Job Properties

Below is a list of job properties for Control-M objects.

Type

Defines the type of job. For example:

```
"CommandJob1": {
    "Type" : "Job:Command",
    "Command" : "echo hello",
    "RunAs" : "user1"
}
```

Many of the other properties that you include in the job's definitions depend on the type of job that you are running. For a list of supported job types and more information about the parameters that you use in each type of job, see Job Types.

Name

Defines the name of a job in an array of jobs. The job array structure is especially useful if you have multiple jobs that bear the same name.

If the job name contains a colon character, escape the colon character with two backslashes.

For more examples of the job array structure, see the examples in Folders and Flows.

The **Name** property is not used in the alternative format, when each job is defined separately under an object that bears the name of the individual job (as shown in the example for the Type property).

```
"Jobs": [
    {
        "Type": "Job:Command",
        "Name":"Job1",
        "Command": "echo I am a Job with name Job1",
        "RunAs": "controlm"
    },
    {
        "Type": "Job:Command",
        "Name":"Job1",
        "Command": "echo I am another Job with the same name",
        "RunAs": "controlm"
    }
]
```

Application, SubApplication

Supplies a common descriptive name to a set of related Jobs, Folders, or SubFolders. The jobs do not necessarily have to run at the same time.

```
"Job1": {
    "Type": "Job:Command",
    "Application": "ApplicationName",
    "SubApplication": "SubApplicationName",
    "Command": "echo I am a Job",
    "RunAs": "controlm"
}
```

Comment

Allows you to write a comment on an object. Comments are not uploaded to Control-M.

```
"JobName": {
    "Type" : "Job:Command",
    "Comment" : "code reviewed by tom",
    "Command" : "echo hello",
    "RunAs" : "user1"
}
```

When

Enables you to define scheduling parameters for Jobs, Folders and SubFolders, including the option of using calendars. If When is used in a Folder or SubFolder, those parameters apply to all Jobs in the Folder or Subfolder.

Several parameters under the When parameter enable you to reference previously defined calendars. For more information about defining calendars (using similar scheduling parameters), see Calendars.

When working in a Control-M Workbench environment, jobs will not wait for time constants and will run in an ad-hoc manner. Once deployed to a Control-M instance, all time constraints will be obeyed.

The following example defines scheduling based on a combination of date and time constraints:

```
"When" : {
    "Schedule":"Never",
                "Months": ["JAN", "OCT", "DEC"],
    "MonthDays":["22","1","11"],
```

```
    "WeekDays":["MON","TUE"],
    "FromTime":"1500",
    "ToTime":"1800"
}
```

The following example defines scheduling based on specific dates that you specify explicitly:

```
"When" : {
    "WeekDays" : [ "NONE" ],
    "Months" : [ "NONE" ],
    "MonthDays" : [ "NONE" ],
    "SpecificDates" : [ "03/01", "03/10" ],
    "FromTime":"1500",
    "ToTime":"1800"
}
```

The following date/time constraints are available for use in the definitions of a Job, Folder, or SubFolder:

Option	Description	Job	Folder	SubFolder
WeekDays	One or more of the following: "SUN","MON","TUE","WED","THU","FRI","SAT" For all days of the week, use "ALL" (the	Yes	Yes	No

	default value). In addition, you can specify a specific day in a specific week of the month using a value with the following format: **DdayWn** For example, **DMONW2** means Monday of the 2nd week of the month. Valid values for n are 1-6.			
Months	One or more of the following: "JAN", "FEB", "MAR", "APR","MAY","JUN", "JUL", "AUG",	Yes	Yes	No

	"SEP", "OCT", "NOV", "DEC" For all months of the year, use "ALL" (the default value).			
MonthDays	One or more days in the range of 1 to 31. For all days of the month, use "ALL" (the default value).	Yes	Yes	No
FromTime	FromTime specifies that a job will not start before this time. Format: HHMM	Yes	Yes	Yes
ToTime	ToTime specifies that a job will not	Yes	Yes	Yes

	start after this time. Format: HHMM To allow the job to be submitted even after its original scheduling date (if it was not submitted on the original date), specify a value of ">".			
Schedule	One of the following options: • "Everyday": scheduling is applied every day, provided that the runnin	Yes	Yes	No

	g criteria are met. • "Never": no scheduling is defined, and the job must be ordered manually.			
SpecificDates	Specific dates for running jobs. For each date, use the format "MM/DD" (enclosed in quotes). Separate multiple dates with commas.	Yes	Yes	No

	The SpecificDates option cannot be used in combination with options WeekDays, Months, or MonthDays. However, since the default for these options is "ALL", you must specify these options with a value of "NONE".			

MonthDay Additional Parameters

You can specify the days of the month the job will run by referencing a predefined calendar set up in Control-M.

```
"When": {
    "MonthDaysCalendar": “Summer2017”
}
```

You can specify the days of the month the job will run by referencing a predefined calendar set up in Control-M, and also using advanced rules specified in the MonthDays parameter:

```
"When": {
    "MonthDaysCalendar": "Summer2017",
    "MonthDays":["1","+2","-3",">4","<5",”D6”,”L7”]
}
```

Where:

MonthDays syntax	Description
1	Day 1 included only if defined in the calendar.
+2	Day 2 included regardless of calendar.
-3	Day 3 excluded regardless of calendar.
>4	Day 4 or next closest calendar working day.
<5	Day 5 or previous closest calendar working day.
D6	The 6th calendar working day.
L7	The 7th from the last calendar working day.
D6PA or D6P*	If MonthDaysCalendar is of type periodical, you can use PA or P* to specify a calendar period name such as A,B,C, or you can use * for *any period*.

-D6 or -L6P*	D and L can also have an exclude specifier.

WeekDays Additional Parameters

You can specify the days of the week that the job will run by referencing a predefined calendar set up in Control-M, and also using advanced rules specified in the WeekDays parameter:

```
"When" : {
    "WeekDaysCalendar" : "Summer2017",
    "WeekDays" : ["SUN","+MON","-TUE",">WED","<THU","D6","L3"]
    }
```

Where:

WeekDays syntax	Description
SUN	Sunday included only if defined in the calendar .
+MON	Monday included regardless of calendar.
-TUE	Tuesday excluded regardless of calendar.
>WED	Wednesday or next closest calendar working day.
<THU	Thursday or previous closest calendar working day.

D6	The 6th calendar working day of the week (where you can specify any number between 0 and 6).
L3	The 3rd from the last calendar working day of the week (where you can specify any number between 0 and 6).
D6PA or D6P*	If WeekDaysCalendar is of type periodical, you can use PA or P* to specify a calendar period name such as A,B,C, or you can use * for any period.
-D6 or -L6P*	D and L can also have an exclude specifier.

Specifying a Period When a Job Can or Cannot Run

In addition to the other date/time elements, you can define a range of dates when a job can or cannot run.

The following example shows a period of time (that is, a range of dates) when the job can run:

```
"When": {
    "StartDate":"20160322",
    "EndDate":"20160325"
    }
```

The following example shows a period of time when the job CANNOT run:

```
"When": {
    "StartDate":"20160322",
    "EndDate":"20160325",
    "ActivePeriod" :false
    }
```

Where:

Parameter	Description
StartDate	Start date for the period when the job can/cannot run.
EndDate	End date for the period when the job can/cannot run.
ActivePeriod	Whether the defined period is a period of activity or inactivity, with the following values: • **true** — Period of activity, when the job CAN run. The default. • **false** — Period of inactivity, when the job CANNOT run.

Relationship Between MonthDays and WeekDays

```
"When" : {
    "Months": ["JAN", "OCT", "DEC"],
    "MonthDays":["22","1","11"],
    “DaysRelation” : “OR”,
    "WeekDays":["MON","TUE"]
    }
```

Parameter	Description
DaysRelation	Logical relationship between MonthDays and WeekDays: • **AND**: The job will run only if the WeekDays and MonthDays contraints are met. • **OR**: The job will run if the WeekDays or MonthDays constraints are met. **Default**: AND

Specifying a Confirmation Calendar

You can specify a predefined Confirmation calendar to use for validation of scheduling dates and you can indicate how to handle jobs that are scheduled for a non-working day in this calendar.

The following example shows how to specify a Confirmation calendar:

```
{
    "Folder1": {
              "Type": "Folder",
              "ControlmServer": "LocalControlM",
```

```
                "Application": "Billing",
                "job1": {
                    "Type": "Job:Command",
                    "Application": "BillingJobs",
                    "RunAs": "control ",
                    "Command": "ls",
                    "When": {
                        "ConfirmationCalendars": {
                            "Calendar": "Holidays",
                            "ExceptionPolicy":
"OrderOnNextConfirmedDay",
                            "ShiftBy": "1"
                        }
                    }
                }
        }
}
```

The following parameters appear under the ConfirmationCalendar object:

Parameter	Description
Calendar	The name of the predefined Confirmation calendar, in which work days are indicated.
ExceptionPolicy	A policy to follow when a job is scheduled on a non-working day. Choose from one of the following options: • DoNotOrder • OrderOnNextConfirmedDay • OrderOnPreviousConfirmedDay

	• OrderAnyway **Default**: DoNotOrder
ShiftBy	The number of additional confirmed days to move the job, beyond the action specified by the exception policy. Specify one of the following: • A positive number of additional days to move forward when the exception policy is OrderOnNextConfirmedDay. • A negative number of additional days to move backward when the exception policy is OrderOnPreviousConfirmedDay. **Default**: 0

Using Rule-based Calendars in Job Scheduling

You can base scheduling on predefined rule-based calendars (RBC). Under the When parameter for a specific Job, Folder, or SubFolder, you can specify the RBCs to include and the RBCs to exclude from scheduling. For more information, see Creating a Rule Based Calendar.

```
"RuleBasedCalJobSimple" : {
  "Type" : "Job:Command",
  "RunAs" : "controlm",
  "Command" : "ls -l",
  "When" : {
    "RuleBasedCalendars":{
```

```
        "Included": ["calendar1"],
        "Excluded": ["calendar2"]
    }
  }
}
```

You can combine the use of rule-based calendars with standard scheduling parameters. In such a case, the Relationship parameter enables you to define the logical relationship (AND/OR) between the criteria defined by the calendars and all other basic scheduling criteria. In other words, you can decide whether either set of criteria, or both sets of criteria, must be satisfied. The default relationship is OR.

```
"RuleBasedCalJobComplex" : {
  "Type" : "Job:Command",
  "RunAs" : "controlm",
  "Command" : "ls -l",
  "When" : {
    "RuleBasedCalendars":{
        "Included": ["weekdays"],
        "Excluded": ["endOfQuarter"],
        "Relationship": "AND"
    },
    "Months":["JAN","FEB","MAR"],
    "WeekDays":["TUE","WED"]
  }
 }
```

Rule-based calendars can also be used with SubFolders. In the following example, two different SubFolders are defined in the parent folder. For one of these SubFolders, RBCs to include and exclude are explicitly specified. These RBCs are either global or from the parent folder. For the other SubFolder, the "USE PARENT"

value is specified instead of the name of an actual RBC, so that all scheduling is inherited from the parent folder (as defined in the next example).

```
{
    "subF1" : {
        "Type" : "SubFolder",
        "When" : {
            "FromTime":"1211",
            "ToTime":"2211",
            "RuleBasedCalendars" : {
                "Included" : [ "calendar1" ],
                "Excluded" : [ "calendar2" ]
            }
        }
    },
    "subF2" : {
        "Type" : "SubFolder",
        "When" : {
            "RuleBasedCalendars" : {
                "Included" : [ "USE PARENT" ]
            }
        }
    }
}
```

For folders, you can define Folder RBCs in addition to using predefined RBCs or other basic scheduling criteria. Folder RBCs are specific to a single folder and are applied to all jobs within the folder. The following example shows how to define RBCs under the folder's When parameter:

```
{
 "RuleBasedTestFolder": {
    "Type": "Folder",
    "ControlmServer": "LocalControlM",
    "When": {
        "RuleBasedCalendars": {
            "endOfQ":{
                "Type": "Calendar:RuleBased",
                "When": {
                    "Months": ["MAR","JUN","SEP","DEC"],
                    "MonthDays": ["29","30","31"]
                }
            },
            "winterWeekendDays": {
                "Type": "Calendar:RuleBased",
                "When": {
                    "Months": ["DEC","JAN","FEB"],
                    "WeekDays": ["SAT","SUN"]
                }
            },
            "Included": ["winterWeekendDays","calendar1"],
            "Excluded": ["endOfQ"]
        },
        "Months": ["APR"],
        "WeekDays": ["FRI","MON"]
    },
    "TestJobInTestFolder": {
        "Type": "Job:Command",
        "RunAs": "controlm",
        "Command": "ls -l",
        "When": {
            "RuleBasedCalendars": {
```

```
                "Included": ["calendar3"],
                "Excluded": ["calendar2"]
            }
        }
    }
  }
}
```

Note the following guidelines:

- Each Folder RBC that you define has its own *When* parameter, with scheduling parameters under it. The scheduling parameters under this When parameter are similar to the scheduling parameters under a *When* parameter of a job or folder, with the following exceptions:
    - The *When*n parameter of an RBC does not support the FromTime and ToTime parameters.
    - For the *When* parameter of an RBC, you can also use the DaysKeepActive parameter.
    - You cannot nest another *RuleBasedCalendars* parameter under the When parameter of an RBC.
- To apply the defined Folder RBCs to the jobs within the folder, you must list each of the defined Folder RBCs in either the Included parameter or the *Excluded* parameter.
- You can list additional predefined Control-M RBCs in the Included and Excluded parameters. In the example above, a predefined RBC named calendar1 is listed along with the Folder RBC winterWeekendDays.
- You can combine the use of RBCs with other scheduling parameters. In the example above, the additional Months and WeekDays settings were added after the *RuleBasedCalendars* definitions. These further scheduling parameters are combined with the RBCs based on a logical AND.
- The folder's scheduling parameters are inherited by each of the jobs in the folder. In addition, for any specific job in the folder, you can add further scheduling parameters or RBCs. In the example above, further RBCs

(calendar3 and calendar2) are associated with a job named TestJobInTestFolder.

Events

Events can be generated by Control-M or can trigger jobs. Events are defined by a name and a date.

Here is a list of the various capabilities of event usages:

- A job can wait for events before running, add events after running, or delete events after running. See WaitForEvents, AddEvents, and DeleteEvents.
- Jobs can add or remove events from Control-M. See Event:Add or Event:Delete.
- You can add or remove events from the Control-M by an API call. See Event Management.

You can set events for a Job, Folder, or SubFolder.

For "Date", you can use the following values:

Date Type	Description
AnyDate	Any scheduled date.
OrderDate	Control-M scheduled date. If you do not specify a Date value, this is the default.
PreviousOrderDate	Previous Control-M scheduled date.
NextOrderDate	Next Control-M scheduled date.
MMDD	Specific date. **Example**: "0511"

WaitForEvents

The following example shows how to define events that the job must wait for before running:

```
"Wait1": {
          "Type": "WaitForEvents",
          "Events": [
              {"Event":"e1"},
              {"Event":"e2"},
              {"Event":"e3", "Date":"AnyDate"}
          ]
}
```

You can specify the logical relationship between events, using logical operators (AND/OR) and parentheses. The default relationship is AND. Note that nesting of parentheses within parentheses is not supported.

```
"Wait2": {
           "Type": "WaitForEvents",
           "Events": [
               "(",
               {"Event":"ev1"},
               "OR",
               {"Event":"ev2"},
               ")",
               "OR",
               "(",
               {"Event":"ev3"},
```

```
            {"Event":"ev4"},
            ")"
        ]
}
```

AddEvents

The following example shows how to specify events for the job to add after running:

```
"add1" :
{
        "Type": "AddEvents",
        "Events": [
            {"Event":"a1"},
            {"Event":"a2"},
            {"Event":"a3", "Date":"1112"}
        ]
}
```

DeleteEvents

The following example shows how to specify events for the job to remove after running:

```
"del1" :
{
        "Type": "DeleteEvents",
        "Events": [
            {"Event":"d1"},
```

```
            {"Event":"d2", "Date":"1111"},
            {"Event":"d3"}
        ]
}
```

If

If statements trigger one or more actions when job-related criteria are fulfilled (for example, the job ended with a specific status or the job failed several times).

The following If statements are available for specifying job-related criteria that must occur for action to be taken:

- If:CompletionStatus
- If:NumberOfReruns
- If:NumberOfFailures
- If:JobNotSubmitted
- If:JobOutputNotFound
- If:NumberOfExecutions
- If:Output
- If:VariableValue

For descriptions of the various actions that can be triggered in response to an If statement that is fulfilled, see If Actions.

If:CompletionStatus

The following example shows an If statement that triggers actions based on job completion status. In this example, if the job runs unsuccessfully, it sends an email and runs another job. You can set this property for a Job, Folder, or SubFolder.

```
"JobName": {
    "Type" : "Job:Command",
    "Command" : "echo hello",
```

```
    "Host" : "myhost.mycomp.com",
    "RunAs" : "user1",
    "ActionIfFailure" : {
        "Type": "If",
        "CompletionStatus": "NOTOK",
        "mailToTeam": {
            "Type": "Action:Mail",
            "Message": "Job %%JOBNAME failed",
            "To": "team@mycomp.com"
        },
        "CorrectiveJob": {
            "Type": "Action:Run",
            "Folder": "FolderName",
            "Job": "JobName"
        }
    }
}
```

If can be triggered based on one of the following CompletionStatus values:

Value	Action
NOTOK	When job fails.
OK	When job completed successfully.
ANY	When the job completed regardless of success or failure.
value	When completion status = *value*. **Example**: *value*=10
Even	When completion status is an even number.

Odd	When completion status is an odd number.
">=5", "<=5", "<5", ">5", "!=5"	When the completion status comparison operator is true.

If:NumberOfReruns

The following example shows how to trigger an action based on number of job reruns. You can set this property for a Job, Folder, or SubFolder.

```
"ActionByNumberOfReruns" : {
    "Type": "If:NumberOfReruns",
    "NumberOfReruns": ">=4",
    "RunJob": {
        "Type": "Action:Run",
        "Folder": "Folder1",
        "Job": "job1"
    }
}
```

Where:

Parameter	Description	Possible Values
NumberOfReruns	Performs an action if the condition of number of job reruns is met.	• "Even" • "Odd" • "!=value" • ">=value" • "<=value" • ">value" • "<value"

		• "value"

If:NumberOfFailures

The following example shows how to trigger an action based on number of job failures. You can set this property for a Job, Folder, or SubFolder.

```
"ActionByNumberOfFailures" : {
    "Type": "If:NumberOfFailures",
    "NumberOfFailures": "1",
    "RunJob": {
        "Type": "Action:Run",
        "Folder": "Folder1",
        "Job": "job1"
    }
}
```

Where:

Parameter	Description	Possible Values
NumberOfFailures	Performs an action if the condition of number of job failures is met.	"value"

If:JobNotSubmitted

The following example shows how to trigger an action based on whether the job is not submitted.

```
"ActionByJobNotSubmitted" : {
    "Type": "If:JobNotSubmitted"
    "RunJob": {
        "Type": "Action:Run",
        "Folder": "Folder1",
        "Job": "job1"
    }
}
```

If:JobOutputNotFound

The following example shows how to trigger an action based on whether the job output is not found.

```
"ActionByOutputNotFound" : {
    "Type": "If:JobOutputNotFound"
    "RunJob": {
        "Type": "Action:Run",
        "Folder": "Folder1",
        "Job": "job1"
    }
}
```

If:NumberOfExecutions

The following example shows how to trigger an action based on number of job executions. You can set this property for a Job, Folder, or SubFolder.

```
"ActionByNumberExecutions" : {
    "Type": "If:NumberOfExecutions",
    "NumberOfExecutions": ">=5"
    "RunJob": {
        "Type": "Action:Run",
        "Folder": "Folder1",
        "Job": "job1"
    }
}
```

Where:

Parameter	Description	Possible Values
NumberOfExecutions	Performs an action if the condition of number of job executions is met.	• "Even" • "Odd" • "!=value" • ">=value" • "<=value" • ">value" • "<value" • "value"

If:Output

The following example shows how to trigger an action based on whether a specified string is found within the job output. You can set this property for a Job, Folder, or SubFolder.

```
"OutputFound":{
    "Type": "If:Output",
    "Code": "myfile.sh",
```

```
    "Statement": "ls -l",
    "RunJob":{
        "Type":"Action:Run",
        "Folder":"Folder1",
        "Job":"job1"
    }
}
```

Where:

Parameter	Description
Code	The string to search for in the output. You can include wildcards in the code — ***** for any number of characters, and **$** or **?** for any single character.
Statement	*(Optional)* Limits the search to a specific statement within the output. If no statement is specified, all statements in the output are searched. You can include wildcards in the statement — ***** for any number of characters, and **$** or **?** for any single character.

If:VariableValue

The following example shows how to trigger an action based on whether a logical condition defined for a variable value is true. This feature requires Control-M/Enterprise Manager version 9.0.21.

```
"NewIPAddressJob" : {
  "Type" : "Job:Command",
  "RunAs" : "emuser",
  "Command" : "echo hello",
  "IfBase:VariableValue" : {
    "Type" : "If:VariableValue",
    "VariableValue" : "1",
    "VariableName": "IPaddress",
    "Operator" : "EndWith",
    "DoNotify_0" : {
      "Type" : "Action:Notify",
      "Message" : "The new IP address ends with 1"
    }
  }
}
```

Where:

Parameter	Description

VariableValue	The value of the variable, either an integer or a string.
	Rules
	• Integer
	  • 1-255 characters for both the Min and Max value field combined if one of the fields contains a variable.
	  • 1-10 characters for each Min and Max value field (11 characters if you are using number signs '**+**' or

	'**-**'), if a variable is not used. • *(z/OS)* 1-64 characters for each Min and Max value field if one of the fields contains a variable. • *(z/OS)* 1-10 characters for each Min and Max value field (11 characters if you are using number signs '**+**' or '**-**') if a variable is not used. • String • 1-255 characters. The resolving string values can contain string lengths of up to 4000 characters. • *(z/OS)* 1-64 characters. The resolving string values can contain string lengths of up to 80 characters. • Special characters: '*****', *(z/OS only)* '**?**'. • String values can contain other variables.
VariableName	The name of the variable. Can be Local, Folder, Global, or Named Pool variable type.

	For a Named Pool variable, specify the Pool name. *(z/OS only)* For a Global variable, you must use the \ prefix , such as \MYVAR.

Operator	Determines the logical condition between VariableName and VariableValue that triggers the action if true.
	Valid operators for integer variables:
	• NotEqualTo
	• GreaterThanOrEqual
	• LessThanOrEqual
	• GreaterThan
	• LessThan
	• EqualTo
	• InRange
	• NotInRange
	To define the range for the InRange and NotInRange operators, add the RangeVariableValue object. For example:
	`"RangeVariableValue" : {`
	`"Min": "1",`
	`"Max" : "10"`
	`},`

	Valid operators for string variables: • Like • NotLike • IsExactly • IsNotExactly • StartsWith • EndWith • Contains • DoesNotContain • IsEmpty • IsNotEmpty

If Actions

The following actions can be triggered in response to an If statement that is fulfilled:

- Action:Mail
- Action:Rerun
- Action:Set
- Action:SetToOK
- Action:SetToNotOK
- Action:StopCyclicRun
- Action:Run
- Action:Notify
- Action:Remedy
- Event:Add
- Event:Delete
- Action:Output

You can set any of these properties for a Job, Folder, or SubFolder.

For descriptions of the various If statements that you can specify to trigger these actions, see If.

Action:Mail

The following example shows an action that sends an e-mail.

```
"mailToTeam": {
    "Type": "Action:Mail",
    "Message": "%%JOBNAME failed",
    "To": "team@mycomp.com"
}
```

The following example shows that you can add optional parameters to the email action.

```
"mailToTeam": {
    "Type": "Action:Mail",
    "Urgency": "Urgent",
    "Subject" : "Completion Email",
    "Message": "%%JOBNAME just completed",
    "To": "team@mycomp.com",
    "CC": "other@mycomp.com",
    "AttachOutput": true
}
```

The following table describes the parameters of the email action:

Parameter	Description
Urgency	Level of urgency of the message — Regular, Urgent, or VeryUrgent. The default is Regular.

Subject	A subject line for the message.
Message	The message text.
To	A list of email recipients to whom the message is directed. Use the semicolon (;) to separate multiple email addresses.
CC	A list of email recipients who receive a copy of the message. Use the semicolon (;) to separate multiple email addresses.
AttachOutput	Whether to include the job output as an email attachment, either true or false. If no value is specified, the default follows the configuration of the Control-M/Server.

Action:Rerun

The following example shows an action that reruns the job.

```
"RerunActionName": {
    "Type": "Action:Rerun"
}
```

Action:Set

The following example shows an action that sets a variable.

```
"SetVariable": {
    "Type": "Action:Set",
    "Variable": "var1",
    "Value": "1"
}
```

Action:SetToOK

The following example shows an action that sets the job status to OK.

```
"SetToOKActionName": {
    "Type": "Action:SetToOK"
}
```

Action:SetToNotOK

The following example shows an action that sets the job status to not OK.

```
"SetToNotOKActionName": {
    "Type": "Action:SetToNotOK"
}
```

Action:StopCyclicRun

The following example shows an action that disables the cyclic attribute of the job.

```
"CyclicRunActionName": {
    "Type": "Action:StopCyclicRun"
}
```

Action:Run

The following example shows an action that runs another job. In this example, the job is set to run as an independent flow.

```
"CorrectiveJob": {
    "Type": "Action:Run",
    "Folder": "FolderName",
    "Job": "JobName",
    "ControlmServer":"RemoteControlM",
    "Date":"010218",
    "Variables":[{"Cvar1":"val1"}, {"Cvar2":"val2"}]
    "RunAsIndependentFlow" : true
}
```

The run action has the following optional properties:

Property	Description
ControlmServer	The Control-M Scheduling Server for the run action. By default, the Control-M Scheduling Server for the run action is the same as defined for the folder.

	You can use this property to specify a different, remote server.
Date	Value to be used as the original scheduling date for the job. The default is OrderDate (that is, the Control-M scheduled date). For any other date, specify the date in the relevant 4-character or 6-character format — **mmdd**, **ddmm**, **yymmdd**, or **yyddmm**, depending on the site standard.
Variables	Variables for the run action
RunAsIndependentFlow	Determines whether the job flow in the folder executes independently from other instances of the same flow. This feature requires Control-M/Enterprise Manager version 9.0.21. The independent flow uses the same events as defined for the folder. **Values**: true | false **Default**: false

Action:Notify

The following example shows an action that sends a notification.

```
"Notifying": {
    "Type": "Action:Notify",
    "Message": "job1 just ran",
    "Destination": "JobLog",
    "Urgency": "VeryUrgent"
}
```

Action:Remedy

The following example shows an action that creates a Remedy ticket.

```
"RemedyTicket": {
    "Type": "Action:Remedy",
    "Summary": "CONTROL-M job %%JOBNAME on node %%NODEID return
code %%COMPSTAT",
    "Urgency": "Low",
    "Message": "CONTROL-M job %%JOBNAME run %%RUNCOUNT ended on
node %%NODEID return code %%COMPSTAT Application: %%APPLIC"
}
```

Event:Add

The following example shows an action that adds an event for the current date.

```
"setEvent1": {
    "Type": "Event:Add",
    "Event": "e1"
}
```

Optional parameters:

```
"setEvent1": {
    "Type": "Event:Add",
    "Event": "e1",
    "Date": "1010"
}
```

Date can have the possible values:

Date Type	Description
AnyDate	Any scheduled date.
NoDate	Not date-specific.
OrderDate	Control-M scheduled date.
PreviousOrderDate	Previous Control-M scheduled date.
NextOrderDate	Next Control-M scheduled date.
MMDD	Specific date. **Example**: "0511"

Event:Delete

The following example shows an action that deletes an event.

```
"unsetEvent2": {
    "Type": "Event:Delete",
    "Event": "e2",
    "Date": "PreviousOrderDate"
}
```

Date can have the following possible values:

- "AnyDate"
- "OrderDate"
- "PreviousOrderDate"
- "NextOrderDate"
- *MMDD*, for example "0511".

Action:Output

The Output action supports the following operations:

-
- Move
- Delete
- Print
- *(z/OS)* ChangeClass.

The following example shows an action that copies the output to the specified destination. In a z/OS job, the additional FromClass parameter is available.

```
"CopyOutput": {
    "Type": "Action:Output",
    "Operation": "Copy",
    "Destination": "/home/copyHere"
}
```

Action:CaptureOutput

The CaptureOutput action enables you to search job output and capture text from the job output into a variable.

The following example shows a job with a CaptureOutput action that captures text when the string "failure" is discovered in the output. In this example, the capture operation starts only 5 lines and 1 word later.

```
"job1": {
    "Type": "Job:Command",
    "RunAs": "controlm",
    "Command": "ls",
    "CaptureOutput": {
        "Type": "Action:CaptureOutput",
        "Capture": "10",
        "Search": "failure",
        "VariableName": "myVar",
        "ForwardBy": {
            "Columns": "1",
            "Delimiter": "Tab",
            "Lines": "5",
            "ColumnsOption" : "words"
        }
    }
}
```

The CaptureOutput action has the following properties:

Property	Description
Capture	The scope of what to capture from job output and store in a variable: • *Number* of words or characters.

	• **UpToEndOfLine** — All the words or characters up to the end of the line. **Default**: 1 (word or character, depending on the value of **ColumnsOption**).
Search	The string to search for in the job output sysout after job processing.
VariableName	Name of a variable in which to store captured text.
ForwardBy	Instructions for skipping ahead (from the point in the output where the search string was detected) before beginning the capture operation.
Columns	The number of words or characters to skip in job output for the capture operation. If Lines=0, the count of columns to skip starts from the end of the search string. Otherwise, the count of columns to skip starts at the beginning of the line. **Default**: 0
Delimiter	The delimiter character to use as a split separator between words in job output for the capture operation, one of the following:

	• WhiteSpace (default) — either space or tab. • Space • Tab • Empty value (default in z/OS job) • *Any other single character that you specify* A delimiter is relevant only if **ColumnsOption** is set to **words**.
Lines	The number of lines to skip from the search string in the job output for the capture operation **Default**: 0
ColumnsOption	The unit to use for column skipping, either **words** (the default) or **characters**

Confirm

Allows you to define a job, folder, or subfolder that requires user confirmation. This can be done by running the run job::confirm command.

```
"JobName": {
    "Type" : "Job:Command",
    "Comment" : "this job needs user confirmation to start execution",
    "Command" : "echo hello",
    "RunAs" : "user1",
```

```
    "Confirm" : true
}
```

CreatedBy

Allows you to specify the Control-M user responsible for job definitions. You can define this property for a Job object or Folder object.

```
"SimpleJob": {
    "Type": "Job:Command",
    "Command": "echo I am a Job.",
    "RunAs": "controlm",
    "CreatedBy":"username"
}
```

The behavior of this property depends on the security policy defined in the Control-M environment, as controlled by the AuthorSecurity parameter in Control-M/Enterprise Manager:

Security Level	Allowed Values	Default Value
Permissive	Any user name.	**ctmdk** (an internal user).
Restrictive	User currently logged in.	User currently logged in.

Critical

Allows you to set a critical job. A critical job is a job that has a higher priority to reserve resources in order to run.

Default: false

```
"Critical": true
```

DaysKeepActive

Allows you to define the number of days to keep jobs that did not run at their scheduled date. You can set this property for a Folder or SubFolder.

Jobs in a folder are kept until the maximum DaysKeepActive value for any of the jobs in the folder has passed. This enables you to retrieve job status of all the jobs in the folder.

```
{
    "DaysKeepActiveFolder": {
        "Type" : "Folder",
        "Defaults": {
            "RunAs":"owner8"
        },
        "keepForeverIfDidNotRun": {
            "Type": "Job:Command",
            "Command":"ls",
            "DaysKeepActive": "Forever"
        },
        "keepForThreeDaysIfDidNotRun": {
            "Type": "Job:Command",
            "Command":"ls",
            "DaysKeepActive": "3"
        }
    }
}
```

Where:

Property	Description
DaysKeepActive	**Valid values**: • 0-98 - keep the job for the specified number of days if it did not execute successfully on its scheduling date. • **Forever** - keep the job indefinitely (or until it is manually deleted), even after it finishes executing successfully. **Default**: 0

Description

Enables you to add a description to a Job, Folder, or SubFolder.

```
{
"DescriptionFolder": {
    "Type" : "Folder",
    "Description":"folder description",
    "SimpleCommandJob": {
        "Type": "Job:Command",
        "Description":"job description",
        "RunAs":"owner8",
        "Command":"ls"
        }
    }
}
```

Documentation

Allows you to add the location and name of a file that contains the documentation for the Job, Folder, or SubFolder.

```
{
    "DocumentationFile":{
        "Path": "C://temp",
        "FileName": "job.txt"
    }
}
```

Allows you to add the URL location of a file that contains the documentation for the Job, Folder, or SubFolder.

```
{
    "DocumentationUrl":{
        "Url": "http://bmc.com"
    }
}
```

EndFolder

Enables you to specify which job is the end point in a folder. After this job completes, no additional jobs in the folder will run, unless they have already started running. The folder is complete once all jobs that are still running complete. Remaining jobs that have not yet started running change to status WAIT SCHEDULE.

Values: true | false

Default: false

```
{
  "EndFolder": {
        "Type": "Folder",
        "EndFolderJob": {
            "Type": "Job:Command",
            "Command": "echo When this job ends, the folder is
complete",
            "RunAs": "controlm",
            "EndFolder": true
        }
    }
}
```

Notification

Allows you to create a notification for certain scenarios before, during and after job execution. You can set notifications for a Job, Folder, or SubFolder.

This example shows a notification sent to JobLog of critical job failure.

```
"NotifyCriticalJobFailure": {
    "Type":"Notify:NotOK",
    "Message": "Critical job failed, details in job output",
    "Urgency": "Urgent",
    "Destination": "JobLog"
}
```

The following parameters are relevant to all notification types.

Parameter	Description

Message	The message to display.
Destination (description for each of the options)	The message is sent to one of the following: • (Default) Alerts - Control-M Alerts window. • JobLog - writes to Control-M job log , to get the job log use the run job:log::get. • Console - operating system console. Or Predefined destination values (for example, FinanceGroup).
Urgency	The message urgency is logged as one of the following: • Regular • Urgent • VeryUrgent **Default**: Regular

Notify:OK

When setting the notification type to OK, if job executed with no errors, the notification "Job run OK" is sent to the JobLog.

```
{
  "Folder1": {
    "Type": "Folder",
```

```
    "job1": {
      "Type": "Job:Command",
      "Command": "ls",
      "RunAs": "user1",
      "Notify1": {
        "Type": "Notify:OK",
        "Message": "Job run OK",
        "Destination": "JobLog"
       }
    }
  }
}
```

Notify:NotOK

When setting the notification type to NotOK, if job executed with errors, the notification "Job run not OK" is sent to the JobLog.

```
{
 "Folder1": {
    "Type": "Folder",
    "job1": {
      "Type": "Job:Command",
      "Command": "ls",
      "RunAs": "user1",
      "Notify1": {
        "Type": "Notify:NotOK",
        "Message": "Job run not OK",
        "Destination": "JobLog"
       }
    }
```

```
  }
}
```

Notify:DoesNotStart

If the job has not started by 15:10, a notification is immediately sent to the email defined in the job with the message that the job has not started.

```
{
"Folder1": {
    "Type": "Folder",
    "job1": {
      "Type": "Job:Command",
      "Command": "ls",
      "RunAs": "user1",
      "Notify3": {
        "Type": "Notify:DoesNotStart",
        "By": "1510",
        "Message": "Job has not started",
        "Destination": "mail",
        "Urgency": "VeryUrgent"
        }
    }
  }
}
```

Parameter	Description
By	**Format**: HHMM Notification sent when job does not start by specified time.

Notify:ExecutionTime

When setting the notification type ExecutionTime to LessThan, if the job completes in less than 3 minutes, the notification "Less than expected" is sent to the Alert destination.

```
{
  "Folder1": {
    "Type": "Folder",
    "job1": {
      "Type": "Job:Command",
      "Command": "ls",
      "RunAs": "user1",
      "Notify1": {
        "Type": "Notify:ExecutionTime",
        "Criteria": "LessThan",
        "Value": "3",
        "Message": "Less than expected"
      }
    }
  }
}
```

Criteria	Value	Description
LessThan	Value in minutes. **Example**: 3	If the job runs less than the defined value, a notification is sent to the defined destination.

GreaterThan	Value in minutes. **Example**: 5	If the job runs longer than the defined value, a notification is sent to the defined destination.
LessThanAverage	Value in minutes or percentage. **Example**: 10%	If the job runs less than the defined value of the average execution time of the job, a notification is sent to the defined destination.
GreaterThanAverage	Value in minutes or percentage. **Example**: 10%	If the job runs longer than the defined value of the average execution time of the job, a notification is sent to the defined destination.

Notify:DoesNotEnd

When setting the notification type DoesNotEnd, if job does not end by the specified time, message is sent to JobLog.

```
{
  "Folder1": {
    "Type": "Folder",
    "job1": {
      "Type": "Job:Command",
      "Command": "ls",
      "RunAs": "user1",
```

```
      "Notify1": {
        "Type": "Notify:DoesNotEnd",
        "By": "1212",
        "Message": "Job does not end",
        "Destination": "JobLog"
      }
    }
  }
}
```

Parameter	Description
By	**Format**: HHMM Notification sent when job does not end by specified time.

Notify:ReRun

When setting the notification type ReRun, when job reruns, message is sent to console.

```
{
  "Folder1": {
    "Type": "Folder",
    "job1": {
      "Type": "Job:Command",
      "Command": "ls",
      "RunAs": "user1",
      "Notify1": {
        "Type": "Notify:ReRun",
```

```
        "Message": "Job5 ReRun",
        "Destination": "Console"
      }
    }
  }
}
```

Notify:LateCyclicSubmit

When setting the notification type **LateCyclicSubmit**, a message is sent if the second or subsequent cyclic submission of a job or folder is late by <x> minutes. This feature requires Control-M/Enterprise Manager version 9.0.21.

```
{
  "Folder1": {
    "Type": "Folder",
    "job1":    {
      "Type": "Job:Command",
      "Command": "ls",
      "RunAs": "user1",
        "Notify:LateCyclicSubmit_0" : {
              "Type" : "Notify:LateCyclicSubmit",
              "By" : "10",
              "Message" : "The cyclic job did not submit on time"
             "Urgency": "Regular",
              "Destination": "JobLog"
      }
    }
  }
}
```

Parameter	Description
By	Determines the number of minutes to wait before sending a message when a cyclic job or folder is not submitted after the specified rerun time. **Format**: MMM **Range**: 000 - 999. **Default**: 0

OverridePath

Enables you to specify an alternative location for a script file without changing the original script file that you specified for a job of type Job Types using the FileName and FilePath properties. The alternative script file in the alternative location must have the same name as the original job script file, in order for it to run instead of the original job script file. This is especially useful for troubleshooting purposes or during development of a new release.

The following example demonstrates the use of the OverridePath property:

```
"JobName": {
    "Type": "Job:Script",
    "FileName": "task1123.sh",
    "FilePath": "/home/user1/scripts",
    "Host": "myhost.mycomp.com",
    "RunAs": "user1",
    "Priority": "XB",
    "OverridePath":"/usr/lib/overridepath" ,
```

```
    "RunAsDummy": true
}
```

PathElement

Enables you to add or update jobs or subfolders as root objects. You define the job or subfolder at the root level, and provide details under the **PathElement** property for the hierarchical positioning of the job or subfolder.

The following example shows a job defined at root level with details of hierarchical positioning under the PathElement property:

```
{
    "SALARIES_JANUARY_2022": {
        "Type": "Job:Database:EmbeddedQuery",
        "PathElement": {
            "Folder": "FINANCE:PAYROLL",
            "Server": "FIN_Server",
            "Library": "CTMP.V900.SCHEDULE"
        },
        "Application": "Azure",
        "Query": "select count(*) from A%%PREV.%%DC_ID.JOB;",
        "ConnectionProfile": "Finance",
        "Host": "sqa"
    }
}
```

The following example shows a subfolder with details of hierarchical positioning under the **PathElement** property:

```
"subF2a" : {
    "Type" : "SubFolder",
    "Application" : "application",
    "PathElement": {
        "Folder": "FINANCE:PAYROLL",
        "Server": "FIN_Server",
        "Library": "CTMP.V900.SCHEDULE"
    }
}
```

Where:

Property	Description
Folder	Specifies a folder/subfolder path where to add the job or subfolder. Use a colon to separate a parent folder from a subfolder.
Server	Specifies the Control-M/Server. Mandatory unless there is only one Control-M/Server in the system.
Library	*(z/OS only)* Specifies the name of the z/OS library.

Priority

Allows you to define the priority that a job has over other jobs. You can set this property for a Job, Folder, or SubFolder.

For Priority values, you can choose between the following options:

Option	Possible values	Default

Small set of textual values (5 values).	• Very High. • High • Medium • Low • Very Low.	Very low
Wide range of 2-character alphanumeric strings.	Values range from AA (lowest) to 99 (highest), including all combinations of two alphanumeric characters: AA-A9...ZA-Z9, 0A-0Z, 01-09, 1A-19...9A-99 Note the following values, which correspond to the values in the other option: • 99 - Very High. • 0A - High. • SA - Medium. • JA - Low. • AA - Very Low.	AA

```
{
    "Folder8": {
        "Type": "Folder",
        "Description": "folder desc",
        "Application": "Billing",
        "SubApplication": "Payable",
```

```
        "SimpleCommandJob": {
            "Type": "Job:Command",
            "Description": "job desc",
            "Application": "BillingJobs",
            "Priority": "High",
            "SubApplication": "PayableJobs",
            "TimeZone": "MST",
            "Host": "agent8",
            "RunAs": "owner8",
            "Command": "ls"
        }
    }
}
```

ReferencePath

Enables you to reference a job or folder from within a subfolder. By doing so, you reuse the referenced job or folder as a dynamic template, instead of specifying full job definitions in the subfolder.

You can further affect the behavior of the specific instance of the job or folder by overriding variables of the referenced job or folder from within the referencing subfolder.

If referencing a job that is stored in a separate folder, use the slash character to define the path. For example: Folder2/Job2

Note that a subfolder that contains the ReferencePath property must not contain any explicit job objects.

This feature requires Control-M/Enterprise Manager version 9.0.21 or higher.

The following example shows a subfolder named JobImpl that references a job named JobTemplate.

```
{
  "Parent": {
    "Type": "Folder",
    "ControlmServer": "LocalControlM",
    "JobImpl": {
      "Type": "SubFolder",
      "ReferencePath": "JobTemplate",
      "RunAs": "centos"
    },
    "JobTemplate": {
      "Type": "Job:Command",
      "Command": "echo %%arg",
      "RunAs": "controlm"
    }
  }
}
```

Rerun

Allows to define cyclic jobs or folders.

The example shows how to define a cyclic job or folder that runs every 2 minutes indefinitely.

```
"Rerun" : {
    "Every": "2"
}
```

The following example shows how to run a job or folder four times where each run starts three days after the previous run ended.

```
"Rerun" : {
    "Every": "3",
    "Units":  "Days",
    "From": "End",
    "Times": "4"
}
```

Property	Description
Every	The frequency at which to run the cyclic job or folder, expressed as a whole number of the specified time unit.
Units	One of the following: "Minutes" "Hours" or "Days". The default is "Minutes".
From	One of the following values: • **Start**: next run time is calculated an N Units from the start time of the current run. • **End**: next run time is calculated as N Units from the end time of current run. • **Target**: run starts every N units . **Default**: "Start"
Times	Number of cycles to run. To run forever, define 0. The default is run forever.

RerunIntervals

Allows to define a set of time intervals for rerun of:

- Jobs
- Folders

The example shows how to define Rerunintervals for the job to run every 12 months, 12 days, 11 hours and 1 minute from the end of the last job run.

```
"RerunIntervals": {
    "Intervals" : ["12m","11h","12d","1m"],
    "From": "End"
}
```

Property	Description
Intervals	The time intervals for job or folder to run again in months, hours, days and minutes.
From	One of the following values: • **Start**: next run time is calculated an N Units from the start time of the current run. • **End**: next run time is calculated as N Units from the end time of current run. • **Target**: run start every N units. **Default**: "Start"

RerunSpecificTimes

Allows to define specific times at which to rerun:

- Jobs
- Folders

The following example shows how to define RerunSpecificTimes to run at those specific times.

```
"RerunSpecificTimes": {
    "At" : ["0900","1100","1230","1710"],
    "Tolerance": "20"
}
```

Property	Description
At	One or more time of day in the format HHMM.
Tolerance	Maximum delay in minutes permitted for a late submission of a specific time.

RerunLimit

Enables you to define non-cyclic jobs or folders, setting a frequency and a limit for the number of times that they can rerun.

The following example shows a non-cyclic job set to run 5 times, once every 3 hours:

```
"jobWithRerunLimit": {
        "Type":"Job:Command",
        "Command":"ls",
        "RunAs":"user1",
        "RerunLimit": {
            "Every": "3",
            "Units": "Hours",
            "Times": "5"
        }
     }
```

The following example shows the same settings (as in the previous example) for a non-cyclic folder:

```
{
    "FolderWithRerunLimit": {
        "Type": "Folder",
        "RerunLimit": {
            "Every": "3",
            "Units": "Hours",
            "Times": "5"
        }
    }
}
```

Property	Description
Every	The frequency at which to run the non-cyclic job or folder, expressed as a whole number of the specified time unit.

	The default (for the combination of Every and Units) is 1 minute.
Units	One of the following: "Minutes" "Hours" or "Days".
Times	Maximum number of times a non-cyclic job or folder can rerun. Default 0, no limit to the number of reruns.

Resources

Resource:Pool

Allows you to set a pool (previously known as quantitative resources or semaphore) on a job, to control access to a resource that is concurrently shared by other jobs. For API command information on resources, see Resource Management.

The following example shows how to add a pool parameter to a job.

```
{
  "FolderRes": {
    "Type": "Folder",
    "job1": {
      "Type": "Job:Command",
      "Command": "ls",
      "RunAs": "pok",
      "Critical": true,
      "sem1": {
        "Type": "Resource:Pool",
```

```
        "Quantity": "3"
      }
    }
  }
}
```

Resource:Lock

Allows you to set a lock (previously known as a *control resource* or *mutex*) as shared or exclusive. If the resource is shared, other jobs can use the resource concurrently. If set to exclusive, the job has to wait until the resource is available before it can run. You can set a lock on a Job, Folder, or SubFolder.

The following example shows how to add a lock parameter to a job.

```
{
  "FolderRes": {
    "Type": "Folder",
    "job1": {
      "Type": "Job:Command",
      "Command": "ls",
      "RunAs": "pok",
      "Critical": true,
      "Lock1": {
        "Type": "Resource:Lock",
        "LockType": "Exclusive"
      }
    }
  }
}
```

RetroactiveJob

Enables you to order a job retroactively to make up for days on which the job did not run. For example, Control-M was down for two days due to a hardware issue; as soon as jobs can run again, this job is scheduled retroactively to run an additional two times, to make up for the days that Control-M was inactive.

Values: true | false

Default: false

```
"RetroactiveJob": {
    "Type": "Job:Command",
    "Command": "echo I am a retroactive order Job, I will be ordered
even after my date",
    "RunAs": "controlm",
    "RetroactiveOrder": true
}
```

RunAs

Enables you to define the OS user responsible for running a job (or all jobs in a folder or subfolder).

By default, jobs are run by the user account where the Control-M/Agent is installed. To specify a diferent user, the agent must be running as root.

```
"Job1": {
    "Type": "Job:Command",
    "Command": "echo I am a Job",
```

```
    "RunAs": "controlm",
}
```

RunAsDummy

Enables you to run a job of any type (other than Dummy) as a dummy job.

This is useful, for example, when a job is temporarily not in use but is still included in a flow. You can temporarily set this job to run as a dummy job, so that there is no impact to the flow.

Values: true | false

Default: false

```
"Job1": {
    "Type": "Job:Command",
    "Command": "echo I am a Job",
    "RunAs": "controlm",
    "RunAsDummy": true
}
```

RunOnAllAgentsInGroup

Allows you to set jobs to run on all agents in the group.

The example shows how to define RunOnAllAgentsInGroup.

Values: true | false

Default: false

```
"jobOnAllAagents": {
    "Type": "Job:Dummy",
```

```
    "RunOnAllAgentsInGroup" : true,
    "Host" : "dummyHost"
}
```

Time Zone

Allows you to set the time zone where the job, folder, or subfolder is scheduled to run. Time zones should be defined at least 48 hours before the intended execution date. We recommend to define the same time zone for all jobs in a folder.

The value for this property can be the 3-letter code that represents the time zone or the ID of the time zone.

For information about configuring the list of possible time zones, see Time Zone.

`"TimeZone":"MST"`

The following table lists the default range of possible time zone values.

Time Zone Code in Control-M	Time Zone ID	Time Offset
HNL	Pacific/Honolulu	GMT-10:00
HAW	HST	GMT-10:00
ANC	AST	GMT-09:00
PST	America/Los_Angeles	GMT-08:00
MST	America/Boise	GMT-07:00
CST	America/Chicago	GMT-06:00
EST	America/New_York	GMT-05:00

ATL	Atlantic/Bermuda	GMT-04:00
RIO	America/Araguaina	GMT-03:00
GMT	——	GMT+00:00
WET	——	GMT+01:00
CET	Europe/Amsterdam	GMT+02:00
EET	Europe/Athens	GMT+03:00
DXB	Asia/Dubai	GMT+04:00
KHI	Asia/Karachi	GMT+05:00
DAC	BST	GMT+06:00
BKK	Asia/Bangkok	GMT+07:00
HKG	Asia/Hong_Kong	GMT+08:00
TYO	Asia/Tokyo	GMT+09:00
TOK	——	GMT+09:00
SYD	Australia/Sydney	GMT+10:00
MEL	Australia/Melbourne	GMT+10:00
NOU	Pacific/Norfolk	GMT+11:00
AKL	Pacific/Auckland	GMT+12:00

Variables

Allows you to use job-level variables with %% notation in job fields. You can set this property for a Job, Folder, or SubFolder.

```
"job1": {
    "Type": "Job:Script",
    "FileName": "scriptname.sh",
    "FilePath":"%%ScriptsPath",
    "RunAs": "em900cob",
    "Arguments":["--date", "%%TodayDate" ],
    "Variables": [
        {"TodayDate": "%%$DATE"},
        {"ScriptsPath": "/home/em900cob"}
    ]
}
```

For specifications of system defined variables such as %%$DATE, see System Variables. For guidelines that pertain to variable names and values, see Variable Names and Values.

For specifications of system defined variables such as %%$DATE, see System Variables. For guidelines that pertain to variable names and values, see Variable Names and Values.

Named pools of variables can share data between jobs using the syntax "\\poolname\variable". Due to JSON character escaping, each backslash in the pool name must be doubled. For example, "\\\\pool1\\date".

```
"job1": {
    "Type": "Job:Dummy",
```

```
    "Variables": [
        {"\\\\pool1\\date": "%%$DATE"}
    ]
},
"job2": {
    "Type": "Job:Script",
    "FileName": "scriptname.sh",
    "FilePath":"/home/user/scripts",
    "RunAs": "em900cob",
    "Arguments":["--date", "%%\\\\pool1\\date" ]
}
```

Jobs in a folder can share variables at the folder level using the syntax "\\variable name" to set and %%variable to use.

```
{
    "Folder1"   : {
        "Type" : "Folder",
        "Variables": [
            {"TodayDate": "%%$DATE"}
        ],
        "job1": {
            "Type": "Job:Dummy",
            "Variables": [
                {"\\\\CompanyName": "compName"}
            ]
        },
        "job2": {
            "Type": "Job:Script",
            "FileName": "scriptname.sh",
            "FilePath":"/home/user/scripts",
```

```
            "RunAs": "em900cob",
            "Arguments":["--date", "%%TodayDate", "--comp",
"%%CompanyName" ]
        }
    }
}
```