_Configuring_for_Different_Environments

This topic describes how to configure and work with different environments. In this section:

Understanding Environments

Environments are collections of variables that can be referenced within fields of your responder or action suite.You can use environment variables to specify endpoints, database table names, connection properties, such as login credentials, etc. The names of variables will be substituted with the assigned values in the active environment. By switching which environment is the "active" environment, you can dynamically switch the environment-specific values at runtime.

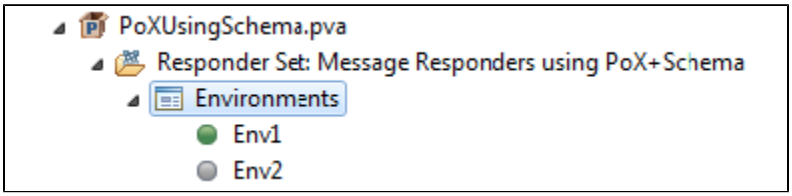
Environments can also be used to switch virtual asset modes. For example, assume you configured a responder to forward traffic to an external endpoint. By using an environment variable for the endpoint (instead of a fixed value), you can easily redirect message forwarding to different endpoints. This allows . pva files to act like proxies; one environment can point to a real asset while another points to a virtual asset.

Environments are automatically defined when you generate a Parasoft asset from a definition, such as WSDL, but you can also manually define environments as described below.

Manually Defining an Environment

Creating and switching environments is done through the **Environments** branch of the

Responder suite's Virtual Asset Explorer node.



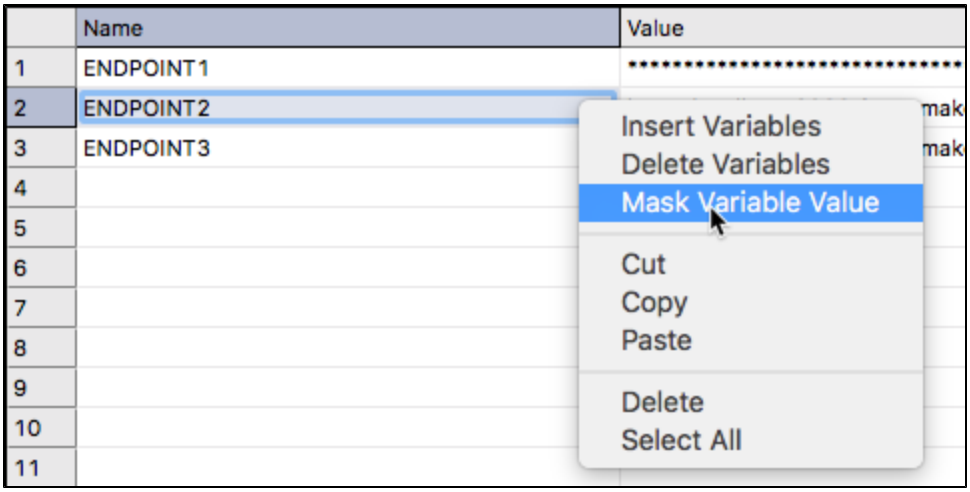
The Environments branch is created by default when a new Responder suite is created.

To add a new environment:

1. Right-click the **Environments** node and choose **NewEnvironment**.
2. Rename the environment (optional) and click on a field in the table to define environment variables and values.

Masking a Variable Value

You can right-click on a variable and choose **Mask Variable Value** to hide the value in the interface.



Using Environment Variables in Tests and Tools

Environment variables can be accessed in test or tool configuration fields using a special syntax. To reference a variable, enclose the variable name in the following character sequence: `${env_name}`.

For example, if you have a variable named HOST, you would reference the variable in a field by typing: `${HOST}`. Variable references may appear anywhere within a field.You can access Environment Variable values from a Extension Tool/Script through the Extensibility API. ExtensionToolContext now has a method called "`getEnvironmentVariableValue(String)`" which will lookup and return the current value of an Environment variable. This will allow you to use the value within your scripts.

> **Note**
>
> If your test case or tool requires the character sequence `${}`, you can escape the sequence by adding a backslash. For example, if SOAtest or Virtualize encounters the value "`\${HOST}`" it will use the value "`${HOST}`" and will not try to resolve the variable. Also note that environment variable names are case sensitive.

Changing Environments from the GUI

To change what environment is active:

- Right-click the node representing the environment you want to make active, then choose **Set as Active Environment**.

Exporting, Importing, and Referencing Environments

You may find that many configuration settings, such as server names and ports, will be common across multiple projects. Rather than duplicating these settings, you can export environment settings to an external file and import or reference the values in other projects.

Exporting Environments

To export an environment:

1. Right-click the node representing the environment you want to export, then choose **Export Environment**.
2. In the file chooser that opens, specify a location for the exported environments file.

The environments configuration will be written in an XML-based text file. If one Environment is selected, a `*.env` file will be created, containing a single environment. If multiple environments are selected, a `*.envs`, or Environment Set, file will be created containing all of the selected environments.

Importing Environments

When you import environments, you are bringing a copy of the values from the external environment file into your project. Further modification to the XML file will not be reflected in your project.

To import an environment:

1. Right-click the **Environments** node, then choose **Import Environment**.
2. In the file chooser that opens, specify the location of the environments file that you want to import.

Referencing Environments

Referencing environments is the most efficient way to share a single environment configuration across multiple projects. Using environment references, you can easily modify the configurations of multiple projects from a single location.

To reference an environment:

1. Right-click the **Environments** node, then choose **Reference Environment**.
2. In the configuration panel that opens on the right, specify the location of the environments file that you want to reference.

Note that when an environment configuration is referenced, you cannot edit the environment variables in the environment directly. However, your project will always use the values reflected in the referenced `*.env` file. Modifying the `*.env` file will propagate changes to all projects that reference it.