Splunk® Application Development Cheat Sheet (v1.0)

https://www.aplura.com/cheatsheets

Development Environment Settings You Should Have

Use these settings in web.conf for a non-production environment. These settings will disable all caching, enable web debug logging, and un-minify JavaScript and CSS for readability.

```
[settings]
minify_js = False
minify_css = False
js_no_cache = True
cacheEntriesLimit = 0
cacheBytesLimit = 0
enableWebDebug = True
```

Do not use these settings in a production environment

See the Splunk Documentation on how to Enable Debug Logging

Useful strptime Directives

Year	%Y
Month (number/name or abbr)	%m/%b
Day	%d
Hour (24 hour/12 hour)	%H/%I
Minute	%M
Second/Millisecond	%S/%3N
Epoch	%s
Time zone (UTC offset/name)	%z/%Z
AM/PM	%p

Useful endpoints

/<language>/debug/refresh	Reloads various configurations without a restart
/<language>/_bump	Breaks the Splunk Web Cache and reloads some components without web restart
/<language>/static/docs/style/index.html	Has several Style Guides, Web Fonts, and other Web Framework Documentation

Adding custom elements to dashboard/form

Scripts and Style
Located in appserver/static

```
<dashboard script="my_script.js, my_script2.js" stylesheet="my_style.css">
```

Scripts are loaded in order (left to right)

A full restart is required for any new script

my_script.js
Located in appserver/static

```
require(["splunkjs/mvc", "backbone", "jquery", "underscore", "splunkjs/mvc/utils", "splunkjs/ready!"],
function (mvc, backbone, $, _, utils, ready) {    alert("Do my thang!"); });
```

Splunk Web Framework uses requireJS to load components.

Monitoring Console Health Check

checklist.conf

```
[my_health_check]
title = My App _internal Check
category = search
tags = myapp, foo, bar
description = This checks the Number of _internal Events.
failure_text = The count is invalid.
suggested_action = Check the error message for possible cause of failure.
search = index=_internal | stats count AS total_failures by host | fillnull total_failures value=0 | eval
message = “Error”, severity_level = case(total_failures==0, 3, total_failures > 0, 0)| rename host as
instance | fields instance total_failures message severity_level
```

The fields in bold are required for the check to display correctly in the Monitoring Console Health Check

Severity Levels and Icons
(Health Check)

Absent OR Level -1	Level 0	Level 1	Level 2	Level 3
Not Applicable	OK	INFO	WARN	CRITICAL

Basic Knowledge

Event Types

Use event types as a base for all dashboard panels, macros, and saved searches. An update to the event type will restore dashboard information for an end-user if the data moves indexes or sourcetypes.
Do NOT put macros within an event type.
I.E. **search = index=main sourcetype=mine `some_restriction`**

Know Your Version

Different versions of Splunk have different search commands.

Plan accordingly when developing an app, and be aware of what versions you are willing to support.

Eval functions

Indirect Ref

```
| eval http_{http_code} = http_message
```

The {} in this allows you to create a field with the value of a different field in the name of the new field

Make Results

```
| makeresults | eval my_field = split("item1,item2,item3", ",")| mvexpand my_field
```

Search-Time Operation Order

EXTRACT → REPORT → KV_MODE → FIELDALIAS → EVAL → LOOKUP

Provided by Aplura, LLC. Splunk Consulting and Application Development Services. sales@aplura.com • https://www.aplura.com

Many Solutions, One Goal

Developing Splunk Apps

Identify Your Audience

Determine the scope of your App. Who are you trying to reach? What is your main objective? Are you going to release this app on Splunkbase, and if so, how will you offer support? The more information you have on your target audience, the easier it will be to determine the best presentation styles, data to use, and method of configuration.

Identify Your Data

Determine where your data is located. Do you need a modular input, syslog collection, or other data source? In what format are your data? Do you need to modify, extend, or otherwise clean up the data prior to indexing? Do you need to onboard your data? What types of data do you have? Identifying what, where, and how your data are presented will help when building the internal data collection and knowledge objects.

Identify Your Branding and Visualizations

Determine the branding aspects. Do you want to brand anything at all? Do you need custom icons, or do you need permission to use third-party icons? Do you need a web developer to override CSS? Considering your branding goals will help focus the effort and resources required to successfully brand your Splunk App.

Best Practices

DO NOT:

- Include indexes.conf, limits.conf, or other administrative configurations
- Hard code paths into any file
- Store passwords/tokens in cleartext
- Leave any “local” configurations (including local.meta)
- Package lookups that are end-user modifiable (user lists, data based on environment)

DO:

- Include a base event type (event type which defines where the data is located for the entire app) as a base for all other macros, searches, dashboards, etc
- Develop in a dedicated environment
- Log any modular/scripted inputs and trap errors
- Design your App for a distributed deployment
- Use Common Information Model accepted fields as applicable
- Test on multiple OS/browser combinations (don’t forget Windows/IE)
- Use Splunk AppInspect on your App prior to SplunkBase submission

Provided by Aplura, LLC. Splunk Consulting and Application Development Services. sales@aplura.com • https://www.aplura.com

APLURA

Many Solutions, One Goal.

v2.5.2