

OPS:Using Variables

From OpsWise Documentation Wiki

Contents

- 1 Overview
- 2 User-Defined Variables
 - 2.1 Resolving User-Defined Variables
 - 2.2 Format for Using Variables
 - 2.3 Creating a New Variable
 - 2.4 Variable Field Descriptions
- 3 Built-In Variables
- 4 Setting Variables in a Workflow
 - 4.1 Overview
 - 4.2 Creating a Set Variable Instruction
 - 4.3 Set Variables Field Descriptions
- 5 Functions

Overview

OpsWise Automation Center supports the following types of variable, all of which can be used in free text fields within tasks:

- User-Defined Variables -- These variables are created by the user.
- Built-In Variables -- These variables are maintained by the system and allow you to access information about task instances and other related data, such as task name, task status, and trigger name.
- Functions -- These calculate some value, such as current date and time, or perform some function, such as ReplaceAll.

Each of these is described in more detail below.

User-Defined Variables

You can define Opswise variables in four different locations:

- Trigger variables are entered by clicking the **Variables** tab in a trigger. Trigger variables are stored in the table ops_local_variable.
- Task variables are entered by clicking the **Variables** tab in a task. Task variables are stored in the table ops_local_variable.
- Workflow variables are entered by clicking the **Variables** tab in a workflow. Workflow variables are stored in the table ops_local_variable.
- Global variables are entered by selecting **Automation Center > Variables** from the Navigation Pane. Global variables are stored in the table ops_variable.

Once the task has been launched and the variables are resolved, they are referred to as task instance variables. The task instance variables are stored in hashed format in the table ops_map_table.

Important Note -- Do not define OpsWise Automation Center variables with the prefix "ops_". That prefix is reserved for built-in variables.

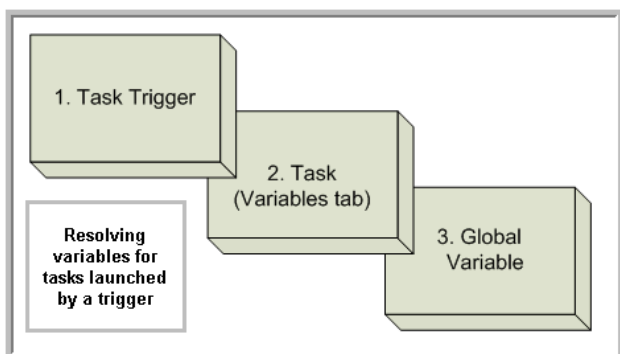
Resolving User-Defined Variables

When OpsWise Automation Center creates a task instance from a task definition, it also resolves all variables specified in its free text fields. Because you can define variables at four different levels (trigger, task, workflow, and global), OpsWise Automation Center follows a prescribed formula to determine which variable takes precedence if duplicate variables have been defined. The general order of precedence is: 1) task trigger, 2) task, 3) workflow trigger, 4) workflow, 5) global, as shown in the diagram below.

The following scenarios provide more detailed information about how OpsWise Automation Center variables are resolved.

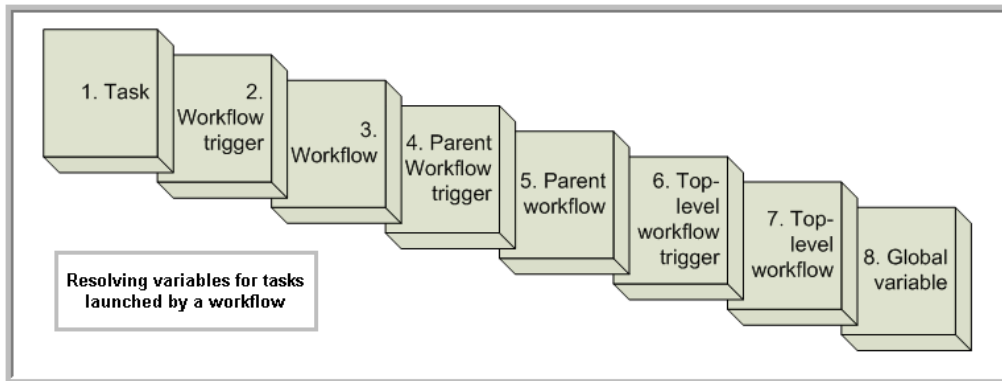
For tasks launched by a trigger:

1. If the trigger defines the variable in the variables tab, that value is used to resolve the variable.
2. If the trigger does not define the variable, the value from the variable tab in the task definition is used.
3. If neither the trigger nor the task define the variable, the variable definition in the global variables table is used.
4. If the global variables table does not define the variable, the variable remains unresolved.



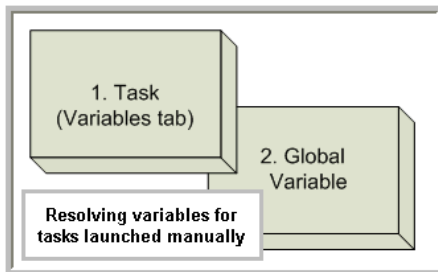
For tasks launched by a workflow:

1. If the task defines the variable in the variables tab, that value is used to resolve the variable.
2. If the task does not define the variable, and the workflow was launched by a trigger, the value defined in the trigger is used.
3. If the workflow's trigger does not define the variable or the workflow was not launched by a trigger, the value defined in the workflow is used.
4. If the workflow does not define the variable, and there is a parent workflow, the value defined in the parent workflow's trigger is used. If the trigger does not define the variable or if there is no trigger, the value defined in the parent workflow is used. If the parent workflow does not define the variable, the system checks up a level for another parent workflow and repeats the process. This continues until the top level workflow is reached.
5. If the top-level workflow does not define the variable, the variable definition in the global variables table is used.
6. If the global variables table does not define the variable, the variable remains unresolved.



For tasks launched manually:

1. If the task defines the variable in the variables tab, that value is used to resolve the variable.
2. If the task does not define the variable, the variable definition in the global variables table is used.
3. If the global variables table does not define the variable, the variable remains unresolved.



Format for Using Variables

When you enter a variable into a text field, precede the variable with the dollar sign (\$) and enclose the variable in parenthesis. You can enter a series of variables or nested variables. Examples are:

```

${variable_name}
${v1}${v2}
${${inner_variable}}

```

Creating a New Variable

1. For global variables, select **Variables** from the Navigation Pane. For trigger, task, or workflow variables, click on the **Variables** tab from the record. OpsWise Automation Center displays a list of variables, if any, as shown in the example below.

Name	Value	Description	Updated	Updated by
demo_ops_download_dir	/home/opswise/download		2008-11-26 16:32:16 -0800	glide.maint
demo_ops_global	Global		2008-12-10 16:46:04 -0800	ops.admin
demo_ops_linux_rc	0		2008-11-26 16:32:30 -0800	glide.maint
demo_ops_os	linux		2009-03-02 09:33:22 -0800	ops.admin
demo_ops_rc	0		2008-11-26 16:32:43 -0800	glide.maint
demo_ops_remove	\${demo_ops_toolsdir}\rm.exe		2008-11-26 16:35:31 -0800	glide.maint
demo_ops_rundir	\${demo_ops_toolsdir}		2008-11-26 16:35:20 -0800	glide.maint
demo_ops_sleep_time	5		2008-11-26 16:33:14 -0800	glide.maint
demo_ops_snooze	\${demo_ops_toolsdir}\snooze_bg.bat		2008-11-26 16:35:07 -0800	glide.maint
demo_ops_toolsdir	\${demo_ops_workspace}\com.jme.opswise.plugins\tools\windows		2008-11-26 16:34:59 -0800	glide.maint
demo_ops_touch	\${demo_ops_toolsdir}\touch.exe		2008-11-26 16:34:52 -0800	glide.maint
demo_ops_workspace	C:\workspace\clipse\opswise		2008-11-26 16:34:04 -0800	glide.maint

- From the wizard, select **New**. OpsWise Automation Center displays the Variable screen:

- Using the field descriptions provided below as a guide, complete the fields as needed.
- Click the **Submit** button to save the record and return to the menu, or, right-click and select **Save** to save the record and remain on the current display.
- If appropriate, repeat these steps for any additional variables you want to add.

Variable Field Descriptions

The table below describes the fields and buttons on the Variables screen.

Field Name	Description
Name	Required. Name used within OpsWise Automation Center to identify this variable. Up to 40 alphanumeric; no spaces. It is the user's responsibility to develop a workable naming scheme for tasks. Important Note -- Do not define OpsWise Automation Center variables with the prefix "ops_". That prefix is reserved for built-in variables.
Value	Optional. The value of the variable, if any.
Description	Optional. The description of the variable, if any.
Submit button	Submits the new record to the database.
Update button	Saves updates to the record.
Delete button	Deletes the current record.

Built-In Variables

Built-in variables are maintained by OpsWise Automation Center. They can be used in free text fields and provide information about tasks instances and triggers. They are resolved when the task is launched into a task instance. Supported built-in variables and their descriptions are provided in the table below. All built-in variables are prefixed with "ops_".

Name	Description
The following built-in variables are associated with task instances:	
<code>#{ops_agent_hostname}</code>	Resolves to the agent hostname.
<code>#{ops_agent_ip}</code>	Resolves to the agent IP address.
<code>#{ops_agent_name}</code>	Resolves to the agent name.
<code>#{ops_cmd}</code>	Resolves to the task command.
<code>#{ops_cmd_parms}</code>	Resolves to the task command parameters.
<code>#{ops_exit_code}</code>	Resolves to the task instance exit code.
<code>#{ops_retry_count}</code>	Resolves to the current retry count.
<code>#{ops_retry_interval}</code>	Resolves to the retry interval (seconds).
<code>#{ops_retry_maximum}</code>	Resolves to the maximum retry count.
<code>#{ops_status}</code>	Resolves to the current task instance status.
<code>#{ops_task_id}</code>	Resolves to the sys_id of the task instance.
<code>#{ops_task_name}</code>	Resolves to the task instance name.
<code>#{ops_task_type}</code>	Resolves to the task instance type.
<code>#{ops_workflow_id}</code>	Resolves to the sys_id of the parent workflow task instance.
When a task is launched by a trigger, the following built-in trigger variables are passed into the task instance:	
<code>#{ops_trigger_name}</code>	Resolves to the name of the trigger that launched the task instance.
<code>#{ops_trigger_file_name}</code>	File Monitor trigger only. Resolves to the name of the file that fired the trigger.
<code>#{ops_trigger_file_size}</code>	File Monitor trigger only. Resolves to the file size of the file that fired the trigger.
<code>#{ops_trigger_file_date}</code>	File Monitor trigger only. Resolves to the file date of the file that fired the trigger.

<code>#{ops_trigger_file_scan}</code>	File Monitor trigger only. Resolves to the scan results.
<code>#{ops_trigger_file_owner}</code>	File Monitor trigger only. Resolves to the file owner of the file that fired the trigger.
<code>#{ops_trigger_file_group}</code>	File Monitor trigger only. Resolves to the file group of the file that fired the trigger.
<code>#{ops_trigger_task_id}</code>	Task Monitor trigger only. Resolves to the task instance sys_id that fired the trigger.
<code>#{ops_trigger_task_name}</code>	Task Monitor trigger only. Resolves to the task instance name that fired the trigger.
<code>#{ops_trigger_task_status}</code>	Task Monitor trigger only. Resolves to the task instance status that fired the trigger.
<code>#{ops_trigger_task_type}</code>	Task Monitor trigger only. Resolves to the task instance type that fired the trigger.
<code>#{ops_trigger_filename_nopath}</code>	Resolves to the file name without any path information.
<code>#{ops_trigger_file_name_simple}</code>	Resolves to the base file name.
<code>#{ops_trigger_file_name_extension}</code>	Resolves to the file extension of a file.
<code>#{ops_trigger_file_path}</code>	Resolves to the directory where the new file was created, but not the file itself.

Setting Variables in a Workflow

Overview

Within a workflow, you can specify instructions that set a variable to a specific value for use within the workflow. The value you set using this method exists only in memory for the period this workflow is running or until another Set Variable instruction sets it to another value. You can create the Set Variable instruction at the workflow level or task level. If you set it at the workflow level, you have the option of setting it based on the workflow, the tasks (children) in the workflow, or both. If you set it at the task level, you have the option of setting it based on information in the task, in the parent workflow, or both.

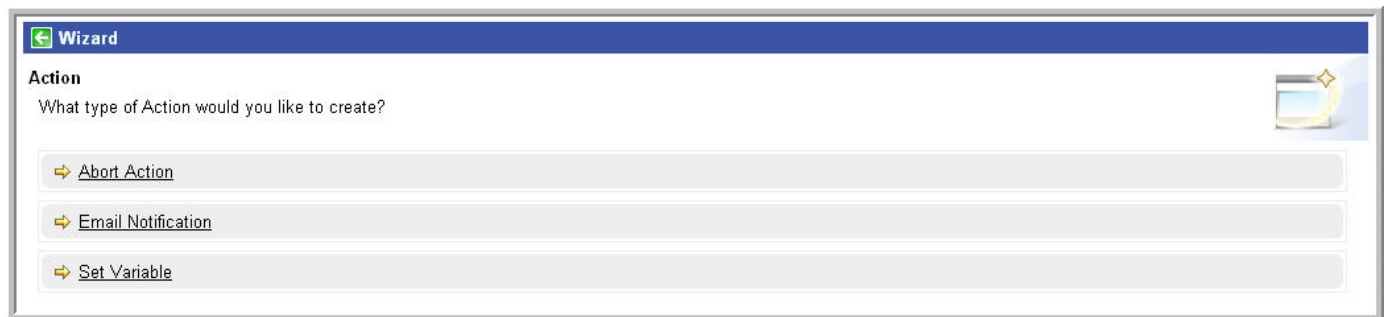
The variable need not exist in the database; you can create a new variable using the Set Variable instruction.

When creating a Set Variable instruction, you can trigger the action based on one or more of the following:

- Status
- Exit codes
- Late start
- Late or early finish

Creating a Set Variable Instruction

1. Display the workflow for which you want to create Set Variable instructions.
2. Click on the **Actions** tab. This tab displays a list of all Actions for this task, including Set Variables, Email Notifications, and Abort Actions.
3. Click **New**. The Actions wizard displays.



4. Click **Set Variable**. The Set Variables screen displays.

5. Using the field descriptions provided below as a guide, complete the fields as needed.
6. Click the **Submit** button to save the record and return to the Actions list, or, right-click and select **Save** to save the record and remain on the current display.
7. If appropriate, repeat these steps for any additional Set Variable instructions you want to add.

Set Variables Field Descriptions

The table below describes the fields and buttons on the Set Variables screen.

Field Name	Description
Action Inheritance	Workflow tasks only. Specifies what records these instructions apply to. Options: <ul style="list-style-type: none"> ■ SELF - These instructions apply only to the workflow and are not inherited by its children tasks. ■ SELF/CHILDREN - These instructions apply to the workflow and its contained tasks (children). ■ CHILDREN - These instructions apply only to the tasks within the workflow (children).
Status	The status of the task instance that will satisfy the trigger. You can specify as many statuses as needed. Options: <ul style="list-style-type: none"> ■ Defined - All task types. The new task instance has been created (the task has been launched). Not yet implemented. ■ Waiting - All task types. The task has been put on hold by a workflow and is waiting to run. ■ Held - All task types. The task has been put on hold by a user. ■ Resource Wait - The task with a virtual resource defined is waiting for enough units to become available on the virtual resource. ■ Undeliverable - Agent-based tasks. The agent is unavailable. ■ Queued - Agent-based tasks only. The task has been queued on a resource. ■ Submitted - z/OS only. The task has been submitted to the z/OS Job Entry subsystem and scheduled by the z/OS Job Scheduler. ■ Action Required - Manual tasks only. When a manual task launches, it goes into Action Required status, meaning a user must perform some manual activity. For details, see Manual task. ■ Started - Agent-based and Manual tasks only. The task has started. For agent-based tasks, this means the agent has received the task. ■ Running - All task types. The task is running. For agent-based tasks, the agent has started running the program. ■ Running Problems - Workflows only. One or more tasks within the workflow has one of the following statuses: <ul style="list-style-type: none"> ■ Held ■ Undeliverable ■ Running Problems (for sub-workflows) ■ Cancel Pending ■ In Doubt ■ Start Failure ■ Cancelled ■ In Doubt - Agent-based tasks only. The agent is "in doubt" about the current status of the task instance. This may occur if an agent or agent connection goes down. In this case, the agent restarts and reviews its data about tasks in progress. If the agent finds a task still running, it resumes normal monitoring. If the agent cannot find the task, this usually indicates that the task completed, but the agent considers the task status to be "in doubt." ■ Start Failure - All task types. The task was unable to start. ■ Cancelled - All task types. The task was cancelled by a user. ■ Failed - All task types. The task ran to a failure status. ■ Skipped - All task types. The task was skipped by a user. ■ Finished - All task types. The task was forced by the user to finish. The user may do this in cases where the task had "Cancelled" or "Failed" status, and the user needed to release other task instances depending on the successful completion of this task instance in a workflow. For more information, see Force Finishing a Task.

	<ul style="list-style-type: none"> ■ Success - All task types. The task has completed successfully
Exit Codes	Specifies one or more exit codes that will trigger the Set Variable. If you specify an exit code, you must also specify at least one status. Use commas to separate multiple exit codes; use a hyphen to specify a range. Example: 1, 5, 22-30.
On Late Start	Execute the Set Variable if the task started late, based on the Late Start Time specified in the task.
On Late Finish	Execute the Set Variable if the task finishes late, based on the Late Finish time specified in the task.
On Early Finish	Execute the Set Variable if the task finishes early, based on the Early Finish Time specified in the task.
Variable Scope	Applies to variables associated with a task in a workflow. Options: <ul style="list-style-type: none"> ■ SELF - The variable is only set within the scope of the task that executes the Set Variable action. ■ PARENT - The variable is set within the scope of the (immediate) parent workflow. After it is set, any task within the parent workflow can access that variable. ■ TOP_LEVEL_PARENT - The variable is set within the scope of the top level parent. Example: Workflow A contains workflow B and workflow B contains workflow C. If a Set Variable action is executed by a task within workflow C with Variable Scope set to TOP_LEVEL_PARENT, then the variable will be set in workflow A's scope. This means that after it is set, tasks in workflow A, workflow B and workflow C can access that variable.
Name	Required. Name of variable being set. Up to 40 alphanumeric.
Value	Optional. The value being written to the variable.
Description	Optional. The description of the variable, if any.
Submit button	Submits the new record to the database.
Update button	Saves updates to the record.
Delete button	Deletes the current record.

Functions

OpsWise Automation Center supports a number of functions that can be specified in free text fields. They are executed when the task is launched into a task instance. They are entered using the following formats:

```

${_function}
${_function(arg1, ... , argn)}

```

The following table lists and describes each supported function.

Name	Description
<code>\${_currentTimeMillis}</code>	Resolves to the current time in milliseconds.
<code>\${_date([format, day_offset, hour_offset, minute_offset])}</code>	Resolves to the current date and time. All parameters are optional. Parameters: <ul style="list-style-type: none"> ■ <i>format</i> - the date format. The default format is yyyy-MM-dd HH:mm:ss Z. For details on the format parameter, go to: <div style="border: 1px dashed black; padding: 2px; margin: 5px 0;"> http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html </div> ■ <i>day_offset</i> - the day offset. ■ <i>hour_offset</i> - the hour offset. ■ <i>minute_offset</i> - the minute offset. Examples: <ul style="list-style-type: none"> ■ <code>\${_date}</code> --> 2008-07-14 12:43:06 -0400 ■ <code>\${_date()}</code> --> 2008-07-14 12:43:06 -0400 ■ <code>\${_date("yyyy-MM-dd",5)}</code> --> 2008-07-19 ■ <code>\${_date("yyyy-MM-dd HH:mm:ss",-2,-1)}</code> --> 2008-07-12 11:43:06 ■ <code>\${_date("",0,0,10)}</code> --> 2008-07-14 12:53:06 -0400
<code>\${_dateadv([format, year_offset, month_offset, day_offset, hour_offset, minute_offset])}</code>	Resolves to the current date and time. All parameters are optional. Parameters: <ul style="list-style-type: none"> ■ <i>format</i> - the date format. The default format is yyyy-MM-dd HH:mm:ss Z. For details on the format parameter, go to: <div style="border: 1px dashed black; padding: 2px; margin: 5px 0;"> http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html </div> ■ <i>year_offset</i> - the year offset. ■ <i>month_offset</i> - the month offset. ■ <i>day_offset</i> - the day offset. ■ <i>hour_offset</i> - the hour offset. ■ <i>minute_offset</i> - the minute offset. Examples:

	<ul style="list-style-type: none"> ■ <code>\${_dateadv}</code> --> 2008-07-29 09:31:42 -0700 ■ <code>\${_dateadv("yyyy-MMM",-1)}</code> --> 2007-Jul ■ <code>\${_dateadv("yyyy-MMM".0,-1)}</code> --> 2008-Jun
<code>\${_guid}</code>	Resolves to a 32 byte GUID (Globally Unique ID).
<code>\${_hostname}</code>	Resolves to the hostname of the machine running the Core, if available.
<code>\${_indexOf(value, str)}</code>	<p>Returns the index within the string <i>value</i> of the first occurrence of the specified substring, <i>str</i>.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ■ <i>value</i> - any string. ■ <i>str</i> - the substring to search for. <p>If the <i>str</i> argument occurs as a substring within the <i>value</i>, then the index of the first character of the first such substring is returned; if it does not occur as a substring, -1 is returned.</p>
<code>\${_ipaddress}</code>	Resolves to the IP address of the machine running the Core.
<code>\${_lastIndexOf(value, str)}</code>	<p>Returns the index within the string <i>value</i> of the rightmost occurrence of the specified substring, <i>str</i>.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ■ <i>value</i> - any string. ■ <i>str</i> - the substring to search for. <p>If the <i>str</i> argument occurs one or more times as a substring within the <i>value</i>, then the index of the first character of the last such substring is returned. If it does not occur as a substring, -1 is returned.</p>
<code>\${_random(max, min)}</code>	<p>Generates a random number between <i>max</i> (inclusive) and <i>min</i> (inclusive).</p> <p>Parameters:</p> <ul style="list-style-type: none"> ■ <i>max</i> - the upper bound (inclusive) on the random number (default 9). ■ <i>min</i> - the lower bound (inclusive) on the random number (default 0).
<code>\${_replaceAll(value, regex, replacement)}</code>	<p>Replaces each substring of <i>value</i> that matches the specified regular expression, <i>regex</i>, with the specified replacement, <i>replacement</i>.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ■ <i>value</i> - the input string. ■ <i>regex</i> - the regular expression. ■ <i>replacement</i> - the replacement string.
<code>\${_resolve(variable_name, default_value)}</code>	<p>Resolves the variable specified by name <i>variable_name</i> and substitutes the default value <i>default_value</i> if the variable cannot be resolved.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ■ <i>variable_name</i> - the variable name. ■ <i>default_value</i> - the default value to use if the variable cannot be resolved.
<code>\${_resultsColumn(name, colname[, rownum, default_value])}</code>	<p>Returns the string value of a row/column from a previously executed SQL task.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ■ <i>name</i> - SQL task name in workflow containing results (use empty string for name if used outside of a workflow within a SQL task notification, etc). ■ <i>colname</i> - name of column to retrieve. ■ <i>rownum</i> - numeric row number in result set to retrieve (default 1). ■ <i>default_value</i> - the default value to return if result not found.
<code>\${_resultsColumnByNo(name, colnum[, rownum, default_value])}</code>	<p>Returns the string value of a row/column from a previously executed SQL task.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ■ <i>name</i> - SQL task name in workflow containing results (use empty string for name if used outside of a workflow within a SQL task notification, etc). ■ <i>colnum</i> - number of column to retrieve. First column in result is 1, 2nd is 2, etc ■ <i>rownum</i> - numeric row number in result set to retrieve (default 1). ■ <i>default_value</i> - the default value to return if result not found.
<code>\${_resultsColumnsCSV(name[, rownum])}</code>	<p>Returns the string values of columns in a specific row in CSV(comma separated values) format.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ■ <i>name</i> - SQL task name in workflow containing results (use empty string for name if used outside of a workflow within a SQL task notification, etc). ■ <i>rownum</i> - numeric row number in result set to retrieve (default 1).
<code>\${_scope}</code>	<p>This displays all the defined and built-in variables associated with the task instance.</p> <p>Example:</p>

	<ul style="list-style-type: none"> ▪ <code>\$_scope</code> --> {ops_workflow_id=, ops_task_type=Unix, ops_status=DEFINED, ops_retry_interval=60, ops_exit_code=0, ops_retry_maximum=0, ops_cmd_parms=, ops_cmd=ls -la; exit \$_random('9')};; ops_retry_count=0, ops_agent_id=67e4994143d2617201cdf4ba9df9ab0a, ops_task_id=84880af243d26172019aa1d25988a8f9, ops_task_name=Opwise - Linux Ls }
<code>\$_siblingid(sibling_name)</code>	<p>Resolves to the <code>sys_id</code> of the first task instance found within the same workflow specified by the <i>sibling name</i>.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ▪ <i>sibling_name</i> - the sibling name. <p>Example:</p> <ul style="list-style-type: none"> ▪ <code>\$_siblingid("Sleep 60")</code> --> 5dbaaab943d26172015e10ab3e894e10
<code>\$_substring(value, beginIndex[, endIndex])</code>	<p>Returns a new string that is a substring of <i>value</i>. The substring begins at the specified <i>beginIndex</i> and extends to the character at index <i>endIndex</i> -1.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ▪ <i>value</i> - the string to make a substring from. ▪ <i>beginIndex</i> - the beginning index, inclusive. ▪ <i>endIndex</i> - the ending index, exclusive. <p>Examples:</p> <ul style="list-style-type: none"> ▪ <code>\$_substring("hamburger", 4, 8)</code> resolves to "urge". ▪ <code>\$_substring("smiles", 1, 5)</code> resolves to "mile".
<code>\$_toLowerCase(value)</code>	<p>Converts all of the characters in the value to lower case using the rules of the default locale.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ▪ <i>value</i> - the string to convert to lower case.
<code>\$_toUpperCase(value)</code>	<p>Converts all of the characters in the value to upper case using the rules of the default locale.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ▪ <i>value</i> - the string to convert to upper case.
<code>\$_trim(value)</code>	<p>Returns a copy of value, with leading and trailing whitespace omitted.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ▪ <i>value</i> - the string to trim.

Retrieved from "http://opswiseframework.com/documentation/index.php/OPS:Using_Variables"

- This page was last modified 23:05, 23 September 2009.