem object>	*Plain*
usual name of <property>	usual names	<string>	<property>	*Plain*
value of <environment variable>	values	<string>	<environment variable>	*Plain*
value of <fixlet_header>	values	<string>	<fixlet_header>	*Plain*
value of <setting>	values	<string>	<setting>	*Plain*
variable <string> of <environment>	variables	<environment variable>	<environment>	*Named*
variable of <environment>	variables	<environment variable>	<environment>	*Plain*
variable of <file>	variables	<string>	<file>	*Plain*
vendor of <pkginfo>	vendors	<string>	<pkginfo>	*Plain*
version <string>	versions	<version>	<world>	*NamedGlobal*
version of <bios>	versions	<string>	<bios>	*Plain*
version of <client>	versions	<version>	<client>	*Plain*
version of <current relay>	versions	<version>	<current relay>	*Plain*
version of <pkginfo>	versions	<string>	<pkginfo>	*Plain*
version of <site>	versions	<integer>	<site>	*Plain*
version string <string> of <module>	version strings	<string>	<module>	*Named*
waiting for download of <action>	waiting for downloads	<boolean>	<action>	*Plain*
wednesday	wednesdays	<day of week>	<world>	*PlainGlobal*
week	weeks	<time interval>	<world>	*PlainGlobal*
weight of <selected server>	weights	<integer>	<selected server>	*Plain*
write of <mode_mask>	writes	<boolean>	<mode_mask>	*Plain*

Key Phrase	Plural	Creates a	From a	*Form*
year	years	<number of months>	<world>	*PlainGlobal*
year <integer>	years	<year>	<world>	*NumberedGlobal*
year <string>	years	<year>	<world>	*NamedGlobal*
year of <date>	years	<year>	<date>	*Plain*
year of <month and year>	years	<year>	<month and year>	*Plain*
zone of <time of day with time zone>	zones	<time zone>	<time of day with time zone>	*Plain*
zoned time_of_day <string>	zoned times_of_day	<time of day with time zone>	<world>	*NamedGlobal*

Casting Operators

The casting operators allow you to convert one type to another.This is a list of the casting operators sorted by key phrase.

Key Phrase	Creates a	From a
<action lock state> as string	<string>	<action lock state>
<binary operator> as string	<string>	<binary operator>
<bios> as string	<string>	<bios>
<bit set> as integer	<integer>	<bit set>
<bit set> as string	<string>	<bit set>
<boolean> as boolean	<boolean>	<boolean>
<boolean> as string	<string>	<boolean>
<cast> as string	<string>	<cast>
<date> as string	<string>	<date>
<day of month> as integer	<integer>	<day of month>
<day of month> as string	<string>	<day of month>
<day of month> as two digits	<string>	<day of month>

Key Phrase	Creates a	From a
<day of week> as string	<string>	<day of week>
<day of week> as three letters	<string>	<day of week>
<day of year> as string	<string>	<day of year>
<environment variable> as string	<string>	<environment variable>
<file content> as lowercase	<file content>	<file content>
<file content> as uppercase	<file content>	<file content>
<file> as string	<string>	<file>
<filesystem object> as string	<string>	<filesystem object>
<folder> as string	<string>	<folder>
<hertz> as string	<string>	<hertz>
<html> as html	<html>	<html>
<html> as string	<string>	<html>
<integer> as bit set	<bit set>	<integer>
<integer> as bits	<bit set>	<integer>
<integer> as day_of_month	<day of month>	<integer>
<integer> as hexadecimal	<string>	<integer>
<integer> as integer	<integer>	<integer>
<integer> as month	<month>	<integer>
<integer> as string	<string>	<integer>
<integer> as year	<year>	<integer>
<ipv4 address> as string	<string>	<ipv4 address>
<language> as string	<string>	<language>
<mode_mask> as integer	<integer>	<mode_mask>
<mode_mask> as string	<string>	<mode_mask>
<mode> as octal string	<string>	<mode>
<mode> as string	<string>	<mode>
<month and year> as string	<string>	<month and year>
<month> as integer	<integer>	<month>

Key Phrase	Creates a	From a
<month> as string	<string>	<month>
<month> as three letters	<string>	<month>
<month> as two digits	<string>	<month>
<number of months> as string	<string>	<number of months>
<operating system> as string	<string>	<operating system>
<pkginfo> as string	<string>	<pkginfo>
<primary language> as string	<string>	<primary language>
<property> as string	<string>	<property>
<rope> as string	<string>	<rope>
<setting> as string	<string>	<setting>
<string> as boolean	<boolean>	<string>
<string> as date	<date>	<string>
<string> as day_of_month	<day of month>	<string>
<string> as day_of_week	<day of week>	<string>
<string> as hexadecimal	<string>	<string>
<string> as html	<html>	<string>
<string> as integer	<integer>	<string>
<string> as left trimmed string	<string>	<string>
<string> as local time	<time>	<string>
<string> as local zoned time_of_day	<time of day with time zone>	<string>
<string> as lowercase	<string>	<string>
<string> as month	<month>	<string>
<string> as right trimmed string	<string>	<string>
<string> as string	<string>	<string>
<string> as time	<time>	<string>
<string> as time interval	<time interval>	<string>

Key Phrase	Creates a	From a
<string> as time zone	<time zone>	<string>
<string> as time_of_day	<time of day>	<string>
<string> as trimmed string	<string>	<string>
<string> as universal time	<time>	<string>
<string> as universal zoned time_of_day	<time of day with time zone>	<string>
<string> as uppercase	<string>	<string>
<string> as version	<version>	<string>
<string> as year	<year>	<string>
<string> as zoned time_of_day	<time of day with time zone>	<string>
<time interval> as string	<string>	<time interval>
<time of day with time zone> as string	<string>	<time of day with time zone>
<time of day> as string	<string>	<time of day>
<time range> as string	<string>	<time range>
<time zone> as string	<string>	<time zone>
<time> as local string	<string>	<time>
<time> as string	<string>	<time>
<time> as universal string	<string>	<time>
<type> as string	<string>	<type>
<unary operator> as string	<string>	<unary operator>
<version> as string	<string>	<version>
<year> as integer	<integer>	<year>
<year> as string	<string>	<year>

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Y

Z