04-15-2025

Naming conventions

A naming convention is a system of using standard terms to classify categories of data so you can organize data in a way that makes sense to you.

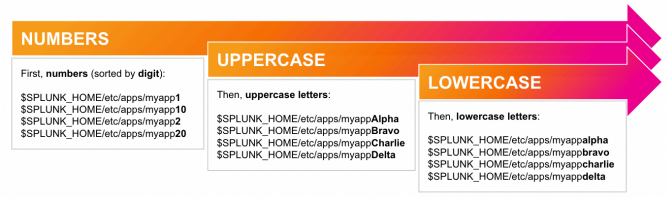
Lexicographical order is alpha-numeric name sorting. The Splunk platform uses lexicographical order to determine priority when processing knowledge object configurations. New Splunk users are often unaware of this, and incorrectly attribute faulty lexicographical ordering to buggy behavior. When you understand how Splunk lexicographical order works, you can use it to your advantage to enhance performance.

When you design a naming convention for your apps and indexes, consider lexicographical order to optimize search efficiency and accuracy.

How lexicographical order works

The Splunk configuration (.conf) files define the logic that controls how the Splunk platform performs actions, and in what order. The Splunk platform determines configuration priorities based on factors such as the current user and current app (scope) and alpha-numeric name sorting (lexicographical naming). This enables you to blend configurations from different files of the same configuration type, tune your data's source type, and increase the performance of indexing and searching.

The Splunk platform prioritizes numbers first, then capital letters, then lowercase letters:



Guidelines for leveraging lexicographical order in naming conventions

Keep these guidelines in mind when applying lexicographical order to a naming convention for your knowledge objects.

Before working with configuration files, check the Admin Manual to get familiar with the nuances of configuration file structure, configuration file directories, configuration file precedence, and when to restart Splunk Enterprise after a configuration file change for a direct `.conf` edit to apply.

To help keep it all straight, Splunk provides btool, a command-line utility, to troubleshoot issues with `.conf` file interactions and precedence. See Use btool to troubleshoot configurations.

Apply lexicographical order to internal app naming

One of the best ways to use lexicographical order is in your internal app's naming conventions. For example, a naming convention such as `<companyname>_<purpose>_<(app|add-on)>` enables you to easily differentiate your apps and configs from those downloaded from Splunkbase.

Lexicographical order affects the order in which apps are loaded, so you can control priority order with your naming convention. For example, if you want one group of objects to be evaluated before another, make sure the naming convention you apply sorts first according to the lexicographical hierarchy shown above.

Set up a global app to establish default behavior

You can use lexicographical order to set up a global app that will load before other apps. This enables you to set a general default behavior. The global app can contain basic settings. Apps that load after the global app can override (or fine tune) these settings as necessary.

For example, say you work for a company called 'Acme' and you created a global app named `Acme_zglobal_ta`. In that app, you have included global configurations, such as non-default port assignments and disabling the Splunk web server. On your search heads, you have deployed `Acme_searchhead_ta`, which enables the web server.

`Acme_zglobal_ta` loads first, which establishes that no Splunk instances will run the web server. `Acme_searchhead_ta` loads second and overrides that default behavior just for your search heads, so the web server will run only on your search heads.

Use unique names

You might run into issues if multiple knowledge objects of the same type share the same name, even if they belong to different apps. Objects get applied in reverse lexicographical order of the app directories. To avoid these issues, give each knowledge object a unique name. To understand more about how this works, see Give knowledge objects of the same type unique names.

Keep naming conventions for knowledge objects simple

If you apply a naming convention to knowledge objects, keep it simple. Use a system others can easily understand and apply. A complex syntax is harder for other users to understand and could discourage them from using it.

Naming conventions for indexes

Another best practice is to apply a naming convention to event indexes you create to manage user access, varying retention policies, or repositories for specific types of data or data from specific sources. A good naming convention

contains descriptors to help identify the index, categories to classify the index, and a summary. This enables you to use wild cards in configuration files, such as `authorization.conf`, to manage access. Create a naming convention that is simple so it's more likely to stay accurate over time. For example, `<companyname>_<purpose>_<sensitivity>_<summary>`.

`<companyname>`

- Include your company name to help differentiate indexes you create from indexes provided by Splunk or other solutions. For example, `acme`.

`<purpose>`

- The purpose descriptor indicates what the index was created to do. It should be generic enough to apply to any data source whose data is routed to that index. For example, `web`, or `firewall`. Keep names generic. Specific values, like product names, team names, organizations, people, and so on, are subject to change. For example, use `firewall` instead of a specific brand of firewall, such as `pan`.

`<sensitivity>`

- Sensitivity is a category that indicates the degree to which data in that index should be protected or restricted to certain user groups, for example, `topsecret`, `secret`, and `confidential`, `pci`, or `prod` and `nonprod`. This enables you to leverage wild cards when assigning permissions to certain indexes to users in certain roles.

`<summary>`

- If you have created a summary index to skim a large data set for a smaller subset of data to search, you can create an identifier for it in your naming scheme. See Use summary indexing for increased search efficiency for more about summary indexing.

Using the example above, you might create the following naming scheme for indexes that serve different types of data:

- `acme_web_pci:` an index for the Acme web server for information that is subject to PCI regulations
- `acme_firewall_prod:` an index for the Acme firewalls in production
- `acme_av_nonprod:` an index for the non-production anti-virus server at Acme
- `acme_all_prod_summary:` a summary index of correlations produced from all of Acme's production indexes

The next example uses wild cards In the `authorization.conf` file to demonstrate how you can use your naming convention to manage access to specific indexes for specific user roles using the following options in the `srchIndexesAllowed` attribute, where * is the name of any index that matches the naming criteria.

- `Acme_*:` For user roles that are allowed to access all data in all Acme indexes
- `Acme_*_nonprod:` For user roles that should never have access to production data
- `Acme_*_pci:` For user roles permitted to view PCI data

Naming conventions for source types

You should follow the naming conventions outlined in Source types for add-ons.

Next steps

Try the examples above using configurations and apps in your sandbox. Make up some scenarios of your own. Use `btool` with the `--debug` flag to explore how they are loaded.

Previous step

Next step

Back to the SSF homepage