Bravura Security

Bravura PSLang

Reference Manual

Software revision: 12.8.0
Document revision: 39090
Last changed: 2025-05-16

Contents

Introduction 1

This manual describes the PSLANG language. This manual should be used when writing scripts to customize various parts of the Bravura Security Fabric, such as interface programs and plugin scripts. The PSLANG language provides functionality to support all Bravura Security Fabric applications; as such, only some of the functionality contained in this reference manual may apply to the particular application you are using.

PSLANG is a scripting language with a syntax much like C, but with a large set of built-in functions, some of which are geared towards the Bravura Security Fabric environment.

PSLANG's features include:

- User-defined functions
- Automatic memory management
- File and socket input/output, including SSL support
- Associative arrays
- A full set of flow-control structures and arithmetic expressions
- Regular expressions

Variables 2

This chapter explains the types of variables in PSLANG and how to use them.

2.1 Using Variables

The use of variables in PSLANG follows these rules:

- Variables must be declared before use.
- Variables may be assigned an initial value in their declaration statement.
- To declare an array without assigning an initial value to it, the declaration must be written with brackets after the variable name, such as: `var $arr[]`.
- Variable identifiers are prefixed with a dollar sign $.
- Variables may contain special characters to make their names more readable, but they must be enclosed in `{}` braces, such as: `var ${target-system} = target`
- The type of a variable is not specified at declaration and the type of a variable can changed at any time by assigning it a new value.

2.2 Variable Types

PSLANG supports the following variable types:

- Integers
- Floating-point numbers
- Strings
- Input/Output handles
- Integer-index arrays
- Associative (string-indexed) arrays
- KVGroups
- Dates
- Structures

2.2.1 Integers

The exact range of integers will depend on the platform on which PSLANG is running. On 64-bit Windows platforms, integers range from -2^{63} to $2^{63}-1$, or from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.

2.2.2 Floating-point numbers

As with integers, the exact range of a floating-point number depends on the platform on which PSLANG is running. On Windows platforms, floating-point numbers have a minimum precision of approximately $-2.22507 \times 10^{-308}$ and a maximum precision of approximately 1.79769×10^{308}. When floating-point numbers are converted to a string (for example, by adding it to a another string), the number is printed to 6 decimal places.

2.2.3 Strings

A string is a series Unicode characters. A string is indexed starting with zero for the first character. This means `$str[4]` references the fifth Unicode character of the string. ASCII is a subset of the UTF-8 character set. For UTF-8 characters, the code pages for the languages need to be installed in order to display the Unicode characters properly on the Bravura Security Fabric server or computer that is trying to display the information from the web pages.

2.2.4 Input/Output Handles

An input/output handle represents a file, a TCP socket, a pipe, or any other input/output “device”. Thus, you can read, write, and close any handle without needing to know the underlying device.

2.2.5 Arrays

An array may contain heterogeneous types. That is, the individual elements in an array do not have to have the same type. There are two types of arrays: integer-indexed and string-indexed.

2.2.5.1 Integer-indexed arrays

Integer-indexed arrays are indexed starting with 0 up to `size($arr)-1`.

> **Note:** `size($arr)` is a function that returns the number of elements in an array.

An integer-indexed array can be expanded by assigning to the index equal to the size of the array. That is, if `size($arr)` is `4` the statement `$arr[4] = 42` expands the array `$arr` by one and assigns `42` to the fifth element in the array, at which point `size($arr)` is `5`.

An integer-indexed array of arrays or *multidimensional* array can be created by assigning an array to another array. For example, `$arr0[0] = 42` then assigning it to another array, `$arr1[0] = $arr0`. The values in the multidimensional array can be accessed as follows, `valAt($arr1[0], 0)`.

> **Note:** Using `$arr1[0][0]` to reference a multidimensional is currently not supported.

2.2.5.2 String-indexed arrays

String-indexed arrays are also called *associative arrays* or *dictionaries*. The following is an example of a string-indexed array:

```
var $arr;
$arr["key one"] = "a value";
$arr["key two"] = "another value";
```

The key-value pairs in a string-indexed array can be accessed with the `keyAt($arr, $i)` and `valAt($arr, $i)` functions, where `$arr` is the array to reference and `$i` is an integer such that `0 ≤ $i < size($arr)`

> **Note:** `keyAt($arr, $i)` is a function that returns the key in an associative array for a given numeric position. See keyAt.
> `valAt($arr, $i)` is a function that returns the value in an associative array for a given numeric position. See valAt.

When an unknown index is used to reference a value in a string-indexed array, 0 (zero) is returned. String-indexed arrays are expanded implicitly by assigning to a previously-unused index.

2.2.6 Structures

Declarations of structures can be global or local and are defined as follows:

```
struct myStruct
{
     var $int
     var $str
     var $arr[]
     var $float
};
```

To create an instance of a structure, write:

```
struct myStruct $struct1
```

To access members of a structure, in this case to assign a value, write:

```
$struct1.$int = 1
```

2.2.7 KVGroups

KVGroup stands for *Key-Value Group*. A KVGroup has a key, a name, and its contents. It contains a set of key-value pairs, as well as a set of "inner" KVGroups. There is no limit to the level of nested KVGroups. KVGroups are used extensively for inter-process communication in Bravura Security Fabric.

The keys (for key-value pairs) and key-name combinations (for groups) are case sensitive.

Example A:

```
"capitals" "group of world capitals" = {
  "USA" = "Washington"
  "Canada" = "Ottawa"
  "Britain" = "London"
}
```

In the above example of a simple group, the group's *key* is "capitals" and its *name* is "group of world capitals". The group contains three key-value pairs mapping countries to capital cities: USA→Washington, Canada→Ottawa, and Britain→London.

> **Note:** The word *key* has a dual use. A group has a key and the items contained within a group (pairs and inner groups) also have keys. KVGroup functions that manipulate a group's key are `kvgCreate`, `kvgSetKey`, and `kvgGetKey`. When other functions use a *key*, the key is used to identify objects within the group.

Example B:

```
"capitals" "world capitals grouped by continent" = {
  "NA" "North America" = {
    "USA" = "Washington"
```



```
    "Canada" = "Ottawa"
  }
  "Eur" "Europe" = {
    "Britain" = "London"
  }
}
```

Example B shows how a KVGroup can contain multiple groups. The main (outer) group contains two groups; the key-name pairs of the contained (inner) groups are (“NA”, “North America”) and (“Eur”, “Europe”).

2.2.8 Dates

Date types are treated as objects in PSLANG. Once a variable is assigned the date value, the date get/set functions can be used on the date variable.

For example:

```
var $oldDate;
toDate($oldDate, "2001-02-01T14:00:00");
setHour($oldDate, 12);
println("hour: " + getHour($oldDate));
```

2.2.8.1 Date representations

A date can be used to store a:

- **date time value**

  The date time value can be expressed as just a date or as a full date, which also includes the time of day. The date only format contains the year (YYYY), month (MM) and day (DD) separated by dashes (-).

  ```
  YYYY-MM-DD
  ```

  A full date contains the year (YYYY), month (MM) and day (DD) separated by dashes (-); a T separator; then the hour (HH), minutes (MM), and seconds (SS) separated by colons (:).

  ```
  YYYY-MM-DDTHH:MM:SS
  ```

- **date time value with interval**

  A date time value with interval is either defined by two dates (start date and end date, interval is deduced), or by a date and an interval. When using a date and an interval, the date can either be a start date (date/interval) or an end date (interval/date). A forward slash (/) is used to separate the two components. Supported formats include:

  ```
  YYYY-MM-DD/YYYY-MM-DD
  YYYY-MM-DDTHH:MM:SS/YYYY-MM-DDTHH:MM:SS
  YYYY-MM-DD/INTERVAL
  YYYY-MM-DDTHH:MM:SS/INTERVAL
  INTERVAL/YYYY-MM-DD
  INTERVAL/YYYY-MM-DDTHH:MM:SS
  ```


The interval definition starts with a P. This can be followed by year (<*n*>Y), month (<*nM*>), and day (<*n*>D) representing the date period, and/or a T separator, followed by hour (<*n*>H), minute (<*n*>M), and second (<*n*>S) representing the time period. The T separator is required if you are defining the time part of the interval.

For example:

```
P1Y       (one year interval)
P1Y5D     (one year and five day interval)
P5DT10H   (five day and ten hour interval)
PT10H     (ten hour interval)
```

- **date time value with recurring interval**

  This builds on the previous date/interval definition. R indicates a recurring interval with n signifying the number of times to repeat the interval. A forward slash (/) is used to separate the recurrence indicator, date, and interval parts of the definition. Supported formats include:

```
Rn/YYYY-MM-DD/YYYY-MM-DD
Rn/YYYY-MM-DDTHH:MM:SS/YYYY-MM-DDTHH:MM:SS
Rn/YYYY-MM-DD/INTERVAL
Rn/YYYY-MM-DDTHH:MM:SS/INTERVAL
Rn/INTERVAL/YYYY-MM-DD
Rn/INTERVAL/YYYY-MM-DDTHH:MM:SS
```

  For example:

```
R5/2008-01-01/2013-01-01
R5/P1Y/2013-01-01
```

> **Note:** The date must be initialized with the desired type. Dates cannot be converted between types after initialization.

> **Note:** The various time functions in Time Functions that handle intervals (for example, setIntervalYear) only work on date variables that have been initialized with an interval or recurring interval.

2.2.8.2 Date specifiers

The date function formatString uses date specifiers to convert a date object to a string.

The supported format codes are summarized as below:

- %Y Year with century, as decimal number
- %y Year without century, as decimal number (00 99)
- %d Day of month as decimal number (01 31)
- %m Month as decimal number (01 12)
- %M Minute as decimal number (00 59)

- `%`H Hour in 24-hour format (00 23)
- `%`S Second as decimal number (00 59)
- `%`e Like `%d`, the day of the month as a decimal number, but a leading zero is replaced by a space.
- `%`k The hour (24-hour clock) as a decimal number (range 0 to 23); single digits are preceded by a blank.
- `%`l The hour (12-hour clock) as a decimal number (range 1 to 12); single digits are preceded by a blank.
- `%`p Either ‘AM’ or ‘PM’ according to the given time value, or the corresponding strings for the current locale.
- `%`j The day of the year as a decimal number (range 001 to 366)
- `%`U The week number of the current year as a decimal number, range 00 to 53
- `%`s The number of seconds since the Epoch,
- `%%` the character `%`

All unsupported codes will be ignored.

2.2.8.3 Date calculations

Date types support the following calculations:

- time +/- interval
  Example: `$time + $interval;`

  > **Note:** Intervals that result in a date between 12:00 AM, January 1, 1970 UTC and 12:00 AM, January 1, 2038 are supported.

- interval +/- interval
  Example: `$interval - $interval;`
- interval * integer
  Example: `$interval * 3`
- time + string
- time comparison
- interval comparison

Language Syntax 3

The syntax of PSLANG is very similar to that of C/C++ and Java. Notable exceptions are:

- Statements in PSLANG may be terminated with semicolons, but they are not required.
- Variables are prefixed with the dollar sign ($).
- The bodies of if, for, and while statements *must* be contained within braces { }.
- Labels are surrounded by brackets [ ], written as follows: `[mylabel]`
- Like Java, but unlike C, the + (addition) operator is defined on strings. The operation appends the right argument to the left argument. If one of the arguments is a number, then the number is converted to a string before performing the operation.

3.1 Comments

PSLANG supports three comment styles:

- C/C++/Java-style comments: /* ... */ can span multiple lines. Unlike C, C++ and Java, this style of comments may be nested.
- C++/Java-style comments: // denotes a comment that extends to the end of the line.
- Unix shell/Perl-style comments: # denotes a comment that extends to the end of the line.

3.2 Basic Types and Literals

Three types of literals exist, one for each basic type:

- Integers: `0`, `4`, `-42`
- Floating-point numbers: `0.0`, `3.2`, `-48.91492501`
- Strings: `"This is a string"`

3.2.1 Implicit conversion

PSLANG can implicitly convert an expression (p11) of a given basic type into another basic type. When you assign an integer value to a floating point value PSLANG automatically converts the int to a float. For example:

```
var $x = 1.23;  // define a float
var $y = 4;     // define an int
$y = $y + $x;   // convert $y to a float
```

PSLANG functions that are passed a float value but expect an int or vice versa automatically convert the value to the expected type. Functions that expect string values cannot accept floats or ints. You must force the conversion using the concatenation operator. For example:

```
var $x = 1234;   // define an int
print ("" + $x); // treat as a string
```

There is no automatic conversion from strings to ints or floats. Instead, you can use the *strtol()* and *strtod()* functions (see String Functions).

> **Note:** Converting from a float to an int may result in a loss of precision. This behavior is the same as in C.

3.2.2 Special characters

Certain characters must be escaped with a backslash to be written into a string literal: double-quote (`\"`), backslash (`\\`), carriage return (`\r`), line feed (`\n`), and horizontal tab (`\t`). Other unprintable characters can be created with the *chr()* function (see chr).

> **Note:** If you incorrectly escape a character that doesn't require escaping, PSLANG will output a warning message and ignore the backslash (`\`).

3.3 Expressions

An expression is some sequence of operations that yields a result. A literal value, a variable, or a function by themselves can be an expression. The following are valid expressions:

```
5
"3"
myfunction( 1 )
5+3
5 + "3"
"5" + "3"
"5" + myfunction( innerfunction() - 6 )
$x + $y + $z
```

A PSLANG expression can include a sequence of variable declarations (including initializers) and function calls. The final *statement* must be an expression; its value is taken as the overall result.

Here is a sequence of statements:

```
var $a = "hello";
var $b = ", world";
var $c = strlen( $a );
$a + $b + " - length: " + $c
```

This is acceptable as a PSLANG expression because the last statement is itself a normal expression. The result here would be: `hello, world - length: 5`.

In the Bravura Security Fabric GUI, the same sequence of statements would be written on one line:

```
var $a = "hello"; var $b = ", world"; var $c = strlen( $a ); $a + $b + " - length:
  →" + $c
```

3.3.1 Types of expressions

PSLANG supports the following types of expressions:

Operation	Description
<string literal>	string literal
<integer literal>	integer literal
– <integer literal>	negate an integer
<floating-point literal>	floating-point literal
– <floating-point literal>	negate a floating-point literal
$var	the value of the variable

$var[expr]	string and array indexing
(expr)	for clarity or to change order of evaluation
expr + expr	addition or string concatenation
expr − expr	subtraction
expr ∗ expr	multiplication
expr / expr	division
expr % expr	modulus
expr isin expr	is left a substring of right?
expr == expr	equality
expr ! = expr	inequality
expr ≤ expr	left is less than or equal to right?
expr < expr	left is less than right?
expr ≥ expr	left is greater than or equal to right?
expr > expr	left is greater than right?
~expr	bitwise negation
expr & expr	bitwise *and*
expr | expr	bitwise *or*
expr && expr	logical *and*
expr|| expr	logical *or*
$var++	post increment operator
$var–	post decrement operator
! expr	logical negation
func(...)	function call
iif($test$, $true\_part$, $false\_part$)	ternary operator

A *boolean expression* yields an integer result, where zero is interpreted as *false*, and any value other than zero is interpreted as *true*.

Bitwise operators (~, &, |) must not be confused with logical operators (!, &&, ||). Although PSLANG does not support "^", the bitwise exclusive or operator (*xor*), you can achieve the same results using: `( A & ~B ) | ( ~A & B )`.

The PSLANG + operator can be used with numerical values, or to append string values. In an expression such as `x+y`, where `x` is a string and `y` is a number, `y` is converted to a string and appended to `x`.

The ternary operator works the same way as the C++ ternary operator (`?:`). The $test$ expression is evaluated, and if the result is non-zero, the $true\_part$ expression is evaluated and returned. Otherwise, the $false\_part$ expression is evaluated and returned.

3.3.2 Order of evaluation

The order of evaluation proceeds according to the precedence of the operators in the expression. The following table lists the precedence of the operators, with the highest priority at the top of the list.

Precedence Level	Operators
1	[ ]
2	! ~
3	$*$ / %
4	$+$ $-$
5	isin
6	$\leq$ $\geq$ $<$ $>$
7	$==$! $=$
8	&
9	|
10	&&
11	||
12	parentheses

Evaluation of the expression moves left to right and stops at the earliest possible pass/fail decision; unnecessary conditions are not evaluated.

3.3.3 PSLANG auto-completion

The user interface can be configured to offer auto-completion of variable or function calls when writing PSLANG expressions. Enabling auto-complete provides quick access to help and formatting information.

> **Note:** Only the PSLANG functions that are in the string, time, and misc sections are available for auto-completion.

The auto-complete drop-down list appears as you type a valid expression into a PSLANG text area. Only variables, functions and completion options that match what you are typing are displayed. For detailed information, click the **?** link to the right of each listed item.

To see the complete drop-down list of all available options:

- Click in an empty PSLANG text field.
- Press the **[↓]** key at a word break while typing an expression in a PSLANG text area.
- Press the **[↓]** key mid-word while typing an expression that does not match any of the available expressions.

Navigation through the list can be done with the mouse, or using the **[↑]** and **[↓]** keys. Selections can be made using the **[Enter]** or **[Tab]** keys. The **[Esc]** key closes the menu.

The PSLANG auto-completion functionality can be disabled or enabled by clicking **disable auto completion** or **enable auto completion** beneath any PSLANG text area.

The default settings for the PSLANG auto-complete functionality are controlled by the **`config.js`** script, located in the \<*instance*>\design\src\js\ directory. In this script, the AutoCompletePslang group has the following options:

Table 3.3: PSLANG auto-complete options in config.js

Option	Description
enabled	Controls whether the script is available to users. The default value is true.
initiallyOff	Controls whether PSLANG auto-completion is enabled when a page loads. The default value is false. If set to true, then PSLANG auto-completion is off by default when a page loads.

3.4 Global Variable Section

While any type of statement is allowed inside a function, only include statements (described later), variable declaration/definition statements, and expression statements are allowed outside of functions. Thus, global variables can be declared and defined before execution begins. Any valid expression may be used to assign an initial value to a variable.

The following types of statements are acceptable in the Global Variable Section:

```
var $one = 1
var $two = $one + 1
var $abc
var $pi = 22.0/7.0   // poor, but easy approximation of pi
var $xyz
```

3.4.1 Hidden Global Variables

The PSLANG interpreter may introduce variables into the global variable section before executing a script. These variables are, in a sense, “hidden” global variables. This normally includes the variables:

`$stdin` The standard input stream,

`$stdout` The standard output stream

`$stderr` The standard error stream.

`$pslangMajorVersion` An integer representing the product major version.

$pslangMinorVersion An integer representing the product minor version.

$pslangPatchVersion An integer representing the product patch version.

$pslangRepositoryVersion An integer representing the product repository version.

$pslangProductVersion A string representing the product version in the format:
<major>,<minor>,<patch>.

$pslangFileVersion A string representing the full file version in the format:
<major>.<minor>.<patch>.<repos>.

$SYSTEM_QUOTE_ARGS Determines whether command-line arguments for system() function need to be quoted or not. If set to 1 (default), arguments are surrounded by double quotes. To disable this, set this variable to 0.

Note that while it is possible to override $stdin and $stdout in PSLANG, it is not recommended. Most errors are written to the instance log (<*Program Files path*>\Hitachi ID\IDM Suite\Logs\<*instance*>\idmsuite.log) and *not* on $stderr, even when running **pslang.exe**.

3.5 Functions

There are two kinds of functions in PSLANG: user-defined functions (written in the script file), and built-in functions (built in to the PSLANG interpreter).

3.5.1 Passing Parameters

A function parameter can have one of four “directions”:

input The parameter is passed by value from the caller. The function may modify its own copy of the value, but the caller’s copy is unchanged after the function returns.

const The parameter is passed by reference but it is not modifiable within the called function. This provides an efficient way to pass large variables (arrays, KVGroups) for read-only purposes. Within the called function, the parameter becomes a “const” variable and may only be used in a read-only manner, and may only be passed to other functions as a value to a const parameter.

output When a function assigns a value to an *output* parameter, the value is passed back when the function is complete into the variable supplied by the caller.

inout Much like an *output* variable, except that the function expects the variable to be set to some value before the function is called.

In most cases, *const* is preferable to *input*, as *const* provides both efficiency and safety.

Global variables can be passed into a functions *inout* parameter allowing the variable to be modified; however, it is not best practice to modify a global variable.

3.5.2 Restrictions

PSLANG may impose certain restrictions when the same variable is (directly) used as more than one argument in a function call. When a variable is used as an argument to a *const* or *input* parameter, the same variable may be used as an argument to any other *const* or *input* parameter. However, when a variable is used as an argument to an *inout* or *output* parameter, the variable may not be used as any other argument in the function.

The rules are more complicated when using elements of an array, but they follow similar logic. That is, if $\$arr$ is an array, then references to elements of $\$arr$ may be considered separate variables in the context of the above rules as long as the array index are different. For example, the function calls in the following script excerpt are legal, regardless of the "direction" of the parameters for $func1$ and $func2$:

```
func1 ( $arr[0], $arr[1] );
func1 ( $arr["first"], $arr["second"] );
func2 ( $arr[42] );
func2 ( $arr["forty-two"] );
```

However, for safety, string and integer indices cannot be used on the same array for one function call if any of the parameters are *inout* or *output*. As such, the following expression would not be legal if any of the parameters for $func1$ are *inout* or *output*:

```
func1 ( $arr[0], $arr["second"] );
```

When using an entire array as one argument in a function, the following rules apply:

- If $\$arr$ is an argument to a *const* or *input* parameter, $\$arr[\$x]$ (where $\$x$ is any string or integer index) may also be passed to any *const* or *input* parameter.
- If $\$arr$ is an argument to a *inout* or *output* parameter, $\$arr[\$x]$ may not be used as another argument to the same function.
- If $\$arr[\$x]$ is an argument to a *inout* or *output* parameter, $\$arr$ may not be used as another argument to the same function.

3.5.3 Variable function call

Variables can include a function call in the following way:

$\$var$.function($\$arg2$, $\$arg3$, ...);

Calling the function in this way treats $\$var$ as the first argument. The variable type must match the first argument type of the function.

An example of calling a function in this way is as follows:

```
var $str = "Hello World!";
$str.println();
```

This is primarily useful for date objects. For example:


```
var $date = now();
$date.setYear(2015);  // equivalent to setYear($date, 2015)
```

The now() function returns the current date and time (UTC).

3.5.4 Return Values

A function can return a value of any type, or it can return none at all. Trying to use the "return value" of a function that does not have a return value results in an error.

3.6 Flow Control Structures

PSLANG has the following flow control structures:

- *if-elsif-else* statements
- *while* loops
- *for* loops
- *break* and *continue* statements
- *goto* statements
- The *iif* (ternary) operator
- *expect* statements

3.6.1 If-Elsif-Else Statements

An if-elsif-else statement is constructed with an *if* clause, followed by zero or more *elsif* clauses, optionally followed by an *else* clause. Some examples:

```
if ( boolean-expression )
  {
  ...statements...
  }


if ( boolean-expression2 )
  {
  ...statements...
  }
else
  {
  ...statements...
  }
```



```
if ( boolean-expression3 )
  {
  ...statements...
  }
elsif ( boolean-expression4 )
  {
  ...statements...
  }
else
  {
  ...statements...
  }


if ( boolean-expression5 )
  {
  ...statements...
  }
elsif ( boolean-expression6 )
  {
  ...statements...
  }
elsif ( boolean-expression7 )
  {
  ...statements...
  }
elsif ( boolean-expression8 )
  {
  ...statements...
  }
```

3.6.2 While Loops

The statements in a *while* loop are executed while the *test* (boolean expression) is true. A while loop is constructed as follows:

```
while ( test )
  {
  ...statements...
  }
```

3.6.3 For Loops

A for loop is much like a while loop; its statements are executed while the *test* boolean expression is true. It provides additional "space" for initializing variables (*init-section*) and for incrementing variables (*increment-section*). It is constructed as follows:

```
for ( init-section; test; increment-section )
  {
  ...statements...
  }
```


The init and increment sections can each have 1 or more statements, separated by commas. The following types of statements are allowed in the init and increment sections:

- Assignment statements
- Variable declaration (optionally with initial value assignment)
- Expressions (including functions)

3.6.4 Break and Continue Statements

As in C/C++, and Java, a *break* statement inside of a *while* or a *for* loop causes execution to immediately jump to the first statement following the end of the looping structure.

A *continue* statement causes execution to jump to the top of the loop. The *test* is still considered before executing the body of the loop. In a *for* loop, the *increment-section* is executed before the *test* is evaluated, as usual.

3.6.5 Goto Statements

A goto statement performs an unconditional jump to the label named in the goto statement. It could be used as follows:

```
var $rc = fopen($handle, "filename", "r")
if ( $rc != 0 )
  {
  goto cleanup
  }
...other statements...
[cleanup]
close($handle)
```

3.6.6 The iif Operator

The ternary operator is not, strictly speaking, a flow-control structure. However, it can be used to replace some simple if-else statements. See Expressions for details.

3.6.7 Expect Statements

An *expect* statement reads from an input handle and tries to match some search patterns. If a pattern is matched, then the control is directed to the label following the pattern. An *expect* statement takes the following form:

```
expect $handle $buffer $maxChars $maxTime
  {
  "pattern1" label1
```



```
$pattern2  label2
overflow   label3
timeout    label4
default    label5
}
```

Where:

- **$handle** is the I/O handle from which to read,
- **$buffer** stores what is read,
- **$maxChars** is the maximum number of characters to read,
- **$maxTime** is the maximum amount of time (in seconds) to wait for input,
- **"pattern1" label1** specifies to go to label1 if "pattern1" is found in the input,
- **$pattern2 label2** specifies to go to label2 if the value of $pattern2 is found in the input,
- **overflow label3** specifies to go to *label3* if more than *maxChars* characters were read,
- **timeout label4** specifies to go to *label4* if the timeout has elapsed.
- **default label5** specifies to go to *label5* if overflow or timeout label is missing and reading is either overflow or timeout.

Additional notes:

- The default label is required if either overflow or timeout are omitted. If both overflow and timeout labels are included, then the default label is optional.
- Any expression evaluating to a string can be used as a pattern. Patterns may include the dot (.) and star (*) wildcard characters. The dot matches any single character, and the star matches 0 or more of any characters.
- The words *default*, *overflow*, and *timeout* are keywords and are written literally.

3.7 Keywords

The following words are keywords (i.e., reserved for use by the language) in PSLANG and may not be used as function or variable names:

break	expect	include	return
const	failure	inout	success
continue	for	input	timeout
default	function	isin	underflow
discard	goto	output	var
else	if	overflow	while

3.8 Including Other Files

With the `include` statement, other files may be "included" into the main PSLANG file. In practice, this means that the global variables and functions declared in the included file are available to the main file. Further, any files included by the included file are also available to the main file, and so on.

The include statement is used as follows:

```
include "otherFile.psl";
```

Include statements may only be written at the global scope of the file, i.e., outside of any function.

As with other statements, the semicolon is optional. Absolute or relative paths may be used, including the use of `..` (two periods) to refer to a parent directory. Relative paths are resolved using the path (and on Windows, the drive) of the *including* file.

The PSLANG interpreter automatically protects against multiple inclusion of the same file, even if their relative pathnames are different.

> **Note:** On Windows, protection against multiple file inclusion does not work across drive letters. For example, if drive f: and drive g: are mapped to the same network share, PSLANG is unable to determine that f:\file.psl and g:\file.psl are the same files.

On Windows, either the slash (`/`) or the backslash (`\`) may be used as a directory separator. Using the slash (`/`) is recommended as it does not require escaping.

Global variable and function names must be unique across the main file and the included files. The PSLANG parser will provide an error if duplicate names occur.

The General-Purpose Interpreter: pslang.exe

4

Typically, a PSLANG program is executed directly by a Bravura Security Fabric service, CGI, or other executable. A general-purpose interpreter, **pslang.exe**, is installed in the util directory. This program is particularly useful for learning to write PSLANG scripts and for checking the syntactical correctness of scripts. PSLANG scripts can also be executed on Unix target systems. A Unix-specific PSLANG interpreter is located in the server directory of the **psunix** archive.

4.1 Simple method of using the interpreter

Type the following on the command line for the simplest way to run:

- **pslang.exe**:

  ```
  pslang.exe <script file name>
  ```

- Unix-specific interpreter:

  ```
  <Unix binary name> <script file name>
  ```

The interpreter loads the given script and runs the function called “main” as the start function. It also attaches the standard file streams stdin, stdout, and stderr to the global variables $stdin, $stdout, and $stderr respectively.

Most errors are written to the instance log (<*Program Files path*>\Hitachi ID\IDM Suite\Logs\<*instance*>\idmsuite.log) and *not* on $stderr.

If the start function provides an integer as a return value, that number is used as the exit code for the **pslang.exe** process. Otherwise, assuming the script executes without error, **pslang.exe** sets its exit code to 0.

4.2 Advanced method of using the interpreter

There are a number of ways to customize the behavior of the Windows and Unix PSLANG executables. The general format of the command line is as follows:

```
pslang.exe [options] <script> [startFunction] [-- arg1 .. argN]
```

where an option begins with a dash (-), and startFunction specifies the function to execute instead of "main". The arguments following `--` may be passed into the script's start function.

4.2.1 Command-line Options for pslang.exe

This section describes the various options that can be passed to **`pslang.exe`**.

4.2.1.1 Option: -parse

Using this option parses the script but does not execute it. When used with this option, **`pslang.exe`** either states that the script is syntactically correct, or it prints errors that occurred during parsing.

A syntactically correct script does not imply that it is a "correct" script. For example, the `-parse` option shows errors such as:

- An incorrect number of parameters for a function
- An opening `/*` for a comment without a closing `*/`

However, there are some errors that can only be caught during script execution. For example, the expression `$var1 * $var2` is syntactically correct, but the expression cannot be evaluated if one or both of `$var1` and `$var2` are strings. Since the types of `$var1` and `$var2` can only be determined at run-time, such an error will not be detected with `-parse`.

4.2.1.2 Option: -r

This option prints the script's return value on the standard error stream, regardless of the return value's type.

4.2.1.3 Option: -in file

This option sets the script's $stdin variable to the given file, rather than the default of standard in.

4.2.1.4 Option: -out file

This option sets the script's $stdout variable to the given file, rather than the default of standard out. The named file is overwritten (i.e., not appended).

4.2.1.5 Option: -err file

This option sets the script's $stderr variable to the given file, rather than the default of standard error. The named file is overwritten (i.e., not appended).

4.2.1.6 Option: -msglevel level

This option specifies the level of PSLANG interpreter messages that will be written. There are four allowed values for level:

- *none*: No messages will be written.
- *err*: Only error-level messages will be written.
- *warn*: Error-level and warning-level messages will be written.
- *info*: All levels (error, warning, and information) of messages will be written.

The messages are written to standard error by default.

4.2.1.7 Option: -expr

This options sets the PSLANG to "expression" execution mode. Any functions in the program are ignored, so only global variable section (see Global Variable Section) is used. The last statement in the global variable section must be an expression-style statement. The value of that expression is provided as the return code.

4.2.1.8 The Script File Name

A dash (`-`) may be used as the script file name, rather than the name of a file. When a dash is used, the PSLANG interpreter reads its program from standard in (stdin). When reading a program in this way, the `include` statement is not supported and reading from $stdin is undefined unless the `-in` option is provided.

4.2.1.9 Start Function and Command-Line Arguments

The start function may have exactly zero or one parameters. If it has one parameter, then the script file name and command-line arguments specified after `--` are passed into that parameter as an array of strings. This parameter must be declared *const*.

Thus, if the command line is `pslang.exe myscript.psl -- one two`, then the parameter passed into the start function will have the following elements:

- `$argv[0] = "myscript.psl"`
- `$argv[1] = "one"`
- `$argv[2] = "two"`

If the parameter is not declared in the start function, any command-line arguments after `--` are ignored.

The following sample code provides a main() function:

```
function main( const $argv )
{
    // Print each element of the $argv array to the screen

    for ( var $i = 0; $i < size( $argv ); $i = $i + 1 ) {
        println( "$argv[ " + $i + " ] = " + $argv[ $i ] );
    }
}
```


Example Scripts 5

This chapter contains a few PSLANG script examples. Two of the most common uses for PSLANG are for plugins, and event actions (exit traps). Other sample scripts may be found in the `samples` directory of the Bravura Security Fabric installation.

5.1 Plugin Script

A plugin script typically reads a KVGroup on standard input, processes the information in some manner, and writes a KVGroup on standard output. See the *Bravura Security Fabric* Reference Manual for more details on plugins.

The script below:

1. Reads a KVGroup from standard input,
2. Checks to ensure the read was successful,
3. Tries to extract a key-value pair called “firstName” from the input,
4. Converts the value of “firstName” to uppercase, if “firstName” is found, and
5. Writes a KVGroup as output.

```
function main()
  {
  var $readOk = 0
  # Read a KVGroup from standard in
  var $inkvg = kvgCreateFromHandle( $stdin, 10, $readOk )
  var $outkvg

  # Create a KVGroup for output
  $outkvg = kvgCreate( "", "" )
  if( $readOk != 0 )
    {
    kvgAddValue( $outkvg, "returnmsg", "Failed to read input." )
    kvgAddValue( $outkvg, "retval", "1" )
    kvgToHandle( $outkvg, $stdout, 0 )
    return 1
    }

  var $firstName
  if( kvgGetValue( $inkvg, "firstName", $firstName ) == 0 )
    {
```

```
    kvgAddValue( $outkvg, "returnmsg", "Input does not contain firstName" )
    kvgAddValue( $outkvg, "retval", "1" )
    kvgToHandle( $outkvg, $stdout, 0 )
    return 1
    }
  $firstName = toupper( $firstName )

  kvgAddValue( $outkvg, "firstName", $firstName )
  # Write the resulting KVGroup to standard out
  kvgToHandle( $outkvg, $stdout, 0 )
  return 0
  }
```

5.2 Event Configuration Script

When a PSLANG program is used in an exit trap, the type of exit trap will determine which function is executed in the script. See the Connector Pack Integration Guide for more details on event configuration.

The script below provides a script for the USER_LOGIN_SUCCESS exit trap. The script:

1. Writes the contents of the *general*, *sessdat*, and *sesslog* sections of the input for the exit trap,
2. Searches for a ticket in the Help Desk System using the *search()* function, and
3. Runs an external program.

```
function USER_LOGIN_SUCCESS
  {
  # Loop over general section
  log( "general has " + size($general) + " items." );
  for( var $i = 0; $i < size( $general ); $i = $i + 1 )
    {
    log( keyAt( $general, $i ) + " " + valAt( $general, $i ) );
    }

  # Loop over sessdat section
  log( "sessdat has " + size($sessdat) + " items." );
  for( var $i = 0; $i < size( $sessdat ); $i = $i + 1 )
    {
    log( keyAt( $sessdat, $i ) + " " + valAt( $sessdat, $i ) );
    }

  # Loop over sesslog section
  log( "sesslog has " + size($sesslog) + " entries." );
  for( var $i = 0; $i < size( $sesslog ); $i = $i + 1 )
    {
    # each sesslog entry is an associative array
    var $entry = $sesslog[$i];
    log( "Entry " + $i + " has " + size( $entry ) + " items." );
    for( var $i = 0; $i < size( $entry ); $i = $i + 1 )
      {
```



```
    log( keyAt( $entry, $i ) + " " + valAt( $entry, $i ) )
    }
  }

log( "Now trying a search...." );
var $searchterms;
$searchterms["foo"] = "bar";
var $ticketid;
var $errmsg;
if( search( $searchterms, $ticketid, $errmsg ) == 0 )
  {
  log( "search failed, error: " + $errmsg );
  }

var $retval;

# print out some predefined variables
log( "trapfile " + $trapfile );
log( "logfile " + $logfile );

log( "Running a process..." );
var $args[];
$args[0] = "hello";
$args[1] = "Two words";

var $stdout_data;
var $returnVal;

$retval = system( "c:\\Program Files\\asdf.exe", $args,
                  "This is my input data", $stdout_data, $returnVal );
if( $retval == 0 )
  {
  # success!
  log( "Ran with success!  output [" + $stdout_data + "]" );
  log( "Return value: " + $returnVal );
  }
else
  {
  log( "Run failed with code " + $retval );
  }

return 0;
}
```


Core Library Function Reference 6

The following functions are part of the standard PSLANG function library and may be used by any PSLANG program.

6.1 String Functions

The string functions perform various operations on strings.

6.1.1 strlen

Platform: all
Details
Calculates the length of a string.

```
int strlen (
        string str
)
```

Parameters
str
[in] The string.

Return values
The length of str.

6.1.2 strtol

Platform: all
Details
Converts a string to an integer.

```
int strtol (
        string str ,
        int error ,
        int base
```

)

Parameters
str
[in] The string.

error
[out] Set to zero if the full string was converted to an integer, one if partially converted, and 2 if none of the string converted.

base
[in] The base that the string is in. Most common bases are decimal (10) and hexadecimal (16). Bases between 2 and 36 (inclusive) are allowed.

Return values
Returns the converted integer, or zero if no conversion was possible.

6.1.3 strtod

Platform: all
Details
Converts a string to a floating-point number.

```
int strtod(
        string str,
        int error
)
```

Parameters
str
[in] The string.

error
[out] Set to zero if the full string was converted to an floating-point number, one if partially converted, and 2 if none of the string was converted.

Return values
Returns the converted number, or zero if no conversion was possible.

6.1.4 strchr

Platform: all
Details
Locates a character in a string.

```
string strchr(
        string str,
        string search
```


)

Parameters
str
[in] The string.

search
[in] A one-character string to search for.

Return values
Returns the substring of str starting with the first occurrence of search in str, or returns a blank string if that character is not in str.

6.1.5 strrchr

Platform: all
Details
Locates a character in a string, starting from the end.

```
string strrchr (
        string str ,
        string search
)
```

Parameters
str
[in] The string.

search
[in] A one-character string to search for.

Return values
Returns the substring of str starting with the last occurrence of search in str, or returns a blank string if that character is not in str.

6.1.6 strcasecmp

Platform: all
Details
Compares two strings, ignoring case.

```
int strcasecmp (
        string s1 ,
        string s2
)
```

Parameters
s1
[in] String one.

s2
[in] String two.

Return values
Returns an integer less than, equal to, or greater than zero if s1 is found, respectively, to be less than, to match, or be greater than s2.

6.1.7 strncasecmp

Platform: all
Details
Compares the first n characters of two strings, ignoring case.

```
int strncasecmp(
        string s1,
        string s2,
        int n
)
```

Parameters
s1
[in] String one.

s2
[in] String two.

n
[in] Number of characters to compare.

Return values
Returns an integer less than, equal to, or greater than zero if the first n characters of s1 are found, respectively, to be less than, to match, or be greater than the first n characters s2.

6.1.8 strcmp

Platform: all
Details
Compares two strings.

```
int strcmp(
        string s1,
        string s2
)
```

Parameters
s1
[in] String one.

s2

[in] String two.

Return values
Returns an integer less than, equal to, or greater than zero if s1 is found, respectively, to be less than, to match, or be greater than s2.

6.1.9 strncmp

Platform: all
Details
Compares the the first n characters of two strings.

```
int strncmp(
        string s1,
        string s2,
        int n
)
```

Parameters
s1
[in] String one.

s2
[in] String two.

n
[in] Number of characters to compare.

Return values
Returns an integer less than, equal to, or greater than zero if the first n characters of s1 are found, respectively, to be less than, to match, or be greater than the first n characters s2.

6.1.10 strpbrk

Platform: all
Details
Searches a string for any of a set of characters.

```
string strpbrk(
        string str,
        string accept
)
```

Parameters
str
[in] The string.

accept
[in] Set of characters to find.

Return values
Returns the substring of str starting with any of the characters in accept. If none of the characters in accept are found in str, then a blank string is returned.

6.1.11 formatnum

Platform: all
Details
Transform a float value to a string with specified format.

```
string formatnum (
        string formatstr ,
        float value
)
```

Parameters
formatstr
[in] The format to transform the float to the string.

value
[in] The float for transformation.

Return values
Returns the string for the float.

6.1.12 proper

Platform: all
Details
Uppercases first roman letters and lowercases the rest in each word in the string.

```
string proper (
        string formatstr
)
```

Parameters
formatstr
[in] The string to process.

Return values
Returns the string after processing.

6.1.13 strstr

Platform: all
Details
Locates a substring within a string.


```
string strstr (
        string haystack ,
        string needle
)
```

Parameters
haystack
[in] The string to search.

needle
[in] The substring to find.

Return values
Returns the substring of haystack starting with needle.

6.1.14 strcontains

Platform: all
Details
The strcontains function determines whether a substring is located within a string.

```
int strcontains (
        string haystack ,
        string needle ,
        int casesensitive
)
```

Parameters
haystack
[in] The string to search.

needle
[in] The substring to find.

casesensitive
[in] If equal to 0, searches will be done case-insensitively, otherwise, searches will be done case-sensitively. This argument can be omitted, if so, searching will be done case-sensitively.

Return values
Whether the substring was found. Set to non-zero if it was, 0 if it was not.Non-zero if the substring is found, 0 if not

6.1.15 strspn

Platform: all
Details
Finds the number of initial characters consisting entirely of a set of characters.


```
int strspn(
        string str,
        string accept
)
```

Parameters
str
[in] The string.

accept
[in] The set of characters to accept.

Return values
Returns the length of the initial segment of str which consists entirely of characters in accept.

6.1.16 strcspn

Platform: all
Details
Finds the number of initial characters consisting entirely not of a set of characters.

```
int strcspn(
        string str,
        string reject
)
```

Parameters
str
[in] The string.

reject
[in] The set of characters to reject.

Return values
Returns the length of the initial segment of str which consists entirely of characters not in reject.

6.1.17 substr

Platform: all
Details
Duplicates a substring of a string.

```
string substr(
        string str,
        int startPos,
        int endPos
)
```

Parameters

str
[in] The string.

startPos
[in] The starting position from which to copy. The first possible position in a string is zero.

endPos
[in] The ending position from which to copy, inclusive. The last possible value for this is strlen($str)-1.

Return values
Returns the substring of str that is in the range [startPos, endPos] (inclusive).

6.1.18 trim

Platform: all
Summary: Minimize a string.
Details
Removes the given set of characters from the given part of the string until a character not in the set is found.

```
string trim(
        string source,
        string removeList,
        int removeWhere
)
```

Parameters
source
[in] The string to process.

removeList
[in] A string of characters containing all the characters to remove. If this is empty, whitespace is assumed (space, tab, '\r', '\n').

removeWhere
[in] Where to remove the characters from:
-1 = start of string forwards
0 = Both the start and end (default)
1 = end of the string backwards

Return values
Returns the string after processing.

6.1.19 chr

Platform: all
Details
Creates a one-character string from the given ASCII code.


```
string chr(
        int ascii
)
```

Parameters
ascii
[in] An ascii code (from 0-65535 inclusive) for one character.

Return values
Returns the one-character string formed from ascii.

6.1.20 ord

Platform: all
Summary: Returns the ordinal value (ASCII index) integer of the character.
Details
Expects a one-character string to determine the given ASCII/ANSI code (32-65535).

```
int ord(
        string char
)
```

Parameters
char
[in] An ASCII character (from 32 to 65535 inclusive).

Return values
Returns the integral value of the character in the ASCII table.

6.1.21 isalnum

Platform: all
Details
Checks whether or not all of the characters in a string match are alphanumeric characters.

```
int isalnum(
        string str
)
```

Parameters
str
[in] The string.

Return values
Returns 1 if all of the characters in str are alphabetic characters or numeric characters. Otherwise returns zero.

6.1.22 isalpha

Platform: all
Details
Checks whether or not all of the characters in a string match are alphabetic characters.

```
int isalpha(
        string str
)
```

Parameters
str
[in] The string.

Return values
Returns 1 if all of the characters in str are alphabetic characters. Otherwise returns zero.

6.1.23 iscntrl

Platform: all
Details
Checks whether or not all of the characters in a string match are control characters.

```
int iscntrl(
        string str
)
```

Parameters
str
[in] The string.

Return values
Returns 1 if all of the characters in str are control characters. Otherwise returns zero.

6.1.24 isdigit

Platform: all
Details
Checks whether or not all of the characters in a string match are numeric characters.

```
int isdigit(
        string str
)
```

Parameters
str
[in] The string.

Return values
Returns 1 if all of the characters in str are numeric characters (0 through 9). Otherwise returns zero.

6.1.25 isgraph

Platform: all
Details
Checks whether or not all of the characters in a string match are printable characters (except for space).

```
int isgraph(
        string str
)
```

Parameters
str
[in] The string.

Return values
Returns 1 if all of the characters in str are printable characters other than space. Otherwise returns zero.

6.1.26 islower

Platform: all
Details
Checks whether or not all of the characters in a string match are lowercase characters.

```
int islower(
        string str
)
```

Parameters
str
[in] The string.

Return values
Returns 1 if all of the characters in str are lowercase characters. Otherwise returns zero.

6.1.27 isprint

Platform: all
Details
Checks whether or not all of the characters in a string match are printable characters.

```
int isprint(
        string str
```


)

Parameters
str
[in] The string.

Return values
Returns 1 if all of the characters in str are printable characters. Otherwise returns zero.

6.1.28 ispunct

Platform: all
Details
Checks whether or not all of the characters in a string match are printable, non-alphanumeric, non-space characters.

```
int ispunct(
        string str
)
```

Parameters
str
[in] The string.

Return values
Returns 1 if all of the characters in str are punctuation characters (a punctuation character is any printable character that is neither a space nor an alphanumeric character). Otherwise returns 0.

6.1.29 isspace

Platform: all
Details
Checks whether or not all of the characters in a string match are whitespace characters.

```
int isspace(
        string str
)
```

Parameters
str
[in] The string.

Return values
Returns 1 if all of the characters in str are whitespace characters (space, form feed, newline, carriage return, horizontal tab, and vertical tab). Otherwise returns 0.

6.1.30 isupper

Platform: all
Details
Checks whether or not all of the characters in a string match are uppercase characters.

```
int isupper (
        string str
)
```

Parameters
str
[in] The string.

Return values
Returns 1 if all of the characters in str are uppercase characters. Otherwise returns 0.

6.1.31 isxdigit

Platform: all
Details
Checks whether or not all of the characters in a string match are hexadecimal digits.

```
int isxdigit (
        string str
)
```

Parameters
str
[in] The string.

Return values
Returns 1 if all of the characters in str are hexadecimal characters (0 through 9, a through f, and A through F). Otherwise returns zero.

6.1.32 tolower

Platform: all
Details
Makes a new copy of a string, converting all uppercase characters to lowercase.

```
string tolower (
        string str
)
```

Parameters
str
[in] The string.

Return values
Returns a lowercase copy of str.

6.1.33 toupper

Platform: all
Details
Makes a new copy of a string, converting all lowercase characters to uppercase.

```
string toupper (
        string str
)
```

Parameters
str
[in] The string.

Return values
Returns an uppercase copy of str.

6.1.34 replace

Platform: all
Details
Takes source, search, and replacement strings, and returns the source string with all occurrences of the search string replaced by the replacement. Returns source if the search string is empty.

```
string replace (
        string source ,
        string search ,
        string replacement
)
```

Parameters
source
[in] Source string.

search
[in] String to search for.

replacement
[in] String to use as replacement.

Return values
Returns source string with replacements performed.

6.1.35 randstr

Platform: all
Details
The randstr function generates a specified length of random string using the specified character sets.

```
string randstr (
        int length ,
        string chars
)
```

Parameters
length
[in] Length of the generated string.

chars
[in] Set of characters.

Return values
Returns generated random string.

6.1.36 replaceVars

Platform: all
Summary: Create a new string based on variable inputs.
Details
The replaceVars() function takes an array of required and optional elements and an associative array of replacement values. It then attempts to replace those source elements with the values to generate a final string.

Variables in the input array must be surrounded by '%'. eg. %username%. All other text is treated as literal.

If a literal % is desired, use a double percent '%%'. If a literal ! is desired, use '%!' Optional elements are marked by starting the string with a '!'.

Any number of variables can be in each line. Each line may be required or optional in the final output. If there are variables that are not replaced in a line and the line is optional, then that entire line is *NOT* added to the output.

```
array replaceVars (
        array source ,
        array vars
)
```

Parameters
source
[const] Array of string components with variables to be replaced and concatenated to form the output string.

vars

[const] An associative array of variable names and values.

Return values
Returns resultant string with known variables replaced.Note: input elements will be added to the output string in the same order the are in the input array.
Note: input elements which are marked as required, but are not replaced will be added literally (including the variable).

6.1.37 regex_replace

Platform: win32
Summary: Replaces matches to the given regular expression with the given replacement string.
Details
Data must have Linux line endings (only '\n') to use '$'.

```
int regex_replace (
        string str ,
        string regexp ,
        string replacement ,
        int insensitive ,
        string result ,
        string error
)
```

Parameters
str
[in] The string to examine.

regexp
[in] The regular expression to use.

replacement
[in] The replacement string for matches.

insensitive
[in] If non-zero, regexp performs a case-insensitive comparison.

result
[out] A string containing all replaced matches.

error
[out] Contains a message when an error occurs.

Return values
Returns 1 if regular expression was correctly parsed and executed. Returns 0 otherwise. You must compare the original and new string to determine if anything was actually replaced.

6.1.38 regex_split

Platform: win32
Details
Breaks a string up using a regular expression to locate the delimiters of interest.

```
int regex_split (
        string str ,
        string regexp ,
        int insensitive ,
        array matches ,
        string error
)
```

Parameters
str
[in] The string to examine.

regexp
[in] The regular expression to use as a delimiter.

insensitive
[in] If non-zero, regexp performs a case-insensitive comparison.

matches
[out] An array of matches containing matches to the given regular expression.

error
[out] Contains a message when an error occurs.

Return values
Returns 1 if regular expression was correctly parsed and executed. Returns 0 otherwise. You must check the size of the result array to determine if the string was split.

6.1.39 regex_search

Platform: win32
Details
Searches a string for one or more matches to the given regular expression. All matches contained within () brackets will be returned through the output array.

```
int regex_search (
        string str ,
        string regexp ,
        int insensitive ,
        array matches ,
        string error
)
```

Parameters

str
[in] The string to examine.

regexp
[in] The regular expression to use.

insensitive
[in] If non-zero, regexp performs a case-insensitive comparison.

matches
[out] An array of strings containing the matches found by the function. Element 0 contains matches for the entire regular expression, and the following elements are matches for each parenthetized subexpression in the main regular expression.

error
[out] Contains a message when an error occurs.

Return values
Returns 1 if the regular expression was correctly parsed and executed. Returns 0 otherwise. You must check the size of the result array to determine if any matches were made.

6.1.40 match

Platform: win32
Details
Test whether a string matches a given regular expression.

```
int match(
        string str,
        string regexp,
        int insensitive
)
```

Parameters
str
[in] The string to examine.

regexp
[in] The regular expression to use.

insensitive
[in] If non-zero, regexp performs a case-insensitive comparison.

Return values
Returns 1 if a match was made. Returns 0 if no matches were made, the regular expression isn't valid, or some other unknown error occurred.

6.1.41 regex_match

Platform: win32
Details
Searches a string to see if it precisely matches the given regular expression. All sub-matches contained within () brackets will be returned through the output array.

```
int regex_match (
        string str ,
        string regexp ,
        int insensitive ,
        array matches ,
        string error
)
```

Parameters
str
[in] The string to examine.

regexp
[in] The regular expression to use.

insensitive
[in] If non-zero, regexp performs a case-insensitive comparison.

matches
[out] An array of strings containing the matches found by the function. Element 0 contains matches for the entire regular expression, and the following elements are matches for each parenthetized subexpression in the main regular expression.

error
[out] Contains a message when an error occurs.

Return values
Returns 1 if the regular expression was correctly parsed and executed. Returns 0 otherwise. You must check the size of the result array to determine if any matches were made.

6.2 Input/Output Functions

The Input/Output functions provide mechanisms for reading from and writing to input/output devices such as files, TCP/IP sockets, and pipes.

6.2.1 File I/O Modes

The `fopen` function requires a *mode* parameter. The values accepted for this parameter are identical to the values allowed for the equivalent C library function:

Mode	Description
`r`	Open text file for reading. The stream is positioned at the beginning of the file.
`r+`	Open for reading and writing. The stream is positioned at the beginning of the file.
`w`	Truncate file to zero length or create text file for writing. The stream is positioned at the beginning of the file.
`w+`	Open for reading and writing. The file is created if it does not exist, otherwise it is truncated. The stream is positioned at the beginning of the file.
`a`	Open for writing. The file is created if it does not exist. The stream is positioned at the end of the file.
`a+`	Open for reading and writing. The file is created if it does not exist. The stream is positioned at the end of the file.

Additionally, the *mode* string may include the letter 'b' as a a last character or as a character between any of the two-character strings above. The 'b' is ignored on Unix systems. On Windows, the 'b' means to open in binary ("untranslated") mode. So, if 'b' is **not** specified, Windows opens a file in *text* mode, meaning that carriage return-linefeed combinations are translated into linefeed characters on input, and linefeed characters are translated into carriage return-linefeed combinations on output.

6.2.2 I/O Return Values

Most I/O functions return one of the following integer values:

Value	Description
0	Success
1	Not connected
2	Timeout
3	Disconnected (after a successful connect)
4	Generic error
5	End of file
6	Invalid I/O handle passed into function

It is important to check the return values of the open-style functions (*fopen*, *popen*, and *connect*) to ensure that the opened handle is valid before attempting to read from it or write to it.

6.2.3 Writetelnet options

This option specifies the format of the buffer in writetelnet command. It supports the following formats:

Value	Description
0	Absolutely writing. For every entry in the buffer, row and column must be specified. The format of an entry can be "row,column,width,string".
1	Relatively writing. PSLANG parses the last screen and find out all the writable positions. The writing will start from the first writable position. In this case, the format of an entry can be "width,string".
2	Matching and relatively writing. PSLANG will not only parse the last screen, but also try to match a pattern and find the first writable position in this line. The writing will start from this position. The format of entry is the same as the one for relatively writing.
3	Unformatted writing. If the last screen is unformatted, an unformatted reply is expected. For an unformatted reply, only one entry is allowed and must be written in position (1,1). So, the format of such an entry is "width,string".

6.2.4 SSL Peer Authentication

The *connectSSL()* function connects to a TCP/IP socket using Secure Sockets Layer (SSL). A successful SSL connection, by itself, does not authenticate the peer (the server). The *authenticatePeer()* function should be used to ensure that the server is properly authenticated. The *authenticatePeer()* function returns one of the following values:

Value	Description
0	ok
2	unable to get issuer certificate
3	unable to get certificate
4	unable to decrypt certificate's signature
5	unable to decrypt CRL's signature
6	unable to decode issuer key
7	certificate signature failure
8	CRL signature failure
9	certificate is not yet valid
10	certificate has expired
11	CRL is not yet valid

12	CRL has expired
13	format error in certificate's notBefore field
14	format error in certificate's notAfter field
15	format error in CRL's lastUpdate field
16	format error in CRL's nextUpdate field
17	out of memory
18	self-signed certificate
19	self-signed certificate in certificate chain
20	unable to get local issuer certificate
21	unable to verify the first certificate
22	certificate chain too long
23	certificate revoked
24	invalid CA certificate
25	path length constraint exceeded
26	unsupported certificate purpose
27	certificate not trusted
28	certificate rejected
29	subject issuer mismatch
30	authority and subject key identifier mismatch
31	authority and issuer serial number mismatch
32	key usage does not include certificate signing
50	application verification failure

In particular, error code 50 is returned if the hostname given in the *connectSSL()* call does not match the hostname given in the certificate presented by the server.

6.2.5 read

Platform: all
Details
Reads a string from an I/O handle.

```
int read(
        handle handle,
        string buffer,
        int length,
        int timeout
)
```

Parameters
handle
[in] The I/O handle to read from.

buffer
[out] The buffer in which to place the results. Note that some characters may still be in buffer even if a code other than IO_OK is returned, particularly if IO_EOF is returned.

length
[in] The number of characters to read.

timeout
[in] Timeout in seconds. A timeout of zero means to wait until the specified number of characters have been read. On Windows, timeout values only work on sockets.

Return values
See I/O Return Values for a list of possible return values.

6.2.6 readline

Platform: all
Details
Reads a line of input.

```
int readline (
        handle handle ,
        string buffer ,
        int length ,
        int timeout
)
```

Parameters
handle
[in] An I/O handle to read from.

buffer
[out] A buffer in which to place the results. The newline characters are thrown away, i.e., they are not put into buffer. Note that some characters may still be in buffer even if a code other than IO_OK is returned, particularly if IO_EOF is returned.

length
[in] The maximum number of characters to read, including the newline character(s).

timeout
[in] Timeout in seconds. A timeout of zero means to wait indefinitely until either the maximum characters have been read or until the end of the line has been read. On Windows, timeout values only work on sockets.

Return values
See I/O Return Values for a list of possible return values.

6.2.7 write

Platform: all
Details
Writes a string to an I/O handle.

```
int write (
        handle handle ,
        string buffer ,
        int timeout
)
```

Parameters
handle
[in] The I/O handle to write to.

buffer
[in] The buffer to write.

timeout
[in] Timeout in seconds. A timeout of zero means to wait until the specified number of characters have been written. On Windows, timeout values only work on sockets.

Return values
See I/O Return Values for a list of possible return values.

6.2.8 setCodepage

Platform: all
Details
Sets the codepage for the data destination of an I/O handle.

```
int setCodepage (
        handle handle ,
        int cp
)
```

Parameters
handle
[in] The I/O handle to write to.

cp
[in] Windows code page number. https://msdn.microsoft.com/en-us/library/windows/desktop/dd317756(v=vs.85).aspx.

Return values
See I/O Return Values for a list of possible return values.

6.2.9 errno

Platform: all
Details
Retrieves the last error number.

```
int errno (
        handle handle
)
```

Parameters
handle
[in] The I/O handle to examine for errors.

Return values
Returns the system-level error number from the last I/O call. This number may provide extra information. It may be zero even if the last I/O call failed.

6.2.10 strerror

Platform: all
Details
Retrieves a string describing the last error.

```
string strerror (
        handle handle
)
```

Parameters
handle
[in] The I/O handle to examine for errors.

Return values
Returns a string describing the error from the last I/O call. A blank string is returned if the last call is successful.

6.2.11 fopen

Platform: all
Details
Opens a file for reading, writing, or both.

```
int fopen (
        handle handle ,
        string filename ,
        string mode
```


)

Parameters
handle
[out] The opened I/O handle. You must call close() on this handle when finished, even if fopen() returns an error code.

filename
[in] The file name to open.

mode
[in] The mode in which to open the file. See File I/O Modes for more details.

Return values
See I/O Return Values for a list of possible return values.

6.2.12 popen

Platform: all
Details
Runs a command, writing to its standard input or reading from its standard output (but not both). When a handle opened with popen is closed, the return value of close() is set to the exit code of the executed program, or -1 if an error occurs. For UNIX, all popen-ed pipes are bi-directional and can be used with write()+expect() to interface with an interactive prompt. For Windows, popen() is not bi-directional and therefore cannot be used with expect().

```
int popen (
        handle handle ,
        string command ,
        string mode
)
```

Parameters
handle
[out] The opened I/O handle. You must call close() on this handle when finished, even if popen() returns an error code.

command
[in] The command line to execute as it would be entered for a command prompt.

mode
[in] Specifies whether or not the pipe is for reading (“r”) or writing (“w”). As discussed in File I/O Modes, “b” may be added to the mode.

Return values
See I/O Return Values for a list of possible return values.

6.2.13 close

Platform: all
Details
Closes an I/O handle and frees any associated system resources.

```
int close (
        handle handle
)
```

Parameters
handle
[in/out] The I/O handle to close. Works on any type of handle.

Return values
For handles opened with popen, the return value is set to the exit code of the executed program, or -1 if an error occurs. For all other handle types, zero is returned.

6.2.14 print

Platform: all
Details
Prints a string to the script's $stdout (standard out) stream, without a newline. If the script does not have a $stdout variable, the string is printed to the interpreter's standard out.

```
int print (
        string str
)
```

Parameters
str
[in] The string to print.

Return values
See I/O Return Values for a list of possible return values.

6.2.15 writeBOM

Platform: all
Details
Adds the byte order mark (BOM). Use this only at the beginning of the file.

You cannot call writeBOM after calling readBOM. Furthermore, writeBOM must only be used on file pointers that point at the start of a file.

```
int writeBOM (
        handle handle ,
        int bom
```


)

Parameters
handle
[in] The file handle opened in w mode

bom
[in] The BOM you wish to write. 1 => utf8

Return values
See I/O Return Values for a list of possible return values.I/O Code

6.2.16 readBOM

Platform: all
Details
Reads the byte order mark (BOM) from a file. The BOM type found is returned with the File handle's current position is set to the first byte after the BOM. If the BOM is not found, the File handle's current position is set to the start of the file.

If the File handle does not point to the beginning of the file, an error is returned and the BOM type is set to unknown; that is, 0.

You cannot call writeBOM after a readBOM call.

```
int readBOM(
        handle handle ,
        int bom
)
```

Parameters
handle
[in] The file handle opened in read mode.

bom
[out] The BOM in the file. 0 => unknown, 1 => utf8, 2 => utf16le, 3 => utf16be, 4 => utf32le, 5 => utf32be.

Return values
See I/O Return Values for a list of possible return values.I/O Code

6.2.17 println

Platform: all
Details
Prints a string to the script's $stdout (standard out) stream, with a newline. If the script does not have a $stdout variable, the string is printed to the interpreter's standard out.

```
int println(
        string str
```


)

Parameters
str
[in] The string to print.

Return values
See I/O Return Values for a list of possible return values.

6.2.18 csvParseInternalHeaders

Platform: all
Details
Reads and parses a CSV (Comma-Separated Values) document, assuming the first row in the document is the header row. Blank lines are ignored.

```
int csvParseInternalHeaders (
        handle handle ,
        array headers ,
        array data ,
        int timeout
)
```

Parameters
handle
[in] An I/O handle from which to read a CSV document.

headers
[out] An array of strings containing the column headers.

data
[out] An array where each element represents a row. Each row is set to an associative array where each element can be accessed by the header title. Note that if more than one column has the same title, it will not be possible to access all the row elements by title. However, one may still access the array elements by index. If there are fewer columns in a row than in the header, the remaining columns will be filled in with blank strings. If there are more columns in a row than in the header, the extra columns are ignored.

timeout
[in] Timeout (seconds).

Return values
An I/O return code. IO_EOF will be returned if the last line in the document does not end with a newline, otherwise IO_OK will be returned to indicate that all lines in the document were read successfully. See I/O Return Values for a list of possible return values.

6.2.19 csvParseExternalHeaders

Platform: all
Details
Reads and parses a CSV (Comma-Separated Values) document, using the provided headers as the header row, and assuming the document has no header row. Blank lines are ignored.

```
int csvParseExternalHeaders (
        handle handle ,
        array headers ,
        array data ,
        int timeout
)
```

Parameters
handle
[in] An I/O handle from which to read a CSV document.

headers
[const] The headers to use when parsing the CSV document.

data
[out] An array where each element represents a row. Each row is set to an associative array where each element can be accessed by the header title. Note that if more than one column has the same title, it will not be possible to access all the row elements by title. However, one may still access the array elements by index. If there are fewer columns in a row than in the header, the remaining columns will be filled in with blank strings. If there are more columns in a row than in the header, the extra columns are ignored.

timeout
[in] Timeout (seconds).

Return values
An I/O return code. IO_EOF will be returned if the last line in the document does not end with a newline, otherwise IO_OK will be returned to indicate that all lines in the document were read successfully. See I/O Return Values for a list of possible return values.

6.2.20 csvParse

Platform: all
Details
Reads and parses a CSV (Comma-Separated Values) document, assuming the document has no header row. Blank lines are ignored.

```
int csvParse (
        handle handle ,
        array data ,
        int timeout
)
```

Parameters

handle
[in] An I/O handle from which to read a CSV document.

data
[out] An array where each element represents a row. Each row is a regular array of strings. Note that each row may have a different number of columns, depending on the data in the CSV document.

timeout
[in] Timeout (seconds).

Return values
An I/O return code. IO_EOF will be returned if the last line in the document does not end with a newline, otherwise IO_OK will be returned to indicate that all lines in the document were read successfully. See I/O Return Values for a list of possible return values.

6.2.21 remove

Platform: all
Details
Deletes a file.

```
string remove (
        string filename
)
```

Parameters
filename
[in] The name of the file to delete.

Return values
Returns an empty string on success or an error message on failure.

6.2.22 read_match

Platform: win32
Details
read_match description

```
int read_match (
        handle handle ,
        array regexps ,
        int insensitive ,
        array matches ,
        string output ,
        string error ,
        int timeout
```


)

Parameters
handle
[in] The I/O handle to read from.

regexps
[in] regexps

insensitive
[in] If non-zero, regexp performs a case-insensitive comparison.

matches
[out] An array of strings containing the matches found by the function. Element 0 contains matches for the entire regular expression, and the following elements are matches for each parenthetized subexpression in the main regular expression.

output
[out] output

error
[out] Contains a message when an error occurs.

timeout
[in] timeout

Return values
Returns non-negative integer values for the matching regex index. -1 timeout, -2 invalid handle, -3 end of file, -4 invalid regex, -5 unknown error.

6.2.23 readline_match

Platform: win32
Details
readline_match description

```
int readline_match(
        handle handle,
        array regexps,
        int insensitive,
        array matches,
        string output,
        string error,
        int timeout
)
```

Parameters
handle
[in] The I/O handle to read from.

regexps
[in] regexps

insensitive
[in] If non-zero, regexp performs a case-insensitive comparison.

matches
[out] An array of strings containing the matches found by the function. Element 0 contains matches for the entire regular expression, and the following elements are matches for each parenthetized subexpression in the main regular expression.

output
[out] output

error
[out] Contains a message when an error occurs.

timeout
[in] timeout

Return values
Returns non-negative integer values for the matching regex index. -1 timeout, -2 invalid handle, -3 end of file, -4 invalid regex, -5 unknown error.

6.3 HTTP Functions

The HTTP functions provide mechanisms for communicating with a web server using HTTP and HTTPS. These functions are only available in Windows versions of PSLANG.

> **Note:** These functions use the WinInet library supplied with Windows; as such they obey all the internet settings in Windows, including those related to SSL certificates. Other functions, such as *connectSSL()*, use the OpenSSL library.

6.3.1 httpOpen

Platform: all
Details
Creates an HTTP(S) handle for use with the http functions. You must call close() on this handle when finished. Multiple HTTP(S) requests may be made with the returned handle before closing it. Cookies are handled automatically, connections and other settings are inherited from Windows Internet Explorer.


```
int httpOpen(
        handle handle,
        string host,
        int port,
        int ssl,
        int useAuth,
        string userid,
        string password
)
```

Parameters
handle
[out] The opened I/O handle. You must call close() on this handle when finished, even if httpOpen() returns an error code.

host
[const] The hostname or IP address to which to connect.

port
[const] The port to which to connect. The standard HTTP port is 80 and the standard HTTPS port is 443.

ssl
[const] Whether or not to use HTTPS (SSL). Set to non-zero when HTTPS is to be used.

useAuth
[const] Whether or not to use the userid and password below for HTTP authentication schemes.

userid
[const] The userid to present for authentication if requested by the server.

password
[const] The password to present for authentication if requested by the server.

Return values
See I/O Return Values for a list of possible return values.

6.3.2 httpIgnoreCertificate

Platform: all
Details
Sets subsequent HTTPS operations to ignore problems with the server's SSL certificate. Ignore certificate verification is not recommended for production systems due to the security implications. (This function is available on Win32-only).

```
void httpIgnoreCertificate(
        handle handle
)
```

Parameters
handle

[in/out] A handle opened with httpOpen().

Return values
None.

6.3.3 httpSetClientCertificate

Platform: all
Details
Sets the client certificate to provide to the server for negotiation of encryption on subsequent HTTP requests. (This function is available on Win32-only).

```
int httpSetClientCertificate (
        handle handle ,
        int provider ,
        string storage ,
        int findType ,
        string content
)
```

Parameters
handle
[in/out] A handle opened with httpOpen().

provider
[const] 1 – file, 2 – current user's system storage, 3 – local machine's system storage.

storage
[const] The certificate storage, can be 'MY','Root','CA' or the certificate file name.

findType
[const] 1 – issuer, 2 – subject, 3 – key identifier, 4 – SHA1 hash, 5 – MD5 hash, 6 – signature hash, 7 – MD5-hashed public key

content
[const] The content for searching, if it is empty the first certificate in storage will be used.

Return values
If successfully found a certificate zero is returned, otherwise non-zero is returned.

6.3.4 httpGet

Platform: all
Details
Starts an HTTP(S) GET operation. The resulting response, if any, may be read with either the read() and readLine() functions or the expect statement.


```
int httpGet(
        handle handle,
        string resource,
        array requestHeaders,
        int timeout,
        int statusCode,
        array responseHeaders
)
```

Parameters
handle
[in/out] A handle opened with httpOpen().

resource
[const] The path and optional query string to request, in the form "/directory/file?querystring". The path must be absolute (that is, start with /).

requestHeaders
[const] Additional headers to send to the server.

timeout
[const] Timeout in seconds, for each of the connect, writing, and subsequent reading operations on the returned handle.

statusCode
[out] Status code returned by the server (e.g., 200). If the operation fails before reading a status code, statusCode is set to 0.

responseHeaders
[out] Headers returned by the server.

Return values
See I/O Return Values for a list of possible return values.

6.3.5 httpPost

Platform: all
Details
Starts an HTTP(S) POST operation. The resulting response, if any, may be read with either the read() function or the expect statement.

```
int httpPost(
        handle handle,
        string resource,
        array requestHeaders,
        string contentType,
        string requestBody,
        int timeout,
        int statusCode,
        array responseHeaders
```


)

Parameters
handle
[in/out] A handle opened with httpOpen().

resource
[const] The path and optional query string to request, in the form "/directory/file?querystring". The path must be absolute (that is, start with /).

requestHeaders
[const] Additional headers to send to the server. The content length is determined automatically from requestBody.

contentType
[const] The content type header to send to the server. If this parameter is blank, the content-type "application/x-www-form-urlencoded" is sent to the server.

requestBody
[const] The body to POST.

timeout
[const] Timeout in seconds, for each of the connect, writing, and subsequent reading operations on the returned handle.

statusCode
[out] Status code returned by the server (e.g., 200). If the operation fails before reading a status code, statusCode is set to 0.

responseHeaders
[out] Headers returned by the server.

Return values
See I/O Return Values for a list of possible return values.

6.3.6 httpUrlEncode

Platform: all
Details
Encodes a name=value pair for use in a URL's query string or for the body of a POST request. Note that a query string must start with a question mark (?) and that name=value pairs must be separated by &.

```
string httpUrlEncode (
        string name ,
        string value
)
```

Parameters
name
[const] The name part of a name=value pair.

value
[const] The value part of a name=value pair.

Return values
A properly encoded name=value pair.

6.3.7 Troubleshooting

6.3.7.1 httpOpen

When using SSL, the httpOpen function may report an error when contacting a server that is not trusted.

To avoid this error you can either add the httpIgnoreCertificate function to ignore the certificate, or you can install the certificate in the Trusted Root Certification Authorities store on the Bravura Security Fabric server.

> **Note:** Ignoring certificate verification is not recommended for production systems due to the security implications.

To install a certificate using Internet Explorer 9:

1. Launch Internet Explorer and connect to the target website (e.g. `https://myapp.example.com`).
   A Security Alert dialog displays.
2. Click **View Certificate** → **Install Certificate . . . .**
   The Certificate Import Wizard displays.
3. Click **Next**, then **Next** to accept the defaults, then **Finish**.
   The Root Certificate Store confirmation dialog displays.
4. Click **Yes** to add the certificate to the Root Store.
   Internet Explorer installs the certificate.

You will need to repeat the above process once the certificate expires.

6.3.7.2 httpGet

If an errno of 12057 is returned while trying to perform a httpGet to a HTTPS URL, the certificate revocation list is unavailable to verify if the server certificate has been revoked.

Either the CRL needs to be published from the CA and installed on the Bravura Security Fabric server, or the setting can be shut off under IE: **Internet options** (inetcpl.cpl) → **Advanced** tab → **Check for server certificate revocation**.

6.4 Time Functions

The functions in this section provide mechanisms for getting the current time and manipulating time values. See Dates for information about date-type variables.

6.4.1 time

Platform: all
Details
Returns the current time.

```
int time (

)
```

Parameters
None.

Return values
Returns the current time in seconds since 12:00 AM, January 1, 1970, UTC.

6.4.2 ctime

Platform: all
Details
Returns a formatted date/time string.

```
string ctime (
        int time
)
```

Parameters
time
[in] A time value, possibly returned from time().

Return values
Returns a formatted UTC time string in the form “Wed Jun 30 21:49:08 1993”, using 3-letter abbreviations for days and months.

6.4.3 startElapsedTime

Platform: all
Details
Returns a value which can be passed to the getElapsedTimeSec function


```
float startElapsedTime (

)
```

Parameters
None.

Return values
Returns a value to be passed into getElapsedTimeSec function.

6.4.4 getElapsedTimeSec

Platform: all
Details
Returns the number of seconds since the input value was assigned using startElapsedTime.

```
float getElapsedTimeSec (
        float null
)
```

Parameters
null
[in] Starting time.

Return values
Returns the number of seconds since the input value was assigned.

6.4.5 mktime

Platform: all
Details
Converts a broken-down representation of UTC time into the same format returned by the time() function. If values for the parameters are outside their normal ranges, the result will be normalized. For example, if mon is set to 1 and mday is set to 32 (i.e., January 32), this function will return a time that is equivalent to February 1.

```
int mktime (
        int seconds ,
        int minutes ,
        int hours ,
        int mday ,
        int mon ,
        int year ,
        int isdst
)
```

Parameters
seconds

[in] The number of seconds after the minute, from 0 to 59.

minutes
[in] The number of minutes after the hour, from 0 to 59.

hours
[in] The number of hours past midnight, from 0 to 23.

mday
[in] The day of the month, from 1 to 31.

mon
[in] The month of year, from 1 to 12.

year
[in] The full (4-digit) year.

isdst
[in] Whether daylight savings time is in effect at the time specified. Set to 0 if it is not, 1 if it is, and -1 if the information is not available.

Return values
Returns time specified by the arguments in seconds since 12:00 AM, January 1, 1970. If the provided time values cannot be represented in this way, -1 is returned.

6.4.6 seconds

Platform: all
Details
Gets the seconds portion of a time value.

```
int seconds(
        int time
)
```

Parameters
time
[in] A time value, possibly returned from time().

Return values
Returns the seconds component of time, where time is broken up into year, month, day, hours, minutes, and seconds, UTC. The value is normally in the range of 0-59, but may be set up to 61 to allow for leap seconds.

6.4.7 minutes

Platform: all
Details
The minutes function gets the minutes portion of a time value.


```
int minutes(
        int time
)
```

Parameters
time
[in] A time value, possibly returned from time().

Return values
Returns the minutes component of time, where time is broken up into year, month, day, hours, minutes, and seconds, UTC. The value is in the range of 0-59.

6.4.8 hours

Platform: all
Details
Gets the hours portion of a time value.

```
int hours(
        int time
)
```

Parameters
time
[in] A time value, possibly returned from time().

Return values
Returns the hours component of time, where time is broken up into year, month, day, hours, minutes, and seconds, UTC. The value is the number of hours past midnight, in the range of 0-23.

6.4.9 dayofmonth

Platform: all
Details
Gets the day of the month portion of a time value.

```
int dayofmonth(
        int time
)
```

Parameters
time
[in] A time value, possibly returned from time().

Return values
Returns the day component of time, where time is broken up into year, month, day, hours, minutes, and seconds, UTC. The value is in the range of 1-31.

6.4.10 month

Platform: all
Details
Gets the month portion of a time value.

```
int month (
        int time
)
```

Parameters
time
[in] A time value, possibly returned from time().

Return values
Returns the month component of time, where time is broken up into year, month, day, hours, minutes, and seconds, UTC. The value is in the range of 1-12, where 1 is for January, 2 is for February, etc.

6.4.11 year

Platform: all
Details
Gets the year portion of a time value.

```
int year (
        int time
)
```

Parameters
time
[in] A time value, possibly returned from time().

Return values
Returns the year component of time, where time is broken up into year, month, day, hours, minutes, and seconds, UTC. The value is the full 4-digit number for the year.

6.4.12 dayofweek

Platform: all
Details
Gets the day of the week from a time value.

```
int dayofweek (
        int time
)
```

Parameters
time

[in] A time value, possibly returned from time().

Return values
Returns a numeric value for the day of the week (UTC) in the range of 1-7, where Sunday is 1, Monday is 2, etc.

6.4.13 dayofyear

Platform: all
Details
Gets the day of the year from a time value.

```
int dayofyear (
        int time
)
```

Parameters
time
[in] A time value, possibly returned from time().

Return values
Returns the day of the year (UTC) in the range of 1-366, where January 1 is 1, December 31 is 365 in a non-leap year and 366 in a leap year.

6.4.14 isdst

Platform: all
Details
Gets daylight savings time information from a time value.

```
int isdst (
        int time
)
```

Parameters
time
[in] A time value, possibly returned from time().

Return values
Returns 1 if daylight savings was in effect during the given UTC time, 0 if not, and -1 if the information is not available.

6.4.15 sleep

Platform: all
Details
Suspends script execution for at least the specified milliseconds.


```
int sleep(
        int milliseconds
)
```

Parameters
milliseconds
[const] Milliseconds to sleep.

Return values
Returns the number of milliseconds that have elapsed since the system was started. The time will wrap around to zero every 49.7 days on a 32-bit system.

6.4.16 toDate

Platform: all
Details
Create a date value from a string.

```
int toDate(
        date dateout,
        string datestring
)
```

Parameters
dateout
[out] The converted date value.

datestring
[const] The string indicating the date value. It could be in any valid date constant format.

Return values
Returns 0 when the input string is converted to date value successfully, other values when the string is not in right date format.

6.4.17 valid

Platform: all
Details
Check if the information contained in the date variable is valid.

```
int valid(
        date dateout
)
```

Parameters
dateout
[in] The converted date value.

Return values

Returns 1 when the date is valid or 0 when it is invalid.

6.4.18 setBusinessDayType

Platform: all
Details
Sets how the weekend is to be included in the calculation of the next date.

```
void setBusinessDayType (
        date cur ,
        int val
)
```

Parameters
cur
[in/out] The date variable.

val
[const] The flag indicating how the weekend is to be included in the calculation of the next date. A value of 0 means not to check whether the next date is part of a weekend. 1 indicates that the next Monday will be used when the next date occurs during a weekend. -1 indicates that the previous Friday will be used when the next date occurs during the weekend.

Return values
None.

6.4.19 getBusinessDayType

Platform: all
Details
Gets how the weekend is to be included in the calculation of the next date.

```
int getBusinessDayType (
        date cur
)
```

Parameters
cur
[in] The date variable.

Return values
Returns the flag for indicating how the weekend influences the calculation of next occurrence date. 0 means not to check whether the next date lies in weeked. 1 indicates that the next Monday will be used when the next date is weekend. -1 indicates that the previous Friday will be used when the next date is weekend.

6.4.20 getNextDate

Platform: all
Details
Get the next occurrence date.

```
date getNextDate (
        date cur
)
```

Parameters
cur
[in] The date with interval.

Return values
Returns the date after the specified interval.

6.4.21 getLdapDateString

Platform: all
Details
Get LDAP date string from the date.

```
string getLdapDateString (
        date cur
)
```

Parameters
cur
[in] The date variable.

Return values
Returns LDAP date string.

6.4.22 setUsingLdapDate

Platform: all
Details
Set date using LDAP date.

```
date setUsingLdapDate (
        float ldapDate
)
```

Parameters
ldapDate
[const] The date in LDAP date format.

Return values
Returns the date.

6.4.23 setUsingADDate

Platform: win32
Details
Set date using Active Directory date.

```
date setUsingADDate (
        string adDate
)
```

Parameters
adDate
[const] The string value for a 64-bit number of 100-nanosecond intervals since January 1, 1601 UTC, as used by Active Directory time stamps

Return values
Returns the date.

6.4.24 getYear

Platform: all
Details
Get the year value for the date variable.

```
int getYear (
        date cur
)
```

Parameters
cur
[in] The date variable.

Return values
Returns -1 if the variable does not have valid year value (of interval type date variable), otherwise return the year value.

6.4.25 setYear

Platform: all
Details
Set the year value for the date variable.


```
void setYear (
        date cur,
        int value
)
```

Parameters
cur
[in/out] The date variable.

value
[const] The value to be set.

Return values
None.

6.4.26 getMonth

Platform: all
Details
Get the month value for the date variable.

```
int getMonth (
        date cur
)
```

Parameters
cur
[in] The date variable.

Return values
Returns -1 if the variable does not have valid month value (of interval type date variable), otherwise return the month value.

6.4.27 setMonth

Platform: all
Details
Set the month value for the date variable.

```
void setMonth (
        date cur,
        int value
)
```

Parameters
cur
[in/out] The date variable.

value

[const] The value to be set.

Return values
None.

6.4.28 getDay

Platform: all
Details
Get the day value for the date variable.

```
int getDay (
        date cur
)
```

Parameters
cur
[in] The date variable.

Return values
Returns -1 if the variable does not have valid day value (of interval type date variable), otherwise return the day value.

6.4.29 setDay

Platform: all
Details
Set the day value for the date variable.

```
void setDay (
        date cur ,
        int value
)
```

Parameters
cur
[in/out] The date variable.

value
[const] The value to be set.

Return values
None.

6.4.30 getHour

Platform: all
Details
Get the hour value for the date variable.

```
int getHour (
        date cur
)
```

Parameters
cur
[in] The date variable.

Return values
Returns -1 if the variable does not have valid hour value (of interval type date variable), otherwise return the hour value.

6.4.31 setHour

Platform: all
Details
Set the hour value for the date variable.

```
void setHour (
        date cur ,
        int value
)
```

Parameters
cur
[in/out] The date variable.

value
[const] The value to be set.

Return values
None.

6.4.32 getMinute

Platform: all
Details
Get the minute value for the date variable.

```
int getMinute (
        date cur
```


)

Parameters
cur
[in] The date variable.

Return values
Returns -1 if the variable does not have valid minute value (of interval type date variable), otherwise return the minute value.

6.4.33 setMinute

Platform: all
Details
Set the minute value for the date variable.

```
void setMinute (
        date cur ,
        int value
)
```

Parameters
cur
[in/out] The date variable.

value
[const] The value to be set.

Return values
None.

6.4.34 getSecond

Platform: all
Details
Get the second value for the date variable.

```
int getSecond (
        date cur
)
```

Parameters
cur
[in] The date variable.

Return values
Returns -1 if the variable does not have valid second value (of interval type date variable), otherwise return the second value.

6.4.35 setSecond

Platform: all
Details
Set the second value for the date variable.

```
void setSecond (
        date cur ,
        int value
)
```

Parameters
cur
[in/out] The date variable.

value
[const] The value to be set.

Return values
None.

6.4.36 getIntervalYear

Platform: all
Details
Get the year interval value for the date variable.

```
int getIntervalYear (
        date cur
)
```

Parameters
cur
[in] The date with interval.

Return values
Returns -1 if the variable does not have valid year interval value (of no interval type date variable), otherwise return the year interval value.

6.4.37 setIntervalYear

Platform: all
Details
Set the year interval value for the date variable.

```
void setIntervalYear (
        date cur ,
        int value
```


)

Parameters
cur
[in/out] The date variable.

value
[const] The value to be set.

Return values
None.

6.4.38 getIntervalMonth

Platform: all
Details
Get the month interval value for the date variable.

```
int getIntervalMonth (
        date cur
)
```

Parameters
cur
[in] The date with interval.

Return values
Returns -1 if the variable does not have valid month interval value (of no interval type date variable), otherwise return the month interval value.

6.4.39 setIntervalMonth

Platform: all
Details
Set the month interval value for the date variable.

```
void setIntervalMonth (
        date cur ,
        int value
)
```

Parameters
cur
[in/out] The date variable.

value
[const] The value to be set.

Return values

None.

6.4.40 getIntervalDay

Platform: all
Details
Get the day interval value for the date variable.

```
int getIntervalDay(
        date cur
)
```

Parameters
cur
[in] The date with interval.

Return values
Returns -1 if the variable does not have valid day interval value (of no interval type date variable), otherwise return the day interval value.

6.4.41 setIntervalDay

Platform: all
Details
Set the day interval value for the date variable.

```
void setIntervalDay(
        date cur,
        int value
)
```

Parameters
cur
[in/out] The date variable.

value
[const] The value to be set.

Return values
None.

6.4.42 getIntervalHour

Platform: all
Details
Get the hour interval value for the date variable.


```
int getIntervalHour (
        date cur
)
```

Parameters
cur
[in] The date with interval.

Return values
Returns -1 if the variable does not have valid hour interval value (of no interval type date variable), otherwise return the hour interval value.

6.4.43 setIntervalHour

Platform: all
Details
Set the hour interval value for the date variable.

```
void setIntervalHour (
        date cur,
        int value
)
```

Parameters
cur
[in/out] The date variable.

value
[const] The value to be set.

Return values
None.

6.4.44 getIntervalMinute

Platform: all
Details
Get the minute interval value for the date variable.

```
int getIntervalMinute (
        date cur
)
```

Parameters
cur
[in] The date with interval.

Return values
Returns -1 if the variable does not have valid minute interval value (of no interval type date variable),

otherwise return the minute interval value.

6.4.45 setIntervalMinute

Platform: all
Details
Set the minute interval value for the date variable.

```
void setIntervalMinute(
        date cur,
        int value
)
```

Parameters
cur
[in/out] The date variable.

value
[const] The value to be set.

Return values
None.

6.4.46 getIntervalSecond

Platform: all
Details
Get the second interval value for date the variable.

```
int getIntervalSecond(
        date cur
)
```

Parameters
cur
[in] The date with interval.

Return values
Returns -1 if the variable does not have valid second interval value (of no interval type date variable), otherwise return the second interval value.

6.4.47 setIntervalSecond

Platform: all
Details
Set the second interval value for the date variable.


```
void setIntervalSecond(
        date cur,
        int value
)
```

Parameters
cur
[in/out] The date variable.

value
[const] The value to be set.

Return values
None.

6.4.48 now

Platform: all
Details
Get the current date.

```
date now(

)
```

Parameters
None.

Return values
Returns the current date.

6.4.49 getNextNDate

Platform: all
Details
Get the next N occurrences of the date.

```
date getNextNDate(
        date cur,
        int times
)
```

Parameters
cur
[in] The date with interval and recurrence time.

times
[const] The specified date after times intervals.

Return values
Returns the date after times intervals.

6.4.50 setFlagTimeDesignator

Platform: all
Details
Set flag for showing time designator 'T' or not.

```
void setFlagTimeDesignator(
        date cur,
        int flag
)
```

Parameters
cur
[in/out] The date variable.

flag
[const] 0 for not showing 'T' but ' ', non-zero value to show 'T'.

Return values
None.

6.4.51 formatString

Platform: all
Details
Get formatted string for the date variable.

```
string formatString(
        date cur,
        string fmt
)
```

Parameters
cur
[in] The date variable.

fmt
[const] The date format. See Date specifiers for a list of supported formats.

Return values
Formatted string for the date variable.

6.4.52 hextime

Platform: all
Details
Get hex format string for the date variable.

```
string hextime (
        date cur
)
```

Parameters
cur
[in] The date variable.

Return values
Formatted string for the date variable.

6.5 KVGroup Functions

See KVGroups for a description of KVGroups.

6.5.1 kvgCreate

Platform: all
Details
Creates a new KVGroup with the given key and name.

```
kvgroup kvgCreate (
        string key ,
        string name
)
```

Parameters
key
[in] The key for the new group.

name
[in] The name for the new group.

Return values
The created group.

6.5.2 kvgCreateFromHandle

Platform: all
Details
Reads data from an I/O handle into a new KVGroup.

```
kvgroup kvgCreateFromHandle (
        handle handle ,
        int timeout ,
        int ioCode
)
```

Parameters
handle
[in] I/O handle to read from. The handle must be opened in binary mode.

timeout
[in] Timeout in seconds. A timeout of zero means to wait indefinitely. On Windows, timeout values only work on sockets.

ioCode
[out] An I/O return code. This function only returns the code for “Success” (0) or “Generic error” (4). See I/O Return Values for a list of possible return values.

Return values
The created group.

6.5.3 kvgCreateFromString

Platform: all
Details
Reads a serialized KVGroup from a string into a new KVGroup.

```
kvgroup kvgCreateFromString (
        string str ,
        int ioCode
)
```

Parameters
str
[const] String containing the serialized KVGroup.

ioCode
[out] An I/O return code. This function only returns the code for “Success” (0) or “Generic error” (4). See I/O Return Values for a list of possible return values.

Return values
The created group.

6.5.4 setkey

Platform: all
Details
Changes the key of a group. This function cannot be called on a non-root kvgroup.

```
void setkey (
        kvgroup kvg ,
        string key
)
```

Parameters
kvg
[in/out] The KVGroup to modify.

key
[in] The new key for kvg.

Return values
None.

6.5.5 kvgSetKey

Platform: all
Details
Changes the key of a group.

```
void kvgSetKey (
        kvgroup kvg ,
        string key
)
```

Parameters
kvg
[in/out] The KVGroup to modify.

key
[in] The new key for kvg.

Return values
None.

6.5.6 setname

Platform: all
Details
Changes the name of a group.


```
void setname (
        kvgroup kvg ,
        string name
)
```

Parameters
kvg
[in/out] The KVGroup to modify.

name
[in] The new name for kvg.

Return values
None.

6.5.7 kvgSetName

Platform: all
Details
Changes the name of a group.

```
void kvgSetName (
        kvgroup kvg ,
        string name
)
```

Parameters
kvg
[in/out] The KVGroup to modify.

name
[in] The new name for kvg.

Return values
None.

6.5.8 getkey

Platform: all
Details
Retrieves the key of a group.

```
string getkey (
        kvgroup kvg
)
```

Parameters
kvg
[const] The KVGroup whose key is to be retrieved.

Return values
The group's key.

6.5.9 kvgGetKey

Platform: all
Details
Retrieves the key of a group.

```
string kvgGetKey (
        kvgroup kvg
)
```

Parameters
kvg
[const] The KVGroup whose key is to be retrieved.

Return values
The group's key.

6.5.10 getname

Platform: all
Details
Retrieves the name of a group.

```
string getname (
        kvgroup kvg
)
```

Parameters
kvg
[const] The KVGroup whose name is to be retrieved.

Return values
The group's name.

6.5.11 kvgGetName

Platform: all
Details
Retrieves the name of a group.

```
string kvgGetName (
        kvgroup kvg
```


)

Parameters
kvg
[const] The KVGroup whose name is to be retrieved.

Return values
The group's name.

6.5.12 kvgKeyExists

Platform: all
Details
Checks if the given key exists in the key-value pairs contained within a group.

```
int kvgKeyExists (
        kvgroup kvg ,
        string key
)
```

Parameters
kvg
[const] The KVGroup to examine.

key
[in] The key to search for. Note: keys are case sensitive.

Return values
Returns 1 if there is at least 1 key-value pair with the given key, otherwise returns 0.

6.5.13 keyexist

Platform: all
Details
Checks if the given key exists in the key-value pairs contained within a group.

```
int keyexist (
        kvgroup kvg ,
        string key
)
```

Parameters
kvg
[const] The KVGroup to examine.

key
[in] The key to search for. Note: keys are case sensitive.

Return values

Returns 1 if there is at least 1 key-value pair with the given key, otherwise returns 0.

6.5.14 kvgGetKeys

Platform: all
Details
Retrieves the list of keys contained within a group.

```
array kvgGetKeys (
        kvgroup kvg
)
```

Parameters
kvg
[const] The KVGroup to examine.

Return values
Returns an array of strings consisting of the unique keys in kvg.

6.5.15 kvgGetValue

Platform: all
Details
Retrieves the first value matching key in a group. Note that the value returned may be (at least in principle) multi-valued. This is represented by quotes around each value. Individual values can be extracted using the parseLine() function.

```
int kvgGetValue (
        kvgroup kvg ,
        string key ,
        string value
)
```

Parameters
kvg
[const] The KVGroup to examine.

key
[in] The key to search for. Note: keys are case sensitive.

value
[out] The first value associated with key, if any.

Return values
Returns 1 if key was found, otherwise returns 0.

6.5.16 nodecount

Platform: all
Details
Returns number of child nodes in this kvg.

```
void nodecount (
        kvgroup kvg
)
```

Parameters
kvg
[const] The KVGroup to examine.

Return values
None.

6.5.17 kvgGetAllValues

Platform: all
Details
Retrieves all of the values matching key in a group.

```
array kvgGetAllValues (
        kvgroup kvg ,
        string key
)
```

Parameters
kvg
[const] The KVGroup to examine.

key
[in] The key to search for. Note: keys are case sensitive.

Return values
Returns the list of values associated with key.

6.5.18 kvgGetGroupKeyNames

Platform: all
Details
Retrieves the list of (key, name) identifiers of groups contained immediately within an outer group. Only the first level of contained groups is considered — this function does not recursively descend into subgroups.

```
array kvgGetGroupKeyNames (
        kvgroup kvg
```


)

Parameters
kvg
[const] The KVGroup to examine.

Return values
Returns an array of key-name group identifiers within kvg. The key/name values are stored inside an associate array. They could be iterated out by using integer index to retrieve name value directly and function keyAt() for key value respectively.

6.5.19 kvgGetGroup

Platform: all
Details
Retrieves the first group matching key and name within a group. Only the first level of contained groups is considered — this function does not recursively descend into subgroups.

```
int kvgGetGroup(
        kvgroup kvg,
        string key,
        string name,
        kvgroup groupOut
)
```

Parameters
kvg
[const] The KVGroup to examine.

key
[in] The key being searched. Note: keys are case sensitive.

name
[in] The name being searched. Note: names are case sensitive.

groupOut
[out] The first group associated with (key, name), if any.

Return values
Returns 1 if a group was found, otherwise returns 0.

6.5.20 kvgGetAllGroups

Platform: all
Details
Retrieves all of the groups matching key and name within a group. Only the first level of contained groups is considered — this function does not recursively descend into subgroups.


```
array kvgGetAllGroups (
        kvgroup kvg ,
        string key ,
        string name
)
```

Parameters
kvg
[const] The KVGroup to examine.

key
[in] The key being searched. Note: keys are case sensitive.

name
[in] The name being searched. Note: names are case sensitive.

Return values
Returns an array of KVGroups with (key, value) identifiers.

6.5.21 kvgGetAllGroupsByKey

Platform: all
Details
Retrieves all of the groups with the given key. Only the first level of contained groups is considered — this function does not recursively descend into subgroups.

```
array kvgGetAllGroupsByKey (
        kvgroup kvg ,
        string key
)
```

Parameters
kvg
[const] The KVGroup to examine.

key
[in] The key being searched. Note: keys are case sensitive.

Return values
Returns an array of KVGroups with the given key.

6.5.22 kvgAddValue

Platform: all
Details
Adds a key-value pair to a group.


```
void kvgAddValue (
        kvgroup kvg ,
        string key ,
        string value
)
```

Parameters
kvg
[in/out] The KVGroup to modify.

key
[in] The key to add.

value
[in] The value to add.

Return values
None.

6.5.23 kvgAddGroup

Platform: all
Details
Adds an inner group to a group.

```
void kvgAddGroup (
        kvgroup kvg ,
        kvgroup groupToAdd
)
```

Parameters
kvg
[in/out] The KVGroup to modify.

groupToAdd
[const] The KVGroup to add.

Return values
None.

6.5.24 kvgAddToGroup

Platform: all
Details
Adds the contents of src to an inner group identified by key and name. Only the first level of contained groups is considered — this function does not recursively descend into subgroups.


```
void kvgAddToGroup(
        kvgroup kvg,
        string key,
        string name,
        kvgroup src
)
```

Parameters
kvg
[in/out] The KVGroup to modify.

key
[in] The key of the group within kvg to modify. Note: keys are case sensitive.

name
[in] The name of the group within kvg to modify. Note: names are case sensitive.

src
[const] The KVGroup whose contents will be added to kvg.

Return values
None.

6.5.25 kvgSetValue

Platform: all
Details
Changes the value associated with an existing key within a group.

```
int kvgSetValue(
        kvgroup kvg,
        string key,
        string value
)
```

Parameters
kvg
[in/out] The KVGroup to modify.

key
[in] The key to modify. Note: keys are case sensitive.

value
[in] The new value for key.

Return values
Returns 1 if key was found and it was changed, otherwise returns 0.

6.5.26 setvalue

Platform: all
Details
Changes the value associated with an existing key within a group.

```
int setvalue (
        kvgroup kvg ,
        string key ,
        string value
)
```

Parameters
kvg
[in/out] The KVGroup to modify.

key
[in] The key to modify. Note: keys are case sensitive.

value
[in] The new value for key.

Return values
Returns 1 if key was found and it was changed, otherwise returns 0.

6.5.27 kvgSetGroup

Platform: all
Details
Replaces the contents of a group identified by key and name with the contents of src. Only the first level of contained groups is considered — this function does not recursively descend into subgroups.

```
int kvgSetGroup (
        kvgroup kvg ,
        string key ,
        string name ,
        kvgroup src
)
```

Parameters
kvg
[in/out] The KVGroup to modify.

key
[in] The key of the group to modify. Note: keys are case sensitive.

name
[in] The name of the group to modify. Note: names are case sensitive.

src

[const] The new group containing the elements that will replace the elements of the existing group identified by (key, name).

Return values
Returns 1 if the group was found and it was replaced, otherwise returns 0.

6.5.28 kvgClear

Platform: all
Details
Clears the contents of a group.

```
void kvgClear (
        kvgroup kvg
)
```

Parameters
kvg
[in/out] The KVGroup to clear.

Return values
None.

6.5.29 clearkvg

Platform: all
Details
Clears the contents of a group.

```
void clearkvg (
        kvgroup kvg
)
```

Parameters
kvg
[in/out] The KVGroup to clear.

Return values
None.

6.5.30 kvgDelPairs

Platform: all
Details
Deletes key-value pairs that match key.


```
int kvgDelPairs(
        kvgroup kvg,
        string key,
        int deleteFirstOnly
)
```

Parameters
kvg
[in/out] The KVGroup to modify.

key
[in] The key to delete. Note: keys are case sensitive.

deleteFirstOnly
[in] Set to 1 to only delete the first matching key; set to 0 to delete all matching keys.

Return values
Returns the number of deleted keys.

6.5.31 delpairs

Platform: all
Details
Deletes key-value pairs that match key.

```
int delpairs(
        kvgroup kvg,
        string key,
        int deleteFirstOnly
)
```

Parameters
kvg
[in/out] The KVGroup to modify.

key
[in] The key to delete. Note: keys are case sensitive.

deleteFirstOnly
[in] Set to 1 to only delete the first matching key; set to 0 to delete all matching keys.

Return values
Returns the number of deleted keys.

6.5.32 kvgDelValues

Platform: all
Details
Deletes key-value pairs matching key and value.


```
int kvgDelValues(
        kvgroup kvg,
        string key,
        string value,
        int deleteFirstOnly
)
```

Parameters
kvg
[in/out] The KVGroup to modify.

key
[in] The key to delete. Note: keys are case sensitive.

value
[in] The value to delete.

deleteFirstOnly
[in] Set to 1 to only delete the first matching key-value pair; set to 0 to delete all matching key-value pairs.

Return values
Returns the number of deleted key-value pairs.

6.5.33 kvgDelGroups

Platform: all
Details
Deletes inner groups matching key and name. Only the first level of contained groups is considered — this function does not recursively descend into subgroups.

```
int kvgDelGroups(
        kvgroup kvg,
        string key,
        string name,
        int deleteFirstOnly
)
```

Parameters
kvg
[in/out] The KVGroup to modify.

key
[in] The key to delete. Note: keys are case sensitive.

name
[in] The name to delete. Note: names are case sensitive.

deleteFirstOnly
[in] Set to 1 to only delete the first group matching key and name; set to 0 to delete all matching groups.

Return values
Returns the number of deleted groups.

6.5.34 delgroups

Platform: all
Details
Deletes inner groups matching key and name. Only the first level of contained groups is considered — this function does not recursively descend into subgroups.

```
int delgroups(
        kvgroup kvg,
        string key,
        string name,
        int deleteFirstOnly
)
```

Parameters
kvg
[in/out] The KVGroup to modify.

key
[in] The key to delete. Note: keys are case sensitive.

name
[in] The name to delete. Note: names are case sensitive.

deleteFirstOnly
[in] Set to 1 to only delete the first group matching key and name; set to 0 to delete all matching groups.

Return values
Returns the number of deleted groups.

6.5.35 kvgToHandle

Platform: all
Details
Writes a group to an I/O handle.

```
int kvgToHandle(
        kvgroup kvg,
        handle handle,
        int indent
)
```

Parameters
kvg
[const] The KVGroup to write.

handle
[in] The I/O handle to which kvg is written.

indent
[in] The indent level (set to zero if unsure).

Return values
An I/O return code. See I/O Return Values for a list of possible return values.

6.5.36 tohandle

Platform: all
Details
Writes a group to an I/O handle.

```
int tohandle (
        kvgroup kvg ,
        handle handle ,
        int indent ,
        string optEncoding ,
        string optFormat ,
        string optCharset ,
        string optBom
)
```

Parameters
kvg
[const] The KVGroup to write.

handle
[in] The I/O handle to which kvg is written.

indent
[in] The indent level (set to zero if unsure).

optEncoding
[in] Serialization encoding option (KVG, KVG1, XML or BER).

optFormat
[in] Serialization format option (PRETTY, DENSE or BARE).

optCharset
[in] Serialization charset option (UTF8 or NATIVE).

optBom
[in] Serialization bom option (DISABLE or ENABLE).

Return values
An I/O return code. See I/O Return Values for a list of possible return values.

6.5.37 kvgToString

Platform: all
Details
Writes a group to a string.

```
string kvgToString (
        kvgroup kvg
)
```

Parameters
kvg
[const] The KVGroup to write.

Return values
Returns the standard serialized form of kvg. See KVGroups for examples of the standard form.

6.5.38 tostring

Platform: all
Details
Writes a group to a string.

```
string tostring (
        kvgroup kvg ,
        string optEncoding ,
        string optFormat
)
```

Parameters
kvg
[const] The KVGroup to write.

optEncoding
[in] Serialization encoding option (KVG, KVG1, XML or BER).

optFormat
[in] Serialization format option (DENSE, PRETTY or BARE).

Return values
Returns the standard serialized form of kvg. See KVGroups for examples of the standard form.

6.5.39 kvgToHandleStart

Platform: all
Details
Writes the beginning (key and name) of group to an I/O handle.


```
int kvgToHandleStart (
        handle handle ,
        string key ,
        string name ,
        int indent
)
```

Parameters
handle
[in] The I/O handle to which the start of a KVGroup is written.

key
[in] The group key to write.

name
[in] The group name to write.

indent
[in] The indent level (set to zero if unsure).

Return values
An I/O return code.

6.5.40 kvgPairToHandle

Platform: all
Details
Writes a key-value pair to an I/O handle.

```
int kvgPairToHandle (
        handle handle ,
        string key ,
        string value ,
        int indent
)
```

Parameters
handle
[in] The I/O handle to which key and value are written.

key
[in] The key to write.

value
[in] The value to write.

indent
[in] The indent level (set to 2 if unsure).

Return values

An I/O return code. See I/O Return Values for a list of possible return values.

6.5.41 kvgToHandleEnd

Platform: all
Details
Writes the end of a group to an I/O handle.

```
int kvgToHandleEnd(
        handle handle,
        int indent
)
```

Parameters
handle
[in] The I/O handle to which the end of a KVGroup is written.

indent
[in] The indent level (set to zero if unsure).

Return values
An I/O return code. See I/O Return Values for a list of possible return values.

6.5.42 kvgDiff

Platform: all
Details
Takes in two kvgroups and returns two groups representing the difference.

```
kvgroup kvgDiff(
        kvgroup kvgOriginal,
        kvgroup kvgChanged
)
```

Parameters
kvgOriginal
[in] Original group.

kvgChanged
[in] Changed group.

Return values
A kvgroup representing the changes.

6.6 Windows Registry Functions

The following functions provide read-only access to the Windows registry.

6.6.1 Root Keys

For each registry function, the *rootKey* must be one of the following strings:

- HKEY_CLASSES_ROOT
- HKEY_CURRENT_CONFIG
- HKEY_CURRENT_USER
- HKEY_LOCAL_MACHINE
- HKEY_USERS
- HKEY_PERFORMANCE_DATA
- HKEY_PERFORMANCE_TEXT
- HKEY_PERFORMANCE_NLSTEXT
- HKEY_DYN_DATA

> **Note:** Some of the above keys only exist in certain versions of Windows. Check the registry on the target machine to ensure that the desired root key exists, or use the RegKeyExists function (see RegKeyExists).

6.6.2 RegReadInt

Platform: all
Details
Reads an integer from the Windows registry.

```
int RegReadInt (
        string rootKey ,
        string subKey ,
        string keyName ,
        int value
)
```

Parameters
rootKey
[in] The root key to examine. For more details, see Root Keys.

subKey
[in] The subkey to examine. For example, "SOFTWARE\\Hitachi ID" is a valid subkey. Note that, as with all literal strings, all backslashes must be escaped with a backslash, as shown in the given example.

keyName
[in] The name of the registry entry to examine within subKey.

value
[out] The value of keyName. Set to 0 if the registry key does not exist.

Return values
Returns 1 if the retrieval was successful, otherwise returns 0.

6.6.3 RegReadString

Platform: all
Details
Reads a string from the Windows registry.

```
int RegReadString(
        string rootKey,
        string subKey,
        string keyName,
        string value
)
```

Parameters
rootKey
[in] The root key to examine. For more details, see Root Keys.

subKey
[in] The subkey to examine. For example, "SOFTWARE\\Hitachi ID" is a valid subkey. Note that, as with all literal strings, all backslashes must be escaped with a backslash, as shown in the given example.

keyName
[in] The name of the registry entry to examine within subKey.

value
[out] The value of keyName. Set to an empty string if the registry key does not exist.

Return values
Returns 1 if the retrieval was successful, otherwise returns 0.

6.6.4 RegReadMultiString

Platform: all
Details
Reads a multi-string from the Windows registry.


```
int RegReadMultiString(
        string rootKey,
        string subKey,
        string keyName,
        array values
)
```

Parameters
rootKey
[in] The root key to examine. For more details, see Root Keys.

subKey
[in] The subkey to examine. For example, "SOFTWARE\\Hitachi ID" is a valid subkey. Note that, as with all literal strings, all backslashes must be escaped with a backslash, as shown in the given example.

keyName
[in] The name of the registry entry to examine within subKey.

values
[out] An array of string values. Set to a 0-element array if the registry key does not exist.

Return values
Returns 1 if the retrieval was successful, otherwise returns 0.

6.6.5 RegReadBinary

Platform: all
Details
Reads a binary value from the Windows registry.

```
int RegReadBinary(
        string rootKey,
        string subKey,
        string keyName,
        array values
)
```

Parameters
rootKey
[in] The root key to examine. For more details, see Root Keys.

subKey
[in] The subkey to examine. For example, "SOFTWARE\\Hitachi ID" is a valid subkey. Note that, as with all literal strings, all backslashes must be escaped with a backslash, as shown in the given example.

keyName
[in] The name of the registry entry to examine within subKey.

values
[out] An array of integer values, 1 element per byte.

Return values
Returns 1 if the retrieval was successful, otherwise returns 0.

6.6.6 RegKeyExists

Platform: all
Details
Determines whether or not the given key exists.

```
int RegKeyExists(
        string rootKey,
        string subKey,
        string keyName
)
```

Parameters
rootKey
[in] The root key to examine. For more details, see Root Keys.

subKey
[in] The subkey to examine. For example, "SOFTWARE\\Hitachi ID" is a valid subkey. Note that, as with all literal strings, all backslashes must be escaped with a backslash, as shown in the given example.

keyName
[in] The name of the registry entry to examine within subKey.

Return values
If rootKey, subKey, and keyName are all non-blank, returns whether or not the key described by the 3 variables exist. If only keyName is blank, returns whether or not the rootKey\subKey combination exists. If subKey and keyName are both blank, returns whether or not rootKey exists. 0 is returned for any other combination of blank and non-blank parameters.

6.6.7 RegGetPSynchRegPath

Platform: all
Details
Returns the correct P-Synch key for the current instance.

```
string RegGetPSynchRegPath(

)
```

Parameters
None.

Return values
Returns the registry key for the current instance of P-Synch. Usually in the form "SOFTWARE\Hitachi ID\P-Synch\instanceName". Combine with a rootKey of "HKEY_LOCAL_MACHINE" to read P-Synch registry

entries.

6.6.8 RegGetSubKeys

Platform: all
Details
Enumerates the subkeys of a given registry key.

```
int RegGetSubKeys (
        string rootKey ,
        string subKey ,
        array subkeys
)
```

Parameters
rootKey
[in] The root key to examine. For more details, see Root Keys.

subKey
[in] The subkey to examine. For example, "SOFTWARE\\Hitachi ID" is a valid subkey. If this parameter is blank, the rootKey's subkeys are listed. Note that, as with all literal strings, all backslashes must be escaped with a backslash, as shown in the given example.

subkeys
[out] An array of string values — the subkeys of rootKey\subKey. Set to a 0-element array if the registry key does not exist.

Return values
Returns 1 if the retrieval was successful, otherwise returns 0.

6.6.9 RegGetValueNames

Platform: all
Details
Enumerates the value names contained within a given registry key.

```
int RegGetValueNames (
        string rootKey ,
        string subKey ,
        array subkeys
)
```

Parameters
rootKey
[in] The root key to examine. For more details, see Root Keys.

subKey
[in] The subkey to examine. For example, "SOFTWARE\\Hitachi ID" is a valid subkey. If this parameter

is blank, the rootKey's value names are listed. Note that, as with all literal strings, all backslashes must be escaped with a backslash, as shown in the given example.

subkeys
[out] An array of string values — the value names of rootKey\subKey. Set to a 0-element array if the registry key does not exist.

Return values
Returns 1 if the retrieval was successful, otherwise returns 0.

6.7 Persistent Storage Functions

The following functions provide access to a persistent storage facility. That is, the strings stored with these functions may be retrieved by other PSLANG scripts or by new instances of the same scripts. These functions are available only on Windows.

Persistent storage functions are not available to the scripted connectors such as the *Win32 Console Connector*.

> **Note:** If functions to remove the values are not called eventually, the values will remain in storage and will continue to use disk space. Thus, a script should remove the values as soon as it knows that the data are no longer needed.
>
> The last access time and last modified time is maintained for each value. You can use these values as criteria for deleting old data.

6.7.1 StoreUpdate

Platform: all
Details
Adds or updates a value in persistent storage. Any existing values for the key are removed.

```
int StoreUpdate(
        string namespace,
        string key,
        string value
)
```

Parameters
namespace
[const] The namespace in which to add or update the key and value. Limited to a length of 100.

key
[const] The key with which to store the value. Limited to a length of 100.

value
[const] The value to store.

Return values
Returns negative value on error, 1 if an existing value was cleared, otherwise returns 0.

6.7.2 StoreAppend

Platform: all
Details
Appends a new value to a key's existing values in persistent storage.

```
int StoreAppend(
        string namespace,
        string key,
        string value
)
```

Parameters
namespace
[const] The namespace in which to append values to a key. Limited to a length of 100.

key
[const] The key with which to store the value. Limited to a length of 100.

value
[const] The value to store.

Return values
Returns negative value on error, 0 if the value already exists (duplicates are not allowed). Otherwise returns 1.

6.7.3 StoreClear

Platform: all
Details
Clears all values associated with a key in persistent storage.

```
int StoreClear(
        string namespace,
        string key
)
```

Parameters
namespace
[const] The namespace in which to clear the key's values.

key
[const] The key with which to store the value.

Return values
Returns negative value on error, otherwise the number of values removed.

6.7.4 StoreDelete

Platform: all
Details
Deletes the given value associated with a key in persistent storage.

```
int StoreDelete (
        string namespace ,
        string key ,
        string value
)
```

Parameters
namespace
[const] The namespace in which to delete a key's value.

key
[const] The key of the value to be deleted.

value
[const] The value to delete.

Return values
Returns negative value on error, 1 if the value was deleted, otherwise returns 0.

6.7.5 StoreGet

Platform: all
Details
Retrieves all values associated with a key in persistent storage.

```
array StoreGet (
        string namespace ,
        string key
)
```

Parameters
namespace
[const] The namespace from which to retrieve values from a key.

key
[const] The key from which to retrieve the values.

Return values
Returns the list of values associated with key. Each element is a string. If the array is empty, then there were no values associated with that key.

6.7.6 StoreGetFirst

Platform: all
Details
Retrieves the first value associated with a key in persistent storage.

```
string StoreGetFirst (
        string namespace ,
        string key
)
```

Parameters
namespace
[const] The namespace from which to retrieve values from a key.

key
[const] The key from which to retrieve the values.

Return values
Returns the first value associated with key. If there are no values, then a blank string is returned. When a blank string is returned, the StoreCount function can be used to determine if there is a value (which happens to be blank) or if there are no values.

6.7.7 StoreGetKeys

Platform: all
Details
Retrieves all keys associated with a namespace in persistent storage.

```
array StoreGetKeys (
        string namespace
)
```

Parameters
namespace
[const] The namespace from which to retrieve keys.

Return values
Returns the list of keys associated with namespace. Each element is a string. If the array is empty, then there were no keys associated with that namespace or the namespace does not exist.

6.7.8 StoreCount

Platform: all
Details
Retrieves the number of values associated with a key in persistent storage.


```
int StoreCount (
        string namespace ,
        string key
)
```

Parameters
namespace
[const] The namespace from which to retrieve values from a key.

key
[const] The key from which to retrieve the values.

Return values
Returns negative value on error, otherwise the number of values associated with key.

6.8 Utility Functions

6.8.1 CallFunction

Platform: all
Details
Call a dynamic function to perform desired action.

```
string CallFunction (
        string funcname ,
        array inargs ,
        array outargs ,
        universal funcretval
)
```

Parameters
funcname
[const] The dynamic function name.

inargs
[const] The input arguments to the function.

outargs
[in/out] The output arguments to the function.

funcretval
[out] The return value from the dynamic function.

Return values
Returning empty string if success, otherwise, returning the error message.

6.8.2 pslangMemoryUSage

Platform: all
Details
Writes a message to indicate the current heap usage information.

```
void pslangMemoryUSage (
        loglevel level
)
```

Parameters
level
[in] The logging level. It could be one of LOG_NONE,LOG_ERROR, LOG_WARNING, LOG_NOTICE, LOG_INFO, LOG_DEBUG, or LOG_VERBOSE. LOG_NONE means writing the message to stdout.

Return values
None.

6.8.3 mail

Platform: win32
Details
Sends an email. If any of from, to, mailServer, or mailPort are unspecified, the corresponding value from global email settings is used. A zero or negative mailPort is considered unspecified. Global email settings are used for userid and password only if mailServer is unspecified.

```
int mail (
        string from ,
        string userid ,
        string password ,
        string to ,
        string cc ,
        string bcc ,
        string subject ,
        string body ,
        string mailServer ,
        int mailPort ,
        string errmsg
)
```

Parameters
from
[in] Address that the message will be from.

userid
[in] Userid used to authenticate to mail server.

password
[in] Password to authenticate to mail server.

to
[in] Comma-separated string of addresses that the message is sent to.

cc
[in] Comma-separated string of carbon copy addresses.

bcc
[in] Comma-separated string of blind carbon copy addresses.

subject
[in] Subject of the message.

body
[in] Body of the message.

mailServer
[in] Hostname of the SMTP server to contact.

mailPort
[in] Port that the SMTP server is listening on.

errmsg
[out] Error message on failure.

Return values
Returns 1 (true) on success, 0 on failure.

6.8.4 system

Platform: all
Details
Runs a program.

```
int system(
        string cmd,
        array argv,
        string stdinData,
        string stdoutData,
        int returnVal,
        int timeout
)
```

Parameters
cmd
[in] The command to run. Note that it requires a full path, with double-backslash inside string constants to separate directories.

argv
[const] Array of arguments. Each element is passed as one command-line parameter and is converted to a string (if necessary).

stdinData
[in] String of data to pass in on the child's stdin.

stdoutData
[out] String of data captured from the child's stdout.

returnVal
[out] Return code of the child process.

timeout
[in] Timeout in seconds. Use -1 for no timeout.

Return values
Success code zero indicates the system function has successfully initialized; returnVal indicates the success or failure of the child process.

6.8.5 log

Platform: all
Details
Writes a message to the program log file.

```
void log(
        loglevel level,
        universal message
)
```

Parameters
level
[in] The logging level. It could be one of LOG_ERROR, LOG_WARNING, LOG_NOTICE, LOG_INFO, LOG_DEBUG, or LOG_VERBOSE. This argument can be omitted. If so, the logging level will be by default LOG_INFO.

message
[in] The message to write to the log.

Return values
None.

6.8.6 getLogDirectory

Platform: win32
Details
Gets the current log file directory. This may be (in order of preference) a subdirectory of the PSTEMP directory, the PSTEMP directory itself, the value of the TEMP environment variable, or the current working directory.

```
string getLogDirectory(
```


)

Parameters
None.

Return values
The log file directory, including a trailing backslash.

6.8.7 defined

Platform: all
Details
Checks if the variable defined.

```
int defined(
        string varname
)
```

Parameters
varname
[in] The variable name to check.

Return values
Returns 1 (true) if the variable exists, otherwise 0.

6.8.8 getvar

Platform: all
Details
Finds the value of the variable specified, if not found, returns default value.

```
universal getvar(
        string varname,
        universal defvalue
)
```

Parameters
varname
[in] The variable name to find.

defvalue
[const] The default value to return.

Return values
Returns the value of the specified variable if found, otherwise return the value of the defvalue.

6.8.9 crypt_md5

Platform: all
Summary: Generate the hashed password (based on Linux/BSD crypt).
Details
Generates a one-way password hash for comparison to stored passwords elsewhere.

```
int crypt_md5 (
        string uncryptPW ,
        string salt ,
        string cryptPW
)
```

Parameters
uncryptPW
[in] Password to encrypt and hash.

salt
[in] Salt to use to vary the hash.

cryptPW
[out] Encrypted and hashed password to use for comparison.

Return values
0 on success, otherwise failure.

6.8.10 ciscoObfuscate

Platform: all
Summary: (Deprecated) Generate the hashed password suitable for Cisco type 7 encryption.
Details
Generates an obfuscation of the password used when setting a user's password.

```
int ciscoObfuscate (
        string plainpass ,
        string obfuscated
)
```

Parameters
plainpass
[in] Password to encrypt and hash.

obfuscated
[out] Obfuscated password to use for setting or updating the Cisco type 7 password.

Return values
0 on success, otherwise failure.

6.8.11 hash

Platform: all
Summary: Generates a hash of the supplied input string, using the specified algorithm supported algorithms are: md5, sha1, sha256, sha512.
Details
Generates a hash of the supplied input string, using the specified algorithm supported algorithms are: md5, sha1, sha256, sha512.

```
string hash (
        string algorithm ,
        string str ,
        string encoding ,
        int salt
)
```

Parameters
algorithm
[in] Algorithm to use for hash can be: md5, sha1, sha256, sha512.

str
[in] String to hash.

encoding
[in] Encoding algorithm to use for the hash supported encodings are: hex, base64.

salt
[in] Include random salt in hash? 0 = don't include salt, 1 = include random salt.

Return values
On success the hashed string is returned, on failure an empty string is returned.

See also:

For information on how to use **`cmd.exe`** with system() refer to Using cmd.exe with system().

6.9 Miscellaneous Functions

6.9.1 encodebase64

Platform: all
Details
Generates the base 64 encoded string.

```
void encodebase64 (
        string instr ,
        string retstr
)
```

Parameters
instr
[const] Input string.

retstr
[out] Base 64 encoded string.

Return values
None.

6.9.2 decodebase64

Platform: all
Details
Generates the base 64 decoded string.

```
void decodebase64 (
        string instr ,
        string retstr
)
```

Parameters
instr
[const] Input string in base 64 encode format.

retstr
[out] Decoded string.

Return values
None.

6.9.3 toarray

Platform: all
Details
Generates the array containing struct information.

```
array toarray (
        struct stvar
)
```

Parameters
stvar
[const] Struct variable.

Return values
The generated array.

6.9.4 getenv

Platform: all
Details
Retrieve an environment variable.

```
string getenv (
        string name
)
```

Parameters
name
[const] The name of the variable to retrieve.

Return values
The environment value.

6.9.5 typeof

Platform: all
Details
Check the type of the variable.

```
string typeof (
        universal variable
)
```

Parameters
variable
[const] The variable to return the type of.

Return values
Returns the type of the variable. Showing one of the following, 'integer', 'string', 'float', 'array', 'kvgroup', 'date', 'struct <struct ID>', 'handle'(for Input/Output handle), 'invalid' (no type).

6.9.6 isInArray

Platform: all
Details
Check if the specified value exists in the array.

```
int isInArray (
        array array ,
        universal value
)
```

Parameters
array
[const] The array.

value
[const] The value for checking. This could be integer, float or string.

Return values
Returns 1 for the value existing in the array and 0 for not existing.

6.9.7 sort

Platform: all
Details
Sort the elements in the array.

```
void sort (
        array array ,
        int method
)
```

Parameters
array
[in/out] The array with homogeneous items of type integer, float or string.

method
[const] The sorting method, 0 for ascending order and other for descending order.

Return values
None.

6.9.8 size

Platform: all
Details
Returns the size of an array.

```
int size (
        array array
)
```

Parameters
array
[const] The array.

Return values
Returns the number of elements in the array.

6.9.9 keyExists

Platform: all
Details
Determines whether or not a given key exists in an associative array.

```
int keyExists (
        array array ,
        string key
)
```

Parameters
array
[const] The associative array to examine.

key
[in] The key to find.

Return values
Returns 1 if the key exists in the array, and 0 if the key does not exist.

6.9.10 index

Platform: all
Details
Gets the position index in an associative array for a given string value.

```
int index (
        array array ,
        string sval
```


)

Parameters
array
[const] The associative array to examine.

sval
[in] A string supposed to be mapping into the array.

Return values
Returns the position index at the given string, returns -1 if the string does not exist in the array.

6.9.11 keyAt

Platform: all
Details
Gets the key in an associative array for a given numeric position.

```
string keyAt (
        array array ,
        int pos
)
```

Parameters
array
[const] The associative array to examine.

pos
[in] A zero-based numeric index into the array. Must be a non-negative number smaller than the size of the array.

Return values
Returns the key at the given position.

6.9.12 valAt

Platform: all
Details
Gets the value in an associative array for a given numeric position.

```
string valAt (
        array array ,
        int pos
)
```

Parameters
array
[const] The associative array to examine.

pos
[in] A zero-based numeric index into the array. Must be a non-negative number smaller than the size of the array.

Return values
Returns the value at the given position. The return type can actually be of any type, not just string.

6.9.13 regexp

Platform: all
Details
Checks whether or not part of a string satisfies a given regular expression.

```
int regexp (
        string str ,
        string regexp ,
        int insensitive ,
        array matches ,
        string error
)
```

Parameters
str
[in] The string to examine.

regexp
[in] The regular expression to use.

insensitive
[in] If non-zero, regexp performs a case-insensitive comparison.

matches
[out] An array of strings containing the matches found by the function. Element 0 contains matches for the entire regular expression, and the following elements are matches for each parenthetized subexpression in the main regular expression.

error
[out] Contains an error message when a value other than 0 or 1 is returned.

Return values
Returns 0 for match, 1 for no match, and other values for various errors.

6.9.14 obscure

Platform: all
Details
Obscures (blurs) a string.


```
string obscure (
        string strIn
)
```

Parameters
strIn
[in] The string to be obscured.

Return values
Returns an obscured string.

6.9.15 unObscure

Platform: all
Details
Unobscures (decodes) a string. The result of unObscure(obscure($str)) is equivalent to $str.

```
string unObscure (
        string strIn
)
```

Parameters
strIn
[in] The string to be unobscured.

Return values
Returns an unobscured string.

6.9.16 parseLine

Platform: all
Details
Parses (decodes) a string representing multiple string values into an array of strings. The result of parseLine(unparseLine($array)) is equivalent to $array.

```
array parseLine (
        string str
)
```

Parameters
str
[in] The string to parse/decode.

Return values
Returns an array of strings corresponding to the items encoded in str.

6.9.17 unparseLine

Platform: all
Details
Unparses (encodes) an array of strings into an encoded string. The result of unparseLine(parseLine($str)) is equivalent to $str.

```
string unparseLine (
        array array
)
```

Parameters
array
[const] The array of strings to unparse/encode.

Return values
Returns an encoded string equivalent to the given array.

6.9.18 escapeForHTMLDisplay

Platform: all
Details
Given a string, produce a modified string which may be safely placed into an HTML document. For example, '<', '>', and '&' are replaced with '<', '>', and '&' respectively.

```
string escapeForHTMLDisplay (
        string str
)
```

Parameters
str
[in] The string to escape.

Return values
Returns an escaped string.

6.9.19 joinStrings

Platform: all
Summary: Reverse splitString().
Details
Joins an array of strings using the characters specified in the delimiter into a single string.

```
string joinStrings (
        array strs ,
        string delim ,
        int skipempty
```


)

Parameters
strs
[in] The strings to combine into a single string.

delim
[in] The character(s) to be used as delimiters.

skipempty
[in] Whether to skip empty word: 1 means skip empty, 0 will keep empty word.

Return values
Returns a string of the array elements joined by the delimiters.

6.9.20 splitString

Platform: all
Details
Splits a string into an array of strings using the characters specified in the delimiter.

```
array splitString (
        string str ,
        string delim ,
        int skipempty
)
```

Parameters
str
[in] The string to split.

delim
[in] The characters to be used as delimiters.

skipempty
[in] Whether to skip empty word: 1 means skip empty, 0 will keep empty word.

Return values
Returns an array of strings split by the delimiters.

6.9.21 srand

Platform: all
Details
Sets the seed for a series of random numbers. This function should be called only once per pslang script. Once the seed has been set, you can use the rand function to get random numbers.

```
int srand (
        int seed
```


)

Parameters
seed
[in] The seed for producing random numbers. Use -1 for system default.

Return values
Always returns 0.

6.9.22 rand

Platform: all
Details
Creates a random number. Ensure srand is used to seed the random number generator.

```
int rand(
```

)

Parameters
None.

Return values
Returns a random number.

6.10 Flat-file Functions

The Flat-file functions provide read-only iteration over various types of flat file databases, such as comma-separated value, or fixed-length record files.

Records in a csv file are parsed using the following format:

- Read records line by line
- Read record values field by field (using quotes to permit commas)

6.10.1 flatFileOpen

Platform: win32
Details
Opens a handle to a flat file database. Output is an array of field names. The returned handle can only be used with flatFileGetNext for forward-only iteration over the records in the database.


```
int flatFileOpen(
        string dataSource,
        handle handle,
        array fields
)
```

Parameters
dataSource
[in] The connection string to open. Details are provider-dependent, but consist of semicolon-separated parameters of the form filename;param1=val1;param2=val2. Literal semicolons can be specified with two semicolons in a row.

handle
[out] The opened flatfile I/O handle.

fields
[out] A list of available field names in the database.

Return values
See I/O Return Values for a list of possible return values.

6.10.2 flatFileGetNext

Platform: win32
Details
Reads the next record out of a handle returned from flatFileOpen. Output is a kvgroup, with the field names as keys. Returns success as long as records are available to be read. Returns end-of-file at the end of the database.

```
int flatFileGetNext(
        handle handle,
        kvgroup record
)
```

Parameters
handle
[in] The flatfile I/O handle to read from.

record
[out] The returned record.

Return values
See I/O Return Values for a list of possible return values.

Bravura Security Fabric API Function Reference

7

Bravura Security Fabric API functions can be used inside PSLANG scripts. To use an Bravura Security Fabric API function, you must:

1. Declare the API handle for calling the Bravura Security Fabric API function.
2. Initialize the API handle by calling the APIInit function and specifying the host and svcid parameters. Currently the API is only available locally so the host parameter should always be specified as localhost.
3. Execute any number of the API functions by calling the APIExecute function. If the function returns records, the first record is automatically returned in the output struct. You can use the APIGetNextRecord function to get the next record that might have been returned by the previously executed APIExecute function. If the function does not return records, an alternative version of APIExecute exists which does not have an output parameter for records.
4. Complete each API function by calling the APIExecClose method.
5. Uninitialize the API handle by calling the APIUnInit function.

The following example details the general sequence and how to use the calling interface functions.

```
include "idapi.psl"

function main( const $argv )
  {
  var $apihandle;
  var $ret;

  $ret = APIInit( "localhost", "3", $apihandle );
  if( $ret != 0 )
    {
    return 0;
    }

  struct Login_scalarInput $inLogin;
  $inLogin.$userid = "jsmith";
  $inLogin.$password = "abcd1234";
  $inLogin.$isadmin = 1;
  var $errmsg;

  $ret = APIExecute( $apihandle, "Login", $inLogin, $errmsg );
  APIExecClose( $apihandle );
  if( $ret == 0 )
    {
```

```
  struct UserGetByID_scalarInput $inUserGetByID;
  struct UserGetByID_vectorOutput $outUserGetByID;
  $inUserGetByID.$userid = "djones";

  $ret = APIExecute( $apihandle, "UserGetByID", $inUserGetByID, $outUserGetByID,
   $errmsg );
  if( $ret == 0 )
    {
    while( $ret == 0 )
      {
      println( "  alias = [" + $outUserGetByID.$alias + "], isadmin = " +
       $outUserGetByID.$isadmin );

      $ret = APIGetNextRecord( $apihandle, $outUserGetByID );
      }
    }
  else
    {
    println( "Could not get user; error message is [" + $errmsg + "]" );
    }
  APIExecClose( $apihandle );

  struct Logout_scalarInput $inLogout;
  $ret = APIExecute( $apihandle, "Logout", $inLogout, $errmsg );
  APIExecClose( $apihandle );
  }
else
  {
  println( "Could not login; error message is [" + $errmsg + "]" );
  }

APIUnInit( $apihandle );
}
```

See the function reference below for a complete list of available Bravura Security Fabric API functions.

7.1 *Bravura Security Fabric* API Calling Interface Function Reference

7.1.1 APIInit

Platform: win32
Details
Initializes a handle for subsequent API calls.

```
int APIInit (
        string host ,
        string svcid ,
        handle outhandle
)
```

Parameters
host
[in] The computer name, IP address, localhost, or 127.0.0.1.

svcid
[in] The ID for the IDAPI service (default = 3).

outhandle
[out] The API handle.

Return values
0 success.

7.1.2 APIUnInit

Platform: win32
Details
Releases the API handle.

```
int APIUnInit (
        handle outhandle
)
```

Parameters
outhandle
[in/out] The API handle.

Return values
0 success.

7.1.3 APIExecute

Platform: win32
Details
Executes the specified API function with provided input data. The first set of output data is placed in the outputData argument. Use APIGetNextRecord (before calling APIExecClose) for the next set of output data.

```
int APIExecute(
        handle apihandle,
        string funcName,
        struct inputData,
        struct outputData,
        string errmsg
)
```

Parameters
apihandle
[in] The API handle.

funcName
[const] API function name.

inputData
[in] API input data.

outputData
[out] First set of output data.

errmsg
[out] Success or an error description.

Return values
Returns 0 on success, and -1 on failure to connect to the service. See **??** for a list of possible error values and their associated meanings.

7.1.4 APIGetNextRecord

Platform: win32
Details
Fetches the next set of output data after the input argument. The output data will be placed in the argument.

```
int APIGetNextRecord(
        handle apihandle,
        struct inoutData
)
```

Parameters
apihandle
[in] The API handle.

inoutData
[in/out] As input specify the last set of data obtained, as output the next set of data will be placed here.

Return values
Returns 0 on success, and -1 on failure to connect to the service. See **??** for a list of possible error values and their associated meanings.

7.1.5 APIExecClose

Platform: win32
Details
Closes the output data handle for a corresponding APIExecute call. No output data will be available to APIGetNextRecord after this call.

```
int APIExecClose(
        handle apihandle
)
```

Parameters
apihandle
[in/out] The API handle.

Return values
Returns 0 on success, and -1 on failure to connect to the service. See **??** for a list of possible error values and their associated meanings.

7.2 *Bravura Security Fabric* API Function Reference

The following are the available API functions and their associated input and output types. Calls to Login (or LoginEx) and Logout are required to begin and end an API session. A typical session would proceed as follows (API calling interface function details removed for clarity):

```
Login...
apiFunction1...
apiFunction2...
...
apiFunctionN...
Logout...
```

User authentication using Login and LoginEx requires a product administrator with the IDAPI caller and OTP IDAPI caller administrative privilege, respectively. See the *Bravura Security Fabric* Documentation for more information.

Searching for resources with ResourceFind

ResourceFind (p267) is used to find a resource of a given type. For each resource type, there is a set of search criteria that can be used to search for resources. See ResourceFind Search Criteria for a list of the search criteria for the supported resource types.

For example, two of the available search criteria for role resources are *enabledOnly* and *assignable*. To find all roles that are enabled and assignable, call ResourceFind (p267) with `ROLE` for the scalar input, and the following vector input:

name	*value*
`enabledOnly`	`T`
`assignable`	`T`

7.2.1 AccountAdd

Record the existence of an account. This only informs the product of the existence of an account, it does not create the account on the target system.

```
struct AccountAdd_scalarInput
    {
    var $targetid
    var $longid
    var $shortid
    };
```

Parameters

targetid
[in] The host ID to be used.

longid
[in] The long ID of the account.

shortid
[in] The short ID of the account.

7.2.2 AccountAttrDelete

Record the removal of values for target system account attributes. This only informs the product of modifications to values, it does not change the target system account attributes on the target system.

```
struct AccountAttrDelete_scalarInput
    {
    var $targetid
    var $longid
    var $attrkey
```



```
    };
```

Parameters

targetid
[in] The host ID to be used.

longid
[in] The long ID of the account.

attrkey
[in] The name of the target attribute key.

7.2.3 AccountAttrLargeCredSet

Propagates a large credential record from request file attribute to account file attribute.

```
struct AccountAttrLargeCredSet_scalarInput
    {
    var $targetid
    var $longid
    var $batchid
    var $attrkey
    };
```

Parameters

targetid
[in] The ID of the managed system.

longid
[in] The ID of the account.

batchid
[in] The ID identifying the batch the submitted large credential file attribute is in.

attrkey
[in] The name of the request attribute.

7.2.4 AccountAttrSet

Record values for target system account attributes. This only informs the product of modifications to values, it does not change the target system account attributes on the target system. If any account attributes exist for the attributes passed in, they will be replaced.

```
struct AccountAttrSet_scalarInput
    {
    var $targetid
    var $longid
```



```
    var $attrkey
    struct AccountAttrSet_vectorInput $inputArray[]
    };

struct AccountAttrSet_vectorInput
    {
    var $value
    };
```

Parameters

targetid
[in] The host ID to be used.

longid
[in] The long ID of the account.

attrkey
[in] The name of the target attribute key.

inputArray
[in] An array of structs of type AccountAttrSet_vectorInput

value
[in] The values of the attribute. At least one value must be specified.

7.2.5 AccountAttrsGet

Retrieves the attribute values associated with the given account. Only the target attributes loaded into Bravura Security Fabric will be returned.

```
struct AccountAttrsGet_scalarInput
    {
    var $longid
    var $targetid
    };

struct AccountAttrsGet_vectorOutput
    {
    var $name
    var $value
    };
```

Parameters

longid
[in] The long ID of the account.

targetid
[in] The ID of the target.

name
[out] The name of the attribute.

value
[out] The value of the attribute.

7.2.6 AccountChallengeResponse

Retrieve a response from a target based on a user-specified challenge string.

```
struct AccountChallengeResponse_scalarInput
    {
    var $longid
    var $targetid
    var $challenge
    var $inputs
    };

struct AccountChallengeResponse_vectorOutput
    {
    var $response
    };
```

Parameters

longid
[in] Account ID to provide a challenge for.

targetid
[in] Target ID to query.

challenge
[in] The challenge string.

inputs
[in] Optional target-specific additional inputs. These are specified as a comma-separated string of name=value pairs.

response
[out] The response returned by the target.

7.2.7 AccountDelete

Record that an account has ceased to exist. This only informs the product of the removal of an account, it does not remove the account on the target system.

```
struct AccountDelete_scalarInput
    {
    var $targetid
```



```
    var $longid
    };
```

Parameters

targetid
[in] The host ID to be used.

longid
[in] The long ID of the account.

7.2.8 AccountGetByAcctAttr

Retrieves accounts by the given target attribute value.

```
struct AccountGetByAcctAttr_scalarInput
    {
    var $targetid
    var $attrkey
    var $value
    var $iscasesensitive
    };

struct AccountGetByAcctAttr_vectorOutput
    {
    var $longid
    var $shortid
    };
```

Parameters

targetid
[in] The ID of the target on which the account resides.

attrkey
[in] The name of the target attribute key.

value
[in] The value of the account attribute.

iscasesensitive
[in] 1 for case sensitive, otherwise case insensitive.

longid
[out] The long ID of the account.

shortid
[out] The short ID of the account.

7.2.9 AccountGetByGUID

Retrieves the target ID and long ID for an account based on its GUID.

```
struct AccountGetByGUID_scalarInput
    {
    var $guid
    };

struct AccountGetByGUID_vectorOutput
    {
    var $targetid
    var $longid
    };
```

Parameters

guid
[in] The account's GUID.

targetid
[out] The ID of the target the account exists on.

longid
[out] The long ID of the account.

7.2.10 AccountGetByName

Retrieves details for an account based on its target ID and long ID.

```
struct AccountGetByName_scalarInput
    {
    var $targetid
    var $longid
    };

struct AccountGetByName_vectorOutput
    {
    var $guid
    };
```

Parameters

targetid
[in] The ID of the target the account exists on.

longid
[in] The long ID of the account.

guid

[out] The account's GUID.

7.2.11 AccountGroupGet

Returns a group given its target ID and either group name or group ID. Group name and group ID cannot both be specified. If the group is not managed, then only the shortid and group ID returned by the function will contain values; the remaining output values will be empty or zero.

```
struct AccountGroupGet_scalarInput
    {
    var $targetid
    var $shortid
    var $groupid
    var $caseinsensitiveflag
    };

struct AccountGroupGet_vectorOutput
    {
    var $shortid
    var $groupid
    var $overridedesc
    var $url
    var $outofbandadd
    var $outofbanddel
    var $location
    var $type
    var $ownerasauth
    var $overrideauth
    var $rbacenforce
    var $daction
    var $saction
    };
```

Parameters

targetid
[in] The ID of the target.

shortid
[in] The name of the target group.

groupid
[in] The ID of the target group.

caseinsensitiveflag
[in] Case insensitivity flag for shortid. 1 is true, 0 is false

shortid
[out] The name of the target group.

groupid
[out] The ID of the target group.

overridedesc
[out] Overriden description.

url
[out] Help URL.

outofbandadd
[out] Detect out-of-band additions to this group and take following actions: if 'N' do nothing, if 'U' generate a request to revert change, if 'A' use idtm to undo change and generate a request to perform the change via workflow. Requires tracking of group changes at the target level to be effective.

outofbanddel
[out] Detect out-of-band deletions to this group and take following actions: if 'N' do nothing, if 'U' generate a request to revert change, if 'A' use idtm to undo change and generate a request to perform the change via workflow. Requires tracking of group changes at the target level to be effective.

location
[out] Workflow location ID.

type
[out] Workflow object type.

ownerasauth
[out] Automatically add group owners as authorizers. 0 = do not add automatically, otherwise add automatically. Ignored if target configuration does not allow groups to be moderated by owners automatically.

overrideauth
[out] Override authorization configuration. O = Use only inherited authorization, N = Do not inherit any authorization, A = Add to inherited authorization. Ignored if target configuration does not allow groups to inherit authorization configuration.

rbacenforce
[out] Whether RBAC enforcement is enabled; 1 = enabled, 0 = disabled

daction
[out] When RBAC enforcement is enabled, what resolution action will be taken for deficit violations; INHERIT=Use parent role setting, ADD=Add resource, EXCEPTION=Request exception

saction
[out] When RBAC enforcement is enabled, what resolution action will be taken for surplus violations; INHERIT=Use system default, REMOVE=Remove resource, EXCEPTION=Request exception

7.2.12 AccountGroupManage

Manages a specified group on a target.


```
struct AccountGroupManage_scalarInput
    {
    var $targetid
    var $groupid
    var $overridedesc
    var $url
    var $outofbandadd
    var $outofbanddel
    var $location
    var $type
    var $ownerasauth
    var $overrideauth
    var $rbacenforce
    var $daction
    var $saction
    };
```

Parameters

targetid
[in] The ID of the target.

groupid
[in] The ID of the managed target group.

overridedesc
[in] Overriden description.

url
[in] Help URL.

outofbandadd
[in] Detect out-of-band additions to this group and take following actions: if 'N' do nothing, if 'U' generate a request to revert change, if 'A' use idtm to undo change and generate a request to perform the change via workflow. Requires tracking of group changes at the target level to be effective.

outofbanddel
[in] Detect out-of-band deletions to this group and take following actions: if 'N' do nothing, if 'U' generate a request to revert change, if 'A' use idtm to undo change and generate a request to perform the change via workflow. Requires tracking of group changes at the target level to be effective.

location
[in] Workflow location ID.

type
[in] Workflow object type.

ownerasauth
[in] Automatically add group owners as authorizers. 0 = do not add automatically, otherwise add automatically. Ignored if target configuration does not allow groups to be moderated by owners automatically.

overrideauth

[in] Override authorization configuration. O = Use only inherited authorization, N = Do not inherit any authorization, A = Add to inherited authorization. Ignored if target configuration does not allow groups to inherit authorization configuration.

rbacenforce
[in] Whether to enable RBAC enforcement; 1 = enable, 0 = disable

daction
[in] When RBAC enforcement is enabled, what resolution action to take for deficit violations; INHERIT=Use parent role setting, ADD=Add resource, EXCEPTION=Request exception

saction
[in] When RBAC enforcement is enabled, what resolution action to take for surplus violations; INHERIT=Use system default, REMOVE=Remove resource, EXCEPTION=Request exception

7.2.13 AccountGroupMemberList

Lists members from a specified managed group.

```
struct AccountGroupMemberList_scalarInput
    {
    var $targetid
    var $groupid
    var $directonly
    };

struct AccountGroupMemberList_vectorOutput
    {
    var $targetid
    var $groupid
    var $accountid
    };
```

Parameters

targetid
[in] The ID of the target.

groupid
[in] The ID of the managed target group.

directonly
[in] Whether to list members only from the given group, or from its child groups, too. 0(zero)=list members in either the given group or its child groups, otherwise list members only in given group.

targetid
[out] The ID of the target on which the account exists.

groupid
[out] The ID of the managed target group to which the account belongs.

accountid
[out] The long ID of the member account. Is empty when listing a child group.

7.2.14 AccountGroupMemberTest

Given a group and one of a profile ID or account ID, tests whether the profile or account is a member in the group, or given two groups, tests whether the second group is a child of the first group. Profile ID, account ID, and child group cannot all be specified. Returns API_SUCCESS if true, ERR_INVALID_RESOURCE_MEMBER if false.

```
struct AccountGroupMemberTest_scalarInput
    {
    var $targetid
    var $groupid
    var $accountid
    var $userid
    var $ctargetid
    var $cgroupid
    var $directonly
    };
```

Parameters

targetid
[in] The ID of the target.

groupid
[in] The ID of the managed target group.

accountid
[in] The account ID to test for membership.

userid
[in] The profile ID to test for membership.

ctargetid
[in] The ID of the target on which the child group exists. If no value is provided, then the value of targetid is assumed.

cgroupid
[in] The ID of the managed group to test for child group membership.

directonly
[in] Whether to test for membership only with the given group, or with its child groups, too. 0(zero)=test for membership in either the given group or its child groups, otherwise test for membership only in given group.

7.2.15 AccountGroupUnmanage

Unmanages a group.

```
struct AccountGroupUnmanage_scalarInput
    {
    var $targetid
    var $groupid
    };
```

Parameters

targetid
[in] The ID of the target.

groupid
[in] The ID of the managed target group.

7.2.16 AccountIsEnabled

Query a target to discover whether a specified account is enabled or not.

```
struct AccountIsEnabled_scalarInput
    {
    var $targetid
    var $longid
    };

struct AccountIsEnabled_vectorOutput
    {
    var $enabled
    };
```

Parameters

targetid
[in] Target ID to query.

longid
[in] Account longid to query.

enabled
[out] Whether the account is enabled: 0 is disabled, anything else is enabled.

7.2.17 AccountList

Retrieves account(s) from a target system, along with attribute values.
Has four modes:

- Longid empty and no attributes specified: Lists all longids from the target system without attributes
- Longid empty and attributes specified: Lists all longids from the target system with values for specified attributes
- Longid specified and no attributes specified: Lists values for all attributes for the given longid
- Longid specified and attributes specified: Lists values for specified attributes for the given longid

```
struct AccountList_scalarInput
    {
    var $targetid
    var $longid
    var $includeinvalid
    struct AccountList_vectorInput $inputArray[]
    };

struct AccountList_vectorInput
    {
    var $attrkey
    };

struct AccountList_vectorOutput
    {
    var $longid
    var $invalid
    var $attrkey
    var $value
    };
```

Parameters

targetid
[in] The ID of the target from which accounts will be retrieved.

longid
[in] Optional. The long ID of the account to list attributes for.

includeinvalid
[in] Optional. Include invalid accounts if true.

inputArray
[in] An array of structs of type AccountList_vectorInput

attrkey
[in] Optional. The name of an attribute to retrieve for the account. If unspecified, only lists accounts (does not return attributes).

longid
[out] The long ID of the account.

invalid
[out] Whether the account is invalid or not.

attrkey
[out] The name of the attribute.

value
[out] The value of the attribute.

7.2.18 AccountPasswordSet

Sets the password on a single account for a single target system. This function does not perform any synchronization across target system groups nor user profiles. The Transaction Manager is used to perform the password reset.

```
struct AccountPasswordSet_scalarInput
    {
    var $targetid
    var $longid
    var $password
    var $expire
    var $queue
    var $inputs
    };

struct AccountPasswordSet_vectorOutput
    {
    var $result
    var $retmsg
    };
```

Parameters

targetid
[in] The target ID of the account to reset.

longid
[in] The account long ID to reset.

password
[in] The new password.

expire
[in] If set to 1, also expire the accounts when setting a new password. If set to 0, only set a new password.

queue
[in] If set to 0, the request is queued to the Transaction Manager with highest priority. If set to 1, the request is queued to the Transaction Manager with normal priority.

inputs
[in] Optional target-specific additional inputs. These are specified as a comma-separated string of name=value pairs.

result
[out] The password reset result for the account/target pair. 0 is success, 1 is failure.

retmsg
[out] A detailed message about the password reset operation.

7.2.19 AccountPasswordSync

Reset a password on an account. Depending on target group configuration, accounts that are associated to the same profile and in the same target group as the specified account may also be synchronized. The API caller needs the "change passwords" ACL to be able to run this function properly. This function does not include password strength check.

```
struct AccountPasswordSync_scalarInput
    {
    var $targetid
    var $longid
    var $password
    var $expire
    var $queue
    var $ignoretsynch
    };

struct AccountPasswordSync_vectorOutput
    {
    var $numAffected
    var $accountid
    var $targetid
    var $result
    var $retmsg
    };
```

Parameters

targetid
[in] The target ID of the account to reset.

longid
[in] The account long ID to reset.

password
[in] The new password.

expire
[in] If set to 1, also expire the accounts when setting a new password. If set to 2, only expire the accounts and do not set a new password. Otherwise, only set a new password.

queue
[in] If set to 1, send the password changes to the Password Manager service and return immediately. If set to 0, perform the password changes and wait for them to complete before returning.

ignoretsynch
[in] If set to 1, ignore target group settings and only work on input accounts. If set to 0, consider target group tsynch option.

numAffected
[out] The number of generated passwords.

accountid
[out] The account ID whose password has been changed.

targetid
[out] The target system ID where the account's password has been synched.

result
[out] The password change result on the account/target pair. 0 is success, 1 is failure

retmsg
[out] A detailed message about the operation.

7.2.20 AccountPasswordSyncTrigger

Test a supplied password against strength rules and/or synchronize the password across associated targets inside the target's group. The source account in the request is NOT reset by this function. Basically, this function behaves similar to Password Filter(trigger) calling Password Manager(idpm.exe) to request password strength check and transparent synchronization. The API caller needs the "change passwords" ACL to be able to run this function properly.

```
struct AccountPasswordSyncTrigger_scalarInput
    {
    var $operation
    var $password
    var $longid
    var $targetid
    };
```

Parameters

operation
[in] Valid options are 1: check password strength, 2: synchronize password, 3: check password strength and synchronize password.

password
[in] The password to be tested or synchronized.

longid
[in] The account whose password is being tested or synchronized.

targetid
[in] The target that the account resides on.

7.2.21 AccountRename

Record the renaming of a target system account. This only informs the product of the rename, it does not rename the target system account.

```
struct AccountRename_scalarInput
    {
    var $targetid
    var $oldlongid
    var $newlongid
    var $newshortid
    };
```

Parameters

targetid
[in] The host ID to be used.

oldlongid
[in] The old long ID of the account.

newlongid
[in] The new long ID of the account.

newshortid
[in] The new short ID of the account.

7.2.22 AccountUpdate

Update account's attributes by directly sending request to Transaction Monitor Service (idtm).
Only applies to Bravura Identity and Bravura Privilege.

```
struct AccountUpdate_scalarInput
    {
    var $accountid
    var $hostid
    struct AccountUpdate_vectorInput $inputArray[]
    };

struct AccountUpdate_vectorInput
    {
    var $attrkey
    var $value
    };
```

Parameters

accountid
[in] The long ID identifying the account.

hostid
[in] The host ID of the account to be updated.

inputArray
[in] An array of structs of type AccountUpdate_vectorInput

attrkey
[in] The key of the value to be updated.

value
[in] The value to update. Values for multi-value should be specified across a series of key/value pairs.

7.2.23 AuthorizerResourceList

Lists the resources an authorizer is statically assigned to.

```
struct AuthorizerResourceList_scalarInput
    {
    var $userid
    };

struct AuthorizerResourceList_vectorOutput
    {
    var $resourceid
    var $restype
    };
```

Parameters

userid
[in] The profile ID of the authorizer.

resourceid
[out] The ID of the resource.

restype
[out] The type of resource. This must be one of the types returned by ResourceTypesList.

7.2.24 CancelReport

Cancel a report.

```
struct CancelReport_scalarInput
    {
    var $reportid
    };
```

Parameters

reportid
[in] Unique report ID currently being run.

7.2.25 CertSavedCertifierInfoGet

Retrieves a saved certification configuration that include a given user as a reviewer.

```
struct CertSavedCertifierInfoGet_scalarInput
    {
    var $id
    };

struct CertSavedCertifierInfoGet_vectorOutput
    {
    var $savedcertid
    var $seggen
    var $certifier
    var $description
    var $userclass
    var $resmemberid
    var $membertype
    var $memberid
    var $nosgroupguid
    var $groupid
    var $hostid
    };
```

Parameters

id
[in] The user ID of the reviewer whose segments will be returned.

savedcertid
[out] Saved certification configuration ID.

seggen
[out] seggen field - type of certification method.

certifier
[out] Segment reviewer ID.

description
[out] Saved certification configuration description.

userclass
[out] Saved certification userclass ID.

resmemberid
[out] resmemberid of the resource.

membertype
[out] Contains the membertype (ROLE/TARGET/MANAGEDGROUP/SOD).

memberid
[out] Contains the memberid.

nosgroupguid
[out] guid of nosgroup.

groupid
[out] Managed group ID.

hostid
[out] Managed group host ID.

7.2.26 CertSavedCertifierUpdate

Updates the reviewer for a saved certification configuration segment.

```
struct CertSavedCertifierUpdate_scalarInput
    {
    var $savedcertid
    var $seggen
    var $certifier
    var $description
    var $userclass
    var $resmemberid
    var $membertype
    var $memberid
    var $nosgroupguid
    var $groupid
    var $hostid
    };
```

Parameters

savedcertid
[in] Saved certification configuration ID.

seggen
[in] seggen field - type of certification method.

certifier
[in] Updated segment reviewer ID.

description
[in] Saved certification configuration description.

userclass
[in] Saved certification userclass ID.

resmemberid
[in] resmemberid of the resource.

membertype
[in] Contains the membertype (ROLE/TARGET/MANAGEDGROUP/SOD).

memberid
[in] Contains the memberid.

nosgroupguid
[in] guid of nosgroup.

groupid
[in] Managed group ID.

hostid
[in] Managed group host ID.

7.2.27 CertStartCertCampaign

Start a certification campaign. The API login must be established by an admin who has the ACLs to initiate API calls.

```
struct CertStartCertCampaign_scalarInput
    {
    var $certdesc
    var $initiator
    var $emailcontent
    var $segdesc
    var $intervaldays
    var $groupmembertype
    var $includeprofiles
    var $includeprofileattributes
    var $enablecomparison
    var $attributegroup
    var $minimumcomparison
    var $lowerthreshold
    var $upperthreshold
    var $instructions
    var $signoffpasswdrequired
    var $validinterval
    var $commentrequired
    var $certmethod
    var $singlecertifier
    var $orgmanagercertifier
    var $notinuccertifier
    var $defaultcertifier
    var $ucpmatching
    struct CertStartCertCampaign_vectorInput $inputArray[]
    };
```



```
struct CertStartCertCampaign_vectorInput
    {
    var $resmembertype
    var $resourcetype
    var $rescertifier
    var $targetid
    var $groupid
    var $sodid
    var $roleid
    var $userid
    var $userclassid
    var $attrid
    var $attrtype
    var $priority
    var $remediationtype
    var $remediationpdr
    var $ucsegment
    var $uccertifier
    var $ucpsegment
    var $certifieractor
    var $userunderreviewactor
    };

struct CertStartCertCampaign_vectorOutput
    {
    var $certsig
    };
```

Parameters

certdesc
[in] Certification campaign description.

initiator
[in] The user ID of the initiator - must have ACL for certification.

emailcontent
[in] Email content to be sent.

segdesc
[in] Segmentation description.

intervaldays
[in] Number of days to wait between sending out invitation.

groupmembertype
[in] Group membership types to review. Default is 'M' if not specified. 'A' = accounts only, 'S' = subgroups only, 'M' = accounts and subgroups

includeprofiles
[in] Include profiles flag. 1 is true, 0 is false

includeprofileattributes
[in] Include profile attributes flag. 1 is true, 0 is false

enablecomparison
[in] Enable comparison by attributes. 1 is true, 0 is false

attributegroup
[in] Attribute group that collects users into peer groups.

minimumcomparison
[in] The minimum number of peers that a user must have to be compared.

lowerthreshold
[in] Lower threshold when comparing percentage of peers with same entitlement

upperthreshold
[in] Upper threshold when comparing percentage of peers with same entitlement

instructions
[in] Initiator instructions to certifier.

signoffpasswdrequired
[in] Whether password is required at segment signoff. 1 is true, 0 is false

validinterval
[in] Override validity interval for certification. -1 is do not override

commentrequired
[in] Whether comments are required for all items. 1 is true, 0 is false

certmethod
[in] Certification method: 'S' : single certifier, 'U' : user class, 'A' : entitlement authorizer, 'O' : orgchart manager, 'R' : defined relationship

singlecertifier
[in] The user ID of the reviewer for single certifier campaign

orgmanagercertifier
[in] The user ID of the reviewer for an orgchart campaign

notinuccertifier
[in] The user ID of the reviewer for a not in userclass segment of a userclass cmapaign

defaultcertifier
[in] The user ID of the default reviewer for a defined relationship campaign

ucpmatching
[in] The participants have to match which of the user classes ALL or ANY

inputArray
[in] An array of structs of type CertStartCertCampaign_vectorInput

resmembertype

[in] The resource member type of the resource being certified. TARGET MANAGEDGROUP SOD ROLE USER USERCLASS ATTR RATR ALLRESOURCE

resourcetype
[in] This field is expected if resmembertype = ALLRESOURCE otherwise empty It should contain the type of resource (TARGET, MANAGEDGROUP, ROLE, SOD) to include or if it contains ALL then all resources should be included.

rescertifier
[in] Certifier for resource (only needed if certification by resource authorizer is used)

targetid
[in] The ID of the target. This field is expected if resmembertype = TARGET or MANAGEDGROUP otherwise empty.

groupid
[in] The ID of the managed group. This field is expected if resmembertype = MANAGEDGROUP otherwise empty.

sodid
[in] The ID of an SOD rule. This field is expected if resmembertype = SOD otherwise empty.

roleid
[in] The ID of a role. This field is expected if resmembertype = ROLE otherwise empty.

userid
[in] A profile ID of the user being certified. This field is expected if resmembertype = USER otherwise empty.

userclassid
[in] A user class defining the users to be certified. This field is expected if resmembertype = USERCLASS otherwise empty.

attrid
[in] An attribute ID to be displayed while certification is run if attribute is ATTR.

attrtype
[in] Resource attribute type if attribute is RATR. Allowed values are MGRP, ROLE, TARG, SODR, MSYS, MACC, GRPAM, GRPGM

priority
[in] The priority of the associated (see previous field) attribute.

remediationtype
[in] Type of remediation: CERTACCTPDR CERTATTRPDR CERTAGRPPDR CERTCGRPPDR CERTNEWPDR CERTROLEPDR CERTUSERPDR CERTXFERPDR CERTSODPDR

remediationpdr
[in] PDR for corresponding remediation type.

ucsegment
[in] User class ID for certification by userclass

uccertifier
[in] Certifier for associated userclass for segment

ucpsegment
[in] User class ID for certification by defined relationship

certifieractor
[in] Actor for CERTIFIER

userunderreviewactor
[in] Actor for USER_UNDER_REVIEW

certsig
[out] The ID identifying the certification round.

7.2.28 CertStartSavedRound

Starts a saved certification campaign. The API login must be established by an admin who has the ACLs to manage certification and initiate API calls.

```
struct CertStartSavedRound_scalarInput
    {
    var $savedcertid
    var $certdesc
    var $emailcontent
    var $segdesc
    var $intervaldays
    };

struct CertStartSavedRound_vectorOutput
    {
    var $certsig
    };
```

Parameters

savedcertid
[in] The ID identifying the saved certification campaign.

certdesc
[in] Certification campaign description.

emailcontent
[in] Email content to be sent.

segdesc
[in] Segmentation description.

intervaldays
[in] Number of days to wait between sending out invitation.

certsig
[out] The ID identifying the certround.

7.2.29 CertStartSingleCertifierConfigRound

Start a single reviewer configuration certification campaign. The API login must be established by an admin who has the ACLs to initiate API calls.

```
struct CertStartSingleCertifierConfigRound_scalarInput
    {
    var $certdesc
    var $certifier
    var $initiator
    var $emailcontent
    var $segdesc
    var $intervaldays
    var $instructions
    struct CertStartSingleCertifierConfigRound_vectorInput $inputArray[]
    };

struct CertStartSingleCertifierConfigRound_vectorInput
    {
    var $resmembertype
    var $resourcetype
    var $sodid
    var $roleid
    };

struct CertStartSingleCertifierConfigRound_vectorOutput
    {
    var $certsig
    };
```

Parameters

certdesc
[in] Certification campaign description.

certifier
[in] The user ID of the reviewer.

initiator
[in] The user ID of the initiator - must have ACL for certification.

emailcontent
[in] Email content to be sent.

segdesc
[in] Segmentation description.

intervaldays
[in] Number of days to wait between sending out invitation.

instructions
[in] Initiator instructions to certifier.

inputArray
[in] An array of structs of type CertStartSingleCertifierConfigRound_vectorInput

resmembertype
[in] The resource member type of the resource being certified. SOD ROLE ALLRESOURCE

resourcetype
[in] This field is expected if resmembertype = ALLRESOURCE otherwise empty It should contain the type of resource (ROLE, SOD) to include or if it contains ALL then all resources should be included.

sodid
[in] The ID of an SOD rule. This field is expected if resmembertype = SOD otherwise empty.

roleid
[in] The ID of a role. This field is expected if resmembertype = ROLE otherwise empty.

certsig
[out] The ID identifying the certround.

7.2.30 CertStartSingleUserRound

Start a selected user certification campaign. The API login must be established by an admin who has the ACLs to initiate API calls.

```
struct CertStartSingleUserRound_scalarInput
    {
    var $certdesc
    var $certifier
    var $initiator
    var $emailcontent
    var $segdesc
    var $intervaldays
    var $groupmembertype
    var $includeprofiles
    var $includeprofileattributes
    var $enablecomparison
    var $attributegroup
    var $minimumcomparison
    var $lowerthreshold
    var $upperthreshold
    var $instructions
    var $signoffpasswdrequired
    var $validinterval
    var $commentrequired
```



```
    struct CertStartSingleUserRound_vectorInput $inputArray[]
    };

struct CertStartSingleUserRound_vectorInput
    {
    var $resmembertype
    var $resourcetype
    var $targetid
    var $groupid
    var $sodid
    var $roleid
    var $userid
    var $userclassid
    var $attrid
    var $attrtype
    var $priority
    var $remediationtype
    var $remediationpdr
    };

struct CertStartSingleUserRound_vectorOutput
    {
    var $certsig
    };
```

Parameters

certdesc
[in] Certification campaign description.

certifier
[in] The user ID of the reviewer.

initiator
[in] The user ID of the initiator - must have ACL for certification.

emailcontent
[in] Email content to be sent.

segdesc
[in] Segmentation description.

intervaldays
[in] Number of days to wait between sending out invitation.

groupmembertype
[in] Group membership types to review. Default is 'M' if not specified. 'A' = accounts only, 'S' = subgroups only, 'M' = accounts and subgroups

includeprofiles
[in] Include profiles flag. 1 is true, 0 is false

includeprofileattributes
[in] Include profile attributes flag. 1 is true, 0 is false

enablecomparison
[in] Enable comparison by attributes. 1 is true, 0 is false

attributegroup
[in] Attribute group that collects users into peer groups.

minimumcomparison
[in] The minimum number of peers that a user must have to be compared.

lowerthreshold
[in] Lower threshold when comparing percentage of peers with same entitlement

upperthreshold
[in] Upper threshold when comparing percentage of peers with same entitlement

instructions
[in] Initiator instructions to certifier.

signoffpasswdrequired
[in] Whether password is required at segment signoff. 1 is true, 0 is false

validinterval
[in] Override validity interval for certification. -1 is do not override

commentrequired
[in] Whether comments are required for all items. 1 is true, 0 is false

inputArray
[in] An array of structs of type CertStartSingleUserRound_vectorInput

resmembertype
[in] The resource member type of the resource being certified. TARGET MANAGEDGROUP SOD ROLE USER USERCLASS ATTR RATR ALLRESOURCE

resourcetype
[in] This field is expected if resmembertype = ALLRESOURCE otherwise empty It should contain the type of resource (TARGET, MANAGEDGROUP, ROLE, SOD) to include or if it contains ALL then all resources should be included.

targetid
[in] The ID of the target. This field is expected if resmembertype = TARGET or MANAGEDGROUP otherwise empty.

groupid
[in] The ID of the managed group. This field is expected if resmembertype = MANAGEDGROUP otherwise empty.

sodid
[in] The ID of an SOD rule. This field is expected if resmembertype = SOD otherwise empty.

roleid
[in] The ID of a role. This field is expected if resmembertype = ROLE otherwise empty.

userid
[in] A profile ID of the user being certified. This field is expected if resmembertype = USER otherwise empty.

userclassid
[in] A user class defining the users to be certified. This field is expected if resmembertype = USERCLASS otherwise empty.

attrid
[in] An attribute ID to be displayed while certification is run.

attrtype
[in] Resource attribute type if attribute is RATR. Allowed values are MGRP, ROLE, TARG, SODR, MSYS, MACC, GRPAM, GRPGM

priority
[in] The priority of the associated (see previous field) attribute.

remediationtype
[in] Type of remediation: CERTACCTPDR CERTATTRPDR CERTAGRPPDR CERTCGRPPDR CERTNEWPDR CERTROLEPDR CERTUSERPDR CERTXFERPDR CERTSODPDR

remediationpdr
[in] PDR for corresponding remediation type.

certsig
[out] The ID identifying the certround.

7.2.31 CertStartSingleUserRoundForGroups

Start a selected user U certification campaign. The API login must be established by an admin who has the ACLs to initiate API calls and start cert rounds. The groups in this round cannot be configured with API parameters. Those groups that U is a member of are automatically included in round.

```
struct CertStartSingleUserRoundForGroups_scalarInput
    {
    var $certdesc
    var $defaultreviewer
    var $certifiee
    var $initiator
    var $emailcontent
    var $segdesc
    var $intervaldays
    struct CertStartSingleUserRoundForGroups_vectorInput $inputArray[]
    };
```



```
struct CertStartSingleUserRoundForGroups_vectorInput
    {
    var $remediationtype
    var $remediationpdr
    };

struct CertStartSingleUserRoundForGroups_vectorOutput
    {
    var $certsig
    };
```

Parameters

certdesc
[in] Certification campaign description.

defaultreviewer
[in] The user ID of the default reviewer (if there is no group owner).

certifiee
[in] The ID of the user that this round is centered on.

initiator
[in] The user ID of the initiator - must have ACL for certification.

emailcontent
[in] Email content to be sent.

segdesc
[in] Segmentation description.

intervaldays
[in] Number of days to wait between sending out invitation.

inputArray
[in] An array of structs of type CertStartSingleUserRoundForGroups_vectorInput

remediationtype
[in] Type of remediation: CERTACCTPDR CERTATTRPDR CERTAGRPPDR CERTCGRPPDR CERTNEWPDR CERTROLEPDR CERTUSERPDR CERTXFERPDR CERTSODPDR

remediationpdr
[in] PDR for corresponding remediation type.

certsig
[out] The ID identifying the certround.

7.2.32 CheckoutParamsGet

Verify a checkout id is valid.


```
struct CheckoutParamsGet_scalarInput
    {
    var $checkoutid
    var $pluginid
    var $accountid
    };

struct CheckoutParamsGet_vectorOutput
    {
    var $key
    var $value
    };
```

Parameters

checkoutid
[in] Unique checkout ID to verify.

pluginid
[in] Name of the clicked disclosure plugin

accountid
[in] Name of the account to disclose

key
[out] The kvg key.

value
[out] The kvg value.

7.2.33 CheckoutStatusGet

Get the status of a checkout

```
struct CheckoutStatusGet_scalarInput
    {
    var $checkoutid
    var $accountid
    };

struct CheckoutStatusGet_vectorOutput
    {
    var $status
    };
```

Parameters

checkoutid
[in] Unique checkout IDs to verify.

accountid
[in] Name of the account to verify

status
[out] Status of the checkout. Can be checked out (O), checked in (I), pending (P), closed (C), expired (E), or revoked (R).

7.2.34 DelegationCancel

Cancel the delegations matching the search criteria.

```
struct DelegationCancel_scalarInput
    {
    var $guid
    var $primary
    var $deleg
    var $type
    };
```

Parameters

guid
[in] The delegation GUID.

primary
[in] The profile ID of the user whose responsibility will be delegated.

deleg
[in] A profile ID for the delegate.

type
[in] The type of the delegation. It could be one of BATCH, CERT, TASK or ALL.

7.2.35 DelegationList

List the delegations matching the search criteria.

```
struct DelegationList_scalarInput
    {
    var $primary
    var $deleg
    var $type
    var $activeonly
    };

struct DelegationList_vectorOutput
    {
    var $guid
    var $primary
```



```
var $lastprimary
var $deleg
var $type
var $batchid
var $segid
var $itemsig
var $startdate
var $enddate
var $delegable
var $acceptneed
var $resptime
};
```

Parameters

primary
[in] The profile ID of the user whose responsibility will be delegated.

deleg
[in] A profile ID for the delegate.

type
[in] The type of the delegation. It could be one of BATCH, CERT, TASK or ALL.

activeonly
[in] Only show active delegations?

guid
[out] The delegation GUID.

primary
[out] The profile ID of the user whose responsibility will be delegated.

lastprimary
[out] The latest primary profile name at the time.

deleg
[out] A profile ID for the delegate.

type
[out] The type of the delegation. It could be one of BATCH, CERT, TASK or ALL.

batchid
[out] The request that the delegation is restricted to.

segid
[out] The certification segment that the delegation is restricted to.

itemsig
[out] The request item (for implementation) that the delegation is restricted to.

startdate

[out] Take effect as of this date, in UTC, ISO format. This should have a valid date for every delegation.

enddate
[out] Stop delegating as of this date, in UTC, ISO format. If empty, the delegation is open-ended.

delegable
[out] Can this delegation be sub-delegated?

acceptneed
[out] Is the delegate required to accept the responsibility.

resptime
[out] When we will cause the default action to occur, if acceptneed is set to true.

7.2.36 DigitalIDGet

Retrieves Digital ID attributes

```
struct DigitalIDGet_scalarInput
    {
    var $longid
    var $targetid
    var $workstnid
    };

struct DigitalIDGet_vectorOutput
    {
    var $nosid
    var $serverid
    var $location
    var $password
    var $didfile
    var $didmd5
    var $forceflag
    };
```

Parameters

longid
[in] The account ID.

targetid
[in] The target ID.

workstnid
[in] The workstation ID.

nosid
[out] The Network Operating System ID.

serverid
[out] The server ID.

location
[out] The Digital ID file location on the workstation.

password
[out] The password used to encrypt the Digital ID file.

didfile
[out] The Digital ID file.

didmd5
[out] The MD5 checksum of Digital ID File.

forceflag
[out] Whether to prefer the Digital ID in the Bravura Security Fabric database over the workstation file. 1 is true, 0 is false

7.2.37 DigitalIDSet

Sets Digital ID attributes

```
struct DigitalIDSet_scalarInput
    {
    var $longid
    var $targetid
    var $workstnid
    var $nosid
    var $password
    var $serverid
    var $didloc
    var $didfile
    var $forceflag
    };
```

Parameters

longid
[in] The user profile ID.

targetid
[in] The target ID.

workstnid
[in] The workstation ID.

nosid
[in] The Network Operating System ID.

password
[in] The password used to encrypt the Digital ID file.

serverid
[in] The server ID.

didloc
[in] The Digital ID file location on the workstation.

didfile
[in] The Base64 encoded digital ID file.

forceflag
[in] Whether to prefer the Digital ID in the Bravura Security Fabric database over the workstation file. 1 is true, 0 is false

7.2.38 DiscoveredSystemAttrGet

Retrieves key/value attribute pair(s) from a given discovered system.

```
struct DiscoveredSystemAttrGet_scalarInput
    {
    var $id
    var $key
    };

struct DiscoveredSystemAttrGet_vectorOutput
    {
    var $key
    var $value
    };
```

Parameters

id
[in] The import id of the discovered system.

key
[in] The key to retrieve. Leave blank to retrieve all keys for this discovered system.

key
[out] The key retrieved.

value
[out] The value of the attribute.

7.2.39 DiscoveredSystemGetByAttr

Retrieves all discovered systems with the given attribute.


```
struct DiscoveredSystemGetByAttr_scalarInput
    {
    var $matchAny
    var $key1
    var $value1
    var $key2
    var $value2
    var $key3
    var $value3
    var $key4
    var $value4
    var $key5
    var $value5
    };

struct DiscoveredSystemGetByAttr_vectorOutput
    {
    var $id
    };
```

Parameters

matchAny
[in] Flag indicating whether search should match on any of the criteria (1) or all of the criteria (0).

key1
[in] The key of the attribute to search on.

value1
[in] Optional. The value of the attribute to search on. If unspecified, attribute values will be ignored during searches and the mere existence of the attribute will affect output.

key2
[in] Optional. The key of the second attribute to search on.

value2
[in] Optional. The value of the second attribute to search on. If unspecified, attribute values will be ignored during searches and the mere existence of the attribute will affect output.

key3
[in] Optional. The key of the third attribute to search on.

value3
[in] Optional. The value of the third attribute to search on. If unspecified, attribute values will be ignored during searches and the mere existence of the attribute will affect output.

key4
[in] Optional. The key of the fourth attribute to search on.

value4
[in] Optional. The value of the fourth attribute to search on. If unspecified, attribute values will be ignored during searches and the mere existence of the attribute will affect output.

key5
[in] Optional. The key of the fifth attribute to search on.

value5
[in] Optional. The value of the fifth attribute to search on. If unspecified, attribute values will be ignored during searches and the mere existence of the attribute will affect output.

id
[out] The ID of a discovered system that matches the given key/value pair.

7.2.40 EntAttrsDel

Delete entitlement attributes matching a given set of criteria.

```
struct EntAttrsDel_scalarInput
    {
    var $enttype
    var $res_id
    var $res_id2
    var $res2_id
    var $res2_id2
    struct EntAttrsDel_vectorInput $inputArray[]
    };

struct EntAttrsDel_vectorInput
    {
    var $name
    var $value
    };
```

Parameters

enttype
[in] The type of entitlement. This must be one of the types returned by EntTypesList.

res_id
[in] The ID of the resource:

- GRPAM (Group account memberships) : the host ID of the group
- GRPGM (Group group memberships) : the host ID of the parent group
- SODR (Segregation of duties rules) : the profile ID of the user
- RMBR (Role memberships) : the ID of the role

res_id2
[in] The second part of the ID of the resource:

- GRPAM (Group account memberships) : the ID of the group

- GRPGM (Group group memberships) : the ID of the parent group
- SODR (Segregation of duties rules) : not applicable
- RMBR (Role memberships) : not applicable

res2_id
[in] The ID of the resource:

- GRPAM (Group account memberships) : the host ID of the account
- GRPGM (Group group memberships) : the host ID of the child group
- SODR (Segregation of duties rules) : the ID of an SOD rule
- RMBR (Role memberships) : the profile ID of the user

res2_id2
[in] The second part of the ID of the resource:

- GRPAM (Group account memberships) : the long ID of the account
- GRPGM (Group group memberships) : the ID of the child group
- SODR (Segregation of duties rules) : not applicable
- RMBR (Role memberships) : not applicable

inputArray
[in] An array of structs of type EntAttrsDel_vectorInput

name
[in] The name of the attribute.

value
[in] The value to delete. If empty, delete all values for the specified attribute.

7.2.41 EntAttrsGet

Get a list of entitlement attributes matching a given set of criteria.

```
struct EntAttrsGet_scalarInput
    {
    var $enttype
    var $res_id
    var $res_id2
    var $res2_id
    var $res2_id2
    var $attrkey
```



```
    var $attrval
    };

struct EntAttrsGet_vectorOutput
    {
    var $enttype
    var $res_id
    var $res_id2
    var $res2_id
    var $res2_id2
    var $attrkey
    var $value
    };
```

Parameters

enttype
[in] The type of entitlement. This must be one of the types returned by EntTypesList.

res_id
[in] The ID of the resource:

- GRPAM (Group account memberships) : the host ID of the group
- GRPGM (Group group memberships) : the host ID of the parent group
- SODR (Segregation of duties rules) : the profile ID of the user
- RMBR (Role memberships) : the ID of the role

res_id2
[in] The second part of the ID of the resource:

- GRPAM (Group account memberships) : the ID of the group
- GRPGM (Group group memberships) : the ID of the parent group
- SODR (Segregation of duties rules) : not applicable
- RMBR (Role memberships) : not applicable

res2_id
[in] The ID of the resource:

- GRPAM (Group account memberships) : the host ID of the account
- GRPGM (Group group memberships) : the host ID of the child group
- SODR (Segregation of duties rules) : the ID of an SOD rule
- RMBR (Role memberships) : the profile ID of the user

res2_id2
[in] The second part of the ID of the resource:

- GRPAM (Group account memberships) : the long ID of the account
- GRPGM (Group group memberships) : the ID of the child group
- SODR (Segregation of duties rules) : not applicable
- RMBR (Role memberships) : not applicable

attrkey
[in] The name of the attribute to get.

attrval
[in] The value of the attribute to get.

enttype
[out] The type of entitlement.

res_id
[out] The ID of the resource:

- GRPAM (Group account memberships) : the host ID of the group
- GRPGM (Group group memberships) : the host ID of the parent group
- SODR (Segregation of duties rules) : the profile ID of the user
- RMBR (Role memberships) : the ID of the role

res_id2
[out] The second part of the ID of the resource:

- GRPAM (Group account memberships) : the ID of the group
- GRPGM (Group group memberships) : the ID of the parent group
- SODR (Segregation of duties rules) : not applicable
- RMBR (Role memberships) : not applicable

res2_id
[out] The ID of the resource:

- GRPAM (Group account memberships) : the host ID of the account
- GRPGM (Group group memberships) : the host ID of the child group
- SODR (Segregation of duties rules) : the ID of an SOD rule
- RMBR (Role memberships) : the profile ID of the user

res2_id2
[out] The second part of the ID of the resource:

- GRPAM (Group account memberships) : the long ID of the account
- GRPGM (Group group memberships) : the ID of the child group
- SODR (Segregation of duties rules) : not applicable
- RMBR (Role memberships) : not applicable

attrkey
[out] The name of the attribute.

value
[out] The value of the attribute.

7.2.42 EntAttrsSet

Sets the entitlement attribute values.

```
struct EntAttrsSet_scalarInput
    {
    var $enttype
    var $res_id
    var $res_id2
    var $res2_id
    var $res2_id2
    var $replace
    struct EntAttrsSet_vectorInput $inputArray[]
    };

struct EntAttrsSet_vectorInput
    {
    var $name
    var $value
    };
```

Parameters

enttype
[in] The type of entitlement. This must be one of the types returned by EntTypesList.

res_id
[in] The ID of the resource:

- GRPAM (Group account memberships) : the host ID of the group
- GRPGM (Group group memberships) : the host ID of the parent group

- SODR (Segregation of duties rules) : the profile ID of the user
- RMBR (Role memberships) : the ID of the role

res_id2
[in] The second part of the ID of the resource:

- GRPAM (Group account memberships) : the ID of the group
- GRPGM (Group group memberships) : the ID of the parent group
- SODR (Segregation of duties rules) : not applicable
- RMBR (Role memberships) : not applicable

res2_id
[in] The ID of the resource:

- GRPAM (Group account memberships) : the host ID of the account
- GRPGM (Group group memberships) : the host ID of the child group
- SODR (Segregation of duties rules) : the ID of an SOD rule
- RMBR (Role memberships) : the profile ID of the user

res2_id2
[in] The second part of the ID of the resource:

- GRPAM (Group account memberships) : the long ID of the account
- GRPGM (Group group memberships) : the ID of the child group
- SODR (Segregation of duties rules) : not applicable
- RMBR (Role memberships) : not applicable

replace
[in] If zero, values passed to the function for the attribute will be appended to the attribute. If non-zero, the values will replace those already assigned to the attribute.

inputArray
[in] An array of structs of type EntAttrsSet_vectorInput

name
[in] The name of the attribute.

value
[in] The value of the attribute.

7.2.43 EntTypesList

Lists supported entitlement types for use with EntAttrsGet, EntAttrsDel, and EntAttrsSet.

```
struct EntTypesList_scalarInput
    {
    var $unused


    };

struct EntTypesList_vectorOutput
    {
    var $name
    var $description
    };
```

Parameters

unused
[in] Not used

name
[out] The name of the entitlement type.

description
[out] The description of the entitlement type.

7.2.44 ExtDBQueryExec

Executes an SQL query against an ODBC data source, returning a result set if any.

```
struct ExtDBQueryExec_scalarInput
    {
    var $query
    var $inclnames
    struct ExtDBQueryExec_vectorInput $inputArray[]
    };

struct ExtDBQueryExec_vectorInput
    {
    var $bindvalue
    };

struct ExtDBQueryExec_vectorOutput
    {
    var $numcol
    var $colidx
    var $value
    };
```


Parameters

query
[in] The SQL query.

inclnames
[in] Set to non-zero if the first record returned in the result set should be the column names.

inputArray
[in] An array of structs of type ExtDBQueryExec_vectorInput

bindvalue
[in] A value for binding against a parameterized query. The bind values must be specified in the same order as the parameter markers that they correspond to in the query.

numcol
[out] The number of columns in the result set. Zero if the query did not produce a result set.

colidx
[out] The zero-based index of the column to which the returned value belongs.

value
[out] A result set value. Each record of the result set is returned one column value at a time.

7.2.45 GenericCheckoutDisclose

Discloses passwords provided by a generic check-out

```
struct GenericCheckoutDisclose_scalarInput
    {
    var $checkoutid
    };

struct GenericCheckoutDisclose_vectorOutput
    {
    var $acctid
    var $wstnid
    var $password
    };
```

Parameters

checkoutid
[in] A check-out ID whose password access is to be disclosed.

acctid
[out] The account ID.

wstnid
[out] The workstation ID.

password
[out] The password.

7.2.46 GenericCheckoutStatusGet

Gets the status of a check-out.

```
struct GenericCheckoutStatusGet_scalarInput
    {
    var $checkoutid
    };

struct GenericCheckoutStatusGet_vectorOutput
    {
    var $status
    };
```

Parameters

checkoutid
[in] A check-out ID to retrieve the status of.

status
[out] The status code of this check-out. Refer to the status field in the pamchkout table in *Bravura Security Fabric* Data Dictionary (**`schema-doc.pdf`**) to interpret this value.

7.2.47 GetRequestCheckoutID

Returns a check-out id given a batchid.

```
struct GetRequestCheckoutID_scalarInput
    {
    var $batchid
    };

struct GetRequestCheckoutID_vectorOutput
    {
    var $checkoutid
    };
```

Parameters

batchid
[in] A request batch ID for a password/groupset/accountset check-out.

checkoutid
[out] The check-out ID.

7.2.48 GroupParentGet

Lists parents of a specified managed group.

```
struct GroupParentGet_scalarInput
    {
    var $targetid
    var $groupid
    var $directonly
    };

struct GroupParentGet_vectorOutput
    {
    var $targetid
    var $groupid
    };
```

Parameters

targetid
[in] The ID of the target.

groupid
[in] The ID of the managed target group.

directonly
[in] Whether to list only direct parent groups of the given group, or all ancestor groups of the given group. 0(zero)=list all ancestor groups of the given group, otherwise list only direct parent groups.

targetid
[out] The ID of the target.

groupid
[out] The ID of the managed target group that is a direct parent or ancestor group.

7.2.49 ImplementerResourceList

Lists the resources the implementer is statically assigned to.

```
struct ImplementerResourceList_scalarInput
    {
    var $userid
    var $inherited
    };

struct ImplementerResourceList_vectorOutput
    {
    var $resourceid
    var $restype
    var $inherited
```



```
    };
```

Parameters

userid
[in] The profile ID of the implementer.

inherited
[in] If set to 0, only return implementers directly assigned to the resource. If non-zero, also return inherited implementers.

resourceid
[out] The ID of the resource.

restype
[out] The type of resource. This must be one of the types returned by ResourceTypesList.

inherited
[out] Whether the implementer was inherited.

7.2.50 InstanceDBConfigGet

Gets the database configuration for the local node.

```
struct InstanceDBConfigGet_scalarInput
    {
    var $unused


    };

struct InstanceDBConfigGet_vectorOutput
    {
    var $authmode
    var $dbuser
    var $dbserver
    var $dbname
    };
```

Parameters

unused
[in] Not used

authmode
[out] Database authentication mode

dbuser
[out] Account used to connect to database

dbserver

[out] Database server and optional instance name.

dbname
[out] The name of the database.

7.2.51 InstanceDBConfigSet

Sets database configuration values.

```
struct InstanceDBConfigSet_scalarInput
    {
    var $dbuser
    var $dbpwd
    };
```

Parameters

dbuser
[in] The user to use to connect to the database.

dbpwd
[in] The password to use to connect to the database.

7.2.52 InstanceGet

Returns information about the instance on which the API server is running (not the instance, if any, from which the API was initiated).

```
struct InstanceGet_scalarInput
    {
    var $unused


    };

struct InstanceGet_vectorOutput
    {
    var $instance
    var $address
    var $guid
    };
```

Parameters

unused
[in] Not used

instance
[out] Name of instance.

address
[out] Address of instance.

guid
[out] Globally unique identifier of the instance.

7.2.53 InstanceList

Returns a list of all instances in the configuration. Replicating and shared schema peers are returned. The local instance will be included in the result set.

```
struct InstanceList_scalarInput
    {
    var $unused


    };

struct InstanceList_vectorOutput
    {
    var $address
    var $port
    var $active
    };
```

Parameters

unused
[in] Not used

address
[out] Address of instance.

port
[out] Port number iddb service is listening on.

active
[out] A boolean that indicates if the instance is alive. Value is 1 if the instance is alive. 0 otherwise.

7.2.54 InstanceProxyList

Returns a list of all proxies in the configuration.

```
struct InstanceProxyList_scalarInput
    {
    var $unused


    };
```



```
struct InstanceProxyList_vectorOutput
    {
    var $address
    var $port
    };
```

Parameters

unused
[in] Not used

address
[out] Address of proxy.

port
[out] Port number proxy service is listening on.

7.2.55 InstanceRunUtility

Runs instance utilities asynchronously `struct InstanceRunUtility_scalarInput`

```
    {
    var $command
    var $reserved1
    var $reserved2
    };
```

Parameters

command
[in] The command to run. Allowed commands are:

- updinst-extdb: updinst.exe -extdb
- psupdate-listdiscovered-full: psupdate.exe -list -loaddb -targetdiscovered -targetmanual -import

reserved1
[in] Place reserved for future use.

reserved2
[in] Place reserved for future use.

7.2.56 InstanceStatus

Returns a key-value list that describes various elements of an instance's status.

```
struct InstanceStatus_scalarInput
    {
    var $address
```



```
    var $port
    };

struct InstanceStatus_vectorOutput
    {
    var $key
    var $value
    };
```

Parameters

address
[in] Address of instance.

port
[in] Port number iddb service is listening on.

key
[out] The status metric's name.

value
[out] The status metric's value.

7.2.57 KMKeyGetByAccount

Tries to fetch the password of a managed account on a specified resource that is managed by Bravura Privilege. The password expiry time is returned as well. It is the user's responsibility not to use the password for any purpose when it is almost expired. The password can be changed around its expiry time by Bravura Privilege dynamically.

```
struct KMKeyGetByAccount_scalarInput
    {
    var $resourceid
    var $accountid
    var $ip
    var $host
    };

struct KMKeyGetByAccount_vectorOutput
    {
    var $password
    var $expirytime
    };
```

Parameters

resourceid
[in] The resource ID where the managed account ID is hosted.

accountid

[in] The long ID of the managed account in the resource.

ip
[in] The IP address of the resource where the managed account is hosted.

host
[in] The DNS hostname of the resource where the managed account is hosted.

password
[out] The password of the account ID in the resource.

expirytime
[out] The approximate ISO-formatted UTC expiry time of this password.

7.2.58 LargeCredentialGet

Retrieves large credential file in chunks

```
struct LargeCredentialGet_scalarInput
    {
    var $targetid
    var $longid
    var $attrkey
    };

struct LargeCredentialGet_vectorOutput
    {
    var $filename
    var $filechunknum
    var $filechunk
    var $md5CheckSum
    };
```

Parameters

targetid
[in] The target ID.

longid
[in] The account (file) ID.

attrkey
[in] The account file attribute key.

filename
[out] The file name.

filechunknum
[out] The file content chunk number.

filechunk

[out] The file content chunk.

md5CheckSum
[out] The MD5 checksum of whole file. Only the last record will have this populated.

7.2.59 LicenseCountsList

Retrieves license compliance object counts.

```
struct LicenseCountsList_scalarInput
    {
    var $force
    };

struct LicenseCountsList_vectorOutput
    {
    var $objname
    var $currentcount
    var $maxcount
    var $tag
    var $severity
    };
```

Parameters

force
[in] Recompute the license status rather than just retrieving the cached status. 0 = use the cached status, otherwise recompute

objname
[out] The name of the object being counted. This is empty for global license errors, such as an expired or invalid license.

currentcount
[out] The current number of these objects in use

maxcount
[out] The maximum number of these objects permitted by the license

tag
[out] Language tag describing the current licensing status, if any

severity
[out] Severity code for the current licensing status. 0 = OK, 1 = imminent, 2 = reached, 3 = exceeded

7.2.60 LicensedProductList

Lists the products licensed for the current instance.


```
struct LicensedProductList_scalarInput
    {
    var $unused


    };

struct LicensedProductList_vectorOutput
    {
    var $name
    };
```

Parameters

unused
[in] Not used

name
[out] Nickname of licensed product. Maybe be one of the following: idarchive, pamlite, idaccess, idcertify, idtrack, iddiscover, idorg, idsynch, ivr, lite, other, psynch, sso, idrole

7.2.61 LocalCounterGet

Atomically gets the current value of the local counter. Note that the value can change after this function returns.

```
struct LocalCounterGet_scalarInput
    {
    var $name
    };

struct LocalCounterGet_vectorOutput
    {
    var $value
    };
```

Parameters

name
[in] Name of the local counter.

value
[out] Current value of the local counter.

7.2.62 LocalCounterGetAndIncrement

Atomically gets and increments the local counter. If the local counter was never set, the local counter will automatically be created and initialized to 0.


```
struct LocalCounterGetAndIncrement_scalarInput
    {
    var $name
    };

struct LocalCounterGetAndIncrement_vectorOutput
    {
    var $value
    };
```

Parameters

name
[in] Name of the local counter.

value
[out] Current value of the local counter.

7.2.63 LocalCounterSet

Atomically sets the local counter to the given value. Note that first use of an uninitialized LocalCounter will automatically set the value to start at 0.

```
struct LocalCounterSet_scalarInput
    {
    var $name
    var $value
    };
```

Parameters

name
[in] Name of the local counter.

value
[in] Value to set the local counter to.

7.2.64 LogEvent

Logs an event.

```
struct LogEvent_scalarInput
    {
    var $starttime
    var $endtime
    var $eventtype
    var $userid
    var $hostid
    var $hostname
```



```
    var $longid
    var $resultcode
    var $result
    };
```

Parameters

starttime
[in] The ISO-formatted UTC starting time of the event. Leave this string empty to indicate the current time.

endtime
[in] The ISO-formatted UTC end time of the event. Leave this string empty to indicate the current time.

eventtype
[in] The event type that is to be logged.

userid
[in] The user ID to be used in the event log.

hostid
[in] The host ID to be used in the event log.

hostname
[in] The host name to be used in the event log.

longid
[in] The long ID to be used in the event log.

resultcode
[in] The result code to log. 0 is success, -1 is failure.

result
[in] The result message to log.

7.2.65 Login

Establishes a session on the server with the given ID and password. The login process uses the authentication list configured through the *Manage the system* (PSA) module. The session is available (and necessary) for future API calls.

```
struct Login_scalarInput
    {
    var $userid
    var $password
    var $isadmin
    var $options
    };
```

Parameters

userid

[in] The profile ID of the user who is connecting.

password
[in] The password used to authenticate the login process.

isadmin
[in] Reserved.

options
[in] (Optional) Additional login parameters, separated by semi-colons. Currently, these are the only supported parameters:

- AuthConsoleUser = <consoleuserid>: Validate the password against a console user instead of the user's account on a target system. The resulting API session will still have userid as the logged-in user.

7.2.66 LoginEx

Establishes a session on the server with the given ID and password of a console user. This console user should have at least the OTP Caller ACL. The session is available (and necessary) for future API calls. The returned new password is used for future Login or LoginEx calls.

```
struct LoginEx_scalarInput
    {
    var $aid
    var $apw
    };

struct LoginEx_vectorOutput
    {
    var $newapw
    };
```

Parameters

aid
[in] A console user ID.

apw
[in] The console user's password.

newapw
[out] The new password.

7.2.67 Logout

Closes the connection to the server and ends the session. A new session must be established using Login or LoginEx before further API calls can be made.


```
struct Logout_scalarInput
    {
    var $unused


    };
```

Parameters

unused
[in] Not used

7.2.68 MAQCreate

Creates an account set. Requires the "Modify properties for this policy" ACL

```
struct MAQCreate_scalarInput
    {
    var $maqid
    var $description
    var $policyid
    };
```

Parameters

maqid
[in] The ID of the new account set.

description
[in] The description of the new account set.

policyid
[in] The managed system policy this account set may draw accounts from.

7.2.69 MAQExplicitUserAdd

Adds an explicit user to an account set. Requires the "Modify properties for this policy" ACL

```
struct MAQExplicitUserAdd_scalarInput
    {
    var $maqid
    var $systemid
    var $accountid
    };
```

Parameters

maqid
[in] The account set to add to.

systemid
[in] The ID of the managed system the account to add exists on.

accountid
[in] The ID of the account to add.

7.2.70 MAQExplicitUserDelete

Removes an explicit user from an account set. Requires the "Modify properties for this policy" ACL

```
struct MAQExplicitUserDelete_scalarInput
    {
    var $maqid
    var $systemid
    var $accountid
    };
```

Parameters

maqid
[in] The account set to remove from.

systemid
[in] The ID of the managed system the account to remove exists on.

accountid
[in] The ID of the account to remove.

7.2.71 MAQMemberAccountList

Retrieves all member accounts of the given account set. Requires the "View properties for this policy" ACL

```
struct MAQMemberAccountList_scalarInput
    {
    var $maqid
    };

struct MAQMemberAccountList_vectorOutput
    {
    var $accountid
    var $wstnid
    };
```

Parameters

maqid
[in] The ID of the account set.

accountid

[out] The ID of the retrieved managed account.

wstnid
[out] The ID of the managed system.

7.2.72 ManagedAccountAssociateWithService

Associates a subscriber with a managed account. Requires the "Manage systems on this policy" Bravura Privilege ACL for a managed system policy that the managed system is a member of. Requires the "Pre-approved check-out of managed accounts" Bravura Privilege ACL for a managed system policy that the managed account is a member of.

```
struct ManagedAccountAssociateWithService_scalarInput
    {
    var $systemid
    var $longid
    var $accountid
    var $subscriberhostid
    var $subscriberid
    var $subscribertype
    var $notifytarget
    var $position
    var $restart
    };
```

Parameters

systemid
[in] The ID of the managed system on which the account the subscriber will subscribe to exists.

longid
[in] The account the subscriber will subscribe to.

accountid
[in] The account ID as specified in the subscriber's configuration on its host.

subscriberhostid
[in] The ID of the target on which the subscriber exists.

subscriberid
[in] A URI for the subscriber.

subscribertype
[in] The type of the subscriber. Can be one of:

- COM: DCOM object
- CUS: Custom
- GRP: User group
- IIS: IIS directory
- SVC: Service

- TAS: Scheduled task

notifytarget
[in] Whether to update the new subscriber immediately. 1 means yes, otherwise no.

position
[in] Optional. Position parameter to pass to agent if notifytarget = 1. Can be one of: pre, post.

restart
[in] Optional. Restart parameter to pass to agent if notifytarget = 1. 1 means yes, otherwise no. Default no.

7.2.73 ManagedAccountDetachFromService

Detach a managed account from a service. Requires the "Manage systems on this policy" Bravura Privilege ACL for a managed system policy that the managed system is a member of. If the managed system is orphaned, the "Manage orphan managed systems" administrator ACL is required

```
struct ManagedAccountDetachFromService_scalarInput
    {
    var $subscriberhost
    var $servicetype
    var $servicename
    var $targetid
    var $accountid
    };
```

Parameters

subscriberhost
[in] The ID of the managed system.

servicetype
[in] The type of the service.

servicename
[in] The name of the service.

targetid
[in] The target ID of the managed account

accountid
[in] The login id of the managed account

7.2.74 ManagedAccountImportPassword

Imports a password, directly setting it in the database and not running any agents or triggering pull-mode override. Non-vault accounts must exist and be attached to a policy but need not have a nonempty pass-

word. Vault accounts will be created if they do not exist already. Requires the "Randomize/override password of managed accounts" Bravura Privilege ACL for the policy of the managed account, if the account already exists. If the account is a vault account that must be created, the "Modify properties for this policy" Bravura Privilege ACL on the vault system's primary policy is also required.

```
struct ManagedAccountImportPassword_scalarInput
    {
    var $systemid
    var $accountid
    var $password
    var $checkstrength
    };
```

Parameters

systemid
[in] The ID of the managed system.

accountid
[in] The ID of the account to import.

password
[in] The password to import for the account.

checkstrength
[in] Whether to validate the password against its policy's password policy. 1 means yes, otherwise no.

7.2.75 ManagedAccountOverridePassword

Overrides an existing managed account in a managed system or, if the system is a vault, creates a new account if one doesn't already exist. Requires the "Randomize/override password of managed accounts" Bravura Privilege ACL for the policy of the managed account, if the account already exists. If the account is a vault account that must be created, the "Modify properties for this policy" Bravura Privilege ACL on the vault system's primary policy is also required. This function returns exactly one workid for managed accounts where it is required, otherwise it does not return any data.

```
struct ManagedAccountOverridePassword_scalarInput
    {
    var $id
    var $accountid
    var $password
    };

struct ManagedAccountOverridePassword_vectorOutput
    {
    var $workid
    };
```

Parameters

id
[in] The ID of the managed system.

accountid
[in] The ID of the account to override or create.

password
[in] The new password for the account.

workid
[out] A unique identifier that can be used to query the results of this randomization using the PAMBatchResultsList function. Only returned for push-mode managed accounts.

7.2.76 ManagedAccountPasswordStatusGet

Retrieves the status of a managed account password.

```
struct ManagedAccountPasswordStatusGet_scalarInput
    {
    var $systemid
    var $accountid
    };

struct ManagedAccountPasswordStatusGet_vectorOutput
    {
    var $status
    };
```

Parameters

systemid
[in] The ID of the managed system.

accountid
[in] The account ID.

status
[out] The status of the managed account's password.

7.2.77 ManagedAccountPolicyIDGet

Retrieves the managed system policy id of a managed account.

```
struct ManagedAccountPolicyIDGet_scalarInput
    {
    var $systemid
    var $accountid
    };
```



```
struct ManagedAccountPolicyIDGet_vectorOutput
    {
    var $policyid
    };
```

Parameters

systemid
[in] The ID of the managed system.

accountid
[in] The account ID.

policyid
[out] The managed system policy id the account is in.

7.2.78 ManagedAccountRandomizePassword

Randomizes the password on a single account for a single managed system. This function does not perform any synchronization across target system groups nor user profiles. Requires the "Randomize/override password of managed accounts" Bravura Privilege ACL for a managed system policy of which the managed account is a member.

```
struct ManagedAccountRandomizePassword_scalarInput
    {
    var $systemid
    var $longid
    };

struct ManagedAccountRandomizePassword_vectorOutput
    {
    var $workid
    };
```

Parameters

systemid
[in] The managed system ID of the account to reset.

longid
[in] The account long ID to reset.

workid
[out] A unique identifier that can be used to query the results of this randomization using the PAMBatchResultsList function.

7.2.79 ManagedAccountStatusGet

Retrieves whether an account is managed, discovered, or unknown to Bravura Privilege.


```
struct ManagedAccountStatusGet_scalarInput
    {
    var $systemid
    var $accountid
    };

struct ManagedAccountStatusGet_vectorOutput
    {
    var $status
    };
```

Parameters

systemid
[in] The ID of the managed system.

accountid
[in] The account ID.

status
[out] The status of the account. M means managed, D means discovered but not managed, U means unknown to Bravura Privilege.

7.2.80 ManagedAccountSubscriberList

List all loaded subscribers.

```
struct ManagedAccountSubscriberList_scalarInput
    {
    var $unused


    };

struct ManagedAccountSubscriberList_vectorOutput
    {
    var $id
    var $targetid
    var $targetdesc
    var $accountid
    var $subscribertype
    var $objectname
    var $objectsid
    };
```

Parameters

unused
[in] Not used

id
[out] The id of the subscriber.

targetid
[out] The target of the managed account.

targetdesc
[out] The description of the target.

accountid
[out] The managed account ID.

subscribertype
[out] The subscriber type. Possible values are SVC for managing services, IIS for managing IIS, TAS for managing scheduled tasks, GRP for managing group members, COM for managing DCOM, and CUS for managing custom accounts.

objectname
[out] The managed object name, such as the service name, the IIS site name, the scheduled task name, the group name or the custom object name.

objectsid
[out] The SID of the object.

7.2.81 ManagedSystemAttrAdd

Adds a key/value pair to a given managed system. The key defined must match a resource attribute defined in a resource attribute group of type "Managed systems". Requires read and write permissions to the resource attribute group. Requires the "Manage systems on this policy" Bravura Privilege ACL for a managed system policy that the managed system is a member of. If the managed system is orphaned the "Manage orphan managed systems" administrator ACL is required.

```
struct ManagedSystemAttrAdd_scalarInput
    {
    var $id
    var $key
    var $value
    var $seqno
    };
```

Parameters

id
[in] The ID of the managed system.

key
[in] The key to add.

value
[in] The value to add.

seqno
[in] The sequence number of a distinguished value for the same key. 0 (or omitted) denotes single value attribute, positive integer denotes multi value attribute.

7.2.82 ManagedSystemAttrDelete

Deletes an attribute key/value pair from a managed system. The key defined must match a resource attribute defined in a resource attribute group of type "Managed systems". Requires read and write permissions to the resource attribute group. Requires the "Manage systems on this policy" Bravura Privilege ACL for a managed system policy that the managed system is a member of. If the managed system is orphaned, the "Manage orphan managed systems" administrator ACL is required.

```
struct ManagedSystemAttrDelete_scalarInput
    {
    var $id
    var $key
    };
```

Parameters

id
[in] The ID of the managed system.

key
[in] The key to delete.

7.2.83 ManagedSystemAttrGet

Retrieves a key/value attribute pair from a given managed system. The key defined must match a resource attribute defined in a resource attribute group of type "Managed systems". Requires read and write permissions to the resource attribute group.

```
struct ManagedSystemAttrGet_scalarInput
    {
    var $id
    var $key
    };

struct ManagedSystemAttrGet_vectorOutput
    {
    var $key
    var $value
    var $seqno
    };
```

Parameters

id

[in] The ID of the managed system.

key
[in] The key to retrieve. Leave blank to retrieve all keys for this workstation.

key
[out] The key retrieved.

value
[out] The value of the attribute.

seqno
[out] The sequence number of the retrieved value (for multi-valued attributes).

7.2.84 ManagedSystemCheckoutList

Retrieves checkouts for managed systems and accounts. Note: At least one of the following parameters, wstnid, acctid or recipientid, should be provided, in order to avoid returning all checkouts for all managed systems and accounts.

```
struct ManagedSystemCheckoutList_scalarInput
    {
    var $wstnid
    var $acctid
    var $status
    var $recipientid
    };

struct ManagedSystemCheckoutList_vectorOutput
    {
    var $wstnid
    var $acctid
    var $status
    var $batchid
    var $checkouttime
    var $expiry
    var $checkintime
    var $recipientid
    };
```

Parameters

wstnid
[in] Optional. The id of the managed system to list checkouts for.

acctid
[in] Optional. The id of the managed account to list checkouts for.

status
[in] Optional. Status of the checkout. Can be checked out (O), checked in (I), pending (P), closed (C),

expired (E), or revoked (R).

recipientid
[in] Optional. Recipient of the checkout.

wstnid
[out] The id of the managed system.

acctid
[out] The id of the managed account.

status
[out] Status of the checkout. Can be checked out (O), checked in (I), pending (P), closed (C), expired (E), or revoked (R).

batchid
[out] The batch ID for the workflow request that caused this checkout, if applicable.

checkouttime
[out] The time the password was checked out.

expiry
[out] The time the checkout is scheduled to expire.

checkintime
[out] The time the password was checked in, if applicable.

recipientid
[out] The recipient of the checkout.

7.2.85 ManagedSystemDelete

Flags a managed system as deleted. Deleted managed systems are processed via the configured deletion policy.

```
struct ManagedSystemDelete_scalarInput
    {
    var $id
    };
```

Parameters

id
[in] The ID of the managed system to delete.

7.2.86 ManagedSystemDuplicateList

List all managed systems that share the specified attributes. For multi-valued attributes, the attributes will be considered identical as long as one of the values matches.


```
struct ManagedSystemDuplicateList_scalarInput
    {
    var $attribute1
    var $attribute2
    var $attribute3
    var $attribute4
    var $attribute5
    };

struct ManagedSystemDuplicateList_vectorOutput
    {
    var $id
    var $time
    var $attribute1
    var $attribute2
    var $attribute3
    var $attribute4
    var $attribute5
    };
```

Parameters

attribute1
[in] The name of the first attribute to use in the identification of duplicates.

attribute2
[in] The name of an optional second attribute to use in the identification of duplicates.

attribute3
[in] The name of an optional third attribute to use in the identification of duplicates.

attribute4
[in] The name of an optional fourth attribute to use in the identification of duplicates.

attribute5
[in] The name of an optional fifth attribute to use in the identification of duplicates.

id
[out] The id of a managed system that was found to be a duplicate.

time
[out] The last time the managed system was contacted.

attribute1
[out] If attribute1 was specified in the input, this will be populated with the value of the attribute for the specified managed system.

attribute2
[out] If attribute2 was specified in the input, this will be populated with the value of the attribute for the specified managed system.

attribute3

[out] If attribute3 was specified in the input, this will be populated with the value of the attribute for the specified managed system.

attribute4
[out] If attribute4 was specified in the input, this will be populated with the value of the attribute for the specified managed system.

attribute5
[out] If attribute5 was specified in the input, this will be populated with the value of the attribute for the specified managed system.

7.2.87 ManagedSystemGetByAttr

Retrieves all managed systems with the given attribute.

```
struct ManagedSystemGetByAttr_scalarInput
    {
    var $key
    var $value
    };

struct ManagedSystemGetByAttr_vectorOutput
    {
    var $id
    };
```

Parameters

key
[in] The key of the attribute to search on.

value
[in] Optional. The value of the attribute to search on. If unspecified, attribute values will be ignored during searches.

id
[out] The ID of a managed system that matches the given key/value pair.

7.2.88 ManagedSystemGuidToId

Retrieves the managed system ID of a managed system internal ID.

```
struct ManagedSystemGuidToId_scalarInput
    {
    var $systemid
    };

struct ManagedSystemGuidToId_vectorOutput
    {
```



```
    var $id
    };
```

Parameters

systemid
[in] The internal ID of the managed system.

id
[out] The ID of a managed system that matches the given internal ID.

7.2.89 ManagedSystemList

List managed systems.

This function accepts and/or returns platform type ID values. See Platform Type ID Values for the list of possible values.

```
struct ManagedSystemList_scalarInput
    {
    var $workstnid
    var $description
    var $ip
    var $status
    var $ostype
    var $wstntype
    var $osversion
    var $logonuser
    var $url
    var $platform
    var $deleted
    var $audittime
    };

struct ManagedSystemList_vectorOutput
    {
    var $workstnid
    var $description
    var $ip
    var $status
    var $ostype
    var $wstntype
    var $osversion
    var $logonuser
    var $url
    var $platform
    var $deleted
    var $audittime
    };
```


Parameters

workstnid
[in] ID of the workstation to locate. Can be empty.

description
[in] ID of the workstation to locate. Can be empty.

ip
[in] IP address of the workstation to locate. Can be empty.

status
[in] Status of the workstation to locate. Can be Managed (M), Unmanaged (U), Inactive (I) or empty.

ostype
[in] Type of the workstation to locate. Can be WINNT, WINNTS, WINNTDC, WIN2000, WIN2000S, WIN2000DC, WINXP, WIN2003S, WIN2003DC, UNIX, LINUX, IDM, SQL, ORACLE, WINVISTA, LONGHORN, WIN2008S, WIN2008DC, WINDOWS7, WINDOWS2008R2S, WIN2008R2DC.

wstntype
[in] Type of the workstation to locate. Can be Workstation (W), Server (S), or empty.

osversion
[in] Workstation/application version string.

logonuser
[in] The ID of the last logged on user.

url
[in] Optional URL with resource information.

platform
[in] Optional platform type to filter on.

deleted
[in] If not specified, list all active managed systems. If specified, only return systems that were deleted before the audittime date.

audittime
[in] The last time the system was updated.

workstnid
[out] ID of the workstation to locate. Can be empty.

description
[out] ID of the workstation to locate. Can be empty.

ip
[out] IP address of the workstation to locate. Can be empty.

status
[out] Status of the workstation to locate. Can be Managed (M), Unmanaged (U), Inactive (I) or empty.

ostype
[out] Type of the workstation to locate. Can be WINNT, WINNTS, WINNTDC, WIN2000, WIN2000S, WIN2000DC, WINXP, WIN2003S, WIN2003DC, UNIX, LINUX, IDM, SQL, ORACLE, WINVISTA, LONGHORN, WIN2008S, WIN2008DC, WINDOWS7, WINDOWS2008R2S, WIN2008R2DC.

wstntype
[out] Type of the workstation to locate. Can be Workstation (W), Server (S), or empty.

osversion
[out] Workstation/application version string.

logonuser
[out] The ID of the last logged on user.

url
[out] Optional URL with resource information.

platform
[out] Optional platform type to filter on.

deleted
[out] Whether or not the system has been flagged for deletion.

audittime
[out] The last time the system was updated.

7.2.90 ManagedSystemMSPAccountCountList

Retrieves actively managed account counts on a managed system grouped by managed system policy
Requires the "View properties for this policy" ACL

```
struct ManagedSystemMSPAccountCountList_scalarInput
    {
    var $systemid
    var $includeinvalid
    };

struct ManagedSystemMSPAccountCountList_vectorOutput
    {
    var $policyid
    var $count
    };
```

Parameters

systemid
[in] The ID of the managed system.

includeinvalid
[in] Optional. Include invalid accounts if true.

policyid
[out] The ID of the policy.

count
[out] The number of accounts belonging to the managed system in the managed system policy.

7.2.91 ManagedSystemManagedAccountList

Retrieves all actively managed accounts on a managed system
Requires the "View properties for this policy" ACL

```
struct ManagedSystemManagedAccountList_scalarInput
    {
    var $systemid
    };

struct ManagedSystemManagedAccountList_vectorOutput
    {
    var $accountid
    };
```

Parameters

systemid
[in] The ID of the managed system.

accountid
[out] The ID of the retrieved managed account.

7.2.92 ManagedSystemPoliciesList

Retrieves all managed system policies.

```
struct ManagedSystemPoliciesList_scalarInput
    {
    var $unused


    };

struct ManagedSystemPoliciesList_vectorOutput
    {
    var $id
    var $owner
    var $description
    var $type
    };
```

Parameters

unused
[in] Not used

id
[out] The ID of the managed system policy.

owner
[out] The owner of the managed system policy.

description
[out] The description of the managed system policy.

type
[out] The type of the managed system policy. H is push, L is pull, M is vault/manual.

7.2.93 ManagedSystemPolicyGet

Retrieves the managed system policy with given ID.

```
struct ManagedSystemPolicyGet_scalarInput
    {
    var $id
    };

struct ManagedSystemPolicyGet_vectorOutput
    {
    var $id
    var $description
    var $type
    };
```

Parameters

id
[in] The ID of the managed system policy.

id
[out] The ID of the managed system policy.

description
[out] The description of the managed system policy.

type
[out] The management type of the retrieved policy. H means push, L means pull, M means vault.

7.2.94 ManagedSystemPolicyList

Gets all managed system policies attached to a given managed system ID.


```
struct ManagedSystemPolicyList_scalarInput
    {
    var $id
    };

struct ManagedSystemPolicyList_vectorOutput
    {
    var $id
    var $description
    var $primary
    var $type
    };
```

Parameters

id
[in] The ID of the managed system.

id
[out] The ID of the managed system policy.

description
[out] The description of the managed system policy.

primary
[out] Whether this policy is the primary policy for the given workstation. 1 is primary, 0 is not.

type
[out] The management type of the retrieved policy. H means push, L means pull, M means vault.

7.2.95 ManagedSystemRestore

Removes the deleted flag from a managed system. This restores it to an actively managed system.

```
struct ManagedSystemRestore_scalarInput
    {
    var $id
    };
```

Parameters

id
[in] The ID of the managed system to restore.

7.2.96 MobilePushNotification

Send push notification to mobile devices. Multiple profiles may be specified as recipients.


```
struct MobilePushNotification_scalarInput
    {
    var $syncmode
    var $title
    var $body
    var $badge
    var $sound
    var $urgent
    var $ttlsecs
    struct MobilePushNotification_vectorInput $inputArray[]
    };

struct MobilePushNotification_vectorInput
    {
    var $userid
    };
```

Parameters

syncmode
[in] Whether to prefer synchronous push notifications. 1 is true, 0 is false.

title
[in] Title of the notification.

body
[in] Message body of the notification.

badge
[in] Badge to display with the notification.

sound
[in] Sound to play for the notification.

urgent
[in] Whether to send an urgent message. 1 is true, 0 is false (It is ignored when sending APNs).

ttlsecs
[in] The total time to live of the message, in seconds.

inputArray
[in] An array of structs of type MobilePushNotification_vectorInput

userid
[in] A profile to send notifications to.

7.2.97 OTPAPIUserCreate

Creates an OTP API caller.


```
struct OTPAPIUserCreate_scalarInput
    {
    var $id
    var $name
    var $password
    var $passwordexpires
    var $cidr
    var $intervalhours
    };
```

Parameters

id
[in] The ID of the new OTP API caller.

name
[in] The name of the new OTP API caller.

password
[in] The OTP API caller's initial password.

passwordexpires
[in] Whether the new OTP API caller's password can expire. 0 is never expire, 1 is can expire

cidr
[in] The CIDR bitmask describing the block of IP addresses that will be permitted to log in as this OTP API caller. Leave blank to set to the default "127.0.0.1/32"

intervalhours
[in] The duration, in hours, for which this user's password should remain valid before being randomized on their next login. Set to 0 to randomize on every log in.

7.2.98 OTPAPIUserDelete

Deletes an OTP API caller.

```
struct OTPAPIUserDelete_scalarInput
    {
    var $id
    };
```

Parameters

id
[in] The ID of the OTP API caller to delete.

7.2.99 OTPAPIUserPasswordUpdate

Updates the password of an OTP API caller.


```
struct OTPAPIUserPasswordUpdate_scalarInput
    {
    var $id
    var $password
    };
```

Parameters

id
[in] The ID of the OTP API caller.

password
[in] The OTP API caller's new password.

7.2.100 OnboardManagedSystem

Onboards a managed system.
Requires the "Manage orphan managed systems" administrator ACL.

```
struct OnboardManagedSystem_scalarInput
    {
    var $systemid
    var $description
    var $address
    var $comptype
    var $templatetargetid
    var $policyid
    var $connectionloginid
    var $connectionpassword
    var $proxy
    };
```

Parameters

systemid
[in] The ID of the managed system to onboard.

description
[in] The description of the retrieved workstation.

address
[in] The network address of the target system.

comptype
[in] Type of the computer. Can be W (Workstation), S (Server), or empty (defaults to Workstation).

templatetargetid
[in] The ID of a template target system.

policyid

[in] The ID of the managed system policy.

connectionloginid
[in] The connecton login account ID.

connectionpassword
[in] The connecton login account password.

proxy
[in] List of proxies to use for the target system, if any.

7.2.101 OrgManagerList

Retrieves a list of managers for the organization.

```
struct OrgManagerList_scalarInput
    {
    var $clevel
    };

struct OrgManagerList_vectorOutput
    {
    var $id
    var $mlevel
    var $clevel
    var $reportto
    };
```

Parameters

clevel
[in] The cLevel of the to-be listed managers. If -1, managers from all cLevels will be listed.

id
[out] User ID.

mlevel
[out] Manually-entered level.

clevel
[out] Calculated level.

reportto
[out] Higher ranked user to whom the user reports.

7.2.102 OrgReportsTo

Tests whether a given user directly or indirectly reports to another user.


```
struct OrgReportsTo_scalarInput
    {
    var $managerid
    var $subordinateid
    var $directonly
    };

struct OrgReportsTo_vectorOutput
    {
    var $reportsto
    };
```

Parameters

managerid
[in] The profile ID of the proposed manager.

subordinateid
[in] The profile ID of the proposed subordinate.

directonly
[in] If non-zero, the condition is successful if and only if the manager is a direct manager. If zero, the condition is successful if the manager is a direct manager or manager though the management chain.

reportsto
[out] Non-zero for true condition, zero for false condition.

7.2.103 OrgUserGet

Retrieves the OrgChart attributes for a user.

```
struct OrgUserGet_scalarInput
    {
    var $id
    };

struct OrgUserGet_vectorOutput
    {
    var $id
    var $mlevel
    var $clevel
    var $reportto
    var $ismanager
    };
```

Parameters

id
[in] The profile ID of the user whose OrgChart attributes will be retrieved.

id
[out] User ID.

mlevel
[out] Manually-entered level.

clevel
[out] Calculated level.

reportto
[out] Higher ranked user to whom the user reports.

ismanager
[out] Is a manager.

7.2.104 OrgUserManagersGet

Retrieves a list of managers (direct and optionally indirect) for a user.

```
struct OrgUserManagersGet_scalarInput
    {
    var $id
    var $directonly
    };

struct OrgUserManagersGet_vectorOutput
    {
    var $id
    var $mlevel
    var $clevel
    var $reportto
    var $ismanager
    };
```

Parameters

id
[in] The profile ID of the user whose managers will be retrieved.

directonly
[in] If non-zero, only the manager who is a direct manager of the user will be retrieved. If zero, both the direct manager and all the managers above him or her in the management chain will be retrieved.

id
[out] User ID.

mlevel
[out] Manually-entered level.

clevel

[out] Calculated level.

reportto
[out] Higher ranked user to whom the user reports.

ismanager
[out] Is a manager.

7.2.105 OrgUserSubordinatesGet

Retrieves a list of subordinates for a manager.

```
struct OrgUserSubordinatesGet_scalarInput
    {
    var $id
    var $directonly
    };

struct OrgUserSubordinatesGet_vectorOutput
    {
    var $id
    var $mlevel
    var $clevel
    var $reportto
    var $ismanager
    };
```

Parameters

id
[in] The profile ID of the manager whose subordinates are to be retrieved.

directonly
[in] If non-zero, only the direct subordinates of the manager will be retrieved. If zero, both the direct subordinates and all the subordinates below them in the management chain will be retrieved.

id
[out] User ID.

mlevel
[out] Manually-entered level.

clevel
[out] Calculated level.

reportto
[out] Higher ranked user to whom the user reports.

ismanager
[out] Is a manager.

7.2.106 PAMBatchResultsList

Retrieves details for a randomization batch.

```
struct PAMBatchResultsList_scalarInput
    {
    var $workid
    };

struct PAMBatchResultsList_vectorOutput
    {
    var $systemid
    var $longid
    var $policyid
    var $status
    var $message
    };
```

Parameters

workid
[in] The unique identifier for this batch.

systemid
[out] The managed system the account exists on.

longid
[out] The managed account's long ID.

policyid
[out] The policy under which this randomization occurred.

status
[out] The result of randomizing this account. S means success, E means error, W means warning, N means none.

message
[out] Informational message.

7.2.107 PSLStoreGet

Retrieves all values associated with the specified key(s) in persistent storage.

```
struct PSLStoreGet_scalarInput
    {
    var $nameSpace
    struct PSLStoreGet_vectorInput $inputArray[]
    };
```



```
struct PSLStoreGet_vectorInput
    {
    var $key
    };

struct PSLStoreGet_vectorOutput
    {
    var $key
    var $value
    };
```

Parameters

nameSpace
[in] The namespace from which to retrieve values for the specified key(s).

inputArray
[in] An array of structs of type PSLStoreGet_vectorInput

key
[in] The key(s) of the values to be retrieved. At least one key must be specified.

key
[out] The key of the retrieved value.

value
[out] Value associated with the specified key.

7.2.108 PSLStoreGetKeys

Retrieves all keys belonging to the specified namespace in persistent storage.

```
struct PSLStoreGetKeys_scalarInput
    {
    var $nameSpace
    };

struct PSLStoreGetKeys_vectorOutput
    {
    var $key
    };
```

Parameters

nameSpace
[in] The namespace from which to retrieve keys.

key
[out] The name of the retrieved key.

7.2.109 PSLStoreNamespacesGet

Returns existing namespaces in persistent storage.

```
struct PSLStoreNamespacesGet_scalarInput
    {
    var $unused


    };

struct PSLStoreNamespacesGet_vectorOutput
    {
    var $nameSpace
    };
```

Parameters

unused
[in] Not used

nameSpace
[out] A namespace.

7.2.110 PSLStoreRemove

Deletes specific value(s) associated with a specified key in persistent storage.

```
struct PSLStoreRemove_scalarInput
    {
    var $nameSpace
    struct PSLStoreRemove_vectorInput $inputArray[]
    };

struct PSLStoreRemove_vectorInput
    {
    var $key
    var $value
    };
```

Parameters

nameSpace
[in] The namespace in which to delete the key/value pairs.

inputArray
[in] An array of structs of type PSLStoreRemove_vectorInput

key
[in] The key of the value to be deleted.

value
[in] The value to delete.

7.2.111 PSLStoreSet

Sets a value(s) in persistent storage. Any existing values for the key are removed.

```
struct PSLStoreSet_scalarInput
    {
    var $nameSpace
    struct PSLStoreSet_vectorInput $inputArray[]
    };

struct PSLStoreSet_vectorInput
    {
    var $key
    var $value
    };
```

Parameters

nameSpace
[in] The namespace in which to set the key and value.

inputArray
[in] An array of structs of type PSLStoreSet_vectorInput

key
[in] The key with which to store the value.

value
[in] The value to store. Values for multi-value storage should be specified across a series of key/value pairs.

7.2.112 PasswordRandomGet

Obtains randomly generated passwords that pass all of the Bravura Security Fabric password rules. No more than 100 passwords will be generated in a single call. The generated number of passwords may not meet the required number. The actual number of generated passwords will be in the output's numGenerated field. When this happens, errmsg will contain a warning indicating how many passwords are generated.

```
struct PasswordRandomGet_scalarInput
    {
    var $userid
    var $numToGenerate
    var $targetid
    };
```



```
struct PasswordRandomGet_vectorOutput
    {
    var $numGenerated
    var $password
    };
```

Parameters

userid
[in] A profile ID.

numToGenerate
[in] The number of passwords to generate. No more than 100 passwords will be generated at once, even if this parameter is larger than 100.

targetid
[in] The ID of a target system to generate passwords for. If nothing is specified, the default policy will be used.

numGenerated
[out] The number of generated passwords returned.

password
[out] A Bravura Security Fabric-generated random password.

7.2.113 PasswordRulesGet

Get the password rules for a specific target. The API caller needs the "change passwords" ACL to be able to run this function properly.

```
struct PasswordRulesGet_scalarInput
    {
    var $targetid
    var $accountid
    };

struct PasswordRulesGet_vectorOutput
    {
    var $tag
    var $status
    var $value
    var $longvalue
    };
```

Parameters

targetid
[in] The ID identifying the target system to get password rules for.

accountid

[in] The account ID on the above target system (used to retrieve userclass selected password policy).

tag
[out] Identifier for the strength rule.

status
[out] The strength rule status. Possible values are 'DIS' (disabled), 'WARN' (produce a warning if rule is broken), and 'REQ' (fail strength check if rule is broken).

value
[out] The strength rule value.

longvalue
[out] The strength rule longvalue.

7.2.114 PasswordStrengthCheck

Test a supplied password against strength rules. This API method is used by the interceptor during transparent synchronization and will delegate the strength check to IDPM. Consider using PasswordStrengthTest API method in all other cases. The API caller needs the "change passwords" ACL to be able to run this function properly. This API requires IDPM to be installed.

```
struct PasswordStrengthCheck_scalarInput
    {
    var $password
    var $longid
    var $targetid
    };

struct PasswordStrengthCheck_vectorOutput
    {
    var $strengthresult
    var $retmsg
    };
```

Parameters

password
[in] The password to be tested.

longid
[in] The account's longid to test the password.

targetid
[in] The target ID associated with the new password.

strengthresult
[out] The password strength result. A value of 0 indicates success, failure otherwise.

retmsg

[out] The detailed returned message from the **idpm** service.

7.2.115 PasswordStrengthMessageGet

Get the password strength message for a specific target. The API caller needs the "change passwords" ACL to be able to run this function properly.

```
struct PasswordStrengthMessageGet_scalarInput
    {
    var $targetid
    var $longid
    var $type
    var $language
    };

struct PasswordStrengthMessageGet_vectorOutput
    {
    var $message
    };
```

Parameters

targetid
[in] The ID identifying the target system to get password rules for.

longid
[in] The account ID on the above target system (used to retrieve userclass selected password policy).

type
[in] The message type: 0 – HTML, 1 – formatted text, 2 – for password transparent synchronization trigger.

language
[in] The skin language to be used to create the password strength message. If it is omitted, the default 'en-us' will be used.

message
[out] The password strength message.

7.2.116 PasswordStrengthTest

Evaluate a password strength against a password policy. This API method does not delegate to IDPM and does not need IDPM to be installed.

The password policy is given by a Password Policy ID. This can be given to the API method in two ways: 1) Directly specified (in the passwordpolicyid input field). 2) Through a Target System ID (in the targetid input field). The password policy id is then inferred from this targetid. The password policy will be the password policy of the Target System Group that the target system (with targetid) belongs to.

The longid and profileid input parameters are optional and only allowed when having specified a Target

System ID.

When only specifying the Target System ID, the given password can only be validated against non-user specific policy rules. Non user specific rules are those rules that are not related to checking the password against a user's user name and user id. To still check these user name rules, an account id (longid) must be specified, in addition to the Target System ID.

In case an account does not exist yet - and there is no longid - a Profile ID may still be specified as input parameter.

The API caller needs the "change passwords" ACL to be able to run this function properly.

```
struct PasswordStrengthTest_scalarInput
    {
    var $password
    var $passwordpolicyid
    var $targetid
    var $longid
    var $profileid
    };

struct PasswordStrengthTest_vectorOutput
    {
    var $strengthresult
    var $retmsg
    };
```

Parameters

password
[in] The password to be tested.

passwordpolicyid
[in] The password policy id to test the password against.

targetid
[in] The target from which the password policy id should be read.

longid
[in] Optional with targetid. The account's longid to test the password.

profileid
[in] Optional. Only valid when testing for idType of Target ID.

strengthresult
[out] The password strength result. A value of 0 indicates success, failure otherwise.

retmsg
[out] Success message if the password check succeeds. In any other case, an empty text.

7.2.117 PolicyAuthorizerList

Retrieves all authorizers for a given managed system policy. If both recipient and requester are specified, will simulate a request and enumerate authorizers based on user classes. If neither recipient nor requester is specified, only statically-defined authorizers will be enumerated.

```
struct PolicyAuthorizerList_scalarInput
    {
    var $id
    var $recipient
    var $requester
    };

struct PolicyAuthorizerList_vectorOutput
    {
    var $id
    var $name
    var $type
    var $phase
    };
```

Parameters

id
[in] The ID of the managed system policy.

recipient
[in] Optional. The recipient of the potential request.

requester
[in] Optional. The requester of the potential request.

id
[out] The ID of the retrieved authorizer.

name
[out] The name of the authorizer.

type
[out] The type of authorization provided by this authorizer.

phase
[out] The phase this authorizer belongs to.

7.2.118 PolicyManagedAccountAdd

Manages an account with a managed system policy.
Requires the "Modify properties for this policy" ACL


```
struct PolicyManagedAccountAdd_scalarInput
    {
    var $policyid
    var $systemid
    var $accountid
    var $setInitialPassword
    };
```

Parameters

policyid
[in] The ID of the managed system policy.

systemid
[in] The ID of the managed system.

accountid
[in] The ID of the account.

setInitialPassword
[in] (deprecated) If nonzero, populate an initial empty password entry for the managed account. This will occur only if there is no existing password known for the managed account.

7.2.119 PolicyManagedAccountDelete

Deletes an account on a managed system from a managed system policy. Returns API_SUCCESS if an accountid that isn't managed by the policy is given.
Requires the "Modify properties for this policy" ACL.

```
struct PolicyManagedAccountDelete_scalarInput
    {
    var $policyid
    var $systemid
    var $accountid
    };
```

Parameters

policyid
[in] The ID of the policy to delete the managed acocunt from.

systemid
[in] The ID of the managed system to remove the account from.

accountid
[in] The ID of the account to remove.

7.2.120 PolicyManagedAccountList

Retrieves all accounts managed by a given managed system policy.

```
struct PolicyManagedAccountList_scalarInput
    {
    var $id
    };

struct PolicyManagedAccountList_vectorOutput
    {
    var $id
    var $wstnid
    var $validPassword
    };
```

Parameters

id
[in] The ID of the managed system policy.

id
[out] The ID of the retrieved managed account.

wstnid
[out] The workstation ID the account is associated with.

validPassword
[out] Whether the instance has a valid password for this account on this system.

7.2.121 PolicyManagedGroupSetList

Returns a list of group sets belonging to a given managed system policy.

```
struct PolicyManagedGroupSetList_scalarInput
    {
    var $resourceid
    };

struct PolicyManagedGroupSetList_vectorOutput
    {
    var $gsetid
    var $description
    };
```

Parameters

resourceid
[in] The ID of the managed system policy.

gsetid
[out] The ID of the group set.

description
[out] The description of the group set.

7.2.122 PolicyManagedGroupSetMembersList

Lists members of a group set.
If the systemid input is not empty, evaluates rules against groups on the specified system.
Will return duplicates if explicitly configured groups also match rules.

```
struct PolicyManagedGroupSetMembersList_scalarInput
    {
    var $resourceid
    var $gsetid
    var $systemid
    };

struct PolicyManagedGroupSetMembersList_vectorOutput
    {
    var $groupid
    var $description
    var $sid
    var $isexplicit
    };
```

Parameters

resourceid
[in] The ID of the managed system policy.

gsetid
[in] The ID of the group set.

systemid
[in] (Optional) The ID of the managed system. If specified, evaluates rule-based members.

groupid
[out] The ID of the group.

description
[out] The description of the group.

sid
[out] The SID for the group, if available.

isexplicit
[out] Whether the group was added to the set explicitly or through rule evaluation. 1 means explicit, 0 means via rules.

7.2.123 PolicyManagedGroupSetTargetList

Returns a list of configured targets for a given group set.

```
struct PolicyManagedGroupSetTargetList_scalarInput
    {
    var $resourceid
    var $gsetid
    };

struct PolicyManagedGroupSetTargetList_vectorOutput
    {
    var $targetid
    };
```

Parameters

resourceid
[in] The ID of the managed system policy.

gsetid
[in] The ID of the group set on the policy.

targetid
[out] The ID of the target.

7.2.124 PolicyManagingNodeSet

Changes the managing node for a managed system policy.

```
struct PolicyManagingNodeSet_scalarInput
    {
    var $policyid
    var $serverid
    var $force
    };
```

Parameters

policyid
[in] The ID of the managed system policy.

serverid
[in] The new managing node's GUID. If this parameter is empty, the policy's managing node will be unassigned.

force
[in] Under normal conditions, managed system policy ownership handoff is cooperative: a safe handoff requires the current managing node to voluntarily relinquish ownership. If the current managing node is down or otherwise unable to participate, the new managing node can be forced. This can cause desynchro-

nization between nodes if misused. Only set this parameter to 1 if you are certain the current managing node is down.

7.2.125 PolicyMemberSystemAdd

Attaches a managed system to a managed system policy.

```
struct PolicyMemberSystemAdd_scalarInput
    {
    var $policyid
    var $systemid
    };
```

Parameters

policyid
[in] The ID of the policy to attach the managed system to.

systemid
[in] The ID of the managed system to attach to the policy.

7.2.126 PolicyMemberSystemDelete

Removes a managed system from a managed system policy. Respects the RES MEMBER SYSTEMS DELETE MODE system variable.

```
struct PolicyMemberSystemDelete_scalarInput
    {
    var $policyid
    var $systemid
    };
```

Parameters

policyid
[in] The ID of the policy to detach the managed system from.

systemid
[in] The ID of the managed system to detach from the policy.

7.2.127 PolicyMemberSystemList

Retrieves all member systems associated with a given managed system policy.

```
struct PolicyMemberSystemList_scalarInput
    {
```



```
    var $id
    };

struct PolicyMemberSystemList_vectorOutput
    {
    var $id
    var $description
    var $type
    };
```

Parameters

id
[in] The ID of the managed system policy.

id
[out] The ID of the retrieved workstation.

description
[out] The description of the retrieved workstation.

type
[out] The management type of the retrieved workstation. H means push, L means pull, M means vault.

7.2.128 PreDefinedRequestList

Return a list of pre-defined request by enable status and/or intended recipients

```
struct PreDefinedRequestList_scalarInput
    {
    var $enabled
    var $reciptype
    };

struct PreDefinedRequestList_vectorOutput
    {
    var $preqid
    var $desc
    var $url
    var $created
    var $enabled
    var $type
    var $deprecated
    var $deprecatedreason
    };
```

Parameters

enabled
[in] Enabled (1 for enabled pre-defined requests, 0 for disabled pre-defined requests, -1 for both)

reciptype
[in] Intended recipient type ('A' for all users, 'E' for existing users, and 'N' for new users). Default to all recipient types.

preqid
[out] Pre-defined request ID

desc
[out] Pre-defined request description

url
[out] URL link that describes the pre-defined request.

created
[out] Created on date

enabled
[out] Whether the pre-defined request is enabled for use.

type
[out] Target audience for this pre-defined request

deprecated
[out] Deprecated date

deprecatedreason
[out] Deprecated reason

7.2.129 PreRequestMemberAdd

Adds a resource to the given pre-defined request.

```
struct PreRequestMemberAdd_scalarInput
    {
    var $preqid
    struct PreRequestMemberAdd_vectorInput $inputArray[]
    };

struct PreRequestMemberAdd_vectorInput
    {
    var $resid
    var $restype
    var $opertype
    var $required
    var $groupid
    };
```

Parameters

preqid

[in] The ID of the pre-defined request.

inputArray
[in] An array of structs of type PreRequestMemberAdd_vectorInput

resid
[in] The ID of the resource. For managed groups, this is either the GUID assigned to it from auto-discovery, or the ID of the target to which the group belongs. For resource type PROFILE, this is assumed to be !!!___ALL___.

restype
[in] The type of the resource. Note that the type is one of the following: ROLE, TARG, TMPL, ATTRGRP, MGRP, PROFILE, and CUSTOM.

opertype
[in] The type of operation. If restype is TARG, then opertype can be UPDT (Update account)(default), ENAU (Enable account), DNAU (Disable account), DELU (Delete account), MVCU (Move account from one context to another), or GRUD (Remove a user from a group). If restype is TMPL, then opertype is ACUA (Add account to user)(default). If restype is MGRP, then opertype can be GRUA (Add a user to a group)(default) or GRUD. If restype is ROLE, then opertype can be RLUA (Add role to user)(default) or DELR (Delete role). If restype is PROFILE, then opertype can be ENAP (enable profile) or DNAP (disable profile). If restype is CUSTOM, then opertype is CUSTOM1. Ignored for attribute groups.

required
[in] If 1, the resource operation is required. Otherwise, the resource operation is optional. If restype is MGRP and opertype is GRUD, then required is 1 by default. If restype is ROLE and opertype is DELR, then required is 1 by default. Ignored for attribute groups.

groupid
[in] The group ID, for managed group members, if resid is the target ID.

7.2.130 PreRequestMemberDelete

Deletes a resource from the given pre-defined request.

```
struct PreRequestMemberDelete_scalarInput
    {
    var $preqid
    struct PreRequestMemberDelete_vectorInput $inputArray[]
    };

struct PreRequestMemberDelete_vectorInput
    {
    var $resid
    var $restype
    var $opertype
    var $groupid
    };
```

Parameters

preqid
[in] The ID of the pre-defined request.

inputArray
[in] An array of structs of type PreRequestMemberDelete_vectorInput

resid
[in] The ID of the resource member. For managed groups, this is either the GUID assigned to it from auto-discovery, or the ID of the target to which the group belongs. For resource type PROFILE, this is assumed to be !!!___ALL___.

restype
[in] The type of the resource member. Note that the type is one of the following: ROLE, TARG, TMPL, ATTRGRP, MGRP, PROFILE, CUSTOM.

opertype
[in] The type of operation. If restype is TARG, then opertype can be UPDT (Update account)(default), ENAU (Enable account), DNAU (Disable account), DELU (Delete account), MVCU (Move account from one context to another), or GRUD (Remove a user from a group). If restype is TMPL, then opertype is ACUA (Add account to user)(default). If restype is MGRP, then opertype can be GRUA (Add a user to a group)(default) or GRUD. If restype is ROLE, then opertype can be RLUA (Add role to user)(default) or DELR (Delete role). If restype is PROFILE, then opertype can be ENAP (enable profile) or DNAP (disable profile). If restype is CUSTOM, the opertype is CUSTOM1.

groupid
[in] The group ID, for managed group members, if resid is the target ID.

7.2.131 PreRequestMemberGet

This function obtains the predefined request information. Note: Some of the variable names and resource type values returned by this function have changed with version 8.3 of this product. Programs which were written to use earlier versions of this function may need to be updated for these changes.

```
struct PreRequestMemberGet_scalarInput
    {
    var $preqid
    var $required
    };

struct PreRequestMemberGet_vectorOutput
    {
    var $resid
    var $groupid
    var $restype
    var $opertype
    var $required
    };
```

Parameters

preqid
[in] The predefine request id.

required
[in] Returns all members if the value is -1. Returns only required members if the value is 1. Returns only optional members if the value is 0.

resid
[out] (formerly memberid) The member resource's identifier.

groupid
[out] The group ID, for managed group members.

restype
[out] (formerly membertype) The member resource's type.

Note that the type is one of the following: ROLE, TARG (formerly TARGET), ATTRGRP (formerly ATTRGROUP), TMPL (formerly TEMPLATE), MGRP (formerly MANAGEDGROUP), PROFILE, or CUSTOM.

opertype
[out] The member resource's operation type.

required
[out] If 1, the member is required. If zero, the member is optional.

7.2.132 QSetQuestionsGet

Get questions from a pre-defined question set.

```
struct QSetQuestionsGet_scalarInput
    {
    var $qsid
    };

struct QSetQuestionsGet_vectorOutput
    {
    var $qid
    var $question
    };
```

Parameters

qsid
[in] The question set ID to return questions from.

qid
[out] The ID used to identify the question.

question
[out] The question text.

7.2.133 RecoverKeyByAccount

Tries to fetch the password of an account on a specified system that is no longer managed by Bravura Privilege. This password is the last known value prior to the system becoming unmanaged and may have been changed locally since. Note that the caller must have the OTP IDAPI Caller privilege, as well as the Recover Last Managed Password ACL, in order to use this function.

```
struct RecoverKeyByAccount_scalarInput
    {
    var $systemid
    var $accountid
    };

struct RecoverKeyByAccount_vectorOutput
    {
    var $password
    var $laststoredtime
    };
```

Parameters

systemid
[in] The unmanaged system ID where the managed account ID is hosted.

accountid
[in] The long ID of the managed account in the resource.

password
[out] The last known password of the account ID on the given system.

laststoredtime
[out] The ISO-formatted UTC time this password was added to history table.

7.2.134 RegSysvarSet

Set system variable located in registry of instance.

```
struct RegSysvarSet_scalarInput
    {
    var $name
    var $value
    };
```

Parameters

name
[in] Name of system variable.

value
[in] Value to assign to system variable. To disable or delete a system variable, use an empty string.

7.2.135 ReplicasAdd

Add replica(s) to this replicated instance. If there are no replicas currently, settings for the local node will be copied from the first vector input; the local node's description can be set by the localdescription scalar parameter. External addresses cannot be set with this function. Failure to propagate the new replication configuration to all nodes does not cause this function to return an error code (failure to validate the configuration will return an error). Instead, success is returned along with a list of nodes that the configuration was successfully propagated to. Note that this list may be larger than the list of inputs as it includes current replicas.

```
struct ReplicasAdd_scalarInput
    {
    var $resynch
    var $localdescription
    struct ReplicasAdd_vectorInput $inputArray[]
    };

struct ReplicasAdd_vectorInput
    {
    var $address
    var $port
    var $description
    var $receive_queue_type
    var $receive_queue_min
    var $receive_queue_max
    var $receive_queue_warn
    var $send_queue_type
    var $send_queue_min
    var $send_queue_max
    var $send_queue_warn
    };

struct ReplicasAdd_vectorOutput
    {
    var $node_id
    };
```

Parameters

resynch
[in] Resynchronize nodes after adding (setting to true is strongly recommended).

localdescription
[in] Optional. The new description of the local node, if non-empty. This parameter is used when configuring replication initially so that all replicas' descriptions have the same format.

inputArray
[in] An array of structs of type ReplicasAdd_vectorInput

address
[in] The address of the new node.

port
[in] The port number of iddb on the new node.

description
[in] Description for the new node.

receive_queue_type
[in] Type for the receive queue on the new node. Must be one of: Relative: receive_queue_max and receive_queue_warn parameters will be interpreted as percentages of disk size. Fixed: receive_queue_max and receive_queue_warn parameters will be interpreted in megabytes. Default: receive_queue_max and receive_queue_warn parameters must be zero.

receive_queue_min
[in] Minimum size of the receive queue, in megabytes. Must be at least 100 MB.

receive_queue_max
[in] Maximum size of the receive queue.

receive_queue_warn
[in] Receive queue usage warning threshold.

send_queue_type
[in] Type for the send queue on the new node. Must be one of: Relative: send_queue_max and send_queue_warn parameters will be interpreted as percentages of disk size. Fixed: send_queue_max and send_queue_warn parameters parameters will be interpreted in megabytes. Default: send_queue_max and send_queue_warn parameters parameters must be zero.

send_queue_min
[in] Minimum size of the send queue, in megabytes. Must be at least 100 MB.

send_queue_max
[in] Maximum size of the send queue.

send_queue_warn
[in] Send queue usage warning threshold.

node_id
[out] Identifier of node that was successfully propagated to.

7.2.136 ReserveAccountId

Reserves account ID.

```
struct ReserveAccountId_scalarInput
    {
    var $resid
    var $accountid
    var $targetid
    var $iscasesensitive
    var $permanent
```



```
    };

struct ReserveAccountId_vectorOutput
    {
    var $resid
    };
```

Parameters

resid
[in] A pre-existing reservation ID with which to associate the reservations in this request. If this is empty, a new reservation ID will be created. If you want to create a reservation that consists of multiple types of reservable items, call this once with an empty ID and then re-use the returned resid upon subsequent calls. This will facilitate revoking the reservation.

accountid
[in] An account ID.

targetid
[in] (Optional) A target ID.

iscasesensitive
[in] 1 for case sensitive, otherwise case insensitive.

permanent
[in] 1 for permanent, otherwise not permanent.

resid
[out] The reservation ID.

7.2.137 ReserveAcctAttrValue

Reserves account attribute values.

```
struct ReserveAcctAttrValue_scalarInput
    {
    var $resid
    var $name
    var $targetid
    var $iscasesensitive
    var $permanent
    struct ReserveAcctAttrValue_vectorInput $inputArray[]
    };

struct ReserveAcctAttrValue_vectorInput
    {
    var $value
    };
```



```
struct ReserveAcctAttrValue_vectorOutput
    {
    var $resid
    };
```

Parameters

resid
[in] A pre-existing reservation ID with which to associate the reservations in this request. If this is empty, a new reservation ID will be created. If you want to create a reservation that consists of multiple types of reservable items, call this once with an empty ID and then re-use the returned resid upon subsequent calls. This will facilitate revoking the reservation.

name
[in] An attribute key name.

targetid
[in] A target ID.

iscasesensitive
[in] 1 for case sensitive, otherwise case insensitive.

permanent
[in] 1 for permanent, otherwise not permanent.

inputArray
[in] An array of structs of type ReserveAcctAttrValue_vectorInput

value
[in] An attribute value.

resid
[out] The reservation ID.

7.2.138 ReserveAttrValue

Reserves attribute values.

```
struct ReserveAttrValue_scalarInput
    {
    var $resid
    var $name
    var $restype
    var $iscasesensitive
    var $permanent
    struct ReserveAttrValue_vectorInput $inputArray[]
    };

struct ReserveAttrValue_vectorInput
    {
```



```
    var $value
    };

struct ReserveAttrValue_vectorOutput
    {
    var $resid
    };
```

Parameters

resid
[in] A pre-existing reservation ID with which to associate the reservations in this request. If this is empty, a new reservation ID will be created. If you want to create a reservation that consists of multiple types of reservable items, call this once with an empty ID and then re-use the returned resid upon subsequent calls. This will facilitate revoking the reservation.

name
[in] An attribute key name.

restype
[in] For attributes only, the type of resource to associate the attribute with: PROF = Profile attribute (default), MGRP = Managed groups, ROLE = Roles, TSYS = Target systems, TMPL = Template accounts, SODR = Segregation of duties rules, MSYS = Managed systems, MACC = Managed accounts, GRPAM = Group account memberships, GRPGM = Group group memberships

iscasesensitive
[in] 1 for case sensitive, otherwise case insensitive.

permanent
[in] 1 for permanent, otherwise not permanent.

inputArray
[in] An array of structs of type ReserveAttrValue_vectorInput

value
[in] An attribute value.

resid
[out] The reservation ID.

7.2.139 ReserveCheck

Checks current usage and the reservation list for one of profile ID, account ID, and attribute values. If ERR_COULD_NOT_RESERVE is returned, then the error message will contain either the reservation under which the ID was reserved, or "IN USE" if the ID is already being used.

```
struct ReserveCheck_scalarInput
    {
    var $key
    var $restype
```



```
    var $flag
    struct ReserveCheck_vectorInput $inputArray[]
    };

struct ReserveCheck_vectorInput
    {
    var $value
    };
```

Parameters

key
[in] The attribute key name.

restype
[in] For attributes only, the type of resource to associate the attribute with: PROF = Profile attribute (default), MGRP = Managed groups, ROLE = Roles, TSYS = Target systems, TMPL = Template accounts, SODR = Segregation of duties rules, MSYS = Managed systems, MACC = Managed accounts, GRPAM = Group account memberships, GRPGM = Group group memberships

flag
[in] For PSLang IDAPI, there are 5 predefined flags, casesensitive, account, attribute, account attribute and profile, available with PSLANG_FLAG_DEF struct. The casesensitive flag could be used along with account, attribute and account attribute to indicate case-sensitive checking. Also, all users could select one of seven values for the flag:

- 2 for account ID, case insensitive mode
- 3 for account ID, case sensitive mode
- 4 for attribute values, case insensitive mode
- 5 for attribute values, case sensitive mode
- 8 for profile ID
- 16 for account attribute values, case insensitive mode
- 17 for account attribute values, case sensitive mode

inputArray
[in] An array of structs of type ReserveCheck_vectorInput

value
[in] A value corresponding to either a profile ID, an account ID, or an attribute. If name is "PROFILEID" or "ACCOUNTID", only the first value specified will be used for the profile ID or account ID, respectively. If a target ID must be specified for the account ID value, then it should be provided as the second value. Account IDs are interpreted as short IDs.

7.2.140 ReserveGet

Retrieve all identifiers grouped by the given reservation ID.


```
struct ReserveGet_scalarInput
    {
    var $resid
    };

struct ReserveGet_vectorOutput
    {
    var $type
    var $target
    var $key
    var $restype
    var $value
    var $permanent
    };
```

Parameters

resid
[in] A pre-existing reservation ID whose reservations are to be retrieved.

type
[out] The reservation type. It could be Attribute, Account or Profile

target
[out] The target used for account and account attribute case

key
[out] The key part of the reservation. It could be attribute key for attribute.datatype, target id for Account type or profile id for Profile type

restype
[out] For attributes only, the type of resource to associate the attribute with: PROF = Profile attribute (default), MGRP = Managed groups, ROLE = Roles, TSYS = Target systems, TMPL = Template accounts, SODR = Segregation of duties rules, MSYS = Managed systems, MACC = Managed accounts, GRPAM = Group account memberships, GRPGM = Group group memberships

value
[out] The value part of the reservation. It could be attribute value for attribute.datatype, account id for Account type or empty string for Profile type

permanent
[out] 1 if the reservation should persist past any request

7.2.141 ReserveProfileId

Reserves profile ID.

```
struct ReserveProfileId_scalarInput
    {
    var $resid
```



```
    var $profileid
    var $permanent
    };

struct ReserveProfileId_vectorOutput
    {
    var $resid
    };
```

Parameters

resid
[in] A pre-existing reservation ID with which to associate the reservations in this request. If this is empty, a new reservation ID will be created. If you want to create a reservation that consists of multiple types of reservable items, call this once with an empty ID and then re-use the returned resid upon subsequent calls. This will facilitate revoking the reservation.

profileid
[in] A profile ID.

permanent
[in] 1 for permanent, otherwise not permanent.

resid
[out] The reservation ID.

7.2.142 ReserveRevoke

Revokes all identifiers grouped by the given reservation ID.

```
struct ReserveRevoke_scalarInput
    {
    var $resid
    };
```

Parameters

resid
[in] A pre-existing reservation ID whose reservations are to be revoked.

7.2.143 ResourceAttrNamesList

Lists information about the attribute names used to configure a given resource type.

```
struct ResourceAttrNamesList_scalarInput
    {
    var $restype
    };
```



```
struct ResourceAttrNamesList_vectorOutput
    {
    var $name
    var $isrequired
    var $isrequiredread
    var $description
    };
```

Parameters

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList where readonly is zero.

name
[out] The name of the attribute.

isrequired
[out] Whether a value for the attribute is required to create the resource.

isrequiredread
[out] Whether a value for the attribute is required to read the resource

description
[out] The description of the attribute.

7.2.144 ResourceAttrsDel

Deletes the attribute values associated with a given existing resource. If the attribute is defined but no values exist for it, then API_NO_MORE_DATA is returned.

```
struct ResourceAttrsDel_scalarInput
    {
    var $resourceid
    var $resourceid2
    var $restype
    struct ResourceAttrsDel_vectorInput $inputArray[]
    };

struct ResourceAttrsDel_vectorInput
    {
    var $name
    var $index
    };
```

Parameters

resourceid
[in] The ID of the resource. For managed groups, this is either the GUID returned by ResourceFind, or the target ID if used in conjunction with the resourceid2 parameter.

resourceid2
[in] For managed groups, this is either the groupid or shortid. For managed accounts, this is the managed system ID. For other resource types this parameter is ignored.

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList.

inputArray
[in] An array of structs of type ResourceAttrsDel_vectorInput

name
[in] The name of the attribute.

index
[in] The zero-based index of the value of the attribute, if the attribute is a multi-value attribute. If -1 is used, all values of the attribute will be deleted. Must be 0 or -1 if it is a single-value attribute.

7.2.145 ResourceAttrsGet

Retrieves the values of specified attributes of a given existing resource. If attribute names are not specified, the values of all attributes associated with the given resource are retrieved. When values of a multi-valued attribute are returned, they are broken up and returned as a series of attribute name/value pairs.

```
struct ResourceAttrsGet_scalarInput
    {
    var $resourceid
    var $resourceid2
    var $restype
    struct ResourceAttrsGet_vectorInput $inputArray[]
    };

struct ResourceAttrsGet_vectorInput
    {
    var $name
    };

struct ResourceAttrsGet_vectorOutput
    {
    var $name
    var $value
    };
```

Parameters

resourceid
[in] The ID of the resource. For managed groups, this is either the GUID returned by ResourceFind, or the target ID if used in conjunction with the resourceid2 parameter.

resourceid2
[in] For managed groups, this is either the groupid or shortid. For managed accounts, this is the managed

system ID. For other resource types this parameter is ignored.

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList.

inputArray
[in] An array of structs of type ResourceAttrsGet_vectorInput

name
[in] The name of the attribute.

name
[out] The name of the attribute.

value
[out] The value of the attribute.

7.2.146 ResourceAttrsSet

Sets the attribute values associated with a given existing resource.

```
struct ResourceAttrsSet_scalarInput
    {
    var $resourceid
    var $resourceid2
    var $restype
    var $replace
    struct ResourceAttrsSet_vectorInput $inputArray[]
    };

struct ResourceAttrsSet_vectorInput
    {
    var $name
    var $value
    };
```

Parameters

resourceid
[in] The ID of the resource. For managed groups, this is either the GUID returned by ResourceFind, or the target ID if used in conjunction with the resourceid2 parameter.

resourceid2
[in] For managed groups, this is either the groupid or shortid. For managed accounts, this is the managed system ID. For other resource types this parameter is ignored.

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList.

replace

[in] If zero, values passed to the function for the attribute will be appended to the attribute. If non-zero, the values will replace those already assigned to the attribute.

inputArray
[in] An array of structs of type ResourceAttrsSet_vectorInput

name
[in] The name of the attribute.

value
[in] The value of the attribute. To specify values for a multi-valued attribute, specify them as a series of attribute name/value pairs.

7.2.147 ResourceAuthInfoDelete

Delete the authorization info for a phase of a resource. Note that this will also delete any static authorizers and user classes configured for this phase.

```
struct ResourceAuthInfoDelete_scalarInput
    {
    var $resourceid
    var $restype
    var $phase
    };
```

Parameters

resourceid
[in] The ID of the resource.

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList.

phase
[in] Which phase the authorization info is for.

7.2.148 ResourceAuthInfoList

List the authorization info for a resource.

```
struct ResourceAuthInfoList_scalarInput
    {
    var $resourceid
    var $restype
    };

struct ResourceAuthInfoList_vectorOutput
    {
    var $phase
```



```
    var $minauth
    var $maxrej
    };
```

Parameters

resourceid
[in] The ID of the resource.

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList.

phase
[out] Which phase the authorization info is for.

minauth
[out] Minimum number of authorizations required for the resource to be approved.

maxrej
[out] Maximum number of denials before the resource is denied (0 means ignore denials, only count approvals).

7.2.149 ResourceAuthInfoSet

Set the authorization info for a phase of a resource.

```
struct ResourceAuthInfoSet_scalarInput
    {
    var $resourceid
    var $restype
    var $phase
    var $minauth
    var $maxrej
    };
```

Parameters

resourceid
[in] The ID of the resource.

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList.

phase
[in] Which phase the authorization info is for.

minauth
[in] Minimum number of authorizations required for the resource to be approved.

maxrej
[in] Maximum number of denials before the resource is denied (0 means ignore denials, only count ap-

provals).

7.2.150 ResourceAuthUserclassCriteriaIsSet

Check whether the authorization user class criteria for all phases is set on the specified resource.

```
struct ResourceAuthUserclassCriteriaIsSet_scalarInput
    {
    var $resourceid
    var $restype
    };

struct ResourceAuthUserclassCriteriaIsSet_vectorOutput
    {
    var $isset
    };
```

Parameters

resourceid
[in] The ID of the resource.

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList.

isset
[out] Whether the authorization criteria is set.

7.2.151 ResourceAuthorizerAdd

Add a statically assigned authorizer to a resource.

```
struct ResourceAuthorizerAdd_scalarInput
    {
    var $resourceid
    var $restype
    var $userid
    var $phase
    };
```

Parameters

resourceid
[in] The ID of the resource.

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList.

userid

[in] Add this authorizer.

phase
[in] Add this authorizer for this phase (if phases are not enabled, use 1).

7.2.152 ResourceAuthorizerDelete

Delete an authorizer statically assigned to a resource.

```
struct ResourceAuthorizerDelete_scalarInput
    {
    var $resourceid
    var $restype
    var $userid
    var $phase
    };
```

Parameters

resourceid
[in] The ID of the resource.

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList.

userid
[in] Delete this authorizer.

phase
[in] Remove the authorizer from this phase (if phases are not enabled, use 1).

7.2.153 ResourceAuthorizerList

Lists the authorizers statically assigned to a resource.

```
struct ResourceAuthorizerList_scalarInput
    {
    var $resourceid
    var $restype
    };

struct ResourceAuthorizerList_vectorOutput
    {
    var $userid
    var $phase
    };
```

Parameters

resourceid
[in] The ID of the resource.

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList.

userid
[out] The profile ID of the authorizer.

phase
[out] The phase the authorizer will be assigned to (if phases are not enabled, use 1).

7.2.154 ResourceAuthorizersGet

Get all the authorizers assigned to a resource.

```
struct ResourceAuthorizersGet_scalarInput
    {
    var $resourceid
    var $resourceid2
    var $restype
    var $requester
    var $recipient
    var $batchsig
    };

struct ResourceAuthorizersGet_vectorOutput
    {
    var $userid
    var $phase
    };
```

Parameters

resourceid
[in] The ID of the resource. For managed groups, this is either the GUID returned by ResourceFind, or the target ID if used in conjunction with the resourceid2 parameter.

resourceid2
[in] For managed groups, this is either the groupid or shortid.

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList.

requester
[in] The requester of the potential request, when batchsig is not provided, or else the information will be extracted from the request.

recipient
[in] Recipient of the potential request, when batchsig is not provided, or else the information will be

extracted from the request.

batchsig
[in] The batch ID of the potential request.

userid
[out] The profile ID of the authorizer.

phase
[out] The phase the authorizer will be assigned to (if phases are not enabled, use 1).

7.2.155 ResourceCancel

Cancels the creation of the given resource. The handle of the resource becomes invalid and can no longer be used in further function calls.

```
struct ResourceCancel_scalarInput
    {
    var $reshandle
    };
```

Parameters

reshandle
[in] The handle of the resource, as returned by ResourceCreate.

7.2.156 ResourceCreate

Initializes the process of creating a new resource. The resource itself is not created until ResourcePost is called.

```
struct ResourceCreate_scalarInput
    {
    var $restype
    };

struct ResourceCreate_vectorOutput
    {
    var $reshandle
    };
```

Parameters

restype
[in] The type of resource to create. This must be one of the types returned by ResourceTypesList where readonly is zero.

reshandle
[out] A handle associated with this resource, to be used with further function calls for configuring and

creating this resource.

7.2.157 ResourceCreateGet

Retrieves the parameters assigned to the given resource undergoing creation, as assigned by previous calls to ResourceCreateSet.

See Resource Configuration Parameters for the complete list of resource types and their parameters supported by this function.

```
struct ResourceCreateGet_scalarInput
    {
    var $reshandle
    };

struct ResourceCreateGet_vectorOutput
    {
    var $name
    var $value
    };
```

Parameters

reshandle
[in] The handle of the resource, as returned by ResourceCreate.

name
[out] The name of the parameter.

value
[out] The value of the parameter.

7.2.158 ResourceCreateSet

Assigns parameters to the given resource undergoing creation. If a value for a given parameter has already been assigned on a previous call to this function, the newer value replaces the older one.

See Resource Configuration Parameters for the complete list of resource types and their parameters supported by this function.

```
struct ResourceCreateSet_scalarInput
    {
    var $reshandle
    struct ResourceCreateSet_vectorInput $inputArray[]
    };

struct ResourceCreateSet_vectorInput
    {
    var $name
```



```
    var $value
    };
```

Parameters

reshandle
[in] The handle of the resource, as returned by ResourceCreate.

inputArray
[in] An array of structs of type ResourceCreateSet_vectorInput

name
[in] The name of the parameter.

value
[in] The value of the parameter.

7.2.159 ResourceDelete

Delete the given resource. "resourceid" is optional to identify the resource to delete; if empty, the resource is located using parameters in the vector of input values.

```
struct ResourceDelete_scalarInput
    {
    var $resourceid
    var $restype
    struct ResourceDelete_vectorInput $inputArray[]
    };

struct ResourceDelete_vectorInput
    {
    var $name
    var $value
    };
```

Parameters

resourceid
[in] The resource ID.

restype
[in] The resource type. This must be one of the types returned by ResourceTypesList where readonly is zero.

inputArray
[in] An array of structs of type ResourceDelete_vectorInput

name
[in] The name of the parameter.

value

[in] The value of the parameter.

7.2.160 ResourceDependenciesGet

Retrieves the dependencies of the given resource. Can be used before calling ResourceDelete, to determine if the resource has dependencies that would prevent it from being deleted. Note that the resource type of some dependencies may not be supported currently by the other resource functions.

```
struct ResourceDependenciesGet_scalarInput
    {
    var $resourceid
    var $restype
    };

struct ResourceDependenciesGet_vectorOutput
    {
    var $resourceid
    var $restype
    };
```

Parameters

resourceid
[in] The resource ID.

restype
[in] The resource type. This must be one of the types returned by ResourceTypesList where readonly is zero.

resourceid
[out] The resource ID of the dependency.

restype
[out] The resource type of the dependency.

7.2.161 ResourceFind

Finds one or more resources matching a given set of criteria.

See Searching for resources with ResourceFind (p142) for details on using this function.

```
struct ResourceFind_scalarInput
    {
    var $restype
    struct ResourceFind_vectorInput $inputArray[]
    };

struct ResourceFind_vectorInput
    {
```



```
    var $name
    var $value
    };

struct ResourceFind_vectorOutput
    {
    var $resourceid
    var $resourceid2
    };
```

Parameters

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList.

inputArray
[in] An array of structs of type ResourceFind_vectorInput

name
[in] The name of the search criterion. This must be one of the names returned by ResourceFindCriteria for the given resource type.

value
[in] The value of the search attribute.

resourceid
[out] The ID of the resource.

resourceid2
[out] The second part of the ID of the resource.

7.2.162 ResourceFindByAttr

Searches for resources only by attributes. All resource attribute.datatypes except the password type are supported. When search attributes are specified, the results of each match are ANDed together. A maximum of five search criteria is allowed (searching by range counts as two search criteria).

```
struct ResourceFindByAttr_scalarInput
    {
    var $restype
    struct ResourceFindByAttr_vectorInput $inputArray[]
    };

struct ResourceFindByAttr_vectorInput
    {
    var $key
    var $valmin
    var $valmax
    };
```



```
struct ResourceFindByAttr_vectorOutput
    {
    var $resourceid
    var $restype
    };
```

Parameters

restype
[in] (Optional) The type of resource. This must be one of the types in (ROLE,MSYS,MACC). If restype is not specified, then all matching resource types will be returned.

inputArray
[in] An array of structs of type ResourceFindByAttr_vectorInput

key
[in] The attribute ID.

valmin
[in] The value of the attribute. If valmax is used, then valmin is the minimum of a range of values for the search attribute.

valmax
[in] The maximum of a range of values for the search attribute.

resourceid
[out] The ID of the resource.

restype
[out] The type of resource. This is one of the types returned by ResourceTypesList.

7.2.163 ResourceFindCriteria

Lists the search criteria by which the given resource type can be found using ResourceFind.

See ResourceFind Search Criteria for the complete list of search criteria returned by this function.

```
struct ResourceFindCriteria_scalarInput
    {
    var $restype
    };

struct ResourceFindCriteria_vectorOutput
    {
    var $name
    var $description
    var $type
    var $isregexp
    };
```


Parameters

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList.

name
[out] The name of the search criterion.

description
[out] The description of the search criterion.

type
[out] The type of the search criterion.

isregexp
[out] (deprecated) Whether the value of the search criterion can be a regular expression. 0 is false, anything else is true.

7.2.164 ResourceGet

Retrieves the parameters associated with the given existing resource.

See Resource Configuration Parameters for the complete list of resource types and their parameters supported by this function.

```
struct ResourceGet_scalarInput
    {
    var $resourceid
    var $resourceid2
    var $resourceid3
    var $restype
    };

struct ResourceGet_vectorOutput
    {
    var $name
    var $value
    };
```

Parameters

resourceid
[in] The ID of the resource. For managed groups, this is either the GUID returned by ResourceFind, or the target ID if used in conjunction with the resourceid2 parameter. For account attributes, this is either the GUID returned by ResourceFind, or the account attribute ID if used in conjunction with resourceid2 parameter.

resourceid2
[in] For managed groups, this is either the groupid or shortid. For account attributes, this is either the platform ID or target ID. For other resource types this parameter is ignored.

resourceid3
[in] For account attributes, this is the override level, 0 (no override), 1 (platform-level override), or 2 (target-level override). For other resource types this parameter is ignored.

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList.

name
[out] The name of the parameter.

value
[out] The value of the parameter.

7.2.165 ResourceImplUserclassCriteriaIsSet

Check whether the implementer user class criteria for all phases is set on the specified resource.

```
struct ResourceImplUserclassCriteriaIsSet_scalarInput
    {
    var $resourceid
    var $restype
    };

struct ResourceImplUserclassCriteriaIsSet_vectorOutput
    {
    var $isset
    };
```

Parameters

resourceid
[in] The ID of the resource.

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList.

isset
[out] Whether the authorization criteria is set.

7.2.166 ResourceImplementerAdd

Add a statically assigned implementer to a resource.

```
struct ResourceImplementerAdd_scalarInput
    {
    var $resourceid
    var $restype
    var $userid
    };
```


Parameters

resourceid
[in] The ID of the resource.

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList.

userid
[in] Add this implementer.

7.2.167 ResourceImplementerDelete

Delete an implementer statically assigned to a resource.

```
struct ResourceImplementerDelete_scalarInput
    {
    var $resourceid
    var $restype
    var $userid
    };
```

Parameters

resourceid
[in] The ID of the resource.

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList.

userid
[in] Delete this implementer.

7.2.168 ResourceImplementerList

Lists the implementers statically assigned to a resource.

```
struct ResourceImplementerList_scalarInput
    {
    var $resourceid
    var $restype
    };

struct ResourceImplementerList_vectorOutput
    {
    var $userid
    };
```

Parameters

resourceid
[in] The ID of the resource.

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList.

userid
[out] The profile ID of the implementer.

7.2.169 ResourceOperActionGet

Returns the action of the given operation(s) of the given resource. If the operation isn't specified, then the actions of all operations of the given resource are returned.

```
struct ResourceOperActionGet_scalarInput
    {
    var $resid
    var $restype
    struct ResourceOperActionGet_vectorInput $inputArray[]
    };

struct ResourceOperActionGet_vectorInput
    {
    var $opertype
    };

struct ResourceOperActionGet_vectorOutput
    {
    var $opertype
    var $action
    };
```

Parameters

resid
[in] The ID of the resource.

restype
[in] The type of the resource. Note that the type is one of the following: TARG and TMPL.

inputArray
[in] An array of structs of type ResourceOperActionGet_vectorInput

opertype
[in] The type of operation. If restype is TARG, then opertype can be ACUA (Create account), DELU (Delete account), DNAU (Disable account), ENAU (Enable account), MVCU (Move account from one context to another), RENU (Rename user), UPDT (Update account), GRUA (Add group membership), GRUD (Delete group membership), GROA (Add group owner), GROD (Delete group owner), GRGA (Add child group) or GRGD (Delete child group). If restype is TMPL, then opertype is ACUA (Create account).

opertype
[out] The type of operation.

action
[out] The type of action. Can be NONE (No override), AGENT (Connector operation), or IMPL (Implementation operation).

7.2.170 ResourceOperActionSet

Sets the action of the given operation(s) of the given resource.

```
struct ResourceOperActionSet_scalarInput
    {
    var $resid
    var $restype
    struct ResourceOperActionSet_vectorInput $inputArray[]
    };

struct ResourceOperActionSet_vectorInput
    {
    var $opertype
    var $action
    };
```

Parameters

resid
[in] The ID of the resource.

restype
[in] The type of the resource. Note that the type is one of the following: TARG and TMPL.

inputArray
[in] An array of structs of type ResourceOperActionSet_vectorInput

opertype
[in] The type of operation. If restype is TARG, then opertype can be ACUA (Create account), DELU (Delete account), DNAU (Disable account), ENAU (Enable account), MVCU (Move account from one context to another), RENU (Rename user), UPDT (Update account), GRUA (Add group membership), GRUD (Delete group membership), GROA (Add group owner), GROD (Delete group owner), GRGA (Add child group) or GRGD (Delete child group). If restype is TMPL, then opertype is ACUA (Create account).

action
[in] The type of action. Can be NONE (No override), AGENT (Connector operation), or IMPL (Implementation operation).

7.2.171 ResourcePost

Posts the given resource and its configured attributes for use in Bravura Security Fabric. Upon successful creation of the resource, the handle to the resource becomes invalid.

```
struct ResourcePost_scalarInput
    {
    var $reshandle
    };
```

Parameters

reshandle
[in] The handle of the resource, as returned by ResourceCreate.

7.2.172 ResourceRead

Read out the values of a specified resource. Requires one or more key-value pairs to identify a single resource. The required keys are those returned by ResourceAttrNamesList where isrequiredread is true.

```
struct ResourceRead_scalarInput
    {
    var $restype
    struct ResourceRead_vectorInput $inputArray[]
    };

struct ResourceRead_vectorInput
    {
    var $name
    var $value
    };

struct ResourceRead_vectorOutput
    {
    var $name
    var $value
    var $type
    };
```

Parameters

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList where readonly is zero.

inputArray
[in] An array of structs of type ResourceRead_vectorInput

name
[in] The name of the parameter.

value
[in] The value of the parameter.

name
[out] The name of the parameter.

value
[out] The value of the parameter.

type
[out] How this parameter should be interpreted. Possible values are "string", "boolean" (1 or 0), "integer", and "kvg".

7.2.173 ResourceSet

Sets the parameters associated with the given existing resource. "resourceid" is optional to identify the resource to update; if empty, the resource is located using parameters in the vector of input values.

See Resource Configuration Parameters for the complete list of resource types and their parameters supported by this function.

```
struct ResourceSet_scalarInput
    {
    var $resourceid
    var $restype
    struct ResourceSet_vectorInput $inputArray[]
    };

struct ResourceSet_vectorInput
    {
    var $name
    var $value
    };
```

Parameters

resourceid
[in] The ID of the resource.

restype
[in] The type of resource. This must be one of the types returned by ResourceTypesList where readonly is zero.

inputArray
[in] An array of structs of type ResourceSet_vectorInput

name
[in] The name of the parameter.

value

[in] The value of the parameter.

7.2.174 ResourceTypesList

Lists supported resource types for use with ResourceAttrNamesList, ResourceCreate, ResourceFind, ResourceFindCriteria, ResourceGet, and ResourceSet.

```
struct ResourceTypesList_scalarInput
    {
    var $unused


    };

struct ResourceTypesList_vectorOutput
    {
    var $name
    var $description
    var $readonly
    };
```

Parameters

unused
[in] Not used

name
[out] The name of the resource type.

description
[out] The description of the resource type.

readonly
[out] Zero if the resource type can be created with ResourceCreate and updated with ResourceSet.

7.2.175 RetrieveCheckoutPassword

Retrieve the password for an account (given wstn and account) currently checked out (given a check-out id).

```
struct RetrieveCheckoutPassword_scalarInput
    {
    var $checkoutid
    var $wstnid
    var $accountname
    };

struct RetrieveCheckoutPassword_vectorOutput
    {
```



```
    var $password
    var $passwordstatus
    var $status
    var $errmsg
    };
```

Parameters

checkoutid
[in] Check-out ID.

wstnid
[in] The workstation ID of the account.

accountname
[in] The account ID.

password
[out] The password for this account, if the checkout was retrieved.

passwordstatus
[out] The password status.

status
[out] The status of this get password attempt.

errmsg
[out] The error message for this get password attempt.

7.2.176 RoleAuthorizerAdd

Adds the specified authorizer to the specified role's defintion.

```
struct RoleAuthorizerAdd_scalarInput
    {
    var $phase
    var $roleid
    var $authorizerid
    };
```

Parameters

phase
[in] Phase in which the authorizer will be asked to respond (for single-phase authorization, specify 1).

roleid
[in] The role's identifier.

authorizerid
[in] The authorizer's identifier.

7.2.177 RoleAuthorizerDelete

Deletes the specified authorizer from the specified role's definition.

```
struct RoleAuthorizerDelete_scalarInput
    {
    var $roleid
    var $authorizerid
    var $phase
    };
```

Parameters

roleid
[in] The role's identifier.

authorizerid
[in] The authorizer's identifier.

phase
[in] The authorizer's phase (specify 1 if phases are not enabled).

7.2.178 RoleAuthorizerList

Retrieves a list of all the authorizers assigned to the specified role. The returned list may be empty if there are no authorizers assigned to the role.

```
struct RoleAuthorizerList_scalarInput
    {
    var $roleid
    };

struct RoleAuthorizerList_vectorOutput
    {
    var $phase
    var $authorizerid
    var $authorizername
    };
```

Parameters

roleid
[in] The role's identifier.

phase
[out] Phase in which the authorizer will be asked to respond.

authorizerid
[out] The ID of the role's authorizer.

authorizername
[out] The name of the role's authorizer.

7.2.179 RoleCreate

Creates a new role without any member resources.

All role IDs will be converted to upper case.

RoleCreate will return one of the following:

- API_SUCCESS: The new role has been successfully created based on the provided parameter values.
- ERR_DESC_MANDATORY: The role must have a non-empty description.
- ERR_UNKNOWN: The operation encountered an unknown internal error. It is advisable to check the logs for both iddb.exe and idapi.exe to help determine the cause.
- ERR_ROLE_ALREADY_EXISTS: The specified role identifier already exists.

```
struct RoleCreate_scalarInput
    {
    var $roleid
    var $description
    var $isEnabled
    var $isAssignable
    var $isEnforced
    };
```

Parameters

roleid
[in] The unique identifier for the role to create.

description
[in] The role's description. It must be non-empty or else the creation request will be denied with ERR_DESC_MANDATORY.

isEnabled
[in] Is the role enabled? (0) is not enabled, (1) is enabled.

isAssignable
[in] Is the role assignable? (0) is not assignable, (1) is assignable.

isEnforced
[in] Is the role enforced? (0) is not enforced, (1) is enforced.

7.2.180 RoleDelete

Deletes an existing role.

```
struct RoleDelete_scalarInput
    {
    var $roleid
    };
```

Parameters

roleid
[in] The role's identifier.

7.2.181 RoleGetByResource

Retrieves a list of all roles that have the given resource as a member resource.

```
struct RoleGetByResource_scalarInput
    {
    var $membertype
    var $memberid
    var $groupid
    };

struct RoleGetByResource_vectorOutput
    {
    var $roleid
    };
```

Parameters

membertype
[in] The type of the member resource. This must be one of the following types: ROLE, TEMPLATE or MANAGEDGROUP.

memberid
[in] The identifier of the member resource to query. If membertype is MANAGEDGROUP, then memberid is the ID of the target.

groupid
[in] The identifier of the managed group associated with the memberid. This parameter is only used when membertype is MANAGEDGROUP.

roleid
[out] The role's identifier.

7.2.182 RoleGetByUser

Retrieves a list of all roles that have the given user as a member.

```
struct RoleGetByUser_scalarInput
    {
    var $userid
    };

struct RoleGetByUser_vectorOutput
    {
    var $roleid
    };
```

Parameters

userid
[in] The identifier of the user to query roles.

roleid
[out] The role's identifier.

7.2.183 RoleList

Gets a list of all roles.

RoleList will return one of the following:

- API_SUCCESS: The operation was successfully executed. The vectorOutput can have 0 to n elements.
- ERR_UNKNOWN: The operation encountered an unknown internal error. It is advisable to check the logs for both iddb.exe and idapi.exe to help determine the cause.
- ERR_NO_SUCH_USER The specified user identifier does not exist. Verify that the spelling is correct.

```
struct RoleList_scalarInput
    {
    var $unused


    };

struct RoleList_vectorOutput
    {
    var $roleid
    };
```

Parameters

unused
[in] Not used

roleid
[out] The role's identifier.

7.2.184 RoleRemovableCheck

Check if a role can be removed, return success if role can be removed.

```
struct RoleRemovableCheck_scalarInput
    {
    var $roleid
    var $disabledonly
    };

struct RoleRemovableCheck_vectorOutput
    {
    var $status
    };
```

Parameters

roleid
[in] The role's identifier.

disabledonly
[in] Ensure role is disabled.

status
[out] Whether the role can be removed: 0: role can be removed 1: role does not exist 2: role is enabled 3: role has users assigned 4: role has dependents

7.2.185 RoleResourceAdd

Adds the specified resource as member to the given role. Both the role and resource must exist before using this API call. It will not create a missing resource.

RoleResourceAdd will return one of the following:

- API_SUCCESS: The specified resource member was successfully added to the role.
- ERR_UNKNOWN: The operation encountered an unknown internal error. It is advisable to check the logs for both iddb.exe and idapi.exe to help determine the cause.
- ERR_INVALID_MEMBER_TYPE: The specified member type is not supported. It must be one of the following: ROLE, TEMPLATE and MANAGEDGROUP.
- ERR_NO_SUCH_TEMPLATE: The specified template account does not exist. Verify that it exists or its spelling.

- ERR_NO_SUCH_GROUP: The specified group does not exist. Verify that it exists, that it is managed, its spelling or that it is a group on the specified target.
- ERR_NO_SUCH_ROLE: Either the parent role (roleid) or the subrole (memberid) does not exist. Verify that the spelling is correct.
- ERR_ROLE_CYCLE: The specified subrole (memberid) is the same as the parent role (roleid).
- ERR_RESOURCE_EXISTS: The resource being added is already a member of the role.

```
struct RoleResourceAdd_scalarInput
    {
    var $roleid
    var $membertype
    var $memberid
    var $groupid
    var $optional
    };
```

Parameters

roleid
[in] The role's identifier.

membertype
[in] The type of the member resource. This must be one of the following types: ROLE, TEMPLATE or MANAGEDGROUP.

memberid
[in] The identifier of the existing member resource to add. If membertype is MANAGEDGROUP, then the memberid is the ID of the target.

groupid
[in] The identifier of the managed group associated with the memberid. This parameter is only used when membertype is MANAGEDGROUP.

optional
[in] The resource is optional if the vlaue is true.

7.2.186 RoleResourceDelete

Deletes the specified member resource from the role's definition. Deletion will only occur if both the role and resource member exist. Does not remove the member resource.

```
struct RoleResourceDelete_scalarInput
    {
    var $roleid
    var $membertype
    var $memberid
    var $groupid
    };
```


Parameters

roleid
[in] The role's identifier.

membertype
[in] The type of the member resource. This must be one of the following types: ROLE, TEMPLATE or MANAGEDGROUP.

memberid
[in] The identifier of the member resource to delete. If membertype is MANAGEDGROUP, then the memberid is the ID of the target.

groupid
[in] The identifier of the managed group associated with the memberid. This parameter is only used when membertype is MANAGEDGROUP.

7.2.187 RoleResourceList

Retrieves a list of all member resources assigned to the specified role. The returned list may be empty if there are no resource members defined for the specified role.

```
struct RoleResourceList_scalarInput
    {
    var $roleid
    var $required
    var $optional
    var $legacy
    };

struct RoleResourceList_vectorOutput
    {
    var $memberid
    var $groupid
    var $membertype
    var $optional
    };
```

Parameters

roleid
[in] The role's identifier.

required
[in] Return Required resources.

optional
[in] Return optional resources.

legacy

[in] Return legacy resources.

memberid
[out] The member resource's identifier. For managed group members, this is the target ID.

groupid
[out] The group ID, for managed group members.

membertype
[out] The member resource's type. This must be one of the following: ROLE, TEMPLATE or MANAGED-GROUP.

optional
[out] Nonzero if the resource is optional in the role.

7.2.188 RoleUpdate

Updates an existing role with the provided values.

RoleUpdate will return one of the following:

- API_SUCCESS: The new role has been successfully created based on the provided parameter values.
- ERR_DESC_MANDATORY: The role must have a non-empty description.
- ERR_UNKNOWN: The operation encountered an unknown internal error. It is advisable to check the logs for both iddb.exe and idapi.exe to help determine the cause.
- ERR_NO_SUCH_ROLE: The specified role identifier does not exist. Verify that the spelling is correct.

```
struct RoleUpdate_scalarInput
    {
    var $roleid
    var $description
    var $isEnabled
    var $isAssignable
    var $isEnforced
    var $isDeprecated
    var $deprecatedReason
    };
```

Parameters

roleid
[in] The identifier for the role to update.

description
[in] The role's description. It must be non-empty or else the update request will be denied with ERR_DESC_MANDATORY.

isEnabled
[in] Is the role enabled? (0) is not enabled, (1) is enabled.

isAssignable
[in] Is the role assignable? (0) is not assignable, (1) is assignable.

isEnforced
[in] Is the role enforced? (0) is not enforced, (1) is enforced.

isDeprecated
[in] Is the role deprecated? (0) is not deprecated, (1) is deprecated.

deprecatedReason
[in] The reason for deprecating the role.

7.2.189 RoleUserList

Retrieves a list of all the users that are members of the specified role. The returned list may be empty if there are no user members for the role.

```
struct RoleUserList_scalarInput
    {
    var $roleid
    };

struct RoleUserList_vectorOutput
    {
    var $userid
    var $username
    };
```

Parameters

roleid
[in] The role ID.

userid
[out] A role member user's identifier.

username
[out] A role member user's name.

7.2.190 RunReport

Initiate and run a report and stream the results. Can save to or load from an existing report.

```
struct RunReport_scalarInput
    {
    var $reporttype
```



```
    var $saveas
    var $loadfrom
    var $numresults
    var $returns
    var $options
    struct RunReport_vectorInput $inputArray[]
    };

struct RunReport_vectorInput
    {
    var $searchfield
    var $searchvalue
    };

struct RunReport_vectorOutput
    {
    var $metadata
    var $value
    };
```

Parameters

reporttype
[in] The report to be run.

saveas
[in] (Optional) If this value is not empty, run the report and save report with this value as the saved report name. This will overwrite any existing saved reports with the same name. This will not take effect if the "loadfrom" input option is specified.

loadfrom
[in] (Optional) If this value is not empty, load data from the report with this name.

numresults
[in] (Optional) Maximum number of results (records) to return through the API. If less than 1 or not specified, indicates all records to be returned.

returns
[in] (Optional)
ROW: Returns results one record at a time (default if undefined).
CELL: Returns results one data value of a record at a time.

options
[in] (Optional) A string of options to be run with the report. This needs to be given in a KVG structure where the key is the option to set.
Valid email options:
to: Comma-separated list of email addresses
cc: Comma-separated list of email addresses
bcc: Comma-separated list of email addresses
attachmenttype: CSV (default if not indicated), HTM, or PDF
subject: Email subject line
message: Email body

Valid UNC options:
filetype: CSV (default if not indicated), HTM, or PDF
unc: UNC address
filename: Filename to save as
credtarget: Target system ID
creduser: Administrator ID

inputArray
[in] An array of structs of type RunReport_vectorInput

searchfield
[in] Input criteria name that maps to an input widget name.

searchvalue
[in] Value to search against. This is set against the widget name chosen as the searchfield.

metadata
[out] The data returned the report with header information as follows:
"reportid": ID of the report
"headercount": Number of header values
"header": Identifies that the value is a header title
"recordcount": Number of records to return
Each row will be an indexed list of records
<record index>: Index of current record value being returned

value
[out] If "returns" input is set to CELL, this value contains individual data values.
Example: one
If "returns" input is set to ROW, this value contains a comma-separated, quoted list of values representing one record.
Example: "one","two","three"

7.2.191 SavedSessionsList

List saved sessions.

```
struct SavedSessionsList_scalarInput
    {
    var $disclosureguid
    var $profileguid
    };

struct SavedSessionsList_vectorOutput
    {
    var $id
    var $profileguid
    var $clonedescription
    var $categorydesc
    var $notes
```



```
    var $resgrpid
    var $pluginctrlid
    var $wstntoken
    var $accountguid
    };
```

Parameters

disclosureguid
[in] Guid of the disclosure.

profileguid
[in] Optional. If we want to get the saved sessions from this user.

id
[out] The identifier for the saved sesssion.

profileguid
[out] The guid for the user's saved session.

clonedescription
[out] The name of the saved session.

categorydesc
[out] The category of the saved session.

notes
[out] The notes of the saved session.

resgrpid
[out] The MSP associated to the managed account's saved sesssion.

pluginctrlid
[out] The disclosure guid associated to the saved session.

wstntoken
[out] The workstation associated to the managed account's saved sesssion.

accountguid
[out] The managed account associated to the saved session.

7.2.192 ScheduledTaskStatus

Provides information on scheduled tasks.

```
struct ScheduledTaskStatus_scalarInput
    {
    var $unused


    };
```



```
struct ScheduledTaskStatus_vectorOutput
    {
    var $taskid
    var $job
    var $enabled
    var $serverid
    var $laststart
    var $lastfinish
    var $lasterror
    var $laststatus
    };
```

Parameters

unused
[in] Not used

taskid
[out] Identifier for the scheduled task.

job
[out] Command line and arguments to be executed.

enabled
[out] "1" if the task is enabled, "0" if disabled.

serverid
[out] Server ID that is configured to run this task. If empty, the task is run by all servers.

laststart
[out] Latest date and time the task started. If empty, the task has never started.

lastfinish
[out] Latest date and time the task finished. If empty, the task has never finished.

lasterror
[out] Error code returned from the job when it last ran.

laststatus
[out] Status of the latest task run. It can be one of the following: - RUNNING - FINISHED - FAILED - NOTRUN - LAUNCHED If empty, the status is unknown.

7.2.193 SessionDataCustomGet

Get the value from the custom session tag

```
struct SessionDataCustomGet_scalarInput
    {
    var $sessid
    var $tag
```



```
    };

struct SessionDataCustomGet_vectorOutput
    {
    var $value
    };
```

Parameters

sessid
[in] The session id

tag
[in] The tag name

value
[out] The tag value

7.2.194 SessionDataCustomSet

Set the value to the custom session tag

```
struct SessionDataCustomSet_scalarInput
    {
    var $sessid
    var $tag
    var $value
    };
```

Parameters

sessid
[in] The session id

tag
[in] The tag name

value
[in] The tag value

7.2.195 SessionDataGet

Get the value from the session tag

```
struct SessionDataGet_scalarInput
    {
    var $sessid
    var $tag
    };
```



```
struct SessionDataGet_vectorOutput
    {
    var $value
    };
```

Parameters

sessid
[in] The session id

tag
[in] The tag name

value
[out] The tag value

7.2.196 SessionDataGetFull

Get the full value from the session tag in chunks. Useful for large values

```
struct SessionDataGetFull_scalarInput
    {
    var $sessid
    var $tag
    };

struct SessionDataGetFull_vectorOutput
    {
    var $value
    };
```

Parameters

sessid
[in] The session id

tag
[in] The tag name

value
[out] The tag value. Sent in consecutive chunks which need to be reassembled

7.2.197 SessionIsValid

Validates a session by checking the session data. If there is a problem with the given session data, the error code may be ERR_SESSID_EXPIRED, ERR_SESSID_BAD_LICENSE, or ERR_SESSID_NOT_FOUND.

```
struct SessionIsValid_scalarInput
    {
```



```
    var $requesterid
    var $isurlencoded
    struct SessionIsValid_vectorInput $inputArray[]
    };

struct SessionIsValid_vectorInput
    {
    var $sessdata
    };
```

Parameters

requesterid
[in] The requester's profile ID.

isurlencoded
[in] Whether sessdata has been URL-encoded. 0 is unencoded, anything else is encoded.

inputArray
[in] An array of structs of type SessionIsValid_vectorInput

sessdata
[in] The session data. If the value is larger than 16383 characters, then it should be broken up into smaller segments and passed to the function, which will reassemble the segments in the order that they were passed. This parameter cannot be used to evaluate multiple session data values.

7.2.198 SubscriberList

List subscribers and their associated accounts. All of the input parameters are optional. If no parameters are provided, all subscribers in all managed systems will be listed. This function deprecates ManagedAccountSubscriberList.

```
struct SubscriberList_scalarInput
    {
    var $SubscriberAddress
    var $SubscriberType
    var $SubscriberTarget
    var $AccountTarget
    var $AccountLongID
    var $AccountShortID
    };

struct SubscriberList_vectorOutput
    {
    var $id
    var $targetid
    var $targetdesc
    var $accountid
    var $subscribertype
    var $objectname
```



```
var $objectsid
var $longid
var $hostid
};
```

Parameters

SubscriberAddress
[in] The managed object name, such as the service name, the IIS site name, the scheduled task name, the group name or the custom object name.

SubscriberType
[in] The subscriber type. Possible values are SVC for managing services, IIS for managing IIS, TAS for managing scheduled tasks, GRP for security group members, COM for managing DCOM, and CUS for managing custom accounts.

SubscriberTarget
[in] The target system the subscriber is from.

AccountTarget
[in] The target system the associated account is from.

AccountLongID
[in] The long ID of the associated account.

AccountShortID
[in] The short ID of the associated account.

id
[out] The id of the subscriber.

targetid
[out] The target system the subscriber is from.

targetdesc
[out] The description of the target system the subscriber is from.

accountid
[out] The associated account ID.

subscribertype
[out] The subscriber type. Possible values are SVC for managing services, IIS for managing IIS, TAS for managing scheduled tasks, GRP for security group members, COM for managing DCOM, and CUS for managing custom accounts.

objectname
[out] The managed object name, such as the service name, the IIS site name, the scheduled task name, the group name or the custom object name.

objectsid
[out] The SID of the object.

longid
[out] The long ID of the associated account.

hostid
[out] The target system the associated account is from.

7.2.199 TargetAttributeGet

Returns the attribute mapping between the product and its targets. The input parameters act as filters for the return values. If empty filters are specified, all attribute definitions are returned without filtering. A zero length filter text equals not specified. If the 'targetid' filter is specified, then any 'platformid' input parameter is always ignored, and the platform that will be filtered is the default platform for the specified 'targetid'. The result will include:

- target level overrides and mappings
- target type level overrides and mappings
- default mappings to profile attributes

If the 'targetid' filter is not specified and 'platformid' input filter is specified, then the result will not include attributes that are defined on the Target Level. It will be filtered to contain the attributes defined for the specified platform and default mappings only.

This function accepts and/or returns platform type ID values. See Platform Type ID Values for the list of possible values.

```
struct TargetAttributeGet_scalarInput
    {
    var $targetattrid
    var $targetid
    var $platformid
    var $profileattrid
    };

struct TargetAttributeGet_vectorOutput
    {
    var $targetattrid
    var $targetid
    var $platformid
    var $profileattrid
    var $override
    var $createaction
    var $updateaction
    var $listattr
    var $setuserattr
    var $makediffs
    };
```

Parameters

targetattrid
[in] (Optional) The target attribute ID.

targetid
[in] (Optional) The target ID.

platformid
[in] (Optional) The platform ID.

profileattrid
[in] (Optional) The profile attribute ID.

targetattrid
[out] The target attribute ID.

targetid
[out] The target ID.

platformid
[out] The platform ID.

profileattrid
[out] The profile attribute ID.

override
[out] The override setting for the attribute. Override 1 is an override on Target Type level. Override 2 is an override on Target Level.

createaction
[out] Action to perform when creating account. Restricted values: C - Exact copy, I - Ignore, R - Copy replacing ID, S - Set.

updateaction
[out] Action to perform when updating account. Restricted values: S - Set always, C - Set when mapped profile attribute changes, I - Ignore.

listattr
[out] Do we list this attribute?

setuserattr
[out] Will this target attribute be used to set profile attribute values during auto-discovery?

makediffs
[out] Should loaddb produce diffs on this attribute, for use with idtrack?

7.2.200 TargetGet

Retrieves information for a specified target.

This function accepts and/or returns platform type ID values. See Platform Type ID Values for the list of possible values.


```
struct TargetGet_scalarInput
    {
    var $targetid
    };

struct TargetGet_vectorOutput
    {
    var $targetid
    var $name
    var $platform
    var $url
    var $address
    var $group
    var $include
    var $casegen
    var $listgen
    var $mandatory
    var $expusr
    var $delalias
    var $agenttime
    var $listtime
    var $lstminsize
    var $proxy
    var $adminresethide
    var $claimalias
    var $disablereset
    var $selfresethide
    var $disableunlock
    var $adminunlockhide
    var $selfunlockhide
    var $adminclaimhide
    var $selfclaimhide
    var $mustselect
    var $disableverify
    var $isapp
    var $selfmanagehide
    var $idarchivepush
    var $idcheck
    var $createdon
    var $contattr
    var $updserver
    var $managegrp
    var $nrplatform
    var $rbacenforce
    var $saction
    var $daction
    var $type
    };
```

Parameters

targetid
[in] The ID of the target system.

targetid
[out] The ID of the target system.

name
[out] Description of the target system.

platform
[out] The type of the target system.

url
[out] (Optional) URL with target information for users.

address
[out] The network address of the target system.

group
[out] The target system group this target system belongs to.

include
[out] Whether to include IDs if no IDInclude rule applies. 1 is true, 0 is false.

casegen
[out] (Optional) Default or external program to generate new IDs in correct case (all upper, all lower, etc.).

listgen
[out] (Optional) External program to convert this target system into a list of other target systems.

mandatory
[out] Whether users must have at least one account on this target. 1 is true, 0 is false.

expusr
[out] Whether to find users whose passwords will expire soon during auto discovery. 1 is true, 0 is false.

delalias
[out] Whether users are allowed to delete accounts on this target. X is DEFAULT, T is ALLOW and F is BLOCK.

agenttime
[out] Agent timeout, in seconds.

listtime
[out] List timeout, in seconds.

lstminsize
[out] Minimum list file size, in bytes.

proxy
[out] Comma-delimited list of proxies.

adminresethide

[out] Whether to allow administrators to change passwords using the Bravura Security Fabric interface. 1 is deny, 0 is allow.

claimalias
[out] Whether to allow other users to claim auto-associated accounts. 1 is allow, 0 is deny.

disablereset
[out] Whether to allow users to reset their passwords. 1 is deny, 0 is allow.

selfresethide
[out] Whether to allow users to change passwords using the Bravura Security Fabric interface. 1 is deny, 0 is allow.

disableunlock
[out] Whether to allow users to unlock their accounts. 1 is deny, 0 is allow.

adminunlockhide
[out] Whether to allow administrators to unlock accounts using the Bravura Security Fabric interface. 1 is deny, 0 is allow.

selfunlockhide
[out] Whether to allow users to unlock accounts using the Bravura Security Fabric interface. 1 is deny, 0 is allow.

adminclaimhide
[out] Whether to allow administrators to attach accounts using the Bravura Security Fabric interface. 1 is deny, 0 is allow.

selfclaimhide
[out] Whether to allow users to attach accounts using the Bravura Security Fabric interface. 1 is deny, 0 is allow.

mustselect
[out] Whether accounts must be included in password changes. 1 is true, 0 is false.

disableverify
[out] Whether to allow users to verify passwords. 1 is deny, 0 is allow.

isapp
[out] Whether this target system is allowed in the certification process. 1 is allowed, 0 is denied.

selfmanagehide
[out] Whether to allow users to update profile information and resources using the Bravura Security Fabric interface. 1 is deny, 0 is allow.

idarchivepush
[out] Whether a Bravura Privilege managed system should be automatically created for this target. 1 is true, 0 is false.

idcheck
[out] Whether accounts should be checked for uniqueness when creating new profile IDs. 1 is true, 0 is false.

createdon
[out] The ISO-formatted UTC time this target system was created.

contattr
[out] Attribute holding the value for _container_dn for this target system.

updserver
[out] Server ID that will update this target.

managegrp
[out] Auto-manage groups. N is NONE, O is OPEN, and M is MODERATED.

nrplatform
[out] Type of network resource.

rbacenforce
[out] Whether to enforce the target system under RBAC roles. 1 is true, 0 is false.

saction
[out] Default action for surpluses of this resource. This will be one of the following actions: ADD, EXCEPTION, INHERIT, or REMOVE.

daction
[out] Default action for deficits of this resource. This will be one of the following actions: ADD, EXCEPTION, INHERIT, or REMOVE.

type
[out] The target type for auto-discovery. M is a manually added target, A is an auto-discovered target, T is A target template.

7.2.201 TokenPINReset

Reset the PIN on a secure authentication token.

```
struct TokenPINReset_scalarInput
    {
    var $userid
    var $targetid
    var $tokenid
    var $newpin
    };

struct TokenPINReset_vectorOutput
    {
    var $newpin
    };
```

Parameters

userid

[in] The user's profile ID.

targetid
[in] The target containing the authentication token.

tokenid
[in] The token ID.

newpin
[in] The new PIN for the token. If this value is not specified, a random PIN is generated and returned as output for this function.

newpin
[out] The specified new PIN or auto-generated new PIN.

7.2.202 TokenResynchronize

Resynchronize a secure authentication token, by supplying a current passcode that appears on the token as well as the following passcode that appears.

```
struct TokenResynchronize_scalarInput
    {
    var $userid
    var $targetid
    var $tokenid
    var $firstcode
    var $secondcode
    };
```

Parameters

userid
[in] The user's profile ID.

targetid
[in] The target containing the authentication token.

tokenid
[in] The token ID.

firstcode
[in] The first passcode.

secondcode
[in] The second passcode.

7.2.203 TranslationGet

Retrieves a translated string from the errmsg.kvg file. If translated text does not exist for the specified language, the untranslated tag is returned.

```
struct TranslationGet_scalarInput
    {
    var $tag
    var $language
    };

struct TranslationGet_vectorOutput
    {
    var $translation
    };
```

Parameters

tag
[in] The tag to look up.

language
[in] The language identifier, composed of language and region. e.g. en-us

translation
[out] The translated string

7.2.204 UCPGeneric1Check

Tests if a user is an actor in a UCP

```
struct UCPGeneric1Check_scalarInput
    {
    var $ucpid
    var $participant
    };

struct UCPGeneric1Check_vectorOutput
    {
    var $ismember
    };
```

Parameters

ucpid
[in] UCP ID

participant
[in] Profile ID to test that maps to the participant

ismember
[out] Whether the participant is a member of the UCP

7.2.205 UCPGeneric1List

Lists the members of the specified UCP actor

```
struct UCPGeneric1List_scalarInput
    {
    var $ucpid
    };

struct UCPGeneric1List_vectorOutput
    {
    var $member
    };
```

Parameters

ucpid
[in] UCP ID

member
[out] Profile ID of the member of the UCP

7.2.206 UCPGeneric1UCAdd

Add user class to a single-actor generic UCP and associates it to the participant

```
struct UCPGeneric1UCAdd_scalarInput
    {
    var $ucpid
    var $ucid
    };

struct UCPGeneric1UCAdd_vectorOutput
    {
    var $ucpmembershipid
    };
```

Parameters

ucpid
[in] UCP ID

ucid
[in] UC ID

ucpmembershipid

[out] Unique identifier for this userclass mapping.

7.2.207 UCPGeneric1UserclassSet

Change participant mappings on an existing user class in a single-actor UCP.

```
struct UCPGeneric1UserclassSet_scalarInput
    {
    var $ucpid
    var $ucpmembershipid
    var $mapToParticipant
    };
```

Parameters

ucpid
[in] UCP ID

ucpmembershipid
[in] Unique identifier for this userclass mapping.

mapToParticipant
[in] The user class participant to which Participant should be re-mapped. Leave this blank if there is no user class participant to which the user class point participant should be mapped.

7.2.208 UCPGeneric2Check

Tests if a user is an actor in a UCP

```
struct UCPGeneric2Check_scalarInput
    {
    var $ucpid
    var $participant1
    var $participant2
    };

struct UCPGeneric2Check_vectorOutput
    {
    var $ismember
    };
```

Parameters

ucpid
[in] UCP ID

participant1
[in] Profile ID to test that maps to the first participant

participant2
[in] Profile ID to test that maps to the second participant

ismember
[out] Whether the participant is a member of the UCP

7.2.209 UCPGeneric2ListParticipant1

Lists the profiles that match the first participant given the other participants.

```
struct UCPGeneric2ListParticipant1_scalarInput
    {
    var $ucpid
    var $participant2
    };

struct UCPGeneric2ListParticipant1_vectorOutput
    {
    var $member
    };
```

Parameters

ucpid
[in] UCP ID

participant2
[in] other actor

member
[out] Profile ID of the first particpant of the UCP

7.2.210 UCPGeneric2ListParticipant2

Lists the profiles that match the second participant given the other participants.

```
struct UCPGeneric2ListParticipant2_scalarInput
    {
    var $ucpid
    var $participant1
    };

struct UCPGeneric2ListParticipant2_vectorOutput
    {
    var $member
    };
```

Parameters

ucpid
[in] UCP ID

participant1
[in] other actor

member
[out] Profile ID of the second particpant of the UCP

7.2.211 UCPGeneric2UCAdd

Adds user classes to a dual-actor generic UCP and associates them to participants

```
struct UCPGeneric2UCAdd_scalarInput
    {
    var $ucpid
    var $ucid
    var $mapto1
    var $mapto2
    };

struct UCPGeneric2UCAdd_vectorOutput
    {
    var $ucpmembershipid
    };
```

Parameters

ucpid
[in] UCP ID

ucid
[in] UC ID

mapto1
[in] The user class participant to which Participant1 should be mapped. Leave this blank if there is no user class participant to which the user class point participant should be mapped.

mapto2
[in] The user class participant to which Participant2 should be mapped. Leave this blank if there is no user class participant to which the user class point participant should be mapped.

ucpmembershipid
[out] Unique identifier for this userclass mapping.

7.2.212 UCPGeneric2UserclassSet

Change participant mappings on an existing user class in a two-actor UCP.


```
struct UCPGeneric2UserclassSet_scalarInput
    {
    var $ucpid
    var $ucpmembershipid
    var $mapToParticipant1
    var $mapToParticipant2
    };
```

Parameters

ucpid
[in] UCP ID

ucpmembershipid
[in] Unique identifier for this userclass mapping.

mapToParticipant1
[in] The user class participant to which Participant1 should be re-mapped. Leave this blank if there is no user class participant to which the user class point participant should be mapped.

mapToParticipant2
[in] The user class participant to which Participant2 should be re-mapped. Leave this blank if there is no user class participant to which the user class point participant should be mapped.

7.2.213 UCPGeneric3Check

Tests if a user is an actor in a UCP

```
struct UCPGeneric3Check_scalarInput
    {
    var $ucpid
    var $participant1
    var $participant2
    var $participant3
    };

struct UCPGeneric3Check_vectorOutput
    {
    var $ismember
    };
```

Parameters

ucpid
[in] UCP ID

participant1
[in] Profile ID to test that maps to the first participant

participant2

[in] Profile ID to test that maps to the second participant

participant3
[in] Profile ID to test that maps to the third participant

ismember
[out] Whether the participant is a member of the UCP

7.2.214 UCPGeneric3ListParticipant1

Lists the profiles that match the first participant given the other participants.

```
struct UCPGeneric3ListParticipant1_scalarInput
    {
    var $ucpid
    var $participant2
    var $participant3
    };

struct UCPGeneric3ListParticipant1_vectorOutput
    {
    var $member
    };
```

Parameters

ucpid
[in] UCP ID

participant2
[in] other actor

participant3
[in] other actor

member
[out] Profile ID of the first particpant of the UCP

7.2.215 UCPGeneric3ListParticipant2

Lists the profiles that match the second participant given the other participants.

```
struct UCPGeneric3ListParticipant2_scalarInput
    {
    var $ucpid
    var $participant1
    var $participant3
    };
```



```
struct UCPGeneric3ListParticipant2_vectorOutput
    {
    var $member
    };
```

Parameters

ucpid
[in] UCP ID

participant1
[in] other actor

participant3
[in] other actor

member
[out] Profile ID of the second particpant of the UCP

7.2.216 UCPGeneric3ListParticipant3

Lists the profiles that match the third participant given the other participants.

```
struct UCPGeneric3ListParticipant3_scalarInput
    {
    var $ucpid
    var $participant1
    var $participant2
    };

struct UCPGeneric3ListParticipant3_vectorOutput
    {
    var $member
    };
```

Parameters

ucpid
[in] UCP ID

participant1
[in] other actor

participant2
[in] other actor

member
[out] Profile ID of the first particpant of the UCP

7.2.217 UCPGeneric3UCAdd

Adds user classes to a triple-actor generic UCP and associates them to participants

```
struct UCPGeneric3UCAdd_scalarInput
    {
    var $ucpid
    var $ucid
    var $mapto1
    var $mapto2
    var $mapto3
    };

struct UCPGeneric3UCAdd_vectorOutput
    {
    var $ucpmembershipid
    };
```

Parameters

ucpid
[in] UCP ID

ucid
[in] UC ID

mapto1
[in] The user class participant to which Participant1 should be mapped. Leave this blank if there is no user class participant to which the user class point participant should be mapped.

mapto2
[in] The user class participant to which Participant2 should be mapped. Leave this blank if there is no user class participant to which the user class point participant should be mapped.

mapto3
[in] The user class participant to which Participant3 should be mapped. Leave this blank if there is no user class participant to which the user class point participant should be mapped.

ucpmembershipid
[out] Unique identifier for this userclass mapping.

7.2.218 UCPGeneric3UserclassSet

Change participant mappings on an existing user class in a three-actor UCP.

```
struct UCPGeneric3UserclassSet_scalarInput
    {
    var $ucpid
    var $ucpmembershipid
    var $mapToParticipant1
```



```
    var $mapToParticipant2
    var $mapToParticipant3
    };
```

Parameters

ucpid
[in] UCP ID

ucpmembershipid
[in] Unique identifier for this userclass mapping.

mapToParticipant1
[in] The user class participant to which Participant1 should be re-mapped. Leave this blank if there is no user class participant to which the user class point participant should be mapped.

mapToParticipant2
[in] The user class participant to which Participant2 should be re-mapped. Leave this blank if there is no user class participant to which the user class point participant should be mapped.

mapToParticipant3
[in] The user class participant to which Participant3 should be re-mapped. Leave this blank if there is no user class participant to which the user class point participant should be mapped.

7.2.219 UCPGeneric4Check

Tests if a user is an actor in a UCP

```
struct UCPGeneric4Check_scalarInput
    {
    var $ucpid
    var $participant1
    var $participant2
    var $participant3
    var $participant4
    };

struct UCPGeneric4Check_vectorOutput
    {
    var $ismember
    };
```

Parameters

ucpid
[in] UCP ID

participant1
[in] Profile ID to test that maps to the first participant

participant2
[in] Profile ID to test that maps to the second participant

participant3
[in] Profile ID to test that maps to the third participant

participant4
[in] Profile ID to test that maps to the four participant

ismember
[out] Whether the participant is a member of the UCP

7.2.220 UCPGeneric4ListParticipant1

Lists the profiles that match the first participant given the other participants.

```
struct UCPGeneric4ListParticipant1_scalarInput
    {
    var $ucpid
    var $participant2
    var $participant3
    var $participant4
    };

struct UCPGeneric4ListParticipant1_vectorOutput
    {
    var $member
    };
```

Parameters

ucpid
[in] UCP ID

participant2
[in] other actor

participant3
[in] other actor

participant4
[in] other actor

member
[out] Profile ID of the first particpant of the UCP

7.2.221 UCPGeneric4ListParticipant2

Lists the profiles that match the second participant given the other participants.


```
struct UCPGeneric4ListParticipant2_scalarInput
    {
    var $ucpid
    var $participant1
    var $participant3
    var $participant4
    };

struct UCPGeneric4ListParticipant2_vectorOutput
    {
    var $member
    };
```

Parameters

ucpid
[in] UCP ID

participant1
[in] other actor

participant3
[in] other actor

participant4
[in] other actor

member
[out] Profile ID of the second particpant of the UCP

7.2.222 UCPGeneric4ListParticipant3

Lists the profiles that match the third participant given the other participants.

```
struct UCPGeneric4ListParticipant3_scalarInput
    {
    var $ucpid
    var $participant1
    var $participant2
    var $participant4
    };

struct UCPGeneric4ListParticipant3_vectorOutput
    {
    var $member
    };
```

Parameters

ucpid

[in] UCP ID

participant1
[in] other actor

participant2
[in] other actor

participant4
[in] other actor

member
[out] Profile ID of the third particpant of the UCP

7.2.223 UCPGeneric4ListParticipant4

Lists the profiles that match the fourth participant given the other participants.

```
struct UCPGeneric4ListParticipant4_scalarInput
    {
    var $ucpid
    var $participant1
    var $participant2
    var $participant3
    };

struct UCPGeneric4ListParticipant4_vectorOutput
    {
    var $member
    };
```

Parameters

ucpid
[in] UCP ID

participant1
[in] other actor

participant2
[in] other actor

participant3
[in] other actor

member
[out] Profile ID of the fourth particpant of the UCP

7.2.224 UCPGeneric4UCAdd

Adds user classes to a four-actor generic UCP and associates it to actors

```
struct UCPGeneric4UCAdd_scalarInput
    {
    var $ucpid
    var $ucid
    var $mapto1
    var $mapto2
    var $mapto3
    var $mapto4
    };

struct UCPGeneric4UCAdd_vectorOutput
    {
    var $ucpmembershipid
    };
```

Parameters

ucpid
[in] UCP ID

ucid
[in] UC ID

mapto1
[in] The user class participant to which Participant1 should be mapped. Leave this blank if there is no user class participant to which the user class point participant should be mapped.

mapto2
[in] The user class participant to which Participant2 should be mapped. Leave this blank if there is no user class participant to which the user class point participant should be mapped.

mapto3
[in] The user class participant to which Participant3 should be mapped. Leave this blank if there is no user class participant to which the user class point participant should be mapped.

mapto4
[in] The user class participant to which Participant4 should be mapped. Leave this blank if there is no user class participant to which the user class point participant should be mapped.

ucpmembershipid
[out] Unique identifier for this userclass mapping.

7.2.225 UCPGeneric4UserclassSet

Change participant mappings on an existing user class in a four-actor UCP.


```
struct UCPGeneric4UserclassSet_scalarInput
    {
    var $ucpid
    var $ucpmembershipid
    var $mapToParticipant1
    var $mapToParticipant2
    var $mapToParticipant3
    var $mapToParticipant4
    };
```

Parameters

ucpid
[in] UCP ID

ucpmembershipid
[in] Unique identifier for this userclass mapping.

mapToParticipant1
[in] The user class participant to which Participant1 should be re-mapped. Leave this blank if there is no user class participant to which the user class point participant should be mapped.

mapToParticipant2
[in] The user class participant to which Participant2 should be re-mapped. Leave this blank if there is no user class participant to which the user class point participant should be mapped.

mapToParticipant3
[in] The user class participant to which Participant3 should be re-mapped. Leave this blank if there is no user class participant to which the user class point participant should be mapped.

mapToParticipant4
[in] The user class participant to which Participant4 should be re-mapped. Leave this blank if there is no user class participant to which the user class point participant should be mapped.

7.2.226 UCPGenericCreate

Initializes a generic user class point

```
struct UCPGenericCreate_scalarInput
    {
    var $ucpid
    var $ucptype
    var $criteriap
    var $description
    };
```

Parameters

ucpid
[in] UCP ID

ucptype
[in] Valid pointids: GENERICSINGLE - single-actor ucp GENERICDOUBLE - double-actor ucp GENERICTRIPLE - triple-actor ucp GENERICFOUR - four-actor ucp

criteriap
[in] U - Union (match any user classes) I - Intersection (match all user class)

description
[in] Description of category (resgroup description)

7.2.227 UCPGenericDelete

Deletes a generic user class point

```
struct UCPGenericDelete_scalarInput
    {
    var $ucpid
    var $ucptype
    };
```

Parameters

ucpid
[in] UCP ID

ucptype
[in] Valid pointids: GENERICSINGLE - single-actor ucp GENERICDOUBLE - double-actor ucp GENERICTRIPLE - triple-actor ucp GENERICFOUR - four-actor ucp

7.2.228 UCPGenericGet

Gets UCP criteria

```
struct UCPGenericGet_scalarInput
    {
    var $ucpid
    var $ucptype
    };

struct UCPGenericGet_vectorOutput
    {
    var $criteriap
    var $description
    };
```

Parameters

ucpid
[in] UCP ID

ucptype
[in] Valid pointids: GENERICSINGLE - single-actor ucp GENERICDOUBLE - double-actor ucp GENERICTRIPLE - triple-actor ucp GENERICFOUR - four-actor ucp

criteriap
[out] Valid criteriaps: U - Union I - Intersection

description
[out] Description of category (resgroup description)

7.2.229 UCPGenericSet

Sets UCP criteria

```
struct UCPGenericSet_scalarInput
    {
    var $ucpid
    var $ucptype
    var $criteriap
    var $description
    };
```

Parameters

ucpid
[in] UCP ID

ucptype
[in] Valid pointids: GENERICSINGLE - single-actor ucp GENERICDOUBLE - double-actor ucp GENERICTRIPLE - triple-actor ucp GENERICFOUR - four-actor ucp

criteriap
[in] Valid criteriaps: U - Union I - Intersection

description
[in] Description of category (resgroup description)

7.2.230 UCPGenericUCDelete

Deletes a user class from a generic UCP

```
struct UCPGenericUCDelete_scalarInput
    {
    var $ucpid
    var $ucptype
    var $ucpmembershipid
    };
```

Parameters

ucpid
[in] UCP ID

ucptype
[in] Valid pointids: GENERICSINGLE - single-actor ucp GENERICDOUBLE - double-actor ucp GENERICTRIPLE - triple-actor ucp GENERICFOUR - four-actor ucp

ucpmembershipid
[in] Unique identifier for this userclass mapping.

7.2.231 UCPGenericUCGet

Lists the user classes belonging to a UCP as well as the mappings

```
struct UCPGenericUCGet_scalarInput
    {
    var $ucpid
    var $ucptype
    };

struct UCPGenericUCGet_vectorOutput
    {
    var $ucid
    var $ucpmembershipid
    var $actor
    var $mapto
    };
```

Parameters

ucpid
[in] UCP ID

ucptype
[in] Valid pointids: GENERICSINGLE - single-actor ucp GENERICDOUBLE - double-actor ucp GENERICTRIPLE - triple-actor ucp GENERICFOUR - four-actor ucp

ucid
[out] User class ID

ucpmembershipid
[out] User class GUID

actor
[out] Name of user class point participant

mapto
[out] Name of user class actor mapped to UCP participant

7.2.232 UserAccountAdd

Manually associates an existing target account with a Bravura Security Fabric profile ID. The API caller needs the "Attach existing IDs" ACL to be able to run this function properly.

```
struct UserAccountAdd_scalarInput
    {
    var $userid
    var $targetid
    var $longid
    var $flag
    };
```

Parameters

userid
[in] The profile ID

targetid
[in] The target of the account

longid
[in] The longid of the account

flag
[in] Determines which flag is set when an account is associated with a profile ID. Accepted values are 0 = NONE, 1 = USER, and 2 = HELPDESK.

7.2.233 UserAccountDelete

Removes the specified account association from the given Bravura Security Fabric profile ID. The API caller needs the "Attach existing IDs" ACL to be able to run this function properly.

```
struct UserAccountDelete_scalarInput
    {
    var $userid
    var $targetid
    var $longid
    };
```

Parameters

userid
[in] The profile ID.

targetid
[in] The target of the account.

longid
[in] The longid of the account.

7.2.234 UserAccountsGet

Retrieves the list of account(s) on target systems for the given profile ID. The API caller needs the "manage profile status" ACL to run this function properly.

This function accepts and/or returns platform type ID values. See Platform Type ID Values for the list of possible values.

```
struct UserAccountsGet_scalarInput
    {
    var $userid
    var $resetOnly
    var $verifyOnly
    var $unlockOnly
    var $challengeresponseOnly
    var $tokenOnly
    var $platformid
    };

struct UserAccountsGet_vectorOutput
    {
    var $longid
    var $shortid
    var $targetid
    var $tgroupid
    var $tsynch
    var $unqpass
    };
```

Parameters

userid
[in] The profile ID of the user.

resetOnly
[in] If this is specified and set to 1, return only accounts on targets that allow the password reset operation.

verifyOnly
[in] If this is specified and set to 1, return only accounts on targets that allow the password verify operation.

unlockOnly
[in] If this is specified and set to 1, return only accounts on targets that allow the account unlock operation.

challengeresponseOnly
[in] If this is specified and set to 1, return only accounts on targets which are challenge/response target.

tokenOnly
[in] If this is specified and set to 1, return only accounts on targets which are token target.

platformid
[in] If this is nonempty, return only accounts on this platform type.

longid
[out] The long ID of the account.

shortid
[out] The short ID of the account.

targetid
[out] The ID of the target where the account resides.

tgroupid
[out] The ID of the target group where the account resides.

tsynch
[out] If set to 1, a password reset on this account should cause accounts in the same target group to be synchronized. If set to 0, do not synchronize.

unqpass
[out] If set to 1, passwords in this target group require a unique password. If set to 0, there is no uniqueness requirement.

7.2.235 UserAccountsUnlock

Unlocks accounts associated with a profile. If no accounts are specified, all associated accounts are unlocked. The API caller needs the "Unlock accounts" ACL to be able to run this function properly.

```
struct UserAccountsUnlock_scalarInput
    {
    var $userid
    struct UserAccountsUnlock_vectorInput $inputArray[]
    };

struct UserAccountsUnlock_vectorInput
    {
    var $longid
    var $targetid
    };

struct UserAccountsUnlock_vectorOutput
    {
    var $longid
    var $targetid
    var $result
    var $retmsg
    };
```

Parameters

userid
[in] The user profile ID to be unlocked.

inputArray
[in] An array of structs of type UserAccountsUnlock_vectorInput

longid
[in] The account ID to unlock.

targetid
[in] The ID identifying the target to unlock account.

longid
[out] Account ID on which an unlock operation was attempted.

targetid
[out] The corresponding target ID.

result
[out] The result of the unlock operation. 0 is success, values other than 0 is failure.

retmsg
[out] Messages returned from the operation.

7.2.236 UserAdminAclTest

Determine if the user has the given ACL (Access to product features).

```
struct UserAdminAclTest_scalarInput
    {
    var $userid
    var $acl
    };

struct UserAdminAclTest_vectorOutput
    {
    var $hasacl
    };
```

Parameters

userid
[in] A profile ID of the user being checked.

acl
[in] The admin ACL string for checking, eg. apicaller, maintain, replication, etc.

hasacl
[out] Whether the user has given Acl, 0 means no, anything else means yes

7.2.237 UserAnswerDelete

Delete question set answers on a user profile.

```
struct UserAnswerDelete_scalarInput
    {
    var $userid
    var $qsid
    var $qid
    };
```

Parameters

userid
[in] The user profile ID identifying the user to validate answers.

qsid
[in] The question set ID the question belongs to. Match all user questions if not specified.

qid
[in] The question ID as returned by UserQuestionsGet. Match all questions within given question set if not specified.

7.2.238 UserAnswerSet

Set an answer to a question on a user profile.

```
struct UserAnswerSet_scalarInput
    {
    var $userid
    var $qsid
    var $question
    var $answer
    };
```

Parameters

userid
[in] The user profile ID identifying the user to validate answers.

qsid
[in] The question set ID the question belongs to.

question
[in] The text of the question.

answer
[in] The answer to set for the question.

7.2.239 UserAnswersValidate

Validate a set of answers against the stored answers in a user profile. If this function succeeds, the supplied answers match the stored answers. The API caller needs "View security questions" ACL to be able to run this function properly.
UserAnswersValidate will return one of the following:

- API_SUCCESS: The supplied answers match the stored answers.
- ERR_VERIFY_FAIL: At least one answer didn't match the stored answer.
- ERR_LOCKED_OUT: Exceeded the system defined MAX_USERAUTH_FAILURE, user profile has been locked out.

```
struct UserAnswersValidate_scalarInput
    {
    var $userid
    struct UserAnswersValidate_vectorInput $inputArray[]
    };

struct UserAnswersValidate_vectorInput
    {
    var $answer
    var $qid
    };
```

Parameters

userid
[in] The user profile ID identifying the user to validate answers.

inputArray
[in] An array of structs of type UserAnswersValidate_vectorInput

answer
[in] The answer to test.

qid
[in] The question identifier, as returned from UserQuestionsGet or QSetQuestionsGet.

7.2.240 UserAttrsGet

Retrieves the attributes of the given profile ID.

```
struct UserAttrsGet_scalarInput
    {
    var $userid
    };
```



```
struct UserAttrsGet_vectorOutput
    {
    var $name
    var $value
    };
```

Parameters

userid
[in] The profile ID of the user.

name
[out] The name of the attribute.

value
[out] The value of the attribute.

7.2.241 UserDisable

Disable the given user profile or, if specified, associated accounts on target systems.

If allAccounts is false or 0 and there are no account/target pairs specified, disable the profile. The API caller needs the "Manage profile status" ACL to be able to run this function in this fashion. In this case, no output is provided.

If allAccounts is true or 1, any account/target pairs are ignored and all of the user's accounts are disabled. Otherwise, all account/target pairs are disabled. The API caller needs the "Disable account" ACL to be able to run this function in this fashion. In this case, output defines the success of each operation.

```
struct UserDisable_scalarInput
    {
    var $userid
    var $allAccounts
    var $disabledreason
    struct UserDisable_vectorInput $inputArray[]
    };

struct UserDisable_vectorInput
    {
    var $longid
    var $targetid
    };

struct UserDisable_vectorOutput
    {
    var $longid
    var $targetid
    var $result
    var $retmsg
    };
```


Parameters

userid
[in] The profile ID of the user.

allAccounts
[in] Disable all accounts associated with the user. If allAccounts is 1 or true, then any account/target pairs passed in will be ignored. If allAccounts is 0 and no account/target pairs are provided, then only profile is disabled.

disabledreason
[in] (Optional) Reason for disabling user profile.

inputArray
[in] An array of structs of type UserDisable_vectorInput

longid
[in] The ID of the account to disable.

targetid
[in] The ID of the target where the account resides.

longid
[out] Account ID on which a disable operation was attempted.

targetid
[out] The ID of the target where the account resides.

result
[out] The disable result on the account/target pair. 0 is success, 1 is failure

retmsg
[out] Messages returned from the operation.

7.2.242 UserEnable

Enable the given user profile or, if specified, associated accounts on target systems.

If allAccounts is false or 0 and there are no account/target pairs specified, enable the profile. The API caller needs the "Manage profile status" ACL to be able to run this function in this fashion. In this case, no output is provided.

If allAccounts is true or 1, any account/target pairs are ignored and all of the user's accounts are enabled. Otherwise, all account/target pairs are enabled. The API caller needs the "Enable account" ACL to be able to run this function in this fashion. In this case, output defines the success of each operation.

```
struct UserEnable_scalarInput
    {
    var $userid
    var $allAccounts
    struct UserEnable_vectorInput $inputArray[]
```



```
    };

struct UserEnable_vectorInput
    {
    var $longid
    var $targetid
    };

struct UserEnable_vectorOutput
    {
    var $longid
    var $targetid
    var $result
    var $retmsg
    };
```

Parameters

userid
[in] The profile ID of the user.

allAccounts
[in] Enable all accounts associated with the user. If allAccounts is 1 or true, then any account/target pairs passed in will be ignored. If allAccounts is 0 and no account/target pairs are provided, then only the user profile is enabled.

inputArray
[in] An array of structs of type UserEnable_vectorInput

longid
[in] The ID of the account to enable.

targetid
[in] The ID of the target where the account resides.

longid
[out] Account ID on which an enable operation was attempted.

targetid
[out] The ID of the target where the account resides.

result
[out] The enable result on the account/target pair. 0 is success, 1 is failure

retmsg
[out] Messages returned from the operation.

7.2.243 UserGetByAccount

Retrieves user ID(s), by the given account on target(s).


```
struct UserGetByAccount_scalarInput
    {
    var $longid
    var $targetid
    };

struct UserGetByAccount_vectorOutput
    {
    var $userid
    var $alias
    var $isadmin
    };
```

Parameters

longid
[in] The account ID on the target system.

targetid
[in] The ID of the target. If empty, all users with an matching longid on any target will be listed.

userid
[out] The profile ID of the user.

alias
[out] The alias of the user.

isadmin
[out] Whether the user is a console user. 1 is true, 0 is false.

7.2.244 UserGetByAcctAttr

Retrieves user profile IDs by the given target attribute value.

```
struct UserGetByAcctAttr_scalarInput
    {
    var $targetid
    var $attrkey
    var $value
    };

struct UserGetByAcctAttr_vectorOutput
    {
    var $userid
    var $alias
    var $isadmin
    };
```

Parameters

targetid
[in] The ID of the target on which the account resides.

attrkey
[in] The name of the target attribute key.

value
[in] The value of the account attribute.

userid
[out] The profile ID of the user.

alias
[out] The alias of the user.

isadmin
[out] Whether the user is a console user. 1 is true, 0 is false.

7.2.245 UserGetByGroup

Retrieves user ID(s), by the given group. If the specified group does not have any member, no users will be returned.

```
struct UserGetByGroup_scalarInput
    {
    var $groupid
    var $targetid
    var $directonly
    };

struct UserGetByGroup_vectorOutput
    {
    var $userid
    var $alias
    var $isadmin
    };
```

Parameters

groupid
[in] The group ID

targetid
[in] The ID of the target.

directonly
[in] Whether to include users with accounts belonging only to the given group, or belonging also to its child groups. 0(zero)=include users with accounts in both the given group and its child groups, otherwise include users with accounts only in given group.

userid
[out] The profile ID of the user.

alias
[out] The alias of the user.

isadmin
[out] Whether the user is a console user.

7.2.246 UserGetByID

Retrieves the basic profile of the given user ID.

```
struct UserGetByID_scalarInput
    {
    var $userid
    };

struct UserGetByID_vectorOutput
    {
    var $userid
    var $alias
    var $guid
    var $isadmin
    };
```

Parameters

userid
[in] The profile ID of the user.

userid
[out] The profile ID of the user.

alias
[out] The alias of the user.

guid
[out] The internal guid of the user.

isadmin
[out] Whether the user is a console user. 1 is true, 0 is false.

7.2.247 UserGetByReqAttr

Retrieves user profile IDs by the given profile attribute value.

```
struct UserGetByReqAttr_scalarInput
    {
```



```
    var $attrkey
    var $value1
    var $value2
    };

struct UserGetByReqAttr_vectorOutput
    {
    var $userid
    var $alias
    var $isadmin
    };
```

Parameters

attrkey
[in] The name of the request attribute.

value1
[in] The value of the request attribute. If value2 is different, then value1 is the minimum of a range of values for the request attribute.

value2
[in] The maximum of a range of values for the request attribute. This should be the same as value1 if the range to search for is a single value.

userid
[out] The profile ID of the user.

alias
[out] The alias of the user.

isadmin
[out] Whether the user is a console user. 1 is true, 0 is false.

7.2.248 UserGroupsGet

Retrieves the managed groups to which the given user belongs on target systems.

```
struct UserGroupsGet_scalarInput
    {
    var $userid
    var $directonly
    };

struct UserGroupsGet_vectorOutput
    {
    var $longid
    var $groupid
    var $targetid
    var $direct
```



```
    };
```

Parameters

userid
[in] The profile ID of the user.

directonly
[in] False will include parent groups in the result

longid
[out] The account ID.

groupid
[out] The ID of the managed target group.

targetid
[out] The ID of the target where the group exists.

direct
[out] 1 if user is directly associated with usergroup. 0 otherwise.

7.2.249 UserIVRList

Retrieves a list of profile IDs matching the specified numeric ID. The API caller needs the "register voice print" ACL to run this function properly.

```
struct UserIVRList_scalarInput
    {
    var $numid
    };

struct UserIVRList_vectorOutput
    {
    var $userid
    var $name
    var $status
    };
```

Parameters

numid
[in] The numeric ID to search for.

userid
[out] A profile ID that matched the given numeric ID.

name
[out] The profile's full name.

status

[out] A flag indicating the profile's status. D is disabled by an administrator, L is locked out due to invalid authentication, A is active.

7.2.250 UserList

List all valid profiles.

```
struct UserList_scalarInput
    {
    var $excludeAdmins
    };

struct UserList_vectorOutput
    {
    var $userid
    var $alias
    var $isadmin
    var $guid
    };
```

Parameters

excludeAdmins
[in] If nonzero, exclude console-only users from the list.

userid
[out] The profile ID of the user.

alias
[out] The alias of the user.

isadmin
[out] Whether the user is a console user.

guid
[out] The internal guid of the user.

7.2.251 UserLockoutSet

Sets user profile attributes related to lockouts. The API caller needs the "Unlock user profile" ACL to be able to run this function properly.

```
struct UserLockoutSet_scalarInput
    {
    var $userid
    var $locked
    var $lastok
    var $lastfail
```



```
    var $failcount
    };
```

Parameters

userid
[in] The profile ID of the user.

locked
[in] The locked status of the user profile. 1 = locked, 0 = unlocked. Specify -1 to ignore this value.

lastok
[in] The ISO-formatted UTC time of the user's last successful login. Specify an empty string to ignore this value.

lastfail
[in] The ISO-formatted UTC time of the user's last unsuccessful login. Specify an empty string to ignore this value.

failcount
[in] The number of consecutive times the user has failed to log in. Specify -1 to ignore this value.

7.2.252 UserPasswordSync

Resets passwords on one or more accounts associated with a specified profile. If no accounts are specified, all associated accounts on resettable targets are reset. If a target ID is specified but an account ID is not specified, then all associated accounts on that target are reset.

Depending on target group configuration, accounts that are associated to the same profile and in the same target group as the specified account may also be synchronized. The API caller needs the "change passwords" ACL to run this function properly.

```
struct UserPasswordSync_scalarInput
    {
    var $userid
    var $password
    var $expire
    var $queue
    var $ignoretsynch
    struct UserPasswordSync_vectorInput $inputArray[]
    };

struct UserPasswordSync_vectorInput
    {
    var $targetid
    var $accountid
    };

struct UserPasswordSync_vectorOutput
    {
```



```
var $numAffected
var $accountid
var $targetid
var $result
var $retmsg
};
```

Parameters

userid
[in] The profile ID associated with the specified account(s).

password
[in] The new password.

expire
[in] If set to 1, also expire the accounts when setting a new password. If set to 2, only expire the accounts and do not set a new password. Otherwise, only set a new password.

queue
[in] If set to 1, send the password changes to IDPM and return immediately. If set to 0, perform the password changes and wait for them to complete before returning.

ignoretsynch
[in] If set to 1, ignore target group settings and only work on input accounts. If set to 0, consider target group tsynch option.

inputArray
[in] An array of structs of type UserPasswordSync_vectorInput

targetid
[in] The target ID of the account to reset.

accountid
[in] The account ID to reset.

numAffected
[out] The number of generated passwords.

accountid
[out] Account ID on which a password reset was attempted.

targetid
[out] The target system ID where the account resides.

result
[out] The password set result on the account/target pair. 0 is success, 1 is failure

retmsg
[out] The message returned from the operation.

7.2.253 UserPasswordVerify

Verify a password on a target account. The API caller needs the "change passwords" ACL to be able to run this function properly.

```
struct UserPasswordVerify_scalarInput
    {
    var $targetid
    var $longid
    var $password
    var $userid
    };

struct UserPasswordVerify_vectorOutput
    {
    var $verifyresult
    var $retmsg
    };
```

Parameters

targetid
[in] The ID identifying the target to verify the user's password.

longid
[in] The account long ID to be verified.

password
[in] The password to be verified.

userid
[in] The profile ID associated with the account (optional).

verifyresult
[out] The verification result. 0 is success, 1 is failure.

retmsg
[out] The message returned by **idpm**.

7.2.254 UserPinValidate

Retrieves the user profile ID according to the pin input if it's valid, otherwise returns nothing. The API caller needs the "register voice print" ACL to run this function properly.

```
struct UserPinValidate_scalarInput
    {
    var $pin
    };
```



```
struct UserPinValidate_vectorOutput
    {
    var $userid
    };
```

Parameters

pin
[in] The pin associated with a profile ID.

userid
[out] The userid who has the input pin.

7.2.255 UserPinValidateEx

Retrieves the user profile ID according to the pin input if it's valid, otherwise returns nothing. The API caller needs the "register voice print" ACL to run this function properly.

```
struct UserPinValidateEx_scalarInput
    {
    var $pin
    var $userid
    };

struct UserPinValidateEx_vectorOutput
    {
    var $userid
    };
```

Parameters

pin
[in] The pin for which to get user profile ID.

userid
[in] The user profile ID.

userid
[out] The userid who has the input pin.

7.2.256 UserQuestionsGet

Get questions from a specified user's profile. The questions returned are randomly chosen from the available set of questions for this profile. The API caller needs "View security questions" ACL to be able to run this function properly.

```
struct UserQuestionsGet_scalarInput
    {
    var $userid
```



```
    var $numpick
    var $ivr
    };

struct UserQuestionsGet_vectorOutput
    {
    var $qid
    var $question
    var $qfileid
    };
```

Parameters

userid
[in] The profile to retrieve questions from.

numpick
[in] The number of questions to pick for this user. When specified as 0 and ivr is 1, pick the minimum number of questions from each ivr question set and return all of them together.

ivr
[in] Specify whether the function is to get ivr questions only.

qid
[out] The ID used to identify the question.

question
[out] The question text.

qfileid
[out] The question wav file ID to be used by Phone Password Manager.

7.2.257 UserRoleAdd

Add user to the role membership.

```
struct UserRoleAdd_scalarInput
    {
    var $userid
    var $roleid
    var $optional
    };
```

Parameters

userid
[in] The user's identifier.

roleid
[in] The role's identifier.

optional
[in] Whether adding the user to role's optional membership. 1 means yes, otherwise no.

7.2.258 UserRoleDelete

Delete user from the role membership.

```
struct UserRoleDelete_scalarInput
    {
    var $userid
    var $roleid
    };
```

Parameters

userid
[in] The user's identifier.

roleid
[in] The role's identifier.

7.2.259 UserSearch

Advanced search for users based on up to 5 search criteria. If there are no matches, returns API_NO_MORE_DATA.

Be sure to grant the API user permission to read the relevant attributes in the profiles of the users being searched. For example, if you wish to search on an attribute such as SSN, and that attribute is protected such that only the user themselves and/or users in HR can read it, then UserSearch() will fail to search on this profile attribute. You fix that by creating a security group that allows your API user to read this attribute.

```
struct UserSearch_scalarInput
    {
    var $matchAny
    var $inclinvalid
    struct UserSearch_vectorInput $inputArray[]
    };

struct UserSearch_vectorInput
    {
    var $field
    var $attribute
    var $comparator
    var $value
    };

struct UserSearch_vectorOutput
    {
    var $userid
```



```
var $alias
var $isadmin
var $isinvalid
};
```

Parameters

matchAny
[in] Flag indicating whether search should match on any of the criteria (1) or all of the criteria (0).

inclinvalid
[in] Flag indicating whether search results should include invalid users.

inputArray
[in] An array of structs of type UserSearch_vectorInput

field
[in] Field to be searched against. Acceptable values are userid, name, hostsearch, groupsearch, rolesearch, directmanager, indirectmanager. This field is susceptible to change depending on the search engine keys defined. Profile attributes should be searched using the attribute vector input. If searching a field, attribute input should be empty. If searching a profile attribute, field input should be empty. Valid search fields: USERID, USERNAME, GROUPID, ROLEID, TARGETID, DIRECT_MANAGER, ALL_MANAGERS

attribute
[in] Profile attribute to be searched against. Any profile attribute defined can be searched for, but ACLs will be checked against and filtered for. Defined fields should be searched using the field vector input. If searching a field, attribute input should be empty. If searching a profile attribute, field input should be empty.

comparator
[in] String indicating the type of comparison to perform. Comparators use the following values: EQ - "is equal to" (strings, integers, dates, boolean) NEQ - "is not equal to" (strings, integers, dates, boolean) LT - "is less than" (strings, integers, dates) LTE - "is less than or equal to" (strings, integers, dates) GT - "is greater than" (strings, integers, dates) GTE - "is greater than or equal to" (strings, integers, dates) CT - "contains" (strings) NCT - "does not contain" (strings) IS - "exists or not exists" (strings, integers, dates, boolean)

value
[in] The value being filtered against. Note that the value needs to correspond to the field or attribute.datatype being searched for. Possible types: strings, integers, dates, or boolean values. Date type require date format YYYY-MM-DD for date only, or YYYY-MM-DD HH:mm for date and time. Note that the API does not do any UTC-timezone conversion. So if you input a DateTime value, it has to be coherent with the matching value as seen in the database. Not the value presented by the Suite GUI (which most likely is converted to current timezone). Boolean type requires "EQ" or "NEQ" comparators and "TRUE" or "FALSE" values. Any type using "IS" comparator must use "SET" or "EMPTY" values.

userid
[out] The profile ID of the user.

alias
[out] The alias of the user.

isadmin

[out] Whether the user is a console user.

isinvalid
[out] Whether the user is an invalid user.

7.2.260 UserSetValid

Make the given user profile valid or invalid. If setting it to valid, the profile is created if necessary. Future runs of auto-discovery may or may not retain this setting.

```
struct UserSetValid_scalarInput
    {
    var $userid
    var $valid
    };
```

Parameters

userid
[in] The profile ID of the user.

valid
[in] When nonzero, make the given user profile valid. Otherwise, make the user profile invalid.

7.2.261 UserSettingsGet

Get user settings

```
struct UserSettingsGet_scalarInput
    {
    var $profilename
    };

struct UserSettingsGet_vectorOutput
    {
    var $value
    };
```

Parameters

profilename
[in] Profile name

value
[out] Combined user setting in JSON format

7.2.262 UserSettingsSet

Set user settings based on a template

```
struct UserSettingsSet_scalarInput
    {
    var $profilename
    var $settings
    };
```

Parameters

profilename
[in] Profile name ID

settings
[in] Template in json format to be saved as user settings for a given profilename

7.2.263 UserSettingsTemplateDelete

Delete user settings template

```
struct UserSettingsTemplateDelete_scalarInput
    {
    var $id
    };
```

Parameters

id
[in] Template ID

7.2.264 UserSettingsTemplateGet

Get user settings template

```
struct UserSettingsTemplateGet_scalarInput
    {
    var $id
    };

struct UserSettingsTemplateGet_vectorOutput
    {
    var $value
    };
```

Parameters

id
[in] Template ID

value
[out] Template in JSON format

7.2.265 UserSettingsTemplateList

Get list of user settings template

```
struct UserSettingsTemplateList_scalarInput
    {
    var $unused


    };

struct UserSettingsTemplateList_vectorOutput
    {
    var $id
    var $value
    };
```

Parameters

unused
[in] Not used

id
[out] Template ID

value
[out] Template in JSON format

7.2.266 UserSettingsTemplateSet

Set user settings template

```
struct UserSettingsTemplateSet_scalarInput
    {
    var $id
    var $settings
    };
```

Parameters

id
[in] Template ID

settings
[in] Settings to be saved as template

7.2.267 UserStatDelete

Deletes a userstat record.

```
struct UserStatDelete_scalarInput
    {
    var $userid
    var $ustattag
    };
```

Parameters

userid
[in] The profile ID of the user.

ustattag
[in] The userstat tag.

7.2.268 UserStatGet

Retrieve userstat records for a specified user.

```
struct UserStatGet_scalarInput
    {
    var $userid
    var $ustattag
    };

struct UserStatGet_vectorOutput
    {
    var $tag
    var $cvalue
    };
```

Parameters

userid
[in] The profile ID of the user.

ustattag
[in] The userstat tag to search for. If empty, return all userstat records.

tag
[out] The userstat tag.

cvalue

[out] The userstat string value.

7.2.269 UserStatSet

Sets a userstat record.

```
struct UserStatSet_scalarInput
    {
    var $userid
    var $ustattag
    var $ustatval
    };
```

Parameters

userid
[in] The profile ID of the user.

ustattag
[in] The userstat tag.

ustatval
[in] The userstat string value.

7.2.270 UserStatusGet

Gets user login status attributes. If the user has never attempted to log in, fails with API_NO_MORE_DATA.

```
struct UserStatusGet_scalarInput
    {
    var $userid
    };

struct UserStatusGet_vectorOutput
    {
    var $locked
    var $disabled
    var $lastok
    var $lastfail
    var $failcount
    var $unlockby
    var $statuschangedby
    var $unlocktime
    var $statuschangedtime
    };
```

Parameters

userid

[in] The profile ID of the user.

locked
[out] The locked status of the user profile. 1 = locked, 0 = unlocked.

disabled
[out] The disabled status of the user profile. 1 = disabled, 0 = enabled.

lastok
[out] The ISO-formatted UTC time of the user's last successful login. If the user has never successfully logged in, returns an empty string.

lastfail
[out] The ISO-formatted UTC time of the user's last unsuccessful login. If the user has never failed to login, returns an empty string.

failcount
[out] The number of consecutive times the user has failed to log in.

unlockby
[out] The ID of the last user to unlock the user profile.

statuschangedby
[out] The ID of the last user to change the user profile's status.

unlocktime
[out] The last time the user profile was unlocked, UTC, in ISO format. If the profile has never been unlocked, returns an empty string.

statuschangedtime
[out] The last time the user profile's status was changed, in UTC, ISO format. If the profile's status has never been changed, returns an empty string.

7.2.271 UserVoiceRegStatGet

Retrieves the user voice registration status The API caller needs the "register voice print" ACL to run this function properly.

```
struct UserVoiceRegStatGet_scalarInput
    {
    var $userid
    };

struct UserVoiceRegStatGet_vectorOutput
    {
    var $regflag
    };
```

Parameters

userid

[in] The user ID for which to get voice registration status

regflag
[out] Whether the user is registered. Any value greater than 0 is registered, 0 is non-registered.

7.2.272 UserVoiceRegStatSet

Sets the voice registration status. The API caller needs the "register voice print" ACL to run this function properly.

```
struct UserVoiceRegStatSet_scalarInput
    {
    var $userid
    var $regflag
    };
```

Parameters

userid
[in] The userid to set voice registration status.

regflag
[in] Whether the user is to be registered. Any value greater than 0 is register, 0 is unregister.

7.2.273 UserclassActorGet

Retrieves the actors used inside a userclass.

```
struct UserclassActorGet_scalarInput
    {
    var $userclass
    };

struct UserclassActorGet_vectorOutput
    {
    var $actor
    var $desc
    var $useexpr
    var $exprdef
    var $expression
    };
```

Parameters

userclass
[in] The userclass ID.

actor
[out] The actor.

desc
[out] The description.

useexpr
[out] Whether to use the list expression. 1 is true, 0 is false

exprdef
[out] Whether to exclude all other criteria in the user class when using list expression. 1 is exclude, 0 is include

expression
[out] List expression.

7.2.274 UserclassCacheRecalculate

Recalculate the membership cache of a user class.

```
struct UserclassCacheRecalculate_scalarInput
    {
    var $userclass
    };
```

Parameters

userclass
[in] The userclass ID.

7.2.275 UserclassList

Lists userclasses matching the criteria specified

```
struct UserclassList_scalarInput
    {
    var $singleActor
    var $excludeBuiltin
    var $excludeDeprecated
    };

struct UserclassList_vectorOutput
    {
    var $id
    var $desc
    };
```

Parameters

singleActor
[in] A boolean that indicates if only single-actor userclasses should be returned.

excludeBuiltin
[in] A boolean that indicates if builtin userclasses should be excluded from the result.

excludeDeprecated
[in] A boolean that indicates if deprecated userclasses should be excluded from the result.

id
[out] The userclass ID.

desc
[out] The userclass description.

7.2.276 UserclassMemberListExplicit

Lists users explicitly assigned to the user class.

```
struct UserclassMemberListExplicit_scalarInput
    {
    var $userclass
    };

struct UserclassMemberListExplicit_vectorOutput
    {
    var $userid
    var $memberguid
    };
```

Parameters

userclass
[in] The userclass ID.

userid
[out] The user profile IDs that assigned to the user class.

memberguid
[out] The internal guid of the member.

7.2.277 UserclassMembershipCheck

Check whether the scenario (actor:userid) specified matches the membership criteria of the user class.

```
struct UserclassMembershipCheck_scalarInput
    {
    var $userclass
    struct UserclassMembershipCheck_vectorInput $inputArray[]
    };
```



```
struct UserclassMembershipCheck_vectorInput
    {
    var $actor
    var $userid
    };

struct UserclassMembershipCheck_vectorOutput
    {
    var $ismember
    };
```

Parameters

userclass
[in] The userclass ID.

inputArray
[in] An array of structs of type UserclassMembershipCheck_vectorInput

actor
[in] The actor.

userid
[in] The profile ID taking the role of this actor.

ismember
[out] Whether the scenario specified matches the membership criteria of the user class. 1 is true, 0 is false.

7.2.278 UserclassMembershipList

Lists the profile IDs for a given actor matching the scenario specified.

```
struct UserclassMembershipList_scalarInput
    {
    var $userclass
    var $actor
    struct UserclassMembershipList_vectorInput $inputArray[]
    };

struct UserclassMembershipList_vectorInput
    {
    var $actor
    var $userid
    };

struct UserclassMembershipList_vectorOutput
    {
    var $userid
    };
```


Parameters

userclass
[in] The userclass ID.

actor
[in] The actor to list.

inputArray
[in] An array of structs of type UserclassMembershipList_vectorInput

actor
[in] The other actor IDs.

userid
[in] The profile ID taking the role of this actor.

userid
[out] The profile IDs that match the scenario specified.

7.2.279 UserclassPointCacheRecalculate

Recalculate the membership cache of a user class point.

```
struct UserclassPointCacheRecalculate_scalarInput
    {
    var $pointid
    var $id
    var $phase
    };
```

Parameters

pointid
[in] The user class point.

id
[in] The instance of the user class point.

phase
[in] The phase of the user class; if phases are not enabled, specify 1.

7.2.280 UserclassPointList

Returns a list of available user class points. Depending on the licenses installed, some user class points may not be listed.

```
struct UserclassPointList_scalarInput
    {
```



```
    var $unused


    };

struct UserclassPointList_vectorOutput
    {
    var $name
    var $description
    };
```

Parameters

unused
[in] Not used

name
[out] name of the user class point

description
[out] description of the user class point

7.2.281 UserclassUserAdd

Adds a user explicitly to a user class.

```
struct UserclassUserAdd_scalarInput
    {
    var $userclass
    var $userid
    };
```

Parameters

userclass
[in] The userclass ID.

userid
[in] User profile ID.

7.2.282 UserclassUserDelete

Removes an explicitly added user from a user class.

```
struct UserclassUserDelete_scalarInput
    {
    var $userclass
    var $userid
    };
```


Parameters

userclass
[in] The userclass ID.

userid
[in] User profile ID.

7.2.283 VerifySystemCredential

Attempts to authenticate on a particular system with the given credentials. Warning: Using invalid credentials may result in account lockout!

This function accepts and/or returns platform type ID values. See Platform Type ID Values for the list of possible values.

```
struct VerifySystemCredential_scalarInput
    {
    var $platformtype
    var $address
    var $userid
    var $password
    var $proxy
    var $runas
    };
```

Parameters

platformtype
[in] The type of agent to use to connect to this system.

address
[in] The address of the system to connect to.

userid
[in] The user ID to use when connecting.

password
[in] The password to use when connecting.

proxy
[in] The proxy of the system to connect to.

runas
[in] Use runas?

7.2.284 WFAbstain

Abstains from the specified batch for the logged-in user.


```
struct WFAbstain_scalarInput
    {
    var $batchsig
    var $reason
    var $primary
    };
```

Parameters

batchsig
[in] The ID identifying the batch.

reason
[in] The reason for the abstention.

primary
[in] The profile ID of the user on whose behalf this is being done. If blank, then delegation is not used.

7.2.285 WFAccept

Accepts an implementation task as the logged-in user.

```
struct WFAccept_scalarInput
    {
    var $batchsig
    var $reqacctsig
    var $primary
    };
```

Parameters

batchsig
[in] The ID identifying the batch.

reqacctsig
[in] The action ID for this task.

primary
[in] The profile ID of the user on whose behalf this is being done. If blank, then delegation is not used.

7.2.286 WFApprove

Approves the specified batch for the logged-in user.

```
struct WFApprove_scalarInput
    {
    var $batchsig
    var $reason
```



```
    var $primary
    };
```

Parameters

batchsig
[in] The ID identifying the batch.

reason
[in] The reason for the approval.

primary
[in] The profile ID of the user on whose behalf this is being done. If blank, then delegation is not used.

7.2.287 WFDecline

Declines an implementation task as the logged-in user.

```
struct WFDecline_scalarInput
    {
    var $batchsig
    var $reqacctsig
    var $reason
    var $primary
    };
```

Parameters

batchsig
[in] The ID identifying the batch.

reqacctsig
[in] The action ID for this task.

reason
[in] The reason for the declination.

primary
[in] The profile ID of the user on whose behalf this is being done. If blank, then delegation is not used.

7.2.288 WFDelegate

Add a delegate.

```
struct WFDelegate_scalarInput
    {
    var $primary
    var $deleg
    var $type
```



```
var $batchid
var $itemid
var $segid
var $reason
var $delegable
var $acceptneed
var $startdate
var $enddate
};
```

Parameters

primary
[in] The profile ID of the user whose responsibility will be delegated.

deleg
[in] A profile ID for the delegate.

type
[in] The type of the delegation. It could be one of BATCH, CERT, TASK or ALL.

batchid
[in] A request batch ID for delegation.

itemid
[in] A request item ID for delegation.

segid
[in] A segmentation ID for delegation.

reason
[in] The reason for the delegation.

delegable
[in] 1 for delegable, otherwise not.

acceptneed
[in] 1 for acceptance needed, otherwise not.

startdate
[in] Take effect as of this date, in UTC, ISO format.

enddate
[in] Stop delegating as of this date, in UTC, ISO format. This is only valid for ALL type delegation.

7.2.289 WFPDRSubmit

An all-in-one api function call that can finish all the steps all together from login to submit. It creates a new workflow request and then submit the request for processing.


```
struct WFPDRSubmit_scalarInput
    {
    var $recipid
    var $recippass
    var $recipemail
    var $requesteremail
    var $requestreason
    var $preqid
    var $reserveid
    var $maqid
    var $parentbatchsig
    var $serverid
    struct WFPDRSubmit_vectorInput $inputArray[]
    };

struct WFPDRSubmit_vectorInput
    {
    var $attrkey
    var $value
    };

struct WFPDRSubmit_vectorOutput
    {
    var $batchsig
    var $serverid
    };
```

Parameters

recipid
[in] The profile ID of the person whom this request is for.

recippass
[in] The initial password for the recipient's account on specified target. Used by ACUA.

recipemail
[in] The email for the recipient.

requesteremail
[in] The email of the requester.

requestreason
[in] The reason for this request.

preqid
[in] Required pre-defined request ID to base this new request on.

reserveid
[in] Optional reservation ID returned by ReserveAccountId, ReserveAttrValue, or ReserveProfileId.

maqid
[in] The ID of the account set.

parentbatchsig
[in] Optional ID identifying the parent request.

serverid
[in] Optional server ID to process the request on. Empty serverid means to run the request locally, or the special string "RANDOM" can be supplied to allow selection of a random server.

inputArray
[in] An array of structs of type WFPDRSubmit_vectorInput

attrkey
[in] The key of the value to be used in operations.

value
[in] The value for action/to update. Values for multi-value should be specified across a series of key/value pairs.

batchsig
[out] The ID identifying the batch.

serverid
[out] Server ID that will process the request.

7.2.290 WFReject

Denies the specified batch for the logged-in user.

```
struct WFReject_scalarInput
    {
    var $batchsig
    var $reason
    var $primary
    };
```

Parameters

batchsig
[in] The ID identifying the batch.

reason
[in] The reason for the denial.

primary
[in] The profile ID of the user on whose behalf this is being done. If blank, then delegation is not used.

7.2.291 WFRequestActionAuthorGet

Retrieves all of the authorizers and implementers associated with the given request action. Results will differ over time as the authorizers and implementers can change with dynamic authorization or as they

approve/deny/accept/complete requests.

```
struct WFRequestActionAuthorGet_scalarInput
    {
    var $batchsig
    var $reqacctsig
    };

struct WFRequestActionAuthorGet_vectorOutput
    {
    var $author
    var $phase
    var $authorizeType
    var $notes
    var $status
    var $authnote
    var $tmstart
    };
```

Parameters

batchsig
[in] The request ID.

reqacctsig
[in] The action ID.

author
[out] The profile ID of the authorizer.

phase
[out] Phase in which the authorizer will be asked to respond.

authorizeType
[out] The type of the request. A is approve creation of the account, M is assign authorizers.

notes
[out] An explanation, if any, for the status of the action.

status
[out] The status code of the action.

authnote
[out] The note provided by authmod

tmstart
[out] The ISO-formatted UTC time the authorizer was first assigned to this request action.

7.2.292 WFRequestActionsGenericSet

Sets the generic actions associated with the given request.

```
struct WFRequestActionsGenericSet_scalarInput
    {
    var $batchsig
    var $sessionid
    var $serverid
    struct WFRequestActionsGenericSet_vectorInput $inputArray[]
    };

struct WFRequestActionsGenericSet_vectorInput
    {
    var $operationType
    var $metaobjnew
    var $metaobjtype
    var $metaobjname
    var $childmetaobjnew
    var $childmetaobjtype
    var $childmetaobjname
    var $targetobjnew
    var $targetobjtype
    var $targetobjname
    var $targetobjhostid
    var $childtargetobjnew
    var $childtargetobjtype
    var $childtargetobjname
    var $childtargetobjhostid
    var $reltype
    var $newid
    var $reason
    var $expanded
    };
```

Parameters

batchsig
[in] The ID identifying the batch.

sessionid
[in] Identifier for the session that contains the in-progress request.

serverid
[in] Server ID that is processing the request, or empty if the request is running locally.

inputArray
[in] An array of structs of type WFRequestActionsGenericSet_vectorInput

operationType
[in] The type of operation. This is one of the types returned by WFRequestOperationTypes.

metaobjnew
[in] Whether the meta object that is the only or the one that is considered the parent object for metaobj-related operations will be created or references a meta object that will be created in this request.

metaobjtype
[in] Type of the (potentially new) meta object that is the only or the one that is considered the parent object for metaobj-related operations.

metaobjname
[in] The name (at the time of the request) of the (potentially new) meta object that is the only or the one that is considered the parent object for metaobj-related operations.

childmetaobjnew
[in] Whether the meta object that is considered the child object for metaobj-related relational operations exists or references a meta object that will be created in this request.

childmetaobjtype
[in] Type of the (potentially new) meta object that is considered the child object for metaobj-related relational operations.

childmetaobjname
[in] The name (at the time of the request) of the (potentially new) meta object that is considered the child object for metaobj-related relational operations.

targetobjnew
[in] Whether the target object that is the only or the one that is considered the parent object for targetobj-related operations will be created or references a target object that will be created in this request.

targetobjtype
[in] Type of the (potentially new) target object that is the only or the one that is considered the parent object for targetobj-related operations.

targetobjname
[in] The name (at the time of the request) of the target object that is the only or the one that is considered the parent object for targetobj-related operations.

targetobjhostid
[in] The target identifier of the (potentially new) target object that is the only or the one that is considered the parent object for targetobj-related operations.

childtargetobjnew
[in] Whether the target object that is considered the child object for targetobj-related relational operations exists or references a target object that will be created in this request.

childtargetobjtype
[in] Type of the (potentially new) target object that is considered the child object for targetobj-related relational operations.

childtargetobjname
[in] The name (at the time of the request) of the (potentially new) target object that is considered the child object for targetobj-related relational operations.

childtargetobjhostid
[in] The target identifier of the (potentially new) target object that is considered the child object for targetobj-related relational operations.

reltype
[in] The relation between the meta/target objects for this operation.

newid
[in] The new ID used by the OBJ_RENAME operation.

reason
[in] (optional) The reason for the action. Used by all action types.

expanded
[in] Set to true if this role has already been expanded.

7.2.293 WFRequestActionsGet

Retrieves the actions associated with the given request.

```
struct WFRequestActionsGet_scalarInput
    {
    var $batchsig
    var $sessionid
    };

struct WFRequestActionsGet_vectorOutput
    {
    var $operationType
    var $reqacctsig
    var $acctid
    var $newid
    var $acctidset
    var $recippass
    var $dstuserid
    var $newgroup
    var $groupid
    var $childgroupid
    var $newchildgroup
    var $newchildgroupid
    var $newchildgrouptargetid
    var $grpname
    var $grpguid
    var $targetid
    var $childtargetid
    var $objectname
    var $objecttype
    var $objsecuritydomain
    var $newctx
    var $parentrole
```



```
var $reason
var $resmemberid
var $roleid
var $sodid
var $srcuserid
var $tplid
var $tpluri
var $userid
var $resourceaddress
var $resourcetype
var $result
var $status
var $authnote
var $authtype
var $bbtag
var $metaobjnew
var $metaobjtype
var $metaobjname
var $childmetaobjnew
var $childmetaobjtype
var $childmetaobjname
var $targetobjnew
var $targetobjtype
var $targetobjname
var $targetobjhostid
var $childtargetobjnew
var $childtargetobjtype
var $childtargetobjname
var $childtargetobjhostid
var $reltype
var $checkoutid
var $expanded
};
```

Parameters

batchsig
[in] The ID identifying the batch.

sessionid
[in] Get the request as it exists in the given session. This is useful when a request is actively being created (or modified) but the user has not yet committed their changes and the API must get the information about the request as it is being changed. If this is empty, or if an actively changed request is not found in the given session, the request will be extracted as it exists currently.

operationType
[out] The type of operation. This is one of the types returned by WFRequestOperationTypes.

reqacctsig
[out] The action ID.

acctid
[out] The account ID on the target. Used by the ARCHREQPWD, CFYA, CFYG, DELU, DNAU, ENAU, GRUA, GROA, GROD, GRUD, MVCU, RENU, LUPD, LDEL, and UPDT action types.

newid
[out] The new ID used by the RENU, RPRO action types.

acctidset
[out] Whether the account ID that will be affected has been determined explicitly. Used by the ACUA action type.

recippass
[out] The initial password for the recipient's account on specified target.

dstuserid
[out] The profile ID of a destination user. Used by the CFYR, ORGTFRSUB and ORGTFRSUBPULL action types.

newgroup
[out] If equal to 1, the group membership adding operation is based on a to-be-created group using CRTG. Used by the GRUA action type.

groupid
[out] The ID of a managed group on the target. Used by the CFYG, CRTG, GROA, GROD, GRUA, and GRUD action types.

childgroupid
[out] The ID of a child managed group on the target. Used by the GRGA, GRGD action types.

newchildgroup
[out] If equal to 1, the child group is a new group that will be created in this request.

newchildgroupid
[out] The ID of a child managed group on the target. Used by the GRGA, GOGA action types. Only set if newchildgroup is 1.

newchildgrouptargetid
[out] The ID of a child managed group on the target. Used by the GRGA, GOGA action types. Only set if newchildgroup is 1.

grpname
[out] The name of a managed group on the target. Only used by the CRTG action type.

grpguid
[out] The guid of a managed group on the target. Only used by the DELG action type.

targetid
[out] The ID of a target. Used by the ARCHREQPWD, CFYA, CFYG, DELU, DNAU, ENAU, GROA, GROD, GRUA, GRUD, MVCU, RENU, LUPD, LDEL, UPDT and GRDB action types.

childtargetid
[out] The ID of the target of childgroupid. Used by GRGA and GRGD action types.

objectname
[out] Name of unknown object member of group, used by GRBD operation.

objecttype
[out] Type of unknown object member of group, used by GRBD operation.

objsecuritydomain
[out] Security domain of unknown object member of group, used by GRBD operation.

newctx
[out] The value of the new context. Used by the MVCU action type.

parentrole
[out] The ID of a parent role. Used by the ACUA, RLUA, DELR, GRUA, ADDM, DELM action types.

reason
[out] The reason for the action. Used by all action types.

resmemberid
[out] The ID of a resource member. Used by the AEDEFICIT action type.

roleid
[out] The ID of a role. Used by the CFYAEDEF, CFYR, RLUA, DELR, and GRUA action types.

sodid
[out] The ID of an SOD rule. Used by the AESOD action type.

srcuserid
[out] The profile ID of a source user. Used by the ORGTFRSUB and ORGTFRSUBPULL action type.

tplid
[out] The ID of a template. Used by the ACUA, GRUA, and DELU action types.

tpluri
[out] The URI template. Used by NRCR, NRUP, NRDL, NRMV action types.

userid
[out] The profile ID of a user. Used by the CFYU, DELR, ORGADDMGR, ORGADDSUB, ORGDELMGR, ORGDELSUB, ORGTFRSUB, ORGTFRSUBPULL, LUPD, and RENU action types.

resourceaddress
[out] The resource address. Used by NRCR, NRUP, NRDL, NRMV action types.

resourcetype
[out] The resource type. Used by NRCR, NRUP, NRDL, NRMV action types.

result
[out] The result of operation.

status
[out] The status of operation.

authnote

[out] The note provided by authmod

authtype
[out] The authentication type. Either phased or parallel.

bbtag
[out] The value of the tag to use for replacement in blackboard rules.

metaobjnew
[out] Whether the meta object that is the only or the one that is considered the parent object for metaobj-related operations will be created or references a meta object that will be created in this request.

metaobjtype
[out] Type of the (potentially new) meta object that is the only or the one that is considered the parent object for metaobj-related operations.

metaobjname
[out] The name (at the time of the request) of the (potentially new) meta object that is the only or the one that is considered the parent object for metaobj-related operations.

childmetaobjnew
[out] Whether the meta object that is considered the child object for metaobj-related relational operations exists or references a meta object that will be created in this request.

childmetaobjtype
[out] Type of the (potentially new) meta object that is considered the child object for metaobj-related relational operations.

childmetaobjname
[out] The name (at the time of the request) of the (potentially new) meta object that is considered the child object for metaobj-related relational operations.

targetobjnew
[out] Whether the target object that is the only or the one that is considered the parent object for targetobj-related operations will be created or references a target object that will be created in this request.

targetobjtype
[out] Type of the (potentially new) target object that is the only or the one that is considered the parent object for targetobj-related operations.

targetobjname
[out] The name (at the time of the request) of the target object that is the only or the one that is considered the parent object for targetobj-related operations.

targetobjhostid
[out] The target identifier of the (potentially new) target object that is the only or the one that is considered the parent object for targetobj-related operations.

childtargetobjnew
[out] Whether the target object that is considered the child object for targetobj-related relational operations exists or references a target object that will be created in this request.

childtargetobjtype
[out] Type of the (potentially new) target object that is considered the child object for targetobj-related relational operations.

childtargetobjname
[out] The name (at the time of the request) of the (potentially new) target object that is considered the child object for targetobj-related relational operations.

childtargetobjhostid
[out] The target identifier of the (potentially new) target object that is considered the child object for targetobj-related relational operations.

reltype
[out] The relation between the meta/target objects for this operation.

checkoutid
[out] Check-out id.

expanded
[out] Set to true if this role has already been expanded.

7.2.294 WFRequestActionsSet

Sets the actions associated with the given request.

```
struct WFRequestActionsSet_scalarInput
    {
    var $batchsig
    var $sessionid
    var $serverid
    struct WFRequestActionsSet_vectorInput $inputArray[]
    };

struct WFRequestActionsSet_vectorInput
    {
    var $operationType
    var $acctid
    var $acctidset
    var $recippass
    var $dstuserid
    var $newgroup
    var $groupid
    var $grpname
    var $grpguid
    var $childgroupid
    var $newchildgroup
    var $newchildgroupid
    var $newchildgrouptargetid
    var $targetid
    var $childtargetid
```



```
var $objectname
var $objecttype
var $accthostid
var $gsetid
var $policyid
var $newctx
var $parentrole
var $reason
var $resmemberid
var $roleid
var $sodid
var $expdate
var $srcuserid
var $newid
var $tplid
var $tpluri
var $userid
var $resourceaddress
var $resourcetype
var $checkoutid
var $expanded
};
```

Parameters

batchsig
[in] The ID identifying the batch.

sessionid
[in] Identifier for the session that contains the in-progress request.

serverid
[in] Server ID that is processing the request, or empty if the request is running locally.

inputArray
[in] An array of structs of type WFRequestActionsSet_vectorInput

operationType
[in] The type of operation. This is one of the types returned by WFRequestOperationTypes.

acctid
[in] The account ID on the target. Used by the ARCHREQPWD, ARCHREQGRP, DELU, DNAU, ENAU, GRUA, GROA, GROD, GRUD, LDEL, LUPD, MVCU, RENU, and UPDT action types.

acctidset
[in] Whether the account ID that will be affected has been determined explicitly. Used by the ACUA action type.

recippass
[in] The initial password for the recipient's account on specified target. Used by ACUA.

dstuserid

[in] The ID of a destination user. Used by the ORGTFRSUB and ORGTFRSUBPULL action type.

newgroup
[in] If equals to 1, the group membership adding operation is based on a to-be-created group using CRTG. Used by the GRUA action type.

groupid
[in] The ID of a managed group on the target. Used by the GROA, GROD, GRUA, GRUD, GRGA, GRGD, CRTG, DELG, AEDEF, AEDEFDEL, AESUR, AESURDEL, AESOD and GRBD action types.

grpname
[in] The name of a managed group on the target. Only used by the CRTG action type.

grpguid
[in] The guid of a managed group on the target. Only used by the DELG action type.

childgroupid
[in] The ID of a child managed group on the target. Used by the GRGA, GRGD action types.

newchildgroup
[in] If equal to 1, the child group is a new group that will be created in this request.

newchildgroupid
[in] The ID of a child managed group on the target. Used by the GRGA, GOGA action types. Only set if newchildgroup is 1.

newchildgrouptargetid
[in] The ID of a child managed group on the target. Used by the GRGA, GOGA action types. Only set if newchildgroup is 1.

targetid
[in] The ID of a target. Used by the ARCHREQPWD, ARCHREQGRP, DELU, DNAU, ENAU, GROA, GROD, GRUA, GRUD, GRGA, GRGD, CRTG, DELG, LDEL, LUPD, MVCU, RENU, UPDT, AEDEF, AEDEFDEL, AESUR, AESURDEL, and AESOD action types.

childtargetid
[in] The ID of the target of childgroupid. Used by GRGA and GRGD action types.

objectname
[in] Name of object loaded from group, used by GRBD operation.

objecttype
[in] Type of object loaded from group, used by GRBD operation.

accthostid
[in] Target system ID to perform lookups for the account to apply temporary group permissions. Used by the ARCHREQGRP action.

gsetid
[in] The ID of the group set. Used by the ARCHREQGRP action.

policyid

[in] The ID of the policy for temporary group access request. Used by the ARCHREQGRP action.

newctx
[in] The value of the new context. Used by the MVCU action type.

parentrole
[in] (optional) The ID of a parent role. Used by the ACUA, RLUA, DELR, GRUA, and GRGA action types.

reason
[in] (optional) The reason for the action. Used by all action types.

resmemberid
[in] The ID of a resource member. Used by the AEDEFICIT action type.

roleid
[in] The ID of a role. Used by the RLUA, DELR, GRUA, AEDEF, and AEDEFDEL action types. Can be used together with tplid only for the AEDEF and AEDEFDEL action types.

sodid
[in] The ID of an SOD rule. Used by the AESOD action type.

expdate
[in] The ISO-formatted UTC expiry date, only applicable for AESOD and AESOD_EXTEND operations.

srcuserid
[in] The profile ID of a source user. Used by the ORGTFRSUB and ORGTFRSUBPULL action type.

newid
[in] The new ID used by the RENU, RPRO action types.

tplid
[in] The ID of a template. Used by the ACUA, DELU, GRUA, GRGA, AEDEF, AEDEFDEL, AESUR, AESURDEL, and AESOD action types. Can be used together with roleid only for the AEDEF and AEDEFDEL action types. Optional for the GRUA action type.

tpluri
[in] The URI template. Used by NRCR, NRUP, NRDL, NRMV action types.

userid
[in] The profile ID of a user. Used by the ENAP, DNAP, DELR, LUPD, RPRO, ORGADDMGR, ORGADDSUB, ORGDELMGR, ORGDELSUB, ORGTFRSUB, ORGTFRSUBPULL, and RENU action types.

resourceaddress
[in] The resource address. Used by NRCR, NRUP, NRDL, NRMV action types.

resourcetype
[in] The resource type. Used by NRCR, NRUP, NRDL, NRMV, CRTC, DELC, UPCO, ADDM, DELM action types.

checkoutid
[in] Check-out id.

expanded
[in] Set to true if this role has already been expanded.

7.2.295 WFRequestAttrActionsPopulate

Adds request actions for affected target/account attributes of the given request. Assumes the profile attributes to update are already populated into the request.

```
struct WFRequestAttrActionsPopulate_scalarInput
    {
    var $batchsig
    var $sessionid
    var $serverid
    };
```

Parameters

batchsig
[in] The ID identifying the batch.

sessionid
[in] Identifier for the session that contains the in-progress request.

serverid
[in] Server ID that is processing the request, or empty if the request is running locally.

7.2.296 WFRequestAttrsGet

Retrieves the request attributes of the recipient of the given request.

```
struct WFRequestAttrsGet_scalarInput
    {
    var $batchsig
    var $sessionid
    };

struct WFRequestAttrsGet_vectorOutput
    {
    var $name
    var $value
    var $attributeType
    var $reqacctsig
    };
```

Parameters

batchsig
[in] The ID identifying the batch.

sessionid
[in] Get the request as it exists in the given session. This is useful when a request is actively being created (or modified) but the user has not yet committed their changes and the API must get the information about the request as it is being changed. If this is empty, or if an actively changed request is not found in the given session, the request will be extracted as it exists currently.

name
[out] The name of the request attribute.

value
[out] The value of the attribute. Each value of a multi-value attribute is specified in its own name/value pair.

attributeType
[out] The type of entity this describes: Profile(P), Request-only(R), Resource(E).

reqacctsig
[out] The action ID if this is a resource attribute

7.2.297 WFRequestAttrsSet

Sets the request attributes of the recipient of the given request.

```
struct WFRequestAttrsSet_scalarInput
    {
    var $batchsig
    var $sessionid
    var $serverid
    var $includeRedundant
    struct WFRequestAttrsSet_vectorInput $inputArray[]
    };

struct WFRequestAttrsSet_vectorInput
    {
    var $name
    var $value
    var $reqacctsig
    };
```

Parameters

batchsig
[in] The ID identifying the batch.

sessionid
[in] Identifier for the session that contains the in-progress request.

serverid
[in] Server ID that is processing the request, or empty if the request is running locally.

includeRedundant
[in] 1 for including redundant attributes, otherwise not.

inputArray
[in] An array of structs of type WFRequestAttrsSet_vectorInput

name
[in] The name of the request attribute.

value
[in] The value of the attribute. Values for a multi-value attribute should be specified across a series of name/value pairs.

reqacctsig
[in] Optional: The action ID to set entity attributes to.

7.2.298 WFRequestAuthinfoGet

Retrieve the authorization information about a request.

```
struct WFRequestAuthinfoGet_scalarInput
    {
    var $batchsig
    var $reqacctsig
    };

struct WFRequestAuthinfoGet_vectorOutput
    {
    var $reqacctsig
    var $phase
    var $nauthreq
    var $nauthrecv
    var $nrejreq
    var $nrejrecv
    };
```

Parameters

batchsig
[in] The ID identifying the batch.

reqacctsig
[in] The action ID.

reqacctsig
[out] The action ID.

phase
[out] The phase of authorization.

nauthreq
[out] Number of approvals required.

nauthrecv
[out] Number of approvals received.

nrejreq
[out] Number of denials required.

nrejrecv
[out] Number of denials received.

7.2.299 WFRequestAuthorizerOpenFind

Lists the open requests where the specified user is an authorizer and they still require input.

```
struct WFRequestAuthorizerOpenFind_scalarInput
    {
    var $authorizer
    };

struct WFRequestAuthorizerOpenFind_vectorOutput
    {
    var $batchsig
    var $isdelegate
    };
```

Parameters

authorizer
[in] The ID of the authorizer.

batchsig
[out] The sighkey of the batch.

isdelegate
[out] Return 1 if this is a delegated certification, 0 if otherwise.

7.2.300 WFRequestCancel

Cancels the given request.

```
struct WFRequestCancel_scalarInput
    {
    var $batchsig
    var $reqid
    var $reason
    var $serverid
    };
```


Parameters

batchsig
[in] The ID identifying the batch.

reqid
[in] The profile ID of the requester or recipient of the request.

reason
[in] The reason for cancelling the request.

serverid
[in] Server ID that is processing the request, or empty if the request is running locally.

7.2.301 WFRequestCertifierOpenFind

Lists the open segments where the specified user is a reviewer and they still require input.

```
struct WFRequestCertifierOpenFind_scalarInput
    {
    var $certifier
    };

struct WFRequestCertifierOpenFind_vectorOutput
    {
    var $segsig
    var $isdelegate
    };
```

Parameters

certifier
[in] The ID of the authorizer.

segsig
[out] The sighkey of the segment.

isdelegate
[out] Return 1 if this is a delegated certification, 0 if otherwise.

7.2.302 WFRequestCheckin

Check in all passwords requested in a batch, if they have been checked out.

```
struct WFRequestCheckin_scalarInput
    {
    var $batchid
    };
```



```
struct WFRequestCheckin_vectorOutput
    {
    var $status
    var $acctid
    var $wstnid
    var $errmsg
    };
```

Parameters

batchid
[in] A request batch ID for which all requested passwords are to be checked in.

status
[out] The status of this password checkin.

acctid
[out] The account ID for this checkin attempt.

wstnid
[out] The workstation ID for this checkin attempt.

errmsg
[out] The error message for this checkin attempt.

7.2.303 WFRequestCheckout

Check out all passwords requested in a batch, if the request has been approved.

```
struct WFRequestCheckout_scalarInput
    {
    var $batchid
    };

struct WFRequestCheckout_vectorOutput
    {
    var $status
    var $acctid
    var $wstnid
    var $errmsg
    var $password
    var $checkoutid
    };
```

Parameters

batchid
[in] A request batch ID for which all requested passwords are to be checked out.

status

[out] The status of this password checkout.

acctid
[out] The account ID for this checkout attempt.

wstnid
[out] The workstation ID for this checkout attempt.

errmsg
[out] The error message for this checkout attempt.

password
[out] The password for this account, if the checkout was successful.

checkoutid
[out] Check-out id.

7.2.304 WFRequestCreate

Creates a new workflow request. Call WFRequestSubmit to submit the request for processing.

```
struct WFRequestCreate_scalarInput
    {
    var $recipid
    var $recipemail
    var $requesteremail
    var $requestreason
    var $preqid
    var $reserveid
    var $maqid
    var $parentbatchsig
    var $serverid
    var $userdesc
    struct WFRequestCreate_vectorInput $inputArray[]
    };

struct WFRequestCreate_vectorInput
    {
    var $role
    var $nosgroupname
    var $nosgrouphostid
    var $subgrpname
    var $subgrphostid
    var $accountname
    var $accounthostid
    var $resourcehostid
    var $resourceaddress
    var $resgroupid
    var $profilename
    };
```



```
struct WFRequestCreate_vectorOutput
    {
    var $batchsig
    var $sessionid
    var $serverid
    };
```

Parameters

recipid
[in] The profile ID of the person whom this request is for.

recipemail
[in] The email for the recipient.

requesteremail
[in] The email of the requester.

requestreason
[in] The reason for this request.

preqid
[in] Optional pre-defined request ID to base this new request on. If specified, the new request will be initialized with attributes and required actions configured on the pre-defined request.

reserveid
[in] Optional reservation ID returned by ReserveAccountId, ReserveAttrValue, or ReserveProfileId.

maqid
[in] The ID of the account set.

parentbatchsig
[in] Optional ID identifying the parent request.

serverid
[in] Optional server ID to process the request on. Empty serverid means to run the request locally, or the special string "RANDOM" can be supplied to allow selection of a random server.

userdesc
[in] (Used for privileged access requests) The description for the request which can be edited by the user before submitting the request. Use the same value for this as for the request attribute with name "DESC_APP" set with WFRequestAttrsSet.

inputArray
[in] An array of structs of type WFRequestCreate_vectorInput

role
[in] Apply the role-type operations to this topic role.

nosgroupname
[in] Apply the nosgroup-type operations to this topic group.

nosgrouphostid
[in] Apply the nosgroup-type operations to this topic group.

subgrpname
[in] Apply the nosgroup-type operations to this topic sub/child group.

subgrphostid
[in] Apply the nosgroup-type operations to this topic sub/child group.

accountname
[in] Apply the account-type operations to this topic account.

accounthostid
[in] Apply the account-type operations to this topic account.

resourcehostid
[in] Apply the resource-type operations to this topic resource.

resourceaddress
[in] Apply the resource-type operations to this topic resource.

resgroupid
[in] Apply the resource-type operations to this topic resource.

profilename
[in] Apply the profile-type operations to this topic profile.

batchsig
[out] The ID identifying the batch.

sessionid
[out] Identifier for the session that contains the in-progress request.

serverid
[out] Server ID that will process the request.

7.2.305 WFRequestFind

Retrieves all of the requests associated with the given search attribute values. If no search attributes are provided, then all requests are returned. Ranged searches are supported only for date-based search attributes.

```
struct WFRequestFind_scalarInput
    {
    var $options
    struct WFRequestFind_vectorInput $inputArray[]
    };

struct WFRequestFind_vectorInput
    {
    var $name
```



```
    var $value1
    var $value2
    };

struct WFRequestFind_vectorOutput
    {
    var $batchsig
    var $batchname
    var $status
    var $prevstatus
    var $recipid
    var $recipemail
    var $requesterid
    var $requesteremail
    var $notes
    var $reqadmin
    var $sessionid
    var $entrydate
    var $lastauth
    var $reason
    var $canceledby
    var $canceleddate
    var $termdate
    var $startdate
    var $segsig
    var $parentbatchsig
    var $superbatchsig
    var $autoressig
    var $reserveid
    var $delegate
    var $location
    var $accttype
    var $preqid
    };
```

Parameters

options
[in] (Optional) Additional parameters, as key-value pairs in bare KVG format. Currently, these are the only supported parameters:

- SortAttr = <attrname>: A search attribute, as returned by WFRequestFindAttrs, by which to sort the requests returned by WFRequestFind. By default, requests will be sorted by entrydate.

inputArray
[in] An array of structs of type WFRequestFind_vectorInput

name
[in] The name of the search attribute. This is one of the attributes returned by WFRequestFindAttrs.

value1

[in] The value of the search attribute. If value2 is used, then value1 is the minimum of a range of values for the search attribute.

value2
[in] The maximum of a range of values for the search attribute. This is ignored if value1 is not used.

batchsig
[out] The ID identifying the batch.

batchname
[out] Human-readable description of request

status
[out] The status code of the request.

prevstatus
[out] The previous status code of the request before it went on hold.

recipid
[out] The profile ID for the recipient of the request.

recipemail
[out] The email for the recipient.

requesterid
[out] The profile ID of the requester.

requesteremail
[out] The email of the requester.

notes
[out] The reason for this request.

reqadmin
[out] (DEPRECATED) The ID of the console user who submitted the request. This will be used if the request is submitted by proxy.

sessionid
[out] The request's session Id

entrydate
[out] When the request was last entered, in UTC, ISO format.

lastauth
[out] The last authorized/deny event, in UTC, ISO format.

reason
[out] Reasons entered by Bravura Security Fabric

canceledby
[out] The profile ID of the user who canceled the request.

canceleddate

[out] Canceled date, in UTC, ISO format.

termdate
[out] Termination date for this user, in UTC, ISO format.

startdate
[out] Take effect as of this date, in UTC, ISO format.

segsig
[out] Certification segment ID.

parentbatchsig
[out] The ID identifying the parent batch.

superbatchsig
[out] Parent batch.

autoressig
[out] Request that caused this automatic assignment request to be created.

reserveid
[out] Reservation ID for this batch.

delegate
[out] Profile ID of the delegate of the requester that caused this request to be issued.

location
[out] Location.

accttype
[out] Account type.

preqid
[out] Pre-defined request used.

7.2.306 WFRequestFindAttrs

Returns a list of search attributes supported by WFRequestFind.

```
struct WFRequestFindAttrs_scalarInput
    {
    var $unused


    };

struct WFRequestFindAttrs_vectorOutput
    {
    var $name
    var $description
    };
```


Parameters

unused
[in] Not used

name
[out] The search attribute.

description
[out] The description of the search attribute.

7.2.307 WFRequestFindByProfileAttr

Retrieves all of the requests associated with the given profile attribute values. At least one search attribute with one search value is required. Up to five search attributes are allowed; any search attributes specified beyond the first five will be ignored. If a profile attribute is specified more than once, then the search value(s) for the last specified one will be used.

```
struct WFRequestFindByProfileAttr_scalarInput
    {
    struct WFRequestFindByProfileAttr_vectorInput $inputArray[]
    var $unused


    };

struct WFRequestFindByProfileAttr_vectorInput
    {
    var $attrkey
    var $value1
    var $value2
    };

struct WFRequestFindByProfileAttr_vectorOutput
    {
    var $batchsig
    var $batchname
    var $status
    var $prevstatus
    var $recipid
    var $recipemail
    var $requesterid
    var $requesteremail
    var $notes
    var $reqadmin
    var $sessionid
    var $entrydate
    var $lastauth
    var $reason
    var $canceledby
```



```
var $canceleddate
var $termdate
var $startdate
var $segsig
var $parentbatchsig
var $superbatchsig
var $autoressig
var $reserveid
var $delegate
var $location
var $accttype
var $preqid
};
```

Parameters

unused
[in] Not used

inputArray
[in] An array of structs of type WFRequestFindByProfileAttr_vectorInput

attrkey
[in] The profile attribute ID.

value1
[in] The value of the profile attribute. If value2 is used, then value1 is the minimum of a range of values for the search attribute.

value2
[in] The maximum of a range of values for the search attribute.

batchsig
[out] The ID identifying the batch.

batchname
[out] Human-readable description of request

status
[out] The status code of the request.

prevstatus
[out] The previous status code of the request before it went on hold.

recipid
[out] The profile ID for the recipient of the request.

recipemail
[out] The email for the recipient.

requesterid
[out] The profile ID of the requester.

requesteremail
[out] The email of the requester.

notes
[out] The reason for this request.

reqadmin
[out] (DEPRECATED) The ID of the console user who submitted the request. This will be used if the request is submitted by proxy.

sessionid
[out] The request's session Id.

entrydate
[out] When the request was last entered, in UTC, ISO format.

lastauth
[out] The last authorized/deny event in UTC, ISO format.

reason
[out] Reasons entered by Bravura Security Fabric

canceledby
[out] The profile ID of the user who canceled the request.

canceleddate
[out] Canceled date, in UTC, ISO format.

termdate
[out] Termination date for this user, in UTC, ISO format.

startdate
[out] Take effect as of this date, in UTC, ISO format.

segsig
[out] Certification segment ID.

parentbatchsig
[out] The ID identifying the parent batch.

superbatchsig
[out] Parent batch

autoressig
[out] Request that caused this automatic assignment request to be created.

reserveid
[out] Reservation ID for this batch.

delegate
[out] Profile ID of the delegate of the requester that caused this request to be issued.

location

[out] Location.

accttype
[out] Account type.

preqid
[out] Pre-defined request used.

7.2.308 WFRequestGSetCheckin

Check in group memberships requested in a batch, if they have been checked out.

```
struct WFRequestGSetCheckin_scalarInput
    {
    var $batchid
    };

struct WFRequestGSetCheckin_vectorOutput
    {
    var $status
    var $userid
    var $acctid
    var $targetid
    var $policyid
    var $gsetid
    var $errmsg
    };
```

Parameters

batchid
[in] A request batch ID for which all requested group memberships are to be checked out.

status
[out] The status of this group set checkin.

userid
[out] The profile ID of a user.

acctid
[out] The account ID for this checkin attempt.

targetid
[out] The managed system ID for this checkin attempt.

policyid
[out] The ID of the policy of the group set.

gsetid
[out] The ID of the group set.

errmsg
[out] The error message for this checkin attempt.

7.2.309 WFRequestGSetCheckout

Check out group memberships requested in a batch, if the request has been approved.

```
struct WFRequestGSetCheckout_scalarInput
    {
    var $batchid
    };

struct WFRequestGSetCheckout_vectorOutput
    {
    var $status
    var $userid
    var $acctid
    var $targetid
    var $policyid
    var $gsetid
    var $errmsg
    var $checkoutid
    };
```

Parameters

batchid
[in] A request batch ID for which all requested group memberships are to be checked out.

status
[out] The status of this group set checkout.

userid
[out] The profile ID of a user.

acctid
[out] The account ID for this checkout attempt.

targetid
[out] The managed system ID for this checkout attempt.

policyid
[out] The ID of the policy of the group set.

gsetid
[out] The ID of the group set.

errmsg
[out] The error message for this checkout attempt.

checkoutid
[out] Check-out id.

7.2.310 WFRequestGenericCheckin

Check in a generic access request. If this function returns ERR_UNKNOWN, the error message may contain a translatable string with more details. TranslationGet can be used to translate the string.

```
struct WFRequestGenericCheckin_scalarInput
    {
    var $checkoutid
    };
```

Parameters

checkoutid
[in] A check-out ID to check in.

7.2.311 WFRequestGenericCheckout

Check out a generic access request. If this function returns ERR_UNKNOWN, the error message may contain a translatable string with more details. TranslationGet can be used to translate the string.

```
struct WFRequestGenericCheckout_scalarInput
    {
    var $batchid
    var $itemid
    };

struct WFRequestGenericCheckout_vectorOutput
    {
    var $checkoutid
    };
```

Parameters

batchid
[in] A request batch ID for an approved generic access request.

itemid
[in] Reserved. Do not use.

checkoutid
[out] The ID of the check-out, if the function succeeded.

7.2.312 WFRequestGet

Retrieves the configuration of the given request.

```
struct WFRequestGet_scalarInput
    {
    var $batchsig
    var $sessionid
    };

struct WFRequestGet_vectorOutput
    {
    var $batchname
    var $recipid
    var $status
    var $recipemail
    var $requesterid
    var $requesteremail
    var $notes
    var $prevstatus
    var $reqadmin
    var $sessionid
    var $entrydate
    var $lastauth
    var $reason
    var $authnote
    var $canceledby
    var $canceleddate
    var $termdate
    var $startdate
    var $segsig
    var $parentbatchsig
    var $superbatchsig
    var $autoressig
    var $reserveid
    var $delegate
    var $preqid
    };
```

Parameters

batchsig
[in] The ID identifying the batch.

sessionid
[in] Get the request as it exists in the given session. This is useful when a request is actively being created (or modified) but the user has not yet committed their changes and the API must get the information about the request as it is being changed. If this is empty, or if an actively changed request is not found in the given session, the request will be extracted as it exists currently.

batchname

[out] Human-readable description of request

recipid
[out] The profile ID of the recipient.

status
[out] The status code of the request.

recipemail
[out] The email for the recipient.

requesterid
[out] The profile ID of the requester.

requesteremail
[out] The email of the requester.

notes
[out] The reason for this request.

prevstatus
[out] The previous status code of the request before it went on hold.

reqadmin
[out] (DEPRECATED) The ID of the console user that submitted the request. This will be used if the request is submitted by proxy.

sessionid
[out] The request's session ID.

entrydate
[out] When the request was last entered, in UTC, ISO format.

lastauth
[out] The last authorized/deny event, in UTC, ISO format.

reason
[out] Reasons entered by Bravura Security Fabric.

authnote
[out] The note provided by authmod

canceledby
[out] The profile ID of the user who canceled the request.

canceleddate
[out] Canceled date, in UTC, ISO format.

termdate
[out] Termination date for this user, in UTC, ISO format.

startdate
[out] Take effect as of this date, in UTC, ISO format.

segsig
[out] Certification segment ID.

parentbatchsig
[out] The ID identifying the parent batch.

superbatchsig
[out] Parent batch.

autoressig
[out] Request that caused this automatic assignment request to be created.

reserveid
[out] Reservation ID for this batch.

delegate
[out] Profile ID of the delegate of the requester that caused this request to be issued.

preqid
[out] ID of the request, if pre-defined.

7.2.313 WFRequestImplementerOpenFind

Lists batch id, item id and authorizer status of the open requests where the specified user is an implementer and they still require input.

```
struct WFRequestImplementerOpenFind_scalarInput
    {
    var $implementer
    };

struct WFRequestImplementerOpenFind_vectorOutput
    {
    var $batchsig
    var $itemsig
    var $status
    var $isdelegate
    };
```

Parameters

implementer
[in] The ID of the authorizer.

batchsig
[out] The sighkey of the batch.

itemsig
[out] The sighkey of the request action.

status

[out] The status of the implementer for the request action.

isdelegate
[out] Return 1 if this is a delegated certification, 0 if otherwise.

7.2.314 WFRequestMAQCheckin

Check in an account set requested in a batch, if it has been checked out.

```
struct WFRequestMAQCheckin_scalarInput
    {
    var $batchid
    };
```

Parameters

batchid
[in] A request batch ID for which an account set is to be checked in.

7.2.315 WFRequestMAQCheckout

Check out an account set requested in a batch, if the request has been approved.

```
struct WFRequestMAQCheckout_scalarInput
    {
    var $batchid
    };

struct WFRequestMAQCheckout_vectorOutput
    {
    var $status
    var $userid
    var $acctid
    var $wstnid
    var $maqid
    var $password
    var $checkoutid
    };
```

Parameters

batchid
[in] A request batch ID for which an account set is to be checked out.

status
[out] The status of this account set checkout.

userid
[out] The profile ID of a user.

acctid
[out] The account ID for this checkout attempt.

wstnid
[out] The managed system ID for this checkout attempt.

maqid
[out] The ID of the account set.

password
[out] The password for this account, if the checkout was successful.

checkoutid
[out] Check-out id.

7.2.316 WFRequestOperationTypes

Returns a list of allowed operation types for assignment to requests.

See Workflow Request Operations for the complete list of operation types returned by this function.

```
struct WFRequestOperationTypes_scalarInput
    {
    var $unused


    };

struct WFRequestOperationTypes_vectorOutput
    {
    var $name
    var $description
    };
```

Parameters

unused
[in] Not used

name
[out] The name of the operation.

description
[out] The description of the operation.

7.2.317 WFRequestPasswordPolicyGet

Get the password policies and request item IDs involved in a specified request.


```
struct WFRequestPasswordPolicyGet_scalarInput
    {
    var $batchsig
    var $sessionid
    };

struct WFRequestPasswordPolicyGet_vectorOutput
    {
    var $ppid
    var $reqacctsig
    };
```

Parameters

batchsig
[in] The ID identifying the batch, given the batchid, return a list of strength rules along with template ids for adding new accounts in the request

sessionid
[in] Get the request as it exists in the given session. This is useful when a request is actively being created (or modified) but the user has not yet committed their changes and the API must get the information about the request as it is being changed. If this is empty, or if an actively changed request is not found in the given session, the request will be extracted as it exists currently.

ppid
[out] The ID of the password policy.

reqacctsig
[out] The action ID.

7.2.318 WFRequestPasswordSet

Set passwords for specified password policies in a particular request.

```
struct WFRequestPasswordSet_scalarInput
    {
    var $batchsig
    var $sessionid
    var $serverid
    var $recippass
    struct WFRequestPasswordSet_vectorInput $inputArray[]
    };

struct WFRequestPasswordSet_vectorInput
    {
    var $ppid
    var $password
    };
```

Parameters

batchsig
[in] The ID identifying the batch.

sessionid
[in] Identifier for the session that contains the in-progress request.

serverid
[in] Server ID that is processing the request, or empty if the request is running locally.

recippass
[in] The initial password for the recipient's accounts, used when password is not specified in the policy, password pair below.

inputArray
[in] An array of structs of type WFRequestPasswordSet_vectorInput

ppid
[in] The ID of the password policy.

password
[in] Password used for this password policy.

7.2.319 WFRequestSet

Configures the given request.

```
struct WFRequestSet_scalarInput
    {
    var $batchsig
    var $sessionid
    var $serverid
    var $batchname
    var $recipid
    var $recipemail
    var $requesteremail
    var $requestreason
    var $reserveid
    };
```

Parameters

batchsig
[in] The ID identifying the batch.

sessionid
[in] Identifier for the session that contains the request, if the request has not been submitted yet.

serverid
[in] Server ID that is processing the request, or empty if the request is running locally.

batchname
[in] Human-readable description of request

recipid
[in] The profile ID of the recipient.

recipemail
[in] The email for the recipient.

requesteremail
[in] The email of the requester.

requestreason
[in] The reason for this request.

reserveid
[in] Optional reservation ID returned by ReserveAccountId, ReserveAttrValue, or ReserveProfileId.

7.2.320 WFRequestStatus

Returns the current status of a given request, including the status of any associated actions.

```
struct WFRequestStatus_scalarInput
    {
    var $batchsig
    };

struct WFRequestStatus_vectorOutput
    {
    var $reqstatus
    var $reqreason
    var $operationType
    var $reqacctsig
    var $acctid
    var $newid
    var $dstuserid
    var $groupid
    var $targetid
    var $newctx
    var $statusreason
    var $notes
    var $parentrole
    var $actreason
    var $resmemberid
    var $roleid
    var $sodid
    var $srcuserid
    var $actstatus
    var $tplid
    var $tpluri
```



```
var $userid
var $resourceaddress
var $resourcetype
};
```

Parameters

batchsig
[in] The ID identifying the batch.

reqstatus
[out] The status code of the request.

reqreason
[out] The reason, if any, for the status of the request.

operationType
[out] This type of operation. This is one of the types returned by WFRequestOperationTypes.

reqacctsig
[out] The action ID.

acctid
[out] The account ID on the target. Used by the ARCHREQPWD, CFYA, CFYG, DELU, DNAU, ENAU, GRUA, GROA, GROD, GRUD, MVCU, RENU, LUPD, LDEL, and UPDT action types.

newid
[out] The new account ID used by the RENU, RPRO action types.

dstuserid
[out] The profile ID of a destination user. Used by the CFYR, ORGTFRSUB and ORGTFRSUBPULL action types.

groupid
[out] The ID of a managed group on the target. Used by the CFYG, GROA, GROD, GRUA, and GRUD action types.

targetid
[out] The ID of a target. Used by the ARCHREQPWD, CFYA, CFYG, DELU, DNAU, ENAU, GROA, GROD, GRUA, GRUD, MVCU, RENU, LUPD, LDEL, and UPDT action types.

newctx
[out] The value of the new context. Used by the MVCU action type.

statusreason
[out] The reason for the status.

notes
[out] An explanation, if any, for the status of the action. Used by all action types.

parentrole
[out] The ID of a parent role. Used by the ACUA, RLUA, DELR, and GRUA action types.

actreason
[out] The reason for the action. Used by all action types.

resmemberid
[out] The ID of a resource member. Used by the AEDEFICIT action type.

roleid
[out] The ID of a role. Used by the CFYAEDEF, CFYR, RLUA, DELR, and GRUA action types.

sodid
[out] The ID of an SOD rule. Used by the AESOD action type.

srcuserid
[out] The profile ID of a source user. Used by the ORGTFRSUB and ORGTFRSUBPULL action type.

actstatus
[out] The status code of the action. Used by all action types.

tplid
[out] The ID of a template. Used by the ACUA, GRUA, and DELU action types.

tpluri
[out] The URI template. Used by NRCR, NRUP, NRDL, NRMV action types.

userid
[out] The profile ID of a user. Used by the CFYU, DELR, ORGADDMGR, ORGADDSUB, ORGDELMGR, ORGDELSUB, ORGTFRSUB, ORGTFRSUBPULL, LUPD, and RENU action types.

resourceaddress
[out] The resource address. Used by NRCR, NRUP, NRDL, NRMV action types.

resourcetype
[out] The resource type. Used by NRCR, NRUP, NRDL, NRMV action types.

7.2.321 WFRequestSubmit

Submits the given request for processing. Returns only after the request is posted and any related plugins have been executed.

```
struct WFRequestSubmit_scalarInput
    {
    var $batchsig
    var $sessionid
    var $serverid
    };
```

Parameters

batchsig
[in] The ID identifying the batch.

sessionid
[in] Identifier for the session that contains the in-progress request.

serverid
[in] Server ID that is processing the request, or empty if the request is running locally.

7.2.322 WFRequestTopicsGet

Retrieves topics of the given request.

```
struct WFRequestTopicsGet_scalarInput
    {
    var $batchsig
    var $sessionid
    };

struct WFRequestTopicsGet_vectorOutput
    {
    var $role
    var $nosgroupname
    var $nosgrouphostid
    var $subgrpname
    var $subgrphostid
    var $accountname
    var $accounthostid
    var $resourcehostid
    var $resourceaddress
    var $resgroupid
    var $profilename
    };
```

Parameters

batchsig
[in] The ID identifying the batch.

sessionid
[in] Get the request as it exists in the given session. This is useful when a request is actively being created (or modified) but the user has not yet committed their changes and the API must get the information about the request as it is being changed. If this is empty, or if an actively changed request is not found in the given session, the request will be extracted as it exists currently.

role
[out] Apply the role-type operations to this topic role.

nosgroupname
[out] Apply the nosgroup-type operations to this topic group.

nosgrouphostid
[out] Apply the nosgroup-type operations to this topic group.

subgrpname
[out] Apply the nosgroup-type operations to this topic sub/child group.

subgrphostid
[out] Apply the nosgroup-type operations to this topic sub/child group.

accountname
[out] Apply the account-type operations to this topic account.

accounthostid
[out] Apply the account-type operations to this topic account.

resourcehostid
[out] Apply the resource-type operations to this topic resource.

resourceaddress
[out] Apply the resource-type operations to this topic resource.

resgroupid
[out] Apply the resource-type operations to this topic resource.

profilename
[out] Apply the profile-type operations to this topic profile.

7.2.323 WFResultSet

Sets the result of an implementation task for the logged-in user.

```
struct WFResultSet_scalarInput
    {
    var $batchsig
    var $reqacctsig
    var $status
    var $reason
    var $primary
    };
```

Parameters

batchsig
[in] The ID identifying the batch.

reqacctsig
[in] The action ID for this task.

status
[in] The status of the task. 0 is failure, 1 is success.

reason
[in] The reason for the result.

primary
[in] The profile ID of the user on whose behalf this is being done. If blank, then delegation is not used.

7.2.324 WFSendMail

Sends an email, via the workflow manager, to either the requester or recipient of a given request, or the authorizer or implementer of an action task associated with the request.

```
struct WFSendMail_scalarInput
    {
    var $batchsig
    var $targetuser
    var $subjecttag
    var $messagetag
    var $notifylevel
    };
```

Parameters

batchsig
[in] The ID identifying the batch.

targetuser
[in] The profile ID of the user receiving the email. The email address used is from the EMAIL request attribute set for the profile or the output from IDSYNCH USERS EMAIL PLUGIN.

subjecttag
[in] Subject tag in emails.

messagetag
[in] Message tag in emails.

notifylevel
[in] The notification level of the email for authorizers. 0 is Reminder, 1 is Initial, 2 is NeverMind, 3 is ThankYou, 4 is Final, 5 is Processing.

7.2.325 WebAppDelete

Delete the specified web app.

```
struct WebAppDelete_scalarInput
    {
    var $id
    };
```

Parameters

id
[in] The web app's id

7.2.326 WebAppGet

Retrieves information for a specified web app.

```
struct WebAppGet_scalarInput
    {
    var $id
    };

struct WebAppGet_vectorOutput
    {
    var $id
    var $name
    var $description
    var $filename
    var $creatorguid
    var $shared
    };
```

Parameters

id
[in] The web app's id

id
[out] The web app's ID

name
[out] The web app's name

description
[out] The web app's description

filename
[out] The web app's file name. Can be empty

creatorguid
[out] The profile GUID of web app's creator

shared
[out] Flag indicating if the web app is shared between users. 0 = not shared, 1 = is shared.

7.2.327 WebAppJsonGet

Retrieves JSON for a specified web app.

```
struct WebAppJsonGet_scalarInput
    {
    var $id
    };
```



```
struct WebAppJsonGet_vectorOutput
    {
    var $json
    };
```

Parameters

id
[in] The web app's id

json
[out] The web app's JSON config. Sent in consecutive chunks which need to be reassembled. When combined, the length will not be more than 4194304

7.2.328 WebAppJsonValidate

Validate web app JSON config.

```
struct WebAppJsonValidate_scalarInput
    {
    var $userid
    struct WebAppJsonValidate_vectorInput $inputArray[]
    };

struct WebAppJsonValidate_vectorInput
    {
    var $json
    };

struct WebAppJsonValidate_vectorOutput
    {
    var $valid
    var $error
    };
```

Parameters

userid
[in] Optional. The profile ID of the user. This is used to return the error message in the appropriate language. The default language will be used if a profile ID is not supplied.

inputArray
[in] An array of structs of type WebAppJsonValidate_vectorInput

json
[in] The web app JSON config to validate. Send as a list of substrings of length 4096 or less, which when combined are no longer than 4194304

valid
[out] Indicates if the web app JSON is valid

error
[out] If valid is false, this will contain an error message indicating the problem

7.2.329 WebAppList

Returns a list of all web apps

```
struct WebAppList_scalarInput
    {
    var $unused


    };

struct WebAppList_vectorOutput
    {
    var $id
    var $name
    var $description
    var $filename
    var $creatorguid
    var $shared
    };
```

Parameters

unused
[in] Not used

id
[out] The web app's ID

name
[out] The web app's name

description
[out] The web app's description

filename
[out] The web app's file name. Can be empty

creatorguid
[out] The profile GUID of web app's creator

shared
[out] Flag indicating if the web app is shared between users. 0 = not shared, 1 = is shared.

7.2.330 WebAppRemoveOverride

Remove a web app override for user on a specific account if it should no longer be allowed to be chosen.

```
struct WebAppRemoveOverride_scalarInput
    {
    var $pluginctrlid
    var $profileguid
    var $accountguid
    var $webappid
    };
```

Parameters

pluginctrlid
[in] The web app disclosure GUID

profileguid
[in] The user's profile GUID

accountguid
[in] The managed account GUID

webappid
[in] The web app's ID

7.2.331 WebAppSet

Add or Update a web app.

```
struct WebAppSet_scalarInput
    {
    var $id
    var $name
    var $description
    var $filename
    var $creatorguid
    var $shared
    struct WebAppSet_vectorInput $inputArray[]
    };

struct WebAppSet_vectorInput
    {
    var $json
    };
```

Parameters

id
[in] The web app's ID

name
[in] The web app's name

description
[in] The web app's description. Blank entries accepted

filename
[in] The web app's file name. Can be empty

creatorguid
[in] Optional. The profile GUID of web app's creator. This must be set if the web app is not shared, otherwise it can be empty

shared
[in] Flag indicating if the web app is shared between users. 0 = not shared, 1 = is shared.

inputArray
[in] An array of structs of type WebAppSet_vectorInput

json
[in] The web app's JSON config. Send as a list of substrings of length 4096 or less, which when combined are no longer than 4194304. Required for create, and optional for update.

7.2.332 agentFinalResponse

Inform the transaction manager that an operation that was flagged as giving an asynchronous response has finished.

```
struct agentFinalResponse_scalarInput
    {
    var $batchsig
    var $itemsig
    var $succeeded
    var $agentresponse
    };
```

Parameters

batchsig
[in] The ID identifying the batch.

itemsig
[in] The action ID for this task.

succeeded
[in] "1" if the task is successed, "0" otherwise.

agentresponse
[in] The reply returned from the agent run

Exit Trap Function Reference 8

The following functions are available in exit trap programs, such as **pxnull**, that can run PSLANG scripts.

8.1 Help Desk System Functions

8.1.1 initializeInterface

Platform: all
Details
Sets up the help desk interface for use

```
int initializeInterface (
        array array ,
        string errmsg
)
```

Parameters
array
[in] Associative array of key-values for the new ticket.

errmsg
[out] Error message on failure.

Return values
Returns 1 (true) on success, 0 on failure.

8.1.2 shutdownInterface

Platform: all
Details
Shuts down the help desk interface

```
int shutdownInterface (
        string errmsg
)
```

Parameters

errmsg
[out] Error message on failure.

Return values
Returns 1 (true) on success, 0 on failure.

8.1.3 create

Platform: all
Details
Create a help desk ticket.

```
int create (
        array array ,
        string ticketid ,
        string errmsg
)
```

Parameters
array
[in] Associative array of key-values for new ticket.

ticketid
[out] The ticket ID that was created.

errmsg
[out] Error message on failure.

Return values
Returns 1 (true) on success, 0 on failure.

8.1.4 query

Platform: all
Details
Execute a statement on a system in the system specific query language.

```
int query (
        string sqlStatement ,
        string errmsg
)
```

Parameters
sqlStatement
[in] A statement to be executed in a platform specific language.

errmsg
[out] Error message on failure.

Return values
Returns 0 on failure, 1 on success.

8.1.5 nextRecord

Platform: all
Details
Gets the next result from a set of results using the handle from query.

```
int nextRecord(
        array row,
        string errmsg
)
```

Parameters
row
[out] The next row in the results.

errmsg
[out] Error message on failure.

Return values
Returns 0 on failure, one on more results, and 2 on end of result set.

8.1.6 search

Platform: all
Details
Search for a help desk ticket.

```
int search(
        array array,
        string ticketid,
        string errmsg
)
```

Parameters
array
[in] Associative array of key-value search terms.

ticketid
[out] The ticket ID that was found.

errmsg
[out] Error message on failure.

Return values
Returns 1 (true) on success, 0 on failure.

8.1.7 update

Platform: all
Details
Update a help desk ticket.

```
int update (
        array array ,
        string errmsg
)
```

Parameters
array
[in] Associative array of key-values to update in the current ticket.

errmsg
[out] Error message on failure.

Return values
Returns 1 (true) on success, 0 on failure.

8.1.8 getField

Platform: all
Details
Find the value a field in the current help desk ticket

```
int getField (
        string field ,
        string value ,
        string errmsg
)
```

Parameters
field
[in] The field to get the value for.

value
[out] The value of the requested field.

errmsg
[out] Error message on failure.

Return values
Returns 1(true) on success, 0 on failure.

Connector Function Reference 9

The functions in this chapter are available to connectors, such as the connector for Win32 console scripts (**agtdos**), that can run PSLANG scripts.

9.1 Connector Functions

9.1.1 readtelnetscreen

Platform: win32
Details
Reads a string from an telnet terminal screen.

```
int readtelnetscreen (
        handle handle ,
        int option ,
        string buffer ,
        int row ,
        int column ,
        int n
)
```

Parameters
handle
[in] The I/O handle to read from.

option
[in] Writetelnet option. See Writetelnet options for a list of possible values.

buffer
[out] The buffer in which to place the results.

row
[in] The row to start reading.

column
[in] The column to start reading.

n
[in] The number of characters to read from screen position (row,column). The trailing space will be

trimmed.

Return values
See I/O Return Values for a list of possible return values.

9.1.2 writetelnet

Platform: win32
Details
Writes a string in some telnet protocol to an I/O handle.

```
int writetelnet (
        handle handle ,
        int option ,
        string buffer ,
        string pattern ,
        int timeout
)
```

Parameters
handle
[in] The I/O handle to write to.

option
[in] Writetelnet option. See Writetelnet options for a list of possible values.

buffer
[in] The buffer to write. See Writetelnet options for the format of each entry. By default, the page is terminated by the Enter Key. You can specify that the page is submitted by pressing a function key rather than the Enter key. Function keys can be submitted using the following format, where NN is a two-digit decimal number, zero padded, in the range 01 to 24:

“pfNN:row1,col1,width1,text1,row2,col2,width2,text2,...”

Every data item in the buffer, such as row, col, width, and text, can be literal value or a variable.

“1,5,$passwidth,$password,$x,$y,...”

Function keys can also be submitted without sending any text:

“pf03:”

In addition to the 24 function keys, the statement also supports the clear key:

“clear:”

When option is relative writing, pslang parses the data stream from the target system to locate the current cursor position, and moves to the next unprotected (writable) field in the screen. The buffer requires only the width of the text, and the text to write. For example:

“width1,text1,width2,text2,...”

The above example writes text1 to the first unprotected field, text2 to the second unprotected field and so on. A tab key can also be sent to skip an unprotected field, for example:

“0,tab,width2,text2,...”

An example for Matching and relatively writing:

“*Your Password*” “8,$password,8,$userid....”

PSLang looks for the line contains “Your Password” and start writing from this line.

pattern
[in] A pattern matching the line to which the buffer should be written. This is only used for matching and relative writing and ignored by other options.

timeout
[in] Timeout in seconds. A timeout of zero means to wait until the specified number of characters have been written.

Return values
See I/O Return Values for a list of possible return values.

9.1.3 connect

Platform: win32
Details
Connects to a server over TCP/IP without using SSL.

```
int connect (
        handle handle ,
        string hostname ,
        int port ,
        int timeout ,
        int option
)
```

Parameters
handle
[out] The opened I/O handle. You must call close() on this handle when finished, even if connect() returns an error code.

hostname
[in] Connect to this host.

port
[in] The TCP port to connect to on hostname.

timeout
[in] Timeout in seconds. A timeout of zero means to wait until the connection has been established, or until the operating system determines that the connection cannot be made.

option
[in] Protocol type. 0 — plain socket; 1 — 3270; 2 — 5250.

Return values
See I/O Return Values for a list of possible return values.

9.1.4 connectSSL

Platform: win32
Details
Connects to a server over TCP/IP using SSL.

```
int connectSSL(
        handle handle,
        string hostname,
        int port,
        int timeout,
        string CAfile,
        string CApath
)
```

Parameters
handle
[out] The opened I/O handle. You must call close() on this handle when finished, even if connect() returns an error code.

hostname
[in] Connect to this host.

port
[in] The TCP port to connect to on hostname.

timeout
[in] Timeout in seconds. A timeout of zero means to wait until the connection has been established, or until the operating system determines that the connection cannot be made.

CAfile
[in] An optional file name containing one or more Certificate Authority certificates in PEM format. Pass a blank string if you have no such file.

CApath
[in] An optional path pointing to a directory containing Certificate Authority (CA) certificates in PEM format. Each file contains one CA certificate. The files are looked up by the CA subject name hash value, which must hence be available. If more than one CA certificate with the same name hash value exists, the extension must be different (e.g. 9d66eef0.0, 9d66eef0.1 etc). The search is performed in the ordering of the extension number, regardless of other properties of the certificates. Pass a blank string if you have no such path.

Return values
See I/O Return Values for a list of possible return values.

9.1.5 connectSSL2

Platform: win32
Details
Connects to a server over TCP/IP using SSL with telnet protocol supported.

```
int connectSSL2 (
        handle handle ,
        string hostname ,
        int port ,
        int timeout ,
        string CAfile ,
        string CApath ,
        int option
)
```

Parameters
handle
[out] The opened I/O handle. You must call close() on this handle when finished, even if connect() returns an error code.

hostname
[in] Connect to this host.

port
[in] The TCP port to connect to on hostname.

timeout
[in] Timeout in seconds. A timeout of zero means to wait until the connection has been established, or until the operating system determines that the connection cannot be made.

CAfile
[in] An optional file name containing one or more Certificate Authority certificates in PEM format. Pass a blank string if you have no such file.

CApath
[in] An optional path pointing to a directory containing Certificate Authority (CA) certificates in PEM format. Each file contains one CA certificate. The files are looked up by the CA subject name hash value, which must hence be available. If more than one CA certificate with the same name hash value exists, the extension must be different (e.g. 9d66eef0.0, 9d66eef0.1 etc). The search is performed in the ordering of the extension number, regardless of other properties of the certificates. Pass a blank string if you have no such path.

option
[in] Protocol type. 0 — plain socket; 1 — 3270; 2 — 5250. Add 16 to the selected protocol type to disable the server certificate check.

Return values
See I/O Return Values for a list of possible return values.

9.1.6 authenticatePeer

Platform: win32
Details
Used with SSL connections after a successful connect. If you are connecting to a server not under your control, you should close any connections where authenticatePeer() does not return zero, as the server's SSL certificate is invalid in some way.

```
int authenticatePeer (
        handle handle
)
```

Parameters
handle
[in] An I/O handle opened with connectSSL().

Return values
A return value from the OpenSSL system. See SSL Peer Authentication for all of the possible values. Zero is returned when the server was successfully authenticated, otherwise a non-zero value is returned representing some failure to authenticate the server. If the given handle is not an SSL socket, -1 is returned.

9.1.7 agentError

Platform: all
Details
Report an error message.

```
void agentError (
        string message
)
```

Parameters
message
[in] Message to report.

Return values
None.

9.1.8 agentWarning

Platform: all
Details
Report a warning message.

```
void agentWarning (
        string message
```


)

Parameters
message
[in] Message to report.

Return values
None.

9.1.9 agentInfo

Platform: all
Details
Report an information message.

```
void agentInfo (
        string message
)
```

Parameters
message
[in] Message to report.

Return values
None.

9.1.10 agentOutput

Platform: win32
Details
Add an agent output

```
void agentOutput (
        string key ,
        string value
)
```

Parameters
key
[in] Key

value
[in] Value

Return values
None.

9.1.11 agentAddAccount

Platform: win32
Details
Called once for each account during the listusers operation.

```
void agentAddAccount (
        string stableId ,
        string accountId ,
        string shortId ,
        kvgroup attrs ,
        array groups
)
```

Parameters
stableId
[in] An account's stableid. Ideally this ID does not change when an account is renamed. If no such ID exists then the accountId should be specified here.

accountId
[in] An account's unique identifier. This Id will be supplied back to the connector for subsequent operations.

shortId
[in] An account's short ID. This Id can be leveraged to auto-assoicate the account to a user profile.

attrs
[const] An account's attributes (optional). A kvgroup list of attributes for the account. NOTE: single-valued attributes should be specified as key/value pairs. Multi-valued attributes should be specified as value only sub-groups. The account's fullname and security domain can be specified by using the $GAttrFullName and $GAttrSecurityDomain pesudo attributes.

groups
[const] A multi-dimentional array of groups. Each element in the array represents a single group and is itself an associative array of group attributes including: stableid, longid, securityDomain (optional) of the group.

Return values
None.

9.1.12 agentAddGroup

Platform: win32
Details
Called once for each group during the listgroups operation.


```
void agentAddGroup(
        string stableId,
        string groupId,
        string shortId,
        kvgroup attrs,
        array owners,
        array members
)
```

Parameters
stableId
[in] A group's stableid. Ideally this ID does not change when a group is renamed. If no such ID exists then the groupId should be specified here.

groupId
[in] A group's unique identifier. This Id will be supplied back to the connector for subsequent operations.

shortId
[in] A group's short ID. This Id will generally be more user friendly but may not be completely unique.

attrs
[const] A group's attributes (optional). A kvgroup list of attributes for the account. NOTE: single-valued attributes should be specified as key/value pairs. Multi-valued attributes should be specified as value only sub-groups. The group's description, security domain, type, systemid, and if its a security or not can be specifiedy by using the $GattrDescription, $GAttrSecurityDomain, $GAttrGroupType, $GAttrSystemid, $GAttrSecurity, pesudo attributes.

owners
[const] A multi-dimentional array of group owners. Each element in the array represents a single owner and is itself an associative array of attributes including: stableid, longid, securityDomain (optional), and type (account|group) of the owner.

members
[const] A multi-dimentional array of group members. Each element in the array represents a single member and is itself an associative array of attributes including: stableid, longid, securityDomain (optional), and type (account|group) of the member.

Return values
None.

9.1.13 agentAddComputer

Platform: win32
Details
Called once for each computer during the listcomputers operation.


```
void agentAddComputer (
        string computerId ,
        string computerDesc ,
        int disabled ,
        kvgroup attrs
)
```

Parameters
computerId
[in] Universally unique identifer for this computer (required).

computerDesc
[in] Descriptive name for this computer resource.

disabled
[in] Flag to indicate if this computer is disabled (0=false,1=true).

attrs
[const] A computer's attributes (optional). A kvgroup list of attributes for the computer. NOTE: single-valued attributes should be specified as key/value pairs. Multi-valued attributes should be specified as value only sub-groups. These attributes can be used to classify computer object for import rule calculations.

Return values
None.

9.1.14 agentAddSubscriber

Platform: win32
Details
Called once for each subscriber during the listsubscribers operation.

```
void agentAddSubscriber (
        string subId ,
        string subDesc ,
        int disabled ,
        string subacct ,
        string stableid ,
        string longid ,
        string shortid ,
        string securityDomain ,
        kvgroup attrs
)
```

Parameters
subId
[in] Universally unique identifer for this subscriber (required).

subDesc
[in] Descriptive name for this subscriber.

disabled
[in] Flag to indicate if this subscriber is disabled or not (0=false,1=true).

subacct
[const] Account name as seen by the object using it (required).

stableid
[const] Stable identifier for the account (required). If the account does not have a stableid the longid should be specified here as well.

longid
[const] Unique identifier for the account e.g. FQDN (required).

shortid
[const] Short identifer for this account e.g CN (required).

securityDomain
[const] Security domian for this account e.g DOMAIN.

attrs
[const] A subscriber's attributes (optional). A kvgroup list of attributes for the subscriber. NOTE: single-valued attributes should be specified as key/value pairs. Multi-valued attributes should be specified as value only sub-groups. These attributes can be used to help determine what subscribers need management and by what policies.

Return values
None.

9.1.15 agentListAttribute

Platform: all
Details
Called once for each attribute during a list attribute operation.

```
void agentListAttribute (
        string name ,
        array value
)
```

Parameters
name
[in] Attribute name.

value
[in] Attribute values. An array of strings; one value per array element.

Return values
None.

9.1.16 agentListUnassigned

Platform: win32
Details
Gets inf unassigned objects requested to be listed.

```
int agentListUnassigned(

)
```

Parameters
None.

Return values
Return value indicates if unassinged or assigned objects should be listed. 0 - assinged objects, 1 - unassigned objects.

9.1.17 agentIsEnabled

Platform: all
Details
Tells the agent whether or not the given user is enabled.

```
void agentIsEnabled(
        int enabled
)
```

Parameters
enabled
[in] 1 if the user is enabled, 0 if not.

Return values
None.

9.1.18 agentIsLocked

Platform: all
Details
Tells the agent whether or not the given user is locked out.

```
void agentIsLocked(
        int enabled
)
```

Parameters
enabled
[in] 1 if the user is locked, 0 if not.

Return values
None.

9.1.19 agentIsPassExpired

Platform: all
Details
Tells the agent whether or not the given user's password has expired.

```
void agentIsPassExpired(
        int enabled
)
```

Parameters
enabled
[in] 1 if the user's password is expired, 0 if not.

Return values
None.

9.1.20 agentIsAcctExpired

Platform: all
Details
Tells the agent whether or not the given user's account has expired.

```
void agentIsAcctExpired(
        int enabled
)
```

Parameters
enabled
[in] 1 if the user's account is expired, 0 if not.

Return values
None.

9.1.21 agentStableId

Platform: win32
Details
Provides the stableid ID of an account or group during create, update, rename, or move context operations.

```
void agentStableId(
        string stableid
```


)

Parameters
stableid
[in] Stable ID.

Return values
None.

9.1.22 agentLongId

Platform: all
Details
Provides the long ID of the user during create, update, rename, or move context operations.

```
void agentLongId (
        string longid
)
```

Parameters
longid
[in] Long ID.

Return values
None.

9.1.23 agentShortId

Platform: all
Details
Provides the short ID of the user during create, update, rename, or move context operations.

```
void agentShortId (
        string shortid
)
```

Parameters
shortid
[in] Short ID.

Return values
None.

9.1.24 agentGroups

Platform: all
Details

Provides the list of groups to which a user belongs.

```
void agentGroups (
        array groups
)
```

Parameters
groups
[const] An array of group IDs.

Return values
None.

9.1.25 agentCommandOutput

Platform: all
Details
Short output from the run command operation

```
void agentCommandOutput (
        string commandOutput
)
```

Parameters
commandOutput
[in] Command Output.

Return values
None.

9.1.26 agentGetConnection

Platform: win32
Summary: Get the current network connection handle.
Details
Return the network session handle for further read/write/expect calls.

```
handle agentGetConnection (

)
```

Parameters
None.

Return values
Returns the opened network I/O handle if an SSH connection, otherwise NULL. No need to close, the agent will close it when the operation is finished.

9.1.27 agentDescription

Platform: all
Details
Provides the group description during create and update.

```
void agentDescription (
        string description
)
```

Parameters
description
[in] Description.

Return values
None.

9.1.28 connectSSH

Platform: win32
Summary: Connect to SSH target from PSLang.
Details
Connects to the current server using the new credentials.

```
int connectSSH (
        string username ,
        string password
)
```

Parameters
username
[in]

password
[in]

Return values
See I/O Return Values for a list of possible return values.

9.2 Deprecated Functions

The functions in this section have all been deprecated. Please migrate your scripts to use the replacement functions. (**`deprecated`**).

9.2.1 agentListUser

Platform: all
Summary: DEPRECATED
Details
The function is deprecated, use agentAddAccount() instead.

```
void agentListUser (
        string longId ,
        string shortId ,
        string fullName ,
        kvgroup attrs ,
        array groups
)
```

Parameters
longId
[in] A user's long ID.

shortId
[in] A user's short ID.

fullName
[in] A user's full name.

attrs
[const] A user's attributes (optional).

groups
[const] A user's groups.

Return values
None.

9.2.2 agentListGroup

Platform: all
Summary: DEPRECATED
Details
The function is deprecated, use agentAddGroup() instead.


```
void agentListGroup (
        string longId ,
        string shortId ,
        string description ,
        array members ,
        array managers
)
```

Parameters
longId
[in] A group's long ID.

shortId
[in] A group's short ID.

description
[in] A group's description.

members
[const] A group's members.

managers
[const] A group's managers.

Return values
None.

9.2.3 agentListGroupSID

Platform: win32
Summary: DEPRECATED
Details
The function is deprecated, use agentAddGroup() instead.

```
void agentListGroupSID (
        string longId ,
        string shortId ,
        string description ,
        string sid ,
        array members ,
        array managers
)
```

Parameters
longId
[in] A group's long ID.

shortId
[in] A group's short ID.

description
[in] A group's description.

sid
[in] A group's unique ID.

members
[const] A group's members.

managers
[const] A group's managers.

Return values
None.

9.2.4 agentListGroupEx

Platform: all
Summary: DEPRECATED
Details
The function is deprecated, use agentAddGroup() instead.

```
int agentListGroupEx (
        string longId ,
        string shortId ,
        string description ,
        array members ,
        array managers ,
        array groupmembers ,
        array groupmanagers ,
        kvgroup attributes
)
```

Parameters
longId
[in] This group's long identifier.

shortId
[in] This group's short identifier.

description
[in] The description of this group.

members
[const] List of user members.

managers
[const] List of user managers.

groupmembers

[const] List of subgroups.

groupmanagers
[const] List of group managers of this group.

attributes
[const] Free-form attributes.

Return values
Zero on success, One if an error parsing any other the field and the group is not added.

9.2.5 agentListResourceComputer

Platform: all
Summary: DEPRECATED
Details
The function is deprecated, use agentAddComputer() instead.

```
void agentListResourceComputer (
        string id ,
        string name ,
        int disabled ,
        kvgroup attrs
)
```

Parameters
id
[in] Universally unique identifer for this computer (required).

name
[in] Descriptive name for this computer resource.

disabled
[in] Flag to indicate if this computer is disabled (0=false,1=true).

attrs
[const] Free-form attributes for this computer (optional - for import rules and classification).

Return values
None.

9.2.6 agentListResourceAccount

Platform: all
Summary: DEPRECATED
Details
The function is deprecated, use agentAddSubscriber() instead.


```
void agentListResourceAccount (
        string id ,
        string name ,
        int disabled ,
        string acctid ,
        string longid ,
        string shortid ,
        kvgroup attrs
)
```

Parameters

id
[in] Universally unique identifer for this account (optional - will auto-generate to longid+targetid).

name
[in] Descriptive name for the object using this account.

disabled
[in] Flag to indicate if this object is disabled (0=false,1=true).

acctid
[const] Account name as seen by the object using it (required).

longid
[const] Long identifier for the account e.g. FQDN (required).

shortid
[const] Short identifer for this account e.g CN (required).

attrs
[const] Free-form attributes for this object/account (optional - for import rules and classification).

Return values
None.

PSLang Scripts for interface programs

10

All interface programs require PSLANG scripts to configure event actions. Configuration file names match the appropriate interface program; for example write **pxnull.cfg** to configure event actions triggered by the **pxnull** program. Programs are listed in IT Service Management (Ticket) Integration Guide (**itsm-ticket-systems.p**

Modify the appropriate sample script in the samples directory and place it in the \<*instance*>\script\ directory. If you cannot find the sample file, try re-running **setup** to modify your installation. Sample files are automatically installed only with complete installations. You can select them in custom installations.

Read this chapter to learn more about the functions and global variables used by interface programs.

10.1 Functions

When creating a configuration file, use the event option name as the function name. For example, if you want **pxnull** to handle failed administrator login attempts, enable the **USER LOGIN FAILURE** option, and include a USER_LOGIN_FAILURE entry function in your **pxnull.cfg** script.

10.2 Global variables

When an interface program runs a function, it automatically defines three associative array variables at global scope. These variables are:

general Contains key-value pairs that describe the event, such as the operation name, date and time of the event, as well as the IP address of the user that generated the event. The key-value pairs include:

operation The name of the event

sessid The identifier for the event

batchsig The batch identifier for the event

hextime The number of seconds since January 1, 1970

ctime The time that the event occurred as a human-readable string

date The date the event occurred in YYYYMMDD format

hhmmss The time the event occurred in HH:MM:SS format

remote_addr The address of the person who made the request

logfilename The log file for the program that ran **pxnull**

adminid The target administrator for the external system, if configured for a IT Service Management (Ticket) system target

adminpwd The password for the external system administrator

sysid The system administrator for the external system, if supported by the target type

syspwd The system administrator password, if supported by the target type

targetaddress The target address for the external system, if configured for a ticket system target.

sessdat Contains key-value pairs in the session of the user that generated the event. The contents will vary greatly from one event to another, but typically includes:

adminid The ID of the help desk user who made the request

userid The ID of the person who made the request

username The full name of the person who made the request

sesslog This variable is an array of associative arrays. Each associative array in SESSLOG corresponds to a single entry in the session log for the current login.

Each associative array includes the following key-value pairs:

tstart Hexadecimal representation of the start time, in the Unix epoch, counted in seconds

tend End time (as above)

reqby The user that issued the operation

user The user affected by the operation

hostid The ID of the target system affected by the operation

host The description of the target system affected by the operation

longid This is the account ID of the user on the target specified in the hostid value

oper The type of operation

result The result code or the error message

Note: Key names are case sensitive and must be lowercase

10.2.1 Additional variables

There are three additional associative arrays available. These arrays are passed into interface programs via event settings. Each array includes key-value pairs for corresponding attributes.

reqattrs for REQUESTER_ATTRIBUTES trap file sections.

This array cannot be used with any module login success or failure event settings.

recattrs for RECIPIENT_ATTRIBUTES trap file sections.

This array can be used with any event setting, except for USER_CREATE_FAILURE.

rattrs for REQUEST_ATTRIBUTES trap file sections.

This array can be used with any workflow-related event settings except for USER_DELETE_FAILURE, USER_ENABLE_FAILURE, USER_CREATE_FAILURE, and USER_DISABLE_FAILURE.

10.3 Initialization function

To set up an interface with a IT Service Management (Ticket) system, the script must call the initializeInterface() PSLANG function for each operation. This function takes an associative array that must include key-values for the target server and administrative credentials. Some target systems require more; for example, Remedy Action Request System requires the form name and language.

See the appropriate chapter for required key-values for each type of external program.

Once initialized, actions related to the operation can be performed. This includes searching, assigning, and updating records or tickets.

Use **`pxnull`** when you need to integrate with systems or business processes that cannot be supported through any of our other interfaces. Note that the search, assign, and append functions are not available to **`pxnull`** due to its generic nature.

10.4 Using ticket target systems

As of *Connector Pack* version 1.1 and Bravura Security Fabric 7.0, you can can specify interface targets in event settings. When you do this, the administrator credentials for the interface system are sent to the interface binary for use in the PSLANG script. This allows you to support multiple servers/systems without having to place the administrator credentials into the interface's configuration file; for example:

```
$initlist["userid"] = $general["adminid"];
$initlist["password"] = $general["adminpwd"] ;
$initlist["server"] = $general["targetaddress"] ;
```

If you use a target ID as one of your comma-separated values in an event setting, *Bravura Security Fabric* includes appropriate administrative credentials in the data it sends to the interface executable. The interface will then provide this additional information to the PSLANG script via the $general associative array.

If a IT Service Management (Ticket) system target supports additional system credentials, the session information can also include the system administrator ID (`sysid`) and password (`syspwd`).

Target types are listed in IT Service Management (Ticket) Integration Guide (**`itsm-ticket-systems.pdf`**). See the appropriate chapter for your external system type to learn how to target the system.

PSLang Scripts for agtdos, agttelnet, and agtssh

11

The following *flexible agents* require PSLANG scripts to integrate with target systems:

Agent	Target type
agtdos	Win32 Console Script
agttelnet	Telnet target
agtssh	SSHD Host target

Read this chapter to learn more about the functions and global variables used by **agtdos**, **agttelnet**, and **agtssh**.

11.1 Script files

The agent reads a script file with one or more functions defined in it. The agent refers to functions that you implement in the script by name and executes the appropriate function for each agent operation. You can write custom scripts which must be added in the \<*instance*>\script\ directory. Official, shipped scripts are included in the agent directory, and use a scripted platform definition file (.con) to call one of the binary agents. The official scripts are designed to work "out of the box", but can be modified to suit your requirements. See **??** for information about the scripted platform definition file format.

The following sample PSLANG scripts are available in the samples directory:

CAUTION: Any sample script should be modified to fit your environment and prevent security exploits.

- **Win32 Console Script** scripts:
  - **agtdos-simple.psl**
  - **agtdos-complete.psl**

    Note: These scripts can be used with targets if required to maintain any list of users or groups.
- **Telnet target** scripts:
  - **agttelnet.psl** for Telnet targets that users access with a terminal emulator

  - **agttelnet-racf.psl** for z/OS targets that users access with a terminal emulator

- **SSHD Host target** scripts:

  - **agtssh-simple.psl** is a sample file that is intended to be used as a starting point to build a customized connector - see the PSLANG scripts for the Aix, Hpux, Linux and Solaris connectors.

11.2 Writing a script

Write scripts for these connectors using PSLANG. Scripts are broken down into functions. Each function must include a name, function parameters (if applicable), and a sequence of statements to execute:

```
function <FunctionName>( <p1>, <p2>, ... )
  {
  <statement>
  <statement>
  ...
  }
```

The scripts can contain a mix of user-defined and built-in functions. The user-defined functions are outlined in User-defined functions. Built-in functions, which are used to pass information to the agent, are outlined in Built-in Functions.

WARNING!: Remember to remove all debugging statements and to review all statements that will create logs before using the script in a production environment. Ensure that no sensitive information is being captured in the logs.

11.2.1 User-defined functions

In order for **agtdos**, **agttelnet**, or **agtssh** to perform an agent operation, you must implement the corresponding function in your script. Write these functions to provide interaction details between Bravura Security Fabric and the target.

The agents use the following functions:

list(const $wantGroups, const $wantAttributes) Lists accounts on the target system, and the attributes for each account.

The parameters passed to the function are as follows:

- `$wantGroups` – determines if this function should retrieve group membership.
  If the value of `$wantGroups` is a non-zero value, write your function to retrieve the group(s) to which each account belongs while listing accounts.

- `$wantAttributes` – determines if this function should retrieve attributes for each account.
  If the value of `$wantAttributes` is a non-zero value, write your function to list the attributes for each account.

If both `$wantGroups` and `$wantAttributes` are `0`, write your function to list accounts on the target system only.

See `listmembermethod` 11.2.1 for more information about listing group membership.

listgroups(const $wantMembers) Lists groups.

The parameters passed to the function are as follows:

- `$wantMembers` – determines if this function should retrieve group membership.
  If the value of `$wantMembers` is non-zero, write your function to retrieve the account(s) that belong to each group while listing groups.
  If this value of `$wantMembers` is `0`, write your function to list groups only.

See `listmembermethod` 11.2.1 for more information about listing group membership.

listresource(const $resourceType) Lists resources for *Bravura Privilege*'s infrastructure auto discovery feature. The parameters passed to the function are as follows:

- `$resourceType` defines which type of resource to list.
  Depending on target system configuration, listresource() will be called multiple times each with a different resourceType value. The values that resourceType can take on are:
  - `ls_compwkstn` – list workstation computer objects.
  - `ls_compsvr` – list server computer objects.
  - `ls_admmember` – list members of administrative groups.
  - `ls_taskacct` – list scheduled task accounts.
  - `ls_comacct` – list COM object accounts.
  - `ls_iisacct` – list IIS virtual directory accounts.
  - `ls_scmacct` – list Service Control Manager accounts.

movecontext(const $info) Moves an account to a new context or location on a context-sensitive target. Supported in *Bravura Identity* and *Bravura Privilege*.

isenabled(const $info) Checks if an account is enabled. Supported in *Bravura Pass* and *Bravura Privilege*.

islocked(const $info) Checks if an account is locked. Supported in *Bravura Pass* and *Bravura Privilege*.

ispwexpired(const $info) Checks if an account's password is expired. Supported in *Bravura Pass* and *Bravura Privilege*.

isacctexpired(const $info) Checks if an account is expired. Supported in *Bravura Pass* and *Bravura Privilege*.

disable(const $info) Disables an account. Supported in *Bravura Identity* and *Bravura Privilege*.

Most systems differentiate between disabled and intruder-locked accounts.

enable(const $info) Enables an account. Supported in *Bravura Identity* and *Bravura Privilege*.

lock(const $info) Locks an account (sets the intruder lockout). Supported in *Bravura Pass* and *Bravura Privilege*.

Most system differentiate between intruder-locked and disabled accounts.

unlock(const $info) Unlocks an account (clears the intruder lockout). Supported in *Bravura Pass* and *Bravura Privilege*.

expirepw(const $info) Expires an account's password. Supported in *Bravura Pass* and *Bravura Privilege*.

unexpirepw(const $info) Unexpires an account's password. Supported in *Bravura Pass* and *Bravura Privilege*.

expireacct(const $info) Expires an account. Supported in *Bravura Pass* and *Bravura Privilege*.

unexpireacct(const $info) Unexpires an account. Supported in *Bravura Pass* and *Bravura Privilege*.

create(const $info) Creates a new account on the target system. This operation creates the account (possibly using a template for some attribute values), then sets other attribute values – including the password for the new account. Supported in *Bravura Identity* and *Bravura Privilege*.

This function should make use of the target system administrator credentials.

delete(const $info) Deletes an existing account on the target system. The typical behavior is to first ensure that the account being deleted exists. Supported in *Bravura Identity* and *Bravura Privilege*.

You may also wish to include instructions to delete any files or objects associated with the account.

verify(const $info) Checks if a given password is the correct, current password for an account. If the application supports the concept of intruder lockout and the verification fails, the intruder lockout counter is incremented. Supported in *Bravura Identity* and *Bravura Privilege*.

admin_verify(const $info) Checks if a given password is the correct, current password for an account without triggering an intruder lockout if the password is not correct. Supported in *Bravura Pass* and *Bravura Privilege*.

verifyreset(const $info) Verifies if the account's password matches the new password, and if the verification fails, administratively sets it to the new password. If the verification succeeds, then the reset is not necessary, and the operation returns success. Supported in *Bravura Pass* and *Bravura Privilege*.

reset(const $info) Administratively resets an account's password to a new value. If the application supports the concept of intruder lockout, then the intruder lockout counter is cleared and the account unlocked. If the application supports the concept of password expiry, then the expiry date is set according to the expiry policy of the application. Disabled accounts will remain disabled. Supported in *Bravura Pass* and *Bravura Privilege*.

change(const $info) Changes the password for an account, from a known current value to a desired new value. If the application supports the concept of intruder lockout, then the intruder lockout counter is cleared and the account unlocked. If the application supports the concept of password expiry, then the expiry date is set according to the expiry policy of the application. Supported in *Bravura Pass* and *Bravura Privilege*.

resetexpirepw(const $info) Administratively resets an account's password to a new value and expires the account's new password, so that the user is forced to change his password the next time he logs in. Supported in *Bravura Pass* and *Bravura Privilege*.

groupuseradd(const $info) Adds an account to a group. Supported in *Bravura Identity* and *Bravura Privilege*.

groupuserdelete(const $info) Removes an account from a group. Supported in *Bravura Identity* and *Bravura Privilege*.

groupgroupadd(const $info) Adds a group to a group. Supported in *Bravura Identity* and *Bravura Privilege*.

groupgroupdelete(const $info) Removes a group from a group. Supported in *Bravura Identity* and *Bravura Privilege*.

update(const $info) Updates attributes for an existing account. Supported in *Bravura Identity* and *Bravura Privilege*.

userattributes(const $info) Lists attributes for a specified account. Supported in *Bravura Identity* and *Bravura Privilege*.

All functions are optional. To indicate success your function must return `0`. To indicate failure your function must return `1`.

> **WARNING!:** Functions which reset or verify passwords in scripts used by *Bravura Privilege* for password randomization must always report the result of the reset or verification accurately. Inaccurate status information may result in checked out passwords not working; additionally, if the credential used to manage the target system is being randomized, *Bravura Privilege* may invalidate its own managed system credential when an erroneous status is returned. Do not assume scripted password changes are always successful; always check return codes.

If your target does not support a particular operation (for example, if there is no intruder lockout mechanism), then omit the corresponding function from the script. If the agent cannot find a particular function in the script, the agent returns a message saying "Function [<*operation*>] not found in script".

The following additional functions (optional) do not represent actual agent operations, however they are also called directly by the agent:

connecttarget(const $cinfo) called before the first operation is performed.

This function must return `0` on success, or `1` on failure.

disconnecttarget(const $cinfo) called after the last operation is performed.

This function must return `0` on success, or `1` on failure.

listmembermethod() used to determine the supported method for listing group membership on the target system.

If both methods are supported, select the method that is the most efficient. This function must return:

- `0` - if the script cannot list group membership
- `1` - if the script list accounts in a group while listing groups
- `2` - if the script lists groups while listing accounts

The calling program uses the result of `listmembermethod`, along with your target-configuration settings, to determine the value of `$wantGroups` and `$wantMembers` (11.2.1).

For example, if the calling program is executed with the option to list group membership, and the result of `listmembermethod` is `1`, the calling program executes the function:

```
listgroups(1)
```


addressattrs () used to extend the address wizard by adding additional address line elements. It is only possible to add new address line elements, *not* remove or change existing ones. This is useful when creating a scripted target system with a scripted platform definition file.

- If not defined, the address wizard behaves as normal.
- From within this function the `addAddressElement` callback function adds a single address line element. The callback function can be called multiple times and takes the following form:
  ```
  addAddressElement(name, type, defaultValue, isRequired,
  restrictedValuesList, description, advanced);
  ```
  Where:
  - name: is the string name of the of the address line component.
  - type: is one of ($AddressTypeInt $AddressTypeString $AddressTypeBoolean $AddressTypeKvgroup $AddressTypeRestricted $AddressTypeScript $AddressTypeDir $AddressTypeFile).
  - defaultValue: string default value or empty if no default value.
  - isRequired: boolean, where 1=required and 0=optional.
  - restrictedValuesList: array of display/actual values for each restricted value.
  - description: Textual description of the element.
  - advanced: boolean, 1=display attribute in advanced section and 0=default

  An example is included in the sample **`agttelnet-racf.psl`**.

The following function is available to **`agtssh`**:

credentialoverride(inout $cinfo) called before all other functions and used to override the adminid and adminpw credentials on the connection to the SSH target.

- The function is optional, and is not called if it is not defined.
- The function is called before all other functions.
- The adminid, adminpw, host, and address elements are readable in the function.
- The adminid and adminpw elements are settable within the credentialoverride function to other values. The return value for the function does not affect the commitment of the override values - **`agtssh`** always uses the returned values for adminid and adminpw for the connection.
- The host, address, sysid, and syspw elements do not commit overridden values.
- The adminid and adminpw elements are accessible via a transitional variable to other functions.
- The variable `$COMMON_SHELL` can be set in the function to specify a custom shell.

11.2.1.1 Input parameters ($cinfo and $info)

The connecttarget() and disconnecttarget() functions each take one parameter, an associative array ($cinfo). Each element is a string. Elements of the array are:

address the target address.

adminid the target system administrator's ID.

adminpw the target system administrator's password.

sysid the target system administrator's ID if the **Is this an additional system password?** checkbox is selected.

syspw the target system administrator's password if the **Is this an additional system password?** checkbox is selected.

Most of the functions that relate to Bravura Security Fabric operations (for example, create(), enable()reset(), change()) also take one parameter, an associative array ($info). Each element is a string unless otherwise indicated. Elements of the array are:

userid the profile ID of the user.

shortid the short ID of the user.

fullname the full name of the user.

acctid the ID for the account that the user has on this system.

newpw the new password (for password reset and create operations).

oldpw the old password (for verify operations).

modeluid the template ID (for create operations).

groupid the group ID (for group* operations).

groupname the group description (for group* operations).

attributes a KVGroup containing information about attributes to set (for create and update operations)

The general format of the KVGroup is as follows:

```
"attributes" "" = {
  "DEFAULT-ACTION" = "COPY"
  "attribute" "givenName" = {
    "SEQUENCE" = "0"
    "GROUP"    = "0"
    "ACTION"   = "VALUE"
    "VALUE"    = "John"
   }
  "attribute" "sn" = {
    "SEQUENCE" = "0"
    "GROUP"    = "0"
    "ACTION"   = "VALUE"
    "VALUE"    = "Doe"
   }
 }
```

ConnectionCredential a KVGroup containing information about target administrator credentials and system credentials (for reset operations).

The general format of the KVGroup is as follows:

```
"ConnectionCredential" "" = {
  "AdminCredential" "" = {
    "admin1" = "{AES}@KK|dE]=GqX=zqJyTIgqL]XCyECV?AsAJ[O?vBJGpAaEGgfK"
    "admin2" = "{AES}e[LkiHEv?=iAGTD|ZA^cKCYJCb?[tByFDp_=e]EhWDkgHzmD"
```



```
  }
 "SysCredential" "" = {
   "sysadm1" = "{AES}fN=>rLOB?^gDICJfc@fYLB@@COGlv=H@IFc@EW?XBJTwFEXB"
  }
}
```

11.2.1.2 Return codes

The return codes for user-defined functions (with the exception of listmembermethod()) for **agtdos**, **agttelnet**, and **agtssh** are as follows:

Table 11.1: Agent return codes

Code	Value	Description
ACSuccess	0	The operation was successful.
ACUnknownError	1	The operation failed with an unknown error.
ACOperationNotSupported	2	The operation is not supported by the target system.
ACNotConnected	3	Can be used in the connecttarget() function if a connection could not be established.
ACAlreadyConnected	4	Can be used in the connecttarget() function if an attempt is made to connect to a target that already has a connection.
ACInvalidServer	5	Can be used in the connecttarget() function if the target address is invalid.
ACObjectAlreadyExists	6	The object to create (for example, user or group) already exists.
ACInvalidUser	7	The user account is invalid or does not exist.
ACInvalidModelUser	8	The template account is invalid or does not exist.
ACUserNotInGroup	9	The user is not in the specified group.
ACTimeout	10	The operation timed out.
ACAccessDenied	11	The target system administrator could not logon due to a bad ID/password pair, account restrictions, and so on.
ACInvalidGroup	12	The group is invalid or does not exist.
ACUserAlreadyGroupMember	13	The user is already a member of the specified group.
ACVerifyFailed	14	The password could not be verified. Use this code ONLY when the specified password is not correct.
ACInitFailed	15	The operation failed to initialize.
ACLockFailed	16	The operation failed to get a lock on a file while reading or writing.
ACScriptError	17	The operation encountered an error in the script.
ACNotLicensed	18	The operation is not licensed.
ACLogFileFailure	19	Failed to write to the log file.
ACImplementerFailure	20	The implementor failed to perform the operation.
ACAdminLocked	21	The target system administrator account is locked.
ACInvalidPasswd	22	The password is invalid.

. . . continued on next page

Table 11.1: Agent return codes (Continued)

Code	Value	Description
ACDelayLoadError	23	The operation encountered an error loading a dll or function point.
ACReadOnlyResource	24	The operation could not write to a resource because it was read only.
ACPluginAborted	25	The plugin failed.
ACPasswordLocked	26	The password is locked.
ACPasswordExpired	27	The password has expired.
ACAccountExpired	28	The account has expired.
ACAccountDisabled	29	The account is disabled.
ACOperationRolledback	30	The operation was rolled back.
ACOperationAborted	31	The operation was aborted.
ACOperationRollbackFailed	32	The rollback operation has failed.
ACEndOfFile	33	The operation has reached the end of file.
ACGroupAlreadyGroupMember	34	The group is already a group member.
ACGroupNotInGroup	35	Group is not a member of group.
ACNull	0xffff	A null code.

11.2.2 Built-in Functions

Apart from the standard PSLANG built-in functions, there are several built-in agent functions available to the agents.

You call the following built-ins from within functions representing agent operations (p438) in order to communicate the results of the operation back to the agent:

agentError reports an error message.

agentWarning reports a warning message.

agentInfo reports an information message.

agentListUser is called once for each user during the list operation.

agentListGroup (or agentListGroupSID) is called once for each group during the list operation.

agentListAttribute is called once for each attribute during the list attribute, create, and update operations.

agentListResourceComputer is called once for each computer object during the list resource operation.

agentListResourceAccount is called once for each account object during the list resource operation.

agentIsEnabled tells the agent whether or not the given user is enabled.

agentIsLocked tells the agent whether or not the given user is locked out.

agentIsPassExpired tells the agent whether or not the given user's password has expired.

agentIsAcctExpired tells the agent whether or not the given user's account has expired.

agentLongId provides the long ID of the user during create, update, rename, or move context operations.

agentShortID provides the short ID of the user during create, update, rename, or move context operations.

agentGroups provides the list of groups to which a user belongs.

agentOutput can return IP information on reset using agentOutput. The keys are ip-address and dns-hostname. This information is used by Bravura Privilege to load current information when the reset occurs. This is included in the **`agtssh-simple.psl`** sample. By default it is disabled. It can be enabled with the $emit_info at the top of the script by setting it to non-zero.

See Connector Function Reference for more information about these built-in functions.

11.2.3 Global variables

In any agent script, input passed to the agent is stored in the `$_inVars` "hidden" global variable. You can declare and define additional global variables for use throughout the script.

Following is an example of the contents of the `$_inVars` KVGroup when performing a list operation:

```
"" "" = {
  "address" = "agtdos.psl"
  "adminid" = "null"
  "adminpw" = "<encrypted password value>"
  "hostid" = "DOS"
  "listdbfilename" = "C:\\Program Files\\Hitachi ID\\IDM Suite\\default\\psconfig\\
   AGTDOS.db"
  "operation" = "listobj"
  "platformname" = "Win32 Console Script"
  "timeout" = "-1"
 "listattributes" "" = {
  }
}
```

11.2.4 Built-in variables

The following built-in variables are available for use with PSLang scripts:

$KeyAccounts

Value: "accounts"

Description: List of accounts.

$KeyAcctID

Value: "acctid"

Description: Name of the target account to operate on.

$KeyAction

Value: "ACTION"

Description: Action to do to the attribute.

$KeyActionCopy

Value: "COPY"
Description: Action to do to the attribute.

$KeyActionIgnore

Value: "IGNORE"
Description: Action to do to the attribute.

$KeyActionReplace

Value: "REPLACE"
Description: Action to do to the attribute.

$KeyActionValue

Value: "VALUE"
Description: Value of the attribute.

$KeyAddress

Value: "address"
Description: The address line to use to connect to the target.

$KeyAddrComps

Value: "addresscomponents"
Description: Key to the sub-array containing the address elements already split.

$KeyAdminID

Value: "adminid"
Description: Name of the account to connect to target as.

$KeyAdminPW

Value: "adminpw"
Description: Password of the account to connect to the target with.

$KeyAttribute

Value: "attribute"
Description: Single account attribute details.

$KeyAttributes

Value: "attributes"
Description: List of account details.

$KeyAuthKey

Value: "authkey"
Description: SSH Authentication key attribute.

$KeyBoolean

Value: "boolean"
Description: true/false address attribute type

$KeyCAPath

Value: "CApath"
Description: Path to certificate file.

$KeyCAFile

Value: "CAfile"
Description: Certificate file name.

$KeyChildGroupID

Value: "childgroupid"
Description:

$KeyCheckCert

Value: "checkCert"
Description: Should the agent check the SSL certificate.

$KeyCommand

Value: "command"
Description: Command to run during runcommand() operation.

$KeyCommandFile

Value: "commandfilename"
Description: File where the output from runcommand() operation is written to.

$KeyCompression

Value: "compression"
Description: SSH address attribute to compress the data transmitted.

$KeyFalse

Value: "false"
Description: String used for boolean address and account attributes.

$KeyFullName

Value: "fullname"
Description: Description attribute of an account.

$KeyGroup

Value: "group"
Description: Single sub KV of a group on the target.

$KeyGroupID

Value: "groupid"
Description: Identifier for the group.

$KeyGroupName

Value: "groupname"
Description: Name of the group.

$KeyGroups

Value: "groups"
Description: List of groups.

$KeyHostID

Value: "hostid"
Description:

$KeyHostKeysDenyUnmatched

Value: "DenyUnmatch"
Description: Tells the SSH library to connect ONLY if the host key matches an existing one.

$KeyHostKeysUpdate

Value: "AllowUpdate"
Description: Allows the SSH connection to update the host key if one already exists (unsafe).

$KeyHostKeysAppend

Value: "AllowAppend"
Description: Tells the SSH libary to add the host key if it is missing, but refuse if it does not match.

$KeyInteger

Value: "integer"
Description: Integral address attribute type.

$KeyKvgroup

Value: "kvgroup"
Description: Address attribute type.

$KeyManagedGroup

Value: "managedGroup"
Description:

$KeyModelUID

Value: "modeluid"
Description: Account to use as a template when creating a new user.

$KeyNewPW

Value: "newpw"
Description: Password for the new account or the new password when resetting an existing account.

$KeyNoPTY

Value: "nopty"
Description:

$KeyOldPW

Value: "oldpw"
Description: Current password for an account.

$KeyOperation

Value: "operation"
Description: The action that is currently underway.

$KeyPort

Value: "port"
Description: IP port to connect to on the target.

$KeyPosition

Value: "position"
Description:

$KeyReadOnly

Value: "readonly"
Description: Address attribute that is not modifiable by the user.

$KeyResource

Value: "resource"
Description:

$KeyResourceAddress

Value: "resourceaddress"
Description:

$KeyResourceType

Value: "resourcetype"
Description:

$KeyRestricted

Value: "restricted"
Description: Address attribute type of restricted values.

$KeyRestrictedValue

Value: "restrictedvalue"
Description: Key for the chosen restricted address attribute value.

$KeyRestrictedValueText

Value: "restrictedvaluetext"
Description: Display value for a restricted address attribute value.

$KeyRestart

Value: "restart"
Description:

$KeyRevision

Value: "rev"
Description:

$KeyScript

Value: "script"
Description: Address attribute type.

$KeyFile

Value: "file"
Description: Address attribute type.

$KeyDir

Value: "dir"
Description: Address attribute type.

$KeyPath

Value: "path"
Description: Address attribute type.

$KeyServer

Value: "server"
Description: Address attribute containing the target to connect to.

$KeyServerInfo

Value: "serverinfo"
Description: Operation to query the target for version and status information.

$KeySettings

Value: "settings"
Description:

$KeyShortID

Value: "shortid"
Description: User's ID

$KeySSL

Value: "SSL"
Description: Address attribute determining if encrypted telnet should be used.

$KeyString

Value: "string"
Description:

$KeySysID

Value: "sysID"
Description: Username for secondary login to the target (e.g. sudo).

$KeySysPassword

Value: "syspassword"
Description: Password for the secondary login to the target.

$KeyTerm

Value: "terminal"
Description: Address attribute determining the terminal type for the telnet connection.

$KeyTimeout

Value: "timeout"
Description: Address attribute defining how long the agent should wait for connection to complete.

$KeyTrue

Value: "true"

Description: String used for boolean address and account attributes.

$KeyWriteOption

Value: "writeoption"

Description: Address attribute determining the terminal type for the telnet connection.

11.2.5 Live debugging

You can use the termdebug KVGroup to interactively monitor what is sent and received by agent scripts over raw TCP connections, SSL raw connections (not HTTP/HTTPS), telnet and SSH connections. To do this, add a supplemental termdebug KVGroup to the agent input. For example:

```
# KVGROUP-V1.0
"" "" = {
  "address" = "telnet_unix.psl"
  "adminid" = "root"
  "adminpw" = "<encrypted password value>"
  "hostid" = "SSH1"
  "listfilename" = "C:\\listing-test.txt"
  "instance" = "globoco"
  "listresource" "" = {
  }
  "operation" = "list"
  "timeout" = "30"
     "termdebug" "" = {
       ## mandatory configurations
         "address" = "127.0.0.1"
          #String representation of the
          #address of the interface to listen to.
         "port" = "3210"
          #string representation of the TCP port to listen to.
          #agtssh listens to this port on IDM Suite server
         "command" = "c:\\windows\\system32\\telnet.exe localhost 3210"
          #Command line to run with full executable path and parameters
          #Quotes and backslashes need to be escaped with \.
          #See below for more info.
       ## optional configurations
         "interactive" = "1"
           #0=no 1=yes
           #If 1, shows a dialog box with the string to
           # be sent, and waits for OK before it is sent.
         "waitforjointimeout" = "10000"
           #Timeout, in milliseconds, of the agent waiting
           # for a client connection.
         "sendtimeout" = "1000"
           #Timeout, in milliseconds, of sending data back to the
           # client.
       }
}
```

WARNING!: Data seen on the debug port is raw decrypted data

11.2.5.1 Command options

The command value can take Cygwin telnet.exe, or nc.exe, or a tn3270/tn5250 client as required, for example:

```
"command" = "c:\\cywgin\\bin\\bash -c \"/bin/telnet.exe localhost 3210 | tee /tmp/
 log1; sleep 3600\""
```

To prevent the window closing after the debug port closes, include `cmd.exe /k telnet ...` for telnet, or `sleep` for Cygwin telnet.exe or nc.exe.

For non-interactive debugging, the command can be something like:

```
"command" = "c:\\windows\\system32\\cmd.exe /c telnet localhost 3210 >> c:\\debuglog.
 txt 2>&1"
```

11.2.5.2 Interactivity options

When interactivity is turned on (`"interactive" = "1"`) the script waits for one of the following inputs before proceeding:

- ok = next step
- cancel = skip next questions, run all steps automatically without any more prompting

If the original agent script or the target itself has very strict timeouts, it is recommended that you do not use the interactive option, or cancel it before the section with critical timing.

11.2.5.3 Adding the termdebug KVGroup to input

You can add the termdebug KVGroup to the agent input by:

- Capturing agent input using the **`pstee`** program, then manually adding the inner KVGroup.
- Wrapping the actual agent in an **`agtdos`** script as described in Global variables.

  The script should:

  1. Read $_inVars
  2. Replace its address (`"address" = "doswrapper.psl"`) with the telnet or SSH address line, then add the termdebug KVGroup
  3. Call the original agent with this group (converted to string) as input. It could also log the new modified agent input for debugging the PSLANG wrapper script.

  This method is more complicated, but has the advantage of working for every agent call and not replaying operations.

  See Core Library Function Reference for more information about KVGroup functions, and system() calls.

11.3 Sanitizing input variables and recommended ID filters

Since the inputs into the **agttelnet**, **agtssh** and **agtdos** scripted connectors can come from untrusted external sources, Bravura Security recommends putting in safe guards to mitigate the risk of injection attacks. The script writer should take extra care to both filter out potentially nefarious user IDs as well as properly escaping input variables before passing them to shell commands.

Characters such as:

```
;&|<>'"()*#
```

should be looked for in user IDs and filtered out during listing so they don't even make it to the *Bravura Security Fabric* system. In addition (or if these characters are required), the SSH script should ensure that they are properly escaped before passing them into the shell. This will avoid injection attacks which could compromise your system.

ID-Track Function Reference

12

The following functions are available in ID-Track (**idtrack**) scripts.

12.1 ID-Track Functions

12.1.1 getAccountAdded

Platform: all
Details
Return accounts that have been added for the current profile.

```
array getAccountAdded (

)
```

Parameters
None.

Return values
An array of idtrack_Account structures, representing added accounts.

12.1.2 getAccountDeleted

Platform: all
Details
Return accounts that have been deleted for the current profile.

```
array getAccountDeleted (

)
```

Parameters
None.

Return values
An array of idtrack_Account structures, representing deleted accounts.

12.1.3 getAccountAttrAdded

Platform: all
Details
Return account attributes that have been added for the current profile.

```
array getAccountAttrAdded(

)
```

Parameters
None.

Return values
An array of idtrack_AccountAttr structures, representing added account attributes.

12.1.4 getAccountAttrDeleted

Platform: all
Details
Return account attributes that have been deleted for the current profile.

```
array getAccountAttrDeleted(

)
```

Parameters
None.

Return values
An array of idtrack_AccountAttr structures, representing deleted account attributes.

12.1.5 getProfileAttrAdded

Platform: all
Details
Return profile attributes that have been added for the current profile.

```
array getProfileAttrAdded(

)
```

Parameters
None.

Return values
An array of idtrack_ProfileAttr structures, representing added profile attributes.

12.1.6 getProfileAttrDeleted

Platform: all
Details
Return profile attributes that have been deleted for the current profile.

```
array getProfileAttrDeleted(

)
```

Parameters
None.

Return values
An array of idtrack_ProfileAttr structures, representing deleted profile attributes.

12.1.7 getGroupMemberAdded

Platform: all
Details
Return group memberships that have been added for the current profile.

```
array getGroupMemberAdded(

)
```

Parameters
None.

Return values
An array of idtrack_GroupMember structures, representing added group memberships.

12.1.8 getGroupMemberDeleted

Platform: all
Details
Return group memberships that have been deleted for the current profile.

```
array getGroupMemberDeleted(

)
```

Parameters
None.

Return values
An array of idtrack_GroupMember structures, representing deleted group memberships.

12.1.9 getAccountAssocChanged

Platform: all
Details
Return account associations that have changed for the current profile.

```
array getAccountAssocChanged (

)
```

Parameters
None.

Return values
An array of idtrack_AccountAssoc structures, representing association changes.

12.1.10 getItemStatus

Platform: all
Details
Return the status and request ID assocated with the current change item. Set by a previous call to setItemStatus.

```
void getItemStatus (
        int status ,
        string requestid
)
```

Parameters
status
[out] Status value.

requestid
[out] Request ID.

Return values
None.

12.1.11 setItemStatus

Platform: all
Details
Set status and request ID assocated with the current change item.

```
void setItemStatus (
        int status ,
        string requestid
```


)

Parameters
status
[const] Status value.

requestid
[const] Request ID.

Return values
None.

12.1.12 autosyncAttrDisable

Platform: all
Details
Disable automatic synchronization for a profile attribute.

```
void autosyncAttrDisable (
        string attrkey
)
```

Parameters
attrkey
[in] ID of profile attribute.

Return values
None.

12.1.13 getNestedGroupAdded

Platform: all
Details
Return child group memberships that have been added.

```
array getNestedGroupAdded (

)
```

Parameters
None.

Return values
An array of idtrack_NestedGroup structures, representing added nested group memberships.

12.1.14 getNestedGroupDeleted

Platform: all
Details
Return child group memberships that have been deleted.

```
array getNestedGroupDeleted(

)
```

Parameters
None.

Return values
An array of idtrack_NestedGroup structures, representing deleted nested group memberships.

Bravura Privilege Function Reference

13

The following functions are available in Bravura Privilege scripts.

13.1 *Bravura Privilege* Account Functions

13.1.1 manageService

Platform: win32
Details
Determine whether an account manages any services. This function is only available during import rule evaluation for managed accounts.

```
int manageService (
        string serviceName ,
        string targetid
)
```

Parameters
serviceName
[in] Managed service name. An empty value means any service.

targetid
[in] A target system ID on which the service is managed. An empty value means any target system.

Return values
1 if the account manages a service, 0 otherwise.

13.1.2 manageTask

Platform: win32
Details
Determine whether an account manages any tasks. This function is only available during import rule evaluation for managed accounts.

```
int manageTask (
        string taskName ,
        string targetid
```

)

Parameters
taskName
[in] Managed task name. An empty value means any task.

targetid
[in] A target system ID on which the task is managed. An empty value means any target system.

Return values
1 if the account manages a task, 0 otherwise.

13.1.3 manageWebSite

Platform: win32
Details
Determine whether an account manages any web sites. This function is only available during import rule evaluation for managed accounts.

```
int manageWebSite(
        string webSiteName,
        string targetid
)
```

Parameters
webSiteName
[in] Managed web site name. An empty value means any web site.

targetid
[in] A target system ID on which the web site is managed. Empty means any target system.

Return values
1 if the account manages a web site, 0 otherwise.

13.1.4 manageCOM

Platform: win32
Details
Determine whether an account manages any COM. This function is only available during import rule evaluation for managed accounts.

```
int manageCOM(
        string COMGuid,
        string targetid
)
```

Parameters
COMGuid
[in] Managed COM guid. An empty value means any COM object.

targetid
[in] A target system ID on which the COM object is managed. An empty value means any target system.

Return values
1 if the account manages a COM object, 0 otherwise.

13.1.5 manageCustom

Platform: win32
Details
Determine whether an account manages any Custom object. This function is only available during import rule evaluation for managed accounts.

```
int manageCustom(
        string Customid,
        string targetid
)
```

Parameters
Customid
[in] Managed Custom id. An empty value means any Custom object.

targetid
[in] A target system ID on which the Custom object is managed. An empty value means any target system.

Return values
1 if the account manages a Custom object, 0 otherwise.

13.1.6 manageResource

Platform: win32
Details
Determine whether an account manages any resource such as a service, task, web site, or COM. This function is only available during import rule evaluation for managed accounts.

```
int manageResource(
        string targetid
)
```

Parameters
targetid
[in] A target system ID on which the resource is managed. An empty value means any target system.

Return values
1 if the account manages a resource, 0 otherwise.

13.1.7 memberOfByName

Platform: win32
Details
Determine whether an account is a member of a group. This function is only available during import rule evaluation for managed accounts.

```
int memberOfByName(
        string targetid,
        string groupname
)
```

Parameters
targetid
[in] A target system ID on which the group is located.

groupname
[in] The group name.

Return values
1 if the account is a member of the group, 0 otherwise.

13.1.8 memberOfBySID

Platform: win32
Details
Determine whether an account is a member of a group. This function is only available during import rule evaluation for managed accounts.

```
int memberOfBySID(
        string targetid,
        string groupsid
)
```

Parameters
targetid
[in] A target system ID on which the group is located.

groupsid
[in] The group SID.

Return values
1 if the account is a member of the group, 0 otherwise.

13.1.9 memberOfByNameNested

Platform: win32
Details

Determine whether an account is a member of a group, either directly or as a member of a subgroup. This function is only available during import rule evaluation for managed accounts.

```
int memberOfByNameNested (
        string targetid ,
        string groupname
)
```

Parameters
targetid
[in] A target system ID on which the group is located.

groupname
[in] The group name.

Return values
1 if the account is a member of the group, 0 otherwise.

13.1.10 memberOfBySIDNested

Platform: win32
Details
Determine whether an account is a member of a group, either directly or as a member of a subgroup. This function is only available during import rule evaluation for managed accounts.

```
int memberOfBySIDNested (
        string targetid ,
        string groupsid
)
```

Parameters
targetid
[in] A target system ID on which the group is located.

groupsid
[in] The group SID.

Return values
1 if the account is a member of the group, 0 otherwise.

13.1.11 existOnTarget

Platform: win32
Details
Determine whether an account exists on the target system. This function is only available during import rule evaluation for managed accounts.

```
int existOnTarget (
        string targetid
```


)

Parameters
targetid
[in] A target system ID on which the account is located.

Return values
1 if the testing account is located on the target ID, 0 otherwise.

13.1.12 accountHostMatch

Platform: win32
Details
Check whether the account host matches the attribute value.

```
int accountHostMatch(
        string attrkey,
        string attrval
)
```

Parameters
attrkey
[in] A discovered computer attribute key.

attrval
[in] A discovered computer attribute value.

Return values
1 if the account host matches the attribute key and value, 0 otherwise.

13.2 *Bravura Privilege* Computer Functions

13.2.1 compAttrExists

Platform: win32
Details
Check whether the computer attribute exists or not.

```
int compAttrExists(
        string attrkey,
        string attrval,
        int mvattr
)
```

Parameters
attrkey
[in] A discovered computer attribute key.

attrval
[in] A discovered computer attribute value.

mvattr
[in] 1 the attrkey is a multi-value attribute.

Return values
1 if the specified attribute exists.

Telephone Password Manager Function Reference

14

The following functions are available in Phone Password Manager scripts.

14.1 Dialogic Functions

The following functions can be used to interact with the Dialogic voice board.

14.1.1 SetHookOff

Platform: win32
Details
Pick up current incoming call. (Only inbound mode can call this function explicitly).

```
int SetHookOff (

)
```

Parameters
None.

Return values
0 means success, -1 means failure.

14.1.2 MakeCall

Platform: win32
Details
Make an outbound call

```
int MakeCall (
        string phoneNo ,
        int timeout ,
        string errbuf
```

)

Parameters
phoneNo
[in] The phone number or address to transfer destination. The valid destination address syntax for H323 protocol are:

- TA: – IP transport address
- TEL: – e164 telephone number
- NAME: – H.323 ID
- URL: – Universal Resource Locator
- EMAIL: – email address

The SIP destination address format is:

- user@host; param=value

timeout
[in] The timeout (seconds).

errbuf
[out] Error messages will be returned in this buffer.

Return values
-1 means failure, otherwise success.

14.1.3 MakeCallEx

Platform: win32
Details
Make an outbound call with Call Progress Analysis.

```
int MakeCallEx (
        string phoneNo ,
        int timeout ,
        int offhook ,
        string errbuf
)
```

Parameters
phoneNo
[in] The phone number or address to transfer destination. The valid destination address syntax for H323 protocol are:

- TA: – IP transport address

- TEL: – e164 telephone number
- NAME: – H.323 ID
- URL: – Universal Resource Locator
- EMAIL: – email address

The SIP destination address format is:

- user@host; param=value

timeout
[in] The timeout (seconds).

offhook
[in] Off hook the channel before making the call.

errbuf
[out] Error messages will be returned in this buffer.

Return values
-1 means failure, otherwise means call analysis result.

- 0: – General connect success
- 1: – Cadence Break
- 2: – Loop Current Drop
- 3: – Positive Voice Detect
- 4: – Positive Answering Machine Detect
- 5: – Connect to PBX

14.1.4 ClearDigits

Platform: win32
Details
Discards any digits in the current Media Resource's digit buffer.

```
int ClearDigits(

)
```

Parameters
None.

Return values
0 means success, -1 means failure.

14.1.5 GetDigits

Platform: win32
Details
Waits for and retrieves touch-tone digits on the current Media Resource.

```
int GetDigits (
        int maxDigits ,
        int maxSeconds ,
        int maxSilence ,
        string stopTones ,
        string digits
)
```

Parameters
maxDigits
[in] Maximum number of digits to wait for.

maxSeconds
[in] Maximum seconds to wait.

maxSilence
[in] Maximum seconds to wait between digits.

stopTones
[in] Touch tone digits that will end input immediately.

digits
[out] The digits stored in the current Media Resource's digit buffer.

Return values
0 means success or timeout, 1 means a stop tone was received, -3 means channel was disconnected, -1 means failure returned from the underlying phone module.

14.1.6 TransferCall

Platform: win32
Details
Makes an outgoing call.

```
int TransferCall (
        string phoneNo ,
        string errbuf
)
```

Parameters
phoneNo
[in] The phone number or address to transfer destination. The valid destination address syntax for H323 protocol are:

- TA: – IP transport address
- TEL: – e164 telephone number
- NAME: – H.323 ID
- URL: – Universal Resource Locator
- EMAIL: – email address

The SIP destination address format is:

- user@host; param=value

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -3 means hangup, -1 means failure.0 on success, non-zero on failure

14.1.7 PlayFile

Platform: win32
Details
Plays a sound file on the current Media Resource.

```
int PlayFile (
        string fileName ,
        int soundFileType ,
        string stopTones
)
```

Parameters
fileName
[in] The name and path of the file to play.

soundFileType
[in] Sound file type. Supported Sound File Types indicates all supported file types.

stopTones
[in] Gives a list of tones that will interrupt the play. Allowed values: a string of zero or more digits, including the star (*) and pound sign (#).

Return values
0 means success, 1 means playing was interrupted by key press, -3 means hangup, -1 means failure.0 or 1 on success (1 indicates that the playing was interrupted by a digit press) negative number on failure.

14.1.8 PlayFileEx

Platform: win32
Details
Plays a sound file for the specified language on the current Media Resource.

```
int PlayFileEx (
        string language ,
        string fileName ,
        string stopTones
)
```

Parameters
language
[in] Language type, for example, en-us.

fileName
[in] The name of the file to play.

stopTones
[in] Gives a list of tones that will interrupt the play. Allowed values: a string of zero or more digits, including the star (*) and pound sign (#).

Return values
0 means success, 1 means playing was interrupted by key press, -3 means hangup, -1 means failure.0 or 1 on success (1 indicates that the playing was interrupted by a digit press) negative number on failure.

14.1.9 SpellText

Platform: win32
Details
Spell the given text.

```
int SpellText (
        string language ,
        string text ,
        string stopTones ,
        int casesen
)
```

Parameters
language
[in] Language Type, for example en-us.

text
[in] The text to be spelled.

stopTones
[in] Gives a list of tones that will interrupt the play. Allowed values: a string of zero or more digits, including the star (*) and pound sign (#).

casesen
[in] 1 = case-sensitive, 0 = case-insensitive.

Return values
0 means success, 1 means playing was interrupted by key press, -3 means hangup, -1 means failure.

14.1.10 SpellTextEx

Platform: win32
Details
Spell the given text.

```
int SpellTextEx (
        string language ,
        string text ,
        string stopTones ,
        int casesen ,
        int callsign
)
```

Parameters
language
[in] Language Type, for example en-us.

text
[in] The text to be spelled.

stopTones
[in] Gives a list of tones that will interrupt the play. Allowed values: a string of zero or more digits, including the star (*) and pound sign (#).

casesen
[in] 1 = case-sensitive, 0 = case-insensitive.

callsign
[in] 1 = with callsign, 0 = without callsign.

Return values
0 means success, 1 means playing was interrupted by key press, -3 means hangup, -1 means failure.

14.1.11 ReadNumber

Platform: win32
Details
Pronounce the given number.


```
int ReadNumber (
        string language ,
        int number ,
        string stopTones
)
```

Parameters
language
[in] Language Type, for example en-us.

number
[in] The number to be read.

stopTones
[in] Gives a list of tones that will interrupt the play. Allowed values: a string of zero or more digits, including the star (*) and pound sign (#).

Return values
0 means success, 1 means playing was interrupted by key press, -3 means hangup, -1 means failure.0 or 1 on success (1 indicates that the playing was interrupted by a digit press) negative number on failure or hangup

14.1.12 PressNumber

Platform: win32
Details
Prompt the user to press the given number.

```
int PressNumber (
        string language ,
        int number ,
        string stopTones
)
```

Parameters
language
[in] Language Type, for example en-us.

number
[in] The number to be read.

stopTones
[in] Gives a list of tones that will interrupt the play. Allowed values: a string of zero or more digits, including the star (*) and pound sign (#).

Return values
0 means success, 1 means playing was interrupted by key press, -3 means hangup, -1 means failure.0 or 1 on success (1 indicates that the playing was interrupted by a digit press) negative number on failure or hangup.

14.1.13 RecordFile

Platform: win32
Details
Records sound from the current Media Resource to a sound file.

```
int RecordFile(
        string fileName,
        int soundFileType,
        int maxSeconds,
        int maxSilences,
        string stopTones,
        int append
)
```

Parameters
fileName
[in] The name and path of the file to record.

soundFileType
[in] Sound file type. Supported Sound File Types indicates all supported file types, -1 means using RecFileType defined in configuration file.

maxSeconds
[in] Maximum time to allow recording, in seconds. The default value is 180.

maxSilences
[in] Maximum period of silence to allow in recording, measured in seconds. Often a silence time-out is used to end the recording of a new voice mail message, for example. (If this parameter is set to 0, the method ignores silence.) The default value is 0.

stopTones
[in] Gives a list of tones that will interrupt the recording. Allowed values: a string of zero or more digits, including the star (*) and pound sign (#).

append
[in] Determines whether the recording is appended to an existing file of the same name. By default, `append` is 0.

Return values
0 means success, 1 means playing was interrupted by key press, -3 means hangup, -1 means failure.

14.1.14 Hangup

Platform: win32
Details
Terminate current running script and release line.

```
void Hangup(
        string message
```


)

Parameters
message
[in] The message to log.

Return values
None.

14.1.15 OnHangup

Platform: win32
Details
Specify user defined hangup handler, returns previous handler, “Auto” and “Disabled” are reserved for change hangup handle mode. Hangup Handle Mode: Automatic(“Auto” as handler’s name): Idtel/IdVoIP will terminate current script and release line once hangup was detected, this is default mode to help current script releases line immediately without be changed.

Disabled(“Disabled” as handler’s name): Idtel/IdVoIP will neither terminate script running nor call user defined procedure, the return value of builtin function have to correspond following process.

User Defined(except “Auto” or “Disabled” as handler’s name): User defined procedure will be called when hangup was detected. a builtin function OnHangup() can specify hangup handle mode or set user defined handler.

User Defined Hangup Handler Prototype: function HandlerName(input $lineNo, input $functionFullName) functionFullName represents functions name of stack traceback, eg: GetDigits@GetUserId@main

```
string OnHangup (
        string function
)
```

Parameters
function
[in] Function will be called once hangup was detected, an empty string only retrieve current setting.

Return values
Returns previous or current setting.

14.1.16 TTSEnumVoices

Platform: win32
Details
Enumerates Text-To-Speech voices which match specified attributes.

```
int TTSEnumVoices (
        string attribs ,
        array null ,
        string errbuf
```


)

Parameters
attribs
[in] Semicolon-separated attributes, e.g.: 'Gender=Female;Language=409.'

null
[out] Returns an array of strings consisting of all available Text-To-Speech voices on current system.

errbuf
[out] Error messages will be returned in this buffer.

Return values
Number of voices found, -1 means failure.

14.1.17 TTSConvertToWave

Platform: win32
Details
Uses Text-To-Speech engines to synthesize text strings into spoken audio by creating synthetic voices.

```
int TTSConvertToWave (
        string text ,
        string voice ,
        int option ,
        int volume ,
        int rate ,
        string langdir ,
        string fileName ,
        int soundFileType ,
        string errbuf
)
```

Parameters
text
[in] The text to convert; complies with 'Synthesis Markup'.

voice
[in] A synthetic voice.

option
[in] Flags passes into TTS engine, 0 means using default value.

volume
[in] An integer value between 0 and 100.

rate
[in] The voice rate adjustment. The valid range is -10 to +10, 0 uses default value.

langdir

[in] Language folder the output will be going to, for example, en-us.

fileName
[in] The name or full path of the file to store result.

soundFileType
[in] Wave sound file type. Supported Sound File Types indicates all supported file types except vox file.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -3 means hangup, -1 means failure.

14.1.18 TTSSpeakText

Platform: win32
Details
Uses Text-To-Speech engines to synthesize text strings into spoken audio by creating synthetic voices.

```
int TTSSpeakText (
        string text ,
        string voice ,
        int option ,
        int volume ,
        int rate ,
        string stopTones ,
        string errbuf
)
```

Parameters
text
[in] the text to convert; complies with 'Synthesis Markup'.

voice
[in] A synthetic voice.

option
[in] flags passes into TTS engine, some values for MS TTS engine: 0 = Default value, Parse the text as XML only if the first character is a left-angle-bracket. 4 = The input string is a file name, and the file text should be spoken. 8 = The input text will be parsed for XML markup. 16 = The The input text will not be parsed for XML markup.

volume
[in] An integer value between 0 and 100.

rate
[in] The voice rate adjustment. The valid range is -10 to +10, 0 uses default value.

stopTones

[in] Gives a list of tones that will interrupt the play. Allowed values: a string of zero or more digits, including the star (*) and pound sign (#).

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -3 means hang up, -1 means failure.

14.1.19 SREnumRecognizers

Platform: win32
Details
Enumerates speech recognizers which match specified attributes.

```
int SREnumRecognizers (
        string attribs ,
        array null ,
        string errbuf
)
```

Parameters
attribs
[in] Semicolon-separated attributes, e.g.: 'Gender=Female;Language=409'.

null
[out] Returns an array of strings consisting of all available speech recognition engines on the current system.

errbuf
[out] Error messages will be returned in this buffer.

Return values
Number of recognizers found, -1 means failure.

14.1.20 SRLoadGrammar

Platform: win32
Details
Load grammar file for recognition engine, uses SRListen to actively listen on this channel, SRRelease to free resource.

```
int SRLoadGrammar (
        string recognizer ,
        string language ,
        string grammar ,
        int option ,
        string errbuf
```


)

Parameters
recognizer
[in] A speech recoginzer engine, an empty string means the default engine is used.

language
[in] Language type, for example, en-us.

grammar
[in] File name of the xml grammar file or CFG compiled grammar file.

option
[in] Grammar option, 0 means using default value.

errbuf
[out] Error messages will be returned in this buffer.

Return values
Returns SR internal handle (greater than zero), -1 means failure.

14.1.21 SRLoadDictation

Platform: win32
Details
Load dictation topic for recognition engine, SRListen to actively listen on this channel, SRRelease to free resource.

```
int SRLoadDictation (
        string recognizer ,
        string topic ,
        int option ,
        string errbuf
)
```

Parameters
recognizer
[in] Speech recognizer engine, an empty string means the default engine is used.

topic
[in] The dictation topic, an empty string means default topic.

option
[in] The dictation option, 0 means default value.

errbuf
[out] Error messages will be returned in this buffer.

Return values
Returns SR internal handle (greater than zero), -1 means failure.

14.1.22 SRListen

Platform: win32
Details
Wait sound recognition result.

```
int SRListen (
        int srhandle ,
        string rule ,
        int timeout ,
        int maxSilences ,
        string stopTones ,
        int option ,
        string result ,
        string errbuf
)
```

Parameters
srhandle
[in] Internal SR handle returned by SRStartDictation/SRStartGrammar.

rule
[in] Rule to be activated, empty string will activate all top level rules (not used for dictation)

timeout
[in] Timeout (in seconds) for this request.

maxSilences
[in] Maximum period of silence to allow in listening.

stopTones
[in] Gives a list of tones that will interrupt the play. Allowed values: a string of zero or more digits, including the star (*) and pound sign (#).

option
[in] 0 – default value; try best to do recognition, 1 – returns immediately once false recognition occurs.

result
[out] recognition result.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, 1 means recognition was interrupted by key press, 2 means recognition failed, 3 means timeout, -3 means hangup, -2 means invalid SR handle, -1 means failure.

14.1.23 SRRelease

Platform: win32
Details
Free loaded grammar or dictation resource.

```
int SRRelease(
        int srhandle,
        string errbuf
)
```

Parameters
srhandle
[in] Internal SR handle returned by SRStartDictation/SRStartGrammar.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -1 means failure, -2 means invalid SR handle.

14.1.24 IsCSPSupported

Platform: win32
Details
Check if current channel supports CSP (continuous speech processing).

```
int IsCSPSupported(

)
```

Parameters
None.

Return values
1 means support CSP, 0 means does not.

14.1.25 SRPlayAndListen

Platform: win32
Details
Start recognition while playing voice file. The playing can be interrupted by speech. Returns recognition result. This function is only supported on a CSP channel.


```
int SRPlayAndListen (
        string language ,
        string text ,
        int srhandle ,
        string rule ,
        int timeout ,
        int maxSilences ,
        int maxDigits ,
        string stopTones ,
        int option ,
        string result ,
        string errbuf
)
```

Parameters

language
[in] Language type, for example, en-us.

text
[in] The text to spell or the names of the file to play, separated by newline('\n'). If the first character is '?', which indicates following text need to spell out, with specified mode: '?0' = spell the text with case-insensitive and without callsign. '?1' = spell the text with case-sensitive without callsign. '?2' = spell the text with case-insensitive with callsign. '?3' = spell the text with case-sensitive and callsign.

srhandle
[in] Internal SR handle returned by SRStartDictation/SRStartGrammar.

rule
[in] Rule to be activated, an empty string will activate all top level rules (not used for dictation).

timeout
[in] Timeout (in seconds) for this request.

maxSilences
[in] Maximum period of silence to allow in listening.

maxDigits
[in] Maximum number of digits to wait for.

stopTones
[in] Touch tone digits that will end input immediately.

option
[in] 0 – default value; try best to do recognition, 1 – returns immediately once false recognition occurs.

result
[out] Recognition result or digits received for user input.

errbuf
[out] Error messages will be returned in this buffer.

Return values

0 means success, 1 means result contains pressed keys, 2 means recognition failed, 3 means timeout, -3 means hangup, -2 means invalid SR handle, -1 means failure.

14.1.26 SRSpeakAndListen

Platform: win32
Details
Activate recognition while playing synthetic voice speaking the text. The playing can be interrupted by speech. Returns recognition result. This function is only supported on a CSP channel.

```
int SRSpeakAndListen (
        string text ,
        string voice ,
        int option ,
        int volume ,
        int rate ,
        int srhandle ,
        string rule ,
        int timeout ,
        int maxSilences ,
        int maxDigits ,
        string stopTones ,
        int recogType ,
        string result ,
        string errbuf
)
```

Parameters
text
[in] The text to convert, complies with 'Synthesis Markup'.

voice
[in] Synthetic voice

option
[in] flags passes into TTS engine, some values for MS TTS engine: 0 = Default value, Parse the text as XML only if the first character is a left-angle-bracket. 4 = The input string is a file name, and the file text should be spoken. 8 = The input text will be parsed for XML markup. 16 = The The input text will not be parsed for XML markup.

volume
[in] An integer value between 0 and 100

rate
[in] The voice rate adjustment. Valid range is -10 to +10, 0 uses default value.

srhandle
[in] Internal SR handle returned by SRStartDictation/SRStartGrammar.

rule

[in] Rule to be activated, empty string will activate all top level rules (not used for dictation)

timeout
[in] Timeout (in seconds) for this request.

maxSilences
[in] Maximum period of silence to allow in listening.

maxDigits
[in] Maximum number of digits to wait for.

stopTones
[in] Touch tone digits that will end input immediately.

recogType
[in] 0 – default value; try best to do recognition, 1 – returns immediately once false recognition occurs.

result
[out] Recognition result or digits received for user input.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, 1 means result contains pressed keys, 2 means recognition failed, 3 means timeout, -3 means hangup, -2 means invalid SR handle, -1 means failure.

14.1.27 SRRecoResult

Platform: win32
Details
Retrieve full recognition result in kvgroup; contains matched rules, properties and elements.

```
int SRRecoResult (
        int srhandle ,
        kvgroup result ,
        string errbuf
)
```

Parameters
srhandle
[in] Internal SR handle returned by SRStartDictation/SRStartGrammar.

result
[out] The group contains detailed recognition results, if any.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -1 means failure, -2 means invalid SR handle.0 on success, non-zero otherwise. The

recognition result has the following format: # KVGROUP-V2.0 RecoResult = SREngineID = ec468149-6916-11d2-9427-00c04f8ef48f; LangID = 1033; GrammarID = 0; RuleID = 1; Rule = Users; Text = “John Doe”; Confidence = 1; Rules “” = Users “” = ID = 1; FirstElement = 0; CountOfElements = 2; Confidence = 1; ; ; Properties “” = AccountId “” = ID = 100; Value = JohnD; FirstElement = 0; CountOfElements = 2; Confidence = 1; ; ; Elements “” = 0 “” = DisplayText = John; LexicalForm = John; RequiredConfidence = 0; ActualConfidence = 1; ; 1 “” = DisplayText = Doe; LexicalForm = Doe; RequiredConfidence = 0; ActualConfidence = 1; ; ; ;

14.1.28 SRRecoFromFile

Platform: win32
Details
Perform recognition from wave file.

```
int SRRecoFromFile (
        int srhandle ,
        string rule ,
        string wavefile ,
        string result ,
        string errbuf
)
```

Parameters
srhandle
[in] Internal SR handle returned by SRStartDictation/SRStartGrammar.

rule
[in] Rule to be activated, an empty string will activate all top level rules (not used for dictation).

wavefile
[in] A wave file.

result
[out] A recognition result.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, 2 means recognition failed, -2 means invalid SR handle, -1 means failure.0 on success, non-zero otherwise.

14.1.29 SRSetRetainAudio

Platform: win32
Details
Enable/disable recognition engine and retain audio for results.


```
int SRSetRetainAudio (
        int srhandle ,
        int onoff ,
        string errbuf
)
```

Parameters
srhandle
[in] Internal SR handle returned by SRStartDictation/SRStartGrammar.

onoff
[in] Enable(1) or disable (0).

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -2 means invalid SR handle, -1 means failure.0 on success, non-zero otherwise.

14.1.30 SRSaveRetainAudio

Platform: win32
Details
Save retained audio to a wave file.

```
int SRSaveRetainAudio (
        int srhandle ,
        string wavefile ,
        int startelement ,
        int elements ,
        string errbuf
)
```

Parameters
srhandle
[in] Internal SR handle returned by SRStartDictation/SRStartGrammar.

wavefile
[in] The name of the wave file that stores the stream audio.

startelement
[in] The element in the result data from which the audio stream should start.

elements
[in] The total number of words.

errbuf
[out] Error messages will be returned in this buffer.

Return values

0 means success, -2 means invalid SR handle, -1 means failure.0 on success, non-zero otherwise.

14.1.31 LoadVocalScript

Platform: win32
Details
Load the vocal script into a pslang array to help the TTS function access vocal text.

```
int LoadVocalScript (
        string language ,
        string scriptfile ,
        array scripts
)
```

Parameters
language
[in] Language Type, for example en-us.

scriptfile
[in] The vocal script file name to load.

scripts
[out] A list of scripts, indexed by wave file name.

Return values
The number of scripts successfully loaded.The number of scripts has been loaded.

14.1.32 QueuePut

Platform: win32
Details
Put an item into the named queue. Any item can only be placed into a single named queue.

```
int QueuePut (
        string queue ,
        string data
)
```

Parameters
queue
[in] The name of the queue.

data
[in] The item to create or update in the queue, an empty value will remove the item from this queue.

Return values
The number of items in the queue.The number of items in the queue.

14.1.33 QueuePeek

Platform: win32
Details
Put an item into the named queue.

```
int QueuePeek (
        string queue ,
        int trunk ,
        string data
)
```

Parameters
queue
[in] The name of the queue.

trunk
[in] The trunk to check.

data
[out] The item retrieved from the queue which belongs to the specified trunk.

Return values
The position of this item in the queue, starts from 0 otherwise -1 for no item can be found.The number of items in the queue.

14.1.34 QueueGet

Platform: win32
Details
Get an item from the named queue.

```
int QueueGet (
        string queue ,
        int timeout ,
        string data ,
        int trunk ,
        string uid
)
```

Parameters
queue
[in] The name of the queue.

timeout
[in] Timeout (milliseconds).

data
[out] The item retrieved from the queue.

trunk
[out] The trunk number which created this item.

uid
[out] The unique ID of the call which created this item.

Return values
0 means success, 3 means timeout, -3 means hangup, -1 means failure.

14.1.35 GetCallUID

Platform: win32
Details
Returns the unique ID for the current call on the specified trunk.

```
string GetCallUID (
        int trunk
)
```

Parameters
trunk
[in] The trunk number to query, -1 means current trunk.

Return values
The unique ID for the current call, empty string means no active call on tbe specified trunk.

14.1.36 Bridge

Platform: win32
Details
Bridges/Un-bridges two channels.

```
int Bridge (
        int channel ,
        int other ,
        int mode
)
```

Parameters
channel
[in] The current channel.

other
[in] The channel to bridge.

mode
[in] 1: half-duplex bridge, 2: full-duplex bridge, 0: un-bridge.

Return values

0 means success, otherwise failure.

14.1.37 Stop

Platform: win32
Details
Stop the current voice function (PlayFile/RecordFile/GetDigits) on the specified channel.

```
int Stop (
        int channel ,
        int mode
)
```

Parameters
channel
[in] The channnel.

mode
[in] 0: asynchronous, 1: synchronous.

Return values
0 means success, otherwise failure.

14.1.38 RecordFileEx

Platform: win32
Details
Enhanced version of RecordFile offers the ability to record sound from current trunk plus an extra trunk to a sound file.

```
int RecordFileEx (
        string fileName ,
        int soundFileType ,
        int maxSeconds ,
        int maxSilences ,
        string stopTones ,
        int append ,
        int tone ,
        int line1 ,
        int mode1 ,
        int line2 ,
        int mode2
)
```

Parameters
fileName
[in] The name and path of the file to record.

soundFileType
[in] Sound file type. Supported Sound File Types indicates all supported file types, -1 means using RecFileType defined in configuration file.

maxSeconds
[in] Maximum time to allow recording, in seconds. (If this parameter is set to 0, the method ignores time limit.)

maxSilences
[in] Maximum period of silence to allow in recording, measured in seconds. Often a silence time-out is used to end the recording of a new voice mail message, for example. (If this parameter is set to 0, the method ignores silence.) The default value is 0.

stopTones
[in] Gives a list of tones that will interrupt the recording. Allowed values: a string of zero or more digits, including the star (*) and pound sign (#).

append
[in] Determines whether the recording is appended to an existing file of the same name. By default, `append` is 0.

tone
[in] Plays a tone before recording, if set to a non-zero value.

line1
[in] The first line to record from.

mode1
[in] Records the transmit side if set to zero, otherwise records the receive side.

line2
[in] The second line to record from.

mode2
[in] Records the transmit side if set to zero, otherwise records the receive side.

Return values
0 means success, 1 means playing was interrupted by key press, -3 means hangup, -1 means failure.

14.1.39 Supported Sound File Types

The PlayFile() and RecordFile() IVR functions support the following sound file types:

Value	Sound File Type
0	4 bit 6kHz ADPCM 24kbps Vox.
1	4 bit 8kHz ADPCM 32kbps Vox.
2	8 bit 6kHz MuLaw PCM 48kbps Vox.
3	8 bit 8kHz MuLaw PCM 64kbps Vox.
4	8 bit 6kHz Linear PCM 48kbps Wave.
5	8 bit 8kHz Linear PCM 64kbps Wave sound file type.

6	8 bit 11Hz Linear PCM 88kbps Wave sound file type.
7	8 bit 6kHz MuLaw PCM 48kbps Wave sound file type.
8	8 bit 8kHz MuLaw PCM 64kbps Wave sound file type.
9	8 bit 11Hz MuLaw PCM 88kbps Wave sound file type.
10	8 bit 6kHz ALaw PCM 48kbps Wave sound file type.
11	8 bit 8kHz ALaw PCM 64kbps Wave sound file type.
12	8 bit 11Hz ALaw PCM 88kbps Wave sound file type.
13	4 bit 6Hz Dialogic ADPCM 24kbps Wave.
14	4 bit 8Hz Dialogic ADPCM 32kbps Wave.
15	8 bit 6kHz ALaw PCM 48kbps Vox.
16	8 bit 8kHz ALaw PCM 64kbps Vox.

14.2 Password Manager Service Functions

The following functions can be used to interact with the Password Manager Service. The pspushpass API must be installed and properly configured in order for these functions to run successfully.

14.2.1 FindUser

Platform: win32
Details
Find a user based on the telephone keypad translation of their profile ID or a numeric login ID on a particular target system. This function returns the profile ID(s) and full name(s) matching a given code.

```
int FindUser (
        string ivrid ,
        string targetid ,
        string userbuf ,
        string errbuf
)
```

Parameters
ivrid
[in] A telephone keypad code corresponding to a user's login ID (e.g., "3733" for "Fred").

targetid
[in] If the numeric ID specified in *ivrid* is actually an account on a target, this parameter must specify the target ID on which the account resides. If this parameter is empty, *ivrid* will be assumed to be the numeric telephone keypad translation of the user's profile ID (where 0 = QZ).

userbuf
[out] The matching user ID(s) and full name(s) will be returned in this buffer with each entry separated by a newline.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise. There are some common error codes return by ivrpushpass functions: -101 - function is not found or DLL is not loaded - The list of users will be put into userbuf in the following format:

```
userid1 full name
userid2 full name
```

14.2.2 GetUserQuestions

Platform: win32
Details
Get a user's IVR questions (questions with numeric answer formats). This function returns a list of questions to ask a user going through the IVR service. All answers will have numeric formats.

```
int GetUserQuestions (
        string userid ,
        string questionbuf ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

questionbuf
[out] A list of the user's configured questions will be returned in this buffer.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise. The list of questions will be put into questionbuf in the following format:

```
QID1 question1
QID2 question2
```

14.2.3 GetUserAccounts

Platform: win32
Summary: Get a list of a user's accounts.
Details
This function returns a list of accounts for a user.


```
int GetUserAccounts (
        string userid ,
        string options ,
        string accountbuf ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

options
[in] Only pass back accounts on target systems with the above capabilities. Leave blank to return all accounts. Otherwise, specify one or more of the following capabilities:

- R - reset passwords
- V - verify passwords
- U - unlock accounts
- A - ACE target
- H - Hard drive encryption target

accountbuf
[out] A list of the user's accounts will be returned in this buffer.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise. The list of accounts will be put into accountbuf in the following format:

```
TARGET1 "account1"
TARGET2 "account2"
```

14.2.4 ValidateUserAnswers

Platform: win32
Summary: Validate a user's numeric answers
Details
Can be used to validate a user's answers to the questions returned through the GetUserQuestions function. All answers must have numeric formats.

```
int ValidateUserAnswers (
        string userid ,
        string answers ,
        string errbuf
```


)

Parameters
userid
[in] A valid profile ID.

answers
[in] Buffer containing the answers to validate. Answers must be in the following format:

```
QID1 "Answer text 1" QID2 "Answer text 2"
```

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.5 RandomPassword

Platform: win32
Summary: Get a random password which meets the password strength rules.
Details
Returns a randomly generated password which matches the configured password strength rules.

```
int RandomPassword(
        string targetid,
        string pwdbuf,
        string errbuf
)
```

Parameters
targetid
[in] The target system on which the password will be generated.

pwdbuf
[out] The random password will be returned in this buffer.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.6 CheckPasswordStrength

Platform: win32
Details

Check if a given password meets the configured password strength rules.

```
int CheckPasswordStrength(
        string userid,
        string pwd,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

pwd
[in] The password to check.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.7 VerifyPassword

Platform: win32
Summary: Verify a user's password on a target system.
Details

```
int VerifyPassword(
        string userid,
        string pwd,
        string targetid,
        string loginid,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

pwd
[in] The password value.

targetid
[in] The target system on which the password will be verified.

loginid
[in] The user's login ID on the target system specified by `targetid`.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.8 ResetSinglePassword

Platform: win32
Summary: Reset a user's password on a specified target.
Details
Reset a user's password on a specified target, it may also synchronize the password in the target group if target group has password synchronization enabled.

```
int ResetSinglePassword(
        string userid,
        string pwd,
        string targetid,
        string loginid,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

pwd
[in] The new password value.

targetid
[in] The target system on which the password will be set.

loginid
[in] The user's login ID on the target system specified by `targetid`.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.9 ResetSinglePasswordEx

Platform: win32
Summary: Reset a user's password on specified target systems.
Details

Reset a user's password on specified target systems, all target systems must remain in same target group. It may also synchronize the password in the target group if target group has password synchronization enabled.

```
int ResetSinglePasswordEx (
        string userid ,
        string pwd ,
        array targetids ,
        array loginids ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

pwd
[in] The new password value.

targetids
[in] The target systems on which the password will be set; an array of strings.

loginids
[in] The user's login IDs on the target systems specified by `targetids`, array of string.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.10 ResetAllPasswords

Platform: win32
Summary: Reset a user's password on all target systems in a specific target system group
Details

```
int ResetAllPasswords (
        string userid ,
        string pwd ,
        string targetid ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

pwd
[in] The new password value.

targetid
[in] The ID of the target system group to which this target system belongs.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.11 ResetAllPasswordsEx

Platform: win32
Summary: Reset a user's password on all target systems in a specific target system group
Details
This function should be used with version 7.0 or higher.

```
int ResetAllPasswordsEx (
        string userid ,
        string pwd ,
        array targetids ,
        array loginids ,
        string errbuf
)
```

Parameters
userid
[in] A valid user ID.

pwd
[in] The new password value.

targetids
[in] The target systems on which the password will be set, array of string.

loginids
[in] The user's login IDs on the target systems specified by `targetids`; array of strings.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.12 ResetExpireSinglePassword

Platform: win32
Summary: Reset and expire a user's password on a single target system
Details

```
int ResetExpireSinglePassword (
        string userid ,
        string pwd ,
        string targetid ,
        string loginid ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

pwd
[in] The new password value.

targetid
[in] The target system on which the password will be set.

loginid
[in] The user's login ID on the target system specified by `targetid`.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.13 ResetExpireSinglePasswordEx

Platform: win32
Summary: Reset and expire a user's password on specified target systems, all target systems must remain in same target system group
Details

```
int ResetExpireSinglePasswordEx (
        string userid ,
        string pwd ,
        array targetids ,
        array loginids ,
        string errbuf
```


)

Parameters
userid
[in] A valid profile ID.

pwd
[in] The new password value.

targetids
[in] The target systems on which the password will be set, array of string.

loginids
[in] The user's login IDs on the target systems specified by `targetids`; array of strings.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.14 ResetExpireAllPasswords

Platform: win32
Summary: Reset and expire a user's password on all target systems in specified target system group.
Details

```
int ResetExpireAllPasswords (
        string userid ,
        string pwd ,
        string targetid ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

pwd
[in] The new password value.

targetid
[in] The ID of the target system group to which this target system belongs.

errbuf
[out] Error messages will be returned in this buffer.

Return values

0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.15 UnlockAllAccounts

Platform: win32
Summary: Unlock all of a user's accounts
Details

```
int UnlockAllAccounts (
        string userid ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.16 UnlockAllAccountsEx

Platform: win32
Summary: Unlock all of a user's accounts.
Details
Function introduced for Version 7.0 and later.

```
int UnlockAllAccountsEx (
        string userid ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.17 UnlockSingleAccount

Platform: win32
Summary: Unlock a user's account on a specific target system.
Details

```
int UnlockSingleAccount (
        string userid ,
        string targetid ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

targetid
[in] Unlock the user's account on this target system.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.18 UnlockSingleAccountEx

Platform: win32
Summary: Unlock a user's account on a specific target system.
Details
Function introduced for Version 7.0 and later.

```
int UnlockSingleAccountEx (
        string userid ,
        string targetid ,
        string longid ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

targetid
[in] Unlock the user's account on this target system.

longid
[in] A valid account ID.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.19 ValidateEnrollmentPIN

Platform: win32
Summary: Validate an enrollment PIN for an IVR user
Details

```
int ValidateEnrollmentPIN (
        string userid ,
        string pin ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

pin
[in] The enrollment PIN to validate.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.20 ValidateEnrollmentPINEx

Platform: win32
Summary: Validate an enrollment PIN for an IVR user.
Details
Function introduced for version 7.1 and higher.

```
int ValidateEnrollmentPINEx (
        string userid ,
        string pin ,
        string errbuf
)
```

Parameters

userid
[in] A valid profile ID.

pin
[in] The enrollment PIN to validate.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.21 ReadUserstat

Platform: win32
Summary: Read a USERSTAT tag.
Details

```
int ReadUserstat (
        string userid ,
        string tag ,
        string value ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

tag
[in] The userstat tag to read.

value
[out] The userstat tag value

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.22 WriteUserstat

Platform: win32
Summary: Write a USERSTAT tag.

Details

```
int WriteUserstat (
        string userid ,
        string tag ,
        string value ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

tag
[in] The userstat tag to set.

value
[in] Set the tag to this string value.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.23 DisableUser

Platform: win32
Summary: Disable a user profile.
Details

```
int DisableUser (
        string userid ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.24 EnableUser

Platform: win32
Summary: Enable a user profile.
Details

```
int EnableUser (
        string userid ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.25 ACEEnableToken

Platform: win32
Summary: Enable an ACE token.
Details

```
int ACEEnableToken (
        string userid ,
        string targetid ,
        string tokenid ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

targetid
[in] The target system on which this account resides.

tokenid
[in] The token serial number on the target system.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.26 ACEDisableToken

Platform: win32
Summary: Disable an ACE token.
Details

```
int ACEDisableToken (
        string userid ,
        string targetid ,
        string tokenid ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

targetid
[in] The target system on which this account resides.

tokenid
[in] The token serial number on the target system.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.27 ACEGetEmergencyCodes

Platform: win32
Summary: Get emergency access codes for an ACE token
Details


```
int ACEGetEmergencyCodes (
        string userid ,
        string targetid ,
        string tokenid ,
        int numcodes ,
        int numhours ,
        int numdigits ,
        string codes ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

targetid
[in] The target system on which this account resides.

tokenid
[in] The token serial number on the target system.

numcodes
[in] The number of codes to generate.

numhours
[in] The number of hours before the codes expire.

numdigits
[in] The number of digits in each code.

codes
[out] The emergency codes will be returned in this buffer.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.28 ACEClearEmergencyCodes

Platform: win32
Summary: Clear emergency access codes for an ACE token.
Details


```
int ACEClearEmergencyCodes(
        string userid,
        string targetid,
        string tokenid,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

targetid
[in] The target system on which this account resides.

tokenid
[in] The token serial number on the target system.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.29 ACESetPIN

Platform: win32
Summary: Set the PIN for an ACE token.
Details

```
int ACESetPIN(
        string userid,
        string targetid,
        string tokenid,
        string pin,
        string newpin,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

targetid
[in] The target system on which this account resides.

tokenid
[in] The token serial number on the target system.

pin
[in] The desired PIN. Leave blank to have a PIN generated.

newpin
[out] If a PIN is generated, the new PIN will be returned in this buffer.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.30 ACEClearPIN

Platform: win32
Summary: Clear the PIN for an ACE token.
Details

```
int ACEClearPIN(
        string userid,
        string targetid,
        string tokenid,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

targetid
[in] The target system on which this account resides.

tokenid
[in] The token serial number on the target system.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.31 ACEResynchronizeToken

Platform: win32
Summary: Resynchronize an ACE token

Details

```
int ACEResynchronizeToken (
        string userid ,
        string targetid ,
        string tokenid ,
        string firstcode ,
        string secondcode ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

targetid
[in] The target system on which this account resides.

tokenid
[in] The token serial number on the target system.

firstcode
[in] The current token code displayed.

secondcode
[in] The next token code displayed.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.32 LogEvent

Platform: win32
Summary: Log an event through the Password Manager service.
Details

```
int LogEvent (
        int eventtype ,
        string reqby ,
        string userid ,
        string hostid ,
        string host ,
        string result ,
        string errbuf
```


)

Parameters
eventtype
[in] A valid event type.

reqby
[in] A valid requester.

userid
[in] A valid profile ID.

hostid
[in] A valid target ID.

host
[in] A valid target name.

result
[in] An event message.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.33 GetChallengeResponse

Platform: win32
Summary: Gets a Challenge/Response
Details
Gets a challenge/response string for a user, possibly requiring the user to provide input in order to generate either the challenge or response (depending on the target system).

```
int GetChallengeResponse(
        string accountid,
        string targetid,
        string challenge,
        string inputs,
        string response,
        string errbuf
)
```

Parameters
accountid
[in] An account ID.

targetid

[in] The target system on which this account resides.

challenge
[in] The challenge string for the request.

inputs
[in] Optional target system specific inputs.

response
[out] The response string will be returned in this buffer.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -101 means function hasn't been found, -1 means failure.0 on success, non-zero otherwise.

14.2.34 LoadInstanceCfg

Platform: win32
Details
The instance configuration for the current trunk rather than the default. Using this technique can allow one HiTPM instance to integrate with one or more HiPM instances.

```
int LoadInstanceCfg(
        string cfgfile
)
```

Parameters
cfgfile
[in] The instance configuration file path.

Return values
0 means success, -101 means function hasn't been found, otherwise means failure.0 on success, non-zero otherwise.

14.3 Voice Print Functions

The following functions can be used to perform voice print related operations. In order for these functions to execute, you must have installed the supported voice print software.

14.3.1 GetVoiceUserList

Platform: win32
Summary: Get the number of users in the voice check DB and, optionally, a list of their IDs.
Details
Determine how many users are in the voice check DB. If listids is non-zero, a list of the user IDs for all users will be returned in the userlist parameter.

```
int GetVoiceUserList (
        int listids ,
        array userlist ,
        string errbuf
)
```

Parameters
listids
[in] If this is non-zero, a list of user IDs will be returned.

userlist
[out] A list of user IDs.

errbuf
[out] Error messages will be returned in this buffer.

Return values
Number of users found, negative result means failure.Number of users in the voice check DB.

14.3.2 CompletedEnrollments

Platform: win32
Summary: Get the number of completed enrollments
Details
Determine how many enrollments the user has completed. An enrollment is considered completed only when all required veriphrases have the minimum number of renditions enrolled.

```
int CompletedEnrollments (
        string userid ,
        int numenrollments ,
        int numphrases ,
        int numrenditions ,
        string errbuf
```


)

Parameters
userid
[in] A valid profile ID.

numenrollments
[in] The number of enrollments to check.

numphrases
[in] The expected number of veriphrases for each enrollment.

numrenditions
[in] The expected number of renditions for each veriphrase.

errbuf
[out] Error messages will be returned in this buffer.

Return values
Number of completed enrollments, negative result means failure.Number of completed enrollments

14.3.3 IsEnrollmentComplete

Platform: win32
Summary: Check if a user has completed a specific enrollment.
Details
Determine if the user has completed a specific enrollment. An enrollment is considered completed only when all required veriphrases have the minimum number of renditions enrolled.

```
int IsEnrollmentComplete (
        string userid ,
        string enrollmentid ,
        int numenrollments ,
        int numphrases ,
        int numrenditions ,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

enrollmentid
[in] The ID of the enrollment.

numenrollments
[in] The number of enrollments to check.

numphrases
[in] The expected number of veriphrases for each enrollment.

numrenditions
[in] The expected number of renditions for each veriphrase.

errbuf
[out] Error messages will be returned in this buffer.

Return values
Returns 1 if the enrollment is completed, 0 if it is not, negative result means failure.1 if the enrollment is completed, 0 if it is not, negative result means failure.

14.3.4 DeleteEnrolledUser

Platform: win32
Summary: Delete a user from the voice enrollment database.
Details
Deletes a user from the voice enrollment database as well as all of the user's enrollments and archived speech data.

```
int DeleteEnrolledUser(
        string userid,
        string errbuf
)
```

Parameters
userid
[in] The enrolled user to delete.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, otherwise means failure.0 on success, non-zero otherwise.

14.3.5 EnrollVoicePrint

Platform: win32
Summary: Enroll a voice print
Details
Prompts the user to answer as many questions as indicated by the numquestions parameter. There must be a corresponding vocal called VeriPhrase_N for each question where <N> is the 1-based index for the question. If any one of these vocals is missing, an error will be returned. Each question will be asked numprompt times and a voice print will be generated for each iteration.


```
int EnrollVoicePrint (
        string language ,
        string userid ,
        int numphrases ,
        int numrenditions ,
        int maxretry ,
        string enrollmentID ,
        string errbuf
)
```

Parameters
language
[in] Language type, for example, en-us.

userid
[in] A valid profile ID.

numphrases
[in] The number of questions to prompt the user.

numrenditions
[in] The number of times to ask each question.

maxretry
[in] The number of failed attempts allowed for each question.

enrollmentID
[in] (Optional) The ID to associate with the enrollment. This parameter can be left blank if users only enroll once.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, 1 means recording was interrupted by key press, -3 means hangup, other negative value means failure.0 on success, non-zero otherwise.

14.3.6 VerifyVoicePrint

Platform: win32
Summary: Verify a voice print.
Details
Verifies a voice print for a user.


```
int VerifyVoicePrint (
        string language ,
        string userid ,
        int numphrases ,
        int numrenditions ,
        int nummatch ,
        int maxattempts ,
        string errbuf
)
```

Parameters
language
[in] Language type, for example, en-us.

userid
[in] A valid profile ID.

numphrases
[in] The total number of questions the user must enroll.

numrenditions
[in] The total number of renditions the user must enroll for each phrase.

nummatch
[in] The number of matches needed for validation.

maxattempts
[in] The number of attempts before the user is denied access.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, 1 means recording was interrupted by key press, -3 means hangup, -2 means user is not enrolled, other negative value means failure.0 on success, -2 if the user is not enrolled, non-zero otherwise.

14.3.7 GetUserVerificationThreshold

Platform: win32
Details
Retrieves the voiceprint verification threshold setting for the specified user.

```
int GetUserVerificationThreshold (
        string userid ,
        int value ,
        string errbuf
)
```

Parameters
userid

[in] A valid profile ID.

value
[out] The user's verification threshold setting.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -2 means user is not enrolled, other negative value means failure.0 on success, -2 if the user is not enrolled, non-zero otherwise.

14.3.8 SetUserVerificationThreshold

Platform: win32
Details
Sets the voiceprint verification threshold for the specified user.

```
int SetUserVerificationThreshold(
        string userid,
        int value,
        string errbuf
)
```

Parameters
userid
[in] A valid profile ID.

value
[in] The new value for the user's verification threshold.

errbuf
[out] Error messages will be returned in this buffer.

Return values
0 means success, -2 means user is not enrolled, other negative value means failure.0 on success, -2 if the user is not enrolled, non-zero otherwise.

Login ID Plugin Functions 15

The following functions can be used within the login ID plugin to manipulate the ID Reservation system.

15.1 ID Reservation

These functions reserve IDs to ensure that generated IDs are unique.

15.1.1 ReserveId

Platform: win32
Summary: Reserves an ID.
Details
Reserves one of profile ID, account ID (on one or all target systems), and attribute values. Only one of profileid, accountid or attributeKey should be provided to this function.

```
int ReserveId(
        string id,
        string profileid,
        string accountid,
        string targetid,
        string attributeKey,
        array attributeValues,
        int caseSensitive
)
```

Parameters
id
[in/out] A pre-existing reservation ID with which to associate the reservations in this request. If this is blank, a new reservation ID will be created. If you want to create a reservation that consists of multiple types of reservable items, call this once with an empty id and then re-use the returned resid upon subsequent calls. This will facilitate revoking the reservation.

profileid
[in] The profile ID to reserve.

accountid
[in] The ID of the account to reserve; used in combination with the targetid parameter. If the targetid parameter is empty, the ID is reserved for all target systems. Both the short ID and long ID are reserved.

targetid

[in] The target ID on which to reserve the given account ID.

attributeKey
[in] The key of the attribute value(s) to reserve.

attributeValues
[in] An array of strings containing the attribute value(s) to reserve.

caseSensitive
[in] 0 for case insensitive reserve attributes checking/updating, otherwise case sensitive. When reserving profile id, the comparison will always be in case insensitive mode.

Return values
0 on success, 1 on usage error, 2 on internal error, 3 on invalid passed-in reservation ID, or 4 when the system is unable to satisfy the requested reservation.

15.1.2 ReserveRevoke

Platform: win32
Summary: Revokes a reservation.
Details
Revokes all identifiers grouped by the given reservation ID.

```
int ReserveRevoke (
        string id
)
```

Parameters
id
[in] A pre-existing reservation ID whose reservations are to be revoked.

Return values
0 on success, 2 on internal error, or 3 on invalid passed-in reservation ID.

Appendices

Using Regular Expressions A

Regular expressions provide a way to identify patterns of text in strings.

Bravura Security Fabric uses regular expressions to:

- Match and block invalid passwords.
- Identify users who it should not manage.
- Extract text from Telnet/TN3270 screens in Telnet connector scripts.
- Extract text from the output of password hashing programs in DBMS connector scripts.

Regular expressions use a shorthand of literal characters and special characters to define a pattern in a precise and compact way. Literal characters are case sensitive. The regular expression parser used by Bravura Security Fabric uses the following special characters in literal strings:

Character	Purpose	Example	Match
.	Matches any single character	`a.c`	`aac`, `abc`, `a1c`, `a2c`
^	Beginning of a line	`^hat`	`hat`, `hatch`, not `chat`
$	End of a line	`hat$`	`hat`, `chat`, not `hatch`
|	Alternation	`job|task`	`job` or `task`
( )	Grouping	`job(12|34)`	`job12` or `job34`
\	Treat the following special character literally	`\$5\.6\\7`	`$5.6\7`

Note: The | alternates the longest possible string. That is, it matches `job` or `task`, and matches both `jotask` and `jobask`.

Grouping with `( )` limits the alternate branches.

A.1 Character sets

You can define a set of characters within a string. Character sets match any single character enclosed in square brackets `[ ]`. For example:

Expression	Match
`j[oa]n`	`jon` or `jan`

Use the `-` character between two characters in a set to indicate a range. A `-` character outside of a set, or at the beginning or end of a set, is treated literally. For example:

Expression	Match
`[a-c]`	`a`, `b`, or `c`
`[1-3]`	`1`, `2`, or `3`
`[ab-]`	`a`, `b`, or `-`
`[B-D]-3`	`B-3`, `C-3`, or `D-3`

If the first character in the set is ^, then the character set matches any character that is *not* in the list. For example:

Expression	Match
`[^oa]`	Any character except `o` and `a`

All other special characters lose their meaning when included in a character set. For example:

Expression	Match
`^[^*\+.-]`	Line must not start with `*`, `\`, `+`, `.`,or `-`

To include a literal `]` in the character set, make it the first character in the set. It can also be excluded by following ^. The `]` character is also treated literally outside of a set. For example:

Expression	Match
`[^]a-]`	Any character besides `]`, `a`, and `-`
`[]a]`	`]` or `a`
`[a]]`	`a]`

A.2 Quantifiers

You can specify quantities of character sets, grouped characters, or individual characters. The quantifiers `*`, `+`, `?`, and `{ }` act on the preceding set, group, or individual character.

Character	Purpose	Example	Match
`*`	Specifies 0 or more consecutive occurrences	`ab*c`	`ac`, `abc`, `abbbbc`
`+`	Specifies 1 or more consecutive occurrences	`ab+c`	`abc`, `abbbbc`
`?`	Specifies 0 or 1 occurrence	`colou?r`	`color` or `colour`
		`(wo)?man`	`woman` or `man`
`{x}`	Exactly <*x*> occurrences	`ab{4}c`	`abbbbc`
`{x,y}`	At least <*x*>, no more than <*y*> occurrences	`ab{2,4}c`	`abbc`, `abbbc`, `abbbbc`
`{x,}`	At least <*x*> occurrences	`ab{2,}c`	`abbc`, `abbbc`
`{,x}`	No more than <*x*> occurrences	`ab{,2}c`	`ac`, `abc`, or `abbc`

A.3 Shorthand expressions

The following expressions are shorthand for longer or more complex regular expressions:

Expression	Purpose	Example	Match
`[:alpha:]`	Matches must consist of letters.	`[:alpha:]8,16`	`aBcmGLeEhhi`
`[:digit:]`	Matches must consist of digits. [:digit:] is equivalent to [:d:] and `\d`.	`[[:digit:]]+`	`2962954576`
`[:upper:]`	Matches must consist of uppercase characters.	`[[:upper:][:digit:]]+`	`9DDD4MLAG13`
`[:alnum:]`	Matches must be alphanumeric. [:alnum:] is short hand for [a-zA-Z0-9] in the ASCII character set, but the use of [:alnum:] is preferred for compatibility with other languages and character sets.	`[:alnum:]7[^[:digit:]]`	`a3dM9DD+`

A.4 Examples

Only allow letters, numbers, exclamation point, and period	`^[a-zA-Z0-9!.]*$`
At most 8 characters long	`^.,8$`
Must start with a letter	`^[a-zA-Z]`
Must have a number	`[0-9]`
Must end in `-admin` or `-test`	`(-admin|-test)$`
Must begin with three capital letters and an underscore	`^[A-Z]{3}_`

See also:

Bravura Security Fabric uses the ECMAScript regular expression grammar. For more information on regular expressions supported by Bravura Security Fabric, see:

- The ECMA Script Language Specification at:
  http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf
  or
- The Microsoft Developer Network (MSDN) TR1 Regular Expressions document at:
  http://msdn.microsoft.com/en-us/library/bb982727.aspx

Using KVGroups B

Bravura Security Fabric uses Key-Value Groups (KVGroups) as a way to structure and communicate data. KVGroups are used extensively for inter-process communication. Plugins, for example, read a KVGroup on standard input, process the information, and write a KVGroup on standard output.

You use KVGroups to write configuration files and scripts for connectors (Lotus Notes, LDAP, IBM OS/400, IBM DB2/UDB), utility programs (**`idtrack`**), and the Bravura Security Fabric GUI.

Read this chapter to learn how to write and interpret KVGroups.

See also:

There are many PSLANG functions to support the reading, creation, and manipulation of KVGroups. See the PSLang Reference Manual for more information.

B.1 Unicode characters in configuration files and PSLANG scripts

There are some connectors, utilities, and plugins that support the new KVGroup 2.0 format. This includes the Lotus Domino Server connector, Lotus Notes ID Files connector, Unix Listener, and the Database Service (iddb). PSLANG files and configuration files (in KVGroup 2.0 format) support Unicode characters.

When saving information in a text file where Unicode characters are included, the file must include the byte-order mark at the beginning of the file. Text editors such as notepad hide this from the user and pose no problems when you modify and save configuration files.

Bravura Security Fabric consumes the byte-order mark and interprets the rest of the information as defined by the byte-order mark. If no byte-order mark is present, Bravura Security Fabric will read the information as a UTF-8 format. ASCII is a subset of the UTF-8 format; hence no conversion from ASCII is required. If information is generated in UTF-16 or UTF-32, the byte-order mark is required for Bravura Security Fabric to interpret the information correctly.

B.2 KVGroup syntax

A KVGroup has a key, a name, and its contents. It contains a set of key-value pairs, as well as a set of "inner" KVGroups. There is no limit to the level of nested KVGroups.

You write a KVGroup as follows:

```
"<key>" "<name>" = {
   <contents>
   ...
}
```

You write a key-value pair as follows:

```
"<key>" = "<value>"
```

> **Note:** The word *key* has a dual use. A group has a key and the items contained within a group (pairs and inner groups) also have keys.

The following example illustrates how a KVGroup can contain multiple groups and key-value pairs:

```
"capitals" "world capitals grouped by continent" = {
 "NA" "North America" = {
   "USA" = "Washington"
   "Canada" = "Ottawa"
 }
 "Eur" "Europe" = {
   "Britain" = "London"
 }
}
```


B.3 KVGroup libraries

KVGroup helper libraries for Python and Perl are available.

Python and Perl libraries are installed in the \<*instance*>\lib\ directory. The files within \<*instance*>\lib\ work with the 32-bit version of the interpreters. The files within \<*instance*>\lib\\x64 work with the 64-bit version of the interpreters.

The hid_kvgroup python module can be imported to provide a kvg class that can be used to read/write KVGroups. For example:

```
import hid_kvgroup
testkvg = hid_kvgroup.kvg()
testkvg.setKey("thekey")
testkvg.getKey()
```

The hid_kvgroup_perl package can be imported to read/write KVGroups. For example:

```
require hid_kvgroup_perl;
my $testkvg = hid_kvgroup_perl::kvg->new;
$testkvg->setKey('thekey');
print $testkvg->getKey();
```


Connector Operations C

This chapter describes Bravura Security Fabric connector operations. Depending on the Bravura Security Fabric application and the capabilities of each target system, not all operations may be activated.

See also:

Check the target chapters in the Connector Pack Integration Guide for lists of supported connector operations for each target type.

C.1 Password verification operations

user verify password	Checks if a given password is the correct, current password for an account. If the application supports the concept of intruder lockout and the verification fails, the intruder lockout counter is incremented.
administrator verify password	Checks if a given password is the correct, current password for an account without triggering an intruder lockout if the password is not correct.
get server information	Connects to the target system using the specified administrative credentials and queries available system attributes such as version. This operation is primarily used to validate the administrative credentials when **Test Credentials** is clicked in the *Manage the system* (PSA) module.
user change password	Changes the password for an account, from a known current value to a desired new value. If the application supports the concept of intruder lockout, then the intruder lockout counter is cleared and the account unlocked. If the application supports the concept of password expiry, then the expiry date is set according to the expiry policy of the application.
administrator reset password	Administratively resets an account's password to a new value. If the application supports the concept of intruder lockout, then the intruder lockout counter is cleared and the account unlocked. If the application supports the concept of password expiry, then the expiry date is set according to the expiry policy of the application. Disabled accounts will remain disabled.

	You can use the ***Connector behavior*** page to change the behavior of this operation for some targets.
administrator reset+expire password	Administratively resets an account's password to a new value and expires the account's new password, so that the user is forced to change his password the next time he logs in.
verify+reset password	Verifies if the account's password matches the new password, and if the verification fails, administratively sets it to the new password. If the verification succeeds, then the reset is not necessary, and the operation returns success.
resynchronize tokens	Resynchronizes a token.
set token pin	Sets the PIN for a token.
challenge response authentication	Uses challenge response authentication; for example to reset tokens or encrypted passwords.
reset hard drive encryption password	Resets the hard drive encryption password on workstations protected with hard drive encryption systems.
generate an unlock code to recover control of a machine after reboot	Uses challenge response authentication to recover control of a machine from the login screen after reboot.
generate an unlock code to recover control of a machine after reboot and set new password	Uses challenge response authentication to recover control of a machine from the login screen after reboot and sets a new password.
generate an unlock code to recover control of a machine from the unlock screen	Uses challenge response authentication to recover control of a machine from the unlock/screensaver screen.
update subscriber attributes/password	Updates subscriber attributes, typically password.

C.2 Account enable/disable operations

When an account is disabled, a user cannot log in using that account even if the user knows the correct password for the account. Most applications differentiate between an enabled/disabled state and an unlocked/locked state.

check account enabled	Checks if an account is enabled.
enable account	Enables an account.
disable account	Disables an account.

C.3 Intruder lock-out operations

When an account is locked out, the user cannot log in using that account even if the user knows the correct password for the account. Most applications differentiate between an enabled/disabled state and an unlocked/locked state.

check account lock	Checks if an account is locked.
lock account	Locks an account (sets the intruder lockout).
unlock account	Unlocks an account (clears the intruder lockout).
lock account + enter emergency access mode	Locks an account, and then enters emergency access mode.

C.4 Expiry operations

There are two types of expiry relating to accounts: password expiry and account expiry. When the password is expired, the application forces the user to change it during his next log in after he has successfully authenticated using his current password. When an account is expired, it is no longer usable, and the user cannot log in using the account even if he knows the correct password for the account. Most applications differentiate between password expiry and account expiry.

expire password	Expires an account's password.
unexpire password	Unexpires an account's password.
check password expiry	Checks if an account's password is expired.
check account expiry	Checks if an account is expired.
expire account	Expires an account.
unexpire account	Unexpires an account.

C.5 Listing operations

list accounts	Lists accounts on the target system, and the attributes for each account.
list attributes	Lists selected attributes for multiple accounts on a target system. Some target systems support incremental listing of attributes. That is, only users with attribute changes since the last listing are returned.
list groups	Lists groups.
list members	Lists members of a managed group.
list account attributes	Lists attributes for a specified account.
list computer objects	Lists computer objects on a target system.
list subscribers	Lists subscriber accounts on a target system.

C.6 Account creation / deletion operations

The typical behavior of all built-in connectors is to copy a template account, and then update the attributes of the copy.

create account	Creates a new account on the target system. This operation creates the account (possibly using a template for some attribute values), then sets other attribute values – including the password for the new account.
delete account	Deletes an existing account on the target system. The typical behavior is to first ensure that the account being deleted exists.

C.7 Account modification operations

update attributes	Updates attributes for an existing account.
add user to group	Adds an account to a group.
delete user from group	Removes an account from a group.
move contexts	Moves an account to a new context or location on a context-sensitive target.
rename account	Renames an existing account's short ID. By default, this operation is *not* supported from the Bravura Security Fabric GUI.

C.8 Group provisioning operations

create group	Creates the specified group. Currently, group creation does not support the setting of any attributes. This operation is *not* supported from the Bravura Security Fabric GUI.
delete group	Deletes the specified group. This operation is *not* supported from the Bravura Security Fabric GUI.
add group to group	Adds a group to a group. Currently, adding group to group does not support the setting of any attributes. This operation is *not* supported from the Bravura Security Fabric GUI.
remove group from group	Removes a group from a group. Currently, deleting group from group does not support the setting of any attributes. This operation is *not* supported from the Bravura Security Fabric GUI.

C.9 Command execution operation

run command This operation executes a command or script via the checked out multiple accounts request.

C.10 Connector operations by application

Table C.1 details which connector operations are supported by each Bravura Security Fabric application. Some operations are not accessible from the GUI.

Table C.1: *Bravura Security Fabric* connector operations by application

Operation	*Bravura Pass*	*Bravura Identity*	*Bravura Privilege*	*Bravura Group*	*Access Certifier*	*Org Manager*
Password verification operations						
user verify password	x	x	x	x	x	x
administrator verify password	x	x	x		x	x
get server information	x	x	x	x	x	x
user change password	x					
administrator reset password	x		x			
administrator reset+expire password	x		x			
verify+reset password	x		x			
reset hard drive encryption password	x					
resynchronize tokens	x					
set token pin	x					
challenge response authentication	x	x	x	x	x	x
generate an unlock code to recover control of a machine after reboot	x					
generate an unlock code to recover control of a machine after reboot and set new password	x					
generate an unlock code to recover control of a machine from the unlock screen	x					
update subscriber attributes/password			x			

. . . continued on next page

Table C.1: *Bravura Security Fabric* connector operations by application (Continued)

Operation	*Bravura Pass*	*Bravura Identity*	*Bravura Privilege*	*Bravura Group*	*Access Certifier*	*Org Manager*
Account enable/disable operations						
check account enabled		x			x	x
enable account		x			x	x
disable account		x			x	x
Intruder lock-out operations						
check account lock		x			x	x
lock account		x			x	x
unlock account	x	x			x	x
lock account + enter emergency access mode	x					
Expiry operations						
expire password	x					
unexpire password	x					
check password expiry	x					
check account expiry		x			x	x
expire account		x			x	x
unexpire account		x			x	x
Listing operations						
list accounts	x	x	x	x	x	x
list attributes	x	x	x	x	x	x
list groups	x	x	x	x	x	x
list members	x	x	x	x	x	x
list account attributes	x	x	x	x	x	x
list computer objects			x			
list subscribers			x			
Account creation / deletion operations						
create account		x				x
delete account		x			x	x
Account modification operations						
update attributes		x			x	x
add user to group		x		x	x	x
delete user from group		x		x	x	x

. . . continued on next page

Table C.1: *Bravura Security Fabric* connector operations by application (Continued)

Operation	*Bravura Pass*	*Bravura Identity*	*Bravura Privilege*	*Bravura Group*	*Access Certifier*	*Org Manager*
move contexts		x			x	x
rename account		x			x	x
Group provisioning operations						
create group		x		x	x	x
delete group		x		x	x	x
run command						

Management Suite Operation Codes

D

Operation codes are logged in **`sesslog`**, and are used in generating event reports.

> **Note:** Some codes may not apply to your licensed products.

Table D.1: Operation codes

Code	Description
AAUT	A help desk user authenticates a user by typing answers to that user's personal questions.
ACEX	A help desk user changes and expires a password from the *Help users* (IDA) module.
ACHG	The Password Manager service (idpm) or Transaction Monitor Service (idtm) changes a password for an account. The password change can originate from the *Help users* (IDA) module, *Change passwords* (PSS) module, API Service (idapi) or Hitachi ID Password Change Notification Module.
ACPS	A product administrator configures *Bravura Security Fabric*.
ACRF	Failed to import a discovered system (administrator credentials could not be established).
ACRS	Imported a discovered system (administrator credentials were set successfully).
ACUA	An account is created on a target system and attached to a user.
ADAT	A connector queries target system address attributes/formats.
ADIS	A product administrator disables a user.
ADMG	A product administrator adds, deletes or modifies an Administrator group.
AEDA	Exception approval added to a user for a deficit.
AEDD	Exception approval removed from a user for a deficit.
AEPA	Exception approval added to a user for a surplus.
AEPD	Exception approval removed from a user for a surplus.
AENA	A product administrator enables a user.
AESA	Exception approval added to a user for a segregation of duties rule.
AESD	Exception approval removed from a user for a segregation of duties rule.

. . . continued on next page

Table D.1: Operation codes (Continued)

Code	Description
AEXA	The Transaction Monitor Service runs a connector to expire an account on a target system.
AEXP	The Transaction Monitor Service runs a connector to expire a password for an account on a target system, without changing it.
AFND	A help desk user finds a particular user in the *Help users* (IDA) module.
AHLP	A product administrator adds, deletes, or modifies another product administrator.
ALGN	A user accesses the *Help users* (IDA) module.
AMPS	A product administrator manages *Bravura Security Fabric*.
APCH	A help desk user changes own password.
ARPS	A product administrator accesses the maintenance menu for *Bravura Security Fabric*.
AULK	A help desk user clears the intruder lock-out on a group of targets for a user selected from the *Help users* (IDA) module.
AVER	The Transaction Monitor Service runs connectors to login as the target administrator and verify a password on an account.
CHKA	A connector checks if an account is expired.
CHKE	A connector checks if user is enabled.
CHKL	A connector checks if user is locked.
CHKP	A connector checks if password is expired.
CHNG	User change.
CHRP	Challenge-response (encrypted hard disks, RADIUS, RSA).
CRTG	*Bravura Identity* creates a group on a target system.
DBCF	Replication connection failure.
DBEL	Manage external database login.
DBCM	Resuming commits since replication backlog is being processed.
DBCS	Suspending commits due to replication issues.
DBCR	Replication connection restored.
DBFP	Failed database call has been recorded.
DBQF	Replication queue failure.
DBTF	Replication transmission failure.
DBWW	Replication water mark reached.
DELG	*Bravura Identity* deletes a group on a target system.
DELR	*Bravura Identity* removes a role from a user.
DELU	*Bravura Identity* deletes an account on a target system.
DLGN	Dashboard login.
DNAP	A *Bravura Security Fabric* profile is disabled.

. . . continued on next page

Table D.1: Operation codes (Continued)

Code	Description
DNAU	*Bravura Identity* disables an account on a target system.
ENAP	A *Bravura Security Fabric* profile is enabled.
ENAU	*Bravura Identity* enables an account on a target system.
FIDT	An incoming SAML authentication request is successfully parsed by the system.
FATH	An outgoing SAML assertion response is successfully generated by the system.
FATN	An outgoing authentication request was sent to the external identity provider.
FASR	An incoming SAML assertion was received from the identity provider.
GOGA	*Bravura Security Fabric* adds an owner group.
GOGD	*Bravura Security Fabric* deletes an owner group.
GOOD	*Bravura Security Fabric* removes an unknown owner from a group.
GRBD	*Bravura Security Fabric* removes an unknown object from a group.
GROA	*Bravura Identity* adds an owner to a group on a target system.
GROD	*Bravura Identity* removes an owner from a group on a target system.
GRUA	*Bravura Identity* adds an account to a group on a target system.
GRUD	*Bravura Identity* removes an account from a group on a target system.
GRGA	*Bravura Identity* adds a group to a group on a target system.
GRGD	*Bravura Identity* removes a group from a group on a target system.
GRPA	*Bravura Security Fabric* adds a parent group.
GRPD	*Bravura Security Fabric* deletes a parent group.
GRUP	*Bravura Security Fabric* updates a group.
GSAD	A group set is added in *Bravura Privilege*.
GSDE	A group set is removed in *Bravura Privilege*.
GSCI	Temporary group membership checked in.
GSCO	Temporary group membership checked out.
GSUP	A group set is updated in *Bravura Privilege*.
HDIS	A product administrator disables another product administrator.
HENA	A product administrator enables another product administrator.
HULK	A product administrator unlocks another product administrator.
HUPD	A target system is added, deleted, or modified using the *Manage the system* (PSA) module.
IDAL	A user accesses the *Help users* (IDA) module.
IDDL	Digital ID file is downloaded using the *Digital ID* (DID) module.
IDGL	A user accesses the *Manage the OrgChart* (IDG) module.
IDOL	A user accesses the *Browse the OrgChart* (IDO) module.
IDPL	A user accesses the *Requests* app.

. . . continued on next page

Table D.1: Operation codes (Continued)

Code	Description
IDRG	Digital ID file is registered using the *Digital ID* (DID) module.
IDRL	A user accesses the *View and update profile* (IDR) module.
IDSL	A user accesses the *Manage delegations* (IDS) module.
IDUP	Digital ID file is uploaded using the *Digital ID* (DID) module.
IDVL	A user accesses the *Manage inventory* (IDV) module.
ILGN	A user logs into the *Generate voice print enrollment PIN* (PSI) module.
IMAT	Discovered account imported.
IMHF	A system fails to be added to the list of managed systems.
IMHS	A system is successfully added to the list of managed systems.
IMHT	Discovered computer imported.
IMPA	An implementer accepts an implementer task.
IMPD	An implementer declines an implementer task.
IMPF	An implementer reports failure to complete a task.
IMPS	An implementer reports successful completion of an implementer task.
INDL	An INF file is downloaded from *Bravura Privilege* to install workstation components.
IPLK	Remote IP locked.
IPUL	Remote IP unlocked.
ISID	ID selected for IVR registration.
KLGN	A user logs into the *Unlock accounts* (PSK) module.
LATT	A connector lists account attributes on a target system.
LCKA	A help desk user is prevented from logging into all CGI modules.
LCKU	A user is prevented from logging into all CGI modules.
LDEL	An alternate login ID is detached from a profile using the *Attach other accounts* (PSL) module or *Help users* (IDA) module.
LDUP	An attempt to add an already-attached login ID to a profile is made using the *Attach other accounts* (PSL) module or *Help users* (IDA) module.
LLGN	A user logs into the *Attach other accounts* (PSL) module
LOCK	The Transaction Monitor Service runs a connector to lock an account on a target system.
LUPD	An alternate login ID is updated in the *Attach other accounts* (PSL) module or *Help users* (IDA) module.
LVER	*Bravura Pass* verifies the login ID's password when a user attempts to add an alternate Login ID to their profile using the *Attach other accounts* (PSL) module.
MADD	A member is added to a role.
MAVW	Model after - profile viewed.
MAED	Model after - request submitted.

. . . continued on next page

Table D.1: Operation codes (Continued)

Code	Description
MDEL	A member is deleted from a role.
MSDL	An MSI file is downloaded from *Bravura Privilege* or *Login Manager* to install workstation components.
MSUD	A managed system policy's service ID is changed in *Bravura Privilege*.
MUPD	A role member is updated.
MVCU	*Bravura Identity* moves an account from one context to another on a target system.
NADD	A notification is added in the *Manage the system* (PSA) module.
NDEL	A notification is deleted in the *Manage the system* (PSA) module.
NEWS	A new session ID is generated within the same CGI session by starting a new operation; for example, a password change or account unlock.
NLGN	A user accesses the *User notifications* (PSN) module via a notification message.
NOOP	*Bravura Security Fabric* runs a NULL operation.
NPCA	A notification pre-condition is added in the *Manage the system* (PSA) module.
NPCD	A notification pre-condition is deleted in the *Manage the system* (PSA) module.
NPCU	A notification pre-condition is updated in the *Manage the system* (PSA) module.
NRCR	A network resource is created.
NRDL	A network resource is deleted.
NRUP	A network resource is updated.
NRMV	A network resource is moved.
NUPD	A notification is updated in the *Manage the system* (PSA) module.
00BA	Out-of-band addition.
00BD	Out-of-band deletion.
OPC1	Custom operation 1.
OPC2	Custom operation 2.
OPC3	Custom operation 3.
OPC4	Custom operation 4.
PDEL	A queued event in the Password Manager service is deleted by a newer request for the same account, before it is executed by a connector.
PEMF	A token's emergency access mode is turned off using the *Manage tokens* (PSP) module or *Help users* (IDA) module.
PEMN	A token's emergency access mode is turned on using the *Manage tokens* (PSP) module or *Help users* (IDA) module.
PINC	A token's pin is cleared through the *Manage tokens* (PSP) module or *Help users* (IDA) module.
PIND	A token is disabled through the *Manage tokens* (PSP) module or *Help users* (IDA) module.

. . . continued on next page

Table D.1: Operation codes (Continued)

Code	Description
PINE	A token is enabled through the *Manage tokens* (PSP) module or *Help users* (IDA) module.
PINR	A token is resynchronized through the *Manage tokens* (PSP) module or *Help users* (IDA) module.
PINS	A token's pin is set through the *Manage tokens* (PSP) module or *Help users* (IDA) module.
PINU	Unlock a user token using the *Manage tokens* (PSP) module or *Help users* (IDA) module.
PIQU	A request is queued in the Transaction Monitor Service.
PLGN	A user logs into the *Manage tokens* (PSP) module.
PMCI	Generic access checked in using *Bravura Privilege*.
PMCO	Generic access checked out using *Bravura Privilege*.
PMIP	Generic access partially checked in using *Bravura Privilege*.
PMOP	Generic access partially checked in using *Bravura Privilege*.
PPCK	The Password Manager service validates password strength.
PPQU	A password is queued for change in the Password Manager service.
PSTR	Password strength rules changed in the *Manage the system* (PSA) module or *Request privileged access* (PSW) module.
PWAR	Access to a password on a managed system in *Bravura Privilege* is checked in by another user.
PWCI	Managed system password checked in using *Bravura Privilege*.
PWCO	Managed system password checked out using *Bravura Privilege*.
QADD	A user adds a new security question in the *Update security questions* (PSQ) module or *Help users* (IDA) module.
QDEL	A user deletes a question in the *Update security questions* (PSQ) module or *Help users* (IDA) module.
QDON	A user's security questions profile is completed.
QLGN	A user logs into the *Update security questions* (PSQ) module.
QSEE	A user views security questions in the *Update security questions* (PSQ) module.
QUPD	A user updates an existing question in the *Update security questions* (PSQ) module or *Help users* (IDA) module.
RCMD	Execute command or script via checked out multiple accounts request in *Request privileged access* (PSW) module
RENU	*Bravura Identity* renames an account on a target system.
RPRO	A profile ID is renamed within Bravura Security Fabric.
RLGN	A user logs into the *Enable password synchronization* (PSR) module.

. . . continued on next page

Table D.1: Operation codes (Continued)

Code	Description
RLUA	The *Bravura Identity* request system (Workflow Manager Service (idwfm) or *View and update profile* (IDR) module) is used to a role to a user.
ROAA	An authorizer is added to a role
ROAD	An authorizer is removed from a role
RPTR	A report program is executed.
RREG	A user registers self for transparent password synchronization in the *Enable password synchronization* (PSR) module.
RSYN	A token is re-synchronized using the *Manage tokens* (PSP) module or *Help users* (IDA) module.
RTRB	Import rule trial run begins.
RTRC	Import rule trial run canceled.
RTRF	Import rule trial run finished.
RUNR	A user unregisters themselves for transparent password synchronization in the *Enable password synchronization* (PSR) module.
RVGS	A group set is requested using *Bravura Privilege*.
RVPW	*Bravura Privilege* receives a request to access an account on a managed system.
SANS	Managed account subscriber manually imported.
SERI	A connector queries target system server information/status.
SLGN	A user logs into the *Change passwords* (PSS) module.
SMON	A user accesses the *Session monitor* app.
SOAA	An authorizer is added to a segregation of duties rule
SOAD	An authorizer is removed from a segregation of duties rule
SRAD	A recorded data censorship rule is added in *Bravura Privilege*.
SRDL	A recorded data censorship rule is deleted from *Bravura Privilege*.
SRES	A user changes own password on a group of targets selected from the *Change passwords* (PSS) module.
SRUD	A recorded data censorship rule is updated in *Bravura Privilege*.
SSOC	A SAML single sign-on session is successfully created.
SSOD	A SAML single sign-on session is successfully terminated.
SULK	A user clears the intruder lock-out on a group of targets selected from the *Unlock accounts* (PSK) module.
TAAA	An authorizer is added to a target
TAAD	An authorizer is removed from a target
TEAA	An authorizer is added to a template
TEAD	An authorizer is removed from a template

. . . continued on next page

Table D.1: Operation codes (Continued)

Code	Description
UEXA	The Transaction Monitor Service runs connectors to un-expire a target system account.
UEXP	The Transaction Monitor Service runs connectors to un-expire a password for a target system account.
ULCK	The Transaction Monitor Service or Password Manager service runs connectors to clear the intruder lock-out for a target system account.
ULKU	A user is unlocked.
ULGN	A user logs into a CGI module or the API Service.
UMAT	Discovered account is unmanaged.
UMHT	Discovered computer is unmanaged.
UPAT	The Workflow Manager Service is used to update profile and request attributes.
UPDT	*Bravura Identity* updates an account's attributes on a target system.
UPRS	A resource is updated.
USUP	A USERSTAT record is updated.
USSR	Userstat search.
UXDL	A UNIX key file is downloaded from *Bravura Privilege* to install UNIX workstation components. (Not supported in *Bravura Security Fabric* 8.2.x)
VERI	The Transaction Monitor Service or Password Manager service runs a connector to verify a password for a target system account.
VRRE	The Transaction Monitor Service or Password Manager service runs a connector to verify a password for a target system account. If the connector fails to verify the password, the connector resets the password to the specified value.
VWUH	A user's profile history is viewed.
WAAP	An access disclosure plugin is added in *Bravura Privilege*.
WAGP	A managed system policy is added to database in *Bravura Privilege*.
WAPW	A new password is added to a manually managed system in *Bravura Privilege*.
WAWS	A managed system is added to *Bravura Privilege*.
WCFT	Workstation ID conflict detected.
WCPW	A password is set on a managed system in *Bravura Privilege*.
WDAP	An access disclosure plugin is deleted from *Bravura Privilege*.
WDGP	A managed system policy is deleted from *Bravura Privilege*.
WDUP	A managed system account is deleted from *Bravura Privilege*.
WDWS	A resource is deleted from the database.
WLGN	A user accesses the *Privileged access* app.
WOPW	A password on a managed system is overridden using the *Request privileged access* (PSW) module.
WRAP	A password to a managed system is randomized using *Bravura Privilege*.

. . . continued on next page

Table D.1: Operation codes (Continued)

Code	Description
WREI	Workstation reinstalled.
WRPW	A new password is requested on a managed system in *Bravura Privilege*.
WUAP	An access disclosure plugin is updated in *Bravura Privilege*.
WUGP	A managed system policy is updated in *Bravura Privilege*.
WUWS	A managed system is updated in *Bravura Privilege*.
WVPH	A *Bravura Privilege* user views a historical (not current) password on a managed system.
WVPW	A *Bravura Privilege* user views a password on a managed system.
WWPW	*Bravura Privilege* records a password to a managed system in the database.

Troubleshooting E

E.1 Using cmd.exe with system()

The system() function must call an executable and not a script; however, it can call the command line interpreter cmd.exe in non-interactive mode with the /c option; in this case, the script to be run and the arguments to be passed to that script are also included as part of the statement. For example:

```
system( "cmd.exe", "/c", "script.bat", "param1", "param2", "param3" )
```

However, due to the way that cmd.exe parses quotation marks, this statement will fail. The system() function will call cmd.exe with the following argument:

```
"cmd.exe" "/c" "script.bat" "param1" "param2" "param3"
```

and then according to the parsing rules, cmd.exe will strip the first and last quotation marks leaving:

```
cmd.exe" "/c" "script.bat" "param1" "param2" "param3
```

This causes a problem when script.bat tries to use param3 and expects proper quote pairing. Any program that expects matched quotes will fail unexpectedly when dealing with the last parameter.

The easiest way to avoid this problem is to append a dummy parameter, "", to the end of the argument list, and never use it. This method ensures that all the necessary parameters are properly quoted. The following MTIL example highlights how to append the extra parameter.

In MTIL, a system() command with the appended "" parameter would look like:

```
system( "cmd.exe", "/c", "script.bat", "param1", "param2", "param3", "" )
```

Note that this problem does not exist with PSLANG, because it can execute batch files directly. There is no need to call `cmd /c batch.bat`, which is necessary for MTIL. For example, the following works in PSLANG:

```
function main()
{
    var $cmd = "test.bat";
    var $arg[];
    $arg[0] = "param1";
    $arg[1] = "param2";
    $arg[2] = "param3";
    var $retval = 0;
    var $retcode = 0;
    var $stdoutStr;
    $retcode = system( $cmd, $arg, "", $stdoutStr, $retval, 5 );
}
```

If test.bat contains:

```
echo %* > test.txt
```

then test.txt will contain:

```
"param1" "param2" "param3"
```

For more information on how cmd.exe parses quotation marks, run `cmd.exe /?` from the command line or search the included help system.

File Locations F

There are three main directories that are created when you install *Bravura Security Fabric* instance:

- <*Program Files path*>\Hitachi ID\IDM Suite\<*instance*>\
- <*Program Files path*>\Hitachi ID\IDM Suite\Logs\<*instance*>\
- <*Program Files path*>\Hitachi ID\IDM Suite\Locks\

When you install *Bravura Security Connector Pack*, files are placed in different locations depending on the type of *Connector Pack*.

For an instance-specific connector pack, the installer, **`connector-pack-x64.msi`**, installs agent connectors in:

<*Program Files path*>\Hitachi ID\IDM Suite\<*instance*>\agent\.

For a global connector pack, the installer, **`connector-pack-x64.msi`**, installs connectors and supporting files in:

- <*Program Files path*>\Hitachi ID\Connector Packs\global\agent\.

See "**??**" in the *Bravura Security Fabric* Documentation for more detail.

Platform Type ID Values G

Several API functions accept and/or return platform type values. These values are abbreviated ID codes rather than complete names of platform types. The following is a list of possible ID values and their associated platform types.

ID	platform type
ACF2	ACF2 (with Mainframe Connector)
AD	Active Directory DN
AS400	IBM OS/400 Server
AS400SCRIPT	IBM OS/400 Server (Script)
AWS	Amazon Web Services
BES	BlackBerry Enterprise Server
BESWS	BlackBerry Enterprise Web Service
BITLOCKER	Bitlocker Hard Drive Encryption
CACHE	Intersystems Cache/Server
CACHESCRIPT	Intersystems Cache/Server (Script)
CIFS	SMB Protocol for Active Directory DN
CONCUR	Concur
CSV	CSV File Connector
CLEARTRUST	RSA Access Manager
DB2	DB2 Database
DB2SCRIPT	DB2 Database (Script)
DOMINO	Lotus Domino Server
DOSKEY	Win32 Console Script
EXG2K7_64BIT	Exchange 2007+ Server
GAPPS	Google Applications
GDOMINO	Lotus Domino Server (Script)
GRPWISE	Groupwise Domain
HEAT	FrontRange HEAT (Customers)
HEATADMIN	FrontRange HEAT (Administrators)
HITRACK	Hitachi Data Systems Hi-Track Monitor
HPSMWS	HP Service Manager, Web Service
HSS	Oracle Hyperion EPM Shared Services

ID	platform type
IDM	Bravura Security Suite
ILEARN	Oracle iLearning
JDEOW_80	JD Edwards EnterpriseOne 8.0
JDEOW_81	JD Edwards EnterpriseOne 8.1
JDEOW_90	JD Edwards EnterpriseOne 9.0
LDAP	LDAP Directory Service
LYNC	Microsoft Lync
NDS	Novell eDirectory
NETPOINT	Oracle Access Manager
NRSHRPT	SharePoint Resource
ODBCSCRIPT	ODBC Database (Script)
ODBCSCRIPT32	ODBC Database (32-bit) (Script)
OFFICE365	Microsoft Office365 Online Service
OPAN	Hitachi IT Operations Analyzer
ORACLE	Oracle Database
ORACLESCRIPT	Oracle Database (Script)
PEOPLESOFT810	PeopleSoft Application Server (PeopleTools 8.1x)
PEOPLESOFT820	PeopleSoft Application Server (PeopleTools 8.20 - 8.48)
PEOPLESOFT849	PeopleSoft Application Server (PeopleTools v8.49+)
PEOPLESOFTHR	Peoplesoft Human Capital Management
PWRSHELL	PowerShell Script
PYTHON	Python Script
RACF	RACF (with Mainframe Connector)
REMEDY	Remedy Action Request System
REMEDY_ITSM	Remedy Action Request System IT Service Manager
RSAAM	RSA Authentication Manager 7.1/8.x
SALESFORCE	Salesforce
SAP	SAP Server
SHRPT	SharePoint Server
SITEMINDER	Netegrity SiteMinder
SMB	SMB Protocol
SOAP	SOAP Web Service
SQL	Microsoft SQL Server
SQLSCRIPT	Microsoft SQL Server (Script)
SSH	SSHD Host target system
SUCCESSFACTORS	SuccessFactors system
SVCNOW	ServiceNow IT Service Management

ID	platform type
SYBASECT	Sybase AES Database
SYBASECTSCRIPT	Sybase AES Database (Script)
TAM	Tivoli Access Manager
TAMSSO	Tivoli Access Manager for Enterprise SSO
TELNET	Telnet target system
TOPSECRET	TopSecret (with Mainframe Connector)
UNIX	Unix target system
VASCO	Vasco IDENTIKEY Server
WEBEX	WebEx
WEBEXCONN	WebEx Connect
WIN2K	Active Directory
WINNT	Windows NT Server
XML	XML-RPC Web Service

ResourceFind Search Criteria

H

ResourceFindCriteria (p269) returns the search criteria for each resource type that can be found with ResourceFind (p267). See Searching for resources with ResourceFind (p142) for an explanation of how to use the criteria with ResourceFind.

H.1 ACCTATTR (Account Attribute) Resource Type

This resource type accepts platform types as one of its search criteria. See Platform Type ID Values for the list of possible values.

name	description	type
id	Account attribute ID	string
override	Override level; 0 = default, 1 = platform type, 2 = target system	integer
platformid	Platform type	string
targetid	Target ID	string
createaction	Action to perform when creating account; C = exact copy, I = ignore, R = copy, replacing ID, S = set	string
updateaction	Action to perform when updating account; S = set always, C = set when mapped profile attribute changes, I = ignore	string
seqno	Order to set this attribute	integer
encoding	Encoding to use; N = no encoding, B = base 64	string
attrtype	attribute.datatype; I = binary, B = boolean, F = memo, N = integer, S = string	string
minvalues	Minimum number of values required	integer
maxvalues	Maximum number of values allowed	integer
userattr	Name of profile attribute to which the value should be set	string
listattr	Do we list this attribute	boolean
setuserattr	Will this target attribute be used to set profile attribute values during auto-discovery	boolean
makediffs	Should iddiscover produce diffs on this attribute	boolean

H.2 ATTR (Profile/Request Attribute) Resource Type

name	description	type
attrkey	Global attribute ID	string
description	Description of attribute	string
type	Type of value; B = boolean, C = string, D = datetime, L = image link, N = integer, M = memo, P = password, U = user, F = file	string
length	Field length	integer
notes	Help text for more attribute information	string
dispformat	Displayed format string	string
format	Format string	string
regexp	Regular expression to validate user input	string
nonevalue	Display value for no value of a boolean data type attribute	string
truevalue	Display value for true value of a boolean data type attribute	string
falsevalue	Display value for false value of a boolean data type attribute	string
minrequire	Minimum number of required values; 0 = optional	integer
maxallowed	Maximum number of allowed values; -1 = infinite	integer
dupallowed	Is this attribute allowed to have duplicate values	boolean
sendable	Allow values of this attribute to be sent in emails	boolean
allowcopy	Is this attribute used for profile comparison	boolean
invalidateauth	Invalidate authorizations if the attribute value is changed	boolean
defaultval	Are there default value(s) for the attribute	boolean
fixedval	Are there restricted values for the attribute	boolean
valplugin	Plugin to run to generate a list of restricted values	string
verify	User needs to re-type attribute value for confirmation	boolean
attrtype	The scope of the attribute; P = Profile, R = Request, I = Request item	string
makediffs	Should iddiscover produce diffs on this attribute, for use with idtrack	boolean
enforceinherit	Inherit validation enforcement from the attribute group to which the attribute belongs	boolean
enforcecreate	Enforce validation when creating accounts	boolean
enforceupdate	Enforce validation when updating accounts; true when always enforcing or enforcing when updating this attribute's values	boolean
enforceother	Enforce validation when updating accounts; true when always enforcing	boolean

name	description	type
displayattr	Display refinement; for boolean data type, C=checkbox, A=radio buttons; for date/time data type, D=date only (calendar); for string or integer data type, B=drop-down, E=drop-down with two columns; for user data type, V=allow invalid; for file or link data types, I=image, T=validate	string
order	How to order the restricted attribute values; A = Actual value, D = Displayed value	string
advsearchable	Is this attribute available for advanced search?	boolean
onreport	Is attribute shown on reports?	boolean
vcardprop	vCard property to which this attribute is mapped	string
authattribute	Is this attribute allowed to be used for authentication	boolean
parentattr	Parent attribute key	string
keepcase	Do entered restricted values keep their case	boolean
submitonly	Is this attribute editable only request creation	boolean

H.3 ATTRGRP (Attribute Group) Resource Type

name	description	type
attrgroupid	Attribute group ID	string
desc	Attribute group description	string
display	How to display the attributes of this group on requests; M = Main page, S = Subsidiary, N = None	string
displayorder	Relative display order of attribute	integer
enforcecreate	Enforce validation during account creation	boolean
enforceupdate	Enforce validation during account updates; true if enforcing always or only when attributes in group change	boolean
enforceother	Enforce validation during account updates; true if enforcing always	boolean
notesabove	Notes above attribute	string
notesbelow	Notes below attribute	string

H.4 MACC (Managed account) Resource Type

name	description	type
policyid	ID of the policy this account is on	string
validity	1 = only valid accounts, 0 = only invalid accounts, -1 = both valid and invalid accounts	integer
isadmin	Is administrator	boolean
workstnid	workstation netbios ID	string
accountname	The name of the account	string

H.5 MGRP (Managed Group) Resource Type

name	description	type
targetid	Target system ID	string
invalidOnly		boolean
issecurity		boolean
grouptype	A target specific string representing the type of group	string
invalidDays		integer
unusedOnly		boolean
nosgroupname	Full ID of this group on this system	string
shortid	Common name of this group on this system	string
notes	Descriptive text about the group	string
location	Which location	string
type	Which type	string
ranking	Popularity ranking	integer
url	URL with group information	string
ownerasauth		boolean
overrideimpl	Override target implementation setting	string
oobdiffsadd	Should idtrack detect additions to managed group and request removal	string
oobdiffsdel	Should idtrack detect deletions from managed group and request addition	string

name	description	type
oobdelay	Flag to delay processing additions to managed group until after users are loaded into the xgrpmbr table. i.e. don't do the oob check the first time to avoid detecting all users as being out of band additions	boolean
oobemailplugin	Plugin to run to generate a list of email recipients for out of band changes to the Managed group	string
rbacenforce	Enforce managed group under RBAC roles	boolean
daction	Default action for deficit of this resource; ADD = Add the resource, EXCEPTION = Request an exception, INHERIT = Inherit default,REMOVE = Remove the resource	string
saction	Default action for surplus of this resource; ADD = Add the resource, EXCEPTION = Request an exception, INHERIT = Inherit default,REMOVE = Remove the resource	string
description	Descriptive name of this group	string
invalid		boolean
automanaged	Read only field indicating whether the group is auto managed by Bravura Security Fabric or not	boolean

H.6 NETRES (Network Resource) Resource Type

name	description	type
type	Network resource type	string
hostid	Target system ID	string

H.7 ROLE (Role) Resource Type

name	description	type
roleid	Role ID	string
enabledOnly	Whether only enabled roles should be returned	boolean
desc	Role description	string
assignable	Whether this role is assignable to a user. This would be false for roles that are used solely as sub-roles	boolean
enabled	While a role is in the midst of being defined by the role miner it should not be displayed to users or otherwise affect anything	boolean
deprecatedreason	Reason for deprecation	string
rbacenforce	Enforce role under RBAC roles?	boolean
daction	Default action for deficit of this resource: ADD, EXCEPTION, INHERIT, or REMOVE	string

H.8 TARG (Target System) Resource Type

name	description	type
id	Unique Identifier for this target system	string
inventory		boolean
platform	Platform of target system	string
proxyOnly		boolean
isapp		boolean
group	Target system group this target belongs to	string
itsm		boolean
type	Type of target system: M = Manual, A = Discovered, T = Template	string
description	Description of target system	string
address	Network address for this target system	string
url	URL with target information for users to understand which targets they could reset passwords/modify login IDs on	string
include	Include IDs if no IDInclude rule applies	boolean
casegen	Default or external program to generate the new ID in correct cases (all upper, all lower, etc.)	string
listgen	External program to convert this target system into a list of other target systems	string

name	description	type
mandatory	Users must have at least one account on this target	boolean
agenttime	Agent timeout, in seconds	integer
listtime	List timeout, in seconds	integer
lstminsize	Minimum list file size, in bytes	integer
groupownerphase	The phase group owner should be added to when being automatically as authorizer	integer
proxy	Proxies	string
persistentserver	Server to use for persistent list	string
persistentport	port to user for persistent list	string
claimalias		boolean
disableverify		boolean
multiowner	Target system supports multiple owners on groups	boolean
isapp		boolean
selfmanagehide		boolean
idarchivepush		boolean
idcheck		boolean
groupinherit		string
implinherit		string
contattr	Attribute holding the value for _container_dn for this host	string
expusr	Find password-soon-to-expire users during auto discovery	boolean
delalias	Specify if users are allow to delete accounts on this target; X = DEFAULT, T = ALLOW, F = BLOCK	string
adminresethide		boolean
disablereset		boolean
selfresethide		boolean
disableunlock		boolean
adminunlockhide		boolean
selfunlockhide		boolean
adminclaimhide		boolean
selfclaimhide		boolean
mustselect		boolean
updserver	Server ID that will update this target	string
managegrp	Auto-manage groups; N = None, M = Moderated by owners, m = Moderated	string
mgrpownerasauth	Automatically add owners as authorizers	boolean
nrplatform	Type of network resource	string
rbacenforce	Enforce target under RBAC roles	boolean

name	description	type
daction	Default action for deficit of this resource; ADD = Add the resource, EXCEPTION = Request an exception, INHERIT = Inherit default,REMOVE = Remove the resource	string
saction	Default action for surplus of this resource; ADD = Add the resource, EXCEPTION = Request an exception, INHERIT = Inherit default,REMOVE = Remove the resource	string
oneaccount	When a user requests a template on this system and already has one existing account, assume that the template request targets that existing account.	boolean
enable	Whether target supports ENABLE capability.	boolean
incremental	Control target's listing (generate full list every time vs incremental)	boolean
persistentlist	Controls if the target has persistent list enabled.	boolean
copyfrom	Targets that current target will copy list data from for its own list data	string

H.9 TMPL (Template Account) Resource Type

name	description	type
targetid	Target system ID	string
inventoryonly	If inventoryonly is -1, then don't search on that field; otherwise, 1/0 = template is/isn't used for inventory provisioning	integer
id	Unique identifier of template	string
description	Description for template	string
inventory	Template is used for inventory provisioning	boolean
location	Location used to help find a suitable template	string
type	Type used to help find a suitable template	string
platform	Type of target system	string
accountname	Long format (e.g. for NDS, LDAP) login ID of the account that would be replicated when creating new IDs	string
blockbatch	Denial of this template blocks entire batch	boolean
parameter	Extra parameters to pass to our agent, which it might use to create new IDs using this template which are not-quite-identical to the template user	string
deleted	Marked as deleted	boolean
requirepwd	Password required or not when creating new users	boolean
native	The template is native template	boolean

name	description	type
overridetarget	Override target authorization setting	string
overrideimpl	Override target implementation setting	string
usercompare	Is the template used for user comparison	boolean

H.10 UC (User class) Resource Type

name	description	type
id	Unique ID of class	string
desc	Description given to this class	string
builtin	Whether this class is built-in (cannot be modified)	boolean
source	User class point that this userclass is created for	string
reason	The reason for deprecating	string

H.11 UGRP (User group) Resource Type

name	description	type
id	Unique ID of user group	string
pointid	ID of user class point associated with group	string
desc	description of user group	string
reset	'Change passwords' privilege	boolean
resetexpire	'Change and expire passwords' privilege	boolean
hdd	'Unlock encrypted systems/accounts' privilege	boolean
unlock	'Unlock accounts' privilege	boolean
profileenable	'Enable/Disable user profile' privilege	boolean
profileunlock	'Unlock user profile' privilege	boolean
view	'View profile information' privilege	boolean
viewacct	'View accounts' privilege	boolean
viewrole	'View roles' privilege	boolean

name	description	type
update	'Update account' privilege	boolean
addrole	'Add roles' privilege	boolean
delrole	'Delete roles' privilege	boolean
create	'Create account' privilege	boolean
enable	'Enable account' privilege	boolean
disable	'Disable account' privilege	boolean
deleteacct	'Delete account' privilege	boolean
managegroups	'Manage group memberships' privilege	boolean
ppmreqrevoke	'Revoke privileged access requests' privilege	boolean
rename	'Rename account' privilege	boolean
renameprofile	'Rename profile' privilege	boolean
movectx	'Move user from one context to another' privilege	boolean
viewhistory	'View profile history' privilege	boolean
delegmang	'Delegate workflow requests' privilege	boolean
flowmang	'Manage workflow requests' privilege	boolean
deleg	'Delegate authority' privilege	boolean
editqa	'Update security questions' privilege	boolean
viewqa	'View security questions' privilege	boolean
bypassqa	'Bypass security questions' privilege	boolean
editids	'Attach other accounts' privilege	boolean
netres	'Request access to network resources' privilege	boolean
vieworg	'Browse the OrgChart' privilege	boolean
manageorg	'View my subordinates' privilege	boolean
pwdaccess	'Request / Check out / Check in access' privilege	boolean
smonmanage	'Request to search all recorded sessions' privilege	boolean
smonsearch	'Search my recorded sessions' privilege	boolean
smonsearchother	'Search all recorded sessions' privilege	boolean
smonview	'Download recorded sessions' privilege	boolean
smonremove	'Remove recorded session packages' privilege	boolean
viewgroups	'Groups' privilege	boolean
reg	'Password synchronization registration' privilege	boolean
ivrreg	'Generate voice print enrollment PIN' privilege	boolean
token	'Manage tokens' privilege	boolean
trackreq	'Track request status' privilege	boolean
adhoccertify	'Initiate a review of all entitlements' privilege	boolean
mobile	'Manage mobile devices' privilege	boolean
personalvault	'Manage personal vault' privilege	boolean

name	description	type
smonbrowserview	'View recorded sessions in browser' privilege	boolean
extendcheckout	'Request check-out extensions' privilege	boolean
viewworkflow	'View only for workflow requests' privilege	boolean
showans	'View answers to security questions' privilege	boolean

H.12 WSTN (Workstation) Resource Type

name	description	type
workstnid	Workstation netbios ID	string
description	Workstation description	string
ip	IP address - max for ipv4 is 15, ipv6 is 45	string
status	Current status of the workstation; M = Workstation passwords are managed, U = Workstation passwords are unmanaged, only a storage	string
ostype	Operating system/Application type; WINNT = Windows NT, WINNTS = Windows NT Server, WINNTDC = Windows NT Domain Controller, WIN2000 = Windows 2000, WIN2000S = Windows 2000 Server, WIN2000DC = Windows 2000 Domain Controller, WINXP = Windows XP, WIN2003S = Windows 2003 Server, WIN2003DC = Windows 2003 Domain Controller, UNIX = Unix, LINUX = Linux, IDM = Bravura Security Fabric, SQL = MS SQL, ORACLE = ORACLE Database, DB2 = DB2 Database, WINVISTA = Windows Vista, LONGHORN = Windows Longhorn	string
wstntype	Type of server/application being managed; W = Workstation, S = Server	string
osversion	Workstation/application version string (e.g Windows 2000 Server)	string
logonuser	User ID of the last logged in user	string
url	URL with resource information	string
platform	Platform of the workstation	string
deleted	Marked as deleted	boolean

Resource Configuration Parameters

ResourceCreateSet (p265) and ResourceSet (p276) accept an array of parameter values to create or update resource configurations used by Bravura Security Fabric. These parameter values are also returned by ResourceCreateGet (p265). The following is a list of the parameters available for each resource type supported by the above API functions.

I.1 Account Attribute Resource Type

This resource type accepts platform types as one of its configuration parameters. See Platform Type ID Values for the list of possible values.

name	description	required to create?	required to update?
ACCTATTR_ID	Identifier for the account attribute on the target system	yes	yes
ACCTATTR_GUID	Unique identifier for the account attribute at a particular override level	no	no
ACCTATTR_DATATYPE	The attribute data type; I=binary, B=boolean, F=memo, N=integer, or S=string	yes	no
ACCTATTR_OVERRIDE	The override scope of the account attribute; 0=default (no override), 1=override for all target systems of the given platform type, 2=override for the given target system. Default account attributes cannot be created, but can be updated if they are overridden using ACCTATTR_OVERRIDE_NEW.	yes	yes

name	description	required to create?	required to update?
ACCTATTR_OVERRIDE_NEW	The new override scope of the existing account attribute; 1=override for all target systems of the given platform type, 2=override for the given target system. The override scope can be changed only between overridden and non-overridden status. Changes to the configuration of a default account attribute are not allowed unless it is overridden first or ACCTATTR_OVERRIDE_NEW is set to 1 or 2 simultaneously with the other configuration changes.	no	no
ACCTATTR_PLATFORM	The platform type; required if ACCTATTR_OVERRIDE=1, ignored if ACCTATTR_OVERRIDE=2	no	no
ACCTATTR_TARGET_ID	The target system ID; required if ACCTATTR_OVERRIDE=2, ignored if ACCTATTR_OVERRIDE=1	no	no
ACCTATTR_MIN_VALUES	Minimum required number of values; 0=optional, >0=the minimum required number	yes	no
ACCTATTR_MAX_VALUES	Maximum allowed number of values; -1=unlimited, >=0 =the maximum allowed number	yes	no
ACCTATTR_CREATE_ACTION	How should the attribute be created during a c̈reate accountöperation; I=ignore the attribute, C=copy attribute values from the model account, R=copy attribute values but replace the account, S=set the attribute to specific values or according to profile/request attributes	yes	no
ACCTATTR_UPDATE_ACTION	How should the attribute be created during an üpdate attributesöperation; I=ignore the attribute, S=set the attribute to specific values or according to profile/request attributes, C=set the attribute to specific values or according to profile/request attributes only when the profile attribute has changed	yes	no
ACCTATTR_SEQUENCE	Attribute creation sequence; <0=set upon account creation, 0=sequence not important for this attribute, >0=set after new account creation	yes	no

name	description	required to create?	required to update?
ACCTATTR_ENCODING	Encoding used to store value (if ACCTATTR_DATATYPE=I, an encoding must be used); N=no encoding, B=Base 64 encoding	yes	no
ACCTATTR_MAP	Map the account attribute to the given profile/request attribute	no	no
ACCTATTR_TRACK	Whether to track changes to attribute values, if ACCTATTR_LOAD is 1; 0=no (default), 1=yes	no	no
ACCTATTR_LOAD	Whether to load the account attribute values during auto discovery; 0=no (default), 1=yes (required if ACCTATTR_TRACK=1 or ACCTATTR_POPULATE=1)	no	no
ACCTATTR_POPULATE	Whether to populate the mapped profile attribute with values from the target system, if ACCTATTR_MAP is set and ACCTATTR_LOAD is 1; 0=do not populate (default), 1=populate	no	no
ACCTATTR_FIXED_VALUES	Values to which the account attribute should be set, supplied as a KVGroup of KVPairs, where each pair is of the format " <attributevalue >" = " <L|P >", where L=literal value, and P=PSLang expression. Any previously assigned fixed values will be replaced by the new set; pass in an empty KVGroup to delete all previously assigned fixed values	no	no
ACCTATTR_PRIORITY	Priority relative to other account attributes mapped to the same profile attribute	no	no

I.2 Attribute Resource Type

name	description	required to create?	required to update?
ATTR_ID	A unique identifier for the attribute; built-in attribute IDs USERID, SHORTID, and FULLNAME are reserved	yes	yes
ATTR_DESC	Description of attribute seen by users	yes	no
ATTR_DATA_TYPE	The attribute data type; C=string, N=integer, B=boolean, M=memo, P=password, D=date/time, U=user, L=link, or F=file	yes	no
ATTR_MIN_REQUIRED	Minimum required number of values; 0=optional (default), >0=the minimum required number (must be 1 for boolean, memo, password, date/time, and file data types)	no	no
ATTR_VALIDATE_DATA	Whether to validate the link attribute value as a valid link, or the user attribute value as a valid profile ID; 0=do not validate (default for link), 1=validate (default for user)	no	no
ATTR_DISPLAY_TYPE	How to display the boolean, date/time, string or integer attribute; for boolean, C=checkbox, A=radio buttons, <empty value >=drop-down (default); for date/time, D=date only (calendar), <empty value >=date/time (default); for string or integer, B=drop-down, E=drop-down with two columns (not allowed if ATTR_MAX_ALLOWED is 1)	no	no
ATTR_MAX_ALLOWED	Maximum number of values allowed (required for string, integer, user, and link data types); -1=unlimited, >0=the maximum allowed number; 1 for all data types other than string, integer, user, and link data types	no	no
ATTR_MAX_LEN	Maximum field length of an attribute value, up to 450 (required for string, integer, and password data types)	no	no
ATTR_UNIQUE	Whether the values for a multi-valued string, integer, or link attribute must be unique; 0=no, 1=yes (default)(must be 1 for boolean, memo, password, date/time, and user data types)	no	no

name	description	required to create?	required to update?
ATTR_EMAIL	Whether the attribute can be included in the information sent in email notifications; 0=cannot be included (default)(must always be the case for password and file data type attibutes), 1=can be included	no	no
ATTR_PROFILE_COMPARE	Whether the attribute can be included on the "Profile comparison page"; 0=cannot be included (default)(must always be the case for password data types), 1=can be included	no	no
ATTR_AUTHCHAIN	Whether the string, integer, password, or date/time attribute is allowed for authentication chains; 0=not allowed (default), 1=allowed	no	no
ATTR_REQUEST_ONLY	Whether the attribute is used for requests only and not saved with the user's profile; 0=saved with user's profile (default), 1=used for requests only	no	no
ATTR_TRACK	Whether to track changes to attribute values; 0=no (default), 1=yes	no	no
ATTR_CONFIRM	Whether users must re-type the string, integer, or password attribute value; 0=no (default)(must always be the case if ATTR_MAX_ALLOWED is -1 or greater than 1), 1=yes	no	no
ATTR_INVALIDATE	Whether an authorizer changing an attribute value will invalidate previous authorizations of a request; 0=no (default), 1=yes	no	no
ATTR_ADVANCED_SEARCH	Whether to include the attribute in advanced searches; 0=no, 1=yes (default)	no	no
ATTR_REPORT	Whether to include the attribute values in reports; 0=no, 1=yes (default)	no	no
ATTR_INPUT_DESC	Description, displayed to users, of the input format of values for string, integer, memo, and password data types	no	no
ATTR_INPUT_FORMAT	Format requirement of input values for string and password data types	no	no
ATTR_INPUT_REGEXP	Regular expression used to validate input values for string, integer, memo, and password data types; requires ATTR_INPUT_DESC to be set to a value if setting ATTRIBUTE_INPUT_REGEXP to an expression	no	no

name	description	required to create?	required to update?
ATTR_VALUE_PLUGIN	Plugin used to generate a list of restricted values for string and integer data types	no	no
ATTR_PARENT	Parent attribute of string or integer attribute	no	no
ATTR_DEFAULT_VALUES	Default values for the string, integer, or boolean data type attribute, supplied as a KVGroup of KVPairs, where each KVPair is of the format " <default value >" = "", or " <parent attribute actual value >" = " <default value >" (supply an empty KVGroup to clear the current values); default boolean values must be either T, F, or N	no	no
ATTR_INHERIT_VALIDATE	Whether to inherit validation enforcement rules from the attribute group to which the attribute belongs; 0=no, 1=yes (default)	no	no
ATTR_CREATE_VALIDATE	Whether to enforce validation when creating new accounts, if ATTR_INHERIT_VALIDATE is 0; 0=no, 1=yes (default)	no	no
ATTR_UPDATE_VALIDATE	Validation behavior when updating existing accounts, if ATTR_INHERIT_VALIDATE is 0; NEVER=never enforce, THIS=enforce when updating values of the attribute, ALWAYS=always enforce (default)	no	no
ATTR_IMAGE_LINK	Whether to display a link or file data type as an image; 0=no (default), 1=yes	no	no
ATTR_POS_BOOL	Text to display for the positive value of a boolean data type attribute (required for boolean datatypes only)	no	no
ATTR_NEG_BOOL	Text to display for the negative value of a boolean data type attribute (required for boolean datatypes only)	no	no
ATTR_NONE_BOOL	Text to display for no value of a boolean data type attribute (required for boolean datatypes only)	no	no

name	description	required to create?	required to update?
ATTR_RESTRICTED_VALUES	Restricted values for string and integer data types, supplied as either a KVGroup of KVPairs, where each pair is of the format " <actual value >" = " <displayed value >", or a KVGroup of KVGroups, where each sub-group uses a parent attribute's actual value as its key and contains KVPairs for the restricted values of the current attribute. If <displayed value >is an empty value, the actual value will be used as the displayed value. Any previously assigned restricted values will be replaced by the new set; pass in an empty KVGroup to delete all previously assigned restricted values	no	no
ATTR_RESTRICTED_VALUE_ORDER	Order in which restricted values for string and integer data types are listed for users; A=actual value order, D=displayed value order	no	no
ATTR_KEEP_CASE	Restricted values are case-sensitive; 0=no (default), 1=yes	no	no
ATTR_VCARD_PROP	Map to vCard property; <empty value >(default), ADR, BDAY, EMAIL, FN, GENDER, GEO, IMPP, LANG, N, NICKNAME, NOTE, ORG, PHOTO, RELATED, ROLE, TEL;TYPE=cell, TEL;TYPE=fax, TEL;TYPE=home, TEL;TYPE=work, TITLE, TZ, UID, URL, XML	no	no
ATTR_SUBMIT_ONLY	Is this attribute editable only request creation; 0=no (default), 1=yes	no	no

I.3 Attribute Group Resource Type

name	description	required to create?	required to update?
ATTRGRP_ID	A unique identifier for the attribute group	yes	yes
ATTRGRP_DESC	Description of attribute group seen by users	yes	no
ATTRGRP_CREATE_VALIDATE	Whether to enforce validation when creating new accounts, if ATTR_INHERIT_VALIDATE of the attribute member is 0; 0=no, 1=yes (default)	no	no
ATTRGRP_UPDATE_VALIDATE	Validation behavior when updating existing accounts, if ATTR_INHERIT_VALIDATE of the attribute member is 0; NEVER=never enforce, THIS=enforce when updating values in this group, ALWAYS=always enforce (default)	no	no
ATTRGRP_NOTES_ABOVE	Notes above attribute	no	no
ATTRGRP_NOTES_BELOW	Notes below attribute	no	no
ATTRGRP_DISPLAY_TYPE	Where the group's attributes are displayed to users; M = main request or profile page, S = subsidiary page, N = hidden from users (default)	no	no
ATTRGRP_DISPLAY_ORDER	Relative display order	no	no
ATTRGRP_DISPLAY_OPER	Which operations a custom request should contain in order to display the group's attributes. Operations are specified as a KVGroup of KVPairs, where the key of each pair is the name of a request operation. Names of request operations are returned by WFRequestOperationTypes. Any previously assigned operations will be replaced by the new ones; pass in an empty KVGroup to delete all previously assigned operations. Ignored if only HiPM is installed. Some operations may not be supported if they are not supported by the installed product license	no	no

name	description	required to create?	required to update?
ATTRGRP_ACL	User group access control to the attribute group. ACLs are specified as a KVGroup of KVPairs, where each pair is of the format " <usergroupid >" = " <permission >", with <permission >being either R (read-only access), W (write-only access), RW (read and write access), or empty value (no access). Any previously assigned ACLs will be replaced by the new ones; pass in an empty KVGroup to delete all previously assigned ACLs	no	no
ATTRGRP_MEMBERS	The attributes that belong to the attribute group. Attributes are specified as a KVGroup of KVPairs, where each pair is of the format " <attributeid >" = " <relativeorder >", with <relativeorder >being the order by which the attribute is listed in relation to the other attributes (must be an integer greater than 0, or an empty value (last position in ordering)). Any previously assigned members will be replaced by the new ones; pass in an empty KVGroup to delete all previously assigned members	no	no

I.4 Managed Group Resource Type

name	description	required to create?	required to update?
MANAGED_GROUP_MGRPID	A unique internal identifier for the managed group	yes	yes

I.5 Managed account Resource Type

name	description	required to create?	required to update?
MACC_POLICYID	ID of the policy this account is on"	no	no
MACC_ISADMIN	Is administrator"	no	no
MACC_INVALID	Is valid account"	no	no
MACC_WORKSTNID	workstation netbios ID"	no	no
MACC_ACCOUNTNAME	The name of the account"	no	no

I.6 Network Resource Resource Type

name	description	required to create?	required to update?
NETWORK_RESOURCE_NETRESID	A unique internal identifier for the network resource	yes	yes

I.7 Role Resource Type

name	description	required to create?	required to update?
ROLE_ID	Unique identifier for the role	yes	yes

I.8 Target Resource Type

name	description	required to create?	required to update?
TARGET_ID	A unique identifier for the new target system. The target system ID can contain letters (A-Za-z), digits (0-9), and other ASCII characters. The target system ID cannot contain the following characters: :*? <>|"	yes	yes
TARGET_TYPE	The target type; M = manually-added target, T = discovery template	yes	no
TARGET_PLAT	The target system's platform type	yes	no
TARGET_DESC	The target system name that is displayed to users	yes	no
TARGET_ADDR	The address of the target system	yes	no

name	description	required to create?	required to update?
TARGET_CREDENTIALS	The administrator credentials of the target system. Use a KVGroup to contain one or more sub-KVGroups, where each sub-KVGroup is an administrator credential: "" "" = { " <adminid1 >" "" = { "PWD" = " <admin1pwd >" } ... " <adminidN >" "" = { "PWD" = " <adminNpwd >" } }. To specify that an administrator credential is a system password, include the KVPair "IS_SYS_PWD" = "1" in the KVGroup of that administrator credential. To specify that an administrator credential should be used to run the connector, include the KVPair "RUN_AS" = "1" in the KVGroup of that administrator credential. To specify whether an administrator credential should be updated by the Privileged Access Manager, include the KVPair "UPDATED_BY_PAM" = "1" (true) or "UPDATED_BY_PAM" = "0" (false) in the KVGroup of that administrator credential; when not specified, the setting is true by default. (The following requires the TARGET_CREDENTIAL_ASSOCIATION system variable enabled) To associate the administrator credential with a managed password, include the KVPairs "MANAGED_SYSTEM" = " <managedsystemid1 >" "MANAGED_ACCOUNT" = " <managedacctid1 >"	no	no
TARGET_URL	The URL of a web page containing a longer description of the target system	no	no
TARGET_PAM_AUTO_CREATE	Whether to configure this target system as a push-mode managed system in Privileged Access Manager; 1 = true (default if TARGET_TYPE = T), 0 = false	no	no
TARGET_MGRP_NRTYPE	CIFS = SMB Protocol for Active Directory DN (you will be managing access to network shares on an Active Directory DN target system), SMB = SMB Protocol (you will be managing access to network shares on an Active Directory target system), NRSHRPT = SharePoint Resource (you will be managing access to sites and documents on a SharePoint target system)	no	no

name	description	required to create?	required to update?
TARGET_CERTIFICATION	Whether this target system is included in a certification process; 1 = yes, 0 = no (default)	no	no
TARGET_PROFILE_SRC	Whether the users of the target system should also be profile users; 1 = yes, 0 = no (default)	no	no
TARGET_PROFILE_SRC_ATTR	The target system attribute to use as the source of profile IDs; by default, the short ID is used	no	no
TARGET_AUTO_ASSOC	Whether to automatically associate user IDs on target systems with identical profile IDs; 1 = yes (default when TARGET_TYPE = M), 0 = no	no	no
TARGET_AUTO_ASSOC_ATTR	The target system attribute to use to auto-associate user IDs on the system with profile IDs; by default, the short ID is used	no	no
TARGET_LIST_ACCT	Whether to execute connector list operations during auto-discovery; 1 = yes (default), 0 = no	no	no
TARGET_LIST_ATTR	Whether to generate a list of attributes for each account during auto-discovery; 1 = yes (default if TARGET_TYPE = M), 0 = no. To use this, TARGET_LIST_ACCT must also be set to 1	no	no
TARGET_LIST_GROUPS	Whether to generate a list of groups for each target system during auto-discovery; 1 = yes (default when TARGET_TYPE = M), 0 = no. To use this, TARGET_LIST_ACCT must also be set to 1	no	no
TARGET_PROFILE_FROM_ENABLED	Whether to create profile IDs from only enabled users on the target system; 1 = yes, 0 = no (default). Requires TARGET_PROFILE_SRC, TARGET_LIST_ATTR, and TARGET_LIST_ACCT to be set to 1	no	no
TARGET_USE_ID_FILTERS	Whether to exclude or include certain users and accounts via the use of ID filters; 1 = include, 0 = exclude (default)	no	no
TARGET_PWD_EXPIRY_CHECK	Whether to check for users on the target system whose passwords will soon expire or have already expired; 1 = yes (default if TARGET_TYPE = M), 0 = no. Cannot be set to 1 if TARGET_LIST_ATTR and TARGET_LIST_ACCT are not also set to 1	no	no

name	description	required to create?	required to update?
TARGET_SUB_HOST_PLUGIN	The name of the plugin to run if the target system represents a collection of actual target systems	no	no
TARGET_MGRP_BY_OWNER	How to manage groups automatically; 1 = manage only groups with owners, 0 = do not manage groups automatically (default), -1 = manage all groups, -2 = manage all groups, no approval required	no	no
TARGET_MGRP_INHERIT_AUTH_CFG	Whether to automatically configure the managed group's authorizers; 1 = yes, 0 = no (default)	no	no
TARGET_MGRP_INHERIT_IMPL_CFG	Whether to automatically configure the managed group's implementers; 1 = yes, 0 = no (default)	no	no
TARGET_CONNECTOR_TIMEOUT	The timeout, in seconds, for the connector; by default, 300 seconds	no	no
TARGET_LIST_TIMEOUT	The timeout, in seconds, for the list operation on the target system; by default, -1 (unlimited)	no	no
TARGET_LIST_MIN_SIZE	The minimum size, in bytes, of list files; by default, 50 bytes	no	no
TARGET_PROXIES	Proxy servers which can run connectors on behalf of the main server. List servers, separated by a semi-colon in the format: <server name >/ <port number >	no	no
TARGET_PERSISTENTSERVER	Address of the server to run persistent list on. Should be left empty to use the local server.	no	no
TARGET_PERSISTENTPORT	Port of the server to run persistent list on. -1 will use the default.	no	no
TARGET_VERIFY_PWD	Whether the target system can be used for authentication; 1 = yes (default), 0 = no	no	no
TARGET_NEW_ID_UNIQUE	Whether to ensure new profile IDs do not conflict with IDs on this target system; 1 = yes (default when TARGET_TYPE = M), 0 = no	no	no
TARGET_CASE_SET	The program to use to set the case of a new ID for this target system; upper.pss = use all uppercase characters; lower.pss = use all lowercase characters (default)	no	no
TARGET_GROUP	The target system group to which the target system belongs; by default, DEFAULT (default target system group)	no	no

name	description	required to create?	required to update?
TARGET_PWD_CHG	Whether passwords can be changed on the target system; 1 = yes (default), 0 = no	no	no
TARGET_PWD_CHG_MUST_SELECT	Whether the target system must be selected for password operations; 1 = yes, 0 = no (default)	no	no
TARGET_UNLOCK	Whether users on the target system can be unlocked; 1 = yes (default), 0 = no	no	no
TARGET_MUST_HAVE_ACCT	Whether profile users must have an account on the target system; 1 = yes, 0 = no (default)	no	no
TARGET_ALLOW_ATTACH_OTHER	Whether a profile user can attach manually a login ID from another user on an auto-associated target system; 1 = yes, 0 = no (false)	no	no
TARGET_ALLOW_DETACH_OTHER	Whether a profile user can detach manually-attached accounts; 1 = yes, 0 = no, -1 = obey PSL_ALLOW_DELETE option (default)	no	no
TARGET_CONTAINER_DN_ATTR	Profile or request attribute ID to use as the container DN when creating or moving accounts on context-sensitive target systems	no	no
TARGET_DISP_ATTACH_OTHER	Whether the target system will appear on the self-service login ID reconciliation page to allow profile users to attach users from this system; 1 = yes, 0 = no (default if TARGET_TYPE = T)	no	no
TARGET_DISP_PERSONAL_INFO	Whether the target system will appear on the IDR user management page; 1 = yes, 0 = no (default if TARGET_TYPE = T)	no	no
TARGET_DISP_PWD_CHG	Whether the target system will appear on the self-service 'Change passwords' page to allow profile users to change their passwords on this system; 1 = yes, 0 = no (default if TARGET_TYPE = T)	no	no
TARGET_DISP_UNLOCK	Whether the target system will appear on the self-service 'Unlock accounts' page to allow profile users to unlock their accounts on this system; 1 = yes, 0 = no (default if TARGET_TYPE = T)	no	no
TARGET_DISP_ADMIN_ATTACH_OTHER	Whether the target system will appear on the administrative login ID reconciliation page to allow help-desk users to attach users from this system to profiles; 1 = yes, 0 = no (default if TARGET_TYPE = T)	no	no

name	description	required to create?	required to update?
TARGET_DISP_ADMIN_PWD_CHG	Whether the target system will appear on the assisted 'Password changes' page to allow help-desk users to change users' passwords on this system; 1 = yes, 0 = no (default if TARGET_TYPE = T)	no	no
TARGET_DISP_ADMIN_UNLOCK	Whether the target system will apper on the user administration 'Unlock accounts' page to allow help-desk users to unlock accounts on this system; 1 = yes, 0 = no (default if TARGET_TYPE = T)	no	no
TARGET_DISCOVERY_OPTIONS	Additional resources to discover and load during auto discovery. Valid options are: ls_compsvr = load computer server objects, ls_compwkstn = load computer workstation objects, ls_scmacct = load service manager accounts, ls_taskacct = load scheduled task accounts, ls_iisacct = load IIS manager accounts, ls_comacct = load DCOM manager accounts, ls_admmember = load group members, ls_expression = load resources satisfying an expression, ls_manual = do not automatically discover resources to load, ls_normacct = enable all accounts to be discovered objects. Multiple discovery options should be separated by commas	no	no
TARGET_INCREMENTAL_LISTING	Whether incremental listing is allowed; 1 = yes (default), 0 = no	no	no
TARGET_PERSISTENTLIST	Whether persistent listing is enabled; 1 = yes (default), 0 = no	no	no
TARGET_ENABLE	Whether target allows enabling accounts; 1 = yes (default), 0 = no	no	no
TARGET_RBACENFORCE	Whether to enable RBAC enforcement; 1 = enable, 0 = disable (default)	no	no
TARGET_DACTION	When RBAC enforcement is enabled, what resolution action to take to for deficit violations; INHERIT=Use parent role setting, ADD=Add resource, EXCEPTION=Request exception	no	no
TARGET_SACTION	When RBAC enforcement is enabled, what resolution action to take to for surplus violations; INHERIT=Use system default, REMOVE=Remove resource, EXCEPTION=Request exception	no	no

name	description	required to create?	required to update?
TARGET_COPYFROM	Targets from which to copy list data as the list data of this target. Each target ID is separated by commas.	no	no

I.9 Template Resource Type

name	description	required to create?	required to update?
TEMPLATE_ID	A unique identifier for the template	yes	no
TEMPLATE_DESC	The description to display to users	yes	yes
TEMPLATE_TARGET	A unique identifier for the target system	yes	yes
TEMPLATE_ACCOUNT	The login ID of the account you want to use as a model or blueprint; required for account provisioning templates, optional for inventory templates	no	no
TEMPLATE_INVENTORY	Whether this is an inventory template; 1 = true, 0 = false (default)	no	no
TEMPLATE_LOCATION	A location for the template; required for inventory templates, optional for account provisioning templates	no	no
TEMPLATE_TYPE	A type for the template; required for inventory templates, optional for account provisioning templates	no	no
TEMPLATE_REQUIRE_PWD	Whether accounts created from this template do not require a password; 1 = true (default), 0 = false	no	no

I.10 User class Resource Type

name	description	required to create?	required to update?
UC_ID	Unique ID of class	yes	yes
UC_DESC	Description given to this class	yes	no
UC_SOURCE	Policy where the userclass can be referenced; [Shared] by default	no	no
UC_IS_DEPRECATED	Whether the user class is deprecated; 1 = true, 0 = false	no	no
UC_REASON	The reason for deprecating; if this is assigned to a non-empty value, then UC_IS_DEPRECATED should be set to true (1) at the same time	no	no
UC_PARTICIPANTS	Participant specifications assigned to the user class. Participant specifications are defined using the string form of a KVGroup containing one or more KVGroups, where each sub-KVGroup is a single participant specification of the following format: " <participant specification ID >" " <description of participant specification >" = { "use_expr" = " <whether to use expression for listing; 1 = use expression, 0 otherwise >", "list_expr" = " <PSLang expression for listing >", "is_excl" = " <whether users listed by the expression are the sole members of the user class; 1 = true, 0 otherwise >" }. The participant description and "use_expr" are required; "list_expr" and "is_excl" are required only if "use_expr" is set to 1. Setting this parameter will overwrite previous participant specifications.	no	no
UC_CRITERIA_PSLANG	PSLang expression criteria that user class participants must satisfy. The criteria are defined using the string form of a KVGroup containing one or more KVPairs, where each KVPair is of the form " <PSLang expressions >" = "". Setting this parameter will overwrite previous PSLang criteria specifications.	no	no

name	description	required to create?	required to update?
UC_CRITERIA_GROUP	Managed group membership criteria that user class participants must satisfy. The criteria are defined using the string form of a KVGroup containing one or more KVGroups, where each sub-KVGroup is of the format " <target ID >" " <group ID >" = { "participant" = " <participant ID >" "ismember" = " <1 or 0 >" }, where "ismember" is whether membership or non-membership in the group determines membership in the user class. Setting this parameter will overwrite previous group criteria specifications.	no	no
UC_CRITERIA_PROFILE_ATTR	Profile attribute membership criteria that user class participants must satisfy. The criteria are defined using the string form of a KVGroup containing one or more KVGroups, where each sub-KVGroup is of the format " <any unique value >" "" = { "participant" = " <participant ID >" "attribute" = " <profile attribute ID >" "casesensitive" = " <1 or 0 >" "participantcompare" = " <comparison participant ID >" "compareoper" = " <"EQ"=equals, "NE"=does not equal, "CN"=contains, "AB"=does not contain, "SW"=starts with, "NS"=does not start with, "NW"=ends with, "NN"=does not end with, "GT"=greater than, "GE"=greater than or equal, "LT"=less than, "LE"=less than or equal >" "value" = " <attribute value (dates should be in CCYYMMDDhhmmss[+-]hhmm format, booleans should be "T" or "F") >" }. "compareoper" and "value" is ignored if "participantcompare" is not empty. Setting this parameter will overwrite previous attribute criteria specifications.	no	no

I.11 User group Resource Type

name	description	required to create?	required to update?
UGRP_ID	Unique ID of user group	yes	yes
UGRP_POINTID	ID of the associated user class point. Must be either ACCESS_RULE_DELEGATED, ACCESS_RULE_GLOBAL, or ACCESS_RULE_SELF_SERVICE.	yes	yes
UGRP_DESC	Description given to this user group	yes	no
UGRP_RESET	"Change passwords" privilege	no	no
UGRP_RESETEXPIRE	"Change and expire passwords" privilege	no	no
UGRP_HDD	"Unlock encrypted systems/accounts" privilege	no	no
UGRP_UNLOCK	"Unlock accounts" privilege	no	no
UGRP_PROFILEENABLE	"Enable/Disable user profile" privilege	no	no
UGRP_PROFILEUNLOCK	"Unlock user profile" privilege	no	no
UGRP_VIEW	"View profile information" privilege	no	no
UGRP_VIEWACCT	"View accounts" privilege	no	no
UGRP_VIEWROLE	"View roles" privilege	no	no
UGRP_UPDATE	"Update account" privilege	no	no
UGRP_ADDROLE	"Add roles" privilege	no	no
UGRP_DELROLE	"Delete roles" privilege	no	no
UGRP_CREATE	"Create account" privilege	no	no
UGRP_ENABLE	"Enable account" privilege	no	no
UGRP_DISABLE	"Disable account" privilege	no	no
UGRP_DELETE	"Delete account" privilege	no	no
UGRP_MANAGEGROUPS	"Manage group memberships" privilege	no	no
UGRP_PPMREQREVOKE	"Revoke privileged access requests" privilege	no	no
UGRP_RENAME	"Rename account" privilege	no	no
UGRP_RENAMEPROFILE	"Rename profile" privilege	no	no
UGRP_MOVECTX	"Move user from one context to another" privilege	no	no
UGRP_VIEWHISTORY	"View profile history" privilege	no	no
UGRP_DELEGMANG	"Delegate workflow requests" privilege	no	no
UGRP_FLOWMANG	"Manage workflow requests" privilege	no	no
UGRP_DELEGATE	"Delegate authority" privilege	no	no
UGRP_EDITQA	"Update security questions" privilege	no	no

name	description	required to create?	required to update?
UGRP_VIEWQA	"View security questions" privilege	no	no
UGRP_BYPASSQA	"Bypass security questions" privilege	no	no
UGRP_EDITIDS	"Attach other accounts" privilege	no	no
UGRP_NETRES	"Request access to network resources" privilege	no	no
UGRP_VIEWORG	"Browse the OrgChart" privilege	no	no
UGRP_MANAGEORG	"View my subordinates" privilege	no	no
UGRP_PWDACCESS	"Request / Check out / Check in access" privilege	no	no
UGRP_SMONMANAGE	"Request to search all recorded sessions" privilege	no	no
UGRP_SMONSEARCH	"Search my recorded sessions" privilege	no	no
UGRP_SMONSEARCHOTHER	"Search all recorded sessions" privilege	no	no
UGRP_SMONVIEW	"Download recorded sessions" privilege	no	no
UGRP_SMONREMOVE	"Remove recorded session packages" privilege	no	no
UGRP_VIEWGROUPS	"Groups" privilege	no	no
UGRP_REGISTER	"Password synchronization registration" privilege	no	no
UGRP_IVRREG	"Generate voice print enrollment PIN" privilege	no	no
UGRP_TOKEN	"Manage tokens" privilege	no	no
UGRP_TRACKREQ	"Track request status" privilege	no	no
UGRP_ADHOCCERTIFY	"Initiate a review of all entitlements" privilege	no	no
UGRP_MOBILE	"Manage mobile devices" privilege	no	no
UGRP_PERSONALVAULT	"Manage personal vault" privilege	no	no
UGRP_SMONBROWSERVIEW	"View recorded sessions in browser" privilege	no	no
UGRP_EXTENDCHECKOUT	"Request check-out extensions" privilege	no	no
UGRP_VIEWWORKFLOW	"View only for workflow requests" privilege	no	no
UGRP_SHOWANS	"View answers to security questions" privilege	no	no

name	description	required to create?	required to update?
UGRP_CRITERIA_UC	User classes that determine membership to the user group. The criteria are defined using the string form of a KVGroup containing one or more sub-KVGroups, where each sub-KVGroup is of the format " <userclassid >" "" = { " <participantid1 >" = <maptype >... " <participantidN >" = <maptype >}, where <maptype >is either "RECIPIENT" or "REQUESTER" if UGRP_POINTID is ACCESS_RULE_DELEGATED; the contents of the sub-KVGroup are ignored if UGRP_POINTID is not ACCESS_RULE_DELEGATED. If there is more than one user class sub-KVGroup, then the name of the main KVGroup must be either "ANY" (participants must match any of the user classes) or "ALL" (participants must match all user classes). Setting this resource parameter will overwrite previous criteria specifications.	no	no

I.12 Workstation Resource Type

name	description	required to create?	required to update?
WORKSTATION_ID	Workstation netbios ID	yes	no
WORKSTATION_IP	Deprecated	no	no
WORKSTATION_NAME	Workstation description	yes	no
WORKSTATION_OSTYPE	Optional. A description of the operating system or other identifying characteristic of the new managed system	no	no
WORKSTATION_OSVERSION	Workstation/application version string (e.g Windows 2000 Server)	no	no
WORKSTATION_TYPE	Deprecated	no	no
WORKSTATION_URL	URL with resource information	no	no

Workflow Request Operations

J

WFRequestOperationTypes (p590) returns a list of allowed operation types for assignment to workflow requests.

Note that the names of custom request operations are not listed in the table below. Currently, 99 custom request operations are supported, named consecutively *CUST1*, *CUST2*, ..., *CUST99*.

name	description
ACUA	Add account to user
ADDM	Add an entitlement as member of a configuration object
AEDEF	Request an exception to not have a required role member
AEDEFDEL	Remove an approved exception to not have a required role member
AESOD	Request an exception to a SoD rule
AESODDEL	Request removal of an exception to a SoD rule
AESODEXT	Extend expiry date of an exception to a SoD rule
AESUR	Request an exception to have a resource that the role doesn't require
AESURDEL	Remove an approved exception to have a resource that the role doesn't require
ARCHREQGRP	Obtain group membership
ARCHREQMAQ	View multiple passwords
ARCHREQPWD	View password
ARCH_EXTEND_CHECKOUT	Extend check-out duration
CFYA	Certify account
CFYAEDEF	Certify exception to deficit
CFYAESOD	Exception for segregation of duties rule
CFYAESUR	Certify exception to surplus
CFYG	Certify group membership
CFYNESODACCT	Account member included in an exception of a SOD rule
CFYNESODGRP	Group member included in an exception of a SOD rule
CFYNESODROLE	Role member included in an exception of a SOD rule
CFYPROFATTR	Certify profile attribute

name	description
CFYR	Certify role membership
CFYRSVSOD	Resolve segregation of duties violation
CFYSIGN	Configuration Certification signoff
CFYSODACCT	Resolve segregation of duties violation - account member
CFYSODGRP	Resolve segregation of duties violation - group member
CFYSODROLE	Resolve segregation of duties violation - role member
CFYSUBG	Certify sub-group group membership
CFYU	Certify profile
CRTC	Create configuration object
CRTG	Create group
DELC	Create configuration object
DELG	Delete group
DELM	Delete an entitlement member from a configuration object
DELR	Delete role
DELU	Delete account
DNAP	Disable profile
DNAU	Disable account
ENAP	Enable profile
ENAU	Enable account
GOGA	Add an owner group to a group
GOGD	Remove an owner group from a group
GOOD	Remove unknown owner from a group
GRBD	Remove unknown object from a group
GRGA	Add a group as a member of another group
GRGD	Remove a group as a member of another group
GROA	Add an owner to a group
GROD	Remove an owner from a group
GRPA	Add a parent group to a group
GRPD	Remove a parent group from a group
GRUA	Add a user to a group
GRUD	Remove a user from a group
GRUP	Update group
LDEL	Delete login association
LUPD	Update login association
MVCU	Move account from one context to another
NESOD	Entitlement to be included in an exception of a SOD rule

name	description
NESODDEL	Entitlement to be removed from an approved exception of a SOD rule
NRAT	Network resource attributes
NRCR	Create a network resource
NRDL	Delete a network resource
NRMV	Move a network resource
NRUP	Update a network resource
OBJREL_ADD	Add object relation
OBJREL_CERTIFY	Certify object relation
OBJREL_DELETE	Delete object relation
OBJREL_UPDATE	Update object relation
OBJ_CERTIFY	Certify Object
OBJ_CREATE	Create Object
OBJ_DELETE	Delete Object
OBJ_DISABLE	Disable Object
OBJ_ENABLE	Enable Object
OBJ_RENAME	Rename Object
OBJ_UPDATE	Update Object
ORGADDMGR	Designate manager
ORGADDSUB	Attach subordinate
ORGDELMGR	Designate non-manager
ORGDELSUB	Detach subordinate
ORGSIGN	OrgChart building signoff
ORGTFRSUB	Transfer subordinate
ORGTFRSUBPULL	Transfer subordinate from another manager
PCHA	Account target for a generic PAM checkout request
PCHO	Generic PAM checkout request
RENU	Rename account
RLUA	Add role to user
RPRO	Rename profile
SERI	Server information
SM_BROWSER_VIEW	View a monitored session in a browser
SM_SEARCH	Search monitored sessions
SM_VIEW	View a monitored session
UMAD	Add explicit user as the userclass member
UMDE	Delete explicit user as the userclass member
UPAT	Update attributes

name	description
UPCO	Update configuration object
UPDT	Update account
UPRT	Update resource attribute
USRA	Create user
VWAT	View profile

PO Box 27008, Calgary RPO Tuscany, Calgary AB T3L 2Y1

Date: | 2025-05-16 File: git:fox:doc/fox/pslang/pslang.tex