Properties

65 minute read

You are browsing an older version of this page.
Click here to view the latest released version.
Refer to **Maintenance Lifecycle** for details.

CloudBees CD provides a powerful data model based around the notion of a property .

- A property is a string value with a name.
- Properties can have arbitrary names and values.
- You can attach properties to any object in the CloudBees CD system, such as a project, procedure, or job.
- After a property is created, it is stored in the CloudBees CD database.
- You can retrieve and modify the property value later.
- You can delete properties that you no longer need.

Properties are used extensively throughout the CloudBees CD system and provide a flexible and powerful mechanism to manage data about your builds.

Property Use Case Examples

- When a job starts, it computes a unique identifier for that build and saves it in a property. Later build steps can retrieve the property value to embed the build number in binaries generated during the job.
- When a build executes, it can set a property on the procedure to identify the source code version used for the build (for example, a tag or timestamp from your source code control system). The next time the build executes, it can retrieve the property value from the procedure and use it to extract information from your source code control system about the files that changed since the last run.
- Suppose you have a collection of machines for testing, and some machines have older test hardware and some machines have newer hardware with slightly different characteristics. You can set a property on each test machine resource to indicate which version of test hardware is available on which machine. Later, when a test executes, the test can retrieve the property for the machine where it ran and configure tests appropriately for that hardware. If you upgrade test hardware on a machine, all you need to do is change the property on that resource to reflect the new version.
- You can set job properties to indicate the build status produced by that job. For example, if your QA team finds fatal flaws in a build, it can mark the job accordingly. Builds that need to be preserved (release candidates or builds undergoing beta testing at customer sites) can be marked with properties so those builds are not deleted.
- When a job executes, properties are set for its job steps that hold metrics such as how many files compiled during a build step, how many tests executed during a test step, or how many errors were detected in the step. These property values are included in reports and can be examined later to compute trends over time. You can define additional properties for metrics useful to you.

CloudBees CD provides Intrinsic properties and allows you to create Custom properties. This Help topic enumerates properties available within CloudBees CD (Intrinsic properties), distinguishes between relative and absolute property paths, and describes property hierarchies.

Note

Intrinsic properties are case-sensitive. Custom properties, like all other object names in the CloudBees CD system, are "case-preserving," but not case-sensitive .

Submit Feedback

- **Intrinsic properties** These properties represent attributes that describe the object to which they are attached, and are provided automatically by CloudBees CD for each similar type object. For example, every project has a `Description` property that can be referenced with a non-local property path such as `/projects/Examples/description`.
- **Custom properties** Custom properties are identical to intrinsic properties and when placed on the same object, are referenced in the same manner, and behave in every way like an intrinsic object-level property with one exception: they are not created automatically when the object is created. Instead, custom properties can be added to objects already in the database before a job is started, or created dynamically by procedure steps during step execution.

Creating or modifying properties

You can set a property value by using one of the following three methods or applications: the automation platform web interface, the ***ectool*** command-line application, or the Perl API.

- Using the web interface, you will see a table labeled Custom Properties on the pages that display details for projects, procedures, and so on. Click **Create New Property** to create a new property for that object, or click on an existing property to change its value.
- Using the ectool application, set a property value with a command like:
- `ectool setProperty /myJob/installStatus complete`

  This command sets the property named `/myJob/installStatus` to the value `complete`, and creates the property if it did not already exist (the meaning of property names like `/myJob/installStatus` is explained below). You can use ectool from within one job step to set properties accessed by later steps in the same job.
- Using the Perl API, you can use an external script or a script in a step with ec-perl as the shell. A Perl code example:

```
use ElectricCommander;
my $ec = new ElectricCommander();
$ec->setProperty ("/myJob/installStatus", "complete", {jobStepId =>
$ENV{COMMANDER_JOBSTEPID}
```

Tip

Using the Perl API may yield better performance if you are requesting multiple API calls in one step.

For more information, see Using the CloudBees CD Perl API.

Using property values

A job step can access a property value by two methods. The first method is substitution , using the `$[ ] notation`. Suppose you enter the following text as a command for a step:

```
make PLATFORM=$[platform]
```

Before running the step, CloudBees CD finds the property named `platform` and substitutes its value in the command string in place of `$[platform]`. For example, if the property contains the value `"windows"`, the actual command executed is:

```
make PLATFORM=windows
```

The substitution method can be used for a command in a step and for any step fields, such as the resource or working directory. This method allows you to compute configuration information in an early step of a job, then use that configuration information to control later steps in the job.

Another way for a step to access the property value is with the ectool `getProperty` command. For example, the following command returns the property value named `platform` :

```
ectool getProperty platform
```

Property sheets and intrinsic properties

A property value can be a simple string or a nested property sheet containing its own properties. Property sheets can be nested to any depth, which allows you to create hierarchical collections of information.

Most objects have an associated "property sheet" that contains "custom properties" created by user scripts. The property sheet is an intrinsic property of the containing object called "property sheet", so to reference a project's custom property "branchName", you could specify `"/projects/aProject/propertySheet/branchName"` .

As a convenience, CloudBees CD allows the "property sheet" path element to be omitted and the path written as `"/projects/aProject/branchName"` . If there is no intrinsic property with the same name, the path will find the property on the property sheet.

Custom properties in a property sheet can be one of two types: string property or a property sheet property. String properties hold simple text values. Property sheet properties hold nested properties. Nested properties are accessed via the property sheet property of their containing sheet.

For example:

`/projects/aProject/propertySheet/topSheet/propertySheet/propB` - or -

`/projects/aProject/topSheet/propB`

All information managed by CloudBees CD exists in the form of properties and property sheets and your own custom-created properties. For example, each project, procedure, and step is represented internally as a property sheet—the command for a step is actually a property associated with the step, and so on. Every value in the CloudBees CD system can be accessed as a property, using the naming facilities described below. These properties are called intrinsic properties. *CloudBees CD enforces some restrictions on intrinsic property values, whereas custom properties can have any value you choose.*

To learn more about intrinsic properties defined for an object, see Using the CloudBees CD Perl API. For example, to learn about intrinsic properties associated with each project, find the documentation for the `getProject` ectool command. The result of running this command is an XML document whose field names and values represent the properties for the project.

For details about the intrinsic properties for each object in CloudBees CD, see Properties.

Property names and paths

Properties are named using multi-level paths such as first/second/third, which refers to a property named "third" in a property sheet named "second" in a property sheet named "first." CloudBees CD also supports an equivalent notation using brackets instead of slashes. In this format, slashes are not considered separators when they appear between brackets.

For example, `first[second]/third` and `first[second][third]` both refer to the same property as first/second/third. The bracket `[ ]` notation is based on matching brackets. For example, `steps[build[1]]` refers to a property named "build[1]" inside a property sheet named "steps" (it does not treat "build" as a property sheet containing a property named "1").

Property names/paths have two forms: absolute and relative .

Absolute property paths

Absolute property paths are referenced by a fully-qualified path syntax that begins with a slash character (" `/` ") followed by a top-level name. This path syntax is similar to a file system path specification. The first component after the `"/"` must be one of several reserved words that select a starting location to look up the property name. For example, consider the property name `/server/Electric Cloud/installDirectory` . `/server` means "lookup" starts in the topmost property sheet associated with the CloudBees CD server. This property name refers to the "installDirectory" property inside the server property sheet.

The system defines the following top-level names (absolute property names):

`/applications/…`

Start in the property sheet containing all applications. For example, `/applications[deploy]` refers to an application named "deploy" and `/applications[deploy]/processes[install]` refers to a process named "install" in that application.

`/artifacts/…`

Start in the property sheet containing all artifacts. For example, `/artifacts/myGrp:myKey/prop1` refers to the `prop1` property on the artifact whose name is `"myGrp:myKey"` .

`/artifactVersions/…`

Start in the property sheet containing all artifact versions. For example, `/artifactVersions/myGrp:myKey:1.0-36/prop1` refers to the `prop1` property on the artifact version whose name is `"myGrp:myKey:1.0-36"` .

Tip

Throughout the API, you can substitute `"groupId:artifactKey:version"` wherever an `artifactVersionName` argument can be specified. This is the same if the `artifactVersionNameTemplate` specified in the artifact is in the form `"groupId:artifactKey:version"` . When the template is different, it is convenient to be able to specify this tuple if you do not know the `artifactVersionName` .

`/components/…`

Start in the property sheet containing all components. For example, `/components[warfile]` refers to a component named "warfile" and `/component[warfile]/processes[backup]` refers to a component process named "backup" in that component.

`/environments/…`

Start in the property sheet containing all components. For example, `/environments[qeserver]` refers to an environment named "qeserver".

`/groups/…`

Start in the property sheet containing all groups. For example, `/groups[dev]` refers to the group named "dev".

`/jobs/…`

Start in the property sheet containing all jobs. For example, `/jobs[ecloud.4096]` refers to the job named "ecloud.4096." You can name a job using either its name (as in the preceding example) or using the unique identifier assigned to it by CloudBees CD.

`/plugins/…`

Start in the property sheet containing all plugins. For example, `/plugins[EC-AgentManagement]` refers to the currently promoted plugin named "EC-AgentManagement" and `/plugins[EC-AgentManagement]/project/procedures[scpCopyFile]` refers to a procedure named "scpCopyFile" in that plugin.

NOTE:

There is a subtle difference between

```
ectool setProperty /plugins/EC-AgentManagement/project/foo ‘bar’
```

and

```
ectool setProperty /plugins/EC-AgentManagement/foo ‘bar’
```

The former places the property on the plugin’s project and can be referenced by `$[/myProject/foo]` while the latter places the property on the plugin object and will not be found via `$[/myPlugin/foo]` .

`/projects/…`

Start in the property sheet containing all projects. For example, `/projects[nightly]` refers to the project named "nightly," and `/projects[nightly]/procedures[main]` refers to a procedure named "main" in that project.

`/repositories/…`

Start in the property sheet containing all artifact repositories. For example, `/repositories/repo1/prop1` refers to the `prop1` property on the repository whose name is `"repo1"` .

`/resourcePools/…`

Start in the property sheet containing all resource pools. For example, `/resourcePools[linuxA]` refers to the resource pool named "linuxA".

`/resources/…`

Start in the property sheet containing all resources. For example, `/resources[linux1]` refers to the resource named "linux1".

`/server/…`

Start in the top-level server property sheet. For example, `/server/a` refers to the server property named "a".

`/users/…`

Start in the property sheet containing all users. For example, `/users[Bob]` refers to the usernamed "Bob".

`/workspaces/…`

Start in the property sheet containing all workspaces. For example, `/workspaces[default]` refers to the workspace named "default."

Relative property paths

Property names that do not begin with `"/"` are relative and looked up starting in the current context . For example, when executing a job step, the current context includes properties defined for the job step, parameters for the current procedure, and global properties on the current job.

Relative property paths are distinguished from absolute property paths because they do not begin with one of the top-level names. To avoid having to construct full property paths, CloudBees CD supports the concept of a relative property path.

In use, the relative property path value is resolved by its context and a defined search order, which results in accessing the value of an absolute fully-specified property value. Contexts and search orders are as follows:

1. In a job step context, CloudBees CD searches for relative property paths in the following order:
   - a property on the job step object
   - a property of the parent job step object, which includes parameters of the procedure on which the step is defined
   - a property on the job object

     NOTE: *** The search can be enabled or disabled by using the `--extendedContextSearch` option on the ectool `getProperty` or `setProperty` commands. When searching for a property value, disable the search by setting the `--extendedContextSearch` switch to "false", requiring the property to exist on the job step object or return false. When writing a new value to a property, enable the search by setting the `--extendedContextSearch` switch to "true", allowing the search for a property within the search order before creating a new one. New properties are created on the job step object.** * Parameters for subprocedure calls from job steps are searched for in a job step context. ***** For procedure parameters for nested subprocedures, properties referenced by parameters are looked up as described [above] for job steps.

2. When expanding schedule parameters, CloudBees CD searches the relative property path in the following order:
   - A property on the procedure being called

- A property on the project on which the procedure being called is defined
- A property on the server

3. In a job name template context, CloudBees CD searches for the relative property path as a property on the job.
4. In any other context, CloudBees CD searches for the relative property path as a property on the context object.

jobSteps Paths

You can also specify a property on a job step as `/jobSteps/parent/propertyName` where:

- `parent` is the root parent of the job step that is not visible on the CloudBees CD UI.
- `propertyName` is the name of the property.

Context-relative Shortcuts to Property Paths

A shortcut can be used to reference a property without knowing the exact name of the object that contains the property. You might think of a shortcut as another part of the property hierarchy. Shortcuts resolve to the correct property path even though its path elements may have changed because a project or procedure was renamed. Shortcuts are particularly useful if you do not know your exact location in the property hierarchical tree.

The table below lists all shortcuts and the context in which they are available. Click on a shortcut name for more information about it.

Shortcut Name	**Available Context**		
job, step, or job step	pipeline, task, stage, or gate	Context Independent	
myApplication	✓	N/A	N/A
myApplicationTier	✓	N/A	N/A
myArtifactVersion	N/A	N/A	`artifactVersionNameTemplate` field for the artifact
myComponent	✓	N/A	N/A
myCredential	✓	✓	N/A
]	✓	N/A	N/A
myEvent	N/A	N/A	✓
myEnvironment	✓	N/A	N/A
myEnvironmentTier	✓	N/A	N/A
myGate	✓	✓	N/A
myGateRuntime	✓	✓	N/A
myGroupTaskRuntime[/myGroupTaskRuntime]	✓	✓	N/A
myJob	✓	N/A	N/A
myJobStep	✓	N/A	N/A
myParent	✓	N/A	N/A

Shortcut Name	Available Context		
myPipeline	✓	✓	
myPipelineRuntime	✓	✓	N/A
myPipelineStageRuntime	✓	✓	
myProcedure	✓	N/A	N/A
myProcess	✓	N/A	N/A
myProcessStep	✓	N/A	N/A
myProject	✓	✓	N/A
myResource	✓	N/A	N/A
myResourcePool	✓	N/A	N/A
myRetrievedArtifact	✓	N/A	N/A
myService	✓	N/A	N/A
myStage	✓	✓	N/A
myStageRuntime	✓	✓	N/A
myStep	✓	N/A	N/A
myState	✓	N/A	N/A
mySubjob	✓	N/A	N/A
mySubworkflow	✓	N/A	N/A
myTask	✓	✓	N/A
myTaskRuntime	✓	✓	N/A
myTransition	✓	N/A	N/A
myTriggeringPipelineRuntime	✓	✓	N/A
myUser	N/A	N/A	✓
myWorkflow	✓	N/A	N/A
myWorkflowDefinition	✓	N/A	N/A
myWorkspace	✓	N/A	N/A

Shortcuts

` /myApplication/…`

Resolves to the `application` object associated with the current job or job step.

` /myApplicationTier/…`

Resolves to the `applicationTier` object associated with the current job step.

` /myArtifactVersion/…`

Resolves to the `artifactVersion` object associated with the current context. The only context where this property has a value is in the `artifactVersionNameTemplate` field for the artifact.

` /myComponent/...`

Resolves to the `component` object associated with the current context.

` /myCredential/...`

Resolves to the credential object associated with the current job step. This form produces an error if the current job step lacks a credential.

[[/myDeployerTaskRuntime]] / `myDeployerTaskRuntime` /...

Resolves to the deployer runtime to retrieve information for the current deployer runtime instance.

Examples:

`/myDeployerTaskRuntime/outcome` —Retrieve the outcome of the deployer task in the current context.

`/myDeployerTaskRuntime/tasks/<subapplication>/job/jobId` —Access the specified deployer subapplication.

`/myDeployerRuntime/currentRunNumber` -Get the current run number.

` /myEvent/...`

This is a special property used only within an "email notifier." `/myEvent/` allows you to refer to fields associated with the notifier itself. You can include these properties, for example, in the text of the email you send out for notification:

`/myEvent/notifier` —Contains the name of the notifier that created the notifier event. You created this name when you created the notifier.

`/myEvent/entity` —Contains the object where the notifier is attached. Two possible values are "job" or "jobStep," depending on where the notifier is attached.

`/myEvent/source` —Contains the name of either the job or the jobStep where the notifier is attached.

`/myEvent/type` —Defines whether the notifier is an "On Start" or an "On Completion" notifier. Two possible values are "STARTED" or "COMPLETED".

`/myEvent/time` —Contains the time when the notifier occurred. The time is always specified in GMT and uses a formatted string, such as `2009-06-11T21:00:56.502Z`

`/myEvent/timeMillis` —Contains the "timestamp" in milliseconds when the notifier occurred. This value is the number of milliseconds since January 1, 1970 GMT. The `timeMillis` corresponding to the time in the previous example is: `1244754056502`

` /myEnvironment/...`

Resolves to the `environment` object associated with the current job or job step.

` /myEnvironmentTier/...`

Resolves to the `environmentTier` object associated with the current job step.

` /myGate/...`

`/myGate/` is limited to `myGate.gateType` (`PRE` or `POST`) or `myGate.parentStage` . It is useful to include information about the active gate and stage in email notifications templates.

` /myGateRuntime/...`

Resolves to the current gate runtime object.

Examples:

`/myGateRuntime/tasks/<taskName>/propertyName` —Access a property on another task in an entry gate.

`/myGateRuntime/currentRunNumber` —Get current run number.

`/myGateRuntime/runNumbers/1/subFlowRuntime/prop1` —Access a property under the first gate run, previously set with `/myGateRuntime/prop1` .

` /myGroupTaskRuntime`

Resolves to the current `myGroupTaskRuntime ` object.

Examples:

`/myGroupTaskRuntime/tasks/<taskName>/propertyName` —Access the subtask runtime within the current group task context.

`/myGroupTaskRuntime/currentRunNumber` -Get the current run number.

` /myJob/…`

Resolves to the current job. `/myJob/` points directly to the job object so you can reference built-in job properties (`jobName, createTime, outcome` , and so on) and custom properties. This shortcut can also be used to hold working data that needs to be passed from one step to another within the job.

` /myJobStep/…`

Resolves to the current job step.

` /myParent/…`

Resolves to the current parent job step or job if this is a top-level step. `/myParent/` points directly to the job or job step object, so you can reference intrinsic or custom properties. For example, you can reference the parameters that were passed to this procedure.

In addition, you can reference a sibling step by calling `/myParent/jobSteps/<sibling step name>` . Or, you can create additional properties on `/myParent/` to pass information from one step in the procedure to another: step1 calls `setProperty /myParent/results 100` step2 can call `getProperty /myParent/results/myParent/` .

` /myPipeline/…`

Resolves to the current `pipeline` definition object. This can be used by a plugin, procedure, or process step executed from a stage task to retrieve property information defined in the pipeline.

` /myPipelineRuntime/…`

Resolves to the current `pipelineRuntime` runtime object. This can be used by a plugin, procedure, or process step executed from a stage task to retrieve property information stored on the pipeline runtime

Examples:

`/myPipelineRuntime/stages/<stageName>/outcome—Retrieve the outcome of a previous stage within the context of the current stage.`

`/myPipelineRuntime/stages/<stageName>/tasks/<taskName>/outcome` —Retrieve the outcome of a task within the context of the current stage.

`/myPipelineRuntime/stages/<stageName>/gates/PRE/tasks/<taskName<propertyName>` —A gate task accesses a property on another task’s entry gate within the current pipeline runtime context.

`/myPipelineRuntime/stages/<stageName>/gates/POST/tasks/<taskName>/<propertyName>` —A gate task accesses a property on another task’s exit gate within the current pipeline runtime context.

`/myPipelineRuntime/stages/<stageName>/tasks/<taskName>/<propertyName>` —A stage task acccesses a property on itself.

`/myPipelineRuntime/stages/<stageName>/gates/PRE/tasks/<taskName>/<propertyName>` —A stage task accesses a property on an entry gate

`/myPipelineRuntime/stages/<stageName>/tasks/<taskName>/job/<propertyName>` —A task accesses a property on a job created by a task.

`/myPipelineRuntime/stages/<stageName>/tasks/<taskName>/workflow/<propertyName>` —A task accesses a property on a workflow created by a task.

` /myPipelineStageRuntime/…`

Resolves to the `pipelineStageRuntime` with the current pipeline runtime. This can be used by a procedure step executed from a stage task to retrieve information for the summary property of the stage runtime.

` /myProcedure/…`

Resolves to the procedure in which the current job step was defined. If the current job step is executing as part of a nested procedure, `/myProcedure/` refers to the innermost nested procedure.

` /myProcess/…`

Resolves to the application or component process in which the current job or job step was defined.

` /myProcessStep/…`

Resolves to the application or component process step in which the current job step was defined.

` /myProject/…`

Resolves to the project in which the current job step was defined. If the job step has nested procedure invocations, this is the project associated with the innermost nested procedure; for example, the project associated with `/myProcedure/` .

` /myResource/…`

Resolves to the resource object assigned to the current job step.

` /myResourcePool/…`

Resolves to the resource pool object that provided the resource for the current job step. Returns null if the step did not specify a resource pool.

` /myRetrievedArtifact/…`

Resolves to the current `RetrievedArtifact ` object available from a component process step.

`/myRetrievedArtifact/artifactVersion` —Version of the retrieved artifact.

`/myRetrievedArtifact/artifactName` —Name of the retrieved artifact.

`/myRetrievedArtifact/componentName` —Name of the component containing this retrieved artifact.

` /myService/…`

Resolves to the current microservice object.

` /myStage/…`

Resolves to the current pipeline Stage object.

` /myStageRuntime/…`

Resolves to the stage runtime object for the current stage runtime instance. This can be used to retrieve information for the summary property of the stage runtime.

Examples:

`myStageRuntime/tasks/test/outcome` —Test the runtime outcome of the previous task in the current stage.

`/myStageRuntime/runnumbers/1/output1` —Get the runtime outcome for specific runs in the current stage.

`/myStageRuntime/runNumbers/1/subFlowRuntime/prop1` —Access the property under the first stage run, previously set with `/myStageRuntime/prop1` .

`/myStageRuntime/currentrunNumber` —Get the current run number.

`/myStageRuntime/tasks/<taskName>/propertyName` —Accesses a property on another task in a stage.

`/myStageRuntime/tasks/groupTask/tasks/<subTaskName>/job/<jobid>` —Access an underlying group task from withing the current stage runtime context.

` /myStep/…`

Resolves to the current `Step` object, where `Step` refers to the static definition of a step that is part of a procedure. This is in contrast to a job step, which represents a step when it executes dynamically in a job. Use `/myJobStep/` to access the runtime job step.

` /myState/…`

Resolves to the current `State` object to reference built-in and custom state properties or find parameter values passed to that state. When accessed from a state, `/myState/` refers to that state. When accessed from a transition, `/myState` refers to the transition’s owning state. When accessed from a job or job step, `/myState/` refers to the state that launched that job as a subjob.

` /mySubjob/…`

Resolves to the current `Subjob ` object used to reference built-in and custom job properties, parameters passed to the job, and properties on steps within that job. When accessed from a transition, `/mySubjob/` refers to the subjob started by the state that owns that transition. This property path is particularly useful in conditions for On Completion transitions because the outcome or other information for the subjob can influence which state the workflow transitions to next.

` /mySubworkflow/...`

Resolves to the current `Subworkflow ` object used to reference built-in and custom workflow properties. When accessed from a transition, `/mySubworkflow/` refers to the subworkflow started by the state that owns that transition. This property path is particularly useful in conditions for On Completion transitions because the active state and other information for the subworkflow can influence which state the workflow transitions to next. You can access information about states and transitions belonging to the workflow by using the path `/mySubworkflow/states/someState` or `/mySubworkflow/states/someState/transitions/someTransition` .

` /myTask/...`

Resolves to the current pipeline `Task `.

` /myTaskRuntime/...`

Resolves to the current `Task ` runtime. This can be used to retrieve information for the summary property of the task runtime.

Examples:

`/myTaskRuntime/runnumbers/2/output1` —Get the runtime outcome for specific runs in the current task.

`/myTaskRuntime/runNumbers/1/subFlowRuntime/prop1` —Access the property under the first task run, previously set with `/myTaskRuntime/prop1` .

`/myTaskRuntime/propertyName` —Task that accesses the property on itself.

`/myTaskRuntime/job/propertyName` —Task that accesses the property on a job created by a task.

`/myTaskRuntime/workflow/propertyName` —Task that accesses the property on a workflow created by a task.

`/myTaskRuntime/currentRunNumber` -Get the current run number.

` /myTransition/...`

Resolves to the current `Transition ` object used to reference intrinsic and custom transition properties. `/myTransition/` is accessible only from a transition.

` /myTriggeringPipelineRuntime/...`

In a situation where a pipeline triggers a sub-pipeline, this property accesses the runtime context of the parent pipeline from within the sub-pipeline runtime context.

Examples:

`/myTriggeringPipelineRuntime/stages/<stageName>/outcome—Retrieve the outcome of a previous stage within the context of the current stage of the triggering pipeline runtime context.`

`/myTriggeringPipelineRuntime/stages/<stageName>/tasks/<taskName>/outcome` —Retrieve the outcome of a task within the context of the current stage of the triggering pipeline.

` /myUser/...`

This property is used only if the current session is associated with the predefined `admin ` user, a user defined as `local`, or a user defined by a Directory Provider (LDAP or ActiveDirectory). This property cannot be used if the user is a project principal, which is normally the case when running inside a CloudBees CD step.

For example, with an interactive login you can use: `ectool getProperty /myUser/userName` (to get the username of the logged in user, or...) `ectool getProperty /myUser/email` (to get the email address)

` /myWorkflow/...`

Resolves to the current workflow object used to reference built-in and custom workflow properties. When accessed from a state or transition, `/myWorkflow/` refers to the "owning" workflow. When accessed from a job or job step, `/myWorkflow/` refers to the workflow whose state launched that job as a subjob. You can access information about states and transitions belonging to the workflow by using the path `/myWorkflow/states/someState` or `/myWorkflow/states/someState/transitions/someTransition` .

` /myWorkflowDefinition/...`

Resolves to the current `WorkflowDefinition` object used to reference built-in and custom workflow properties.

` /myWorkspace/...`

Resolves to the current `Workspace` object associated with the current job step. :PRODUCT: CloudBees CD :LATEST-VERSION: 10.0.2

Property name substitutions

Property names can contain references to other properties, which are then substituted into the property name before looking it up. For example, consider the following property name:

`/myStep/$[/myProcedure/name]`

If the value of `/myProcedure/name` is "xyz", the property above is equivalent to `/myStep/xyz` .

Expandable properties

Property values can contain property references using the `"$[]"` notation.

For example:

1. Create a property named "foo" with a value of `hello $[bar]` .
2. Create a property named "bar" with a value of `world` .
3. Reference "foo" (either using `$[foo]` or `ectool getProperty foo`). The value "hello world" is returned.

If you want just the literal value of "foo" (useful in the UI, for example), you can use the `expand` option in ectool:

`ectool getProperty foo --expand false` . The value `"hello $[bar]"` will be returned.

Properties are expanded by default when you use `getProperty` or `getProperties` .

If the value of a property contains `"$[]"` but you do not want it to be interpreted as a property reference, you can use the expandable option:

`ectool setProperty symbols '$[!@]#' --expandable false` .

This option can be toggled in the web UI as well. Properties are expandable by default.

Because you cannot control where your expandable property might be referenced (and therefore which context is used during expansion), we recommend using absolute paths when referencing a property from the value of another property.

In the example above, if you define both "foo" and "bar" as properties on a project "proj1", you might assume there is no problem with the value of "foo". However, if you later reference "foo" from a job under "proj1" (for example, `$[/myProject/foo`), foo will be referenced with the job step as its context. Therefore, when the value of "foo" is expanded, you will get a PROPERTY_REFERENCE_ERROR because "bar" is not defined in the context of the job step.

Custom property names and values

The following properties are used by the standard CloudBees CD UI, so you should use these property names whenever possible, and avoid using these names in ways that conflict with the definitions below.

- `compiles` —the number of files compiled during the job step

- `diagFile` —the filename in the top-level directory of the job’s workspace, containing diagnostics extracted from the step’s log file
- `errors` —the number of errors (compilation failures, test failures, application crashes, and so on) that occurred during the job step. When property errors are set by postp, the step outcome is set to error also.
- `tests` —the number of tests executed by the job step, including successes and failures
- `testsSkipped` —the number of tests skipped during the job step
- `warnings` —the number of warnings that occurred during the job step. When property warnings is set by postp, the step outcome is set to warning also.
- `preSummary` —if this property exists, its value is displayed in the "Status" field (on the Job Details page) for this step. This property appears before whatever would normally be displayed for status. If the property contains multiple lines separated by newline characters, each line is displayed on a separate line in the status field.
- `postSummary` —if this property exists, its value is displayed in the "Status" field (on the Job Details page) for this step. This property appears after whatever would normally be displayed for status. If the property contains multiple lines separated by newline characters, each line is displayed on a separate line in the status field.
- `summary` —if this property exists, its value is displayed in the "Status" field (in the job reports) for this step, replacing whatever would normally be displayed for status. If the property contains multiple lines separated by newline characters, each line is displayed on a separate line in the status field.

The property hierarchy

All CloudBees CD properties fall into a hierarchical structure, and you can reference any property using an absolute path from the root of the hierarchy. This includes properties you define and intrinsic properties defined by CloudBees CD.

For example, each step contains a property "resource" that provides the resource name to use for that step. All steps of a procedure exist as property sheets underneath the procedure. All procedures in a project exist as property sheets underneath the project, and so on.

The examples below illustrate some nesting relationships between objects.

For example, the notation " project /procedures[procedureName]" means each project object contains a property sheet named "procedures" holding all procedures defined within that project. Within the procedures property sheet, there is a nested property sheet for each procedure, named after the procedure. Thus, the name "/projects[a]/procedures[b]" refers to a procedure named "b" contained in a project named "a."

- Each project contains procedures, schedules, credential definitions, workflow definitions, and workflows: project /procedures[procedureName] project /schedules[scheduleName] project /credentials[credentialName] project /workflowDefinitions[workflowName] project /workflows[workflowName]
- Each procedure contains steps and parameters. The parameters are "formal parameters," meaning they specify parameter names the procedure will accept, whether each parameter is required, and so on: procedure /steps[stepName] procedure /formalParameters[paramenterName]
- Steps that invoke subprocedures contain parameter values for the subprocedure. These are called "actual parameters" because they provide actual values that will be passed into the subprocedure: step /actualParameters[parameterName]
- Each job contains a collection of job step objects for steps in the outermost procedure, along with parameter values passed into the job when it was invoked: job /jobSteps[stepName] job /actualParameters[parameterName]
- If a job step invokes a nested subprocedure, its property sheet contains parameter values that were passed into the nested subprocedure, plus all job steps corresponding to that procedure: jobStep /actualParameters[parameterName] jobStep /jobSteps[stepName]

- Schedules contain parameter values for the procedures they invoke. These are called "actual parameters" because they provide actual values passed into the procedure: schedule /actualParameters[parameterName]
- Workflow definitions contain state definitions: workflowDefinition /stateDefinitions[stateDefinitionName]
- Transition definitions contain actual parameters: transitionDefinition /actualParameters[parameterName]
- State definitions contain transition definitions, actual parameters, and formal parameters: stateDefinition /actualParameters[parameterName] stateDefinition /formalParameters[parameterName] stateDefinition /transitionDefinitions[transitionDefinitionName]
- Workflows contains states: workflow /states[stateName]
- Transitions contain actual parameters: transition /actualParameters[parameterName]
- States contain transitions, actual parameters, and formal parameters: state /actualParameters[parameterName] state /formalParameters[parameterName] state /transitions[transitionName]

Special property references

CloudBees CD also supports several special property reference forms that are described in the following subsections.

increment

Use this form to increment the value of an integer property before returning its value. For example, suppose property xyz has the value 43. The property reference `$[/increment xyz]` first increments the value of property xyz to 44, then returns 44.

timestamp

Use this form to generate a formatted timestamp value. For example, the property reference `$[/timestamp yyyy-MMM-d hh:mm]` returns the current time in a form such as "2007-Jun-19 04:36". The pattern following `/timestamp` specifies how to format the time and defaults to "yyyyMMddHHmm". The pattern follows conventions for the Java class SimpleDateFormat, where various letters are substituted with various current time elements. For more information about the SimpleDateFormat class, see link:https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html\https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html . Here are some of the supported substitutions:

y—Year, such as "2007" or "07"

M—Month, such as "April", Apr", or "04"

w—Week in year, such as "27"

W—Week in month, such as "3"

D—Day in year, such as "194"

d—Day in month, such as "18"

E—Day in the week, such as "Tuesday" or "Tue"

a—am/pm marker, such as "PM"

H—Hour in day (0-23), such as "7"

h—Hour in am/pm (1-12), such as "11"

m—Minute in hour, such as "30"

s—Second in minute, such as "15"

S—Millisecond, such as "908"

z—Time zone such as "Pacific Standard Time", "PST", or "GMT-08:00"

Z—Time zone such as "-0800"

Repeated letters cause longer forms to be substituted. For example, "yy" substitutes the last 2 digits of the current year, whereas "yyyy" substitutes the 4-digit year number. Single quotes can be used to substitute text directly without the above interpretations.

javascript

This form executes a Javascript code fragment inside the CloudBees CD server and returns the result computed by that code. For example, `$[/javascript 4*2]` returns "8.0". Javascript code can be arbitrarily long and include multiple statements. The value of the last statement is returned by the property reference.

Javascript code executes in an interpreter that provides access to CloudBees CD properties:

1. The normal way to access property values is through Javascript objects. Global Javascript objects named "server," "projects," and so on, exist and correspond to all top-level objects in an absolute property path. For example, there is a Javascript object named `server` that corresponds to the path `/server`, and a Javascript object named `myJob` that corresponds to the path `/myJob`. You can use either `"."` or `"[]"` notation to access properties within the object. For a more complete list of top-level objects, see Absolute property paths . Examples: `$[/javascript myJob.mightExist]` this is a safe way to refer to an optional parameter `$[/javascript (myJobStep.errors > 0) ? "step failed" : "no errors"]` test a value and return another string based on the result `$[/javascript server.settings.ipAddress]` refer to a property in a nested property sheet `$[/javascript server["CloudBees"].installDirectory]` if the property name has a space, use `[""]` notation `$[/javascript server["CloudBees"]["installDirectory"]]` do not use the `"." ` in front of the `"["`
2. You can also call the function, `"getProperty"`. It takes a property name as argument and returns the value of that property. The case where this is most useful is when you want to access another special property reference. Examples: `$[/javascript getProperty ("/timestamp yyyy-MMM-d hh:mm")]` returns the current time `$[/javascript getProperty ("/increment /myJob/jsCount")]` this is the only way to change a property via javascript For example, consider the following property reference: `$[/javascript (getProperty("/myJobStep/errors") > 0) ? "step failed" : "no errors"]` This example returns "step failed" if the property "errors" on the current job step had a value greater than zero, and it returns "no errors" otherwise.
3. If calling the `"setProperty"` function, similar to `getProperty`, there are two variations on the function:
    - A global function variation that uses the current context object. The global function takes 2 or 3 arguments. The 3-argument version takes a context object, a path, and a value. The 2-argument version omits the context object and uses the current context object (for example, job step). Example: `setProperty (myProject, "foo", "bar")` would set the value of the "foo" property on the current project to "bar".
    - An object function variation that can be called on objects, which uses that object as a context object. The object function takes two arguments: a "path" and a "value". Example: `server.setproperty ("foo", "bar")` would set the "foo" property on the server object to the value "var".

Intrinsic properties listed by object type

CloudBees CD Deploy intrinsic properties

For details about the intrinsic properties for each object in CloudBees CD Deploy, use the `describeObject ` API command. For example, to list intrinsic properties for the application object:

ec-perl

```
$cmdr->describeObject application
```

ectool

```
ectool describeObject application
```

For more information about the `describeObject` command, see Miscellaneous management commands.

Automation Platform intrinsic properties

This section describes the intrinsic properties for some (but not all) of the objects in the Automation Platform. For each object type in the following list, there is a link to a table that includes a list of intrinsic properties applicable to that object and provides the property type and description.

`acl`	`repository`
`artifact`	`resource`
`artifactVersion`	`resourcePool`
`credential`	`resourceUsage`
`directoryProvider`	`schedule`
`emailConfig`	`server`
`emailNotifier`	`state`
`formalParameter`	`stateDefinition`
`gateway`	`step`
`group`	`transition`
`job`	`transitionDefinition`
`jobStep`	`user`
`logEntry`	`workflow`
`procedure`	`workflowDefinition`
`project`	`workspace`
`property`	`zones`
`propertySheet`	

Property Type definitions in the Automation Platform

This table provides Property Type definitions for each property listed in the following tables. (A Property Type precedes each property description in the Description column.)

Type	**Definition**
boolean	One of two possible values—either true or false. These values are more frequently represented by the numbers "0" and "1", where "0" equals false and "1" equals true.
date	A millisecond precision UTC date in ISO 8601 form: `[ YYYY]-[MM]-[DD]T[hh]:[mm]Z` For example, `2007-06-19T04:36:22.000Z`

Type	Definition
id	Each time an object is created, CloudBees CD generates a unique ID number for that object.
name	This is a unicode string value with a maximum of 255 characters.
number	This is a simple integer numeric value.
reference	This property refers to another object.
string	This is a unicode string value with a maximum CLOB size of the database, but only the first 450 characters are indexed, which means a defined search will not "see" beyond the first 450 characters.

In the following tables, the Description column displays the property type preceding the property description.

Object Type: acl Description: **An acl is an Access Control List.**		
Property Name	Description	
aclId	` id:` The unique identifier for this `acl` object. Other objects can refer to this `acl` by its ID.	
inheriting	` boolean:` If true, the ACL inherits ACEs from the ACL's parent.	
ownerType	This is any type of object the ACL controls and can be any of the objects listed below.	
	` acl admin artifactartifactVersioneventformalParametergroupjobjobSteplicensenotifierprocedureprocedureStep`	` project property propertySheet repository resource resourcePoolserverschedulesystemObject useruserSettings workspaceplugin`
parentId	` id:` The parent ACL.	

Object Type: **artifact Description: An artifact is an object that contains zero or more artifact versions. An artifact has two purposes: 1. To group artifact versions and provide a template for naming the versions 2. To restrict who can publish artifact versions, based on** `groupId:artifactKey`	
Property Name	Description
acl	` reference:` ` acl `
artifactId	` id:` The artifact's ID number.
artifactKey	` string:` User-specified identifier for this artifact. This field may consist of alphanumeric characters, spaces, underscores, hyphens, and periods.
artifactName	` name:` The name of this artifact.
artifactVersionNameTemplate	` name:` The template for artifact version names published to this artifact.
createTime	` date:` The time when this object was created.
description	` string:` A user-specified text description of the object.

Object Type: artifact Description: An artifact is an object that contains zero or more artifact versions. An artifact has two purposes: 1. To group artifact versions and provide a template for naming the versions 2. To restrict who can publish artifact versions, based on `groupId:artifactKey`

groupId	` id:` A user-generated group name for this artifact. This field may consist of alphanumeric characters, spaces, underscores, hyphens, and periods.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
propertySheet	` reference: ` ` propertySheet `

Object Type: artifactVersion Description: An artifact version is an object that represents a user-defined unit of related files typically produced by one job and consumed by one or more other jobs.

Property Name	Description
acl	` reference: acl`
artifactKey	` string: ` User-specified identifier for this artifact. This field may consist of alphanumeric characters, spaces, underscores, hyphens, and periods.
artifactName	` name: ` The name of the artifact.
artifactVersionId	` id: ` The CloudBees CD-generated ID number for this artifact version.
artifactVersionName	` name: ` The name of the artifact version.
artifactVersionState	` string: ` Possible values are: `available|publishing|unavailable`
buildNumber	` number:` User-defined build number component of the version attribute for the artifact version.
createTime	` date: ` The time when this object was created.
description	` string:` A user-specified text description of the object.
groupId	` id:` A user-generated group name for this artifact. This field may consist of alphanumeric characters, spaces, underscores, hyphens, and periods.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
majorMinorPatch	` string:` ` major.minor.patch` component of the version attribute for the artifact.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
propertySheet	` reference: propertySheet`
publisherJobId	` id:` The CloudBees CD-generated ID number for the job that published the artifact version.
publisherJobName	` string:` The name of the job that published the artifact version.

Object Type: artifactVersion Description: **An artifact version is an object that represents a user-defined unit of related files typically produced by one job and consumed by one or more other jobs.**	
publisherJobStepId	` id:` The CloudBees CD-generated ID number for the job step that published the artifact version.
qualifier	` string:` User-defined qualifier component of the version attribute for the artifact.
repositoryName	` name:` The name of the artifact repository.
version	` string:` An artifact version specification uses the following form: major.minor.patch.qualifier.buildNumber

Object Type: credential Description: **In CloudBees CD, a credential is an object that stores a username and password for later use.**	
Property Name	Description
acl	` reference: acl`
createTime	` date: ` The time when this object was created.
credentialId	` id:` The credential's ID number.
credentialName	` name:` The name of this credential.
description	` string:` A user-specified text description of the object.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
password	` string: ` The password corresponding to the username and this credential.
projectName	` name:` The name of the project that contains this credential.
propertySheet	` reference: propertySheet`
userName	` name:` A saved string that represents the name portion of a credential, typically a user account name.

Object Type: directoryProvider Description: **A directoryProvider is an object containing information about an external directory service (LDAP or ActiveDirectory).**	
Property Name	Description
acl	` reference: acl`
commonGroupNameAttribute	` string: ` The attribute in a group record that contains the common group name. If specified, this name is used only when searching for groups from an external provider.
createTime	` date:` The time when this object was created.
description	` string:` A user-specified text description of the object.
directoryProviderId	` id:` The ID of the directory provider.

Object Type: directoryProvider Description: **A directoryProvider is an object containing information about an external directory service (LDAP or ActiveDirectory).**	
domainName	` string: ` The domain name from which Active Directory servers are automatically discovered.
emailAttribute	` string: ` The attribute in a user record that contains the user's email address. If the attribute was not specified, the account name and domain name are concatenated to form an email address.
enableGroups	` boolean:` Determines whether or not external groups are enabled for the directory provider. Defaults to "true".
fullUserNameAttribute	` name: ` The attribute in a user record that contains the user's full name (first and last) for display in the UI.
groupBase	` string: ` This string is prepended to the ` basedn` to construct the directory DN that contains group records.
groupMemberAttributes	` string: ` A comma-separated attribute name list that identifies a group member.
groupMemberFilter	` string: ` Two common forms of group record in LDAP directories: POSIX style groups where members are identified by account name, and `groupOfNames` or `uniqueGroupOfNames` records where members are identified by the full user DN. Both forms are supported, so the query is passed to parameters: ` "{0}"` is replaced with the full user record DN, and `"{1}"` is replaced with the user's account name.
groupNameAttribute	` name:` The group record that contains the name of the group.
groupSearchFilter	` string: ` This LDAP query is performed in the context of the groups directory to enumerate group records.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
managerDn	` name:` The DN of a user who has read-only access to LDAP user and group directories.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
propertySheet	` reference: propertySheet`
providerIndex	` number:` The index that specifies the search order across multiple directory providers. For example: 2 LDAP providers, one with index "0" and one with index "1" means the providers will be searched in that numerical order.
providerName	` name:` This human-readable name will be displayed in the user interface to identify users and groups that come from this provider.
providerType	` string: <ldap|activedirectory>`
realm	` string: ` An identifier (string)used for LDAP directory providers so users and groups (within LDAP) can be uniquely identified in "same name" collisions across multiple directory providers. The realm is appended to the user or group name when stored in the CloudBees CD server. For example, <user>@dir (where the realm is set to "dir").
url	` string: ` The server URL is in the ` formprotocol://host:port/basedn`. Protocol is either ldap or ldaps (for secure LDAP).

Object Type: directoryProvider Description: **A directoryProvider is an object containing information about an external directory service (LDAP or ActiveDirectory).**	
userBase	` string: ` This string is prepended to the `basedn` to construct the directory DN that contains user records.
userNameAttribute	` name: ` The attribute in a user record that contains the user's account name.
userSearchFilter	` string: ` This LDAP query is performed in the context of the user directory to search for a user by account name. The string "{0}" is replaced with the user's login ID. Typically, the query compares a user record attribute with the substituted user login ID.
userSearchSubtree	` boolean:` If true, the subtree below the user base is searched recursively.
useSSL	` boolean:` This flag is used to specify SSL to communicate with your Active Directory servers. Note: Transport Layer Security (TLS) has replaced Secure Sockets Layer version 3.0 (SSLv3) on the CloudBees CD web server and the CloudBees CD server.

Object Type: emailConfig Description: **An emailConfig is an object that stores information created and used to communicate with the email server.**	
Property Name	Description
acl	` reference: acl`
configName	` string: ` The name of the email configuration.
createTime	` date:` The time when this object was created.
description	` string:` A user-specified text description of the object.
emailConfigId	` id:` The CloudBees CD-generated ID for the email configuration.
emailConfigName	` string: ` The name of the email configuration.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
mailFrom	` string: ` The email address used as the email sender address for notifications.
mailHost	` string: ` The name of the email server host.
mailPort	` number:` The port number for the mail server. The protocol software determines the default value (25 for SMTP and 465 for SSMTP).
mailProtocol	` string: ` This is either SSMTP or SMTP (not case sensitive). Default is SMTP.
mailUser	` name: ` An individual or a generic name like "CloudBees CD" — the name of the email user on whose behalf CloudBees CD sends email notifications.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.

Object Type: emailConfig Description: An emailConfig is an object that stores information created and used to communicate with the email server.	
propertySheet	\` reference: propertySheet\`

Object Type: emailNotifier Description: An emailNotifier is an object that stores information created and used to notify users about various types information, including status.	
Property Name	Description
acl	\` reference: acl\`
condition	\` string:\` Only send mail if the condition evaluates to "true". The condition is a string subject to property expansion. The notification will NOT be sent if the expanded string is "false" or "0". If no condition is specified, the notification is ALWAYS sent.
configName	\` string: \` The name of the email configuration.
container	\` id:\` The object ID to which the email notifier is attached (for example: procedure-123).
containerType	\` string:\` The type of object that the notifier is attached to (procedure, step, job, jobstep).
createTime	\` date:\` The time when this object was created.
description	\` string:\` A user-specified text description of the object.
destinations	\` string:\` A space-separated list of valid email addresses, email aliases, or CloudBees CD usernames, or a string subject to property expansion that expands into such a list.
emailNotifierId	\` id:\` The CloudBees CD-generated ID for the email notifier.
eventType	\` string:\` < `onEnter|onStart|onCompletion` > `onStart` triggers an email notification when the job or job step begins. `onCompletion` , default, triggers an email notification when the job finishes, no matter how it finishes.
formattingTemplate	\` string: \` The email address used as the email sender address for notifications.
lastModifiedBy	\` name:\` This shows who (generally, a username) last modified this object.
modifyTime	\` date:\` The time when this object was last modified.
notifierName	\` name:\` The name of the email notifier.
owner	\` name:\` The person (username) who created the object.
propertySheet	\` reference: propertySheet\`

Object Type: formalParameter Description: A formalParameter is a parameter expected by a procedure, including its name, a default value, and an indication of whether the parameter is required. Formal parameters are different from "actual parameters"--- formal parameters define the type of parameters a procedure is expecting, and actual parameters provide values to use at run-time.	
Property Name	Description
container	\` string: \` An object ID for a "container" that contains formal parameters.

Object Type: formalParameter Description: **A formalParameter is a parameter expected by a procedure, including its name, a default value, and an indication of whether the parameter is required. Formal parameters are different from "actual parameters"--- formal parameters define the type of parameters a procedure is expecting, and actual parameters provide values to use at run-time.**	
containerType	` string: ` The type of object containing the formal parameter.
createTime	` date:` The time when this object was created.
defaultValue	` string:` The `formalParameter` 's default value.
description	` string:` A user-specified text description of the object.
expansionDeferred	` boolean:` Determines whether or not the property expansion is deferred.
formalParameterId	` id:` This is this formal parameter's ID.
formalParameterName	` name:` This is this formal parameter's name.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
required	` boolean:` If true, a value for this parameter must be supplied when the procedure is called.
type	` name:` The custom type of `formalParameter.`

Object Type: gateway Description: **To communicate with a resource, workspace, or artifact repository server in another zone, a "gateway" must be created. A gateway object contains two resource (agent) machines, for example, GatewayResource1 and GatewayResource2—each configured to communicate with the other. One gateway resource resides in the source zone and the other in the target zone. A gateway is bidirectional and informs the CloudBees CD server that each gateway machine is configured to communicate with its other gateway machine (in another zone).**	
Property Name	Description
acl	` reference: acl`
createTime	` date:` The time when this object was created.
description	` string:` A user-specified text description of the object.
gatewayDisabled	` boolean: ` If set to 1 (true), the gateway is disabled.
gatewayId	` id:` The unique CloudBees CD-generated ID for this gateway.
gatewayName	` string:` The name of the gateway.
hostName1	` string:` The agent host name where Resource1 resides. This host name is used by Resource2 to communicate with Resource1 . Do not specify this option is you want to use the host name from Resource1's definition.
hostName2	` string: The agent host name where Resource2 resides. This host name is used by Resource1 to communicate with Resource2. Do not specify this option is you want to use the host name from Resource2's definition.`
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
modifyTime	` date:` The time when this object was last modified.

Object Type: gateway Description: **To communicate with a resource, workspace, or artifact repository server in another zone, a "gateway" must be created. A gateway object contains two resource (agent) machines, for example, GatewayResource1 and GatewayResource2—each configured to communicate with the other. One gateway resource resides in the source zone and the other in the target zone. A gateway is bidirectional and informs the CloudBees CD server that each gateway machine is configured to communicate with its other gateway machine (in another zone).**	
owner	` name:` The person (username) who created the object.
port1	` number:` The port number used by Resource1 —defaults to the port number used by the resource.
port2	` number:` The port number used by Resource2 —defaults to the port number used by the resource.
propertySheet	` reference: propertySheet`
resourceName1	` string:` The name of your choice for the first of two required gateway resources. Do **not** include "spaces" in a resource name.
resourceName2	` string:` The name of your choice for the second of two required gateway resources. Do **not** include "spaces" in a resource name.

Object Type: group Description: **This is any group of users known to the CloudBees CD server, including groups defined locally within the server and those groups defined in external repositories such as LDAP or ActiveDirectory.**	
Property Name	Description
acl	` reference: acl`
createTime	` date:` The time when this object was created.
groupId	` id:` The unique CloudBees CD-generated group ID.
groupName	` name:` This is this group's name.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
modifyTime	` date:` The time when this object was last modified.
mutable	` boolean:` If true, the member list of this group is editable within CloudBees CD via the web UI or the `modifyGroup` API.
owner	` name:` The person (username) who created the object.
propertySheet	` reference: propertySheet`
providerName	` name:` The name of the directory provider that controls this group.

Object Type: job Description: **A job is a CloudBees CD structure associated with invoking a procedure. A new job is created each time a procedure begins to execute. The job keeps track of all data associated with the procedure's execution, such as the running time of each step and any errors that may occur during the step. CloudBees CD retains job information after the job completes so you can examine what occurred.**	
Property Name	Description
abortStatus	` string:` If set, indicates the step was aborted. Values will be either `ABORT` or `FORCE_ABORT` —indicating how the step was aborted.
abortedBy	` name:` This is the user who issued the "abort."

Object Type: job Description: **A job is a CloudBees CD structure associated with invoking a procedure. A new job is created each time a procedure begins to execute. The job keeps track of all data associated with the procedure's execution, such as the running time of each step and any errors that may occur during the step. CloudBees CD retains job information after the job completes so you can examine what occurred.**	
acl	` reference: acl`
actualParameter	` reference: propertySheet` An `actualParameter` is an object that provides the value for a parameter, which is passed to a procedure when it is invoked. Actual parameters can be provided for jobs and nested subprocedures within a job. Actual parameters are different from "formal parameters"- formal parameters define the parameters a procedure is expecting, and actual parameters provide values to use at run-time.
callingState	` string:` The full property path to the "calling state", which can appear on `subjobs` and `subworkflows` of a workflow.
callingStateId	` id:` The CloudBees CD-generated ID number for the calling state.
combinedStatus	Provides more inclusive step status output—the resulting query output may contain up to three sub-elements: `status|message|properties`
createTime	` date:` The time when this object was created.
credentialName	` name:` The name of the credential being used for impersonation when the job runs commands on a resource.
deleted	The object was marked for background deletion. Possible values are "0" or "1". Default is "0" (not set).
directoryName	` name:` The name of this job's directory within each workspace for this job.
elapsedTime	` number:` The number of milliseconds between the start and end times for the job.
errorCode	` errorCode:` When the outcome is `error` , this property displays the error code, identifying which error occurred.
errorMessage	` string:` When the outcome is `error` , this property displays the error description.
external	` boolean :` If 'true", the job is external.
finish	` date:` The time this job completed.
jobId	` id:` This is this job's ID number, which is a UUID.
jobName	` name:` This is this job's name.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
launchedByUser	` name:` The name of the user or project principal that explicitly launched the job. This property is blank when the job is launched by a schedule.
licenseWaitTime	The sum of time all job steps had to wait for a license.
liveProcedure	` name:` The current procedure name from which this job was created – if the procedure was renamed since the job launched, this will be the procedure's new name, and if the procedure was deleted, this will be null.

Object Type: job Description: A job is a CloudBees CD structure associated with invoking a procedure. A new job is created each time a procedure begins to execute. The job keeps track of all data associated with the procedure's execution, such as the running time of each step and any errors that may occur during the step. CloudBees CD retains job information after the job completes so you can examine what occurred.	
liveSchedule	` name:` The current schedule name that launched this job – if the schedule was renamed since the job was launched, this will be the schedule's new name, and if the schedule was deleted, this will be null.
modifyTime	` date:` The time when this object was last modified.
outcome	The overall result of the job: `success` , `warning` , `error` , or `skipped` .
owner	` name:` The person (username) who created the object.
priority	Values can be `low,` `normal` (default), `high` , or `highest` .
procedureName	` name:` The name of the procedure that defines the job's steps.
projectName	` name:` The name of the project that contains this job.
propertySheetId	` reference: propertySheet`
resourceWaitTime	The sum of time all job steps had to wait for a resource. This could indicate that eligible resources for the step have reached their step limit, are in-use but the step requires a resource exclusively, or resources are down.
runAsUser	` name:` The name of the user being impersonated in this job.
scheduleName	` name:` The schedule name that launched this job – this field differs from `liveSchedule` in that it is written at job creation time only, and not changed, even if the schedule is renamed or deleted.
start	` date:` The time when this job began executing.
status	Possible values are: `pending` —the job was created, but it is waiting for prerequisite steps to complete `runnable` —the job step is waiting for a resource `scheduled` —the job was assigned a resource, but the command has not started running `running` —the job/job step is running a command on the assigned resource `completed` —the job/job step has completed
totalWaitTime	The total sum of license, resource, a precondition, and workspace wait times job steps were restricted and had to wait to process.
workspaceWaitTime	The sum of time all job steps had to wait for a workspace.

Object Type: jobStep Description: A jobStep is an instance of a step that ran—a single invocation of a command on a resource or a single call to a subprocedure within the context of a job .	
Property Name	Description
abortStatus	` string:` If set, indicates the step was aborted. Values will be either `ABORT` or `FORCE_ABORT` —indicating how the step was aborted.
abortedBy	` name:` The user who issued the "abort."

Object Type: jobStep Description: A jobStep is an instance of a step that ran—a single invocation of a command on a resource or a single call to a subprocedure within the context of a job .	
acl	` reference: acl`
actualParameter	` reference: propertySheet` An `actualParameter` is an object that provides the value for a parameter, which is passed to a procedure when it is invoked. Actual parameters can be provided for jobs and nested subprocedures within a job. Actual parameters are different from "formal parameters"- formal parameters define the parameters a procedure is expecting, and actual parameters provide values to use at run-time.
alwaysRun	` boolean:` If true, this step runs even if the job is aborted before the step completes. Note that a "force abort" will abort an `alwaysRun` step.
assignedResourceName	` name:` The name of the resource assigned to this step by the resource scheduler.
broadcast	` boolean:` If true, this step runs on all resources in a pool.
combinedStatus	Provides more inclusive step status output—the resulting query output may contain up to three sub-elements: `status|message|properties`
command	` string:` This property specifies the command this step runs.
condition	` string:` If this element is not present, the event is ALWAYS triggered. If specified, the event is triggered only if the value of the condition argument is TRUE; if not true, a boolean value of "false" or "0" was used. Condition arguments can be a literal, a fixed value string, or a string subject to property expansion.
conditionExpanded	` boolean:` The result of the expansion on the step condition.
createTime	` date:` The time when this object was created.
delayUntil	For a step that was rescheduled due to a resource or workspace problem, this is the next time the step will be eligible to run.
elapsedTime	` number:` The number of milliseconds between the start and end times for the `jobStep` .
errorCode	` errorCode:` This property appears when the outcome is `error` and displays the error code, identifying which error occurred.
errorHandling	This is the error handling policy copied from the procedure step and indicates how the step responds to errors: `failProcedure` —The current procedure continues, but the overall status is error (default). `abortProcedure` —Aborts the current procedure, but allows already-running steps in the current procedure to complete. `abortProcedureNow` —Aborts the current procedure and terminates running steps in the current procedure. `abortJob` —Aborts the entire job, terminates running steps, but allows alwaysRun steps to run. `abortJobNow` —Aborts the entire job and terminates all running steps, including alwaysRun steps. `failProcedure` — The current procedure continues, but the overall status is error (default). `ignore` —Continues as if the step succeeded.
errorMessage	` string:` When the outcome is `error` , this property displays an error message description.

Object Type: jobStep Description: A jobStep is an instance of a step that ran—a single invocation of a command on a resource or a single call to a subprocedure within the context of a job .	
exclusive	` boolean:` If true, this step acquires and retains its resource exclusively.
exclusiveMode	` string:` Possible values are: `none` , `job` , `step` , `call`
exitCode	` number:` This is this step's exit code.
external	` boolean:` If "true", the job is external.
finish	` date:` The time when this job step completed.
hostName	` name:` The name of the host where this step was invoked (copied from the resource)
job	` id:` This is the ID number of the job that owns the job step. The string representation of the job is its name, so `$[job]` evaluates to the job name, but `$[job/foo]` is also legal and refers to the foo property of the job.
jobId	` id:` This is this job's ID number, which is a UUID.
jobName	` name:` This is the name of this step's job.
jobStepId	` id:` This is this step's ID number.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
licenseWaitTime	The length of time this job step had to wait to process because the license limit was reached or exceeded.
liveProcedure	` string:` The current procedure name for the procedure step from which the job or job step was created – if the procedure step was renamed since the job or job step was launched, this is the procedure step's new name, and if the procedure step was deleted, this will be null.
liveProcedureStep	` name:` This property shows the current procedure name for the procedure step from which this job step was created – if the procedure step was renamed since the job was launched, this will be the procedure step's new name, and if the procedure step was deleted, this will be null.
logFileName	` name:` The name of the log file produced by this step, relative to the job's workspace directory.
modifyTime	` date:` The time when this object was last modified.
outcome	` string:` The overall result of the job—possible values are: `success,` `warning,` `error,` or `skipped`
owner	` name:` The person (username) who created the object.
parallel	` boolean:` If true,this step runs in parallel with other adjacent steps also marked to run in parallel.
postExitCode	` number:` This is this step's post processor exit code.
postLogFileName	` name:` The log file name produced by this step's post processor.
postProcessor	` string:` The post processor name used to gather information about this step.

Object Type: jobStep Description: A jobStep is an instance of a step that ran—a single invocation of a command on a resource or a single call to a subprocedure within the context of a job .	
precondition	` string:` By default, if a step has no precondition, it will run when scheduled. Set this property to make a step wait until one or more dependent conditions are met. When a job step is eligible to transition from pending to runnable, a precondition is evaluated. A precondition is a fixed text or text embedding property reference that is evaluated to TRUE or FALSE. An empty string, a \"0\" or \"false\" is interpreted as FALSE. Any other result string is interpreted as TRUE. The step will block until the precondition is TRUE.
procedureName	` name:` The name of the procedure that contains this `jobStep` .
projectName	` name:` The name of the project that contains this jobStep.
propertySheet	` reference: propertySheet`
releaseExclusive	` boolean:` A Boolean value indicating whether this step should release its resource upon completion.
releaseMode	` string:` Possible values are: `none` , `release` , `releaseToJob`
resourceName	` name:` The name of the resource or pool this step should use to run on.
resourceSource	This property indicates whether the resource for the job step is explicitly specified by the step or was defaulted from the procedure: procedure or `procedureStep.`
resourceWaitTime	The length of time this job step stalled because it could not get a resource. This could indicate that eligible resources for the step have reached their step limit, are in-use but the step requires a resource exclusively, or resources are down.
retries	` number:` This is a number—the number of retries before the request times out.
runAsUser	` name:` The name of the user being impersonated in this job step.
runnable	` date:` The time when the step became runnable.
runTime	` number:` The number of milliseconds the step command spent running on a resource.
shell	` name:` The shell used to execute the step's commands on a resource. The script name is inserted into the command at the position of a `"{0}"` marker in the command, or appended as a final argument if no marker is present. A filename suffix adjacent to the marker will be appended to the script name. For more information on shells, see the Shell definition in the Step Help topic.
start	` date:` The time when this job step began executing.
status	Possible values are: `pending` —the job step was created, but it is waiting for prerequisite steps to complete `runnable` —the job step is waiting for a resource `scheduled` —the job step was assigned a resource, but the command has not started running `running` —the job/job step is running a command on the assigned resource `completed` —the job/job step has completed
stepName	` name:` This is this step's name.

Object Type: jobStep Description: **A jobStep is an instance of a step that ran—a single invocation of a command on a resource or a single call to a subprocedure within the context of a job .**	
subprocedure	` name:` The nested procedure name to call when this step executes.
subproject	` name:` This property appears if the subprocedure element is present and specifies the project to which the subprocedure belongs (by default, the current project is used.)
timeLimit	` number:` The maximum length of time this step is allowed to run.
timeout	` date:` The time when this job step will be automatically aborted if it has not yet completed.
totalWaitTime	The sum of resource, workspace, and license wait times for this job step.
waitTime	` number:` The number of milliseconds the step spent between runnable and running (for example, waiting for a resource).
workingDirectory	` name:` The name of this step's working directory.
workspaceName	` name:` The workspace name used by the `jobStep` object.
workspaceWaitTime	The time this job step had to wait because no workspace was found or available.

Object Type: logEntry Description: **A logEntry is an object containing various types of information available in a log file generated from anywhere in the CloudBees CD system. The Event Log can be configured for auto-deletion. By default, Event Log retention is 30 days. The CloudBees CD server automatically marks old log entries for deletion and the background deleter cleans out old log entries. To change the default settings, go to Administration > Server > Settings and modify the Event log retain time and the Maximum background delete delay fields Note: Setting the "retain" period to "0" disables the automatic deletion mechanism, allowing you to use an external cleanup script to implement a custom cleanup policy.**	
Property Name	Description
category	(currently not used)
container	` string:` Typically, this is the type and name of the workflow or job with a corresponding ID.
containerName	` name:` The name of the container.
containerType	` string:` The type of object the log entry pertains to (for example, procedure, job step, step).
deleted	`byte:` The object was marked for background deletion. Possible values are `"0"` or `"1"` . Default is `"0"` (which means "deleted" is not set).
logEntryId	` id:` The log entry CloudBees CD-generated ID.
message	` string:` The message text will either be informational or display the warning or error message. The message may contain important information about a resource or workspace issue.
principal	` string:` The user or project principal from the session that was active when the event occurred.
severity	` string:` Severity can be either `TRACE` , `DEBUG` , `INFO` , `WARN` , `ERROR`

Object Type: logEntry Description: A logEntry is an object containing various types of information available in a log file generated from anywhere in the CloudBees CD system. The Event Log can be configured for auto-deletion. By default, Event Log retention is 30 days. The CloudBees CD server automatically marks old log entries for deletion and the background deleter cleans out old log entries. To change the default settings, go to Administration > Server > Settings and modify the Event log retain time and the Maximum background delete delay fields **Note:** Setting the "retain" period to "0" disables the automatic deletion mechanism, allowing you to use an external cleanup script to implement a custom cleanup policy.	
subject	` string:` The object associated with the message.
subjectName	` name:` The name of the object associated with the message.
subjectType	` string:` (similar to `container`) Refers to the object the event concerns. This may be the same as the container, or it may be a different object that is related to the event in some manner.
time	` string:` The time the event was logged.

Object Type: procedure Description: A procedure describes a series of actions to be performed on one or more resources. A procedure contains steps and properties and can define formal parameters.	
Property Name	Description
acl	` reference: acl`
createTime	` date:` The time when this object was created.
credentialName	` name:` The name of the credential assigned to this job step.
description	` string:` A user-specified text description of the object.
jobNameTemplate	` name:` The template used to name jobs created from this procedure.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
procedureId	` id:` This is this procedure's ID number.
procedureName	` name:` This is this procedure's name.
projectName	` name:` The name of the project that contains this procedure.
propertySheet	` reference: propertySheet`
resourceName	` name:` The name of the resource or pool this procedure should use to run on.
workspaceName	` name:` The workspace name used by the procedure object.

Object Type: project Description: A project is an object used in CloudBees CD to organize information. A project contains procedures, schedules, credentials, and properties.	
Property Name	Description
acl	` reference: acl`
createTime	` date:` The time when this object was created.
credentialName	` name:` The name of the credential assigned to this project.

Object Type: project Description: **A project is an object used in CloudBees CD to organize information. A project contains procedures, schedules, credentials, and properties.**	
deleted	The object was marked for background deletion. Possible values are "0" or "1". Default is "0" (not set).
description	` string:` A user-specified text description of the object.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
pluginName	` name:` The name of the plugin associated with this project.
projectId	` id:` This is this project's ID.
projectName	` name:` The name of the project.
propertySheet	` reference: propertySheet`
resourceName	` name:` The name of the resource.
workspaceName	` name:` This is the workspace name used by the project.

Object Type: property Description: **A property is a string value with a name. Properties can have arbitrary names and values. You can attach properties to any object in the CloudBees CD system, such as a project, procedure, or job. After a property is created, it is stored in the CloudBees CD database. You can retrieve and/or modify the value later, and you can delete properties you no longer need. Properties provide a flexible and powerful mechanism to manage data about your builds. NOTE: The names " `properties` " and " `project` " are not valid property names.**	
Property Name	Description
createTime	` date:` The time when this object was created.
description	` string:` A user-specified text description of the object.
expandable	` boolean:` If set to true, the property value will undergo string expansion when it is retrieved.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
canonicalPath	` string:` The path that specifies the property object.
propertyId	` id:` This property's ID number.
propertyName	` name:` This property's name.
propertySheetId	` reference: propertySheet—` If the property is a nested property sheet, the `propertySheet` refers to the nested sheet.
value	` string:` The property or actual parameter's value—if this property is a string property.

Object Type: propertySheet Description: A property value can be a simple string or a nested propertySheet containing its own properties. Property sheets can be nested to any depth, which allows you to create hierarchical collections of information. Most objects have an associated property sheet that contains "custom properties" created by user scripts.

Property Name	Description
acl	` reference: acl`
createTime	` date:` The time when this object was created.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
propertySheetId	` id:` This property sheet’s ID.

Object Type: repository Description: A repository is an object that stores artifact versions. This object primarily contains information on how to connect to a particular artifact repository. Similar to steps in a procedure, repository objects are in a user-specified order. When retrieving artifact versions, repositories are queried in this order until one containing the desired artifact version is found.

Property Name	Description
acl	` reference: acl`
createTime	` date:` The time when this object was created.
description	` string:` A user-specified text description for this object.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
propertySheet	` reference: propertySheet`
repositoryDisabled	` boolean:` Determines whether the repository is disabled. Default is "false".
repositoryId	` id:` The CloudBees CD-generated ID number of the repository.
repositoryIndex	` number:` The order of the repository, within a list of repositories.
repositoryName	` name:` The name of the repository.
url	` string:` The server URL is in the form `protocol://host:port/.` Typically, the repository server is configured to listen on port 8200 for `https` requests, so a typical URL looks like `https://host:8200/` .
zoneName	` name : ` ` ` The name of the zone where this repository is or will reside.

Object Type: resource Description: A resource is associated with a server machine available to CloudBees CD for running steps. A resource has a logical name in addition to a host name. A resource can be associated with one or more pools. The resource may be a "proxy target" machine that communicates with the CloudBees CD server though a CloudBees CD agent proxy machine.

Property Name	Description
acl	` reference: acl`
agentState	` string:` Specific information about an agent, including the `state` of the agent. Possible values are: `unknown|alive|down`
artifactCacheDirectory	` string:` The directory on the agent host where retrieved artifacts are stored.
createTime	` date:` The time when this object was created.
description	` string:` A user-specified text description of the object.
exclusiveJobId	` id:` The ID number of the job that owns this resource, which occurs when one of the job's steps requests exclusive use of the resource for the duration of the job.
exclusiveJobName	` name:` The name of the job that owns this resource, which occurs when one of the job's steps requests exclusive use of the resource for the duration of the job.
exclusiveJobStepId	` id:` The ID number of the job step that owns this resource, which occurs when one of the steps request exclusive use of the resource for the duration of the job.
exclusiveJobStepName	` name:` The name of the job step that owns this resource, which occurs when one of the steps request exclusive use of the resource for the duration of the job.
gateways	The gateway names associated with this resource.
hostName	` name:` The computer name or IP address for the machine containing the CloudBees CD agent for this resource.
hostOS	` string:` The full name of the host operating system, plus its version. However, if this host is a proxy, "proxied" appears as the host description/name.
hostPlatform	` string:` Examples for "platform" are: Windows, Linux, HPUX, and so on. However, if this host is a proxy, "proxied" appears as the `hostPlatform` name.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
lastRunTime	` date:` The most recent time a job step ran on the resource.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
pools	` string:` A space-separated list of one or more pool names where this resource is a member. Steps defined to run on a resource pool will run on any available member (resource) in the pool.
port	` number:` The port number for this resource.
propertySheet	` reference: propertySheet`
proxyCustomization	` string:` This property displays the customization of the CloudBees CD proxy agent.

Object Type: resource Description: **A resource is associated with a server machine available to CloudBees CD for running steps. A resource has a logical name in addition to a host name. A resource can be associated with one or more pools. The resource may be a "proxy target" machine that communicates with the CloudBees CD server though a CloudBees CD agent proxy machine.**	
proxyHostName	` name:` The name of the CloudBees CD agent being used to proxy to another agent, the proxy target.
proxyPort	` number:` The port number of the CloudBees CD agent being used to proxy to another agent, the proxy target.
proxyProtocol	` name:` The name of the proxy agent protocol used for communication from the CloudBees CD proxy agent to the proxy target—default is SSH.
repositoryNames	A "new line" separated list of repository names.
resourceDisabled	` boolean:` A Boolean value indicating whether this resource was disabled.
resourceId	` id:` This is this resource’s ID.
resourceName	` name:` The name of the resource or pool.
shell	` name:` The shell used to execute the step’s commands on a resource. If no shell was defined on a step, the shell defined on the resource is used, or if no shell was defined on the resource, a default shell is used. For shells: The script name is inserted into the command at the position of a `"{0}"` marker in the command, or appended as a final argument if no marker is present. A filename suffix adjacent to the marker will be appended to the script name. For more information on shells, see the Shell definition in the Step Help topic.
stepCount	` number:` The current number of executing steps on this resource.
stepLimit	` number:` This property specifies the maximum number of steps that can run on this resource at one time.
trusted	` boolean:` If "true", this agent is trusted.
useSSL	` boolean:` A Boolean value indicating whether this resource uses SSL for communication. Note: Transport Layer Security (TLS) has replaced Secure Sockets Layer version 3.0 (SSLv3) on the CloudBees CD web server and the CloudBees CD server.
workspaceName	` name:` The workspace name used by this resource.
zoneName	` sting: ` The name of the zone where this resource resides.

Object Type: resourcePool Description: **A resource pool is a container for a group of resources.**	
Property Name	Description
acl	` reference: acl`
autoDelete	` boolean: ` If "true", the resource pool is deleted when the last resource is removed or deleted.
createTime	` date:` The time when this object was created.
description	` string:` A user-specified text description of the object.

Object Type: resourcePool Description: A resource pool is a container for a group of resources.	
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
lastResourceUsed	` name:` The name of the most recently used resource from the pool.
modifyTime	` date:` The time when this object was last modified.
orderingFilter	` string:` A Javascript block invoked when scheduling resources for a pool. A Javascript block is not required unless you need to override the default resource ordering behavior.
owner	` name:` The person (username) who created the object.
propertySheet	` reference: propertySheet`
resourcePoolDisabled	` boolean:` A Boolean value indicating whether this resource pool was disabled.
resourcePoolId	` id:` This resource pool's ID number.
resourcePoolName	` name:` The name of the resource pool.

Object Type: resourceUsage Description: Provides usage information for all resources in the system. For any job step running on a resource, there is a resource usage record that contains the ID and name of the job, job step, and resource.	
Property Name	Description
jobId	` id: ` This is this job's ID number, which is a UUID.
jobName	` name: ` The name of the job.
jobStepId	` id: ` The ID number of the job step.
jobStepName	` name: ` The name of the job step.
licenseWaitTime	The time this job step had to wait because no license was found or available.
resourceId	` id:` The CloudBees CD-generated resource ID number.
resourceName	` name: ` The name (or names) of the resource.
resourcePoolId	` id: ` The CloudBees CD-generated resource pool's ID number.
resourcePoolName	` name: ` The name of the resource pool.
resourceUsageId	` id: ` The unique ID of the resource usage record.
resourceWaitTime	The time this job step had to wait because no resource was found or available. This could indicate that eligible resources for the step have reached their step limit, are in-use but the step requires a resource exclusively, or resources are down.
waitReason	Possible values are: `license`, `resource`, or `workspace`.
workspaceWaitTime	The time this job step had to wait because no workspace was found or available.

Object Type: schedule Description: Schedules are used to execute procedures and determine when specific procedures run.

Property Name	Description
acl	` reference: acl`
actualParameter	` reference: propertySheet` An actualParameter is an object that provides the value for a parameter, which is passed to a procedure when it is invoked. Actual parameters can be provided for jobs and nested subprocedures within a job. Actual parameters are different from "formal parameters"- formal parameters define the parameters a procedure is expecting, and actual parameters provide values to use at run-time.
beginDate	` string:` The first day on which the schedule is active, in the form YYYY-MM-dd. For example, "2009-06-25"
createTime	` date:` The time when this object was created.
credentialName	` name:` The name of the credential being used for impersonation when the schedule launches a job.
description	` string:` A user-specified text description of the object.
endDate	` string:` The end of the range of dates the schedule is active, in the form: `YYYY-MM-dd` . The actual end date is not included in the range. For example, `"2009-06-25"`
interval	` string:` A floating point number that represents the time to wait between invocations of the schedule.
intervalUnits	` string:` These are the units to use when interpreting the interval value. The units can be `hours` , `minutes` , `seconds` , or `continuous.`
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
lastRunTime	` date:` The last time a job was launched by a schedule.
misfirePolicy	` string:` The policy the schedule uses when it is unable to launch a job at the scheduled time: `ignore` —wait until the next scheduled time to run ` runOnce` —run immediately, then resume normal scheduling
modifyTime	` date:` The time when this object was last modified.
monthDays	` string:` This property specifies which days of the month this schedule should run—shown as a space-separated list of numbers (1-31).
owner	` name:` The person (username) who created the object.
priority	` string:` Values can be `low` , `normal` (default), `high` , or `highest`
procedureName	` name:` The name of the procedure that contains this `schedule` .
projectName	` name:` The name of the project that contains this schedule.
propertySheet	` reference: propertySheet`
scheduleDisabled	` boolean:` A Boolean value indicating whether this schedule was disabled.

Object Type: schedule Description: Schedules are used to execute procedures and determine when specific procedures run.	
scheduleId	` id:` This schedule's ID number.
scheduleName	` name:` This property displays this schedule's name and differs from the `liveSchedule` property found under a job in that the `liveSchedule` property is written at job creation time only, and not changed, even if the schedule is renamed or deleted.
startTime	` string:` The time of day when this schedule should start running, in the form: `HH:mm:ss`
stopTime	` string:` The time of day when this schedule should stop running, in the form: `HH:mm:ss`
timeZone	` string:` A Java-compatible time zone string.
weekDays	` string:` This property specifies which days of the week this schedule should run. This is a space-separated list of `MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY SATURDAY SUNDAY`. (Note: These are not localized strings.)

Object Type: server Description: This is the CloudBees CD server. There is only one such object in your database. **Read-only server properties** Two read-only server properties are available for use in a step, a script, or an email notifier, which can be used to identify the CloudBees CD server location. `/server/hostName` —returns the CloudBees CD server name `/server/hostIP` —returns the CloudBees CD server IP address These properties are especially useful for building a URL link to a CloudBees CD web page. For example, a link in an email notifier that links back to a Job Details page. A sample link: `https://$/server/hostName/commander/jobDetails.php?jobId=$[jobId]`	
Property Name	Description
acl	` reference: acl`
createTime	` date:` The time when this object was created.
hostIP	` string:` The server's IP address.
hostName	` name:` Generally refers to the computer name or IP address for the machine containing the CloudBees CD server.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
propertySheet	` reference: propertySheet`
serverId	` id:` The server's ID number.
serverName	` string:` The CloudBees CD server's name—defaults to "server".

Object Type: state Description: A state object belongs to a workflow and corresponds to the state definition, including a pointer to the workflow, formal parameters (cloned from definition), expanded actual parameters (form the last time a transition was taken to this state), and the "process" which includes the procedure or workflow (cloned from definition), unexpanded actual parameters (cloned from definition, expanded and passed to the process on entry), and the "last run" entity reference.	
Property Name	Description
acl	` reference: acl`

Object Type: state Description: **A state object belongs to a workflow and corresponds to the state definition, including a pointer to the workflow, formal parameters (cloned from definition), expanded actual parameters (form the last time a transition was taken to this state), and the "process" which includes the procedure or workflow (cloned from definition), unexpanded actual parameters (cloned from definition, expanded and passed to the process on entry), and the "last run" entity reference.**	
active	` boolean:` If "true", the state of the workflow is active.
actualParameter	` reference: propertySheet` An `actualParameter` is an object that provides the value for a parameter, which is passed to a procedure when it is invoked. Actual parameters can be provided for jobs and nested subprocedures within a job. Actual parameters are different from "formal parameters"- formal parameters define the parameters a procedure is expecting, and actual parameters provide values to use at run-time.
createTime	` date:` The time when this object was created.
description	` string:` A user-specified text description of the object.
errorMessage	` string:` When the outcome is `error` , this property displays an error message description.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
projectName	` name:` The name of the project that contains this state.
propertySheet	` reference: propertySheet`
stateId	` id:` The CloudBees CD-generated ID number for the state.
stateName	` name: ` The name of the state.
subjob	` name: ` The name of the subjob.
subparameters	` reference: propertySheet`
subprocedure	` name: ` The name of the nested procedure called when a step runs.
subproject	` string: ` If a subprocedure argument was used, this is the name of the project where that subprocedure is found. By default, the current project is used.
substartingState	` name: ` Name of the starting state for the workflow launched when the state is entered.
subworkflow	` name: ` The name of the subworkflow—a workflow called by another workflow.
subworkflowDefinition	` name: ` The name of the subworkflow definition.
workflow	` name: ` The name of the workflow.

Object Type: stateDefinition Description: **A stateDefinition is a named object that belongs to a workflow definition, which includes specifications for whether or not the state is "startable", formal parameters, notifications, and the process. A process includes a procedure or workflow, the starting state (for workflows only), and actual parameters.**	
Property Name	Description
acl	` reference: acl`

Object Type: stateDefinition Description: **A stateDefinition is a named object that belongs to a workflow definition, which includes specifications for whether or not the state is "startable", formal parameters, notifications, and the process. A process includes a procedure or workflow, the starting state (for workflows only), and actual parameters.**	
actualParameter	` reference: propertySheet` An actualParameter is an object that provides the value for a parameter, which is passed to a procedure when it is invoked. Actual parameters can be provided for jobs and nested subprocedures within a job. Actual parameters are different from "formal parameters"- formal parameters define the parameters a procedure is expecting, and actual parameters provide values to use at run-time.
createTime	` date:` The time when this object was created.
description	` string:` A user-specified text description of the object.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
projectName	` name:` The name of the project that contains this state definition.
propertySheet	` reference: propertySheet`
startable	` boolean:` "True" means this state definition can be the initial state of an instantiated workflow.
stateDefinitionId	` id:` The CloudBees CD-generated ID number for the state definition.
stateDefinitionName	` name:` The name of the state definition.
subprocedure	` name:` The name of the nested procedure called when a step runs.
subproject	` string:` If a subprocedure argument was used, this is the name of the project where that subprocedure is found. By default, the current project is used.
substartingState	` name:` Name of the starting state for the workflow launched when the state is entered.
subworkflowDefinition	` name:` The name of the subworkflow definition.
workflowDefinitionName	` name:` The name of the workflow definition.

Object Type: step Description: **A step includes a command or script executed on a single resource and is the smallest unit of work CloudBees CD understands. Each step specifies a resource on which it should run (either the name of a specific machine or the name of a pool of equivalent machines, in which case CloudBees CD picks a machine from the pool). A step can be given a time limit and if the step does not complete within the specified time, CloudBees CD automatically aborts it. Steps are ordered within a procedure and normally execute sequentially. However, it is possible to mark a consecutive range of steps for parallel execution, so all steps in that range run concurrently. A step in one procedure can invoke another procedure instead of running a command. The invoking step provides parameters needed by the nested procedure.**	
Property Name	Description
acl	` reference: acl`

Object Type: step Description: A step includes a command or script executed on a single resource and is the smallest unit of work CloudBees CD understands. Each step specifies a resource on which it should run (either the name of a specific machine or the name of a pool of equivalent machines, in which case CloudBees CD picks a machine from the pool). A step can be given a time limit and if the step does not complete within the specified time, CloudBees CD automatically aborts it. Steps are ordered within a procedure and normally execute sequentially. However, it is possible to mark a consecutive range of steps for parallel execution, so all steps in that range run concurrently. A step in one procedure can invoke another procedure instead of running a command. The invoking step provides parameters needed by the nested procedure.	
actualParameter	` reference: propertySheet` An `actualParameter` is an object that provides the value for a parameter, which is passed to a procedure when it is invoked. Actual parameters can be provided for jobs and nested subprocedures within a job. Actual parameters are different from "formal parameters"- formal parameters define the parameters a procedure is expecting, and actual parameters provide values to use at run-time.
alwaysRun	` boolean:` If true, this step runs even if the job is aborted before the step completes. Note that a force abort will abort an `alwaysRun` step.
broadcast	` boolean:` If true, this step will run on all resources in a pool.
command	` string:` This property specifies the command this step runs.
condition	` string:` If this element is not present, the event is ALWAYS triggered. If specified, the event is triggered only if the value of the condition argument is TRUE; if not true, a boolean value of "false" or "0" was used. Condition arguments can be a literal, a fixed value string, or a string subject to property expansion.
createTime	` date:` The time when this object was created.
credentialName	` name:` The credential name assigned to this step.
description	` string:` A user-specified text description of the object.
errorHandling	This is the error handling policy copied from the procedure step and indicates how the step responds to errors: `failProcedure` —The current procedure continues, but the overall status is error (default). `abortProcedure` —Aborts the current procedure, but allows already-running steps in the current procedure to complete. `abortProcedureNow` —Aborts the current procedure and terminates running steps in the current procedure. `abortJob` —Aborts the entire job, terminates running steps, but allows alwaysRun steps to run. `abortJobNow` —Aborts the entire job and terminates all running steps, including alwaysRun steps. `ignore` —Continues as if the step succeeded.
exclusive	` boolean:` If true, this step acquires and retains its resource exclusively.
exclusiveMode	` string:` Possible values are: `none` , `job` , `step` , `call`
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
logFileName	` name:` The name of the log file produced by this step, relative to the job’s workspace directory.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
parallel	` boolean:` If true, this step runs in parallel with other adjacent steps also marked to run in parallel.

Object Type: step Description: A step includes a command or script executed on a single resource and is the smallest unit of work CloudBees CD understands. Each step specifies a resource on which it should run (either the name of a specific machine or the name of a pool of equivalent machines, in which case CloudBees CD picks a machine from the pool). A step can be given a time limit and if the step does not complete within the specified time, CloudBees CD automatically aborts it. Steps are ordered within a procedure and normally execute sequentially. However, it is possible to mark a consecutive range of steps for parallel execution, so all steps in that range run concurrently. A step in one procedure can invoke another procedure instead of running a command. The invoking step provides parameters needed by the nested procedure.	
postLogFileName	` name:` This property displays the log file name produced by this step's post processor.
postProcessor	` string:` This property displays the post processor name that is used to gather information about this step. Typically, this is either `postp` or `postp <options>` , or an appropriate command or path including options to a postprocessor available on the CloudBees CD server.
precondition	` string: By default, if a step has no precondition, it will run when scheduled. Set this property to make a step wait until one or more dependent conditions are met. When a job step is eligible to transition from pending to runnable, a precondition is evaluated. A precondition is a fixed text or text embedding property reference that is evaluated to TRUE or FALSE. An empty string, a \"0\" or \"false\" is interpreted as FALSE. Any other result string is interpreted as TRUE. The step will block until the precondition is TRUE.`
procedureName	` name:` The name of the procedure that contains this `step` .
projectName	` name:` The name of the project that contains this step.
propertySheet	` reference: propertySheet`
releaseExclusive	` boolean:` A Boolean value indicating whether this step should release its resource upon completion.
releaseMode	` string:` Possible values are: `none` , `release` , `releaseToJob`
resourceName	` name:` The resource's name this step should use to run on.
shell	` name:` The shell used to execute the step's commands on a resource. The script name is inserted into the command at the position of a " `{0}` " marker in the command, or appended as a final argument if no marker is present. A filename suffix adjacent to the marker will be appended to the script name. For more information on shells, see the Shell definition in the Step Help topic.
stepId	` id:` The step's ID number.
stepName	` name:` The step's name.
subprocedure	` name:` The nested procedure name to call when this step executes.
subproject	` name:` This property is displayed if the subprocedure element is present and specifies the project to which the subprocedure belongs (by default, the current project is used.)
timeLimit	` number:` A floating point number that specifies the maximum length of time this step is allowed to run.
timeLimitUnits	The units to use when interpreting the time limit. Units can be `hours` , `minutes` , or `seconds` .
workingDirectory	` name:` The name of the step's working directory.
workspaceName	` name:` The workspace name used by this step.

Object Type: transition Description: A transition object belongs to a state and corresponds to the transition definition, and includes the target state (unexpanded actual parameters cloned from definition, expanded and passed to the target state on entry), the Javascript condition, and the trigger (`onEnter|onStart|onCompletion|manual`).

Property Name	Description
acl	` reference: acl`
actualParameter	` reference: propertySheet` An actualParameter is an object that provides the value for a parameter that is passed to the target state when the transition is taken.
condition	` string:` If empty or non-zero, the transition is allowed to trigger. If set to "0", the transition is ignored.
createTime	` date:` The time when this object was created.
description	` string:` A user-specified text description of the object.
index	The numeric index of a transition that indicates the transition order in the state definition's transition list.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
projectName	` name:` The name of the project that contains this transition.
propertySheet	` reference: propertySheet`
stateName	` name: ` The name of the state.
targetState	` string:` The target state for the transition definition.
transitionId	` id:` The CloudBees CD-generated ID for the transition.
transitionName	` name:` The name of the transition.
trigger	` string:` Possible values are: `onEnter` – before any actions `onStart` – after a subjob or workflow is created `onCompletion` – after a subjob or workflow completes `manual` – when a user manually requests a transition
workflowName	` name:` The name of the workflow.

Object Type: transitionDefinition Description: A transitionDefinition is a named object that belongs to a state definition, which includes the target state definition (with actual parameters to the target state, the Javascript condition, and the trigger (`onEnter|onStart|onCompletion|manual`).

Property Name	Description
acl	` reference: acl`
actualParameter	` reference: propertySheet` An actualParameter is an object that provides the value for a parameter passed to the target state.
condition	` string:` If empty or non-zero, the transition is allowed to trigger. If set to "0", the transition is ignored.
createTime	` date:` The time when this object was created.

Object Type: transitionDefinition Description: A transitionDefinition is a named object that belongs to a state definition, which includes the target state definition (with actual parameters to the target state, the Javascript condition, and the trigger (`onEnter|onStart|onCompletion|manual`).	
description	` string:` A user-specified text description of the object.
index	The numeric index of a transition that indicates the transition order in the state definition’s transition list.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
projectName	` name:` The name of the project that contains this transition definition.
propertySheet	` reference: propertySheet`
stateDefinitionName	` name:` The name of the state definition.
targetState	` string:` The target state for the transition definition.
transitionDefinitionId	` id:` The CloudBees CD-generated ID for the transition definition.
transitionDefinitionName	` name:` The name of the transition definition.
trigger	` string:` Possible values are: `onEnter` – before any actions `onStart` – after a subjob or workflow is created `onCompletion` – after a subjob or workflow completes `manual` – when a user manually requests a transition
workflowDefinitionName	` name:` The name of the workflow definition.

Object Type: user Description: User refers to any user currently logged into the CloudBees CD system or any user who may use CloudBees CD, but may not be currently logged in.	
Property Name	Description
acl	` reference: acl`
createTime	` date:` The time when this object was created.
email	` name:` Displays this user’s email address.
fullUserName	` name:` The user’s real name.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
modifyTime	` date:` The time when this object was last modified.
mutable	` boolean:` If true, the user is editable within CloudBees CD via the web UI or the `modifyUser` API.
owner	` name:` The person (username) who created the object.
propertySheet	` reference: propertySheet`
providerName	` name:` The name of the directory provider that controls this user.

Object Type: user Description: **User refers to any user currently logged into the CloudBees CD system or any user who may use CloudBees CD, but may not be currently logged in.**	
userId	` id:` A unique ID number for a user object.
userName	` name:` Displays this user's name and also appears in a group object and displays the name of a user who belongs to this group.

Object Type: workflow Description: **A workflow object includes a pointer to the workflow definition, sets of states, the active state, the starting stare, parameters to the initial state, "complete"—a boolean value determining whether or not the workflow is complete, and a log containing the transactions taken and user events.**	
Property Name	Description
acl	` reference: acl`
activeState	` string:` The name of the active state on the workflow object.
actualParameter	` reference: propertySheet` An actualParameter is an object that provides the value for a parameter that is passed to the target state when the transition is taken.
callingState	` string:` The full property path to the state that created this workflow.
callingStateId	` id:` The ID number for the full property path to the state that created this workflow.
completed	` boolean: ` If "true", the workflow is completed and no additional transactions will be evaluated.
createTime	` date:` The time when this object was created.
deleted	` boolean:` The object was marked for background deletion. Possible values are "0" or "1". Default is "0" (not set).
elapsedTime	The time this object ran from start to finish.
finish	` date:` The date/time when this object finished.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
launchedByUser	` string:` The name of the user or project principal that explicitly launched the workflow.
liveWorkflowDefinition	` name:` The current workflow definition name for the workflow definition from which the workflow was created.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
projectName	` name:` The name of the project that contains this workflow.
propertySheet	` reference: propertySheet`
start	` date:` The date/time when this workflow began to run.
startingState	` string:` The initial state of the workflow.
workflowDefinitionName	` name:` The name of the workflow definition.
workflowId	` id:` The CloudBees CD-generated ID for the workflow.

Object Type: workflow Description: **A workflow object includes a pointer to the workflow definition, sets of states, the active state, the starting stare, parameters to the initial state, "complete"—a boolean value determining whether or not the workflow is complete, and a log containing the transactions taken and user events.**	
workflowName	` name:` The name of the workflow.

Object Type: workflowDefinition Description: **A workflowDefinition object contains state and transition definitions, including the workflow name template (analogous to a job name template on a procedure), and ordered sets of state and transition definitions.**	
Property Name	Description
acl	` reference: acl`
createTime	` date:` The time when this object was created.
description	` string:` A user-specified text description of the object.
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
projectName	` name:` The name of the project that contains this workflow definition.
propertySheet	` reference: propertySheet`
workflowDefinitionId	` id:` The CloudBees CD-generated ID for the workflow definition.
workflowDefinitionName	` name:` The name of the workflow definition.
workflowNameTemplate	` string:` Template used to determine the default names for workflows launched from a workflow definition.

Object Type: workspace Description: **CloudBees CD provides each job step with an area on the disk it can use for "working files" and results. This disk area is called a job workspace . (The job workspace defaults to a directory under the workspace directory.) A job step can create whatever files it needs within its workspace, and CloudBees CD automatically places files such as step logs in the workspace. Normally, a single workspace is shared by all steps in a job, but different steps within a job could use different workspaces. The location of the job step workspace is displayed on the Job Details page for the job under "Details" for the step.**	
Property Name	Description
acl	` reference: acl`
agentDrivePath	` name:` The drive-letter based mount point for this workspace on Windows agents.
agentUncPath	` name:` The UNC path for this workspace on Windows agents.
agentUnixPath	` name:` The UNIX path for this workspace on UNIX agents.
createTime	` date:` The time when this object was created.
credentialName	` name:` The name of the credential being used when a resource connects to a network share while setting up the workspace.
description	` string:` A user-specified text description of the object.

Object Type: workspace Description: CloudBees CD provides each job step with an area on the disk it can use for "working files" and results. This disk area is called a job workspace . (The job workspace defaults to a directory under the workspace directory.) A job step can create whatever files it needs within its workspace, and CloudBees CD automatically places files such as step logs in the workspace. Normally, a single workspace is shared by all steps in a job, but different steps within a job could use different workspaces. The location of the job step workspace is displayed on the Job Details page for the job under "Details" for the step.	
lastModifiedBy	` name:` This shows who (generally, a username) last modified this object.
local	` boolean: If "true", this workspace is local.`
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
propertySheet	` reference: propertySheet`
workspaceDisabled	` boolean:` A Boolean value that indicates whether this workspace was disabled.
workspaceId	` id:` The workspace's ID number.
workspaceName	` name:` The workspace's name.
zoneName	`name:` The name of the zone where this workspace resides.

Object Type: zone Description: CloudBees CD provides the ability to create zones—often used to enhance security. * A zone is a way to partition a collection of agents to secure them from use by other groups. For example, you might choose to create a developers zone, a production zone, and a test zone—agents in one zone cannot directly communicate with agents in another zone. * A default zone is created during CloudBees CD installation. The server implicitly belongs to the default zone, which means all agents in this zone can communicate with the server directly (without the use of a gateway). * Each zone can have one or more "gateway agents", which you define. Gateway agents are used for communication from one zone to another zone. For more information, see the Gateways Help topic.	
Property Name	Description
acl	` reference: acl`
createTime	` date:` The time when this object was created.
description	` string:` A user-specified text description for the object.
lastModifiedBy	` name:` This person (generally, a username) last modified this object.
modifyTime	` date:` The time when this object was last modified.
owner	` name:` The person (username) who created the object.
propertySheet	` reference: propertySheet`
resources	` string:` A space-separated list of resources contained in this zone.
zoneId	` id ` `: ` The CloudBees CD-generated ID for this zone.
zoneName	` string:` The name of the zone.

Property error codes

The following list provides error codes that may appear as a value within an `errorCode` property.

ABORTED	AGENT_UNKNOWN_COMMAND

ACCESS_DENIED	AGENT_WRONG_FILE_TYPE
AGENT_BAD_WORKINGDIR	BAD_AGENT_RESPONSE
AGENT_FAILED_CONNECT_BROKER	BAD_LOGFILE_PATH
AGENT_FAILED_CREATE_FILE	CANCELED
AGENT_FAILED_CREATE_WORKSPACE	CIRCULAR_PROCEDURE_REFERENCE
AGENT_FAILED_IMPERSONATION	CORRUPT_CREDENTIAL
AGENT_FAILED_MAP_DRIVE	DUPLICATE_JOB_NAME
AGENT_FAILED_PROXYTARGET_PING	EMPTY_SUBPROCEDURE
AGENT_INCOMPATIBLE_VERSION	FAILED_JOB_RENAME
AGENT_INTERNAL_ERROR	FORMAL_PARAMETER_ERROR
AGENT_INVALID_CWD	INTERNAL_ERROR
AGENT_INVALID_MESSAGE	INVALID_EMAIL_HEADER
AGENT_INVALID_PING_TOKEN	MY_EVENT_EXPANSION_ERROR
AGENT_INVALID_WORKSPACE	NO_EMAIL_HEADERS_SEPARATOR
AGENT_IO_CONNECTION_REFUSED	NONEXISTENT_EMAIL_CONFIG
AGENT_IO_CONNECTION_RESET	NONEXISTENT_PROCEDURE
AGENT_IO_ERROR	NONEXISTENT_RESOURCE
AGENT_IO_NO_ROUTE_TO_HOST	NONEXISTENT_WORKSPACE
AGENT_IO_PORT_UNREACHABLE	NOTIFIER_EXPANSION_ERROR
AGENT_MALFORMED_XML	POST_PROCESSOR_ERROR
AGENT_NONABSOLUTE_PATH	POST_PROCESSOR_OUTPUT_FAILURE
AGENT_NONEXISTENT_DIR	PROPERTY_REFERENCE_ERROR
AGENT_NONEXISTENT_FILE	RESOURCE_WITHOUT_HOSTNAME
AGENT_PING_FAILED	SERVER_SHUTDOWN
AGENT_STREAM_STOPPED	TIMEOUT
AGENT_TIMEOUT	UNKNOWN_HOST
AGENT_UNKNOWN_CMDID	AGENT_UNKNOWN_COMMAND
	AGENT_WRONG_FILE_TYPE
AGENT_UNKNOWN_COMMAND	BAD_AGENT_RESPONSE
AGENT_WRONG_FILE_TYPE	BAD_LOGFILE_PATH
BAD_AGENT_RESPONSE	CANCELED
BAD_LOGFILE_PATH	CIRCULAR_PROCEDURE_REFERENCE
CANCELED	CORRUPT_CREDENTIAL

CIRCULAR_PROCEDURE_REFERENCE	DUPLICATE_JOB_NAME
CORRUPT_CREDENTIAL	EMPTY_SUBPROCEDURE
DUPLICATE_JOB_NAME	FAILED_JOB_RENAME
EMPTY_SUBPROCEDURE	FORMAL_PARAMETER_ERROR
FAILED_JOB_RENAME	INTERNAL_ERROR
FORMAL_PARAMETER_ERROR	INVALID_EMAIL_HEADER
INTERNAL_ERROR	MY_EVENT_EXPANSION_ERROR
INVALID_EMAIL_HEADER	NO_EMAIL_HEADERS_SEPARATOR
MY_EVENT_EXPANSION_ERROR	NONEXISTENT_EMAIL_CONFIG
NO_EMAIL_HEADERS_SEPARATOR	NONEXISTENT_PROCEDURE
NONEXISTENT_EMAIL_CONFIG	NONEXISTENT_RESOURCE
NONEXISTENT_PROCEDURE	NONEXISTENT_WORKSPACE
NONEXISTENT_RESOURCE	NOTIFIER_EXPANSION_ERROR
NONEXISTENT_WORKSPACE	POST_PROCESSOR_ERROR
NOTIFIER_EXPANSION_ERROR	POST_PROCESSOR_OUTPUT_FAILURE
POST_PROCESSOR_ERROR	PROPERTY_REFERENCE_ERROR
POST_PROCESSOR_OUTPUT_FAILURE	RESOURCE_WITHOUT_HOSTNAME
PROPERTY_REFERENCE_ERROR	SERVER_SHUTDOWN
RESOURCE_WITHOUT_HOSTNAME	TIMEOUT
SERVER_SHUTDOWN	UNKNOWN_HOST
TIMEOUT	AGENT_UNKNOWN_COMMAND
UNKNOWN_HOST	

← Preflight Builds

Postprocessors →