# API Naming Convention

## Vendor Agnostic API Naming Convention

### Configuration Parameters Naming Convention

Configuration parameters are defined inside environment property files or added to the runtime configuration of the application. In instances where these parameters are defined inside property files, these parameters are commonly entered as key-value pairs.

For example, for VA vet360 API, there is a need to store a configuration parameter for the relational database. The database configuration includes its host, port, username, password, and instance. All of these variables are considered configuration parameters that are managed either in one of the environment property files or as runtime arguments when launching an application (e.g., Linux runtime arguments, and Java JVM arguments).

The naming conventions for the configuration parameters should reflect the following:

* Application name
  + This will be a short name for the application that is associated with this property. For example, “vet360” for the Vet360 application.
* Platform type:
  + “db” for database
  + “jms” for JMS queue
  + “mule” for Mulesoft platform parameters
  + “web” for webserver
  + etc
* Platform name (if applicable):
  + “sqlserver” for SQL Server database
  + “mongo” for MongoDB
  + “oracle” for Oracle database
  + “apache” for Apache web server
  + “tomcat” for Tomcat instance
  + etc
* Parameter type:
  + “env” for environment instance
  + “host” for host name
  + “port” for port number
  + “username” for username
  + “password” for password
  + “instance” for the database instance name
  + etc

An example of a full name of a configuration parameter:

* vetapi.mule.host
* vetapi.mule.audituser
* vet360.db.oracle.host
* vet360.db.oracle.port
* vet360.web.iis.port

**Important**: Since configuration parameters are managed in Environment Property files, which already are named by the specific environment (“dev.configuration”), there is no need to add an environment name into the name of the configuration parameter. In fact, no environment-specific indicators, keywords, or flags should be part of the name of any configuration parameter. Configuration parameters are reused across environments and, as such, are not tied to a specific environment.

Once a configuration parameter is placed in an environment property file tied to a specific environment, the value of that parameter becomes specific to that environment.

For example, **mule.env**=dev inside the dev.configuration property file. The same **mule.env**=qa is now inside the qa.configuration file.

### Configuration Properties Naming Convention Guidelines

Here are some general guidelines and rules when choosing naming convention for configuration parameters:

* Use short names rather than lengthy names
* Environment specific keywords (e.g. dev, or prod) should never be in the parameter name.
* Names should be descriptive enough to clearly indicate which platform or system a parameter is associated with, the name of the platform, and the type of the parameter
* All parameters across the VA should be consistently named based on a single canonical model. Example model for the parameter names can contain these globally defined parameter types (this is just an example of what the model can contain):
  + host
  + port
  + instance
  + env
  + username
  + password
  + path

## MuleSoft specific API Naming Convention

### Configuration Parameter Naming Convention

#### Use Case

Can create a configuration parameter for each target Mulesoft environment and deploy the same artifact across multiple environments (e.g., dev, qa, production), without modifying the keys of the property parameters. Standardize on the configuration parameter naming convention.

#### Global Properites

Mule 4 has the ability to define global properties. A developer can use the <global-property> element to set a placeholder value from within the Mule configuration, such as from within another Mule configuration file. Example of a definition of a global property is shown below:



The properties are managed in mule-app.properties file in Mule 3 and Mule 4, but with the new 4.x release Mulesoft has also provide an ability to manage global properties inside a YAML file. The format of a YAML file allows to structure all properties in a nested tree. For example, instead of the above “smtp.host”, we can now store the same property inside a YAML file like this:

smtp:

host: “{host name}”,

subject: “{subject name”

With that said, the naming conventions for the global configuration parameters should still follow the naming conventions defined in the previous, Technology-Agnostic, section.

Mulesoft has a number of reserved global configuration parameters, which all start with “mule.” in the name of the parameter. For example, “mule.env” is reserved to the name of the Mule environment. In addition, there are other reserved Mule properties that are tied to the application deployed. For example, “app.name” is the name of the application running within the Mule runtime.

In a Mule process flow, a developer can reference these parameters like this: ${mule.env}, by adding a dollar sign “$” and wrapping the parameter name with the curly braces. In Mule Expression Language (MEL) used inside an expression such as inside a DataWeave Transformations control, accessing properties is done with a reserved “#p[]”, where you would place the name of the property inside the square brackets. For example, “#p[‘vet360.db.oracle.host’]”.