Understanding host and check datas and How They Work (NEW)

Host and check datas

One of the main features that makes Shinken Enterprise so flexible is the ability to use datas in command defintions. Datas allow you to reference information from hosts, services, and other sources in your commands.

data Substitution - How datas Work

Before Shinken Enterprise executes a command, it will replace any datas it finds in the command definition with their corresponding values. This data substitution occurs for all types of commands that Shinken Enterprise executes - host and service checks, notifications, event handlers, etc.

Certain datas may themselves contain other datas. These include the "$HOSTNOTES$", "$HOSTNOTESURL$", "$HOSTACTIONURL$", "$SERVICENOTES$", "$SERVICENOTESURL$", and "$SERVICEACTIONURL$" datas.

Tip: If, you need to have the '$' character in one of your command (and not referring to a data), please put "$$" instead. Shinken Enterprise will replace it well

Example 1: Host Address data

When you use host and service datas in command definitions, they refer to values for the host or service for which the command is being run. Let's try an example. Assuming we are using a host definition and a check_ping command defined like this:

- Host:

Property	Value
Name	linuxbox
address	192.168.1.2
check_command	check_ping

- Command

Property	Value
Name	check_ping
command_line	/var/lib/Shinken Enterprise/libexec/check_ping -H $HOSTADDRESS$ -w 100.0,90% -c 200.0,60%

The expanded/final command line to be executed for the host's check command would look like this:

```
/var/lib/Shinken Enterprise/libexec/check_ping -H 192.168.1.2 -w 100.0,90% -c 200.0,60%
```

You can use a single command definition to check an unlimited number of hosts. Each host can be checked with the same command definition because each host's address is automatically substituted in the command line before execution.

Example 2: Command Argument datas

You can pass arguments to commands as well, which is quite handy if you'd like to keep your command definitions rather generic. Arguments are specified in the object (i.e. host or service) definition, by separating them from the command name with exclamation points (!) like so:

The check:

Property	Value
host_name	linuxbox
Description	PING
command	check_ping!200.0,80%!400.0,40%

In the example above, the service check command has two arguments (which can be referenced with $ARGn$. The $ARG1$ data will be "200.0,80%" and "$ARG2$" will be "400.0,40%" (both without quotes). Assuming we are using the host definition given earlier and a check_ping command defined like this:

The command:

Property	Value
Name	check_ping
command line	/var/lib/Shinken Enterprise/libexec/check_ping -H $HOSTADDRESS$ -w $ARG1$ -c $ARG2$

The expanded/final command line to be executed for the service's check command would look like this:

```
/var/lib/Shinken Enterprise/libexec/check_ping -H 192.168.1.2 -w 200.0,40% -c 400.0,80%
```

If you need to pass bang (!) characters in your command arguments, you can do so by escaping them with a backslash (\). If you need to include backslashes in your command arguments, they should also be escaped with a backslash.

Custom Variable datas

Any custom object variables that you define in host, service, or contact definitions are also available as datas. Custom variable datas are named as follows:

- "$_HOSTvarname$"
- "$_SERVICEvarname$"
- "$_CONTACTvarname$"

Take the following host definition with a custom variable called ""_MACADDRESS""...

Host definiton

Property	Value
Name	linuxbox
address	192.168.1.1
_MACADDRESS	00:01:02:03:04:05

The "_MACADDRESS" custom variable would be available in a data called "$_HOSTMACADDRESS$".