

Extending SPL with Custom Search Commands

Jacob Leverich

Software Engineer, Splunk

.conf2016

splunk >

Disclaimer

During the course of this presentation, we may make forward looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC. The forward-looking statements made in the this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not, be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

Who am I?

- Splunker for 2 years, based in San Francisco
- Engineering lead for...
 - Machine Learning Toolkit
 - ITSI Anomaly Detection and Adaptive Thresholding features
 - Splunk custom search command interface
- Implemented Search Command Protocol Version 2
- Die-hard Longhorns fan



Agenda

- Introduction to Custom Search Commands
- How do Custom Search Commands work?
 - High-level concepts
 - Low-level details
- Types of Search Commands
- How to create new Custom Search Commands
- Wrap-up

Introduction to Custom Search Commands



.conf2016

What is a Custom Search Command?

- A user-defined SPL command.

splunk Search | Splunk 6.4.1 Search | Splunk 6.4.1 Jacob

127.0.0.1:8004/en-US/app/search/search?q=search%20index%3D_internal%0A%7C%20timecha...

App: Search & R... Administrator Messages Settings Activity Help Find

Search Pivot Reports Alerts Dashboards Search & Reporting

New Search Save As Close

index=_internal
| timechart span=1h sum(bytes) as bytes_per_hour
| eventstats avg(bytes_per_hour) as avg, stdev(bytes_per_hour) as sd
| where bytes_per_hour > avg+2*sd

All time

717,267 events (before 7/31/16 12:31:47.000 PM) No Event Sampling Job Smart Mode

Events Patterns Statistics (6) Visualization

20 Per Page Format Preview

_time	bytes_per_hour	avg	sd
2016-06-27 15:00	537341332	25656193.821656	130630657.753866
2016-07-07 10:00	561302261	25656193.821656	130630657.753866
2016-07-18 08:00	1119869427	25656193.821656	130630657.753866
2016-07-27 20:00	546033010	25656193.821656	130630657.753866
2016-07-31 11:00	541538796	25656193.821656	130630657.753866
2016-07-31 12:00	555149531	25656193.821656	130630657.753866

| search

| timechart

| eventstats

| where

Search | Splunk 6.4.1 Login | Splunk Jacob

127.0.0.1:8004/en-US/app/search/search?q=search%20index%3D_internal%20%7C%20GOCRAZY

splunk> App: Search & R... Administrator Messages Settings Activity Help Find

Search Pivot Reports Alerts Dashboards Search & Reporting

New Search

index=_internal | GOCRAZY All time Q

487 of 489 events matched No Event Sampling Job Smart Mode

Events (487) Patterns Statistics Visualization

Format Timeline Zoom Out Zoom to Selection Deselect 1 second per column

Jul 31, 2016 12:46:36 PM

List Format 20 Per Page Prev 1 2 3 4 5 6 7 8 9 ... Next

< Hide Fields	All Fields	i	Time	Event
Interesting Fields		>	7/31/16 12:48:14.763 PM	m__Furd_i_7x=ar_N1=pr100a,0i ncdhagaagxedros - o0e=7oc.iwsre l_ gei,06pein:_ cwig4,oxlt_arm=e0OneI0ede0iuta,d=,4n1m 0t0ua:iip3dshoe p3_oe8 =m-2lntpax,0,l1t cag m0 oxn6aa0g._mt7t_uktli,,girt0h=n11t1l=a0=eawy0pw=mdlann,na0g= c0r= dg0-.xsr c= a0=ettm2t0usmd0,e,0-ro 0 sdm
a cieipntl 1		>	7/31/16 12:48:14.763 PM	=8 24mm1so=npti.s9-F1as=c 2a0u 311=,ma806700egr1-0 pg,_uI4-92u9 t47vdM0ds3se,_r 3, ic0siv9 6a :r4 0 =xex=eu v06ee3c61e,- 177 Sodu0op3r3r5=Na,04_y1:d2_atp17
a copnmtneo 1		>	7/31/16 12:48:14.589 PM	ereltc18%=3/t".5.T 7eed0=acPd.rr-%a4vKS8He 9." /.lii3p10elsanrdce2qpich/3 SyWni6 e0me9mIntpm1xb.&i0gey & st&.04pfti-/OiSe &/:/ cs&.s1e24cjr0da36e6 rtC5cne2fas6C Or5s00)p1p:nw:1ad3:rr/12mU3in/ahsna"nel/ntin0o_&3l9hs.ni8se4.0ias.i0ua2S4=fdlast 4l=Mntf. ea0aKa9=u2e/o8e6okmrfh9l)asdae-o 0e%07ga a8otevpoa 0G.l_30z8t1.u.1g 0eb 2a(ldp9b00pbe3to3eyeie;7.ca4y==J8Ta4adp7:6"t=e0t1.r/yHsci 47035""&41re, i5s a 66 ?T..cse011nra./Mt11/1pehcoo9ts0e =eercm7s1ih40I7ve/c.sbpT1spe4(1v190i7l.sien5 6cidsp_6-d/9.sE s1 d/rr0syv80i =.19yfb&ty3/j?mDetk102s3h4:hakes1fp.fG90opc&zsal sbh 5sln5=t.%/tat63631sat_lteMaesl/_X7/1fla Ou&p&=f0ni4ai[cf=eg2MpA -htr1..
a d_eatryea 1		>	7/31/16 12:48:14.588 PM	svi_05Hq-?ectt32a&125smeK2dsedume6Ta0010n.Scpcb.oClef mv18r3 leHm/8e 0l .cenp1ge tct1co7Eam th0n7h_m5i80cp5:1p&respa&i1v7sK%1e;ias_e_fc15o/p 1t/rr/yC4e&mf4/0./Ne/e1sJc0d.s050"ng0eeof1h_0=cep.=9=aS.1s]=ea3rb 2k)0C1=l&ta9:pp4a od..4ib(?GU83e82eTh608ee6n0.sh1_k=vj-/a2=W96v97p.1lyr8.fot4 l p.i0aef7ad/s"fmT.yM s%a0.s0ets4sD/5nene="8_t.re.le0 :t" rM1p4/_2azf=/3s""/ap ./r8s4iu99Usad544/r1s06a53pu04&si=7er.lt/1a/&o:ihrairm.t36ba6rr/nAeqs.cyo31&031

< Hide Fields All Fields

Interesting Fields

- a cieipntl 1
- a copnmtneo 1
- # d_eatryea 1
- # d_eatymda 1
- a d_eatyyda 1
- a d_entezoo 1
- # d_eutrhoa 1
- a dc_100+
- # dis 4
- a dmnrn_eirfoae 1
- # dsd_ncteoa 60
- a deni 1
- a aam 32
- a ec_rptuoey 3
- # elif 17

What is a Custom Search Command?

- A user-defined SPL command.
- Can be used to extend the SPL language!

Who uses Custom Search Commands?

- Partners
 - Concanon, etc.
- Customers
 - Use-case specific analytics
- Splunk!
 - **predict** command
 - DB Connect
 - Machine Learning Toolkit
- Anyone who wants to extend the Splunk platform
 - Integration with 3rd party services
 - Implementation of custom logic

How do Custom Search Commands work?



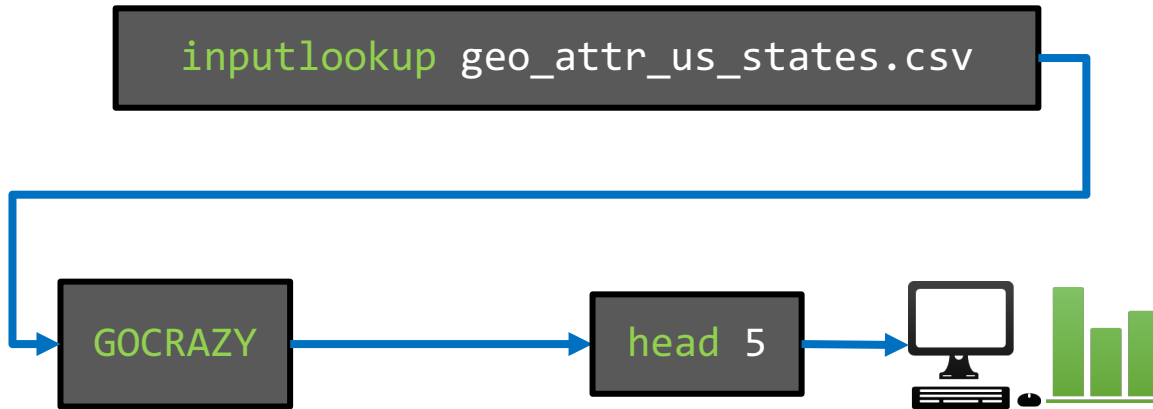
.conf2016

How do Custom Search Commands work?

1. When parsing SPL, splunkd interrogates each command.
“Are you a Custom Search Command?”
2. If so, spawn external process and allow it to parse arguments.
3. During search, pipe search results through external process.

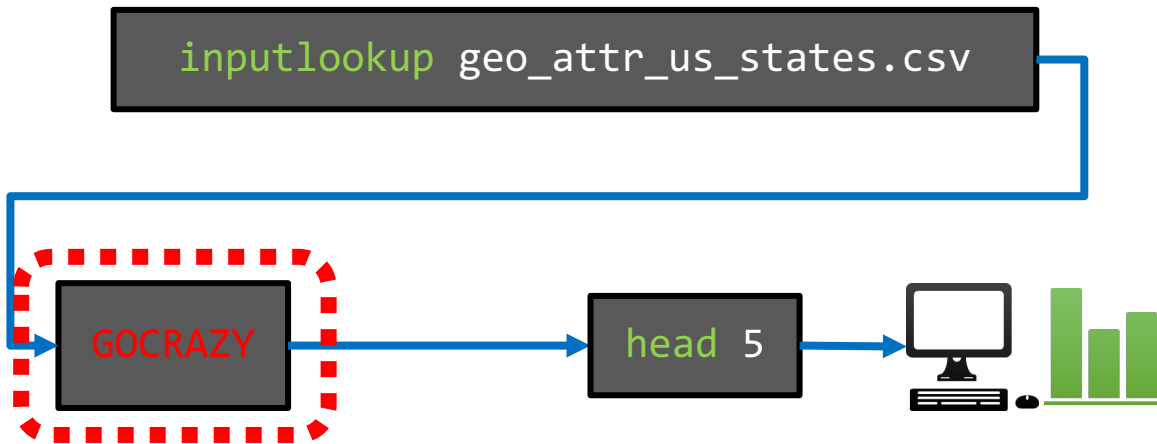
Parsing #1: Split search into commands

```
| inputlookup geo_attr_us_states.csv | GOCRAZY | head 5
```



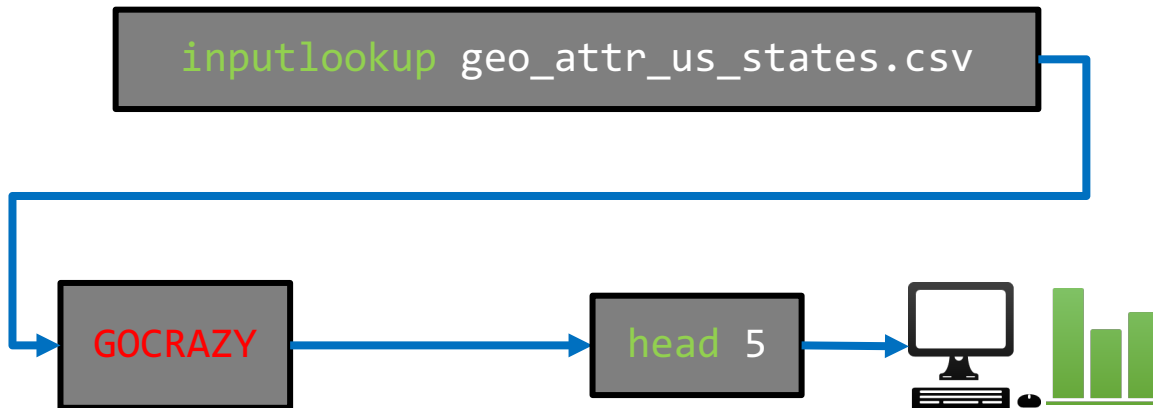
Parsing #2: Look for custom search commands

```
| inputlookup geo_attr_us_states.csv | GOCRAZY | head 5
```



Parsing #3: Spawn external process

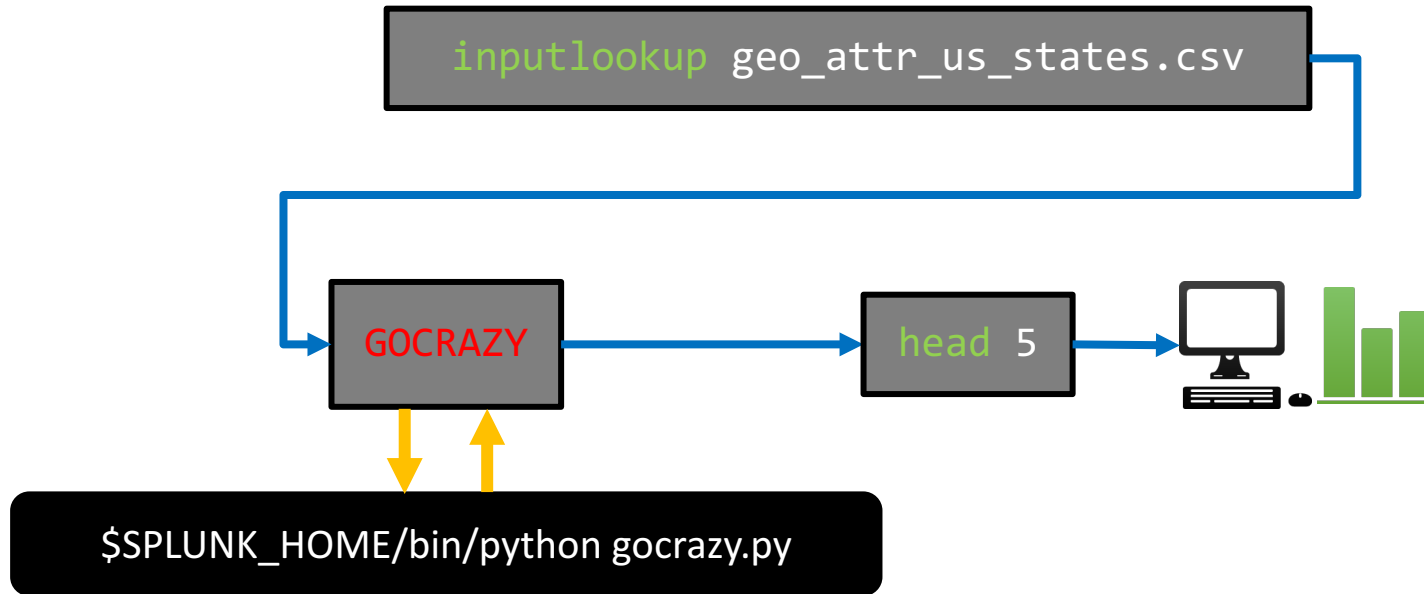
```
| inputlookup geo_attr_us_states.csv | GOCRAZY | head 5
```



```
$SPLUNK_HOME/bin/python gocrazy.py
```

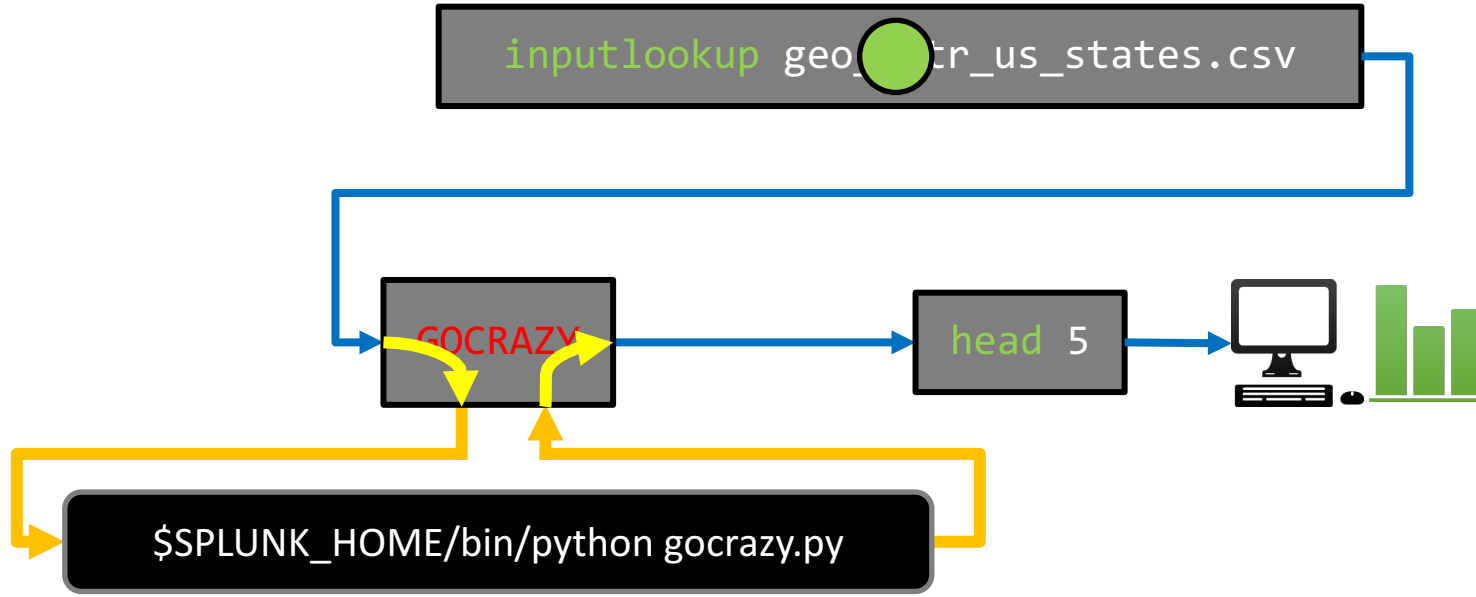
Parsing #4: Let external process parse arguments

```
| inputlookup geo_attr_us_states.csv | GOCRAZY | head 5
```



Search: Pipe results **through** external process

```
| inputlookup geo_attr_us_states.csv | GOCRAZY | head 5
```



Recap: high-level concepts

- Enable you to register new SPL commands, extend the language.
- Allow you to intercept and modify search results during a search.
 - CSV in → CSV out
- Implemented as a external process (i.e. a program you write).
 - Typically written in Python.

Custom Commands: low-level details

- How results are exchanged between splunkd and external process
- “Types” of search commands

splunkd ↔ custom command

- There are two “protocols” for custom commands:
 - Version 1, legacy protocol used by Intersplunk.py (available since Splunk 3.0)
 - Version 2, new protocol used by Python SDK (available since 6.3)
 - In both protocols, all communication over stdin/stdout
- Version 2 protocol
 - Spawns external process once, streams results through chunk by chunk
 - Simple commands.conf configuration
 - “chunked=true”
 - Support for platform-specific programs
- Version 1 protocol
 - Spawns external process for each chunk of search results (!)
 - “Transforming” commands limited to 50,000 events

Search Command protocol comparison

Protocol	APIs	Performance	Scalability	Simple configuration	Platform-specific programs	Programming languages
Version 1 (legacy)	Intersplunk.py, Python SDK	✗	✗	✗	✗	Python
Version 2	Python SDK	✓	✓	✓	✓	Python, Javascript, bash, Shell, <i>arbitrary binaries</i>

Search Command Protocol Version 2

- Transaction-oriented
 - splunkd sends a command, external process responds with reply
- Simple bi-directional transport protocol:
 - ASCII transport header
 - JSON metadata payload
 - CSV search results payload
- Every search starts with a “getinfo” command (capability exchange)
- Subsequently, issues “execute” commands with search results

Transport “chunk”

Metadata length

Data length

```
chunked 1.0, 22 54
```

```
{“action”: “execute”}
```

```
_raw,a,b,c  
hello,0,1,2  
everyone,3,4,5  
howareyou,6,7,8
```

Transport header

Metadata (JSON)

Data payload (CSV)

Example: GOCRAZY

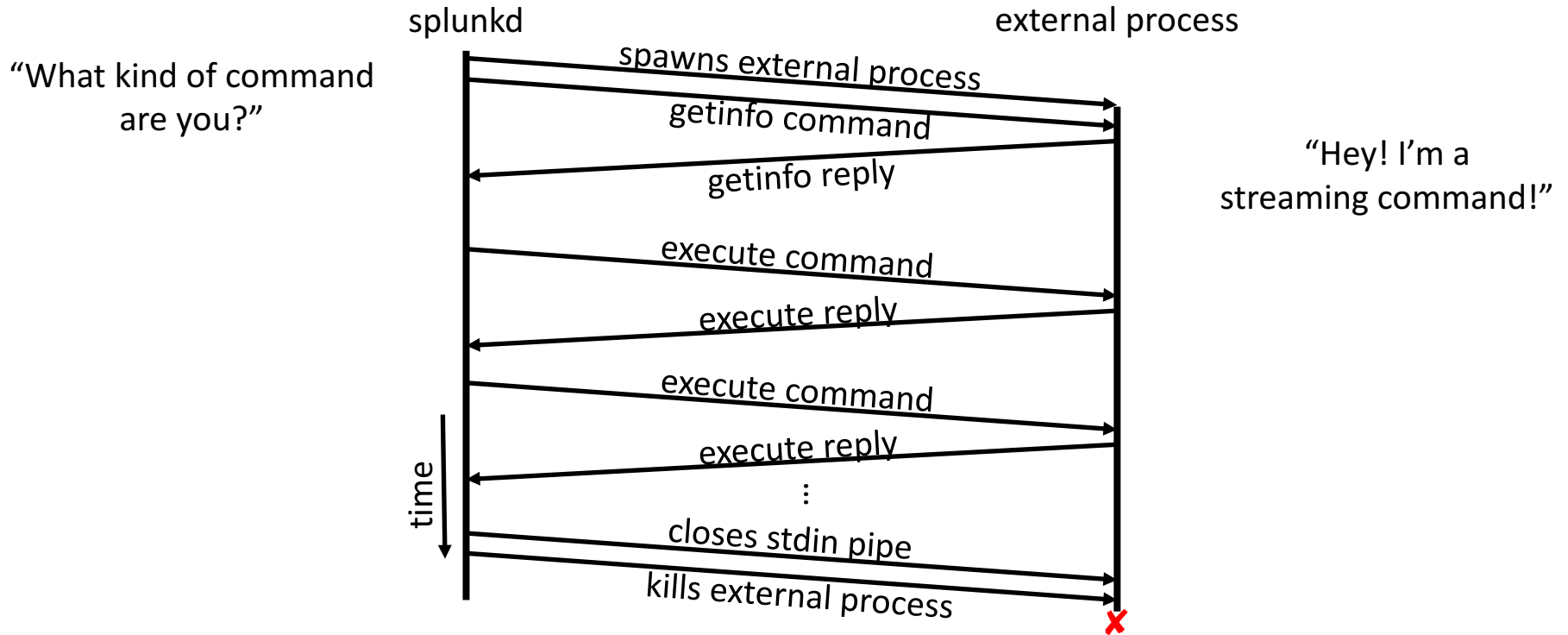
| `inputlookup geo_attr_us_states.csv` | `head 5` | `GOCRAZY`

```
chunked 1.0,22,106
{"action": "execute"}
state_code,state_fips,state_name
AL,01,Alabama
AK,02,Alaska
AZ,04,Arizona
AR,05,Arkansas
CA,06,California
```

```
$SPLUNK_HOME/bin/python
gocrazy.py
```

```
chunked 1.0,18,106
{"finished": true}
dste_aecot,pste_asfit,mste_aenat
LA,10,aaalbmA
KA,20,laaskA
ZA,40,iaorzna
RA,50,Akaasnr
AC,60,iCifolarna
```


Protocol Version 2: Transaction timeline



“getinfo” command

- Metadata in the getinfo command sent by splunkd:
 - Command arguments
 - Full SPL query string
 - Execution context (app, user)
 - Search sid
 - splunkd URI and auth token (for making REST requests)
- Metadata in the custom command’s reply:
 - Type of search command (streaming/stateful/reporting/etc.)
 - Which fields splunkd should extract (required fields)
 - Whether or not it generates results (e.g. must be first search command)

Sample “getinfo” metadata

```
{
  "action": "getinfo",
  "streaming_command_will_restart": false,
  "searchinfo": {
    "earliest_time": "0",
    "raw_args": [
      "LinearRegression", "petal_length", "from", "petal_width"
    ],
    "session_key": "...",
    "maxresultrows": 50000,
    "args": [
      "LinearRegression", "petal_length", "from", "petal_width"
    ],
    "dispatch_dir": "/Users/jleverich/builds/conf_mlapp_demo/var/run/splunk/dispatch/1475007525.265",
    "command": "fit",
    "latest_time": "0",
    "sid": "1475007525.265",
    "splunk_version": "6.5.0",
    "username": "admin",
    "search": "%7C%20inputlookup%20iris.csv%20%7C%20fit%20LinearRegression%20petal_length%20from%20petal_width",
    "splunkd_uri": "https://127.0.0.1:8090",
    "owner": "admin",
    "app": "Splunk_ML_Toolkit"
  },
  "preview": false
}
```

“execute” command

- Metadata in execute command sent by splunkd
 - Whether or not preceding commands are “finished”
- Metadata in the custom command’s reply:
 - Whether or not this command is “finished”
- splunkd and search commands negotiate completion of search
 - Both must indicate “finished” = True

Types of Search Commands



.conf2016

Types of Search Commands

- “Streaming” commands
- “Stateful Streaming” commands
- “Transforming” commands
 - “Events” commands
 - “Reporting” commands

“Streaming” commands

- Process search results one-by-one
 - Can't maintain global state
 - Must not re-order search results
- Eligible to run at Indexers
 - Can run in parallel on Indexers
- Examples:
 - `eval`
 - `where`
 - `rex`

“Streaming” command example

Remote results

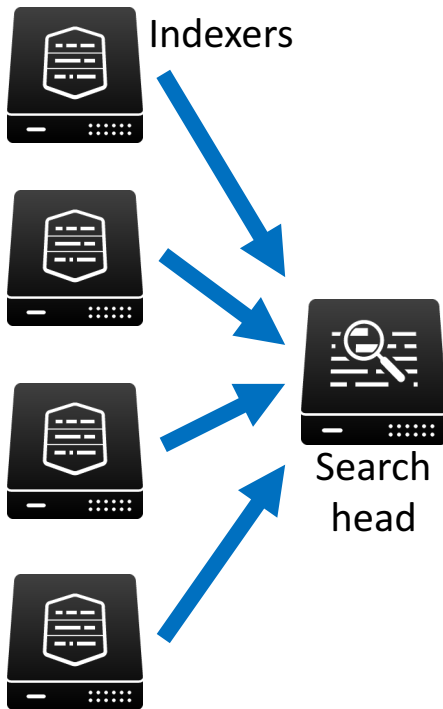
field_A	field_B	field_C
the	jumps	dog

field_A	field_B	field_C
quick	over	oops

field_A	field_B	field_C
brown	the	too

field_A	field_B	field_C
fox	lazy	many

... | eval foo="bar" | ...



Final search results

field_A	field_B	field_C	foo
the	jumps	dog	bar
quick	over	oops	bar
brown	the	too	bar
fox	lazy	many	bar

“Stateful Streaming” commands

- Process search results one-by-one
 - **Can** maintain global state
 - Must not re-order search results
- Only run at Search Head
- Examples:
 - accum
 - streamstats
 - dedup

“Stateful Streaming” command example

... | accum foo | ...

field_A	field_B	field_C	foo
the	jumps	dog	1
quick	over	oops	1
brown	the	too	1
fox	lazy	many	1



field_A	field_B	field_C	foo
the	jumps	dog	1
quick	over	oops	2
brown	the	too	3
fox	lazy	many	4

“Events” commands

- Process search results as a whole
 - May re-order search results
 - Typically maintain all fields in each event, especially:
 - `_raw`, `_time`, `index`, `sourcetype`, `source`, `host`
- Only run at Search Head
- May run several times for “preview”
- Examples:
 - `sort`
 - `eventstats`

“Events” command example

```
... | sort field_A | ...
```

field_A	field_B	field_C	foo
the	jumps	dog	1
quick	over	oops	2
brown	the	too	3
fox	lazy	many	4



field_A	field_B	field_C	foo
brown	the	too	3
fox	lazy	many	4
quick	over	oops	2
the	jumps	dog	1

“Reporting” commands

- Process search results as a whole
 - Typically transform the results (e.g. aggregate, project, summarize, etc.)
- Only run at Search Head
- May run several times for “preview”
- Results show up in the “Statistics” tab
- Examples:
 - stats
 - timechart
 - transpose

“Reporting” command example

... | stats count | ...

field_A	field_B	field_C	foo
the	jumps	dog	1
quick	over	oops	2
brown	the	too	3
fox	lazy	many	4



count
4

Beware of large result sets!

- “Events” and “Reporting” commands process results as a whole.
 - May contain 1,000,000s of search results!
 - Write Streaming or Stateful commands instead when possible.
- Build-in capacity limits, or spill results to disk when necessary.

Streaming “pre-op”

- Commands may specify a “pre-op” to prepend in SPL

... | stats count | ... **➔** ... | prestats count | stats count | ...

- Communicated to splunkd in getinfo metadata (streaming_preop)
- Useful to parallelize computation, reduce volume of data transfer
- Must be “Streaming” (i.e., may run at Indexers)

Implementing Custom Search Commands with the Splunk SDK for Python

.conf2016

Basic steps to create a search command

1. Create an “App”
2. Deploy the Python SDK for Splunk in the **bin** directory
3. Write a script for your Custom Search Command
4. Register your command in **commands.conf**
5. Restart Splunk Enterprise
6. *(optional)* Export the command to other apps

Create an "App"

The screenshot shows the Splunk Manager interface for managing apps. The browser address bar indicates the URL is `127.0.0.1:8004/en-US/manager/search/apps/local`. The page title is "Apps". Below the title, there are three buttons: "Browse more apps" (highlighted in green), "Install app from file", and "Create app". A red arrow points to the "Create app" button. Below the buttons, it says "Showing 1-19 of 19 items" and "Results per page 25". A table lists installed apps with columns for Name, Folder name, Version, Update checking, Visible, Sharing, Status, and Actions.

Name	Folder name	Version	Update checking	Visible	Sharing	Status	Actions
SplunkForwarder	SplunkForwarder		Yes	No	App Permissions	Disabled Enable	
SplunkLightForwarder	SplunkLightForwarder		Yes	No	App Permissions	Disabled Enable	
ML Toolkit and Showcase	Splunk_ML_Toolkit	1.2	Yes	Yes	App Permissions	Enabled Disable	Launch app
Python for Scientific Computing	Splunk_SA_Scientific_Python_darwin_x86_64	1.2	Yes	No	App Permissions	Enabled Disable	Edit propert
Log Event Alert Action	alert_logevent	6.4.1	Yes	No	App Permissions	Enabled Disable	Edit propert
Webhook Alert Action	alert_webhook	6.4.1	Yes	No	App Permissions	Enabled Disable	Edit propert

Deploy the Python SDK in the **bin** directory

```
cd $SPLUNK_HOME/etc/apps/MyNewApp/bin
```

```
pip install -t . splunk-sdk
```

Write a script for your Custom Search Command

```
$SPLUNK_HOME/etc/apps/MyNewApp/bin/foobar.py
```

```
import sys
from splunklib.searchcommands import dispatch, StreamingCommand, Configuration

@Configuration()
class FoobarCommand(StreamingCommand):
    def stream(self, records):
        for record in records:
            record['foo'] = 'bar'
            yield record

if __name__ == "__main__":
    dispatch(FoobarCommand, sys.argv, sys.stdin, sys.stdout, __name__)
```

Register your command in `commands.conf`

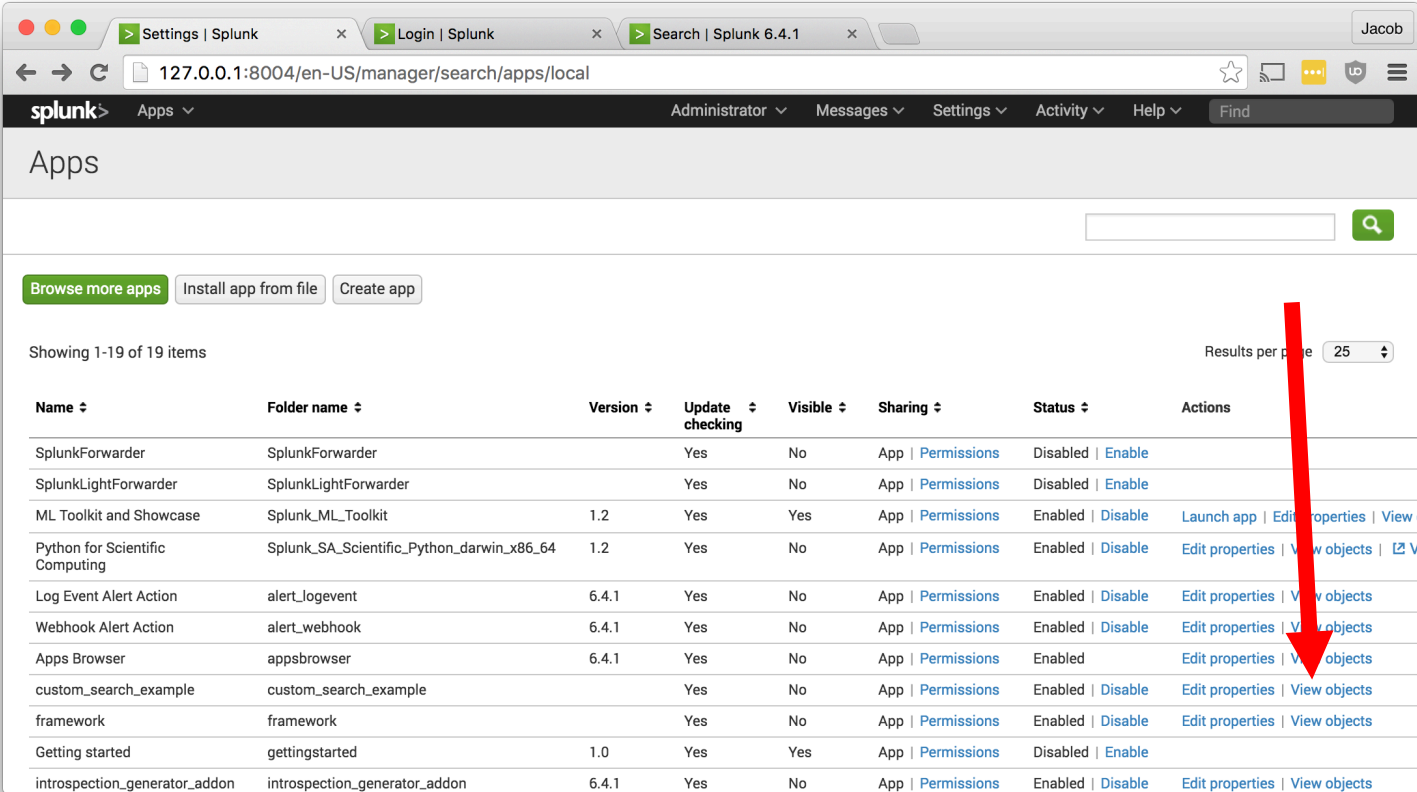
```
$SPLUNK_HOME/etc/apps/MyNewApp/default/commands.conf
```

```
[foobar]  
chunked=true  
# filename=foobar.py    ## <--- optional
```

Restart Splunk Enterprise

```
$SPLUNK_HOME/bin/splunk restart
```

Export to other apps (optional)



The screenshot shows the Splunk web interface for managing apps. The browser address bar indicates the URL is `127.0.0.1:8004/en-US/manager/search/apps/local`. The page title is "Apps". Below the title, there are buttons for "Browse more apps", "Install app from file", and "Create app". A search bar is also present. The main content area shows a list of 19 items, with the first few rows visible in the table below. A red arrow points to the "View objects" link in the "Actions" column for the "Apps Browser" app.

Showing 1-19 of 19 items Results per page 25

Name	Folder name	Version	Update checking	Visible	Sharing	Status	Actions
SplunkForwarder	SplunkForwarder		Yes	No	App Permissions	Disabled Enable	
SplunkLightForwarder	SplunkLightForwarder		Yes	No	App Permissions	Disabled Enable	
ML Toolkit and Showcase	Splunk_ML_Toolkit	1.2	Yes	Yes	App Permissions	Enabled Disable	Launch app Edit properties View objects
Python for Scientific Computing	Splunk_SA_Scientific_Python_darwin_x86_64	1.2	Yes	No	App Permissions	Enabled Disable	Edit properties View objects View objects
Log Event Alert Action	alert_logevent	6.4.1	Yes	No	App Permissions	Enabled Disable	Edit properties View objects
Webhook Alert Action	alert_webhook	6.4.1	Yes	No	App Permissions	Enabled Disable	Edit properties View objects
Apps Browser	appsbrowser	6.4.1	Yes	No	App Permissions	Enabled	Edit properties View objects
custom_search_example	custom_search_example		Yes	No	App Permissions	Enabled Disable	Edit properties View objects
framework	framework		Yes	No	App Permissions	Enabled Disable	Edit properties View objects
Getting started	gettingstarted	1.0	Yes	Yes	App Permissions	Disabled Enable	
introspection_generator_addon	introspection_generator_addon	6.4.1	Yes	No	App Permissions	Enabled Disable	Edit properties View objects

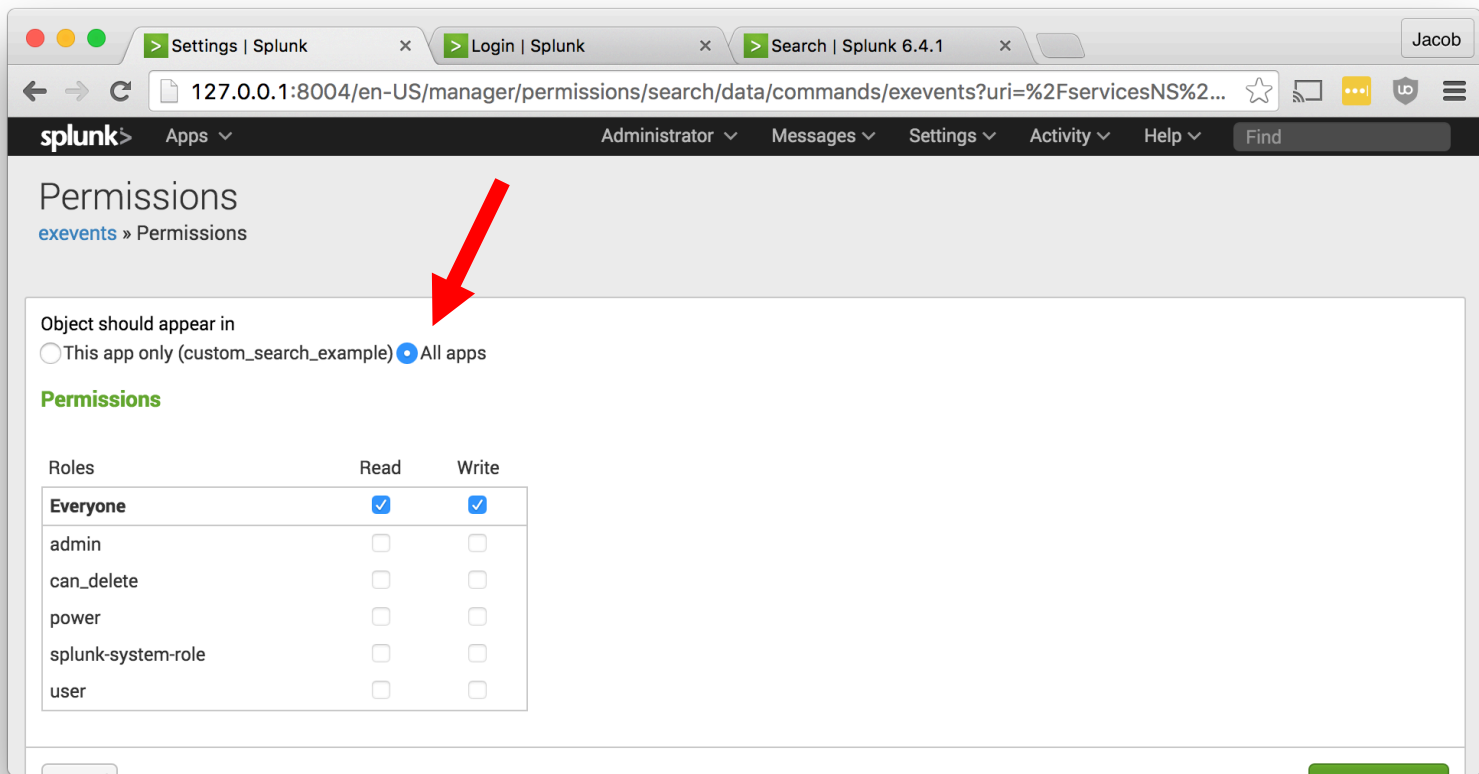
Export to other apps (optional)

The screenshot shows the Splunk web interface for managing configurations. The browser tabs include 'Settings | Splunk', 'Login | Splunk', and 'Search | Splunk 6.4.1'. The URL is '127.0.0.1:8004/en-US/manager/search/admin/directory?ns=custom_search_example&app_only=1'. The user is logged in as 'Administrator'. The page title is 'All configurations'. Below the title, there are filters for 'App context' (set to 'custom_search_example (custo)') and 'Owner' (set to 'Any'). A search bar is also present. A checkbox is checked for 'Show only objects created in this app context'. The main content area shows a table of configurations with 6 items. A red arrow points to the 'Permissions' link in the 'Sharing' column of the first row.

Showing 1-6 of 6 items Results per page 25

Name	Config type	Owner	App	Sharing	Status
exevents	commands	No owner	custom_search_example	Global Permissions	Enabled Disable
exreport	commands	No owner	custom_search_example	Global Permissions	Enabled Disable
exstateful	commands	No owner	custom_search_example	Global Permissions	Enabled Disable
exstream	commands	No owner	custom_search_example	Global Permissions	Enabled Disable
gocrazy	commands	No owner	custom_search_example	Global Permissions	Enabled Disable
levenshtein	commands	No owner	custom_search_example	Global Permissions	Enabled Disable

Export to other apps (optional)



The screenshot shows the Splunk web interface for configuring permissions for the 'exevents' object. The browser tabs include 'Settings | Splunk', 'Login | Splunk', and 'Search | Splunk 6.4.1'. The user 'Jacob' is logged in. The page title is 'Permissions' with a breadcrumb 'exevents » Permissions'. A red arrow points to the 'Object should appear in' section, where the 'All apps' radio button is selected. Below this is a 'Permissions' table with columns for 'Roles', 'Read', and 'Write'.

Object should appear in

This app only (custom_search_example) All apps

Permissions

Roles	Read	Write
Everyone	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
admin	<input type="checkbox"/>	<input type="checkbox"/>
can_delete	<input type="checkbox"/>	<input type="checkbox"/>
power	<input type="checkbox"/>	<input type="checkbox"/>
splunk-system-role	<input type="checkbox"/>	<input type="checkbox"/>
user	<input type="checkbox"/>	<input type="checkbox"/>

Example Streaming Command

```
$SPLUNK_HOME/etc/apps/MyNewApp/bin/exstream.py
```

```
import sys
from splunklib.searchcommands import dispatch, StreamingCommand, Configuration

@Configuration()
class ExStreamCommand(StreamingCommand):
    def stream(self, records):
        for record in records:
            record['foo'] = 'bar'
            yield record

if __name__ == "__main__":
    dispatch(ExStreamCommand, sys.argv, sys.stdin, sys.stdout, __name__)
```

Example Stateful Streaming Command

```
$SPLUNK_HOME/etc/apps/MyNewApp/bin/exstateful.py
```

```
import sys
from splunklib.searchcommands import dispatch, StreamingCommand, Configuration

@Configuration(local=True)
class ExStatefulCommand(StreamingCommand):
    def stream(self, records):
        for record in records:
            record['foo'] = 'bar'
            yield record

if __name__ == "__main__":
    dispatch(ExStatefulCommand, sys.argv, sys.stdin, sys.stdout, __name__)
```

Example Events Command

```
$SPLUNK_HOME/etc/apps/MyNewApp/bin/exevents.py
```

```
import sys
from splunklib.searchcommands import dispatch, EventingCommand, Configuration

@Configuration()
class ExEventsCommand(EventingCommand):
    def transform(self, records):
        l = list(records)
        l.sort(key=lambda r: r['_raw'])
        return l

if __name__ == "__main__":
    dispatch(ExEventsCommand, sys.argv, sys.stdin, sys.stdout, __name__)
```

Example Reporting Command

```
$SPLUNK_HOME/etc/apps/MyNewApp/bin/exreport.py
```

```
import sys
from splunklib.searchcommands import dispatch, ReportingCommand, Configuration

@Configuration()
class ExReportCommand(ReportingCommand):
    @Configuration()
    def map(self, records):
        return records

    def reduce(self, records):
        count = 0
        for r in records:
            count += 1
        return [{'count': count}]

if __name__ == "__main__":
    dispatch(ExReportCommand, sys.argv, sys.stdin, sys.stdout, __name__)
```

A little advice

- Custom commands are **programs** that run on Splunk instances
 - **BEWARE UNVALIDATED INPUT!**
 - Sanitize user arguments AND search results
- Use role-based access control to r
- Be prepared to handle 1,000,000s
- **Be excellent to each other.**

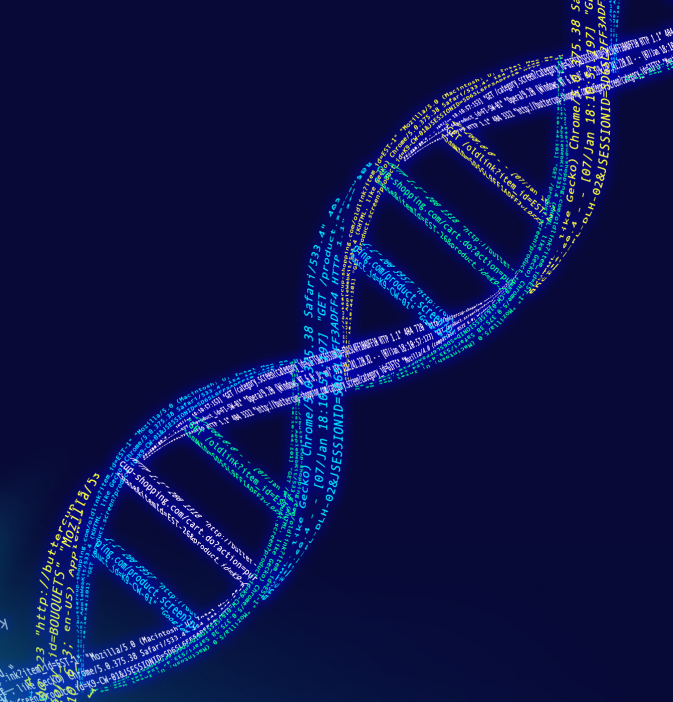


What Now?

- <https://github.com/splunk/splunk-sdk-python>
 - https://github.com/splunk/splunk-sdk-python/tree/master/examples/searchcommands_app
- Dev Portal Documentation
 - <http://dev.splunk.com/view/python-sdk/SP-CAAUEU2>
- Detailed specification for Protocol Version 2 available by request
- PM Contact: Mark Groves <mgroves@splunk.com>

THANK YOU

.conf2016



Streaming Commands only serialize required fields

```
{“required_fields”: [“fieldX”], ...}
```

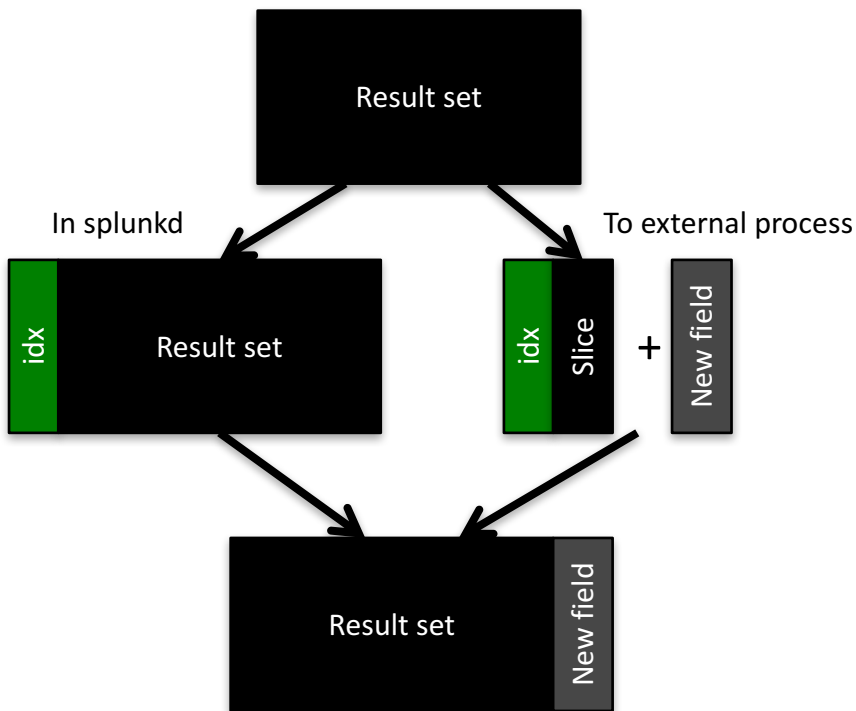
Internal result set

```
_raw,_time,_cd,_indextime,...,fieldX  
a,1400000000,x:y,1400000010,...,BOB  
a,1400000001,x:y,1400000011,...,JIM  
a,1400000002,x:y,1400000012,...,BOB  
a,1400000003,x:y,1400000013,...,JIM  
a,1400000004,x:y,1400000014,...,JIM  
a,1400000005,x:y,1400000015,...,BOB  
a,1400000006,x:y,1400000016,...,JIM  
a,1400000007,x:y,1400000017,...,BOB  
a,1400000008,x:y,1400000018,...,BOB  
a,1400000009,x:y,1400000019,...,JIM
```

External result set

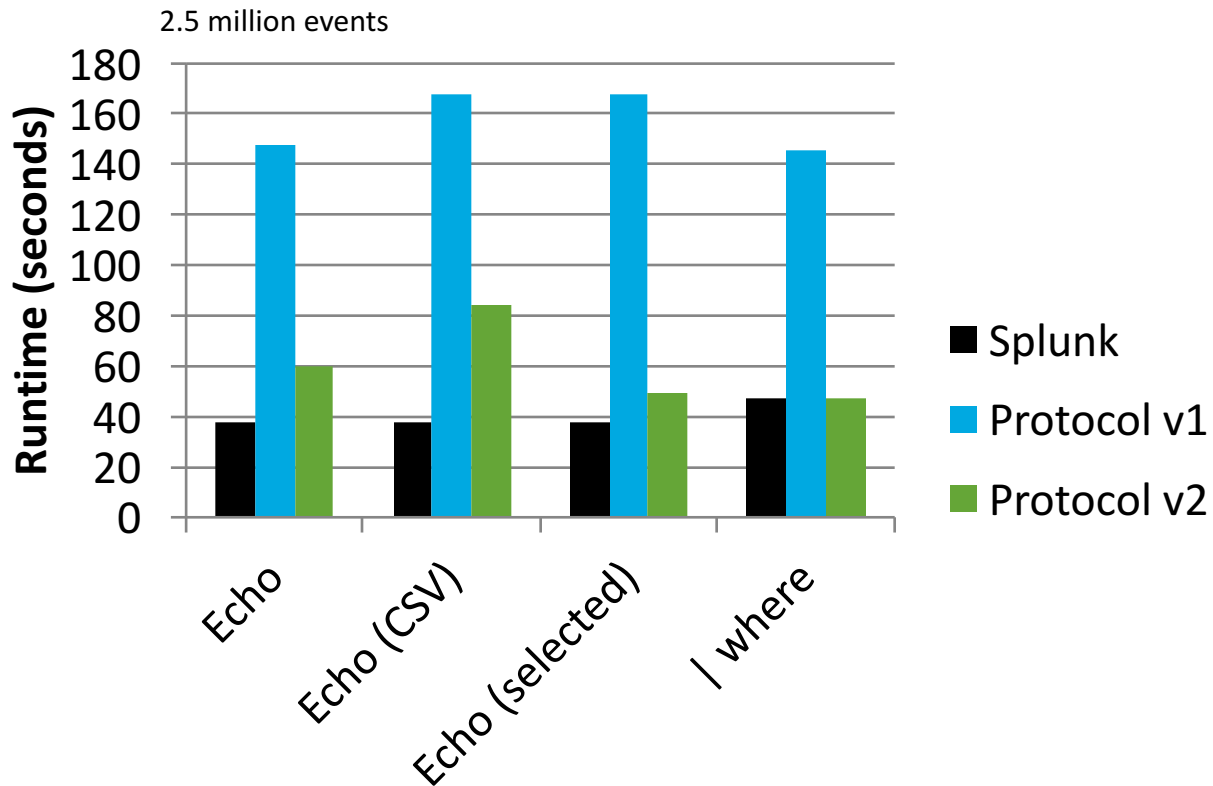
```
_chunked_idx,fieldX  
0,BOB  
1,JIM  
2,BOB  
3,JIM  
4,JIM  
5,BOB  
6,JIM  
7,BOB  
8,BOB  
9,JIM
```

“Right outer-join” on required fields



- Supports
 - Removing events
 - Adding events
 - Editing fields
 - Adding fields
- Can't re-order events

Performance comparison



“Streaming” command example

```
... | eval foo="bar" | ...
```

field_A	field_B	field_C
the	jumps	dog
quick	over	oops
brown	the	too
fox	lazy	many



field_A	field_B	field_C	foo
the	jumps	dog	bar
quick	over	oops	bar
brown	the	too	bar
fox	lazy	many	bar