Understanding the Declarative Wizard in Nagios XI

Purpose

This document describes the structure of the Declarative Wizard configuration wizard format and how it differs from traditional Configuration Wizards.

Note that the Declarative Wizard format is available in **Nagios XI 2026R2+**.

What are Declarative Wizards?

Declarative Wizards are a brand-new Configuration Wizard format that is designed from the ground up to streamline the process of creating wizards. The Declarative Wizard format is primarily composed of a JSON file that contains metadata, steps, hosts & services, plugins, and requirements to build a form that can be used to configure hosts and services in Nagios XI. Unlike the existing Configuration Wizards, form items are not built directly as HTML, but instead by selecting an input type, which will get converted into a preset form item at runtime.

An example Declarative Wizard, **Network Device SNMP,** is pre-loaded in Nagios XI to help you understand the settings and capabilities. You can learn more about the Wizard Builder here:

Using the Wizard Builder in Nagios XI

What is the Structure of Declarative Wizards?

The structure of Declarative Wizard can be thought of as being composed of the following sections:

Metadata

Top-level organizational information like the name and description of the wizard.

- All metadata will be found at the top-level of the wizard JSON object. The following fields are supported:
  - `"name": string` - Name of the wizard
  - `"displayTitle": string` - Title of the wizard as displayed to the user.
  - `"description": string` - Short description of the wizard's purpose and capabilities.
  - `"version": string` - Revision information for the wizard.
  - `"documentationUrl": string` - Will display as a clickable "Docs" link at the top of the Wizard Runner.
  - `"filterGroups": string array` - Filter groups for organizing wizards by type.

Steps

Pages on the Wizard that contain fields.

- Steps are composed as a JSON array known as “steps”. The following fields are supported:
  - `“title”: string` - Name of the step as displayed to the user.
  - `“description”: string` - Description of the step as displayed to the user.
  - `“schema”: object` - Contains field definitions for required fields, field types, and properties.

Hosts

Nagios Hosts that are to be created by the wizard containing the services.

- Hosts are composed as a JSON array known as “hosts”. The following fields are supported
  - `“source”: string` - Always set to “form”
  - `“hostname”: string` - Name of the host.
  - `“address”: string` - IP address of the host.
  - `“services”: object array` - See **Services.**
  - `“alias”: string` - Optional alternate display name.
- Hosts and services can utilize tokens from form input. The value of a field can be referenced by using the field’s name as defined in a step’s schema object.
  - For example, a field named field_name can have its value referenced by typing {{field_name}}
  - An example of this behavior can be found in the Network Device (SNMP) Declarative Wizard.

Services

Nagios Services that are to be created by the wizard to facilitate service checks

- Services are composed as a JSON array known as “services”. The following fields are supported
  - `“serviceDescription”: string` - Display name of the service.
  - `“checkCommand”: string` - Name of the Nagios check command to be used by the service.
  - `“checkCommandArgs”: string` - Arguments passed to the provided check command.
  - `“notes”: string` - Description of the service.
  - `“forEach”: string` - Key of an Object Discovery command. This will use the service as a template for each item of the key type found in the corresponding Object Discovery command output.

- Hosts and services can utilize tokens from form input. The value of a field can be referenced by using the field's name as defined in a step's schema object.
    - For example, a field named field_name can have its value referenced by typing {{field_name}}
    - Additionally, services can reference values from Object Discovery commands by referencing a column's key.
    - An example of this behavior can be found in the Network Device (SNMP) Declarative Wizard.

Prerequisite Checks

Nagios Plugins that will verify user input from fields. Wizard progress will be blocked unless the Prerequisite Check plugin returns exit code **0** (OK / Up)

- Please see the following **Object Discovery** section for information on the structure of both Prerequisite Checks and Object Discovery

Object Discovery

A wizard-specific plugin type that returns a JSON array containing key-value pairs to pass retrieved information to the Declarative Wizard for the purpose of procedurally building services. An example of this has been provided in the Imported Plugins directory as `discover_snmp_interfaces.sh`.

- The Imported Plugins directory can be found at `[install location]/html/includes/configwizard_plugins/`
- Prerequisite Checks and Object discovery are composed as a JSON array known as `"prereqChecks"`. The following fields are supported.
    - **`"afterStep": integer`** - Index of the step the command should follow.
    - **`"label": string`** - Name for the command step.
    - **`"checkCommand": string`** - If a registered Nagios check command is used, the name of the check command is placed here.
    - **`"checkCommandArgs": string`** - '`!`' separated list of arguments to be passed to the provided check command.
        - Each item separated by the '`!`' will map to `$ARG1$`, `$ARG2$`, …, `$ARGn$` for n arguments.
    - **`"plugin": string`** - If an Imported Plugin is used (Wizard specific plugin that can be imported via the Wizard Builder interface), place the name of the plugin here.
    - **`"pluginArgs": string`** - Arguments to be used by an Imported Plugin.
    - "kind": string - Include and set to "object-discovery" if the current command is a discovery command.
    - **`"id": string`** - Key for the object discovery command used to link to a service's `"forEach"` field.

- `"columns": object array` - Each column represents expected output from an Object Discovery command. Contains fields `"key"` and `"label"` which are both strings.
- `"minSelected": integer` - Minimum number of objects selected from Object Discovery output.

Requirements

Define required Imported Plugins, Check Commands, and system packages that are required by the wizard, as well as an error message that will display when attempting to run the configuration wizard if all requirements are not present.

- Requirements are composed of the following fields
  - `"plugins": string array` - Represents Imported Plugins that are expected to be run by this Declarative Wizard.
  - `"commands": string array` - Represents the expected registered check commands to be used by the wizard and/or the services it creates.

Finishing Up

This completes the documentation on Understanding the Declarative Wizard Format in Nagios XI. If you have additional questions or other support-related questions, please visit the Nagios Support Forum, Nagios Documentation Hub, or Nagios Library:

Visit Nagios Support Forum Visit Documentation Hub Visit Nagios Library