Defining Provisioning Actions

A provisioning action is a set of actions that execute when you provision a particular component instance. They can be executed on-demand or when a simulated test environment is provisioned in Continuous Testing Platform.

Provisioning actions can be used to automate configurations that are commonly applied in your environments. For example, you might run a script for changing certain environment settings, then update a settings file via FTP. You could use a provisioning action to add or erase data in a Data Repository. Or, if you're not working with message proxies, you might want to automatically go to an application server's web admin console and configuring it to point to a virtual asset instead of a real endpoint (or vice versa).

Provisioning actions are defined in the Virtualize desktop.The structure and management of .pvn files and action suites is similar to that of .pva files and Responder suites (respectively); each project can contain multiple .pvn/.pva files, each of those can contain any number of action suites or Responder suites, data sources can be added and used to parameterize tools, tools can be added as outputs, and so on. This topic focuses on the options and operations that are unique to .pvns and action suites. For details on how to define provisioning actions, see the Virtualize User's Guide.

This topic covers:

> **Warning — ProvisioningAssets project is required**
>
> All provisioning actions must be created within a Virtualize project named ProvisioningAssets. If they are not, CTP will not be able to retrieve the list of provisioning actions or execute any of the provisioning actions.

Specifying Provisioning Actions in .pvn files and Action Suites

Provisioning actions are specified by building a suite of tools that may include:

- Browser Playback: Repeats an action you record via a web interface.
- DB: Sends a query to a database you specify and returns the results from that query in an XML
- form.
- Extension Tool for Custom Scripting: Performs any option you have expressed in a script.
- FTP Client: Puts or gets files on an FTP server.
- External: Invokes a third-party executable.
- Messaging Client: Sends messages over the network using a variety of transports.
- REST Client: Sends messages to HTTP servers using the REST architecture.
- SOAP Client: Sends messages to SOAP servers.

You can perform any set of actions that can be expressed via Virtualize's flexible tool set.

To specify provisioning actions:

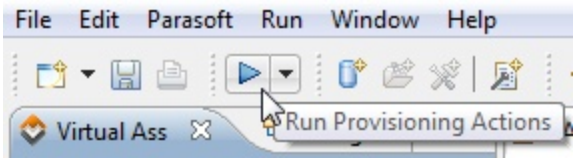
1. If you have not already done so, create a ProvisioningAssets project within your Virtualize workspace. Provisioning actions must be saved inside a project with this exact name.
2. Open the Provisioning Action (.pvn) File wizard in one of the following ways:
    - To add an action suite within a new .pvn, choose **File> New> Provisioning Action (.pvn) file**.
    - To add an action suite within an existing .pvn, right-click the node where you want it added, then choose **Add New> Action Suite**. Next, select the **ProvisioningAssets** project, enter a file name, then click **Next**.
3. In the Action Suite Type selection wizard page, specify what type of action suite you want to create.
    - If you the suite involves playing back actions performed in a web browser, choose **Web> Record Web Scenario**, and complete the wizard as described in Browser Playback.
4. (Optional) If you want to add an additional tool to the action suite, right-click the node where you want it added, choose **Add New> Action**, then select the desired tool from the Add Action wizard.

Executing Provisioning Action Files and Action Suites in Virtualize

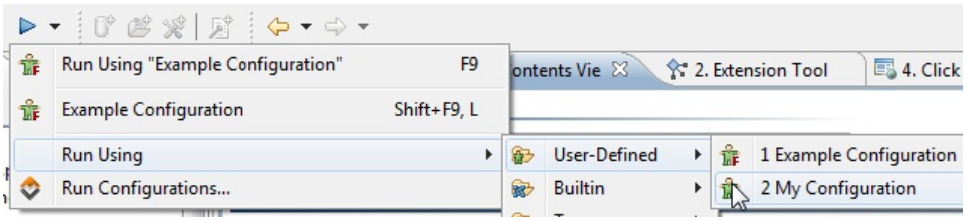
Executing provisioning actions in Virtualize is a fast and easy way to confirm that the actions operate as expected before you integrate them into your environment provisioning on CTP. Doing this testing and debugging directly in the UI where you can create and modify the actions is typically more efficient than doing so externally (e.g., from CTP).

To execute the actions defined in .pvn files and action suites:

1. In the Virtual Asset Explorer, select the node(s) that represent the actions you want performed.
2. Do one of the following:
    - To execute the actions with the default execution settings, click the **Run Provisioning Actions** button.

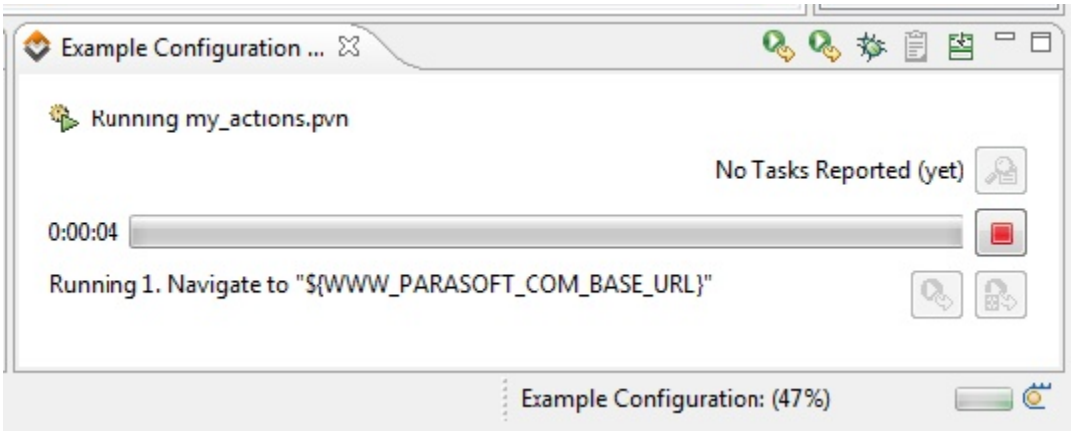


- To execute the actions with customized execution settings, choose the appropriate Run Configuration from the **Run Using** section of the **Run Provisioning Actions** button's pull-down menu. For details on configuring customized execution settings, see Customizing Run Configurations.

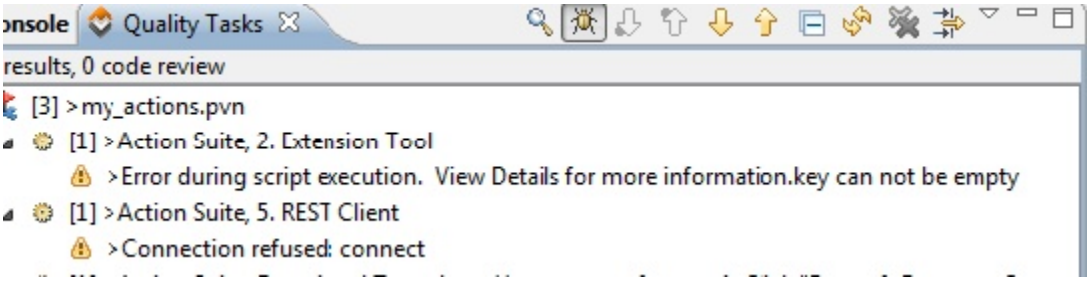


Reviewing Execution Results

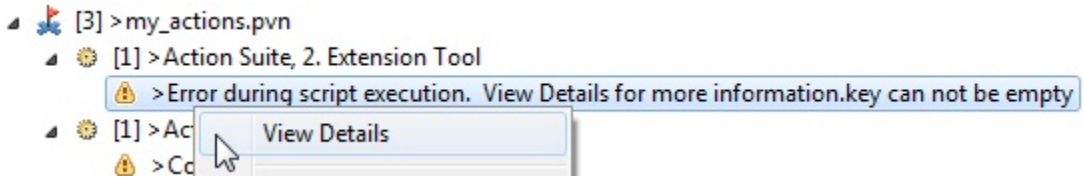
As the actions execute, progress will be reported in the Test Progress view. The label on this view will reflect that Run Configuration that you selected for execution. With the default settings, it will be called "Example Configuration."



After provisioning actions are executed, any problems discovered will be reported in the Quality Tasks view.



For more details about a reported problem, right-click the node, then choose **View Details**.



Customizing Run Configurations

If you want to run provisioning actions with customized execution configurations (for example, you want to use a specific environment or browser other than the one specified at the action set level), you can create a custom Run Configuration that captures these preferences, then execute the actions with that customized configuration.

To create a custom Run Configuration:

1. Open the Run Configurations panel by choosing **Parasoft> Run Configurations**.
2. Click **New**.
3. Configure the **Execution** tab options as desired (see details below).
4. (Optional) Set the Run Configuration as a Favorite Run Configuration by right-clicking it, choosing **Set as Favorite** from the shortcut menu, then specifying what "favorite" position you want it to take (default, 1, 2, or 3). The configuration will then be set as a Favorite Configuration, and the "favorite" icon will be added to that configuration in the Run Configurations tree. The configuration in the default position will be run when you click the **Run Provisioning Actions**.
5. Click **Apply**, then **Close**.

Execution tab Settings

- **Override default environment during execution**: Configures Virtualize to always use the specified environment for actions run with this Run Configuration—regardless of what environment is active in the Virtual Asset Explorer.
- **Use playback engine**: Allows you to override a web scenario's browser engine settings at the time of execution. Selenium is required for Safari and HTML 5 support.
- **Use browser**: Allows you to override a web scenario's browser playback settings at the time of execution. See Browser Playback for details.

Configuring Provisioning Action Files and Action Suites

To configure properties for a .pvn or action suite, double-click its node, then modify options in the configuration panel that opens.

Many of the available options (Notes, Variables, SOAP) are the same as the ones available for Responder suites. These are described in Configuring Responder Suite Properties.

In addition to those common options, you can also configure Action Execution options and Browser Playback options (applicable if you recorded web actions for playback) for each .pvn file and action suite.

Browser Playback options allow you to determine which browsers are used for playback as well as specify authentication settings. See Configuring the Browser Used for details.

Action Execution options allow you to control factors such as whether:

- Actions run sequentially or concurrently.
- Actions can be run independently, or should be run in groups.
- One action depends on the result of another action.
- The suite should loop until a certain condition being met.

Action Execution

You can customize the following execution options for action execution:

- **Execution Mode**: These options determine the concurrency of action execution
  - **Actions run Sequentially**: Choose this option to tell Virtualize to run each action, and child action suite of this action suite, separately from the others.
  - **Actions run Concurrently**: Choose this option to tell Virtualize to run all actions and child action suites of this action suite at the same time. Actions will run simultaneously. Actions will run one at a time.
- **Action Relationship**: These options determine how Virtualize will iterate through the rows of your data sources.
  - **Actions are individually runnable**: (Default) Virtualize iterates through all data source rows for each action. When an individual action executes, it will use every row of the data source before the next action or action suite is executed. When a child action suite is executed, Virtualize will wait for all of its children to finish before the next action or action suite is executed.
  - **Abort Scenario on Fatal Error**: To stop running actions if the previous action resulted in a fatal error, check the **Abort Scenario on Fatal Error** check box.

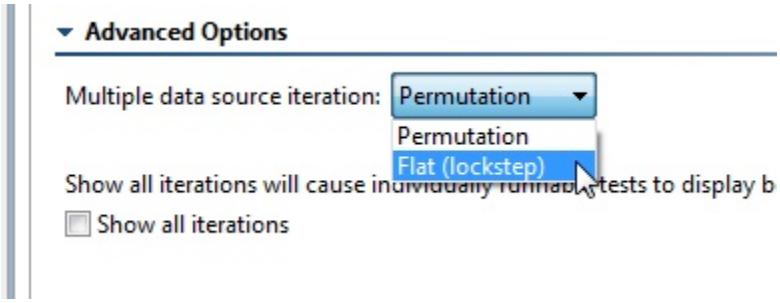
    - This option is only available when both **Actions run Sequentially** is selected and **Actions are individually runnable** is not selected. This case occurs when a set of actions in an action suite are dependent on each other, cannot be run apart from each other, and must run sequentially. If the option is enabled, and if an action within the scenario being run has a fatal error, the rest of the actions in the scenario will not be run. If it is disabled, even if a fatal error occurs, the remaining actions in the scenario will be run.
- **Actions run as group**: (Default for scenarios) Virtualize runs all actions for each row of the data source. In this case, a data source row is chosen, and each action and child action suite is executed for that row. Once all children have executed, a new row is chosen and the process repeats.
- **Actions run all sub-groups as part of this group**: Virtualize treats all actions contained in this action suite like direct children of this action suite. Virtualize will then iterate through them as a group
- **Advanced Options> Multiple data source iteration**: These options determine how Virtualize will iterate when multiple data sources are used within a single action suite whose actions are NOT individually runnable. If all data sources do not have the same number of rows, iteration will stop at the last row of the smallest data source. Available options ar
    - **Permutation**: Action execution will permute over rows in different data sources. For example, iteration of two data sources (A & B)— each with 3 rows— would look like the following:

Data Source A (3 rows)	Data Source B (3 rows)
row 1	row 1
row 2	row 1
row 3	row 1
row 1	row 2
row 2	row 2
row 3	row 2
row 1	row 3
row 2	row 3
row 3	row 3

    - **Flat (lockstep)**: Action execution will iterate over rows in different data sources together at the same time. For example, iteration of two data sources (A & B)— each with 3 rows— would look like the following:

Data Source A (3 rows)	Data Source B (3 rows)
row 1	row 1
row 2	row 2
row 3	row 3

    Note that this option is available only in the top-level (parent) suite in your project .pvn file. You need to expand the **Advanced Options** section to see it.



- **Advanced Options> Show all iterations**: This option determines if all data source iterations (including those for individually-runnable actions) are counted and shown. It is enabled by default. When this option is disabled, Virtualize does not show all of the data source iterations for individually-runnable actions. In other words, if an action was parameterized on a data source with 50 rows, Virtualize would report that as a single action run. As a result, if there are failures on multiple data source rows, there could be more failures than action runs.

Action Flow Logic

Virtualize allows you to create actions that are dependent on the success or failure of previous actions, set-up actions, or tear-down actions. This helps you create an efficient workflow within the action suite. In addition, you can also influence action suite logic by creating while loops and if/else statements that depend on the value of a variable.

Options can be set at the action suite level (options that apply to all actions in the action suite), or for specific actions.

Action Flow Logic Options

In many cases, you may want to have Virtualize repeatedly perform a certain action until a certain condition is met. Action suite flow logic allows you to configure this.

Understanding the Options

To help you automate such scenarios, Virtualize allows you to choose between two main action flow types:

- **While variable**: Repeatedly perform a certain action until a variable condition is met. *This requires variables to be set*.
- **While pass/fail**: Repeatedly perform a certain action until a pass/fail condition is met (e.g., one of the actions in the action suite either passes or succeeds).

Setting the Options

To configure action flow logic options that apply across the action suite:

1. Open the **Execution Options> Action Flow Logic** tab, then select the top-level node.
2. Select the desired flow type.
    - You can choose from **while variable** or **while pass/fail** loop flow (see above for an explanation of the different types) or **none** (if you do not want execution flow to depend on a condition being met).
3. (Optional) Customize the **Maximum number of loops** setting, which determines the maximum number of loops to run if the specified condition is never met.
4. If you chose **while/pass fail** flow specify the loop conditions by going to **Loop until one of the action(s)** and choose **succeeds** or **fails**—depending on which outcome you want to occur before the action suite proceeds.
5. If you chose **while variable** flow, set the while and do conditions as follows:

    - **while**: Select the desired variable from the drop-down list. The items in this list depend on the variables you added to the **Variables** tab.
        - If the variable you select was defined as a boolean value, you will be able to select from either **true** or **false** radio buttons.
        - If the variable you select was defined as an integer, a second drop-down menu displays with **==** (equals), **!=** (not equal), **<** (less than), **>** (greater than), **<=** (less than or equal to), **>=** (greater than or equal to). In addition, a text field is available to enter an integer.
    - **do**: Allows you to determine the action for the variable in the while loop. The following options are available:
        - **Nothing**: If the variable condition is met, do nothing.
        - **Increment**: (For integer values only) If the variable condition is met, increment the variable.
        - **Decrement**: (For integer values only) If the variable condition is met, decrement the variable
        - **Negate**: (For boolean values only) If the variable condition is met, negate the variable.

Action-Specific Logic Options

- **Action Result Dependency**: If the current (selected) action should run only if another action succeeds, fails, or is skipped, then specify the name of that dependent action here. For example, if Action 4 depends on the results of Action 1, select Action 4 in the left panel, then choose Action 1 from the drop-down menu. Then, specify the condition under which the current action should run. Options are:
    - **Success**: Select if the subsequent action should be run according to the success of the action selected in the **Action** drop-down menu. If the action selected in the **Action** drop-down menu does not succeed, the subsequent action will not run.
    - **Failure**: Select if the subsequent action should be run according to the failure of the action selected in the **Action** drop-down menu. If the action selected in the **Action** drop-down menu does not fail, the subsequent action will not run.
    - **Skipped**: Select if the subsequent action should be run if the action selected in the **Action** drop-down menu was skipped. If the action selected in the **Action** drop-down menu is not skipped, the subsequent action will not run.
- **Variable Condition**: Allows you to determine whether or not an action is run depending on variables added to the **Variable** table (for more information on adding variables). If no variables were added, then the Variable Condition options are not available. The following options are available if variables were defined:
    - **Variable Condition drop-down**: Select the desired variable from the drop-down list. The items in this list depend on the variables you added to the **Variable** table.
        - If the variable you select was defined as an integer, a second drop-down menu displays with **==** (equals), **!=** (not equal), **<** (less than), **>** (greater than), **<=** (less than or equal to), **>=** (greater than or equal to). In addition, a text field is available to enter an integer. For example:

If x != 13 (x does not equal 13), the action will run, however, if x does equal 13, the action will not be run.

        - If the variable you select was defined as a boolean value, you will be able to select from either **true** or **false** radio buttons. For example:

If variable x1 is false, the action will run, however, if x1 is true, the action will not be run.

- **Delay in milliseconds**: Lets you set a delay before and/or after action execution.