The ‘state’ of Splunk

Using the KVStore to maintain App State

Stefan Sievert

Client Architect, Splunk Inc.

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.

Agenda

- Why do we need a state store for apps?
- App KV Store overview
- App KV Store example app & code
- Configuration & API
- Summary
- Looking ahead
- Q&A

Houston, We Have…

splunk>

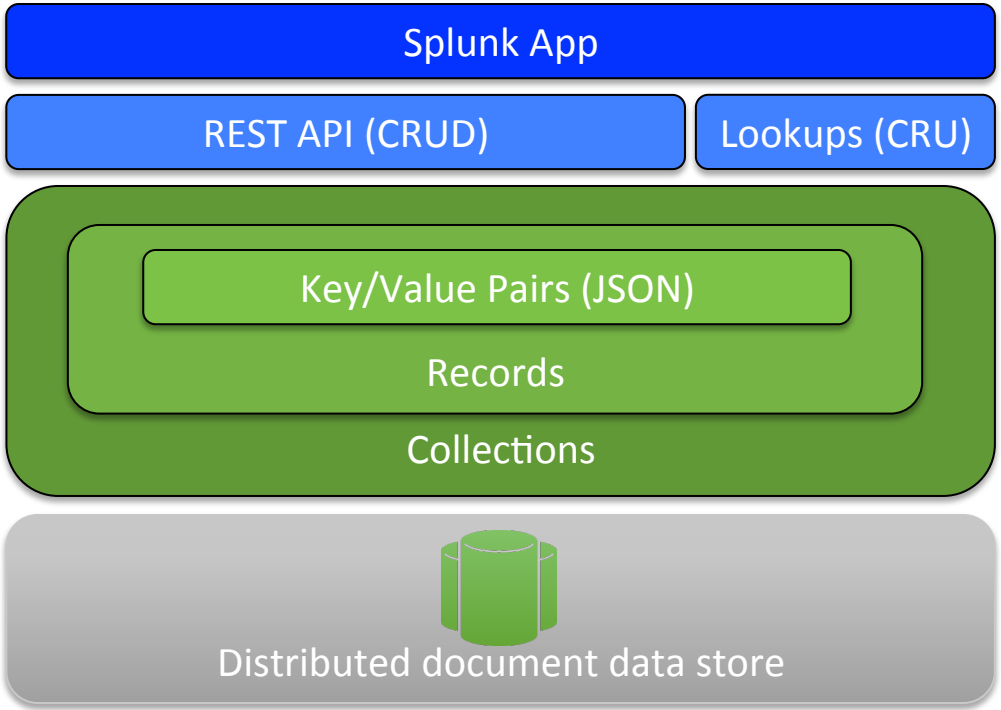
What's The Problem To Solve?

- Managing and using non-event data required csv files or external data stores
- Keeping state in an application to enable user workflow was not easy
- Keeping processing state (checkpoint data) in a modular input
- No mechanism to distribute dynamic state or data for search time event enrichment

App KV Store Overview

splunk>

App KVStore Overview



- Schema support
- Data types can be enforced for fields (number, boolean, time, string)
- Support for field accelerations (index)

Concepts

- **Collections** are the containers for your data, similar to a database table. Collections exist within the context of a given app.
- **Records** contain each entry of your data, similar to a row in a database table.
- **Fields** correspond to key names, similar to the columns in a database table. Fields contain the values of your data. You can optionally enforce data types (number, boolean, time, and string) for field values.
- **_key** is a reserved field that contains the unique ID for each record. It can be user-provided or app auto-generated.
- **_user** is a reserved field that contains the user ID for each record. This field cannot be overridden.
- **Accelerations** improve search performance by making searches that contain accelerated fields return faster.

App KVStore Uses

- Tracking workflow, e.g. for an incident review system
- Controlling an App job queue
- Managing a UI session (saving user or app state)
- Storing user metadata
- Storing checkpoint data for modular inputs
- Caching results of splunk queries or from an external data store
- …and more

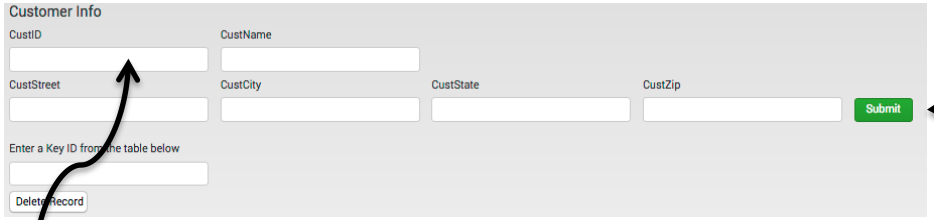
Example App - Demo

splunk>

Example App - Implementation

splunk>

Demo implementation, Create Record




```
var input1 = new TextInput({
    "id": "input1",
    "value": "$form.CustID$",
    "el": $('#input1')
}, {tokens: true}).render();

input1.on("change", function(newValue) {
    FormUtils.handleValueChange(input1);
});
```

```
// Create a service object using the Splunk SDK for JavaScript
// to send REST requests
var service = mvc.createService({ owner: "nobody" });
```

```
var submit = new SubmitButton({
    id: 'submit',
    el: $('#search_btn')
}, {tokens: true}).render();

submit.on("submit", function() {
    submitTokens();

    // When the Submit button is clicked, get all the form fields by accessing token values
    var tokens = mvc.Components.getInstance("default");
    var form_id = tokens.get("CustID");
    var form_name = tokens.get("CustName");
    var form_street = tokens.get("CustStreet");
    var form_city = tokens.get("CustCity");
    var form_state = tokens.get("CustState");
    var form_zip = tokens.get("CustZip");

    // Create a dictionary to store the field names and values
    var record = {
        "CustID": form_id,
        "CustName": form_name,
        "CustStreet": form_street,
        "CustCity": form_city,
        "CustState": form_state,
        "CustZip": form_zip
    };

    // Use the request method to send a REST POST request
    // to the storage/collections/data/{collection}/ endpoint
    service.request(
        "storage/collections/data/mycollection/",
        "POST",
        null,
        null,
        JSON.stringify(record),
        {"Content-Type": "application/json"},
        null).done(function() {
                // Run the search again to update the table
                search1.startSearch();

                // Clear the form fields
                $("#formCustomerInfo input[type=text]").val("");
        });

});
```

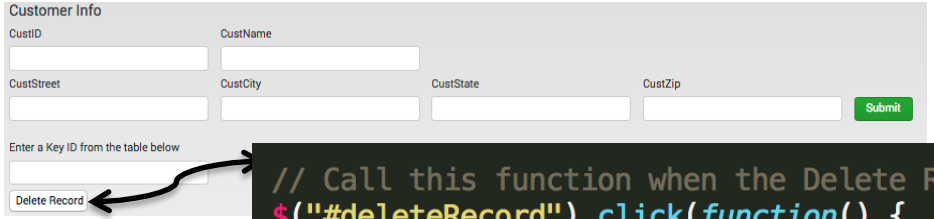
Create Record Details

```
// When the Submit button is clicked, get all the form fields by accessing token values
var tokens = mvc.Components.getInstance("default");
var form_id = tokens.get("CustID");
// GET OTHER FORM FIELDS

// Create a dictionary to store the field names and values
var record = {
    "CustID": form_id,
    "CustName": form_name,
    "CustStreet": form_street,
    "CustCity": form_city,
    "CustState": form_state,
    "CustZip": form_zip
};

// Use the request method to send a REST POST request
// to the storage/collections/data/{collection}/ endpoint
service.request( "storage/collections/data/mycollection/", "POST", null, null, JSON.stringify(record),
                {"Content-Type": "application/json"}, null
                ).done(function() {
                    // Run the search again to update the table
                    search1.startSearch();
                    // Clear the form fields
                    $("#formCustomerInfo input[type=text]").val("");
                });
```

Demo Implementation, Delete Record




```
// Call this function when the Delete Record button is clicked
$("#deleteRecord").click(function() {
    // Get the value of the key ID field
    var tokens = mvc.Components.getInstance("default");
    var form_keyid = tokens.get("KeyID");

    // Delete the record that corresponds to the key ID using
    // the del method to send a DELETE request
    // to the storage/collections/data/{collection}/ endpoint
    service.del("storage/collections/data/mycollection/" + encodeURIComponent(form_keyid))
        .done(function() {
            // Run the search again to update the table
            search1.startSearch();
        });
    return false;
});
```

Configuration & API

splunk>

KVStore Configuration

- Define your collection in `collections.conf`:
  `[mycollection]`
  Optionally, define schema/data types here (not used in demo app)

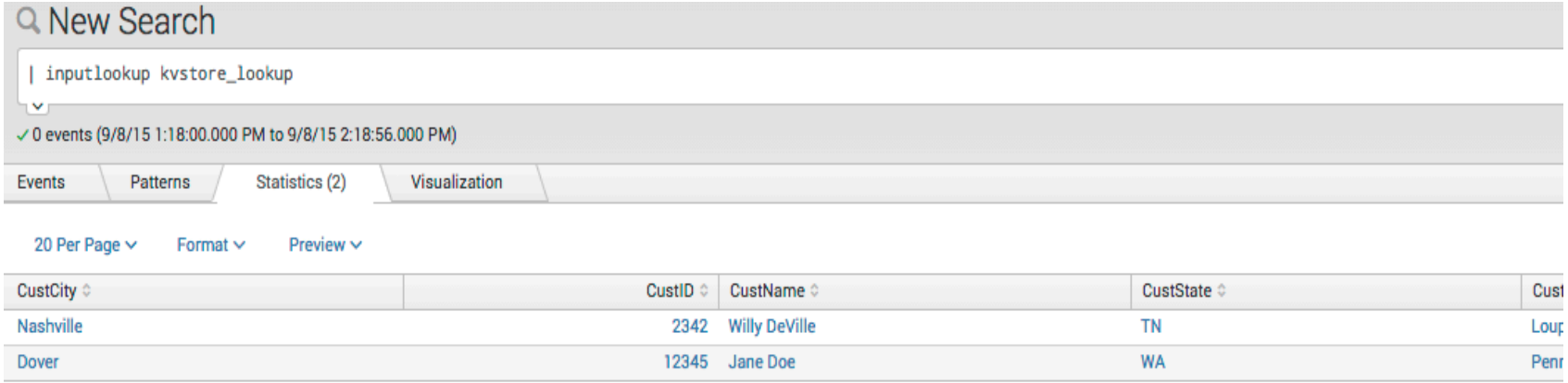
- Define a lookup to use it in `transforms.conf`:
  [kvstore_lookup]
  external_type = kvstore
  collection = mycollection
  fields_list = _key, CustID, CustName, CustStreet, CustState, CustCity, CustZip

KVStore REST Endpoints

- **storage/collections/config** – list all collections
- **storage/collections/config/{collection}** – CRUD collection
- **storage/collections/data/{collection}** – CRUD record(s)
- **storage/collections/data/{collection}/{key}** – CRUD record
- **storage/collections/data/{collection}/batch_save** – Bulk CU

Using SPL

Search: | `inputlookup <lookupname>`



Update: | `inputlookup kvstorecoll_lookup | search _key=544948df3ec32d7a4c1d9755 | eval CustName="Marge Simpson" | eval CustCity="Springfield" | outputlookup kvstorecoll_lookup append=True`

Using CURL

Command line input:

```
$ curl -k -u admin:passwd https://localhost:8089/servicesNS/nobody/
kvstoretutorial/storage/collections/data/mycollection
```

Output:

```
[ { "CustID" : "2342", "CustName" : "Willy DeVille", "CustStreet" : "Loup 29", "CustCity" : "Nashville",
"CustState" : "TN", "CustZip" : "12345", "_user" : "nobody", "_key" : "55c51fa657d58305865af4d1" } ]
```

Summary

splunk>

Summary

- Use it:
  - If you want to store and persist non-event data
  - If you need per-record inserts/updates
  - If you need multi-user access locking
  - If you need to enforce data types
- Don’t use it:
  - As a general replacement for lookup files
  - If you need automatic lookups (not supported today)
  - If you need distributed search-based lookups on the index tier (limited to SH today)

Other Example Uses

- Splunk for Enterprise Security
  - To enable the built-in workflow feature
  - To store notable events
- Splunk for IT Service Intelligence:
  - To store notable events
- Feeling creative already…?
  - Integrate a chat panel into your dashboard(s) for collaboration?
  - Tag/Flag events for later review?
  - ….

Looking Ahead

splunk>

What's New In 6.3 For App KV Store

- Two main KVStore improvements
  - Distributed lookups: Improved efficiency at scale
  - Lookup filtering: Filter lookup results without using a subsequent search command

You May Want To Review…

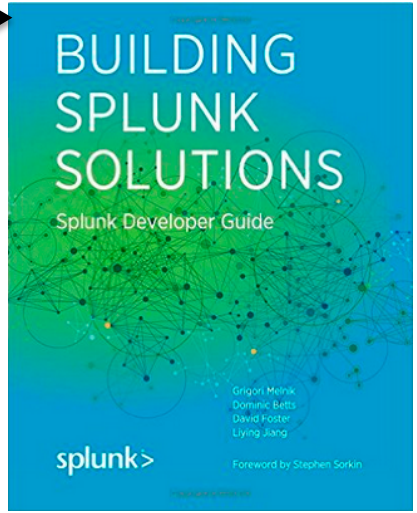
Related breakout sessions and activities…

- Matthew Modestino (TELUS), **Wednesday 10am - 10:45am**
  "Keep your Eyes on the KPIs! …"
  Using KVStore in an app used to do 24x7 NOC alerting integration

- Scott Haskell (Splunk), **Wednesday 11:15am – 12pm**
  "Modular Inputs – If You Build It, They Will Come"
  Using KVStore to keep checkpoint data in a modular input

Further Reading

- Splunk Developer Guidance
  http://dev.splunk.com/view/dev-guide/SP-CAAAE2X
- Splunk Reference App (PAS)
  https://splunkbase.splunk.com/app/1934/
- KVStore Lookups
  http://docs.splunk.com/Documentation/Splunk/latest/Knowledge/ConfigureKVstorelookups
- KVStore REST endpoints
  http://docs.splunk.com/Documentation/Splunk/latest/RESTREF/RESTkvstore
- Monitor the KVStore with the DMC
  http://docs.splunk.com/Documentation/Splunk/6.2.5/Admin/KVStoreInstance
- Splunk Developer Portal
  http://dev.splunk.com
- KVStore tutorial (demo app)
  http://dev.splunk.com/view/webframework-features/SP-CAAAEZT



Questions?

splunk>

THANK YOU

splunk>