Monitor Configuration

Tiffany M Frazier
Steve Dawson
Jonathan Burket
Angelo Sapello
Gregory L Frazier

Monitor Configuration

The MVEE Monitor is configured to launch a set of ATD variants primarily via a configuration file in JSON format, along with a handful of command-line options. The monitor continues to support configuration via command-line arguments alone ("legacy mode"), but its use is deprecated. New users of the MVEE should configure the monitor via configuration files.

A single configuration file is intended to capture a single ATD configuration, although the configuration schema provides for some flexibility in the specification of a set of variants to be launched; that is, it is possible to specify many different individual variant configurations and then group various subsets of them into different variant sets, without having to create a separate configuration file for each such grouping.

Command Line

Note: the following examples assume the monitor executable is in the current executable search path; please adjust accordingly.

Current usage information is available via the following command:

```
monitor -h
```

An ATD can be launched via a command of the following form:

```
monitor [<common options>] [-s <variant set>] [-- <arg> [...]]
  Launches variants from configuration file; mode options:
  -s <variant set>  Identifier of variant set to launch
                    Default is "default", if defined; otherwise, launches all
variants.
  <arg> [...]       Additional arguments passed to the variants after those in the
configuration file.

Common options:
  -f <conf-file>    Path to monitor configuration file.
                    Default: atd.conf
  -V                Verbose logging; may be repeated.
  -C                Allow core dumps (of unlimited size).
  -p                Pin variants to cores (default).
  -P                Do not pin variants to cores.
  -D                Detach variants from monitor; launches variants in separate
sessions and with all monitor file descriptors closed.
  -I                Ignore errors. (Allows monitor to run w/out the MVEE.).
  -M                Monochrome (disable log color).
  -R                Enable ASLR for variants (default).
  -r                Disable ASLR for variants.
```

By default, the monitor looks for a configuration file named 'atd.conf' in the current working directory; otherwise, '-f' can be used to specify the path to a configuration file (an absolute path, or relative to the current working directory).

Legacy Mode

For completeness, legacy mode is invoked via a command of the following form:

```
monitor [<common options>] <N> -v p0=r0 p1=r1 ... -v p0=r0 p1=r1... -A <arg0> [...]
  Launches variants from command line (legacy mode); required arguments:
  <N>                 The number of variants
  -v                  Demark the aliases for variants; must be specified <N> times.
  p=r                 The aliases for each variant, pattern=replacement.
  -A                  Marks the start of the argument list for the variants;
                      <arg0> is the absolute path to the executable (subject to
alias).
```

The common options are the same as above. Note that it is also possible to use a configuration file in legacy mode; in this case, command-line arguments override all variant-specific configuration in the configuration file (see Configuration Schema).

Configuration Files

Due to its length, the complete configuration schema is listed at the end of this page. To help get MVEE users started with configuration files, several examples are provided.

Example 1: Simple, two-variant configuration

The following example specifies what is probably the simplest useful configuration: two variants for an ATD that requires no command-line arguments or environment variables, using defaults for all other settings.

```
{
        "variant" : {
                "global" : {
                        "exec" : {
                                "path" : "/myapp"
                        }
                },
                "specs" : {
                        "variant_1" : {
                                "exec" : {
                                        "alias" : ["/myapp=/path/to/myapp_1"]
                                }
                        },
                        "variant_2" : {
                                "exec" : {
                                        "alias" : ["/myapp=/path/to/myapp_2"]
                                }
                        }
                }
        }
}
```

In this example, 'variant.global.exec.path' specifies an aliased absolute path to the ATD executable, while 'variant.specs.variant_1.exec.alias' and 'variant.specs.variant_2.exec.alias' specify the alias substitutions for the executables for the two variants, so that '/path/to/myapp_1' and '/path/to/myapp_2' will be the paths to the actual executables. For users of legacy mode, the above configuration corresponds to the following command line:

```
monitor 2 -v /myapp=/path/to/myapp_1 -v /myapp=/path/to/myapp_2 -A /myapp
```

Example 2: Adding arguments and environment variables

```
{
        "variant" : {
                "global" : {
                        "exec" : {
                                "path" : "/myapp",
                                "argv" : ["-a", "arg1", "arg2", "arg3"]
                                "env" : ["COMMON_SETTING=1"]
                        }
                },
                "specs" : {
                        "variant_1" : {
                                "exec" : {
                                        "alias" : ["/myapp=/path/to/myapp_1"],
                                        "env" : ["VARIANT_SETTING=1"]
                                }
                        },
                        "variant_2" : {
                                "exec" : {
                                        "alias" : ["/myapp=/path/to/myapp_2"],
                                        "env" : ["VARIANT_SETTING=2"]
                                }
                        }
                }
        }
}
```

This is almost identical to Example 1, except that the variants will be invoked with a set of common command-line arguments and have two variables added to their environments: 'COMMON_SETTING=1' (for both), 'VARIANT_SETTING=1' (for variant_1), and 'VARIANT_SETTING=2' (for variant_2). Variant-specific command-line arguments (overriding any common command-line arguments) are also supported; see Configuration Schema for details.

Example 3: Variant sets, monitor log file, and checkpoint/restore configuration

The following example uses many of the elements in the current configuration schema.

```
{
  "atd" : {
    "name" : "THTTPD",
    "description" : "A lightweight HTTP server"
  },
  "monitor" : {
   "detach_variants" : true,
   "dump_variant_core" : false
  },
  "variant" : {
    "global" : {
      "settings" : {
        "aslr_mode" : 0,
        "disable_syscall_checks" : false,
        "pin_to_core" : true
      },
      "exec" : {
        "path" : "/atd/thttpd",
        "argv" : ["-D", "-p", "8080", "-l", "/atd/thttpd.log"],
        "env" : []
      }
    },
    "specs" : {
      "variant_1" : {
        "exec" : {
          "alias" : ["/atd=/path/to/thttpd/1"],
          "env" : []
        }
      },
      "variant_2" : {
        "exec" : {
          "alias" : ["/atd=/path/to/thttpd/2"],
          "env" : []
        }
      },
      "variant_3" : {
        "exec" : {
          "alias" : ["/atd=/path/to/thttpd/3"],
          "env" : []
        }
      },
      "variant_4" : {
        "exec" : {
          "alias" : ["/atd=/path/to/thttpd/4"],
          "env" : []
        }
      },
      "variant_5" : {
        "exec" : {
          "alias" : ["/atd=/path/to/thttpd/5"],
          "env" : []
        }
      }
    },
    "sets" : {
      "default" : ["variant_1", "variant_2", "variant_3"],
      "small" : ["variant_1", "variant_2"],
      "large" : ["variant_1", "variant_2", "variant_3", "variant_4"],
         "alt_large" : ["variant_1", "variant_3", "variant_4", "variant_5"]
        }
  },
  "cr" : {
    "checkpoint" : {
      "criu" : {
        "exec" : {
          "path" : "/usr/sbin/criu",
          "argv" : ["--tcp-established", "--ext-unix-sk", "-v4",
                    "--raven-files", "/atd/thttpd.log"]
        }
      },
      "dump" : {
        "top_dir" : "/tmp/images/thttpd"
      },
      "monitor" : {
        "exec" : {
          "path" : "tools/cr_dump.py"
        },
        "timeout" : 20
      }
    },
    "restore" : {
      "criu" : {
        "exec" : {
          "path" : "/usr/sbin/criu",
          "argv" : ["--tcp-established", "--ext-unix-sk", "-v4",
                    "--raven-files", "/atd/thttpd.log"]
        }
      },
      "dump" : {
        "top_dir" : "/tmp/images/thttpd"
      },
      "monitor" : {
        "exec" : {
          "path" : "tools/cr_restore.py"
        },
        "timeout" : 20
      },
      "max_attempts" : 1
    }
  }
}
```

In this example, five variants are defined and grouped into four different sets. If a variant set is not selected at launch, the set named "default" would be used; if no set named "default" is configured, all defined variants would be launched.

Configuration Specification

Configuration key and subkeys					Type	Description	Notes
atd					object	General information about the ATD	
	name				string	Name of the ATD	
	description				string	Brief description of the ATD	
monitor					object	Monitor configuration	
	log				object	Monitor log configuration	
		file			object	Configures logging directly to a file	By default, the monitor logs to stderr; users may prefer to capture stderr rather than configure the monitor to log to a file. Note that when logging to a file, some initial log messages may still appear on stderr.
			path		string	Path to log file; may be absolute or relative to the working directory of the running monitor	
			append		boolean	Whether to append to the log file (if existing); default is **true**	

			create_mode		string	String representation of an integer literal specifying the permissions of a newly created log file; default is **"0644"**	This is a string rather than an integer, as JSON does not support octal or hexadecimal integer literals. The string value may specify a decimal, octal (leading "0") or hexadecimal (leading "0x") integer.
		level			string	Valid log levels are (in increasing order of verbosity): "ERROR", "WARNING", "NOTICE", "INFO", "DEBUG", "TRACE"; default is **"IN FO"**; modified by '-V' command-line option	
	detach_variants				boolean	If **true**, variants are launched in separate sessions (via setsid()) and with all monitor file descriptors closed; default is **false**; command-line option '-D' detaches variants, overriding this setting.	
	dump_variant_core				boolean	Controls whether core files are generated when variants terminate abnormally; default is **fal se**; can also be specified via the '-C' command-line option	
	pre_launch_command				string	Specifies a shell command line to be executed (via system()) prior to launching the variants. The command may be a format string using a limited number of conversion specifiers, introduced by '%'. These currently include: %: insert a literal '%' character c: insert the full path to the configuration file used to launch the monitor d: insert the full path to the directory containing the monitor executable m: insert the monitor PID as a decimal string	If the command terminates with other than an exit status of 0, the monitor exits.
	checkpoint				object	Checkpoint and variant-refresh control.	
		interval			integer	Number of seconds between automatic checkpoints. Default is **0** (= no automatic checkpoints).	The interval is measured from the completion of the most recent checkpoint, so checkpoints will occur slightly less frequently than every <interval> seconds.
		refresh_period			integer	Number of checkpoints to perform before invoking variant refresh. Default is **0** (= no variant refresh).	For example, setting the value to 3 would cause refresh to be invoked after every third checkpoint.
		refresh_command			string	A shell command line to be executed to set up variant refresh (e.g., by marshaling checkpointed variant state to a new/different set of variants). The command may be a format string, as in monitor.pre_launch_command. The command is invoked with two additional arguments: (1) the path to the most recent checkpoint directory; (2) the path to the directory where newly marshaled variants for the refresh operation should be stored.	If the command exits with status other than 0, the refresh operation is aborted, but the monitor continues.
		pre_refresh_command			string	A system command to be executed at the beginning of a variant refresh operation, *prior* to taking a checkpoint for the refresh operation.	If the command exits with status other than 0, the refresh operation is aborted, but the monitor continues.

		post_refresh_command			string	A system command to be executed at the end of a variant refresh operation, *after* the refreshed (marshaled) variants have been restored but prior to resuming variant execution.	If the command exits with status other than 0, the refresh operation is aborted and the monitor shuts down.
	divergence				object	Divergence-handling configuration.	
		actions			string array	A *sequence* of actions to be performed by the monitor upon notification of divergence by the MVEE. Action keywords are case insensitive. Available actions are BACKTRACE: print a function-call trace for each variant to the monitor log CHECKPOINT: take a checkpoint (of the divergent variant state) SYSTEM(<cmd>): execute the shell command line <cmd>, which may be a format string, as in monitor.pre_launch_command RECOVER: invoke the monitor's default recovery procedure (rollback to most recent restorable checkpoint) SHUTDOWN: terminate the monitor (and ATD). The default is **["SHUTDOWN"]**.	Unlike triggered or automatic checkpointing (of normal variant state), checkpointing on divergence does not (and cannot) guarantee that variants are in consistent states at the time of checkpointing. If any specified divergence-handling action fails, the monitor shuts down. If, upon completion of the specified action sequence, the ATD is not running, the monitor shuts down.
	subreaper				boolean	Controls whether the monitor configures itself (via prctl) as a child process subreaper, in which the monitor (rather than the system *init* process) becomes the reaper of any ATD descendant processes that are not reaped (wait()-ed for) by their parent processes. When enabled, the monitor will receive SIGCHLD for such processes and will reap them via wait(). Default is **true**.	
variant					object	Variant configuration	
	global				object	Global (default) variant configuration	
		settings			object	Global variant behavior settings	
			aslr_mode		integer	Enables or disables ASLR for all variants; **0 = off (default)**; 1+ = on; command-line option '-R' enables ASLR, overriding this setting.	Originally intended to control the level of ASLR applied (e.g. stack-only, stack + heap), but this is currently not possible on a per-process basis, so it functions only as an on/off control.
			disable_syscall_checks		boolean	Controls whether variants are launched with system call cross-checks disabled; default is **f alse**	
			disable_ta1_xchecks		boolean	Controls whether variants are launched with cross-checks via writes to negative file descriptors disabled; default is **false**	
			pin_to_core		boolean	Controls whether variant processes are pinned to a single CPU core; default is **true**; command line options '-p' and '-P' override this setting.	Applies only to variants, not the monitor process; core assignments are made in a simple, round-robin fashion.

			non_overlapping_mmaps		integer	Controls whether the MVEE will ensure non-overlapping mmap regions for the variants. Default is **0** (disabled). Non-zero value enables.	
		exec			object	Global (default) variant execution settings	
			path		string	Absolute path (subject to alias) of the ATD executable and used as argv[0]; can be overridden by variant-specific configuration	This corresponds to the first '-A' legacy-mode command-line argument.
			argv		string array	Command-line arguments (argv[1]..argv[n]) used by all variants; can be overridden by variant-specific configuration	This corresponds to the remaining '-A' legacy-mode command-line arguments.
			env		string array	Array of environment variables ("<key>=<value>") added to the environment of all variants	The environment of a variant consists of the monitor's environment, plus any global environment variables (variant.global.exec.env), plus any variant-specific environment variables (variant.specs.<variant-id>.exec.env).
	specs				object	Variant-specific configuration	
		<variant-id>			object	*<variant-id>* is a user-chosen string identifier for an individual variant; an arbitrary number of variants can be specified, each with a unique <variant-id>	
			exec		object	Variant-specific execution settings	
				alias	string array	Array of variant aliases (<pattern>=<replacement>); at least one is required for each variant	This corresponds to the '-v' legacy-mode command-line option.
				path	string	Absolute path (subject to alias) of the ATD executable and used as argv[0]; overrides variant.global.exec.path	Has no legacy-mode equivalent. Having variants with different paths (or paths that are different from the global.exec.path, if defined) will almost certainly cause divergence. This setting is designed to be used in combination with disabling syscall checks.
				argv	string array	Variant-specific command-line arguments (argv[1]..argv[n]); overrides variant.global.exec.argv	Has no legacy-mode equivalent.
				env	string array	Array of variant-specific environment variables ("<key>=<value>"); added to variant.global.exec.env	The environment of a variant consists of the monitor's environment, plus any global environment variables (variant.global.exec.env), plus any variant-specific environment variables (variant.specs.<variant-id>.exec.env).
	sets				object	Variant-set definitions	A single configuration file can specify an arbitrary number of variants for an ATD; these can then be grouped into identified sets to enable different variant groupings to be run and tested (via the '-s' command-line option, available as of Version 2).
		<set-id>			string array	*<set-id>* is either "default" or a user-chosen string identifier for a variant set; the value is an array of variant IDs (subkeys of variant.specs); an arbitrary number of variant sets may be defined	A <set-id> of "default" specfies a set of variants to be launched if no other set is selected. If a "default" set is not defined and no other set is selected, all configured variants (subkeys of variant.specs) are launched (in lexicographic order of <variant-id>).
cr					object	Checkpoint/Restore configuration	

	checkpoint				object	Configuration for variant checkpointing	
		criu			object	CRIU configuration	
			exec		object	CRIU execution configuration	
				path	string	Path to CRIU executable; may be absolute, or relative to the monitor's working directory; default: **"/usr/sbin/criu"**	
				argv	string array	CRIU command-line options (for the "dump" operation)	
		dump			object	Dump configuration	
			top_dir		string	Path to top-level directory to contain CRIU dump images	Inside this directory, each checkpoint is generated in a timestamp-named subdirectory, within which the CRIU images of each dumped process are generated in a subdirectory named by the pid of the dumped process.
		monitor			object	Monitor configuration for checkpointing	
			exec		object	Configuration of the external dump tool invoked by the monitor for checkpointing	
				path	string	Path to the external dump tool; may be absolute, or relative to the directory containing the monitor executable; default is **"tools/cr_d ump.py"**	The calling convention for the external dump script is <script-path> [-v] [-c <path-to-monitor-configuration-file>] [-l <log-file>] <pid-to-dump> [...] The exit code of the dump script specifies the number of dump failures (i.e., 0 indicates all pids successfully dumped).
			timeout		integer	Maximum time in seconds to wait for the CRIU dump of all variants to complete; default is **20**	
			pre_checkpoint_command		string	Specifies a system command to be executed just prior to invoking a CRIU checkpoint of the variants (after variant execution has been paused). The command may be a format string, as monitor.pre_launch_command. The path to the checkpoint directory is appended automatically to the command string.	If the command exits with status other than 0, the current checkpoint operation is aborted.
			post_checkpoint_command		string	Specifies a system command to be executed just after successful completion of a CRIU checkpoint of the variants. The command may be a format string, as monitor.pre_launch_command. The path to the checkpoint directory is appended automatically to the command string.	If the command exits with status other than 0, the current checkpoint operation is aborted.
	restore				object	Configuration for variant restoration	
		criu			object	CRIU configuration	
			exec		object	CRIU execution configuration	

				path	string	Path to CRIU executable; may be absolute, or relative to the monitor's working directory; default: **"/usr/sbin/criu"**	
				argv	string array	CRIU command-line options (for the "restore" operation)	
		dump			object	Dump (checkpoint) configuration	
			top_dir		string	Path to top-level directory that contains CRIU dump images for restoration	See also cr.checkpoint.dump.top_dir.
		monitor			object	Monitor configuration for restoring	
			exec		object	Configuration of the external tool invoked by the monitor for restoring	
				path	string	Path to the external restore tool; may be absolute, or relative to the directory containing the monitor executable; default is **"tools/cr_re store.py"**	The calling convention for the external restore script is <script-path> [-v] [-c <path-to-monitor-configuration-file>] [-l <log-file>] <dump-dir> The exit code of the restore script specifies the number of successfully restored variants.
			timeout		integer	Maximum time in seconds to wait for the CRIU restoration of all variants to complete; default is **20**	
			pre_restore_command		string	Specifies a system command to be executed just prior to invoking CRIU to restore a set of variants. The command may be a format string, as in monitor.pre_launch_command. The path to the directory containing the checkpoint to be restored is appended automatically to the command string.	If the command exits with status other than 0, the current restore attempt is aborted.
		max_attempts			integer	Maximum total number of times to attempt restoration for a single restoration request; default is **1**	In the current version, the monitor attempts to restore from the most recent *restorable* checkpoint it can find for the running ATD. A checkpoint is considered restorable if it is complete and has not failed to restore in any previous attempt.

Appendix: JSON Schema

The json schema for the configuration file:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "atd": {
      "type": "object",
      "properties": {
        "name": {
          "type": "string"
        },
        "description": {
          "type": "string"
      },
      "class": {
        "type": "string"
      },
      "server": {
        "type": "string"
      },
      "minimum_memory": {
        "type": "integer"
      }
    },
    "additionalProperties": false
  },
  "monitor": {
    "type": "object",
    "properties": {
      "log": {
        "type": "object",
        "properties": {
          "file": {
            "type": "object",
            "properties": {
              "path": {
                "type": "string"
              },
              "append": {
                "type": "boolean"
              },
              "create_mode": {
                "type": "string"
              }
            },
            "additionalProperties": false
          },
          "level": {
            "enum": ["ERROR", "WARNING", "NOTICE", "INFO", "DEBUG", "TRACE"]
          }
        },
        "additionalProperties": false
      },
      "detach_variants": {
        "type": "boolean"
      },
      "dump_variant_core": {
        "type": "boolean"
      },
      "pre_launch_command": {
        "type": "string"
      },
        "checkpoint": {
          "type": "object",
          "properties": {
            "interval": {
              "type": "integer"
            },
            "refresh_period": {
              "type": "integer"
            },
            "refresh_command": {
              "type": "string"
            },
            "pre_refresh_command": {
              "type": "string"
            },
            "post_refresh_command": {
              "type": "string"
            }
          }
        },
        "divergence": {
          "type": "object",
          "properties": {
            "actions": {
              "type": "array",
              "items": {
                "type": "string"
              }
            }
          }
        },
        "subreaper": {
          "type": "boolean"
        }
      },
      "additionalProperties": false
    },
    "variant": {
      "type": "object",
      "properties": {
        "global": {
          "type": "object",
          "properties": {
            "exec": {
              "type": "object",
              "properties": {
                "argv": {
                  "type": "array",
                  "items": {
```

```
        "type": "string"
      }
    },
    "env": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "path": {
      "type": "string"
    },
    "mvee-only-vars": {
      "type": "array",
      "items": {
        "type": "string"
      }
    }
  },
  "additionalProperties": false
},
"logs": {
  "type": "array",
  "items": {
    "type": "string"
  }
},
"settings": {
  "type": "object",
  "properties": {
    "aslr_mode": {
      "type": "integer"
    },
    "disable_syscall_xcheck": {
      "type": "string"
    },
    "disable_syscall_checks": {
      "type": "boolean"
    },
    "disable_ta1_xchecks": {
      "type": "boolean"
    },
    "pin_to_core": {
      "type": "boolean"
    },
    "non_overlapping_mmaps": {
      "type": "integer"
    },
    "COMMENT": {
            "type": "string"
          }
        },
        "additionalProperties": false
      },
      "marshaling": {
        "type": "object",
        "properties": {
          "launcher": {
            "type": "string"
          },
          "class_path": {
            "oneOf": [
              {
                "type": "array",
                "items": {
                  "type": "string"
                }
              },
              {
                "type": "string"
              }
            ]
          },
          "class_name": {
            "type": "string"
          },
          "process_map": {
            "type": "string"
          }
        }
      }
    }
  },
  "sets": {
    "type": "object",
    "patternProperties": {
      "^.*$": {
        "type": "array",
        "items": {
          "type": "string"
        }
      }
    }
  },
  "specs": {
    "type": "object",
    "patternProperties": {
      "^[^\\.]*$": {
              "type": "object",
              "properties": {
                "exec": {
                  "type": "object",
                  "properties": {
                    "alias": {
                      "type": "array",
                      "items": {
                        "type": "string"
                      },
                      "minItems": 1
                    },
                    "path": {
                      "type": "string"
                    },
                    "env": {
                      "type": "array",
                      "items": {
                        "type": "string"
                      }
                    }
                  },
                  "additionalProperties": false,
                  "required": [
                    "alias"
                  ]
                },
                "marshaling": {
                  "type": "object",
                  "properties": {
                    "initial_dump_dir": {
                      "type": "string"
                    }
                  },
                  "additionalProperties": true
                }
              },
              "additionalProperties": false
            }
          },
          "additionalProperties": false
        }
      },
      "required": [
        "specs"
      ]
    },
    "cr": {
      "type": "object",
"properties": {
  "checkpoint": {
    "type": "object",
    "properties": {
      "criu": {
        "properties": {
          "exec": {
            "type": "object",
            "properties": {
              "path": {
                "type": "string"
              },
              "argv": {
                "type": "array",
                "items": {
                  "type": "string"
                }
              }
            },
            "additionalProperties": false
          }
        },
        "additionalProperties": false
      },
      "dump": {
        "type": "object",
        "properties": {
          "top_dir": {
            "type": "string"
          }
        },
        "additionalProperties": false
      },
      "monitor": {
        "type": "object",
        "properties": {
          "exec": {
            "type": "object",
            "properties": {
              "path": {
                "type": "string"
              }
            },
            "additionalProperties": false
          },
          "timeout": {
            "type": "integer"
          },
          "pre_checkpoint_command": {
```

```
          "type": "string"
        },
        "post_checkpoint_command": {
          "type": "string"
        }
      },
      "additionalProperties": false
    }
  },
  "additionalProperties": false
},
"restore": {
  "type": "object",
  "properties": {
    "criu": {
      "properties": {
        "exec": {
          "type": "object",
          "properties": {
            "path": {
              "type": "string"
            },
            "argv": {
              "type": "array",
              "items": {
                "type": "string"
              }
            }
          },
          "additionalProperties": false
        }
      },
      "additionalProperties": false
    },
    "dump": {
      "type": "object",
      "properties": {
        "top_dir": {
          "type": "string"
        }
      },
      "additionalProperties": false
    },
    "monitor": {
      "type": "object",
      "properties": {
        "exec": {
          "type": "object",
          "properties": {
                "path": {
                  "type": "string"
                }
              },
              "additionalProperties": false
            },
            "timeout": {
              "type": "integer"
            },
            "pre_restore_command": {
              "type": "string"
            }
          },
          "additionalProperties": false
        },
        "max_attempts": {
          "type": "integer"
        }
      },
      "additionalProperties": false
    }
  },
  "additionalProperties": false
}
 },
 "additionalProperties": false
}
```