



stonebranch
you imagine IT. we automate IT.

Opswise Controller 6.1.x

Variables and Functions

© 2015 by Stonebranch, Inc. All Rights Reserved.

1. Variables and Functions	3
1.1 Variables and Functions Overview	4
1.2 User-Defined Variables	5
1.3 Built-In Variables	12
1.4 Launching With Variables	30
1.5 Trigger With Variables	31
1.6 Creating a Set Variable Action within a Task or Workflow	32
1.7 Listing and Setting Variables from the Command Line	36
1.8 Functions	37

Variables and Functions



Variables



Built-In Variables

[Overview](#)

[User-Defined Variables](#)



Using Variables

[Setting Variables under Special Circumstances](#)

[Launching With Variables](#)

[Triggering with Variables](#)

[Creating a Set Variable Action within a Task or Workflow](#)

[Listing and Setting Variables from the Command Line](#)



Functions

[Overview](#)

[Date Functions](#)

[Mathematical Functions](#)

[SQL/Stored Procedure Functions](#)

[String Functions](#)

[System Functions](#)

[Overview](#)

[Agent Variables](#)

[Application Monitor Triq](#)

[Cluster Node Variables](#)

[File Monitor Task Instar](#)

[File Transfer Task Insta](#)

[FTP File Monitor Task I](#)

[OMS Server Variables](#)

[SAP Task Instance Vari](#)

[SQL and Stored Proced](#)

[System Monitor Task In](#)

[Task Instances Variable](#)

[Task Monitor Task Insta](#)

[Trigger Variables \(all tri](#)

[z/OS Task Instance Var](#)



The information on these pages also is located in the Opwise Controller 6.1.x Variables and Functions.pdf.

Variables and Functions Overview

- [Variables and Functions](#)
- [Types of Variables](#)
- [Setting Variables under Special Circumstances](#)

Variables and Functions

Variables and functions can be used in free-text fields within tasks and workflows. When a variable or function is specified in a free-text field, the Controller inserts its value into the field when the task or workflow is run.

Triggers can pass variables and functions into the tasks and workflows that they launch.

Additionally, email notifications for Controller resources (agents, OMS servers, and cluster nodes) can use [Built-In Variables](#) that are specific to that type of resource.

Types of Variables

Opwise Controller supports the following types of variables, all of which can be used in free text fields within tasks:

User-Defined Variables	These variables are created by the user for use within triggers, tasks, and workflows.
Built-In Variables	These variables, maintained by the Controller, allow you to access information about task instances and other related data, such as task name, task status, and trigger name.
Functions	These variables calculate some value, such as current date and time, or perform some function, such as <code>_replaceAll</code> .

Setting Variables under Special Circumstances

The Controller also supports several features that allow you to set variables under special circumstances:

- [Manually launch tasks and temporarily set user-defined variables.](#)
- [Manually launch all of the tasks associated with a trigger while supplying variable values used by the task\(s\)](#) (see [Triggering with Variables](#)).
- Use the Set Variable action to [set variables within a task or workflow](#).
- Use the `ops-variable-set` CLI command to set variables.

User-Defined Variables

- Overview
- Variable Naming Conventions
- Resolving User-Defined Variables
 - For Tasks Launched by a Trigger
 - For Tasks Launched by a Workflow
 - For Tasks Launched Manually
- Format for Using Variables
- Creating a Variable
- Creating a Global Variable
 - Global Variable Details
 - Global Variable Details Field Descriptions
- Creating a Variable Specific to a Trigger, Task, or Workflow

Overview

User-defined Opwise Controller variables are available for use in triggers, tasks, and Workflows.

You can define variables to be either:

- Available to a [single trigger, task, or workflow](#)
- Available to all triggers, tasks, and workflows; that is, [Global](#).

You define variables specific to single a trigger, task, or workflow on the **Variables** tab in the Details of that [trigger, task, or workflow](#). These variables are stored in the ***ops_local_variable*** table.

You define Global variables by either:

- Selecting **Other > Variables** from the [Automation Center](#) navigation pane.
- Using the [Set Variable](#) action for a task or workflow.

Global variables are stored in the ***ops_variable*** table.

Variable Naming Conventions

- Variable names must begin with a letter.
- Allowable characters are alphanumeric (upper or lower case), and underscore (_).
- White spaces are not permitted
- Variable names are not case-sensitive.



Warning

Do not define Controller variables with the prefix **ops_**. That prefix is reserved for built-in variables.

Resolving User-Defined Variables

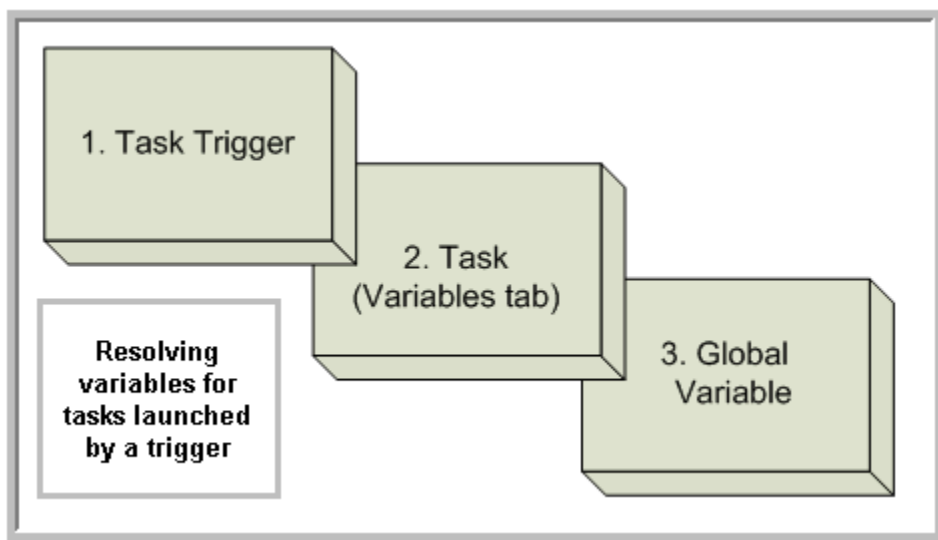
When the Controller creates a task instance from a task, it also resolves all variables specified in its free text fields. Because you can define variables at four different levels (trigger, task, workflow, and global), the Controller follows a prescribed formula to determine which variable takes precedence if duplicate variables have been defined. The general order of precedence, each of which may or may not exist in any given situation, is as follows:

1. Task trigger (highest precedence)
2. Task
3. Workflow trigger
4. Workflow
5. Global (lowest precedence)

The following scenarios provide more detailed information about how Controller variables are resolved.

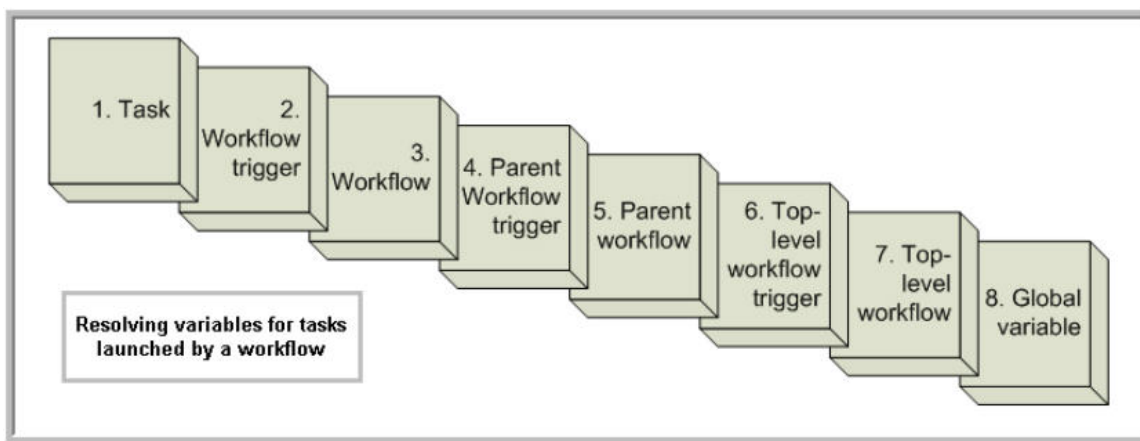
For Tasks Launched by a Trigger

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 Details 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.



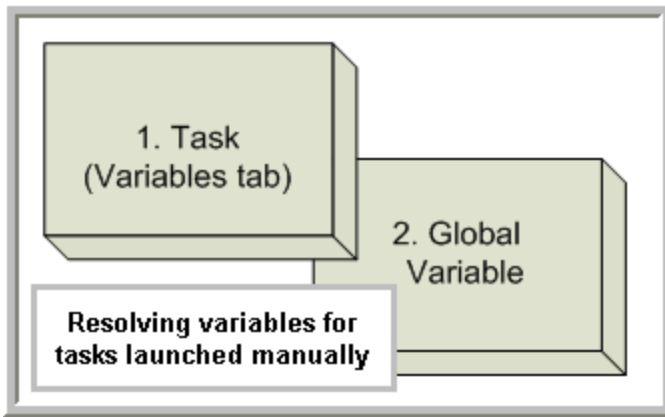
For Tasks Launched by a Workflow

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.
5. If the parent workflow's trigger does not define the variable or if there is no trigger, the value defined in the parent workflow is used.
6. If the parent workflow does not define the variable, the Controller checks up a level for the trigger on the next parent workflow.
7. If that trigger does not define the variable, it checks for variables associated with the workflow. (This continues until the top level workflow is reached.)
8. If the top-level workflow does not define the variable, the variable definition in the global variables table is used.
9. 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 curly braces ({ }). You can enter a series of variables or nested variables. Examples are:

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

Creating a Variable

You can create variables that are:

1. Available on a [Global](#) level; that is, available for all triggers, tasks, and Workflows.
2. Available only for a [specific trigger, task, or Workflow](#).

Creating a Global Variable

To create a Global variable that is available for all triggers, tasks, and Workflows:

Step 1 From the [Automation Center](#) navigation pane, select **Other > Variables**. The Variables list displays a list of all Global variables. (You also can define a Global variable by using the [Set Variable](#) action for a task or workflow.)

Below the list, Variable Details for a new Global variable displays.

The screenshot shows the 'Variables' interface. At the top, there is a 'Variables' tab with a dropdown menu showing '5 Variables'. Below this is a table with columns: Name, Value, Description, Updated By, and Updated. The table contains five rows of variables, all with a value of '1' and updated by users named 'stonebranch-user-01' through 'stonebranch-user-05' on 2014-06-13. Below the table is the 'Variable Details' section, which has tabs for 'Variable' and 'Versions'. The 'Variable' tab is active, showing a 'Details' form with fields for Name, Value, Description, and Member of Business Services. The 'Version' field is set to '1'. There are 'Save' and 'New' buttons at the bottom of the details form.

Name	Value	Description	Updated By	Updated
stonebranch_variable_01	1		stonebranch-user-01	2014-06-13 14:52:37 -0400
stonebranch_variable_02	1		stonebranch-user-02	2014-06-13 14:52:47 -0400
stonebranch_variable_03	1		stonebranch-user-03	2014-06-13 14:52:50 -0400
stonebranch_variable_04	1		stonebranch-user-04	2014-06-13 14:52:53 -0400
stonebranch_variable_05	1		stonebranch-user-05	2014-06-13 14:52:57 -0400

Step 2 Enter / select Details for a new Variable, using the field descriptions below as a guide.

- Required fields display in **boldface**.
- Default values for fields, if available, display automatically.

To display more of the Details fields on the screen, you can temporarily [hide the list](#).



Note

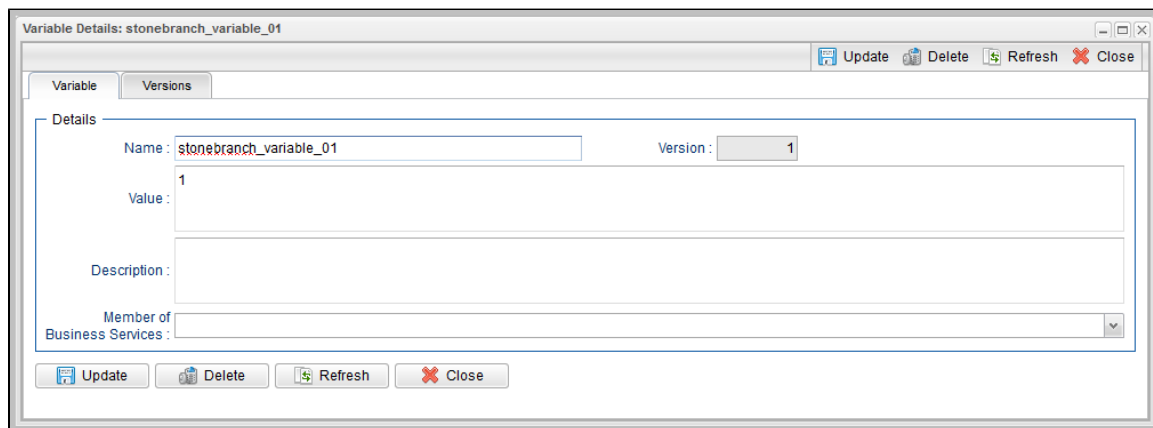
If you view [Global Variable Details](#) for an existing Global Variable by clicking a Variable in the list, and then want to create a new Global Variable, you must click the **New** button that displays above and below the Details.

Step 3 Click the **Save** button, or right-click in the Details and click **Save**, to save the record.

Global Variable Details


The following Variable Details is for an existing Global Variable.

See the [field descriptions](#) below for a description of all fields that display in the Global Variable Details.



Global Variable Details Field Descriptions

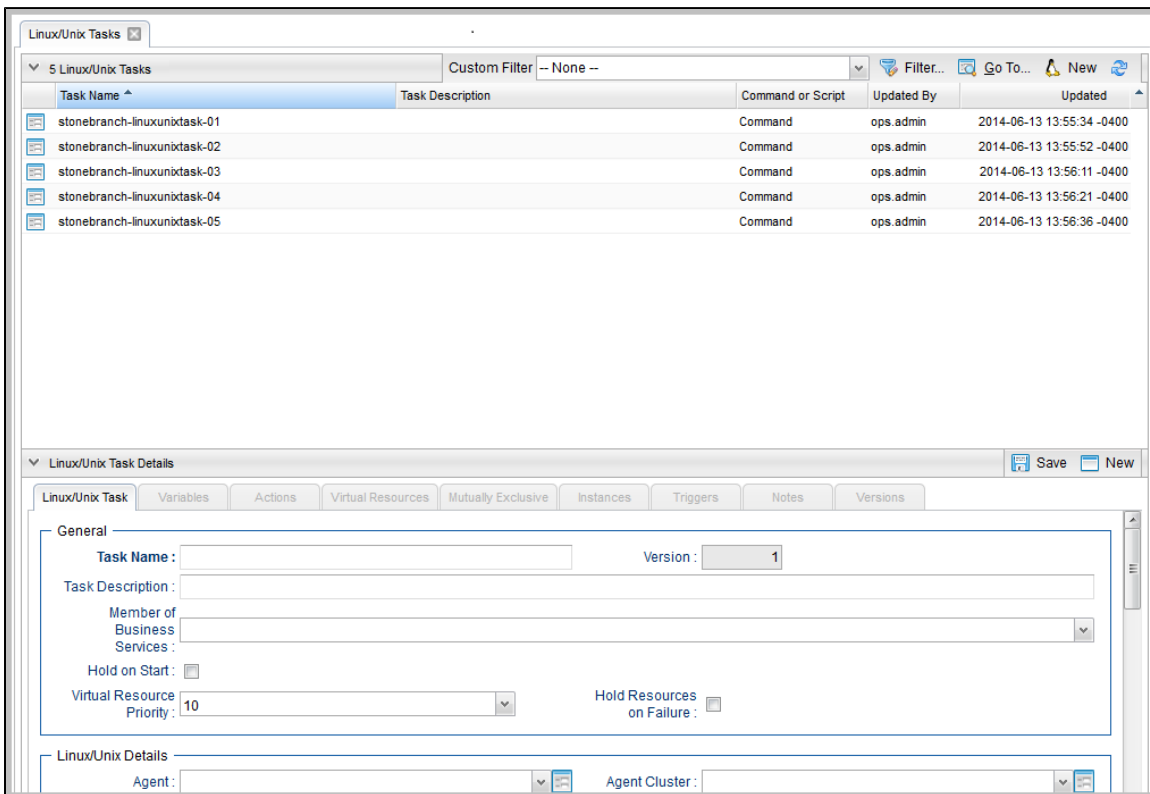
The following table describes the fields and buttons in the Variables Details.

Field Name	Description
Name	Name of the variable. Up to 40 alphanumeric. The name must begin with an alphabetic character and can consist of: alphas (a-z, A-Z), numerics 0-9, _ (underscore). White spaces are not permitted; names are not case-sensitive. <div style="background-color: #ffe6e6; padding: 10px; border: 1px solid #ccc;"> <p> Important Do not define variables with the prefix ops_. The ops_ prefix is reserved for built-in variables.</p> </div>
Version	System-supplied. The version number of the current record, which is incremented by the Controller every time a user updates a record. Click the Versions tab to view previous versions. For details, see Record Versioning .
Value	Value of the variable.
Description	Optional. Description of this variable.
Member of Business Services	User-defined; allows you to select one or more Business Services that this record belongs to.
Save	Saves a new variable record in the Controller database.
Update button	Saves updates to the record.
Delete button	Deletes the current record.
Refresh	Refreshes any dynamic data displayed in the Details.
Close	For pop-up view only; closes the pop-up view of this task.

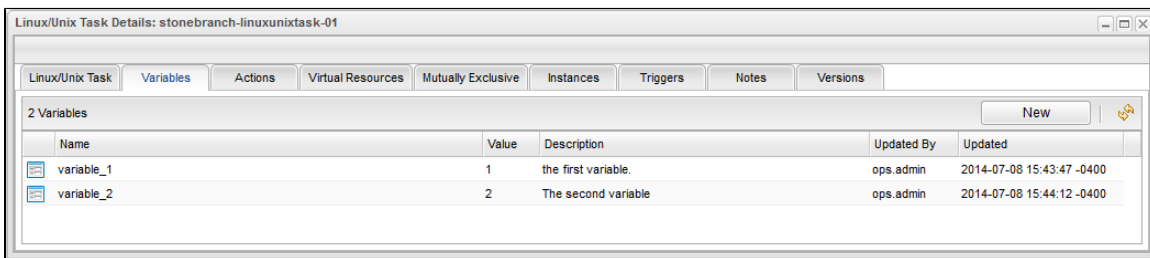
Creating a Variable Specific to a Trigger, Task, or Workflow

To create a variable that is specific to a single trigger, task, or Workflow:

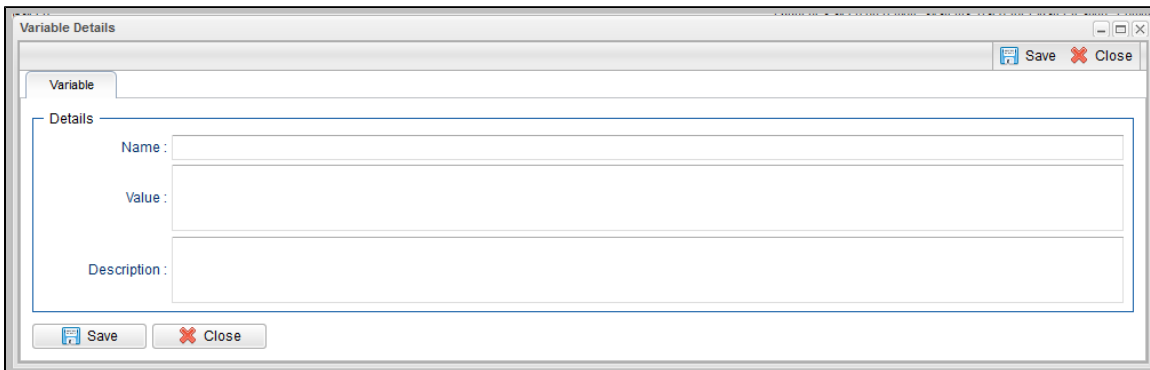
Step 1 From the Automation Center navigation pane, select **Trigger** > <trigger type> or **Tasks** > <task type>. The records list for that trigger or task type displays. For example:



Step 2 Open on the list and click the **Variables** tab to display a list of any currently defined variables specific to that record.



Step 3 Click the **New** button to display Variables Details for a new variable.



Step 4 Using the field descriptions provided for Global Variable Details as a guide, complete the fields as needed.

Step 5 Click the **Save** button, or right-click in the Details and click **Save**, to save the record.

Step 6 If appropriate, repeat these steps for any additional variables you want to add.

Built-In Variables

- Overview
- Agent Variables
 - Agent Hostname
 - Agent IP Address
 - Agent IP Address
 - Agent Mode
 - Agent Name
 - Agent Queue Name
- Application Monitor Trigger Variables
 - Trigger Application Name
 - Trigger Application Status
 - Trigger Application sys_id
 - Trigger Application Type
- Cluster Node Variables
 - Cluster Node Hostname
 - Cluster Node ID
 - Cluster Node IP Address
 - Cluster Node Mode
 - Cluster Node Name
 - Cluster Node Running Time
 - Cluster Node Start Time
- File Monitor Task Instance/Trigger Variables
 - Base File Name
 - File Directory
 - File Directory (with Final Directory Separator)
 - File Directory (without Final Directory Separator)
 - File Extension
 - Separator
 - Trigger File Date
 - Trigger File Group
 - Trigger File Name
 - Trigger File Name (No Path)
 - Trigger File Owner
 - Trigger File Scan Result
 - Trigger File Size
- File Transfer Task Instance Variables
 - Destination Password
 - Destination User ID
 - Source Password ID
 - Source User ID
- FTP File Monitor Task Instance Variables
 - Base Trigger File Name
 - Files Matching Wildcard
 - Remote Trigger File Name
 - Remote Trigger File Name (No Path)
 - Trigger File Directory
 - Trigger File Directory (with Final Directory Separator)
 - Trigger File Directory (without Final Directory Separator)
 - Trigger File Extension
 - Trigger Wildcard
 - Trigger Wildcard Path Only
 - Trigger Wildcard Path Only (without Final Slash)
- OMS Server Variables
 - Last OMS Server Connected
 - OMS Server IP Address
 - OMS Server Status
 - OMS Server sys_id
- SAP Task Instance Variables
 - SAP InfoPackage Request ID
 - SAP Job ID
 - SAP Job Name
 - SAP Process Chain ID
 - SAP Process Chain Log ID
- SQL and Stored Procedure Task Instance Variables
 - Error Message
 - Processed Rows
 - Return Code for SQL Statement Outcome

- System Monitor Task Instance Variables
 - Actual Size
 - Actual Size (Rounded)
 - Actual Size (Scale)
 - Scale
 - Size
 - Size (Rounded)
- Task Instance Variables
 - Command
 - Command Parameters
 - End Time
 - Execution User ID
 - Launch Time
 - Maximum Retry Count
 - Parent Workflow Instance sys_id
 - Parent Workflow Name
 - Retry Count
 - Retry Interval
 - Running Time
 - Running Time (Text Format)
 - Script ID
 - Script Name
 - Starting Time
 - Task Instance Attempts
 - Task Instance Exit Code
 - Task Instance Status
 - Task Instance sys_id
 - Task Name
 - Task Reference Count
 - Task Type
- Task Monitor Task Instance/Trigger Variables
 - Trigger Task Name
 - Trigger Task Status
 - Trigger Task sys_id
 - Trigger Task Type
 - Trigger Workflow
- Trigger Variables
 - Trigger Name
 - Trigger Time
- z/OS Task Instance Variables
 - JCL Location
 - Job Number
 - Override JCL Location
 - Submitted JCL Location

Overview

Built-in variables are maintained by Opwise Controller and provide information about task instances, agents, Opwise Message Service (OMS), and cluster nodes. They can be used in free text fields in triggers, tasks, task actions, and email notifications for agents, OMS servers, and cluster nodes.

Supported built-in variables and their descriptions are provided below. All built-in variables are prefixed with `ops_`.

Agent Variables

The following agent variables can be used to pass information into an [Agent notification](#).

Some of these variables, as noted, also can be used to pass agent information into an agent-based task (Windows, Linux/Unix, z/OS, and SAP).

Agent Hostname

Description	Resolves to the agent hostname. You also can use this variable in task notifications; see Creating Email Notifications and Creating SNMP Notifications .
Syntax	<code>\${ops_agent_hostname}</code>
Example	

Agent IP Address

Description	Resolves to the agent IP address (see also <code>\${ops_agent_ip}</code>).
Syntax	<code>\${ops_agent_ipaddr}</code>
Example	

Agent IP Address

Description	Resolves to the agent IP address. You also can use this variable in task notifications; see Creating Email Notifications and Creating SNMP Notifications .
Syntax	<code>\${ops_agent_ip}</code>
Example	

Agent Mode

Description	Resolves to the agent operational mode (Active, Offline).
Syntax	<code>\${ops_agent_mode}</code>
Example	

Agent Name

Description	Resolves to the agent name. You also can use this variable in task notifications; see Creating Email Notifications and SNMP Notification Actions .
Syntax	<code>\${ops_agent_name}</code>
Example	

Agent Queue Name

Description	Resolves to the agent queue name. You also can use this variable in task notifications; see Creating Email Notifications and Creating SNMP Notifications .
Syntax	<code>\${ops_agent_id}</code>
Example	

Application Monitor Trigger Variables

When a task is launched by an [Application Monitor trigger](#), the following built-in variables are passed into the task being launched by the trigger:

Trigger Application Name

Description	Resolves to the name of the Application being monitored by the trigger.
Syntax	<code>\${ops_trigger_appl_name}</code>
Example	

Trigger Application Status

Description	Resolves to the status of the Application being monitored by the trigger.
--------------------	---------------------------------------------------------------------------

Syntax	<code>\${ops_trigger_appl_status}</code>
Example	

Trigger Application sys_id

Description	Resolves to the <code>sys_id</code> of the application.
Syntax	<code>\${ops_trigger_appl_id}</code>
Example	

Trigger Application Type

Description	Resolves to the type of Application being monitored by the trigger, as defined by the Application Type field.
Syntax	<code>\${ops_trigger_appl_type}</code>
Example	

Cluster Node Variables

The following cluster node variables allow you to pass information into a cluster node (Controller server) notification:

Cluster Node Hostname

Description	Resolves to the hostname of this cluster node.
Syntax	<code>\${ops_cluster_hostname}</code>
Example	<pre>ops_cluster_hostname = MACHINEC19A</pre>

Cluster Node ID

Description	Resolves to the cluster node's internally-generated build ID.
Syntax	<code>\${ops_cluster_id}</code>
Example	<pre>ops_cluster_id = MACHINEC19A:8080-opwise</pre>

Cluster Node IP Address

Description	Resolves to the IP address of this cluster node.
Syntax	<code>\${ops_cluster_ipaddr}</code>
Example	<pre>ops_cluster_ipaddr = 10.N.N.NN</pre>

Cluster Node Mode

Description	Resolves to the current mode of this cluster node (Offline, Active, Passive). For more information, see Viewing Node Status .
Syntax	<code>\${ops_cluster_mode}</code>
Example	<pre>ops_cluster_mode = Active</pre>

Cluster Node Name

Description	<code>\${ops_cluster_name}</code> is an alias for the <code>\${ops_cluster_id}</code> variable.
Syntax	<code>\${ops_cluster_name}</code>
Example	<pre>ops_cluster_name = MACHINEC19A:8080-opwise</pre>

Cluster Node Running Time

Description	Resolves to the numbers of days, hours, and minutes that this cluster node has been running since it was last started.
Syntax	<code>\${ops_cluster_uptime}</code>
Example	<pre>ops_cluster_uptime = 7 Seconds</pre>

Cluster Node Start Time

Description	Resolves to the date and time the cluster node (server) was started.
Syntax	<code>\${ops_cluster_start_time}</code>
Example	<pre>ops_cluster_start_time = 2011-09-26 17:35:01 -0400</pre>

File Monitor Task Instance/Trigger Variables

When one or more tasks are launched by a [File Monitor trigger](#) after the conditions in its associated File Monitor task are met, the built-in variables described below are passed into the tasks being launched by the trigger.

For example, the File Monitor trigger may specify the launch of a Windows task each time the associated File Monitor task detects the creation of a specific file. The Windows task might use one of these built-in variables as a command argument. Or, if the File Monitor task is not associated with a trigger but is running within a workflow, on completion you can propagate one or more of these built-in variable values to the parent workflow level using the [Set Variable](#) action. This allows you to pass information from the File Monitor task to a successor task within the same workflow hierarchy.

Base File Name

Description	Resolves to the base file name.
Syntax	<code>\${ops_trigger_file_name_simple}</code>

File Directory

Description	Resolves to the directory where the new file was created, but not the file itself. If the existence or non-existence of the final directory separator is a requirement, we recommend the use of <code>\${ops_trigger_file_fullpath}</code> and <code>\${ops_trigger_file_fullpath_no_separator}</code> , respectively.
Syntax	<code>\${ops_trigger_file_path}</code>
Example	
Example	

File Directory (with Final Directory Separator)

Description	Resolves to the directory where the new file was created, but not the file itself; includes the final directory separator.
Syntax	<code>\${ops_trigger_file_fullpath}</code>
Example	

File Directory (without Final Directory Separator)

Description	Resolves to the directory where the new file was created, but not the file itself; does not include the final directory separator.
Syntax	<code>\${ops_trigger_file_fullpath_no_separator}</code>
Example	

File Extension

Description	Resolves to the file extension of a file.
Syntax	<code>\${ops_trigger_file_name_extension}</code>
Example	

Separator

Description	Resolves to the separator appropriate to the platform where the agent is running. For Windows, resolves to a backslash (\); for Linux/Unix, resolves to forward slash (/). This variable may be useful if you want to piece together a pathname using a combination of text and variables.
Syntax	<code>\${ops_trigger_file_separator}</code>
Example	<pre> <code>\${ops_trigger_file_fullpath}sub_folder_name \${ops_trigger_file_separator}filename.txt</code> </pre>

Trigger File Date

Description	Resolves to the file date of the file that fired the trigger.
Syntax	<code>\${ops_trigger_file_date}</code>
Example	

Trigger File Group

Description	Resolves to the file group of the file that fired the trigger.
--------------------	----------------------------------------------------------------

Syntax	<code>\${ops_trigger_file_group}</code>
Example	

Trigger File Name

Description	Resolves to the name of the file that fired the trigger.
Syntax	<code>\${ops_trigger_file_name}</code>
Example	

Trigger File Name (No Path)

Description	Resolves to the name of the file that fired the trigger, but without any path information.
Syntax	<code>\${ops_trigger_file_name_nopath}</code>
Example	

Trigger File Owner

Description	Resolves to the file owner of the file that fired the trigger.
Syntax	<code>\${ops_trigger_file_owner}</code>
Example	

Trigger File Scan Result

Description	Resolves to the result of the file scan: FOUND or NOT_FOUND.
Syntax	<code>\${ops_trigger_file_scan}</code>
Example	

Trigger File Size

Description	Resolves to the file size of the file that fired the trigger.
Syntax	<code>\${ops_trigger_file_size}</code>
Example	

File Transfer Task Instance Variables

File Transfer variables are available for use in [UDM scripts](#).



Note

These variables differ from all other built-in variables in that they are resolved by Universal Data Mover (UDM) on a UDM agent, not by the Opwise Controller. File Transfer variables are sent to an agent unresolved and UDM performs all resolution for them. The resolved value is never available to the Controller.

Unlike the syntax of built-in variables resolved by Opwise Controller - `${<variable-name>}` - the syntax of File Transfer variables is the same as all [UDM variables](#) - `$(<variable-name>)`.

The following example illustrates the correct way to code them:

```
open src=srcserver user=${ops_src_cred_user} pwd=${ops_src_cred_pwd} dst=dstserver
user=${ops_dst_cred_user} pwd=${ops_dst_cred_pwd}
```

Destination Password

Description	Resolves to the destination password.
Syntax	\$(ops_dst_cred_pwd)
Example	

Destination User ID

Description	Resolves to the destination user ID.
Syntax	\$(ops_dst_cred_user)
Example	

Source Password ID

Description	Resolves to the source password.
Syntax	\$(ops_src_cred_pwd)
Example	

Source User ID

Description	Resolves to the source user ID.
Syntax	\$(ops_src_cred_user)
Example	

FTP File Monitor Task Instance Variables

The following built-in variables are available for FTP File Monitor task instances and provide information about the file or file(s) that matched the monitor's criteria.

You can use these variables in an FTP File Monitor action or in a successor task instance by propagating one or more of these built-in variable values to a parent workflow using the [Set Variable](#) action.

Base Trigger File Name

Description	Resolves to the base file name.
Syntax	\${ops_trigger_file_name_simple}
Example	

Files Matching Wildcard

Description	Resolves to a comma-separated list of files that matched the wildcard, if one was specified in the Remote Filename field in the FTP File Monitor task.
Syntax	\${ops_trigger_files}

Example

```
ops_trigger_files = COMPANY-2011-11-22.xls, COMPANY-2011-11-23.xls,COMPANY-2011-11-24.xls
```

Remote Trigger File Name

Description	Resolves to the remote file name.
Syntax	<code>\${ops_trigger_file_name}</code>
Example	

Remote Trigger File Name (No Path)

Description	Resolves to the remote file name without any path information.
Syntax	<code>\${ops_trigger_file_name_nopath}</code>
Example	

Trigger File Directory

Description	Resolves to the directory where the remote file is located, but not the file itself. <code>\${ops_trigger_file_path}</code> is an alias for <code>\${ops_trigger_file_fullpath_no_separator}</code> .
Syntax	<code>\${ops_trigger_file_path}</code>
Example	

Trigger File Directory (with Final Directory Separator)

Description	Resolves to the directory where the remote file is located, but not the file itself; includes the final directory separator.
Syntax	<code>\${ops_trigger_file_fullpath}</code>
Example	

Trigger File Directory (without Final Directory Separator)

Description	Resolves to the directory where the remote file is located, but not the file itself; does not include the final directory separator.
Syntax	<code>\${ops_trigger_file_fullpath_no_separator}</code>
Example	

Trigger File Extension

Description	Resolves to the file extension of the file.
Syntax	<code>\${ops_trigger_file_name_extension}</code>
Example	

Trigger Wildcard

Description	Resolves to the contents of the Remote Filename field in the FTP File Monitor task.
Syntax	<code>\${ops_trigger_wildcard}</code>

Example	<pre>ops_trigger_wildcard = /home/prod/stonebranch/COMPANY*.xls</pre>
----------------	-----------------------------------------------------------------------

Trigger Wildcard Path Only

Description	Resolves to the path only, with the final slash but without the file name, from the Remote Filename field in the FTP File Monitor task.
Syntax	<code>\${ops_trigger_wildcard_path}</code>
Example	<pre>ops_trigger_wildcard_path = /home/prod/stonebranch/</pre>

Trigger Wildcard Path Only (without Final Slash)

Description	Resolves to the path only, without the final slash and without the file name, from the Remote Filename field in the FTP File Monitor task.
Syntax	<code>\${ops_trigger_wildcard_path_no_separator}</code>
Example	<pre>ops_trigger_wildcard_path_no_separator = /home/prod/stonebranch</pre>

OMS Server Variables

The following OMS Server variables allow you to pass information into an [OMS Server](#) notification.

Last OMS Server Connected

Description	Resolves to the last OMS Server connected to the Controller in an OMS HA cluster.
Syntax	<code>\${ops_oms_last_connected}</code>
Example	

OMS Server IP Address

Description	Resolves to the OMS Server IP address.
Syntax	<code>\${ops_oms_server_address}</code>
Example	

OMS Server Status

Description	Resolves to the current status of the OMS Server.
Syntax	<code>\${ops_oms_status}</code>
Example	

OMS Server sys_id

Description	Resolves to the <code>sys_id</code> of the OMS server.
Syntax	<code>\${ops_oms_id}</code>
Example	

SAP Task Instance Variables

For an SAP task instance, where applicable, the following built-in variables resolve to the SAP jobname and SAP jobid of the job running in the SAP system. If you need to use the SAP jobname and/or the SAP jobid from one SAP task instance in a successor SAP task instance, you can use the [Set Variable](#) action to propagate these built-in variable values to the parent workflow.

SAP InfoPackage Request ID

Description	Resolves to the SAP InfoPackage Request ID.
Syntax	<code>\${ops_sap_requestid}</code>
Example	

SAP Job ID

Description	Resolves to the SAP job ID.
Syntax	<code>\${ops_sap_jobid}</code>
Example	

SAP Job Name

Description	Resolves to the SAP job name.
Syntax	<code>\${ops_sap_jobname}</code>
Example	

SAP Process Chain ID

Description	Resolves to the SAP Process Chain ID.
Syntax	<code>\${ops_sap_chainid}</code>
Example	

SAP Process Chain Log ID

Description	Resolves to the SAP Process Chain Log ID.
Syntax	<code>\${ops_sap_logid}</code>
Example	

SQL and Stored Procedure Task Instance Variables

The following built-in variables are used in [SQL](#) tasks and [Stored Procedure](#) tasks to collect `SQLException` data, if any:

Error Message

Description	Resolves to any error message generated by the database.
--------------------	----------------------------------------------------------

Syntax	<code>\${ops_sql_error_msg}</code>
Example	

Processed Rows

Description	Resolves to the number of rows processed.
Syntax	<code>\${ops_sql_rows}</code>
Example	

Return Code for SQL Statement Outcome

Description	Resolves to a return code that indicates the outcome of the most recently executed SQL statement.
Syntax	<code>\${ops_sql_state}</code>
Example	

System Monitor Task Instance Variables

The following System Monitor variables show the results for **Resource Available** and **Actual Available** that can be utilized in [System Monitor](#) tasks.

Actual Size

Description	Actual size determined by the agent.
Syntax	<code>\${ops_sm_actual_size}</code>
Example	

Actual Size (Rounded)

Description	Same as <code>ops_sm_actual_size</code> , except rounded to the nearest integer.
Syntax	<code>\${ops_sm_actual_int_size}</code>
Example	

Actual Size (Scale)

Description	Scale of the actual size determined by the agent.
Syntax	<code>\${ops_sm_actual_scale}</code>
Example	

Scale

Description	Scale specified in the By Scale field for Resource Available of the System Monitor task definition.
Syntax	<code>\${ops_sm_scale}</code>
Example	

Size

Description	Size specified in the Resource Available field of the System Monitor task definition.
Syntax	\${ops_sm_size}
Example	

Size (Rounded)

Description	Same as ops_sm_size, except that ops_sm_int_size is rounded to the nearest integer.
Syntax	\${ops_sm_int_size}
Example	

Task Instance Variables

The following built-in variables are associated with task instances for [all task types](#):

Command

Description	For tasks that launch a command on a Windows, Linux/Unix, or z/OS machine; resolves to the task command.
Syntax	\${ops_cmd}
Example	

Command Parameters

Description	For tasks that launch a command on a Windows, Linux/Unix, or z/OS machine; resolves to the task command parameters.
Syntax	\${ops_cmd_parms}
Example	

End Time

Description	Resolves to the task ending time.
Syntax	\${ops_end_time}
Example	

Execution User ID

Description	Resolves to the ID of the user who launched the task or to the ID of the user who enabled the trigger that launched the task.
Syntax	\${ops_execution_user}
Example	

Launch Time

Description	Resolves to the task launch time. For workflows, all descendants will have the same launch time as the top-level workflow.
Syntax	\${ops_launch_time}
Example	

Maximum Retry Count

Description	Resolves to the maximum retry count.
Syntax	<code>\${ops_retry_maximum}</code>
Example	

Parent Workflow Instance `sys_id`

Description	Resolves to the <code>sys_id</code> of the parent workflow task instance.
Syntax	<code>\${ops_workflow_id}</code>
Example	

Parent Workflow Name

Description	Resolves to the name of the parent workflow.
Syntax	<code>\${ops_workflow_name}</code>
Example	

Retry Count

Description	Resolves to the current retry count.
Syntax	<code>\${ops_retry_count}</code>
Example	

Retry Interval

Description	Resolves to the retry interval (seconds).
Syntax	<code>\${ops_retry_interval}</code>
Example	

Running Time

Description	Resolves to the task running time in milliseconds.
Syntax	<code>\${ops_duration}</code>
Example	<code>ops_duration = 130000</code>

Running Time (Text Format)

Description	Resolves to the task running time in a more readable representation of the duration time.
Syntax	<code>\${ops_duration_text}</code>
Example	<code>ops_duration_text = 2 Minutes 10 Seconds)</code>

Script ID

Description	For Windows, Linux/Unix, and SAP tasks where a Script or SAP Definition from Scripts is specified; resolves to the Controller system ID of the script.
Syntax	<code>\${ops_script_id}</code>

Example**Script Name**

Description	For Windows, Linux/Unix, and SAP tasks where a Script or SAP Definition from Scripts is specified; resolves to the Controller name of the script.
Syntax	<code>\${ops_script_name}</code>
Example	

Starting Time

Description	Resolves to the task starting time.
Syntax	<code>\${ops_start_time}</code>
Example	

Task Instance Attempts

Description	Resolves to the current task instance attempt count. Each Re-run operation increments the attempt. Initial attempt is 1.
Syntax	<code>\${ops_attempt}</code>
Example	

Task Instance Exit Code

Description	Resolves to the task instance exit code, if any.
Syntax	<code>\${ops_exit_code}</code>
Example	

Task Instance Status

Description	Resolves to the current task instance status.
Syntax	<code>\${ops_status}</code>
Example	

Task Instance sys_id

Description	Resolves to the <code>sys_id</code> of the task instance.
Syntax	<code>\${ops_task_id}</code>
Example	

Task Name

Description	Resolves to the task name.
Syntax	<code>\${ops_task_name}</code>
Example	

Task Reference Count

Description	Resolves to the current task reference count. Each time an instance is created from a specific task, it gets a unique task reference count for that task. For example, if you launch a task twice, the first instance will have task reference count 1, and the second will have task reference count 2.
Syntax	<code>\${ops_task_ref_count}</code>
Example	

Task Type

Description	Resolves to the task type.
Syntax	<code>\${ops_task_type}</code>
Example	

Task Monitor Task Instance/Trigger Variables

When the conditions of a Task Monitor task are met and its associated [Task Monitor trigger](#) launches one or more tasks, the following built-in variables are passed into the task instances being launched by the trigger.

For example, the Task Monitor trigger may specify an Email task that will launch each time the conditions in the associated Task Monitor task are met. You might want to specify one or more of these variables in the body of the email.

If the Task Monitor task is not associated with a trigger but is running within a workflow, on completion you can propagate one or more of these built-in variable values to the parent workflow level by using the [Set Variable](#) action. This allows you to pass information from the Task Monitor task to a successor task within the same workflow hierarchy.

Trigger Task Name

Description	Resolves to the name of the task instance that fired the trigger.
Syntax	<code>\${ops_trigger_task_name}</code>
Example	

Trigger Task Status

Description	Resolves to the status of the task instance that fired the trigger.
Syntax	<code>\${ops_trigger_task_status}</code>
Example	

Trigger Task sys_id

Description	Resolves to the sys_id of the task instance that fired the trigger.
Syntax	<code>\${ops_trigger_task_id}</code>
Example	

Trigger Task Type

Description	Resolves to the type of the task instance that fired the trigger.
Syntax	<code>\${ops_trigger_task_type}</code>
Example	

Trigger Workflow

Description	Resolves to the name of the workflow instance that fired the trigger. This variable is available only for a Task Monitor task that has a Workflow Condition specified. If a workflow condition is specified, <code>\${ops_trigger_workflow_name}</code> will resolve to the name of the workflow instance that the workflow condition matched.
Syntax	<code>\${ops_trigger_workflow_name}</code>
Example	

Trigger Variables

The following built-in variables are associated with [all trigger types](#):

When a task is launched by a trigger, the values of the following built-in variables, if they are specified in the task, are passed into the task instance.

Trigger Name

Description	Resolves to the name of the trigger that launched the task instance.
Syntax	<code>\${ops_trigger_name}</code>
Example	

Trigger Time

Description	Resolves to the scheduled time of the trigger or, if the trigger is not scheduled, the actual trigger time. If the task is triggered by date/time, it resolves to that specified date/time.
Syntax	<code>\${ops_trigger_time}</code>
Example	

z/OS Task Instance Variables

The following built-in variables are available for z/OS task instances:

JCL Location

Description	Resolves to the file and member name containing the JCL script.
Syntax	<code>\${ops_jcl_location}</code>
Example	

Job Number

Description	Resolves to the job number assigned to the job by JES.
Syntax	<code>\${ops_job_id}</code>
Example	

Override JCL Location

Description	Resolves to the file and member name of the JCL location containing a potential override JCL script.
--------------------	------------------------------------------------------------------------------------------------------

Syntax	<code>\${ops_override_jcl_location}</code>
Example	

Submitted JCL Location

Description	Resolves to the file and member name of the JCL location that was actually used for job submission.
Syntax	<code>\${ops_submitted_jcl_location}</code>
Example	

Launching With Variables

For information on how to launch a task with variables, see [Provide Temporary Variable Values and Launch a Task Manually on the Manually Running and Controlling Tasks](#) page.

Trigger With Variables

For information on how to use variables when manually launching tasks associated with a trigger, see [Triggering with Variables](#) (in the [Triggers and Calendars](#) section of this documentation).

Creating a Set Variable Action within a Task or Workflow

- [Overview](#)
- [Variables and Variable Scope](#)
- [Creating a Set Variable Action](#)
- [Set Variable Details Field Descriptions](#)

Overview

The Set Variable action allows you to set a variable to a specific value for a task or workflow, and to select a scope (level of usage) for that variable (see [Variables and Variable Scope](#), below). Unless you set the [scope of the variable](#) to **GLOBAL**, which specifies that the variable can be accessed at any time by any task, workflow, or trigger, the value exists in memory only for the time that the task or workflow is running, or until another Set Variable action sets the variable to another value.



Note

Variables with a Variable Scope set to **GLOBAL** are added to the list of global variables on the [Variables list \(Automation Center > Other > Variables\)](#) after the task or workflow is run.

You can use the Set Variable action to create a new variable or modify an existing variable.

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

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

Variables and Variable Scope

A variable defined for a task under the **Variables** tab for that task is used only by that task.

A variable defined for a workflow under the **Variables** tab for that workflow is available for any task in that workflow; a task will use the variable value defined for the workflow unless the variable is defined for that task.

A variable defined for a task or workflow on a Set Variable action screen let you specify, in the [Variable Scope](#) field, the scope of that variable. You can specify that a variable be available for:

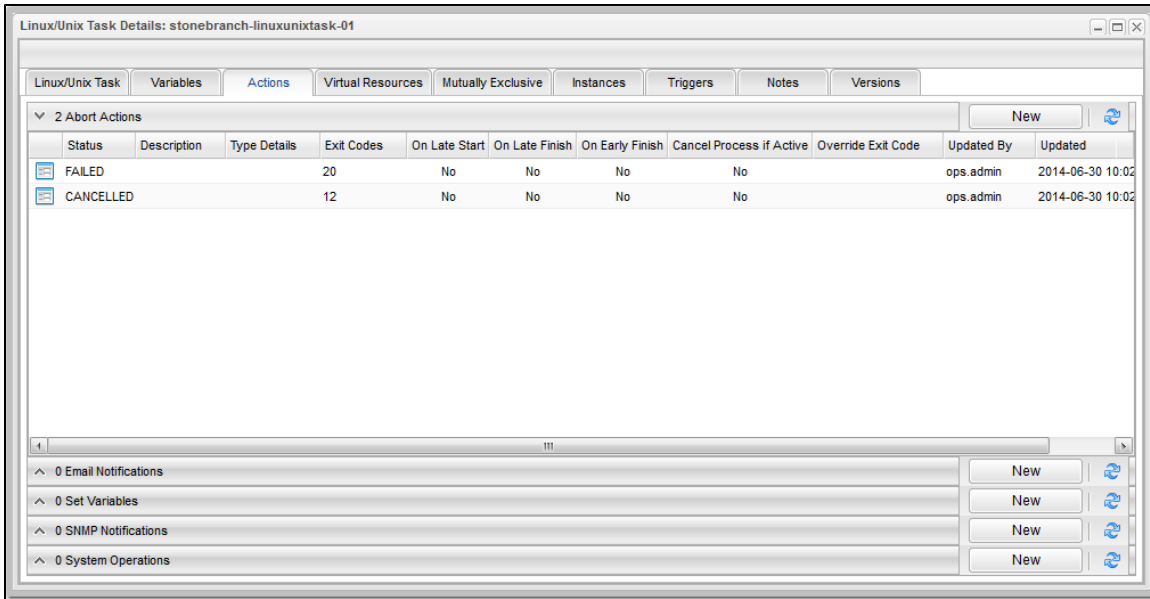
- Only the task where it is set.
- All tasks within the task's parent (immediate) workflow.
- All tasks within the task's top-level parent workflow.
- All tasks and workflow instances.

For example, if you set a variable for a task to be available within the scope of its parent workflow, the value of that variable is propagated up to the parent workflow level. As each task in the workflow is run, that value is available for that task.

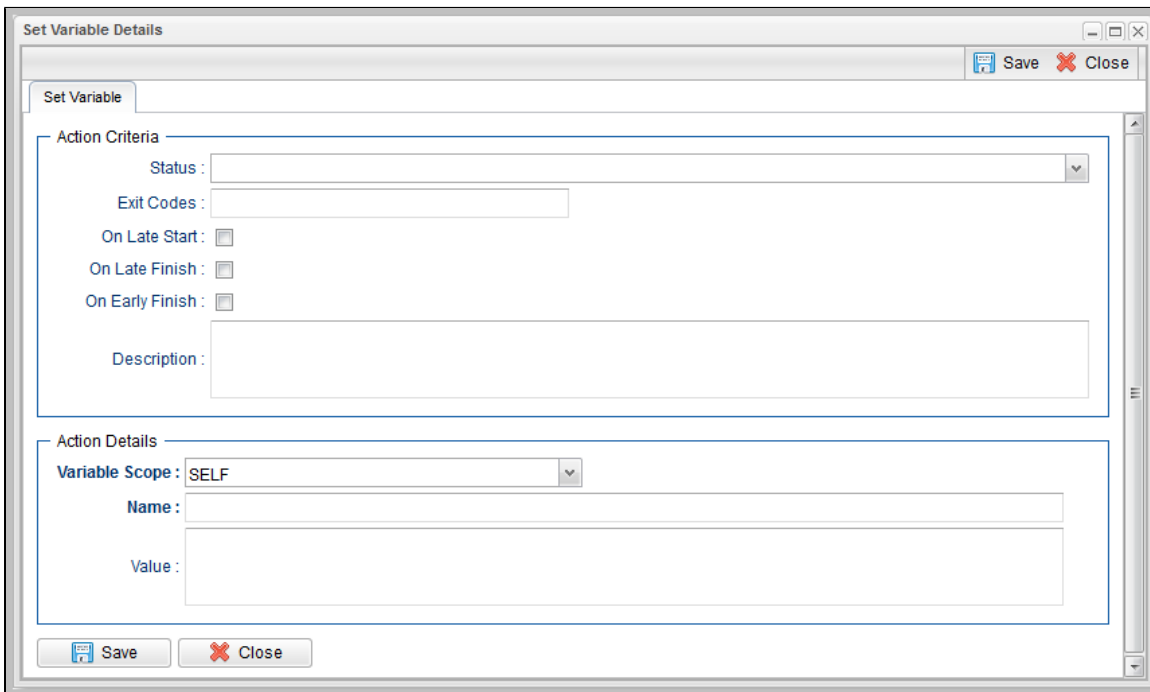
Creating a Set Variable Action

Step 1 Display the Task Details of the task for which you are creating the Set Variable action.

Step 2 Click the **Actions** tab. A list of any defined Actions for that task displays.



Step 3 Click the **New** button that displays on the Set Variables row. The Set Variable Details pop-up displays.



Step 4 Using the field descriptions below as a guide, complete the fields as needed.


Step 5 Click the **Save** button, or right-click in the Details and then click **Save**, to save the record and return to the Actions List.

Step 6 If appropriate, repeat these steps for any additional Set Variable actions you want to create.

Set Variable Details Field Descriptions

The table below describes the fields and buttons in the Set Variable Details.

Field Name	Description
------------	-------------

Action Criteria	This section contains criteria for performing the action.															
Type Details	Displays - on the Set Variables actions list - the Variable Scope , Name , and Value for this action.															
Action Inheritance	For Workflow tasks only; the records this action applies to. Options: <ul style="list-style-type: none"> • SELF - This action applies only to the workflow; it is not inherited by its children tasks. • SELF/CHILDREN - This action applies to the workflow and its contained tasks (children). • CHILDREN - This action applies only to the tasks within the workflow (children). 															
Status	The status that will trigger the action. To trigger a Set Variable action, you can specify status only, or status and exit code. You can specify as many statuses as needed.															
Exit Codes	Specifies one or more exit codes that will trigger the event. 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	Generates the action or notification if the task started late, based on the Late Start Time specified in the task.															
On Late Finish	Generates the action or notification if the task finishes late, based on the Late Finish time specified in the task.															
On Early Finish	Generates the action or notification if the task finishes early, based on the Early Finish Time specified in the task.															
Description	Description of this action.															
Action Details	This section contains additional details about the action.															
Variable Scope	Applies to variables associated with a task in a workflow. Options: <table border="1" data-bbox="263 1228 1484 1633"> <thead> <tr> <th>Scope</th> <th>Scope Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>SELF</td> <td>1</td> <td>The variable is set only within the scope of the task that executes the Set Variable action.</td> </tr> <tr> <td>PARENT</td> <td>2</td> <td>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.</td> </tr> <tr> <td>TOP_LEVEL_PARENT</td> <td>3</td> <td>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.</td> </tr> <tr> <td>GLOBAL</td> <td>4</td> <td>The variable is set at the global variable level and, as such, is accessible by any task, workflow, or trigger. If the global variable is not already defined, it will be created.</td> </tr> </tbody> </table>	Scope	Scope Value	Description	SELF	1	The variable is set only within the scope of the task that executes the Set Variable action.	PARENT	2	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	3	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.	GLOBAL	4	The variable is set at the global variable level and, as such, is accessible by any task, workflow, or trigger. If the global variable is not already defined, it will be created.
Scope	Scope Value	Description														
SELF	1	The variable is set only within the scope of the task that executes the Set Variable action.														
PARENT	2	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	3	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.														
GLOBAL	4	The variable is set at the global variable level and, as such, is accessible by any task, workflow, or trigger. If the global variable is not already defined, it will be created.														
Name	Name of the variable. Up to 40 alphanumeric. The name must begin with an alphabetic character and can consist of: alphas (a-z, A-Z), numerics 0-9, _ (underscore). White spaces are not permitted; names are not case-sensitive. <div data-bbox="305 1795 1445 1906" style="background-color: #ffe6e6; padding: 10px;">  Important Do not define variables with the prefix ops_. The ops_ prefix is reserved for built-in variables. </div>															

Value	Value of the variable.
Buttons	This section identifies the buttons displayed above and below the Action Details that let you perform various actions.
Save	Saves a new Action record in the Controller database.
Update	Saves updates to the record.
Delete	Deletes the current record.
Refresh	Refreshes any dynamic data displayed in the Details.
Close	Closes the Details pop-up of this action.

Listing and Setting Variables from the Command Line

To list and set variables from the command line, use the [List Variables](#) (ops-variable-list) and [Set Variables](#) (ops-variable-set) commands of the Opwise Controller [Command Line Interface \(CLI\)](#).

Functions

- Overview
- Formatting Rules
- Function Categories
- Date Functions
 - Return Nth Business Day of Month
 - Return Number of Business Days between Dates
 - Resolve to Current Unix Epoch Time
 - Resolve to Current Date and Time
 - Resolve to Current Date and Time (Advanced)
 - Return Nth Day of Month
 - Return Day of Week
 - Return Days between Dates
 - Return Date with Offsets
 - Return Date with Offsets (Advanced)
 - Return Non-Business Day of Month
- Mathematical Functions
 - Return Modulo
 - Add
 - Subtract
 - Multiply
 - Divide
 - Return Absolute Value
- SQL/Stored Procedure Functions
 - Return SQL Results from Current Task
 - Return SQL Results from Sibling Task
 - Return String Value of Row/Column by Column Name
 - Return String Value of Row/Column by Column Number
 - Return Column Names for SQL Results from Current Task
 - Return Column Names for SQL Results from Sibling Task
 - Return String Values of Columns
 - Return SQL Warnings from Current Task
 - Return SQL Warnings from Sibling Task
- String Functions
 - Return Index of Substring in String Value
 - Return Index of Substring Plus Offset in String Value
 - Return Index of Rightmost Occurrence of Substring in String Value
 - Return Index of Rightmost Occurrence of Substring Plus Offset in String Value
 - Return Length of Value
 - Replace Substring of Value with Regular Expression
 - Return New String that is Substring of Value
 - Convert Characters in Value to Lower Case
 - Convert Characters in Value to Upper Case
 - Return Copy of Value with Whitespace Omitted
 - Return Index of Substring within String Variable
 - Return Index of Substring Plus Offset in String Variable
 - Return Index of Rightmost Occurrence of Substring in String Variable
 - Return Index of Rightmost Occurrence of Substring Plus Offset in String Variable
 - Return Length of Variable
 - Replace Substring of Variable with Regular Expression
 - Return New String that is Substring of Variable
 - Convert Characters in Variable to Lower Case
 - Convert Characters in Variable to Upper Case
 - Return Copy of variable
- System Functions
 - Resolve to GUID (Globally Unique ID)
 - Resolve to Host Name
 - Resolve to IP Address
 - Generate Random Number
 - Resolve Variable
 - Resolve Variable (Advanced)
 - Display Variables
 - Resolve to SYS_ID
 - Resolve to Variable Value

Overview

Variables and functions can be used in free-text fields within tasks and workflows. When a variable or function is specified in a free-text field, the Controller inserts its value into the field when the task or workflow is run.

Also, triggers can pass variables and functions into the tasks and workflows they launch.

Opswise Controller supports a number of functions that can be specified in free-text fields. They are resolved when a task instance runs or when a [Set Variable](#) action containing a function is executed.

Functions are entered using the following formats:

```
$_function
$_function(arg1, ..., argN)
```

Formatting Rules

- Functions must be written either in all lower case or exactly as shown in the tables on this page.
- Any parameter can be quoted. Strings must be quoted with single or double quotation marks.
- All functions allow nesting to one level. That is, a function can be an argument to another function.

You must use a double underscore preceding the name of a nested function:

```
$_{__function(arg1, ..., argN)}
```

For example:

```
$_{substring("${ops_trigger_file_name_simple}", "${__indexOf("${ops_trigger_file_name_simple}", "-")}")}
```


Function Categories

There are five categories of Functions:

- [Date functions](#)
- [Mathematical functions](#)
- [System functions](#)
- [String functions](#)
- [SQL/Stored Procedure functions](#)


Date Functions

Return Nth Business Day of Month

Description	Returns the Nth business day of month for the month of the date specified. Optionally, can start from the end of the month. <div style="background-color: #ffffcc; padding: 5px; border: 1px solid #ccc;">  Whether a holiday is treated as a business day or a non-business day is specified by the Exclude Holidays for Business Days Opswise Controller system property. </div>
Syntax	<code>\$_{businessDayOfMonth(index, [date, format, reverse])}</code>

Parameters	<ul style="list-style-type: none"> <code>index</code> Required; Nth business day of month. <code>date</code> Optional; Date in format yyyy-MM-dd. Default is the current date. <code>format</code> Optional; Format of returned date. Default is yyyy-MM-dd. (For details on the <code>format</code> parameter, see http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html) <code>reverse</code> Optional; Specification (<code>true</code> or <code>false</code>) for starting from the end of the month. Default is <code>false</code>.
Examples	<pre> \${_businessDayOfMonth(1)} --> 2012-08-01 \${_businessDayOfMonth(1,"2012-09-01")} --> 2012-09-04 \${_businessDayOfMonth(1,"2012-09-01","",true)} --> 2012-09-28 </pre>

Return Number of Business Days between Dates

Description	<p>Returns the number of business days between <code>date1</code> and <code>date2</code>.</p> <ul style="list-style-type: none"> If return value is <code>> 0</code>, <code>date2</code> is after <code>date1</code>. If return value is <code>< 0</code>, <code>date2</code> is before <code>date1</code>. If return value is <code>0</code>, <code>date1</code> is equal to <code>date2</code>. <p>The start date is inclusive, but the end date is not.</p> <div style="background-color: #ffffcc; padding: 10px; border: 1px solid #ccc;">  Whether a holiday is treated as a business day or a non-business day is specified by the Exclude Holidays for Business Days Opwise Controller system property. </div>
Syntax	<code>\${_businessDaysBetween(date1, date2)}</code>
Parameters	<p>Parameters:</p> <ul style="list-style-type: none"> <code>date1</code> Required; First date in format yyyy-MM-dd. <code>date2</code> Required; Second date in format yyyy-MM-dd.
Example	<pre> \${_businessDaysBetween("2012-08-01","2012-09-01")} --> 23 </pre>

Resolve to Current Unix Epoch Time

Description	Resolves to the current time in milliseconds since Wed Dec 31 1969 19:00:00 GMT-0500 (EST) – the start of Unix epoch time.
Syntax	<code>\${_currentTimeMillis}</code>
Parameters	n/a
Example	

Resolve to Current Date and Time

Description	Resolves to the current date and time.
Syntax	<code>\${_date([format, day_offset, hour_offset, minute_offset])}</code>

Parameters	<ul style="list-style-type: none"> • <code>format</code> Optional; Date format. The default format is <code>yyyy-MM-dd HH:mm:ss Z</code>. For details on the <code>format</code> parameter, see http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html • <code>day_offset</code> Optional; +/- number of days to offset. • <code>hour_offset</code> Optional; +/- number of hours to offset. • <code>minute_offset</code> Optional; +/- number of minutes to offset.
Examples	<pre> \${_date} --> 2012-07-14 12:43:06 -0400 \${_date()} --> 2012-07-14 12:43:06 -0400 \${_date("yyyy-MM-dd", 5)} --> 2012-07-19 \${_date("yyyy-MM-dd HH:mm:ss", -2, -1)} --> 2012-07-12 11:43:06 \${_date("", 0, 0, 10)} --> 2012-07-14 12:53:06 -0400 </pre>

Resolve to Current Date and Time (Advanced)

Description	Resolves to the current date and time.
Syntax	<code>\${_dateadv([format, year_offset, month_offset, day_offset, hour_offset, minute_offset])}</code>
Parameters	<ul style="list-style-type: none"> • <code>format</code> Date format. The default format is <code>yyyy-MM-dd HH:mm:ss Z</code>. For details on the <code>format</code> parameter, see http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html • <code>year_offset</code> Optional; +/- number of years to offset. • <code>month_offset</code> Optional; +/- number of months to offset. • <code>day_offset</code> Optional; +/- number of days to offset. • <code>hour_offset</code> Optional; +/- number of hours to offset. • <code>minute_offset</code> Optional; +/- number of minutes to offset.
Examples	<pre> \${_dateadv} --> 2012-07-29 09:31:42 -0700 \${_dateadv("yyyy-MMM", -1)} --> 2011-Jul \${_dateadv("yyyy-MMM", 0, -1)} --> 2012-Jun </pre>

Return Nth Day of Month

Description	Returns the Nth day of month for the month of the date specified. Optionally, can start from the end of the month.
Syntax	<code>\${_dayOfMonth(index, [date, format, reverse])}</code>
Parameters	<ul style="list-style-type: none"> • <code>index</code> Required; Nth day of month. • <code>date</code> Optional; Date in format <code>yyyy-MM-dd</code>. Default is the current date. • <code>format</code> Optional; Format of returned date. Default is <code>yyyy-MM-dd</code>. • <code>reverse</code> Optional; Specification (<code>true</code> or <code>false</code>) for starting from the end of the month. Default is <code>false</code>.

Examples

```

${_dayOfMonth(5)} --> 2012-08-05
${_dayOfMonth(15,"2012-09-01","MM/dd/yyyy")} --> 09/15/2012
${_dayOfMonth(1,"2012-09-01","",true)} --> 2012-09-30

```


Return Day of Week

Description	Returns the day of week for the specified date as a number.
Syntax	<code>\${_dayOfWeek([date, first_dow, first_dow_value])}</code>
Parameters	<ul style="list-style-type: none"> <code>date</code> Optional; Date in format yyyy-MM-dd. Default is the current date. <code>first_dow</code> Optional; Specification for whether the week starts on Sunday or Monday. Values are sun and mon (not case-sensitive). Default is sun. <code>first_dow_value</code> Optional; Starting value for the first day of week. Value must be a non-negative number. Default is 1.
Example	<pre> \${_dayOfWeek} --> 6 \${_dayOfWeek()} --> 6 \${_dayOfWeek("2012-07-04")} --> 4 \${_dayOfWeek("2012-07-04", "mon")} --> 3 </pre>


Return Days between Dates

Description	<p>Returns the number of days between date1 and date2.</p> <ul style="list-style-type: none"> If return value is > 0, date2 is after date1. If return value is < 0, date2 is before date1. If return value is 0, date1 is equal to date2. <p>The start date is inclusive, but the end date is not.</p>
Syntax	<code>\${_daysBetween(date1, date2)}</code>
Parameters	<ul style="list-style-type: none"> <code>date1</code> Required; First date in format yyyy-MM-dd. <code>date2</code> Required; Second date in format yyyy-MM-dd.
Example	<pre> \${_daysBetween("2012-08-01", "2012-09-01")} --> 31 </pre>

Return Date with Offsets


Description	<p>Returns the date after applying offsets. Optionally, can specify the output format.</p> <div style="background-color: #ffffcc; padding: 10px; border: 1px solid #ccc;">  Whether a holiday is treated as a business day or a non-business day is specified by the Exclude Holidays for Business Days Opwise Controller system property. </div>
Syntax	<pre>\$_formatDate([date, format, day_offset, use_business_days, hour_offset, minute_offset])</pre>
Parameters	<ul style="list-style-type: none"> • date Date in format yyyy-MM-dd HH:mm or yyyy-MM-dd. Time (HH:mm) is optional. Default is the current date and time. • format Format of returned date. Default is the format used when specifying the date parameter: yyyy-MM-dd HH:mm or yyyy-MM-dd. For details on the <code>format</code> parameter, see http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html • day_offset +/- number of days to offset. • use_business_days Specification (<code>true</code> or <code>false</code>) for whether <code>day_offset</code> is for business days. Default is <code>false</code>. • hour_offset +/- number of hours to offset. • minute_offset +/- number of minutes to offset.
Example	<pre>\$_formatDate --> 2012-08-24 15:37 \$_formatDate() --> 2012-08-24 15:37 \$_formatDate("", "MMdyyyy", 5) --> 08292012 \$_formatDate("2012-09-01", "", 5) --> 2012-09-06 \$_formatDate("2012-09-01", "", -5) --> 2012-08-27 \$_formatDate("2012-09-01", "", 5, true) --> 2012-09-10</pre>

Return Date with Offsets (Advanced)

Description	<p>Returns the date after applying offsets. Optionally, can specify the output format.</p> <div style="background-color: #ffffcc; padding: 10px; border: 1px solid #ccc;">  Whether a holiday is treated as a business day or a non-business day is specified by the Exclude Holidays for Business Days Opwise Controller system property. </div>
Syntax	<pre>\$_formatDateAdv([date, format, year_offset, month_offset, day_offset, use_business_days, hour_offset, minute_offset])</pre>

Parameters	<ul style="list-style-type: none"> • <code>date</code> Optional; Date in format yyyy-MM-dd HH:mm or yyyy-MM-dd. Time (HH:mm) is optional. Default is the current date and time. • <code>format</code> Optional; Format of returned date. Default is the format used when specifying the date parameter: yyyy-MM-dd HH:mm or yyyy-MM-dd. For details on the <code>format</code> parameter, see http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html • <code>year_offset</code> Optional; +/- number of years to offset. • <code>month_offset</code> Optional; +/- number of months to offset. • <code>day_offset</code> Optional; +/- number of days to offset. • <code>use_business_days</code> Optional; Specification (true or false) for whether <code>day_offset</code> is for business days. Default is false. • <code>hour_offset</code> +/- number of hours to offset. • <code>minute_offset</code> +/- number of minutes to offset.
Examples	<pre> \${_formatDateAdv} --> 2012-08-24 15:55 \${_formatDateAdv()} --> 2012-08-24 15:55 \${_formatDateAdv("","MMddyyyy",1)} --> 08242013 \${_formatDateAdv("2012-09-01","",0,1)} --> 2012-10-01 \${_formatDateAdv("2012-09-01","",0,-1)} --> 2012-08-01 \${_formatDateAdv("2012-09-01","",0,0,5,false)} --> 2012-09-06 </pre>

Return Non-Business Day of Month

Description	<p>Returns the Nth non-business day of month for the month of the date specified. Optionally, can start from the end of the month.</p> <div style="background-color: #ffffcc; padding: 5px; border: 1px solid #ccc;">  Whether a holiday is treated as a business day or a non-business day is specified by the Exclude Holidays for Business Days Opwise Controller system property. </div>
Syntax	<code>\${_nonBusinessDayOfMonth(index, [date, format, reverse])}</code>
Parameters	<ul style="list-style-type: none"> • <code>index</code> Required; Nth non-business day of month. • <code>date</code> Optional; Date in format yyyy-MM-dd. If blank, defaults to the current date. • <code>format</code> Optional; Format of returned date. Default is yyyy-MM-dd. For details on the <code>format</code> parameter, see http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html • <code>reverse</code> Optional; Specification (true or false) for starting from the end of the month. Default is false.
Examples	<pre> \${_nonBusinessDayOfMonth(1)} --> 2012-08-04 \${_nonBusinessDayOfMonth(1,"2012-09-01")} --> 2012-09-01 \${_nonBusinessDayOfMonth(1,"2012-09-01","",true)} --> 2012-09-30 </pre>

Mathematical Functions

Return Modulo

Description	Return the modulo (remainder) of the dividend divided by divisor.
Syntax	<code>\$_mod(dividend, divisor)</code>
Parameters	<ul style="list-style-type: none"> • dividend Integer being divided by the divisor. • divisor Integer being used to divide the dividend.
Example	<pre>\$_mod("10", "2") --> 0 \$_mod("10", "3") --> 1 \$_mod("70", "65") --> 5</pre> <p>Using Variables for dividend and divisor (<code>dividend = 23</code>, <code>divisor = 5</code>):</p> <pre>\$_mod("\${dividend}", "\${divisor}") --> 3</pre>

Add

Description	Return the sum of the augend added with the addend.
Syntax	<code>\$_add(augend, addend)</code>
Parameters	<ul style="list-style-type: none"> • augend Integer to which the addend is being added. • addend Integer being added to the augend.
Example	<pre>\$_add("77", "33") --> 110</pre> <p>Using Variables for augend and addend (<code>augend = 17</code>, <code>addend = 5</code>):</p> <pre>\$_add("\${augend}", "\${addend}") --> 22</pre>

Subtract

Description	Return the difference of the subtrahend subtracted from the minuend.
Syntax	<code>\$_subtract(minuend, subtrahend)</code>
Parameters	<ul style="list-style-type: none"> • minuend Integer from which the subtrahend is being subtracted. • subtrahend Integer being subtracted from the minuend.

Example	<pre> \$_subtract("77","33") --> 44 \$_subtract("33","77") --> -44 </pre> <p>Using Variables for minuend and subtrahend ($\\${\text{minuend}} = 100$, $\\${\text{subtrahend}} = 5$):</p> <pre> \$_add("\$_minuend","\$_subtrahend") --> 95 </pre>
----------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Multiply

Description	Return the product of the multiplicand multiplied with the multiplier.
Syntax	$\${_multiply}(\text{multiplicand}, \text{multiplier})$
Parameters	<ul style="list-style-type: none"> • <code>multiplicand</code> Integer being multiplied by the multiplier. • <code>multiplier</code> Integer being used to multiply the multiplicand.
Example	<pre> \$_multiply("7","20") --> 140 </pre> <p>Using Variables for multiplicand and multiplier ($\\${\text{multiplicand}} = 100$, $\\${\text{multiplier}} = 5$):</p> <pre> \$_multiply("\$_multiplicand","\$_multiplier") --> 500 </pre>

Divide

Description	Return the quotient of the dividend divided by divisor.
Syntax	$\${_divide}(\text{dividend}, \text{divisor})$
Parameters	<ul style="list-style-type: none"> • <code>dividend</code> Integer being divided by the divisor. • <code>divisor</code> Integer being used to divide the dividend.

Example	<pre> \$_divide("7","20") --> 0 \$_divide("20","7") --> 2 \$_divide("20","5") --> 4 </pre> <p>Using Variables for dividend and divisor ($\\${dividend} = 100$, $\\${divisor} = 5$)</p> <pre> \$_divide("\${dividend}","\${divisor}") --> 20 </pre>
----------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Return Absolute Value

Description	Return the absolute value of the parameter.
Syntax	<code>{_abs(parameter)}</code>
Parameters	<ul style="list-style-type: none"> parameter Integer (positive or negative value).
Example	<pre> \$_abs("-1200") --> 1200 \$_abs("1200") --> 1200 </pre> <p>Using Variables for parameter ($\\${parameter} = -100$):</p> <pre> \$_abs("\${parameter}") --> 100 </pre>

SQL/Stored Procedure Functions

Return SQL Results from Current Task

Description	Returns all SQL results from the current SQL or Stored Procedure task. Columns are separated by the specified <code>separator</code> and rows are separated by a new line.
Syntax	<code>\$_resultsAll([separator, rowSeparator])</code>
Parameters	Parameters: <ul style="list-style-type: none"> <code>separator</code> Optional; Column separator (default = comma). <code>rowSeparator</code> Optional; Overrides default New Line character.
Example	

Return SQL Results from Sibling Task

Description	Returns all SQL results from a sibling SQL or Stored Procedure task, within the same workflow. Columns are separated by the specified <code>separator</code> and rows are separated by a new line.
--------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Syntax	<code>\${_resultsAllFromTask(name[, separator, rowSeparator])}</code>
Parameters	<ul style="list-style-type: none"> • <code>name</code> Required; Name of the sibling task that the results should come from. The task must be within the same workflow. • <code>separator</code> Optional; Column separator (default = comma). • <code>rowSeparator</code> Optional; Overrides default New Line character.
Example	...

Return String Value of Row/Column by Column Name

Description	Returns the string value of a row/column from a previously executed SQL task within the same workflow, or from the current SQL task.
Syntax	<code>\${_resultsColumn(name, colname[, rownum, default_value])}</code>
Parameters	<ul style="list-style-type: none"> • <code>name</code> Required; Name of a sibling SQL task within the same workflow from which you want the function to fetch results. If you want to execute the function against the current task, use an empty string for the name parameter. • <code>colname</code> Required; Name of column to retrieve. • <code>rownum</code> Optional; Numeric row number in result set to retrieve (default = 1). • <code>default_value</code> Optional; Default value to return if result not found.
Example	...

Return String Value of Row/Column by Column Number

Description	Returns the string value of a row/column from a previously executed SQL task within the same workflow, or from the current SQL task.
Syntax	<code>\${_resultsColumnByNo(name, colnum[, rownum, default_value])}</code>
Parameters	<ul style="list-style-type: none"> • <code>name</code> Required; Name of a sibling SQL task within the same workflow from which you want the function to fetch results. If you want to execute the function against the current task, use an empty string for the name parameter. • <code>colnum</code> Required; Number of column to retrieve. First column in result is 1, second is 2, and so on. • <code>rownum</code> Optional; Numeric row number in result set to retrieve (default = 1). • <code>default_value</code> Optional; Default value to return if result not found.
Example	...

Return Column Names for SQL Results from Current Task

Description	Returns the column names for the SQL results from the current SQL or Stored Procedure task. Column names are separated by the specified <code>separator</code> .
Syntax	<code>\${_resultsColumnNames([separator])}</code>
Parameters	<ul style="list-style-type: none"> • <code>separator</code> Optional; Column name separator (default = comma).
Examples	...

Return Column Names for SQL Results from Sibling Task

Description	Returns the column names for the SQL results from a sibling SQL or Stored Procedure task, within the same workflow. Column names are separated by the specified <code>separator</code> .
Syntax	<code>\$_resultsColumnNamesFromTask(name[, separator])</code>
Parameters	<ul style="list-style-type: none"> <code>name</code> Required; Name of the sibling task that the results should come from. The task must be within the same workflow. <code>separator</code> Optional; Column name separator (default = comma).
Examples	...

Return String Values of Columns

Description	Returns the string values of columns in a specific row in CSV (comma-separated values) format, from a previously executed SQL task within the same workflow, or from the current SQL task.
Syntax	<code>\$_resultsColumnsCSV(name[, rownum])</code>
Parameters	<ul style="list-style-type: none"> <code>name</code> Required; Name of a sibling SQL task within the same workflow from which you want the function to fetch results. If you want to execute the function against the current task, use an empty string for the <code>name</code> parameter. <code>rownum</code> Optional; Numeric row number in result set to retrieve (default = 1).
Example	...

Return SQL Warnings from Current Task

Description	Returns all SQL warnings from the current SQL or Stored Procedure task. Columns are separated by the specified <code>separator</code> and rows are separated by a new line.
Syntax	<code>\$_SQLWarnings([separator])</code>
Parameters	<ul style="list-style-type: none"> <code>separator</code> Optional; Column separator (default = comma).
Example	...

Return SQL Warnings from Sibling Task

Description	Returns all SQL warnings from a sibling SQL or Stored Procedure task, within the same workflow. Columns are separated by the specified <code>separator</code> and rows are separated by a new line.
Syntax	<code>\$_SQLWarningsFromTask(name[, separator])</code>
Parameters	<ul style="list-style-type: none"> <code>name</code> Required; Name of the sibling task that the warnings should come from. The task must be within the same workflow. <code>separator</code> Optional; Column separator (default = comma).
Example	...

String Functions

String functions pass in either a **value** or a **variable**; for each String function that passes in a **value**, there is a corresponding String function that

passes in a **variable**.

String functions that pass in a variable are prefixed with `_var`. The variables must be fully resolved; they cannot resolve to a function.

In the **Syntax** for each String function in the following tables, the name of each function that passes a **value** and the name of the corresponding function that passes a **variable** link to each other.



Note

Indexing functions use zero-based numbering; that is, the initial element is assigned the index 0.

Return Index of Substring in String Value

Description	Returns the index within the string value of the first occurrence of the specified substring, <code>str</code> .
Syntax	<code>\${_indexOf(value, str)}</code>
Parameters	<ul style="list-style-type: none"> <code>value</code> Any string. <code>str</code> Substring to search for. If the <code>str</code> argument occurs as a substring within the value, 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.
Example	...

Return Index of Substring Plus Offset in String Value

Description	Returns the index within this string of the first occurrence of the specified substring plus the specified offset. The integer returned is the smallest value.
Syntax	<code>\${_indexOfWithOffset(value, str, offset)}</code>
Parameters	<ul style="list-style-type: none"> <code>value</code> Required; Any string. <code>str</code> Required; Substring to search for. If the <code>str</code> argument occurs as a substring within the value, 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. <code>offset</code> Required; Number (positive or negative) to offset the found index.
Example	...

Return Index of Rightmost Occurrence of Substring in String Value

Description	Returns the index within the string value of the rightmost occurrence of the specified substring, <code>str</code> .
Syntax	<code>\${_lastIndexOf(value, str)}</code>
Parameters	<ul style="list-style-type: none"> <code>value</code> Required; Any string. <code>str</code> Required; Substring to search for. If the <code>str</code> argument occurs one or more times as a substring within the value, 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.
Example	...

Return Index of Rightmost Occurrence of Substring Plus Offset in String Value

Description	Returns the index within this string of the rightmost occurrence of the specified substring, plus the specified offset. The returned index is the largest value.
Syntax	<code>\$_lastIndexOfWithOffset(value, str, offset)</code>
Parameters	<ul style="list-style-type: none"> <code>value</code> Required; Any string. <code>str</code> Required; Substring to search for. If the <code>str</code> argument occurs as a substring within the <code>value</code>, 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. <code>offset</code> Required; Number (positive or negative) to offset the found index.
Example	...

Return Length of Value

Description	Returns the length of <code>value</code> .
Syntax	<code>\$_length(value)</code>
Parameters	<ul style="list-style-type: none"> <code>value</code> Required; Any string.
Example	...

Replace Substring of Value with Regular Expression

Description	Replaces each substring of <code>value</code> that matches the specified regular expression, <code>regex</code> , with the specified replacement.
Syntax	<code>\$_replaceAll(value, regex, replacement)</code>
Parameters	<ul style="list-style-type: none"> <code>value</code> Required; Input string. <code>regex</code> Required; Regular expression. <code>replacement</code> Required; Replacement string.
Example	

Return New String that is Substring of Value

Description	Returns a new string that is a substring of <code>value</code> . The substring begins at <code>begin_index</code> and extends to the character at <code>end_index - 1</code> .
Syntax	<code>\$_substring(value, begin_index[, end_index])</code>
Parameters	<ul style="list-style-type: none"> <code>value</code> Required; String to make a substring from. <code>begin_index</code> Required; Beginning index, inclusive. <code>end_index</code> Optional; Ending index, exclusive.

Example

```

${_substring("hamburger", 4, 8)}
  resolves to "urge".
${_substring("smiles", 1, 5)}
  resolves to "mile".

```

Convert Characters in Value to Lower Case

Description	Converts all of the characters in the <code>value</code> to lower case using the rules of the default locale.
Syntax	<code>\${_toLowerCase(value)}</code>
Parameters	<ul style="list-style-type: none"> <code>value</code> Required; String to convert to lower case.
Example	...

Convert Characters in Value to Upper Case

Description	Converts all of the characters in the <code>value</code> to upper case using the rules of the default locale.
Syntax	<code>\${_toUpperCase(value)}</code>
Parameters	<ul style="list-style-type: none"> <code>value</code> Required; String to convert to upper case.
Example	...

Return Copy of Value with Whitespace Omitted

Description	Returns a copy of <code>value</code> , with leading and trailing whitespace omitted.
Syntax	<code>\${_trim(value)}</code>
Parameters	<ul style="list-style-type: none"> <code>value</code> Required; String to trim.
Example	...

Return Index of Substring within String Variable

Description	Returns the index within the string variable of the first occurrence of the specified substring, <code>str</code> .
Syntax	<code>\${_varIndexOf(variableName, str)}</code>
Parameters	<ul style="list-style-type: none"> <code>variableName</code> Required; Variable that this function is passing in. <code>str</code> Required; Substring to search for. If the <code>str</code> argument occurs as a substring within the variable, the index of the first character of the first such substring is returned; if it does not occur as a substring, -1 is returned.
Example	...

Return Index of Substring Plus Offset in String Variable

Description	Returns the index within this string of the first occurrence of the specified substring plus the specified offset. The integer returned is the smallest variable.
Syntax	<code>\$_varIndexOfWithOffset(variableName, str, offset)</code>
Parameters	<ul style="list-style-type: none"> <code>variableName</code> Required; Variable that this function is passing in. <code>str</code> Required; Substring to search for. If the <code>str</code> argument occurs as a substring within the variable, 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. <code>offset</code> Required; Number (positive or negative) to offset the found index.
Example	...

Return Index of Rightmost Occurrence of Substring in String Variable

Description	Returns the index within the string variable of the rightmost occurrence of the specified substring, <code>str</code> .
Syntax	<code>\$_varLastIndexOf(variableName, str)</code>
Parameters	<ul style="list-style-type: none"> <code>variableName</code> Required; Variable that this function is passing in. <code>str</code> Required; Substring to search for. If the <code>str</code> argument occurs one or more times as a substring within the variable, 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.
Example	...

Return Index of Rightmost Occurrence of Substring Plus Offset in String Variable

Description	Returns the index within this string of the rightmost occurrence of the specified substring, plus the specified offset. The returned index is the largest variable.
Syntax	<code>\$_varLastIndexOfWithOffset(variableName, str, offset)</code>
Parameters	<ul style="list-style-type: none"> <code>variableName</code> Required; Variable that this function is passing in. <code>str</code> Required; Substring to search for. If the <code>str</code> argument occurs as a substring within the variable, 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. <code>offset</code> Required; Number (positive or negative) to offset the found index.
Example	...

Return Length of Variable

Description	Returns the length of <code>variableName</code> .
Syntax	<code>\$_varLength(variableName[, useEmptyForUndefined])</code>
Parameters	<ul style="list-style-type: none"> <code>variableName</code> Required; Variable that this function is passing in. <code>useEmptyForUndefined</code> Optional; Specification (true or false) for the handling of a missing variable name. Default is false. <ul style="list-style-type: none"> If <code>useEmptyForUndefined = true</code>, the function will return 0. If <code>useEmptyForUndefined = false</code>, the function will remain unresolved if the variable name does not exist.
Example	...

Replace Substring of Variable with Regular Expression

Description	Replaces each substring of <code>variableName</code> that matches the specified regular expression, <code>regex</code> , with the specified replacement.
Syntax	<code>\$_varReplaceAll(variableName, regex, replacement)</code>
Parameters	<ul style="list-style-type: none"> <code>variableName</code> Required; Variable that this function is passing in. <code>regex</code> Required; Regular expression. <code>replacement</code> Required; Replacement string.
Example	...

Return New String that is Substring of Variable

Description	Returns a new string that is a substring of <code>variableName</code> . The substring begins at <code>begin_index</code> and extends to the character at <code>{end_index}-1</code> .
Syntax	<code>\$_varSubstring(variableName, beginIndex, endIndex)</code>
Parameters	<ul style="list-style-type: none"> <code>variableName</code> Required; Variable that this function is passing in. <code>begin_index</code> Required; Beginning index, inclusive. <code>end_index</code> Required; Ending index, exclusive.
Examples	<pre> \$_substring("hamburger", 4, 8) resolves to "urge". \$_substring("smiles", 1, 5) resolves to "mile". </pre>

Convert Characters in Variable to Lower Case

Description	Converts all of the characters in the variable to lower case using the rules of the default locale.
Syntax	<code>\$_varToLowerCase(variableName)</code>
Parameters	<ul style="list-style-type: none"> <code>variableName</code> Required; Variable that this function is passing in.
Example	...

Convert Characters in Variable to Upper Case

Description	Converts all of the characters in the variable to upper case using the rules of the default locale.
Syntax	<code>\$_varToUpperCase(variableName)</code>
Parameters	<ul style="list-style-type: none"> <code>variableName</code> Required; Variable that this function is passing in.
Example	...

Return Copy of variable

Description	Returns a copy of <code>variableName</code> , with leading and trailing whitespace omitted.
Syntax	<code>\${_varTrim(variableName)}</code>
Parameters	<ul style="list-style-type: none"> <code>variableName</code> Required; Variable that this function is passing in.
Example	...

System Functions

Resolve to GUID (Globally Unique ID)

Description	Resolves to a 32-byte GUID (Globally Unique ID).
Syntax	<code>\${_guid}</code>
Parameters	(none)
Example	...

Resolve to Host Name

Description	Resolves to the hostname of the machine running the Controller, if available.
Syntax	<code>\${_hostname}</code>
Parameters	(none)
Example	...

Resolve to IP Address

Description	Resolves to the IP address of the machine running the Controller.
Syntax	<code>\${_ipaddress}</code>
Parameters	(none)
Example	...

Generate Random Number

Description	Generates a random number between <code>max</code> (inclusive) and <code>min</code> (inclusive)
Syntax	<code>\${_random([max, min])}</code>
Parameters	<ul style="list-style-type: none"> <code>max</code> Optional; Upper bound (inclusive) on the random number (default = 9). <code>min</code> Optional; Lower bound (inclusive) on the random number (default = 0).
Example	...

Resolve Variable

Description	Resolves the variable specified by the <code>variable_name</code> parameter and substitutes the <code>default_value</code> if the variable cannot be resolved.
Syntax	<code>`\${_resolve}(variable_name, default_value)`</code>
Parameters	<ul style="list-style-type: none"> <code>variable_name</code> Required; Variable name. <code>default_value</code> Required; Default value to use if the variable cannot be resolved.
Example	...

Resolve Variable (Advanced)

Description	Resolves the variable specified by the <code>variable_name</code> parameter and substitutes the default value if the variable cannot be resolved.
Syntax	<code>`\${_resolveadv}(variable_name, default_value, [use_default_if_blank])`</code>
Parameters	<ul style="list-style-type: none"> <code>variable_name</code> Required; Variable name. <code>default_value</code> Required; Default value to use if the variable cannot be resolved. <code>use_default_if_blank</code> Optional; Specification (true or false) for whether or not to use the default value if the variable is empty or blank. (If <code>use_default_if_blank</code> is false, <code>_resolveadv</code> behaves like <code>_resolve</code>.)
Example	...

Display Variables

Description	Displays all the defined and built-in variables associated with the task instance.
Syntax	<code>`\${_scope}`</code>
Parameters	(none)
Example	<pre>`\${_scope}` --> {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=84880af243d26172019aald25988a8f9, ops_task_name=Opwise - Linux Ls}</pre>

Resolve to SYS_ID

Description	Resolves to the <code>sys_id</code> of the first task instance found within the same workflow specified by the sibling name.
Syntax	<code>`\${_siblingid}(sibling_name)`</code>
Parameters	<ul style="list-style-type: none"> <code>sibling_name</code> Required; Sibling name.

Example

```

${_siblingid("Timer 60")}
--> 5dbaaab943d26172015e10ab3e894e10

```

Resolve to Variable Value

Description	Locates the specified variable in the specified sibling task instance within the same workflow and resolves to the variable value.
Syntax	<code>\${_varLookup(sibling_name, variable_name[,def])}</code>
Parameters	<ul style="list-style-type: none"> • <code>sibling_name</code> Required; Name of the sibling task instance from which the function is collecting the variable value. • <code>variable_name</code> Required; Name of the variable being collected by the function. • <code>def</code> Optional; default value to return if the variable is not defined in the sibling task instance.
Example	...