

Forward-Looking Statements



This presentation may contain forward-looking statements regarding future events, plans or the expected financial performance of our company, including our expectations regarding our products, technology, strategy, customers, markets, acquisitions and investments. These statements reflect management's current expectations, estimates and assumptions based on the information currently available to us. These forward-looking statements are not guarantees of future performance and involve significant risks, uncertainties and other factors that may cause our actual results, performance or achievements to be materially different from results, performance or achievements expressed or implied by the forward-looking statements contained in this presentation.

For additional information about factors that could cause actual results to differ materially from those described in the forward-looking statements made in this presentation, please refer to our periodic reports and other filings with the SEC, including the risk factors identified in our most recent quarterly reports on Form 10-Q and annual reports on Form 10-K, copies of which may be obtained by visiting the Splunk Investor Relations website at www.investors.splunk.com or the SEC's website at www.sec.gov. The forward-looking statements made in this presentation are made as of the time and date of this presentation. If reviewed after the initial presentation, even if made available by us, on our website or otherwise, it may not contain current or accurate information. We disclaim any obligation to update or revise any forward-looking statement based on new information, future events or otherwise, except as required by applicable law.

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. We undertake no obligation either to develop the features or functionalities described or to include any such feature or functionality in a future release.

Splunk, Splunk>, Data-to-Everything, D2E and Turn Data Into Doing are trademarks and registered trademarks of Splunk Inc. in the United States and other countries. All other brand names, product names or trademarks belong to their respective owners. ©2021 Splunk, Inc. All rights reserved.

How To Build and Scale Services and KPIs Using Service Templates in IT Service Intelligence

ITO1165C

Tony Nesavich

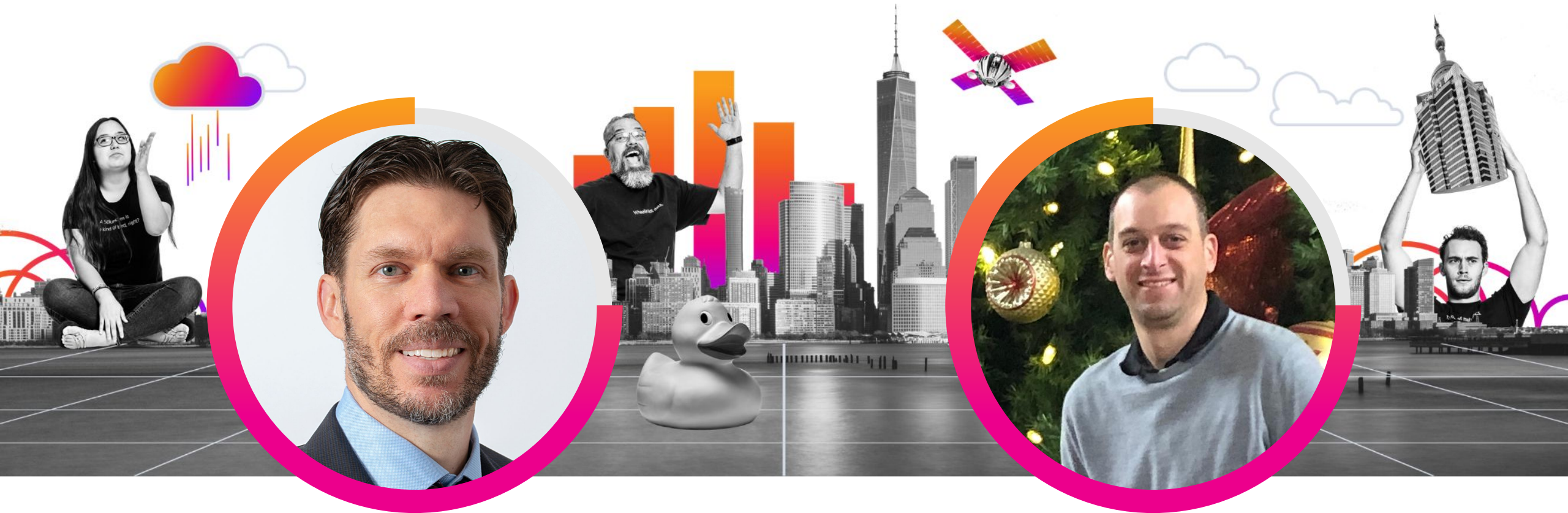
IT Ops & ITSI SME | Splunk

Jeff Wiedemann

IT Ops Strategist & ITSI SME | Splunk

splunk>  .conf21





Tony Nesavich

IT Ops & ITSI SME | Splunk

Jeff Wiedemann

IT Ops Strategist & ITSI SME | Splunk

Disclaimer... Buckle up and focus!

Example code is available for everything we cover today so focus on concepts vs. notes as we will move fast and will be unpacking some cool stuff together!

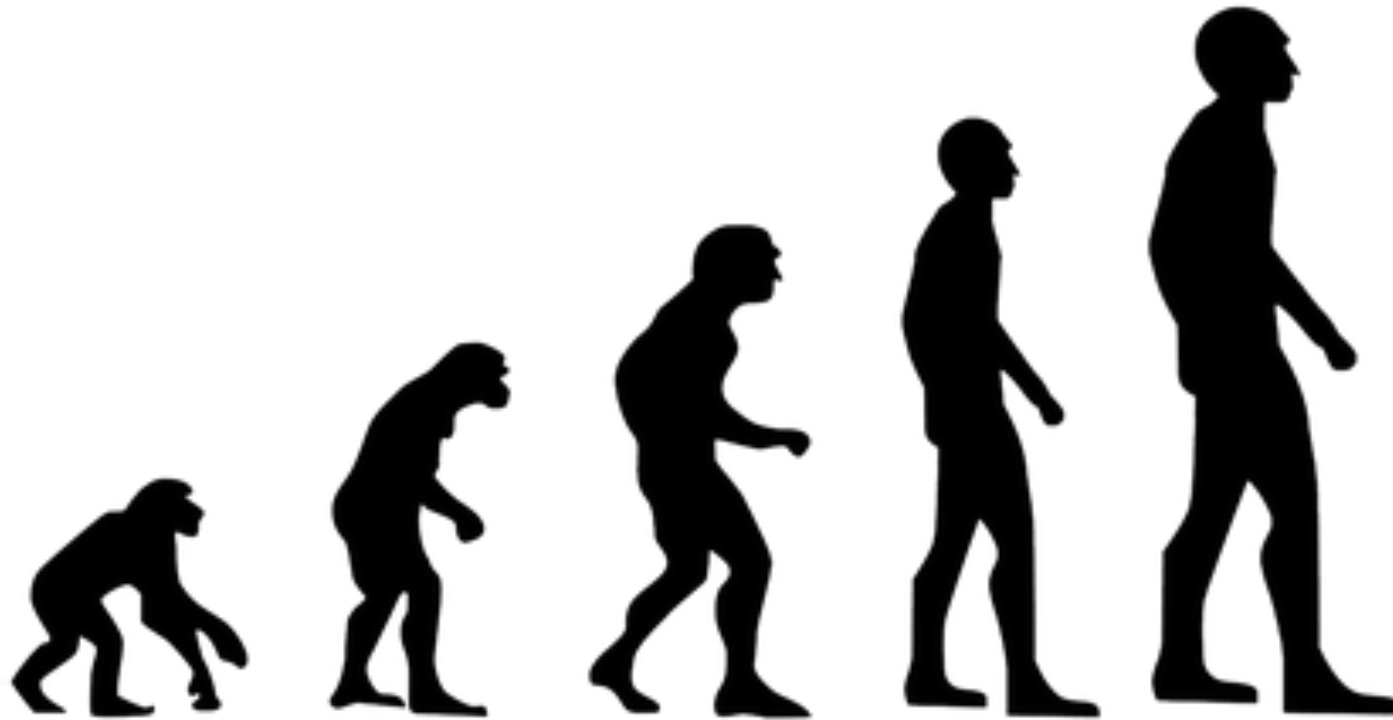


Agenda

- 1) Why automate service tree builds?
Background to example covered in this session
- 2) Laying the groundwork
Requirements for this approach
- 3) Walkthrough
- 4) Best practices
Covered throughout & relevant



Origins of approach / Why



See it in action!



Is this right for you?

Are you ready to follow this process?

- You have a large and uniform service tree where many nodes look about the same

Is this right for you?

Are you ready to follow this process?

- You have a large and uniform service tree where many nodes look about the same
- You are trying to ensure your service tree is "mirrored" in another system (CMDB)

Is this right for you?

Are you ready to follow this process?

- You have a large and uniform service tree where many nodes look about the same
- You are trying to ensure your service tree is "mirrored" in another system (CMDB)
- Your service tree should ideally expand and shrink as things change in the environment

Is this right for you?

Are you ready to follow this process?

- You have a large and uniform service tree where many nodes look about the same
- You are trying to ensure your service tree is "mirrored" in another system (CMDB)
- Your service tree should ideally expand and shrink as things change in the environment
- You can "see" the tree structure within the raw data. Hierarchies, relationships, etc.

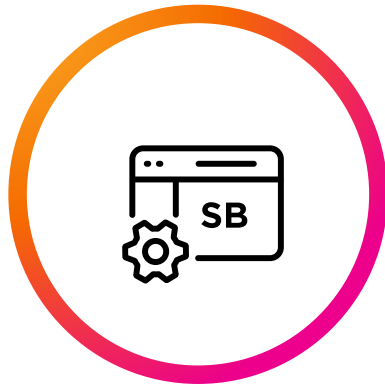
Foundations we build on

Structure within the data



Define service trees using data

Entities



Critical to how KPI results are filtered and divided within the service tree

Service Templates



Support performant, scalable, dynamic management of the service tree

Automatic Service Creation



Automatically build service tree and schedule recurring build

Foundations we build on

Structure



Structure

Structure

- Anything with reliable definitions/tagging to build off

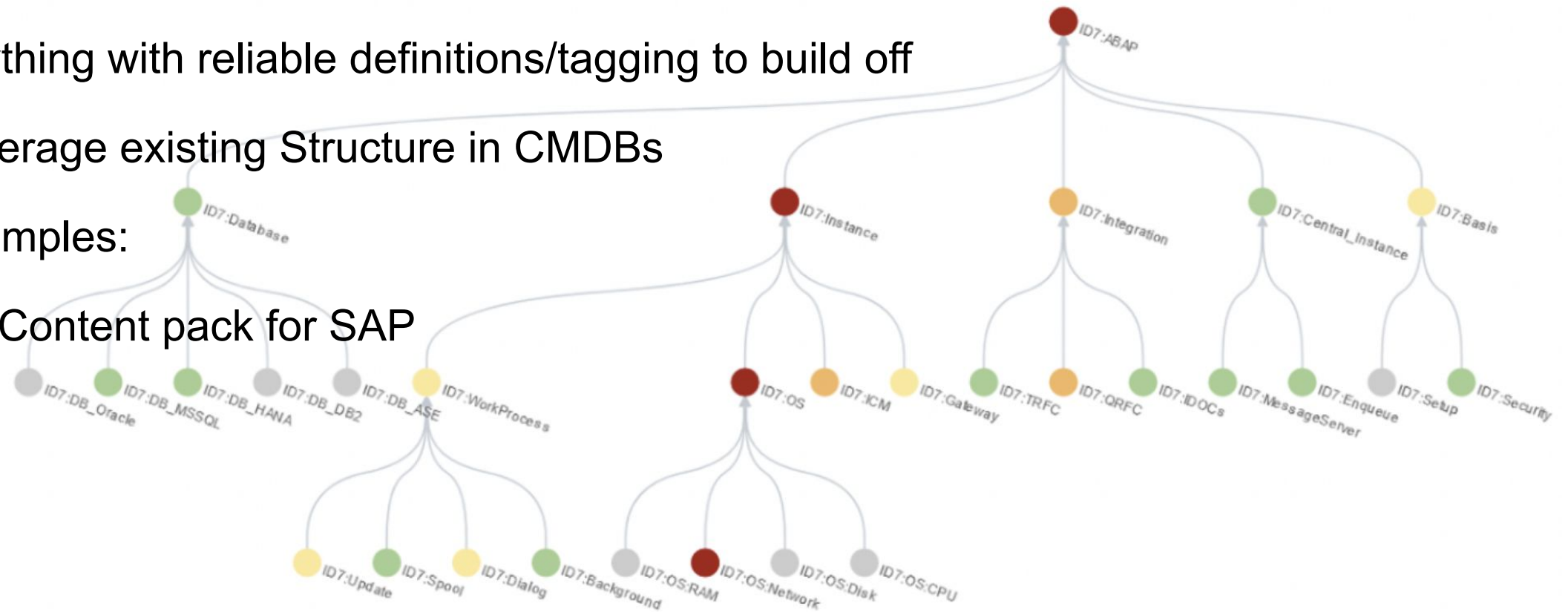
Structure

- Anything with reliable definitions/tagging to build off
- Leverage existing structure in CMDBs

Structure

- Anything with reliable definitions/tagging to build off
- Leverage existing Structure in CMDBs
- Examples:

- Content pack for SAP

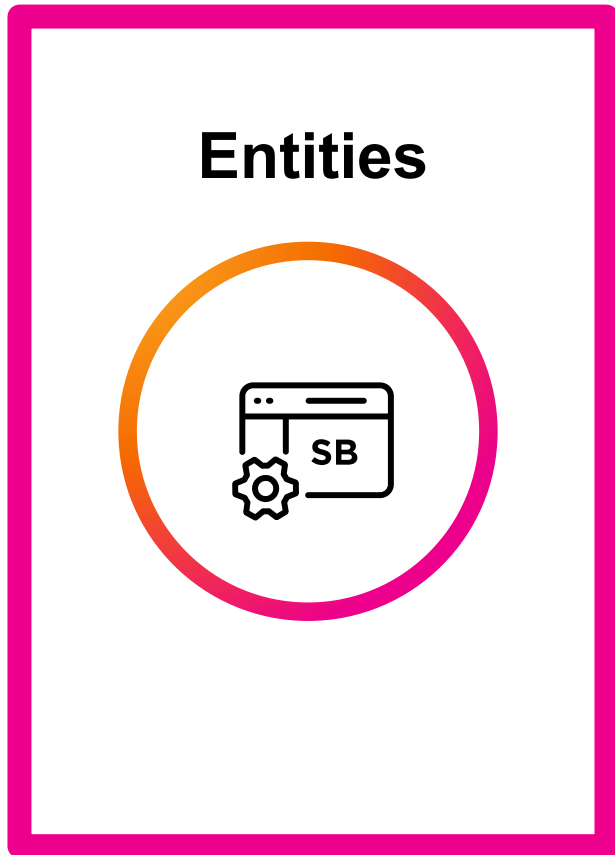


Foundations we build on

Structure



Entities

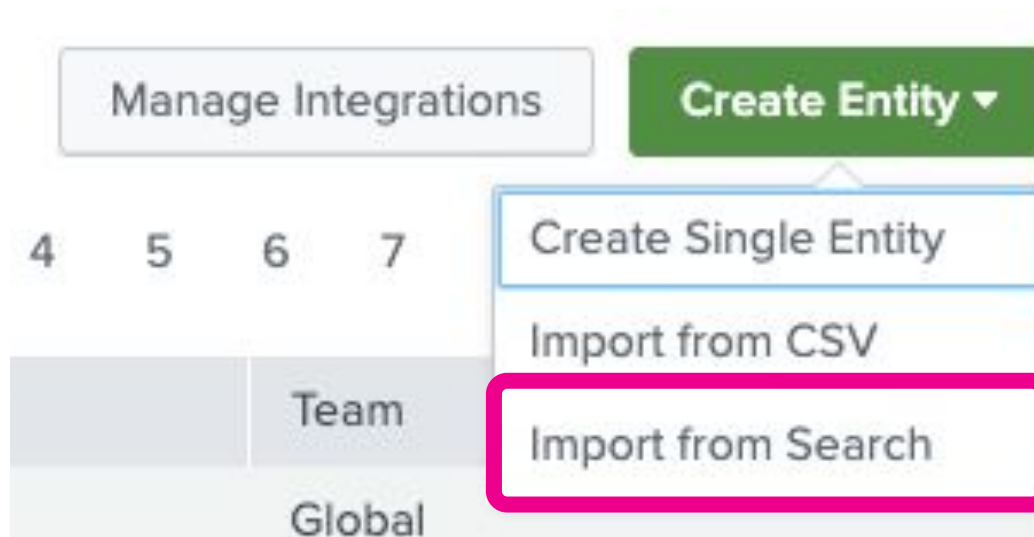


Entities

Entities

Best Practice

- Import them from search!



Entity metadata is critical. Get it right!

Entity Title - *Crucial for duplicate entity detection and merging*. Keep it simple and intuitive. Watch out for duplication. (hostname vs fqdn)

Entity Alias Fields - *Should always match a unique field in the raw data* used to drive KPI results filtering. *Required* for automatic service tree creation. (Examples: host, fqdn, ip_address, instance_id, moid)

Entity Info Fields - Drives entity filtering rules in the Service. *Required* for automatic service tree creation. (Examples: OSName, ApplicationName, Region, Client)

Schedule recurring imports

Entities

Search Based Entity Import Example

```
<ENTITY-DATA-SEARCH>
```

```
| eval entity_title=<PRIMARY-ALIAS-FIELD>
| dedup entity_title

| eval entity_type="<OPTIONAL-ENTITY-TYPE>"
| eval entity_type_info=entity_type

| table entity_title <PRIMARY-ALIAS-FIELD> <ALIAS-2> <ALIAS-N> <INFO-1> <INFO-2> <INFO-N> entity_type_info entity_type
```

Search raw data where entities can be discovered. Relevant field extractions required.

Entities

Search Based Entity Import Example

<ENTITY-DATA-SEARCH>

```
| eval entity_title=<PRIMARY-ALIAS-FIELD>
| dedup entity_title

| eval entity_type="<OPTIONAL-ENTITY-TYPE>"
| eval entity_type_info=entity_type

| table entity_title <PRIMARY-ALIAS-FIELD> <ALIAS-2> <ALIAS-N> <INFO-1> <INFO-2> <INFO-N> entity_type_info entity_type
```

Wisely choose an entity title. Deduplicate results by entity_title for most recent entity information.

Entities

Search Based Entity Import Example

```
<ENTITY-DATA-SEARCH>
```

```
| eval entity_title=<PRIMARY-ALIAS-FIELD>
| dedup entity_title
```

```
| eval entity_type="<OPTIONAL-ENTITY-TYPE>"
| eval entity_type_info=entity_type
```

```
| table entity_title <PRIMARY-ALIAS-FIELD> <ALIAS-2> <ALIAS-N> <INFO-1> <INFO-2> <INFO-N> entity_type_info entity_type
```

Specify the entity type. Optional, but recommended. Facilitates filtering and navigation in ITSI.

Entities

Search Based Entity Import Example

<ENTITY-DATA-SEARCH>

```
| eval entity_title=<PRIMARY-ALIAS-FIELD>
| dedup entity_title
```

```
| eval entity_type="<OPTIONAL-ENTITY-TYPE>"
| eval entity_type_info=entity_type
```

```
| table entity_title <PRIMARY-ALIAS-FIELD> <ALIAS-2> <ALIAS-N> <INFO-1> <INFO-2> <INFO-N> entity_type_info entity_type
```

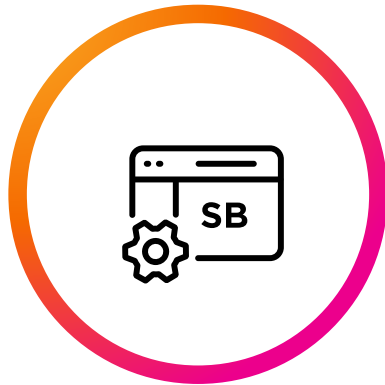
Tidy up the final results for readability and easier importing

Foundations we build on

Structure



Entities



**Service
Templates &
Base Searches**



Service Templates

Best practice for migrating a manually built service to a service template

Build Service Templates

Best practice for migrating a manually built service to a service template

Build manual services first, and model Service Templates from them

- Simplifies the build and troubleshooting process
- Allows for iterative QA

Build Service Templates

Best practice for migrating a manually built service to a service template

Build manual services first, and model Service Templates from them

Ensure entity filtering rules on the service are dynamic

- Service must have entities defined
- Entity filtering must use entity info fields (not entity title or alias fields unless wildcarded)

Build Service Templates

Best practice for migrating a manually built service to a service template

Build manual services first, and model Service Templates from them

Ensure entity filtering rules on the service are dynamic

Must use “filter to entities in service” configuration field

- Filters result set to just entities matching the entity alias in the raw data
- Allows for base search to function without specifying entities
- [See Docs](#) for more information

Build Service Templates

Best practice for migrating a manually built service to a service template

Build manual services first, and model Service Templates from them

Ensure entity filtering rules on the service are dynamic

Must use “filter to entities in service” configuration field

Refactor ad-hoc KPIs into KPI base searches

- To optimize search performance - (Think **Big** and Small)

Build Service Templates

Best practice for migrating a manually built service to a service template

Build manual services first, and model Service Templates from them

Ensure entity filtering rules on the service are dynamic

Must use “filter to entities in service” configuration field

Refactor ad-hoc KPIs into KPI base searches

Test and validate the Service and KPIs look correct in the Service Analyzer

Yay! Now create service template from service!



Create Template from Service

Best practice for migrating a manually built service to a service template

A service is a collection of KPIs and entities that represent a real-world IT service.

69 Services Bulk Action ▾

<input type="checkbox"/>	i	Name ▲	Actions	Status
<input type="checkbox"/>	>	AT Conf Talk Service	Edit ▾	<input checked="" type="checkbox"/> Enab
<input type="checkbox"/>	>	AWS	Edit	Disat
<input type="checkbox"/>	>	AWS EC2	Edit Team	Disat
<input type="checkbox"/>	>	AWS Lambda	Link to Service Template	Disat
<input type="checkbox"/>	>	Azure	Create Service Template	Disat
<input type="checkbox"/>	>	Azure Functions	Put Service in Maintenance Mode	Disat
<input type="checkbox"/>	>	Azure VM	Clone	Disat
<input type="checkbox"/>	>	Cloud	Delete	Disat

IT Service Intelligence Starter Pack - Example Service Template ✎

An example Service Template that uses all of Splunk's best practice recommendations for service templates for you to

Entities KPIs Settings Linked Services

Specify entity rules to dynamically filter KPIs. Entity rules are optional. When defining rules you can specify the field val

If you will be creating services in bulk, you can use the **matches a value to be defined in the service** and **does not ma**

matches a value to be ... ▾

+ Add Rule (AND)

+ Add Set of Rules (OR)

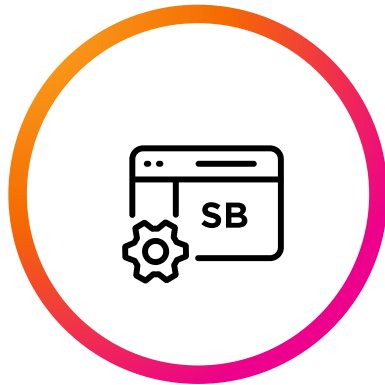
- matches
- does not match
- matches a value to be defined in the service
- does not match a value to be defined in the service

Foundations we build on

Structure



Entities



**Service
Templates &
Base Searches**



**Automatic
Service Tree
Build**



Automatic Service Tree Build

Search Based Entity Import Example

```
| `itsi_starter_pack_get_example_entity_data`
```

```
| eval subtree_namespace="foo:"
```

```
| eval is_raw=1
```

Bring in the data (will depend on use case)

Automatic Service Tree Build

Search Based Entity Import Example

```
| eval comment="Build the top level service first."
```

```
    service_name should be populated with the name you want to give the top level of the tree.
```

```
    dependent_service_name should be populated with the second tier service names"
```

```
| appendpipe
  [ stats values(subtree_namespace) as subtree_namespace count by region
  | eval service_name = "Starter Pack Example Tree"
  | eval dependent_service_name = subtree_namespace.region
  | eval template=""
  | eval entity_info_INFO-FIELD1=null(), entity_info_INFO-FIELD2=null() ]
```

Define service_name, dependent services, service templates, & entity_info

Automatic Service Tree Build

Search Based Entity Import Example

```
| eval comment="Build the second tier services next.
```

```
    service_name should match the value chosen for dependent_service_name above.
```

```
    dependent_service_name should be populated with the third tier service names"
```

```
| appendpipe
```

```
  [ stats values(subtree_namespace) as subtree_namespace count by region
```

```
  | eval service_name = subtree_namespace.region
```

```
  | eval dependent_service_name = ""
```

```
  | eval template="IT Service Intelligence Starter Pack - Example Service Template"
```

```
  | eval entity_info_region=region, entity_info_INFO-FIELD2=null() ]
```

Define service_name, dependent services, service templates, & entity_info

Automatic Service Tree Build

Search Based Entity Import Example

```
| search (NOT is_raw=1)
| rename service_name as "Service Title", dependent_service_name as "Dependent Services", template as
"Service Template Link"
| table "Service Title" "Dependent Services" "Service Template Link" entity_info_*
```

Tidy up the final results for readability and easier importing

See it in action!



Breaking it down

Simple to follow steps to convert a manual tree into an automatic tree

1. Create Entities from search.

Breaking it down

Simple to follow steps to convert a manual tree into an automatic tree

1. Create Entities from search.
2. Manually build a representative sample of your service tree

Breaking it down

Simple to follow steps to convert a manual tree into an automatic tree

1. Create Entities from search.
2. Manually build a representative sample of your service tree
3. Configure entity filtering rules for each service

Breaking it down

Simple to follow steps to convert a manual tree into an automatic tree

1. Create Entities from search.
2. Manually build a representative sample of your service tree
3. Configure entity filtering rules for each service
4. Convert KPIs into base searches which use entity filtering rules

Breaking it down

Simple to follow steps to convert a manual tree into an automatic tree

1. Create Entities from search.
2. Manually build a representative sample of your service tree
3. Configure entity filtering rules for each service
4. Convert KPIs into base searches which use entity filtering rules
5. Convert services into service templates and update entity filtering rules to be defined in service template

Breaking it down

Simple to follow steps to convert a manual tree into an automatic tree

1. Create Entities from search.
2. Manually build a representative sample of your service tree
3. Configure entity filtering rules for each service
4. Convert KPIs into base searches which use entity filtering rules
5. Convert services into service templates and update entity filtering rules to be defined in service template
6. Build the service tree SPL that programmatically defines the tree

Breaking it down

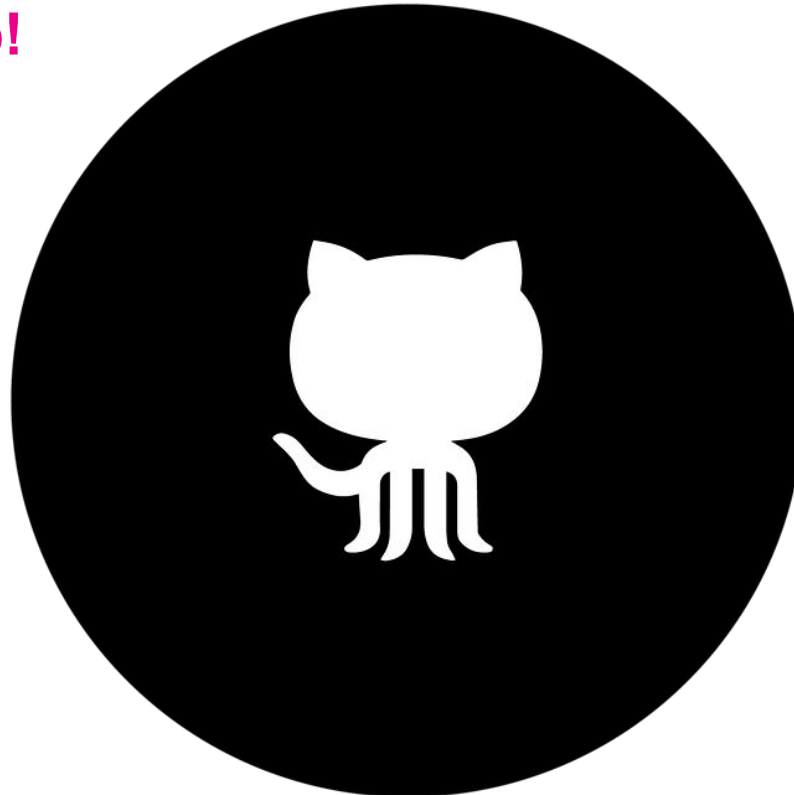
Simple to follow steps to convert a manual tree into an automatic tree

1. Create Entities from search.
2. Manually build a representative sample of your service tree
3. Configure entity filtering rules for each service
4. Convert KPIs into base searches which use entity filtering rules
5. Convert services into service templates and update entity filtering rules to be defined in service template
6. Build the service tree SPL that programmatically defines the tree
7. Create the services automatically using create service from search. Schedule to run as needed

Call to action

Time Savings & next steps, how to do this on your own

More examples on GitHub!



Call to action

Time Savings & next steps, how to do this on your own

github.com/splunk/itsi-cp-starter-pack

More examples on GitHub!



Thank You

Please provide feedback via the

SESSION SURVEY

