

ITRS Group Ltd.

# ITRS INSIGHTS V1.4

## TITLE USER GUIDE

Copyright 2016 ITRS Group Ltd. All rights reserved.

Information in this document is subject to change without notice. The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of those agreements. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or any means electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's personal use without the written permission of ITRS Group Ltd.

ITRS Group Ltd.

6th Floor, The Bonhill Building, 15 Bonhill Street,  
London, EC2A 4DN, UK

t: +44 (0)20 7638 6700

f: +44 (0)20 7256 5760

e: [europa@itrsgroup.com](mailto:europa@itrsgroup.com)

# CONTENTS

## CHAPTER 1

ITRS Insights Query, Search and Transform Data User Guide ..... 6

APPENDIX ..... 15

Organizing Data in ITRS Insights ..... 16

GLOSSARY ..... 27

INDEX ..... 29

**This page intentionally left blank to ensure**



## ITRS INSIGHTS QUERY, SEARCH AND TRANSFORM DATA USER GUIDE

### Query Data

Every query starts with a from followed by the entity that you are querying

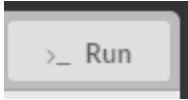
from <stream\_name>

If you are querying historical data (already in the system) then you also need the operator `historical` in front of the stream name. If you are using real-time data streaming in then nothing is needed. For example:


```
from /streams/hce/electricity/usage
```

or

```
from historical /streams/demo/geneos/cpu
```

Executing just this (by hitting the run button)  will give you a table of all the contents of the stream `/streams/demo/geneos/cpu`

Your stream\_names may be complex based on your structure so you can easily copy a stream name to the clip board from your home page

From the data streams section of your homepage if you hover over the  icon then you will get the option to copy to clipboard

## Streams of data by collection

 houseprice

 uk 

You can then just paste (CTRL-v) into the query box.

---

That is however the most basic of queries a great deal more can be constructed using the SQL like language that Insights provides. Below is a table of the available syntax **selecting one in the left column** will take you a more detailed description including the form and some examples.

from	Specifies the stream that will be the source of data for the query.	FROM /streams/demo/geneos/cpu
Where	Filters data based on a condition which follows.	FROM /streams/demo/geneos/cpu WHERE user > 50
Select	Selects which fields should appear in the results.	FROM /streams/demo/geneos/cpu SELECT host, user, kernel
Aggregation	Perform aggregate calculations on the data.	FROM HISTORICAL /streams/demo/geneos/cpu SELECT avg(user) AS user, avg(kernel) AS kernel
Take	Limits the Return only a certain number of results.	FROM /streams/demo/geneos/cpu TAKE 100
Order By	Orders results in ascending or descending order.	from historical /streams/demo/geneos/cpu order by host ascending
Group By	Group and aggregate results into distinct buckets.	FROM /streams/demo/geneos/cpu GROUP BY host SELECT host, avg(user) AS user
Time Window	Group and aggregate into windows based on a time field in the data.	FROM /streams/demo/geneos/cpu WINDOW OF 3 seconds SELECT start, end, avg(user) as user
As	Applies an alias to an output field	FROM /streams/demo/geneos/cpu WHERE user > 50 AS cpu_data
Into	Gives the query a name so it can be referenced from other queries.	FROM /streams/demo/geneos/cpu WHERE user > 50 INTO result
Unique	Prevents duplicate rows from appearing in the results.	FROM /streams/demo/geneos/cpu SELECT UNIQUE host
Join	Joins two streams, including over a time window.	FROM /streams/demo/geneos/cpu WHERE user > 50 INTO c INNER JOIN r ON c.host == r.host
Comment	Allows descriptive text to be added to a query.	FROM /streams/demo/geneos/cpu TAKE 10 -- This is a comment!
Fuzzy	Enables search within a query for SSR Only	FROM /streams/demo/infrastructure/log FUZZY "My_Search_Text*"

More information on field identifiers, formats and types can be found here: [Field Identifiers and Formats](#).

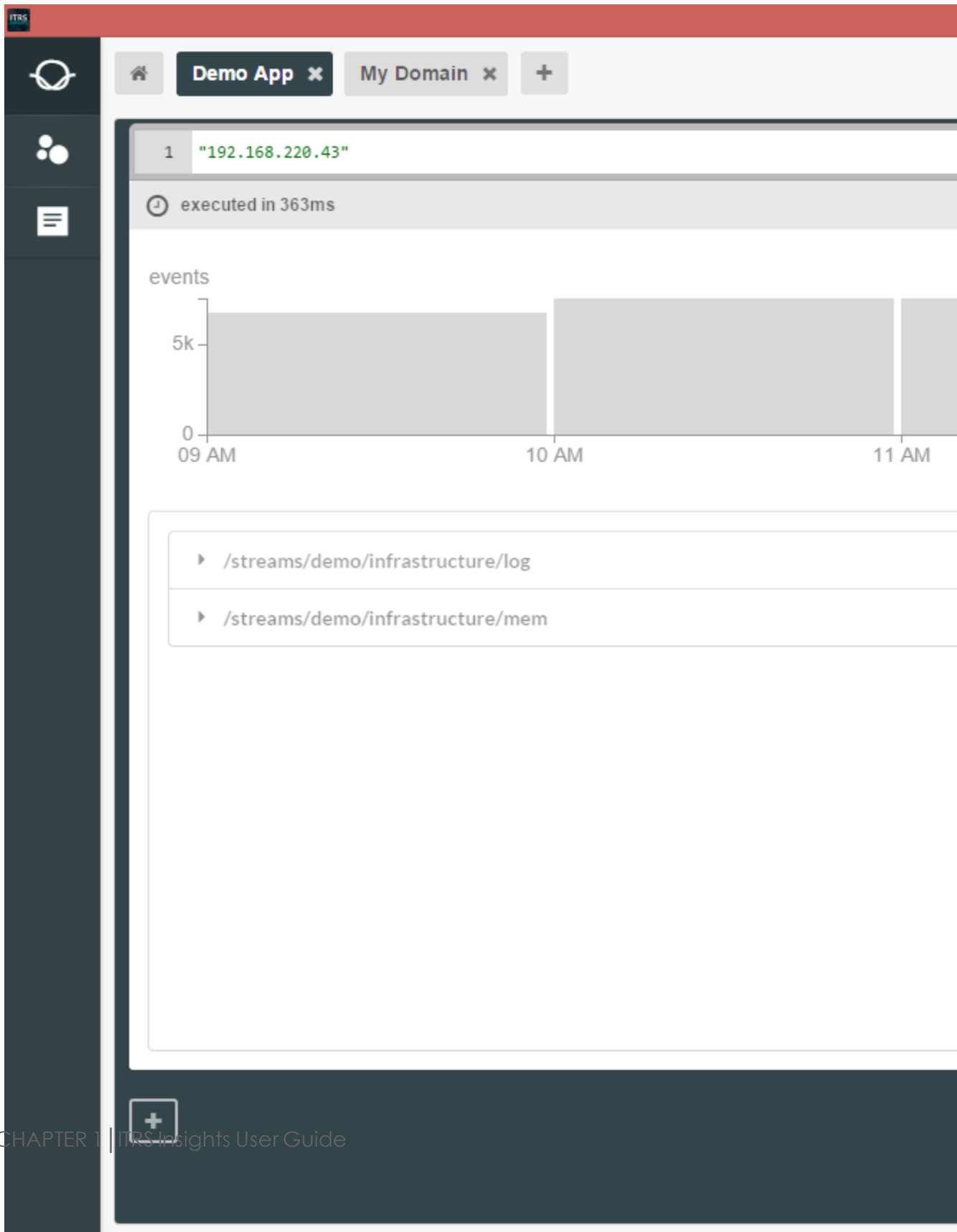
Search Data

Free Text Search

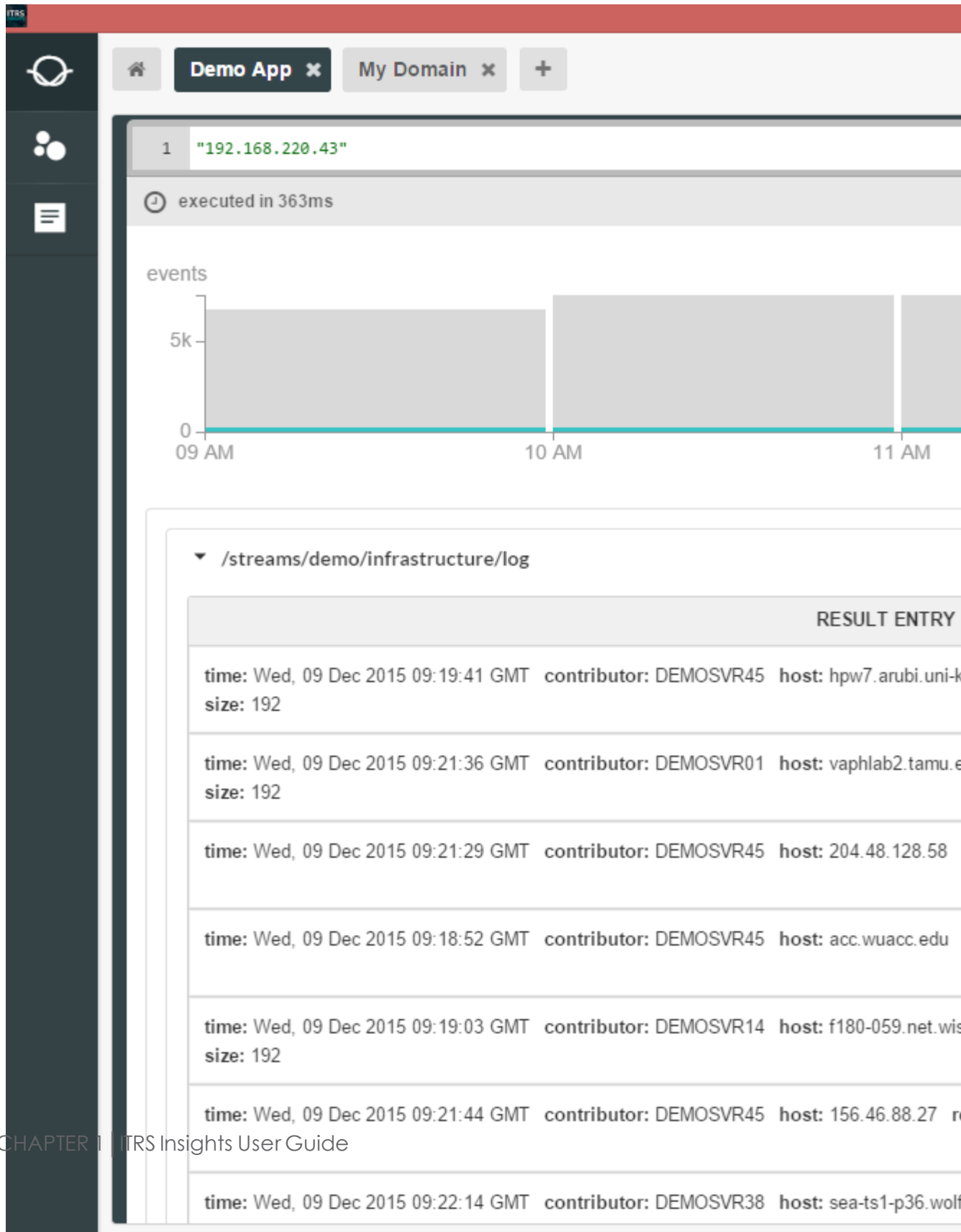
The search capability in Insights allows you to do free text search, similar to an internet search engine it search the contents of the stream for a particular value or string that you give it. Because of how searches work and require structured data **You can only do searches on data in the semi-structured repository (SSR).**

The search results are displayed on return, if there is more that one stream in which they are found then the streams are show in sections. For example searching for host "192.168.220.43" in a collection of streams from ITRS Geneos returns:





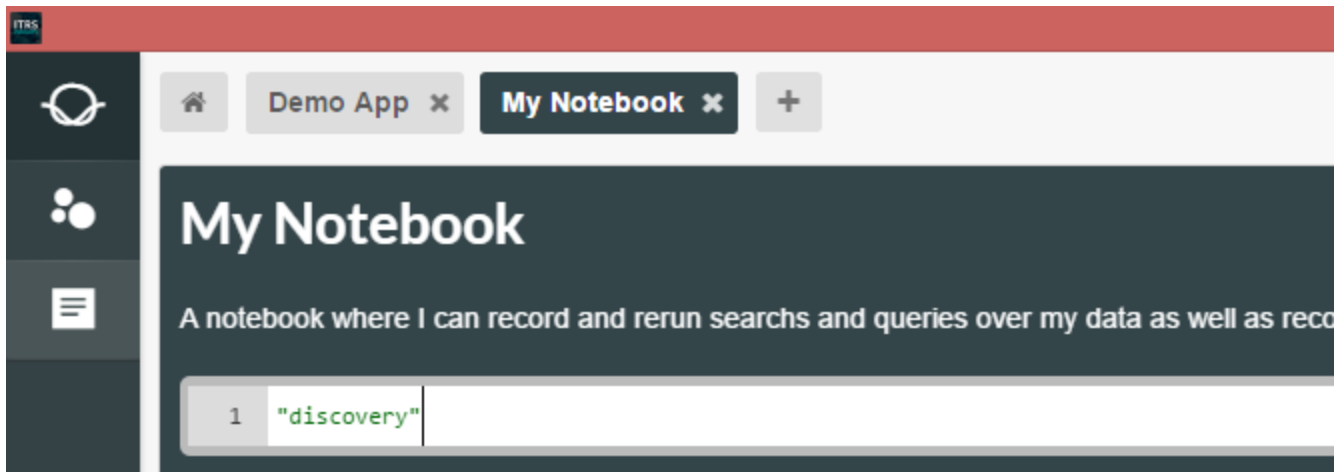
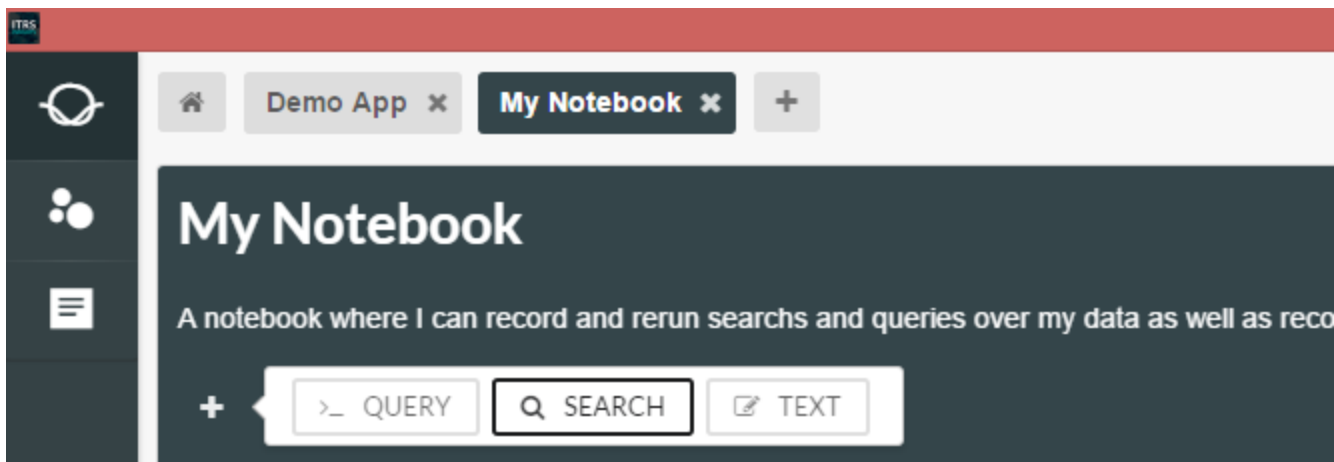
In the main panel are displayed the 2 streams in which it is found and on the right hand side is the taxonomy. You can use this to select a subset of the results that match, doing this highlights the proportions on the histogram at the top. So for example if we select `Status code 200` we see just those entries on the left and on the histogram above we can see 253 entries from 12 to 1pm (the number is shown when hovering over the bar).



You can use the wildcard ``*`` in your search, but the search must **\*not\*** start with it.

For more information, see: ITRS Insights Dashboard & Notebook User Guide.

Selected the + that is at the top of the window and select the `search` option



Type in the text you wish to search for within quotes " " and hit `Search`.  
If you want to time limit the boundaries of the search then you can add a start (first date box) and end (second date box) time and date for the search.

#### Relevance score

A search returns the results sorted by the relevance score. Currently, the score is calculated using the local index statistics which could result in skewed results if the data is not uniformly distributed in the cluster. In the future we will support scoring based on the global index statistics.

#### Search in queries

The free text search can also be used in real-time and historical queries using the **fuzzy** keyword:

```
from /streams/demo/infrastructure/log  
fuzzy "gif"
```

#### The difference between search and query

Search is basically looking through the data and searching for some specific text (be it characters, numbers etc.) and returning to you all the instances where this occurs. A search does not attempt in anyway to interpret or know the data, just to look at it and match an expression.

A query on the other hand seeks to in some way understand and work with the data, so for example in its most basic form a query can look into data to find instances of a number that is greater than 5. To do this requires that the system understands what is a number (and what is not) and the comparative order of the numbers, this is why you can only run queries on data that has a schema. However that is just the simplistic of queries, through that manipulation and comparison of the data you can process and analyse the data in any number of ways. Insights provides you, as well as the basic manipulation functions, with easy and quick access to complex and sophisticated algorithms for things such as anomaly detection and Univariate analysis.

## Transform Data

The data store in Insights is deliberately immutable so the data from the original stream can **not** be changed. However it is easy to change and transform data by using any of the Query Functions creating a derived stream from the original, whether that original is stored or not.

So for instance if I have all the flight data for US flights but I want a stream that has only the airline with the carrier code "AA". I can issue

```
From historical /streams/demo/usa/flightdata  
where UniqueCarrier == "AA"  
into myNewStream
```

or with a Geneos stream

```
FROM HISTORICAL /streams/demo/geneos/cpu WHERE max(user, kernel) > 50
```

# APPENDIX

## APPENDIX A - TITLE

Content

## ORGANIZING DATA IN ITRS INSIGHTS

### Introduction

To help you get the most from Insights, it's important to understand a little about the organizational concepts that underpin it.

Because Insights' query tool acts like a pipeline - you can build successive queries to be applied to one set of results in order to iteratively pare down the data - categorizing where your data is coming from, and defining the hierarchy of who owns it and what type it is, will make it much easier for you to query it later on.

The good news is that there is a limited number of hierarchical levels available - three main ones and two sub types - so it won't take you long to read this document and then get on with the much more exciting prospect of actually using Insights.

And don't forget, when you're finished here, check out some of our user guides:

- "ITRS Insights Query, Search and Transform Data User Guide" on page 6

- ITRS Insights Domain & Contributors User Guide

- ITRS Insights Dashboard & Notebook User Guide

- ITRS Insights Template User Guide

Cities are getting smarter.

For the purposes of illustrating how we organize data in Insights, we're going to imagine a smart city...



A key part of smart cities will be establishing the right infrastructure to collect and intelligently analyze streams of data from connected 'things'. This is where Insights and Valo come in.

To enable you to make sense of your data and to really leverage Insights' ability to query it, we've implemented a straightforward hierarchy and organizational structure for it:

**Streams** and **Contributors** > **Domains** > **Tenants**

Now, let's look at how each of these works in action.

Stream

Before you can query or search your data, you have to actually get it into Insights. You do this through data **streams**.

**Stream**: append-only data (messages or events) coming from one or more external **Contributors**. Each stream has a **schema** that defines what information is expected, and in what format. Multiple streams can be organized into **Collections**.



#### Example Stream

Arcadia Avenue a street of 20 houses all with the latest smart electricity meters, all sending information to their energy supplier HighCostEnergy.com (HCE) through an underground cable.

In the village center, HCE has a hub to transmit and process this information. Our street has a single cable running down it to the hub, but within that cable are thousands of fiber optic cables all taking information from the energy consuming devices in the house (refrigerator, toaster, TV, coffee machine, boiler, etc.).

In this example, the cable contains the stream of data that is passing to the hub, but that stream is made up of all the data coming from the individual fibre optic cables, thus **the stream of data is made up of the data from all the individual contributor devices**: the toaster at house Number 2, the refrigerator at Number 6, the coffee machine at Number 13, and so on.

One challenge when you have a number of different contributors is to **ensure that everyone is speaking the same language and meaning the same thing**, otherwise comparisons and analytics will have limited value. To do this, **we use a document of understanding (a schema) that defines what data the stream contains in what format**. For instance, we may want the stream to provide information as data points (also known as payloads) about the time of day, the type of appliance, and the current consumption.

Because the schema ensures that the data flowing in is in a common format, we can then meaningfully compare and analyze it based on those common attribute values. We can analyze which refrigerator consumes the most energy and because our schema clearly detailed to format that the time would come in. We can compare all the appliances that are on at 8am in each house without fear that some payloads

included with a time stamp of 8 would in fact be for 8pm. We can analyze pretty much anything we want, in fact.

Now for the vaguely technical bit:

Each stream has a unique name (a Unique Resource Identifier - URI) that allows us to easily categorize and identify it. In Insights, therefore, our smart house stream might be called something like `/streams/hce/electricity/usage`

This URI has the form:

```
/stream-  
s/<tenant>/<collection>/<streamName>
```

Where:

`tenant` = the organizational unit the stream belongs to  
(see: "Tenants" on page 24)

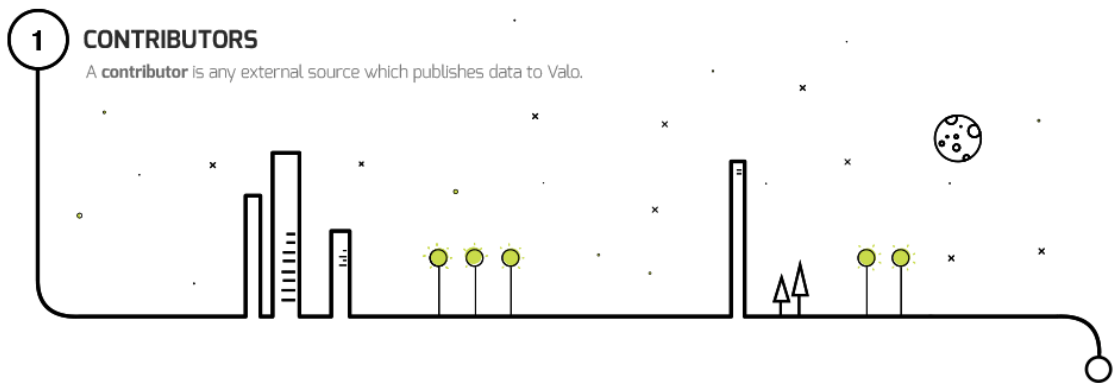
`collection` = the collection the stream is contained in

`streamName` = the name of the stream

Contributor

With our streams added, we can define who the external sources are. We do so with **contributors**.

**Contributor:** anything that generates and publishes data to Valo or Insights. A probe or an appare example contributors. Contributors of the same type have common characteristics but can also have different metadata to one another. For example, a probe contributor on a Windows server might be able to provide more metadata than probe contributor on a Linux server. Contributors can be grouped together by common attributes to create **Domains**.



Example Contributor

**We use the concept of contributors to assign metadata to each data source.** For example, if the source is a sensor in a heating system, we might categorize it based on what type of heater it's in, who owns it, who it was manufactured by, or who installed it. You get the idea.

Like streams, each contributor has a unique URI.

If we imagine that the electric kettle in the kitchen at house Number 4 contained a probe, it could have a URI along the lines of `/contributors/hce/Num4/kettle`, for example.

The contributor URI has the form:

```
/contributors/<tenant>/<type>
```

Where:

`tenant` = the organizational unit the stream belongs to

`type` = the type of contributor (for example probe or app)

#### Domain

Now that we've got our stream(s) and contributor(s) categorized, we can further define our data sources.

Creating domains has several distinct advantages. Firstly, they're dynamic - they're an attribute of a contributor (as opposed to a flag), so create a domain and it'll automatically detect any new contributors that come into it. Secondly, domains provide another layer of granularity, control and time-saving when it comes to actually analyzing the data.

So, let's group our contributors together to form **Domains**.

**Domain:** two or more contributors grouped together by a common attribute. Domains help to organize and group different data sources in intuitive ways.

#### Example Domain

In our street of smart metered houses, **we could define house Number 6 as a domain, thus every smart device that is identified as being in house number 6 is considered part of that domain for the purposes of our analysis.**

But a domain doesn't have to be a locational entity. We could define a domain of TVs on Arcadia Avenue, thus every contributing device that is identified as a TV that is within our street is considered part of our domain for analysis.

A powerful attribute of Insights' domains is that because the domain is defined by an attribute of the contributor rather than a flag that has to be set manually at installation, domains can be dynamic, automatically picking up new contributors coming into the domain.

So for instance if Mr Smith in No2 Arcadia Avenue goes out and buys a new TV for his teenage son to have in his bedroom then once the TV is plugged into the system it is automatically included in the Arcadia Ave TV domain.

Having domains also means you can have higher level metrics or data that relate to that domain which can be published, the value of this is you can publish this higher level information to people and systems who are interested in the domain but for instance do not have permission to see all the details of the

data within the domain. Again looking at our example Mr Smith could see the energy consumption of all the TVs in his house, but without having access to the energy consumption (and thus potentially the hours of watching!) that came from the new TV in his son's bedroom. Or perhaps HCE could have access to the energy consumption of each house within the street without being allowed access to the consumption of the individual devices in each house, which might be considered somewhat invasive.

A domain is used at query time to filter a stream to only include data entries for contributors and tags defined by the domain. The filter is dynamic, so a contributor which no longer matches the query will be automatically removed from the domain.

A domain definition is identified by a unique URI in the following format

```
/domain/tenant/collection/name
```

Tenants

Still with us? Great. Only two more steps to go.

Before we can complete the organization of the data come from our smart city into Insights, we're going to define the highest level of organizational attribute: **Tenants**.



**Tenant:** the highest level of data grouping in Insights and Valo. Within a company, it may be different departments such as Sales or Engineering. If a government is collecting data from its cities, the tenant could be the city name. Tenants can be further subcategorized as **Collections**.

#### Tenants Example

**A tenant is the highest level of grouping. It helps us to understand who owns the data and, if we so wish, to place virtual walls around the data limiting visibility and access.**

In our smart house example, we could say that **HCE is a tenant as it is the organization to which the stream of electricity consumption data goes:** the stream name is:

```
/streams/hce/electricity/usage.
```

100 miles away in another area of the country, there maybe another street of smart houses called Privet Drive, this road is supplied by the local company Really Expensive Electricity (REE). In this instance we may call the stream:

```
/streams/ree/electricity/usage, where REE is the tenant.
```

You can assign tenants within a company, perhaps to different departments such as Sales or Engineering. For big businesses,

you may define tenants by the city or office they're located in. The bottom line is that you can organize your data how you wish.

#### Collections

Finally, we can further logically group streams inside a tenant using **Collections**.

**Collections:** two or more streams grouped together inside a tenant.

#### Example Collection

**Inside a collection are the names of the individual streams.**

In our smart metered houses example, a collection might consist of two streams: one from Arcadia Avenue, and the other from Privet Drive.





# INDEX

## C

Collection 26  
Contributor 20

## D

Domain 22

## S

Schema 19  
Stream 17

## T

Tenant 24

## U

URI 20-21, 24