cloudify-plugins-common Documentation

Release 3.2

Gigaspaces

June 02, 2015

Contents

This is the API reference to the cloudify-plugins-common module which is required when writing any cloudify plugin (and workflow).

Contents:

CHAPTER 1

Context

class cloudify.context.**ContextCapabilities**(*endpoint*, *instance*)
Bases: object

Maps from instance relationship target ids to their respective runtime properties

get_all()
Returns all capabilities as dict.

class cloudify.context.**CommonContext**(*ctx=None*)
Bases: object

class cloudify.context.**BootstrapContext**(*bootstrap_context*)
Bases: object

Holds bootstrap context that was posted to the rest service. (usually during the bootstrap process).

class PolicyEngine(*policy_engine*)
Bases: object

Cloudify policy engine related configuration

start_timeout
Returns the number of seconds to wait for the policy engine to start

class BootstrapContext.**CloudifyAgent**(*cloudify_agent*)
Bases: object

Cloudify agent related bootstrap context properties.

min_workers
Returns the minimum number of workers for agent hosts.

max_workers
Returns the maximum number of workers for agent hosts.

user
Returns the username used when SSH-ing during agent installation.

remote_execution_port
Returns the port used when SSH-ing during agent installation.

agent_key_path
Returns the path to the key file on the management machine used when SSH-ing during agent installation.

BootstrapContext.**cloudify_agent**
Returns Cloudify agent related bootstrap context data

Return type *CloudifyAgent*

BootstrapContext.**policy_engine**
Returns Cloudify policy engine related bootstrap context data

Return type *PolicyEngine*

BootstrapContext.**resources_prefix**
Returns the resources prefix that was configured during bootstrap. An empty string is returned if the resources prefix was not configured.

class cloudify.context.**EntityContext**(*context*, **_)
Bases: object

class cloudify.context.**BlueprintContext**(*context*, **_)
Bases: *cloudify.context.EntityContext*

id
The blueprint id the plugin invocation belongs to.

class cloudify.context.**DeploymentContext**(*context*, **_)
Bases: *cloudify.context.EntityContext*

id
The deployment id the plugin invocation belongs to.

class cloudify.context.**NodeContext**(**args*, ***kwargs*)
Bases: *cloudify.context.EntityContext*

id
The node's id

name
The node's name

properties
The node properties as dict (read-only). These properties are the properties specified in the blueprint.

class cloudify.context.**NodeInstanceContext**(**args*, ***kwargs*)
Bases: *cloudify.context.EntityContext*

id
The node instance id.

runtime_properties
The node instance runtime properties as a dict (read-only).

Runtime properties are properties set during the node instance's lifecycle. Retrieving runtime properties involves a call to Cloudify's storage.

update()
Stores new/updated runtime properties for the node instance in context in Cloudify's storage.

This method should be invoked only if its necessary to immediately update Cloudify's storage with changes. Otherwise, the method is automatically invoked as soon as the task execution is over.

host_ip
Returns the node instance host ip address.

This values is derived by reading the `host_id` from the relevant node instance and then reading its `ip` runtime property or its node_state `ip` property.

relationships
Returns a list of this instance relationships

Returns list of RelationshipContext

Return type list

class cloudify.context.**RelationshipContext**(*relationship_context*, *endpoint*, *node*)
Bases: *cloudify.context.EntityContext*

Holds relationship instance data

target
Returns a holder for target node and target instance

Return type *RelationshipSubjectContext*

type
The relationship type

type_hierarchy
The relationship type hierarchy

class cloudify.context.**RelationshipSubjectContext**(*context*, *endpoint*, *modifiable*)
Bases: object

Holds reference to node and node instance.

Obtained in relationship operations by *ctx.source* and *ctx.target*, and by iterating instance relationships and for each relationship, reading *relationship.target*

class cloudify.context.**CloudifyContext**(*ctx=None*)
Bases: *cloudify.context.CommonContext*

A context object passed to plugins tasks invocations. The context object is used in plugins when interacting with the Cloudify environment:

```
from cloudify import ctx

@operation
def my_start(**kwargs):
    # port is a property that was configured on the current instance's
    # node
    port = ctx.node.properties['port']
    start_server(port=port)
```

instance
The node instance the operation is executed for.

This property is only relevant for NODE_INSTANCE context operations.

node
The node the operation is executed for.

This property is only relevant for NODE_INSTANCE context operations.

source
Provides access to the relationship's operation source node and node instance.

This property is only relevant for relationship operations.

target
Provides access to the relationship's operation target node and node instance.

This property is only relevant for relationship operations.

type
The type of this context.

Available values:

- DEPLOYMENT
- NODE_INSTANCE
- RELATIONSHIP_INSTANCE

execution_id
The workflow execution id the plugin invocation was requested from. This is a unique value which identifies a specific workflow execution.

workflow_id
The workflow id the plugin invocation was requested from. For example:

`install`, `uninstall` etc...

task_id
The plugin's task invocation unique id.

task_name
The full task name of the invoked task.

task_target
The task target (RabbitMQ queue name).

plugin
The plugin name of the invoked task.

operation
The current operation context.

capabilities
Maps from instance relationship target ids to their respective runtime properties

NOTE: This feature is deprecated, use 'instance.relationships' instead.

logger
A Cloudify context aware logger.

Use this logger in order to index logged messages in ElasticSearch using logstash.

bootstrap_context
System context provided during the bootstrap process

Return type *BootstrapContext*

send_event(*event*)
Send an event to rabbitmq

Parameters **event** – the event message

provider_context
Gets provider context which contains provider specific metadata.

get_resource(*resource_path*)
Retrieves a resource bundled with the blueprint as a string.

Parameters **resource_path** – the path to the resource. Note that this path is relative to the blueprint file which was uploaded.

download_resource(*resource_path*, *target_path=None*)
Retrieves a resource bundled with the blueprint and saves it under a local file.

Parameters

- **resource_path** – the path to the resource. Note that this path is relative to the blueprint file which was uploaded.
- **target_path** – optional local path (including filename) to store the resource at on the local file system. If missing, the location will be a tempfile with a generated name.

Returns

The path to the resource on the local file system (identical to target_path parameter if used).

raises an `cloudify.exceptions.HttpException`

Raises `cloudify.exceptions.HttpException` on any kind of HTTP Error.

Raises `IOError` if the resource failed to be written to the local file system.

class `cloudify.context.`**OperationContext**(*operation_context*)

Bases: `object`

name

The name of the operation.

retry_number

The retry number (relevant for retries and recoverable errors).

max_retries

The maximum number of retries the operation can have.

retry(*message=None*, *retry_after=None*)

Specifies that this operation should be retried.

Usage: return ctx.operation.retry(message='...', retry_after=1000)

:param message A text message containing information about the reason for retrying the operation.

:param retry_after How many seconds should the workflow engine wait before re-executing the operation.

CHAPTER 2

Decorators

cloudify.decorators.**operation**(*func=None*, ***arguments*)

Decorate plugin operation function with this decorator. Internally, if celery is installed, will also wrap the function with a `@celery.task` decorator

The `ctx` injected to the function arguments is of type `cloudify.context.CloudifyContext`

The `ctx` object can also be accessed by importing `cloudify.ctx`

Example:

```
from cloudify import ctx

@operations
def start(**kwargs):
    pass
```

cloudify.decorators.**workflow**(*func=None*, ***arguments*)

Decorate workflow functions with this decorator. Internally, if celery is installed, will also wrap the function with a `@celery.task` decorator

The `ctx` injected to the function arguments is of type `cloudify.workflows.workflow_context.CloudifyWorkflo`

The `ctx` object can also be accessed by importing `cloudify.workflows.ctx`

Example:

```
from cloudify.workflows import ctx

@workflow
def reinstall(**kwargs):
    pass
```

exception cloudify.decorators.**RequestSystemExit**

Bases: `exceptions.SystemExit`

CHAPTER 3

Exceptions

exception cloudify.exceptions.**NonRecoverableError**
Bases: exceptions.Exception

An error raised by plugins to denote that no retry should be attempted by by the executing workflow engine.

exception cloudify.exceptions.**RecoverableError**(*message=None*, *retry_after=None*)
Bases: exceptions.Exception

An error raised by plugins to explicitly denote that this is a recoverable error (note that this is the default behavior). It is possible specifying how many seconds should pass before a retry is attempted thus overriding the bootstrap context configuration parameter: cloudify.workflows.retry_interval

Parameters **retry_after** – How many seconds should the workflow engine wait before re-executing the task the raised this exception. (only applies when the workflow engine decides that this task should be retried)

exception cloudify.exceptions.**OperationRetry**(*message=None*, *retry_after=None*)
Bases: *cloudify.exceptions.RecoverableError*

An error raised internally when an operation uses the ctx.operation.retry API for specifying that an operation should be retried.

exception cloudify.exceptions.**HttpException**(*url*, *code*, *message*)
Bases: *cloudify.exceptions.NonRecoverableError*

Wraps HTTP based exceptions that may be raised.

Parameters

- **url** – The url the request was made to.
- **code** – The response status code.
- **message** – The underlying reason for the error.

exception cloudify.exceptions.**CommandExecutionException**(*command*, *error*, *output*, *code*)
Bases: exceptions.Exception

Indicates a failure to execute a command.

Parameters

- **command** – The command executed
- **error** – process stderr output
- **output** – process stdout output
- **code** – process exit code

exception cloudify.exceptions.**TimeoutException**(**args*)

Bases: exceptions.Exception

Indicates some kind of timeout happened.

exception cloudify.exceptions.**ProcessExecutionError**(*message*, *error_type=None*, *traceback=None*)

Bases: exceptions.RuntimeError

Raised by the workflow engine when workflow execution fails.

CHAPTER 4

Manager

class cloudify.manager.**NodeInstance**(*node_instance_id*, *node_id*, *runtime_properties=None*, *state=None*, *version=None*, *host_id=None*, *relationships=None*)

Bases: object

Represents a deployment node instance. An instance of this class contains runtime information retrieved from Cloudify's runtime storage as well as the node's state.

delete(*key*)

runtime_properties

The node instance runtime properties.

To update the properties, make changes on the returned dict and call update_node_instance with the modified instance.

state

The node instance state.

To update the node instance state, change this property value and call update_node_instance with the modified instance.

node_id

relationships

cloudify.manager.**get_rest_client**()

Returns A REST client configured to connect to the manager in context

Return type cloudify_rest_client.CloudifyClient

cloudify.manager.**download_resource**(*resource_path*, *logger*, *target_path=None*)

Download resource from the manager file server.

Parameters

- **resource_path** – path to resource on the file server
- **logger** – logger to use for info output
- **target_path** – optional target path for the resource

Returns path to the downloaded resource

cloudify.manager.**download_blueprint_resource**(*blueprint_id*, *resource_path*, *logger*, *target_path=None*)

Download resource from the manager file server with path relative to the blueprint denoted by blueprint_id.

Parameters

- **blueprint_id** – the blueprint id of the blueprint to download the resource from
- **resource_path** – path to resource relative to blueprint folder
- **logger** – logger to use for info output
- **target_path** – optional target path for the resource

Returns path to the downloaded resource

cloudify.manager.**get_resource**(*resource_path*, *base_url=None*)
Get resource from the manager file server.

Parameters **resource_path** – path to resource on the file server

Returns resource content

cloudify.manager.**get_blueprint_resource**(*blueprint_id*, *resource_path*)
Get resource from the manager file server with patch relative to the blueprint denoted by `blueprint_id`.

Parameters

- **blueprint_id** – the blueprint id of the blueprint to download the resource from
- **resource_path** – path to resource relative to blueprint folder

Returns resource content

cloudify.manager.**get_node_instance**(*node_instance_id*)
Read node instance data from the storage.

Parameters **node_instance_id** – the node instance id

Return type *NodeInstance*

cloudify.manager.**update_node_instance**(*node_instance*)
Update node instance data changes in the storage.

Parameters **node_instance** – the node instance with the updated data

cloudify.manager.**get_node_instance_ip**(*node_instance_id*)
Get the IP address of the host the node instance denoted by `node_instance_id` is contained in.

cloudify.manager.**update_execution_status**(*execution_id*, *status*, *error=None*)
Update the execution status of the execution denoted by `execution_id`.

Returns The updated status

cloudify.manager.**get_bootstrap_context**()
Read the manager bootstrap context.

cloudify.manager.**get_provider_context**()
Read the manager provider context.

CHAPTER 5

Mocks

class cloudify.mocks.**MockNodeInstanceContext**(*id=None*, *runtime_properties=None*)
Bases: object

id

runtime_properties

update()

class cloudify.mocks.**MockNodeContext**(*id=None*, *properties=None*)
Bases: object

id

name

properties

class cloudify.mocks.**MockContext**(*values=None*)
Bases: dict

class cloudify.mocks.**MockCloudifyContext**(*node_id=None*, *node_name=None*, *blueprint_id=None*, *deployment_id=None*, *execution_id=None*, *properties=None*, *runtime_properties=None*, *capabilities=None*, *related=None*, *source=None*, *target=None*, *operation=None*, *resources=None*, *provider_context=None*, *bootstrap_context=None*)
Bases: *cloudify.context.CloudifyContext*

Cloudify context mock that can be used when testing Cloudify plugins.

execution_id

capabilities

logger

provider_context

bootstrap_context

download_resource(*resource_path*, *target_path=None*)

get_resource(*resource_path*)

CHAPTER 6

Utils

cloudify.utils.**get_local_ip**()
Return the IP address used to connect to this machine by the management. machine

cloudify.utils.**get_manager_ip**()
Returns the IP address of manager inside the management network.

cloudify.utils.**get_manager_file_server_blueprints_root_url**()
Returns the blueprints root url in the file server.

cloudify.utils.**get_manager_file_server_url**()
Returns the manager file server base url.

cloudify.utils.**get_manager_rest_service_port**()
Returns the port the manager REST service is running on.

cloudify.utils.**id_generator**(*size=6*, *chars='ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'*)
Generate and return a random string using upper case letters and digits.

class cloudify.utils.**LocalCommandRunner**(*logger=None*, *host='localhost'*)
Bases: object

run(*command*, *exit_on_failure=True*, *stdout_pipe=True*, *stderr_pipe=True*)
Runs local commands.

Parameters

- **command** – The command to execute.
- **exit_on_failure** – False to ignore failures.
- **stdout_pipe** – False to not pipe the standard output.
- **stderr_pipe** – False to not pipe the standard error.

Returns A wrapper object for all valuable info from the execution.

Return type *CommandExecutionResponse*

class cloudify.utils.**CommandExecutionResponse**(*command*, *std_out*, *std_err*, *return_code*)
Bases: object

Wrapper object for info returned when running commands.

Parameters

- **command** – The command that was executed.
- **std_out** – The output from the execution.

- **std_err** – The error message from the execution.
- **return_code** – The return code from the execution.

cloudify.utils.**setup_default_logger**(*logger_name*, *logger_level=10*, *handlers=None*, *remove_existing_handlers=True*)

Parameters

- **logger_name** – Name of the logger.
- **logger_level** – Level for the logger (not for specific handler).
- **handlers** – An optional list of handlers (formatter will be overridden); If None, only a StreamHandler for sys.stdout will be used.
- **remove_existing_handlers** – Determines whether to remove existing handlers before adding new ones

Returns A logger instance.

Return type Logger

CHAPTER 7

Logs

cloudify.logs.**message_context_from_cloudify_context**(*ctx*)
Build a message context from a CloudifyContext instance

cloudify.logs.**message_context_from_workflow_context**(*ctx*)
Build a message context from a CloudifyWorkflowContext instance

cloudify.logs.**message_context_from_workflow_node_instance_context**(*ctx*)
Build a message context from a CloudifyWorkflowNode instance

class cloudify.logs.**CloudifyBaseLoggingHandler**(*ctx*, *out_func*, *message_context_builder*)
Bases: logging.Handler

A base handler class for writing log messages to RabbitMQ

flush()

emit(*record*)

class cloudify.logs.**CloudifyPluginLoggingHandler**(*ctx*, *out_func=None*)
Bases: *cloudify.logs.CloudifyBaseLoggingHandler*

A handler class for writing plugin log messages to RabbitMQ

class cloudify.logs.**CloudifyWorkflowLoggingHandler**(*ctx*, *out_func=None*)
Bases: *cloudify.logs.CloudifyBaseLoggingHandler*

A Handler class for writing workflow log messages to RabbitMQ

class cloudify.logs.**CloudifyWorkflowNodeLoggingHandler**(*ctx*, *out_func=None*)
Bases: *cloudify.logs.CloudifyBaseLoggingHandler*

A Handler class for writing workflow nodes log messages to RabbitMQ

cloudify.logs.**init_cloudify_logger**(*handler*, *logger_name*, *logging_level=20*)
Instantiate an amqp backed logger based on the provided handler for sending log messages to RabbitMQ

Parameters

- **handler** – A logger handler based on the context
- **logger_name** – The logger name
- **logging_level** – The logging level

Returns An amqp backed logger

cloudify.logs.**send_workflow_event**(*ctx*, *event_type*, *message=None*, *args=None*, *additional_context=None*, *out_func=None*)
Send a workflow event to RabbitMQ

Parameters

- **ctx** – A CloudifyWorkflowContext instance
- **event_type** – The event type
- **message** – The message
- **args** – additional arguments that may be added to the message
- **additional_context** – additional context to be added to the context

cloudify.logs.**send_workflow_node_event**(*ctx*, *event_type*, *message=None*, *args=None*, *additional_context=None*, *out_func=None*)

Send a workflow node event to RabbitMQ

Parameters

- **ctx** – A CloudifyWorkflowNode instance
- **event_type** – The event type
- **message** – The message
- **args** – additional arguments that may be added to the message
- **additional_context** – additional context to be added to the context

cloudify.logs.**send_plugin_event**(*ctx*, *message=None*, *args=None*, *additional_context=None*, *out_func=None*)

Send a plugin event to RabbitMQ

Parameters

- **ctx** – A CloudifyContext instance
- **message** – The message
- **args** – additional arguments that may be added to the message
- **additional_context** – additional context to be added to the context

cloudify.logs.**send_task_event**(*cloudify_context*, *event_type*, *message=None*, *args=None*, *additional_context=None*, *out_func=None*)

Send a task event to RabbitMQ

Parameters

- **cloudify_context** – a __cloudify_context struct as passed to operations
- **event_type** – The event type
- **message** – The message
- **args** – additional arguments that may be added to the message
- **additional_context** – additional context to be added to the context

cloudify.logs.**populate_base_item**(*item*, *message_type*)

cloudify.logs.**amqp_event_out**(*event*)

cloudify.logs.**amqp_log_out**(*log*)

cloudify.logs.**stdout_event_out**(*event*)

cloudify.logs.**stdout_log_out**(*log*)

cloudify.logs.**create_event_message_prefix**(*event*)

CHAPTER 8

Workflows

8.1 Workflow Tasks Graph

class cloudify.workflows.tasks_graph.**TaskDependencyGraph**(*workflow_context*)

Bases: object

A task graph builder

Parameters **workflow_context** – A WorkflowContext instance (used for logging)

add_task(*task*)

Add a WorkflowTask to this graph

Parameters **task** – The task

get_task(*task_id*)

Get a task instance that was inserted to this graph by its id

Parameters **task_id** – the task id

Returns a WorkflowTask instance for the requested task if found. None, otherwise.

remove_task(*task*)

Remove the provided task from the graph

Parameters **task** – The task

add_dependency(*src_task*, *dst_task*)

Add a dependency between tasks. The source task will only be executed after the target task terminates. A task may depend on several tasks, in which case it will only be executed after all its 'destination' tasks terminate

Parameters

- **src_task** – The source task
- **dst_task** – The target task

sequence()

Returns a new TaskSequence for this graph

execute()

Start executing the graph based on tasks and dependencies between them. Calling this method will block until one of the following occurs:

1.all tasks terminated

2.a task failed

3.an unhandled exception is raised

4.the execution is cancelled

Note: This method will raise an api.ExecutionCancelled error if the execution has been cancelled. When catching errors raised from this method, make sure to re-raise the error if it's api.ExecutionsCancelled in order to allow the execution to be set in cancelled mode properly.

Also note that for the time being, if such a cancelling event occurs, the method might return even while there's some operations still being executed.

tasks_iter()
An iterator on tasks added to the graph

class cloudify.workflows.tasks_graph.**forkjoin**(**tasks*)
Bases: object

A simple wrapper for tasks. Used in conjunction with TaskSequence. Defined to make the code easier to read (instead of passing a list) see TaskSequence.add for more details

class cloudify.workflows.tasks_graph.**TaskSequence**(*graph*)
Bases: object

Helper class to add tasks in a sequential manner to a task dependency graph

Parameters **graph** – The TaskDependencyGraph instance

add(**tasks*)
Add tasks to the sequence.

Parameters **tasks** – Each task might be:

- A WorkflowTask instance, in which case, it will be added to the graph with a dependency between it and the task previously inserted into the sequence
- A forkjoin of tasks, in which case it will be treated as a "fork-join" task in the sequence, i.e. all the fork-join tasks will depend on the last task in the sequence (could be fork join) and the next added task will depend on all tasks in this fork-join task

8.2 Workflow API

cloudify.workflows.workflow_api.**has_cancel_request**()
Checks for requests to cancel the workflow execution. This should be used to allow graceful termination of workflow executions.

If this method is not used and acted upon, a simple 'cancel' request for the execution will have no effect - 'force-cancel' will have to be used to abruptly terminate the execution instead.

Note: When this method returns True, the workflow should make the appropriate cleanups and then it must raise an ExecutionCancelled error if the execution indeed gets cancelled (i.e. if it's too late to cancel there is no need to raise this exception and the workflow should end normally).

Returns whether there was a request to cancel the workflow execution

exception cloudify.workflows.workflow_api.**ExecutionCancelled**
Bases: exceptions.Exception

This exception should be raised when a workflow has been cancelled, once appropriate cleanups have taken place.

8.3 Workflow Context

class cloudify.workflows.workflow_context.**CloudifyWorkflowRelationshipInstance**(*ctx*, *node_instance*, *nodes_and_instance*, *relationship_instance*)

Bases: object

A node instance relationship instance

Parameters

- **ctx** – a CloudifyWorkflowContext instance
- **node_instance** – a CloudifyWorkflowNodeInstance instance
- **nodes_and_instances** – a WorkflowNodesAndInstancesContainer instance
- **relationship_instance** – A relationship dict from a NodeInstance instance (of the rest client model)

target_id
The relationship target node id

target_node_instance
The relationship's target node CloudifyWorkflowNodeInstance instance

relationship
The relationship object for this relationship instance

execute_source_operation(*operation*, *kwargs=None*, *allow_kwargs_override=False*, *send_task_events=True*)
Execute a node relationship source operation

Parameters

- **operation** – The node relationship operation
- **kwargs** – optional kwargs to be passed to the called operation

execute_target_operation(*operation*, *kwargs=None*, *allow_kwargs_override=False*, *send_task_events=True*)
Execute a node relationship target operation

Parameters

- **operation** – The node relationship operation
- **kwargs** – optional kwargs to be passed to the called operation

class cloudify.workflows.workflow_context.**CloudifyWorkflowRelationship**(*ctx*, *node*, *nodes_and_instances*, *relationship*)

Bases: object

A node relationship

Parameters

- **ctx** – a CloudifyWorkflowContext instance
- **node** – a CloudifyWorkflowNode instance
- **nodes_and_instances** – a WorkflowNodesAndInstancesContainer instance
- **relationship** – a relationship dict from a Node instance (of the rest client mode)

target_id
The relationship target node id

target_node
The relationship target node WorkflowContextNode instance

source_operations
The relationship source operations

target_operations
The relationship target operations

is_derived_from(*other_relationship*)

Parameters **other_relationship** – a string like cloudify.relationships.contained_in

class cloudify.workflows.workflow_context.**CloudifyWorkflowNodeInstance**(*ctx*, *node*, *node_instance*, *nodes_and_instances*)

Bases: object

A plan node instance

Parameters

- **ctx** – a CloudifyWorkflowContext instance
- **node** – a CloudifyWorkflowContextNode instance
- **node_instance** – a NodeInstance (rest client response model)
- **nodes_and_instances** – a WorkflowNodesAndInstancesContainer instance

set_state(*state*)
Set the node state

Parameters **state** – The node state

Returns the state set

get_state()
Get the node state

Returns The node state

send_event(*event*, *additional_context=None*)
Sends a workflow node event to RabbitMQ

Parameters

- **event** – The event
- **additional_context** – additional context to be added to the context

execute_operation(*operation*, *kwargs=None*, *allow_kwargs_override=False*, *send_task_events=True*)
Execute a node operation

Parameters

- **operation** – The node operation
- **kwargs** – optional kwargs to be passed to the called operation

id
The node instance id

node_id
The node id (this instance is an instance of that node)

relationships
The node relationships

node
The node object for this node instance

modification
Modification enum (None, added, removed)

logger
A logger for this workflow node

contained_instances
Returns node instances directly contained in this instance (children)

get_contained_subgraph()
Returns a set containing this instance and all nodes that are contained directly and transitively within it

class cloudify.workflows.workflow_context.**CloudifyWorkflowNode**(*ctx*, *node*, *nodes_and_instances*)

Bases: object

A plan node instance

Parameters

- **ctx** – a CloudifyWorkflowContext instance
- **node** – a Node instance (rest client response model)
- **nodes_and_instances** – a WorkflowNodesAndInstancesContainer instance

id
The node id

type
The node type

type_hierarchy
The node type hierarchy

properties
The node properties

plugins_to_install
The plugins to install in this node. (Only relevant for host nodes)

host_id

host_node

number_of_instances

relationships
The node relationships

operations
The node operations

instances
The node instances

get_relationship(*target_id*)
Get a node relationship by its target id

class cloudify.workflows.workflow_context.**WorkflowNodesAndInstancesContainer**(*workflow_context*, *raw_nodes*, *raw_node_instances*)

Bases: object

nodes

get_node(*node_id*)
Get a node by its id

Parameters **node_id** – The node id

Returns a CloudifyWorkflowNode instance for the node or None if not found

get_node_instance(*node_instance_id*)
Get a node instance by its id

Parameters **node_instance_id** – The node instance id

Returns a CloudifyWorkflowNode instance for the node or None if not found

class cloudify.workflows.workflow_context.**CloudifyWorkflowContext**(*ctx*)
Bases: *cloudify.workflows.workflow_context.WorkflowNodesAndInstancesContainer*

A context used in workflow operations

Parameters **ctx** – a cloudify_context workflow dict

graph_mode()
Switch the workflow context into graph mode

Returns A task dependency graph instance

execution_id
The execution id

workflow_id
The workflow id

local
Is the workflow running in a local or remote context

logger
A logger for this workflow

send_event(*event*, *event_type='workflow_stage'*, *args=None*, *additional_context=None*)
Sends a workflow event to RabbitMQ

Parameters

- **event** – The event
- **event_type** – The event type
- **args** – additional arguments that may be added to the message
- **additional_context** – additional context to be added to the context

update_execution_status(*new_status*)
Updates the execution status to new_status. Note that the workflow status gets automatically updated before and after its run (whether the run succeeded or failed)

execute_task(*task_name*, *task_queue=None*, *kwargs=None*, *node_context=None*, *send_task_events=True*, *total_retries=None*, *retry_interval=None*)
Execute a task

Parameters

- **task_name** – the task named
- **task_queue** – the task queue, if None runs the task locally
- **kwargs** – optional kwargs to be passed to the task
- **node_context** – Used internally by node.execute_operation

local_task(*local_task*, *node=None*, *info=None*, *kwargs=None*, *task_id=None*, *name=None*, *send_task_events=True*, *override_task_config=False*, *total_retries=None*, *retry_interval=None*)
Create a local workflow task

Parameters

- **local_task** – A callable implementation for the task
- **node** – A node if this task is called in a node context
- **info** – Additional info that will be accessed and included in log messages
- **kwargs** – kwargs to pass to the local_task when invoked
- **task_id** – The task id

remote_task(*task*, *cloudify_context*, *task_id*, *send_task_events=True*, *total_retries=None*, *retry_interval=None*)
Create a remote workflow task

Parameters

- **task** – The underlying celery task
- **cloudify_context** – A dict for creating the CloudifyContext used by the called task
- **task_id** – The task id

class cloudify.workflows.workflow_context.**LocalTasksProcessing**(*thread_pool_size=1*)
Bases: object

start()

stop()

add_task(*task*)

class cloudify.workflows.workflow_context.**CloudifyWorkflowContextHandler**(*workflow_ctx*)
Bases: object

get_context_logging_handler()

get_node_logging_handler(*workflow_node_instance*)

bootstrap_context

get_send_task_event_func(*task*)

get_update_execution_status_task(*new_status*)

get_send_node_event_task(*workflow_node_instance*, *event*, *additional_context=None*)

get_send_workflow_event_task(*event*, *event_type*, *args*, *additional_context=None*)

get_operation_task_queue(*workflow_node_instance*, *operation_executor*)

operation_cloudify_context

get_set_state_task(*workflow_node_instance*, *state*)

get_get_state_task(*workflow_node_instance*)

send_workflow_event(*event_type*, *message=None*, *args=None*)

download_blueprint_resource(*resource_path*, *target_path=None*)

start_deployment_modification(*nodes*)

finish_deployment_modification(*modification*)

rollback_deployment_modification(*modification*)

class cloudify.workflows.workflow_context.**RemoteCloudifyWorkflowContextHandler**(*workflow_ctx*)

Bases: *cloudify.workflows.workflow_context.CloudifyWorkflowContextHandler*

get_context_logging_handler()

get_node_logging_handler(*workflow_node_instance*)

bootstrap_context

get_send_task_event_func(*task*)

get_update_execution_status_task(*new_status*)

get_send_node_event_task(*workflow_node_instance*, *event*, *additional_context=None*)

get_send_workflow_event_task(*event*, *event_type*, *args*, *additional_context=None*)

get_operation_task_queue(*workflow_node_instance*, *operation_executor*)

operation_cloudify_context

get_set_state_task(*workflow_node_instance*, *state*)

get_get_state_task(*workflow_node_instance*)

send_workflow_event(*event_type*, *message=None*, *args=None*)

download_blueprint_resource(*resource_path*, *target_path=None*)

start_deployment_modification(*nodes*)

finish_deployment_modification(*modification*)

rollback_deployment_modification(*modification*)

class cloudify.workflows.workflow_context.**LocalCloudifyWorkflowContextHandler**(*workflow_ctx*, *storage*)

Bases: *cloudify.workflows.workflow_context.CloudifyWorkflowContextHandler*

get_context_logging_handler()

get_node_logging_handler(*workflow_node_instance*)

bootstrap_context

get_send_task_event_func(*task*)

get_update_execution_status_task(*new_status*)

get_send_node_event_task(*workflow_node_instance*, *event*, *additional_context=None*)

get_send_workflow_event_task(*event*, *event_type*, *args*, *additional_context=None*)

get_operation_task_queue(*workflow_node_instance*, *operation_executor*)

operation_cloudify_context

get_set_state_task(*workflow_node_instance*, *state*)

get_get_state_task(*workflow_node_instance*)

send_workflow_event(*event_type*, *message=None*, *args=None*)

download_blueprint_resource(*resource_path*, *target_path=None*)

class cloudify.workflows.workflow_context.**Modification**(*workflow_ctx*, *modification*)
Bases: object

added

Returns Added and related nodes

Return type *ModificationNodes*

removed

Returns Removed and related nodes

Return type *ModificationNodes*

id

finish()
Finish deployment modification process

rollback()
Rollback deployment modification process

class cloudify.workflows.workflow_context.**ModificationNodes**(*modification*, *raw_nodes*, *raw_node_instances*)
Bases: *cloudify.workflows.workflow_context.WorkflowNodesAndInstancesContainer*

class cloudify.workflows.workflow_context.**WorkflowDeploymentContext**(*cloudify_context*, *workflow_ctx*)
Bases: *cloudify.context.DeploymentContext*

start_modification(*nodes*)
Start deployment modification process

Parameters **nodes** – Modified nodes specification

Returns Workflow modification wrapper

Return type *Modification*

cloudify.workflows.workflow_context.**task_config**(*fn=None*, ***arguments*)

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

C

A

B

C

D

E

F

G

H

I

S

T

U

W