

Documentation

This documentation covers the installation, configuration, administration and use of Stroom and its related applications.

- 1: [Quick Start Guide](#)
 - 1.1: [Running Stroom](#)
 - 1.2: [Feeds](#)
 - 1.3: [Pipeline Processing](#)
 - 1.4: [Indexing](#)
 - 1.5: [Dashboards](#)
- 2: [Installation Guide](#)
 - 2.1: [Setup](#)
 - 2.1.1: [Apache Forwarding](#)
 - 2.1.2: [Java Key Store Setup](#)
 - 2.1.3: [MySQL Setup](#)
 - 2.1.4: [Processing Users](#)
 - 2.1.5: [Securing Stroom](#)
 - 2.2: [Stroom 5 Installation](#)
 - 2.3: [Stroom 6 Architecture & Deployment](#)
 - 2.4: [Stroom 6 Installation](#)
 - 2.5: [Stroom Proxy Installation](#)
 - 2.6: [Stroom Upgrades](#)
 - 2.7: [Upgrades](#)
 - 2.7.1: [v6 to v7 Upgrade](#)
 - 2.8: [Configuration](#)
 - 2.8.1: [Nginx Configuration](#)
 - 2.8.2: [Stroom Configuration](#)
 - 2.8.3: [Stroom Proxy Configuration](#)
 - 2.8.4: [Stroom Log Sender Configuration](#)
 - 2.8.5: [MySQL Configuration](#)
- 3: [User Guide](#)
 - 3.1: [Application Programming Interfaces \(API\)](#)
 - 3.1.1: [Query API](#)
 - 3.2: [Concepts](#)
 - 3.2.1: [Streams](#)
 - 3.3: [Dashboards](#)
 - 3.3.1: [Dashboard Expressions](#)
 - 3.3.1.1: [Aggregate Functions](#)
 - 3.3.1.2: [Cast Functions](#)
 - 3.3.1.3: [Date Functions](#)
 - 3.3.1.4: [Link Functions](#)
 - 3.3.1.5: [Logic Functions](#)
 - 3.3.1.6: [Mathematics Functions](#)
 - 3.3.1.7: [Rounding Functions](#)
 - 3.3.1.8: [Selection Functions](#)
 - 3.3.1.9: [String Functions](#)
 - 3.3.1.10: [Type Checking Functions](#)
 - 3.3.1.11: [URI Functions](#)
 - 3.3.1.12: [Value Functions](#)
 - 3.3.2: [Dictionaries](#)
 - 3.3.3: [Direct URLs](#)
 - 3.3.4: [Queries](#)

- 3.4: [Data Retention](#)
 - 3.5: [Data Splitter](#)
 - 3.5.1: [Simple CSV Example](#)
 - 3.5.2: [Simple CSV example with heading](#)
 - 3.5.3: [Complex example with regex and user defined names](#)
 - 3.5.4: [Multi Line Example](#)
 - 3.5.5: [Element Reference](#)
 - 3.5.5.1: [Content Providers](#)
 - 3.5.5.2: [Expressions](#)
 - 3.5.5.3: [Output](#)
 - 3.5.5.4: [Variables](#)
 - 3.5.6: [Match References, Variables and Fixed Strings](#)
 - 3.5.6.1: [Concatenation of references](#)
 - 3.5.6.2: [Expression match references](#)
 - 3.5.6.3: [Use of fixed strings](#)
 - 3.5.6.4: [Variable reference](#)
 - 3.6: [Editing and Viewing Data](#)
 - 3.7: [Event Feeds](#)
 - 3.8: [Finding Things](#)
 - 3.9: [Nodes](#)
 - 3.10: [Pipelines](#)
 - 3.10.1: [File Output](#)
 - 3.10.2: [Parser](#)
 - 3.10.2.1: [Context Data](#)
 - 3.10.2.2: [XML Fragments](#)
 - 3.10.3: [Reference Data](#)
 - 3.10.4: [XSLT Conversion](#)
 - 3.10.4.1: [XSLT Functions](#)
 - 3.10.4.2: [XSLT Includes](#)
 - 3.11: [Properties](#)
 - 3.12: [Roles](#)
 - 3.13: [Security](#)
 - 3.14: [Stroom Jobs](#)
 - 3.15: [Tools](#)
 - 3.15.1: [Command Line Tools](#)
 - 3.15.2: [Stream Dump Tool](#)
 - 3.16: [Volumes](#)
- 4: [How Tos](#)
 - 4.1: [Installation](#)
 - 4.1.1: [Apache Httpd/Mod_JK configuration for Stroom](#)
 - 4.1.2: [Database Installation](#)
 - 4.1.3: [Installation](#)
 - 4.1.4: [Installation of Stroom Application](#)
 - 4.1.5: [Installation of Stroom Proxy](#)
 - 4.1.6: [NFS Installation and Configuration](#)
 - 4.1.7: [Node Cluster URL Setup](#)
 - 4.1.8: [Processing User setup](#)
 - 4.1.9: [SSL Certificate Generation](#)
 - 4.1.10: [Testing Stroom Installation](#)
 - 4.1.11: [Volume Maintenance](#)
 - 4.2: [Reference Feeds](#)
 - 4.2.1: [Create a Simple Reference Feed](#)
 - 4.3: [Search](#)
 - 4.3.1: [Elasticsearch integration](#)
 - 4.3.2: [Search API](#)
 - 4.3.3: [Solr integration](#)

- 4.4: [Administration](#)
 - 4.4.1: [System Properties](#)
- 4.5: [Authentication](#)
 - 4.5.1: [Create a user](#)
 - 4.5.2: [Login](#)
 - 4.5.3: [Logout](#)
- 4.6: [Event Feeds](#)
 - 4.6.1: [Apache HTTPD Event Feed](#)
 - 4.6.2: [Event Processing](#)
- 4.7: [General](#)
 - 4.7.1: [Feed Management](#)
 - 4.7.2: [Raw Source Tracking](#)
 - 4.7.3: [Task Management](#)
- 5: [Sending Data to Stroom](#)
 - 5.1: [Example Clients](#)
 - 5.1.1: [curl \(Linux\)](#)
 - 5.1.2: [curl \(Windows\)](#)
 - 5.1.3: [event-logging \(Java library\)](#)
 - 5.1.4: [send_to_stroom.sh \(Linux\)](#)
 - 5.1.5: [Simple C# Client](#)
 - 5.1.6: [Simple Java Client](#)
 - 5.1.7: [stroom-log-sender \(Docker\)](#)
 - 5.1.8: [VBScript \(Windows\)](#)
 - 5.1.9: [wget \(Windows\)](#)
 - 5.2: [Header Arguments](#)
 - 5.3: [Java Keystores](#)
 - 5.4: [Payloads](#)
 - 5.5: [Response Codes](#)
 - 5.6: [SSL Configuration](#)
- 6: [Stroom Proxy](#)
 - 6.1: [Apache Forwarding](#)
 - 6.2: [Running with docker](#)
 - 6.3: [Stroom Proxy Installation](#)

This documentation is applicable to Stroom version "Legacy" and was generated on 21 Jan 2022, 16:49 UTC.

To see the documentation for other versions of Stroom click the version drop-down at the top of the screen.

1 - Quick Start Guide

How to setup an instance of Stroom and get started processing data.

In this quick-start guide you will learn how to use Stroom to get from a file that looks like this:

```
id,date,time,guid,from_ip,to_ip,application
1,6/2/2018,10:18,10990cde-1084-4006-aaf3-7fe52b62ce06,159.161.108.105,217.151.32.69,Tres-Zap
2,12/6/2017,5:58,633aa1a8-04ff-442d-ad9a-03ce9166a63a,210.14.34.58,133.136.48.23,Sub-Ex
3,6/7/2018,11:58,fabdeb8a-936f-4e1e-a410-3ca5f2ac3ed6,153.216.143.195,152.3.51.83,0tcom
4,9/2/2018,19:39,9481b2d6-f66a-4d21-ae30-26cc6ee8eced,222.144.34.33,152.67.64.215,Tres-Zap
5,8/10/2018,20:46,f8e7b436-c695-4c38-9560-66f91be199e2,236.111.169.13,239.121.20.54,Greenlam
6,10/6/2018,6:16,87c54872-ef9c-4693-a345-6f0dbcb0ea17,211.195.52.222,55.195.63.55,Bitwolf
7,12/1/2017,15:22,81a76445-f251-491c-b2c1-c09f3941c879,105.124.173.107,24.96.94.189,Kanlam
8,8/28/2018,23:25,4499f9e4-afab-4ede-af45-e12ccc8ad253,250.230.72.145,145.77.179.145,Greenlam
9,12/6/2017,21:08,4701fe3d-19cb-4ed5-88e5-a49f669b4827,192.17.20.204,123.174.48.49,Veribet
10,5/24/2018,3:35,a202d4a6-44c1-4d9d-a6b5-c2d3a246f5bd,151.153.61.111,191.59.2.47,Ventosanzap
11,10/11/2018,16:06,73bf0fde-091c-41eb-9ef5-c6a742b447a6,112.210.36.45,89.113.178.178,Sub-Ex
12,3/22/2018,2:29,3f54a04f-3045-4400-927e-fa211c61daf6,22.113.244.30,213.13.252.165,Solarbreeze
13,11/26/2017,19:47,925ff706-ebd3-4045-a8c1-a938a196ba39,239.76.244.27,50.1.42.87,Stronghold
```

Quick start test data ([Download mock_stroom_data.csv](#))

```
id,guid,from_ip,to_ip,application
1,10990cde-1084-4006-aaf3-7fe52b62ce06,159.161.108.105,217.151.32.69,Tres-Zap
2,633aa1a8-04ff-442d-ad9a-03ce9166a63a,210.14.34.58,133.136.48.23,Sub-Ex
...
```

To this XML:

```
<?xml version="1.1" encoding="UTF-8"?>
<Events xmlns:stroom="stroom" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Event>
    <Id>1</Id>
    <Guid>10990cde-1084-4006-aaf3-7fe52b62ce06</Guid>
    <FromIp>159.161.108.105</FromIp>
    <ToIp>217.151.32.69</ToIp>
    <Application>Tres-Zap</Application>
  </Event>
  <Event>
    <Id>2</Id>
    <Guid>633aa1a8-04ff-442d-ad9a-03ce9166a63a</Guid>
    <FromIp>210.14.34.58</FromIp>
    <ToIp>133.136.48.23</ToIp>
    <Application>Sub-Ex</Application>
  </Event>
  ...

```

You will go from a clean vanilla Stroom to having a simple [pipeline](#) that takes in CSV data and outputs that data transformed into XML. Stroom is a generic and powerful tool for ingesting and processing data: it's flexible because it's generic so if you do want to start processing data we would recommend you follow this tutorial otherwise you'll find yourself struggling.

We're going to do the following:

1. [Get, configure, and run Stroom](#)
2. [Get some data into Stroom](#)
3. [Set up a pipeline to process the data](#)
4. [Index the data](#)
5. [Show the data on a dashboard](#)

All the things we create here are available as a [content pack](#) ([external link](#)), so if you just wanted to see it running you could get there quite easily.

Note: The CSV data used in *mock_stroom_data.csv* (linked to above) is randomly generated and any association with any real world IP address or name is entirely coincidental.

1.1 - Running Stroom

1.1.1 Getting and Running Stroom

For this quick start you want a simple single-node Stroom. You will want to follow [these instructions](#). They do require Docker and Docker Compose, so make sure you've installed those first.

At the risk of sowing confusion you should know that there are different ways of running Stroom. Here are the full options:

- [Run using Docker Hub images \(recommended\)](#)
- [Install a stroom v5.x release](#)
- [Install a stroom v6.x release](#)
- From source you can:
 - [Build and run from IntelliJ](#)

1.1.2 Basic configuration

1.1.2.1 Enable processing of data streams

Automatic processing isn't enabled by default: you might first want to check other settings (for example nodes, properties, and volumes). So we need to enable Stream Processing. This is in Tools -> Jobs menu::



Next we need to enable Stream Processor jobs:

A screenshot of the Stroom application showing the 'Jobs' table. The table has columns for 'Job', 'Enabled', and 'Description'. The 'Stream Processor' row is highlighted with a blue background. The 'Enabled' column for this row has a checked checkbox. The 'Description' column for this row states: 'Job to process streams matching stream processor filters with their associated pipelines'. At the bottom of the table, there is a link 'Enabling stream processing'.

Job	Enabled	Description
Disable Unused Accounts	<input checked="" type="checkbox"/>	Job to disable unused accounts
File System Clean	<input checked="" type="checkbox"/>	Job to process a volume deleting files that are no longer indexed (maybe the retention period has past or the volume is full)
Node Status	<input checked="" type="checkbox"/>	Job to record status of node (CPU and Memory usage)
Proxy Aggregation	<input checked="" type="checkbox"/>	Job to pick up the data written by the proxy and store it in Stroom
Stream Processor	<input checked="" type="checkbox"/>	Job to process streams matching stream processor filters with their associated pipelines
Volume Status	<input checked="" type="checkbox"/>	Update the usage status of volumes owned by the node

Below the list of jobs is the properties pane. The Stream Processor's properties show the list of nodes. You should have one. You'll need to enable it by scrolling right:

A screenshot of the Stroom application showing the properties pane for the 'Stream Processor'. The pane has columns for 'Job', 'Node', 'Type', 'Max', 'Cur', 'Last Executed', and 'Enabled'. The 'Job' column shows 'Stream Processor'. The 'Node' column shows 'node1a'. The 'Type' column shows 'Distributed'. The 'Max' column has a value of '20' with a dropdown arrow. The 'Cur' column has a value of '0'. The 'Last Executed' column is empty. The 'Enabled' column has a checked checkbox. At the bottom of the pane, there is a link 'Enabling the nodes for the stream processor'.

Job	Node	Type	Max	Cur	Last Executed	Enabled
Stream Processor	node1a	Distributed	20	0		<input checked="" type="checkbox"/>

So now we've done that lets [get data into stroom](#).

1.2 - Feeds

How to get data into Stroom.

1.2.1 Getting data into Stroom

1.2.1.1 Create the feed

In real life you might configure Stroom to watch for new files in a directory. In this tutorial we'll be uploading data but the result will be the same: raw event data sitting on a feed.

1. A lot of Stroom's functionality is available through right-click context menus. If you right-click *System* in the tree you can create new things. Create a new folder and call it something like `Stroom 101`:



2. Right-click again and create a feed. The name needs to be capitalised, e.g. `CSV_IN`.

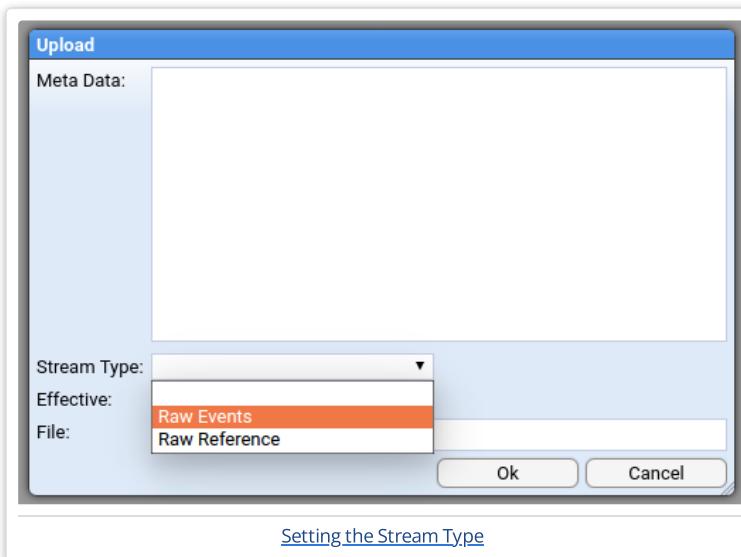


3. This will open a new tab for the feed. We want to add some data to the feed so click on *Data* at the top of the tab.



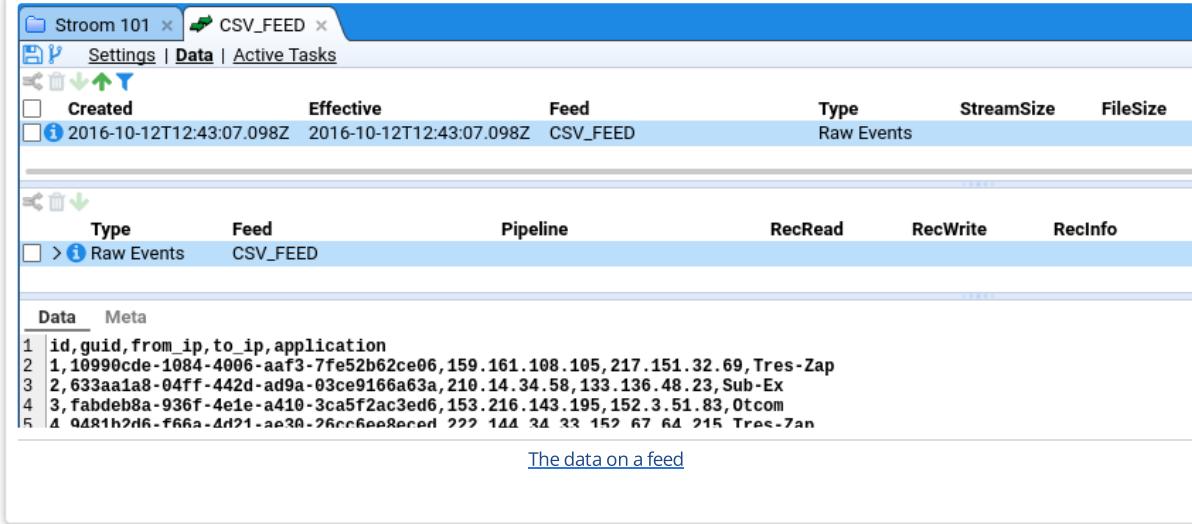
4. Then click the green up arrow to get the file upload dialog.

5. We're going to be putting in unprocessed events, known in Stroom as *raw* events. That's the type of stream this feed will contain, so that's the Stream Type you need to select.



6. Download [this file](#), then click **choose file** from the dialog, select the file, and then **ok** everything until you're back at the feed.

That's it, there's now data in Stroom. You should be able to see it in the data table (you might need to click the refresh  button):



The screenshot shows the Stroom interface with a tab titled "CSV_FEED". The main pane displays a table of "Raw Events" with columns: Created, Effective, Feed, Type, StreamSize, and FileSize. One row is shown: "2016-10-12T12:43:07.098Z" (Created), "2016-10-12T12:43:07.098Z" (Effective), "CSV_FEED" (Feed), "Raw Events" (Type), "0B" (StreamSize), and "0B" (FileSize). Below this is another table with columns: Type, Feed, Pipeline, RecRead, RecWrite, and RecInfo. It shows a single entry: "Raw Events" (Type), "CSV_FEED" (Feed), and "CSV_FEED" (Pipeline). At the bottom, there are tabs for "Data" and "Meta", and a link "The data on a feed".

Created	Effective	Feed	Type	StreamSize	FileSize
2016-10-12T12:43:07.098Z	2016-10-12T12:43:07.098Z	CSV_FEED	Raw Events	0B	0B

Type	Feed	Pipeline	RecRead	RecWrite	RecInfo
> Raw Events	CSV_FEED	CSV_FEED			

Data Meta

[The data on a feed](#)

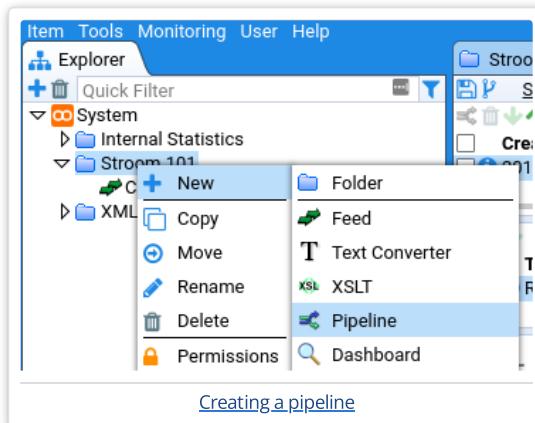
Now you can do all sorts of things with the data: transform it, visualise it, index it. It's [Pipelines](#) that allow all these things to happen.

1.3 - Pipeline Processing

Creating pipelines to process and transform data.

1.3.1 Create a pipeline

Pipelines control how data is processed in Stroom. Typically you're going to want to do a lot of the same stuff for every pipeline, i.e. similar transformations, indexing, writing out data. You can actually create a template pipeline and inherit from it, tweaking what you need to for this or that feed. We're not doing that now because we want to show how to create one from scratch.



1. Create a pipeline by right-clicking our `stroom 101` folder. Call it something like `csv to XML pipeline`.
2. Select *Structure* from the top of the new tab. This is the most important view for the pipeline because it shows what will actually happen on the pipeline.
3. Check *Advanced Mode* so that we can actually edit things.



We already have a `Source` element. Unlike most other pipeline elements this isn't something we need to configure. It's just there to show the starting point. Data gets into the pipeline via other means - we'll describe this in detail later.

1.3.1.1 Add a data splitter

Data splitters are powerful, and there is [a lot we can say](#) about them. Here we're just going to make a basic one.

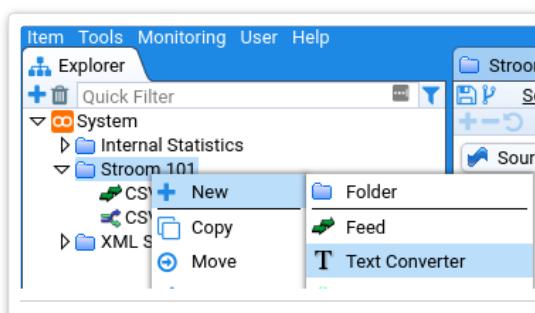
1.3.1.1.1 Create a CSV splitter

We have CSV in the following form:

```
id,guid,from_ip,to_ip,application
1,10990cde-1084-4006-aaf3-7fe52b62ce06,159.161.108.105,217.151.32.69,Tres-Zap
2,633aa1a8-04ff-442d-ad9a-03ce9166a63a,210.14.34.58,133.136.48.23,Sub-Ex
```

To process this we need to know if there's a header row, and what the delimiters are. This is a job for a *Data Splitter*.

The splitter is actually a type of *Text Converter*, so let's create one of those:



Call it something like `csv splitter`. In the new tab you need to tell the *Text Converter* that it'll be a *Data Splitter*:



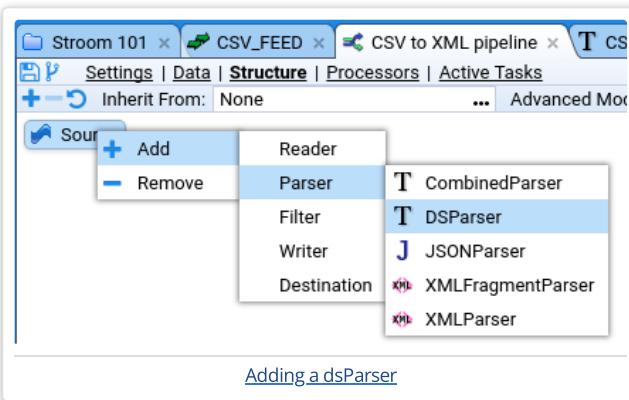
Now go to the *Conversion* tab. What you need to put in here is specific to the built-in *Data Splitter* functionality, so I'm just going to tell you what you're going to need:

```
<?xml version="1.1" encoding="UTF-8"?>
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3
  <!-- The first line contains the field names -->
  <split delimiter="\n" maxMatch="1">
    <group>
      <split delimiter="," containerStart="\"" containerEnd="\"">
        <var id="heading" />
      </split>
    </group>
  </split>

  <!-- All subsequent lines are records -->
  <split delimiter="\n">
    <group>
      <split delimiter="," containerStart="\"" containerEnd="\"">
        <data name="$heading$1" value="$1" />
      </split>
    </group>
  </split>
</dataSplitter>
```

You can see that it uses the `data-splitter-v3.0.1.xsd` that we imported earlier. Save it by clicking the save icon .

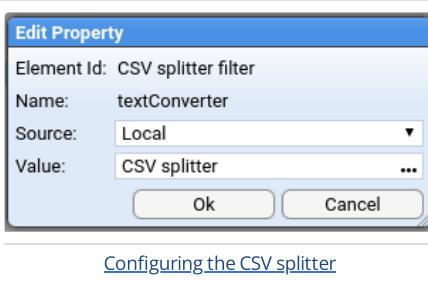
So we now have a configured, re-usable data splitter for CSV files that have headers. We need to add this to our pipeline as a filter, so head back to the pipeline's Structure section and add a DSParser, as below. Call it something like `csv splitter filter`:



Now we have a pipeline that looks like this:



Click on the `csv splitter filter` element and the pane below will show its properties. We need to tell it to use our newly created `csv splitter`. Double click the `textConverter` property and change `value` to the actual CSV splitter:



1.3.1.1.2 Test the csv splitter

So now we have CSV data in Stroom and a pipeline that is configured to process CSV data. We've done a fair few things so far and are we sure the pipeline is correctly configured? We can do some debugging and find out.

In Stroom you can step through your records and see what the output is at each stage. It's easy to start doing this. The first thing to do is to open your `csv_FEED` feed, then click the big blue *stepping* button at the bottom right:

```
55|54,2b7337a4-e245-45c1-b15b-d182ddf7a446,199.15.196.199,150.132.238.116,Bitchip
56|55,03ed4c11-783c-4208-8d85-71bc72838a74,71.84.168.61,184.203.62.40,Veribet
57|56,c7acc412-7490-459d-9208-b9f4aa1c5c0a,126.35.238.179,0.51.223.13,Alphazap
58|57,900a715e-96f2-470a-bbbf-76048fb8b77f,140.228.247.238,116.178.79.7,Tin
59|58,296ff789-5b10-416a-acef-318dc8f666f2,226.217.207.110,171.94.142.208,Kanlam
60|59,6037a7ae-df3d-489d-b5d1-a7facaa916073,212.241.212.170,206.146.52.55,Latlux
61|60,e46a4e65-79b8-457e-9860-008e49cc599c,99.46.226.254,67.81.66.1,Ventosanzap
62|61,52c11e8a-ab45-4e95-b0b5-7d27b700c48f,44.195.55.31,211.84.161.160,Zoolab
63|62,e72f2af8-cc39-4356-a463-671b8ed8bcff,25.92.147.157,44.140.226.157,Sub-Ex
64|63,51daa082-308d-46a2-962a-b29ade9d8d17,134.90.134.103,105.215.16.119,Tempsoft
65|64,a7f8e970-5b16-4175-b99b-df6ecf415d1d,191.31.234.129,109.226.93.112,Tres-Zap
66|
```

[Starting pipeline stepping](#)



You'll be asked to select a pipeline:



[Selecting a pipeline to step through](#)

Now you get a view that's similar to your feed view, except it also shows the pipeline:

The screenshot shows the Stroom 101 interface with several tabs at the top: 'Stroom 101', 'CSV_FEED', 'CSV splitter', 'CSV to XML pipeline', and 'CSV to XML pipeline'. The active tab is 'CSV splitter'. Below the tabs, there is a 'Source' node followed by a 'CSV splitter filter' node. A 'Data' pane displays the following CSV data:

```
1 id,guid,from_ip,to_ip,application
2 1,10990cde-1084-4006-aaf3-7fe52b62ce06,159.161.108.105,217.151.32.69,Tres-Zap
3 2,633aa1a8-04ff-442d-ad9a-03ce9166a63a,210.14.34.58,133.136.48.23,Sub-Ex
4 3,fabdeb8a-936f-4e1e-a410-3ca5f2ac3ed6,153.216.143.195,152.3.51.83,0tcom
```

Below the data, a link labeled 'Debugging - source data' is visible.

It also has stepping controls. Click the green step forward icon . You should see something like this:

The screenshot shows the Stroom 101 interface with the same tabs and nodes as the previous screenshot. The 'CSV splitter filter' node is active. The 'Data' pane now highlights the second row of the CSV data in yellow:

```
1 id,guid,from_ip,to_ip,application
2 1,10990cde-1084-4006-aaf3-7fe52b62ce06,159.161.108.105,217.151.32.69,Tres-Zap
3 2,633aa1a8-04ff-442d-ad9a-03ce9166a63a,210.14.34.58,133.136.48.23,Sub-Ex
4 3,fabdeb8a-936f-4e1e-a410-3ca5f2ac3ed6,153.216.143.195,152.3.51.83,0tcom
5 4,9481b2d6-f66a-4d21-ae30-26cc6ee8eced,222.144.34.33,152.67.64.215,Tres-Zap
6 5,f8e7b436-c695-4c38-9560-66f91be199e2,236.111.169.13,239.121.20.54,Greenlam
```

Below the data, a link labeled 'Stepping through the CSV data' is visible.

Great! If you don't see this then there's something wrong. Click on `CSV splitter filter`. You'll see the conversion code and hopefully some errors. Some issues might be: did you remember to import the data splitter schema into Stroom? Did you remember to configure the *Text Converter* to be a *Data Splitter*?

If everything went fine then click the step forward button a few more times and you'll see the yellow selection move down as you process each row.

What we actually want to see is the output from the *Text Converter*, so click on `CSV splitter filter`. You'll see the conversion code we entered earlier and below two panes, one containing the CSV and one containing the split-up text, in XML form:

The screenshot shows the Stroom 101 interface with the 'CSV splitter filter' node active. Below the nodes, there are two panes. The left pane contains the CSV data:

```
4,9481b2d6-f66a-4d21-ae30-26cc6ee8eced,222.144.34.33,152.67.64
```

The right pane contains the XML output:

```
<?xml version="1.1" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <record>
    <data name="id" value="4" />
    <data name="guid" value="9481b2d6-f66a-4d21-ae30-26cc6ee8eced" />
    <data name="from_ip" value="222.144.34.33" />
    <data name="to_ip" value="152.67.64.215" />
    <data name="application" value="Tres-Zap" />
  </record>
</records>
```

[The output from a working data splitter](#)

So here we have some XML in a basic format we call the *records* format. You can see the schema for *records* in the `XML schemas` folder.

1.3.1.2 Add XSLT to transform records format XML into something else

1.3.1.2.1 Create the XSLT filter

This process is very similar to creating the csv splitter :

1. Create the [XSLT](#) filter
2. Add it to the pipeline
3. Step through to make sure it's doing what we expect

Create the XSLT filter, calling it something like xslt :



On the new tab click on xslt . This is another big text field but this one accepts XSLT. This one will be very basic and just takes the split up data and puts it into fields. The XSLT for this is below but if you'd like to tinker then go ahead.

```
<?xml version="1.1" encoding="UTF-8" ?>
<xsl:stylesheet xpath-default-namespace="records:2" xmlns:stroom="stroom" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:>
<xsl:template match="records">
<Events>
<xsl:apply-templates />
</Events>
</xsl:template>
<xsl:template match="record">
<xsl:variable name="id" select="data[@name='id']/@value" />
<xsl:variable name="guid" select="data[@name='guid']/@value" />
<xsl:variable name="from_ip" select="data[@name='from_ip']/@value" />
<xsl:variable name="to_ip" select="data[@name='to_ip']/@value" />
<xsl:variable name="application" select="data[@name='application']/@value" />

<Event>
<Id><xsl:value-of select="$id" /></Id>
<Guid><xsl:value-of select="$guid" /></Guid>
<FromIp><xsl:value-of select="$from_ip" /></FromIp>
<ToIp><xsl:value-of select="$to_ip" /></ToIp>
<Application><xsl:value-of select="$application" /></Application>
</Event>
</xsl:template>
</xsl:stylesheet>
```

Make sure you save it.

Go back to the Structure section of the pipeline and add an *XSLTFilter* element. Call it something like xslt filter .



The screenshot shows the Stroom pipeline editor interface. At the top, there are several tabs: 'Stroom 101', 'CSV_FEED', 'CSV splitter', 'CSV to XML pipeline', and 'CSV'. Below the tabs, there's a toolbar with icons for 'Settings', 'Data', 'Structure', 'Processors', and 'Active Tasks'. A dropdown menu 'Inherit From:' is set to 'None'. To the right of the menu, there's an 'Advanced Mode:' checkbox which is checked. In the main workspace, there's a 'Source' node connected to a 'CSV splitter filter' node, which is then connected to an 'XSLT filter' node. A context menu is open over the 'XSLT filter' node, with the 'Filter' option highlighted. Other options in the menu include 'Reader', 'Parser', 'Destination', 'AnalyticOutputFilter', 'ID EnrichmentFilter', 'IndexingFilter', 'RecordCountFilter', 'RecordOutputFilter', and 'ReferenceDataFilter'.

ted by double-clicking xslt in the properties:

Configuring the XSLT filter						
Property Name	Value	Source	Inherited Value	Inherited From	Default Value	Description
suppressXSLTNottrue	true	Inherit	true		true	If XSLT cannot be found to match the name pat
xslt		Inherit				The XSLT to use.
xsltNamePattern		Inherit				A name pattern to load XSLT dynamically.

[Configuring the XSLT filter](#)

[Add the XSLT filter](#)

In the dialog make sure the value is the xslt filter. Save the pipeline.

1.3.1.2.2 Test the XSLT filter

We're going to test this in the same way we tested the CSV splitter, by clicking the big blue button on the feed. Click the step forward button a few times to make sure it's working then click on the XSLT element. This time you should see the XSLT filter there too, as well as the basic XML being transformed into more useful XML:

The screenshot shows the Stroom pipeline editor interface. At the top, there are several tabs: 'Stroom 101', 'CSV_FEED', 'CSV splitter', 'CSV to XML pipeline', 'XSLT', and 'CSV to XML pipeline'. Below the tabs, there's a toolbar with icons for 'Settings', 'Data', 'Structure', 'Processors', and 'Active Tasks'. A context menu is open over the 'XSLT' node, with the 'Filter' option highlighted. Other options in the menu include 'Reader', 'Parser', 'Destination', 'AnalyticOutputFilter', 'ID EnrichmentFilter', 'IndexingFilter', 'RecordCountFilter', 'RecordOutputFilter', and 'ReferenceDataFilter'.

The main workspace shows a pipeline with nodes: Source → CSV splitter filter → XSLT filter. The XSLT filter node has its configuration open, showing the XSLT code:

```

<?xml version="1.1" encoding="UTF-8" ?>
<xsl:stylesheet xpath-default-namespace="records:2" xmlns:stroom="stroom" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0">
  <xsl:template match="records">
    <Events>
      <xsl:apply-templates />
    </Events>
  </xsl:template>
  <xsl:template match="record">
    <xsl:variable name="id" select="data[@name='id']/@value" />
    <xsl:variable name="guid" select="data[@name='guid']/@value" />
    <xsl:variable name="from_ip" select="data[@name='from_ip']/@value" />
    <xsl:variable name="to_ip" select="data[@name='to_ip']/@value" />
    <xsl:variable name="application" select="data[@name='application']/@value" />

    <Event>
      <Id><xsl:value-of select="$id" /></Id>
      <Guid><xsl:value-of select="$guid" /></Guid>
      <FromIp><xsl:value-of select="$from_ip" /></FromIp>
      <ToIp><xsl:value-of select="$to_ip" /></ToIp>
      <Application><xsl:value-of select="$application" /></Application>
    </Event>
  </xsl:template>
</xsl:stylesheet>

```

Below the XSLT configuration, the raw input data is shown:

```

<?xml version="1.1" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0">
  <record>
    <data name="id" value="3" />
    <data name="guid" value="fabdeb8a-936f-4e1e-a410-3ca5f2ac3ed6" />
    <data name="from_ip" value="153.216.143.195" />
    <data name="to_ip" value="152.3.51.83" />
    <data name="application" value="Otcom" />
  </record>
</records>

```

And the transformed output data is shown:

```

<?xml version="1.1" encoding="UTF-8"?>
<Events xmlns:stroom="stroom" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0">
  <Event>
    <Id>3</Id>
    <Guid>fabdeb8a-936f-4e1e-a410-3ca5f2ac3ed6</Guid>
    <FromIp>153.216.143.195</FromIp>
    <ToIp>152.3.51.83</ToIp>
    <Application>Otcom</Application>
  </Event>
</Events>

```

[Debugging the XSLT filter](#)

Fantastic! Data converted! Well done if you've got this far. Really, there are lots of steps and things that could go wrong and you've persevered. There's a few more things to get this pipeline ready for doing this [task](#) for real. We need to get this data to a destination.

1.3.1.3 Outputting the transformed data

1.3.1.3.1 Create the XML writer

What's an XML Writer and why do you need one? The XSLT filter doesn't actually write XML but instead just passes XML events from one filter to another. In order to write XML out you need an XML writer. You don't need to create one outside the pipeline (in the way you did with the csv splitter and the xslt filter). An XML writer is just added to the pipeline like this:



That's it, no other configuration necessary.

1.3.1.3.2 Create the destination

We need to do something with the serialised XML. We'll write it to a stream. To do this we create a stream appender:



Unlike the `Source` element this element needs to be configured. We need to configure two things: the `streamType` and the `destination feed`.

1.3.1.3.2.1 Setting the feed

We'll send the output to the `csv_FEED` - all data associated with this feed will be in the same place. To do that we edit the `feed` property and set it to `csv_FEED`:



We also need to edit the `streamType` property: We set the `streamType` to Events:

Property Name Value

Source
Local
Inherit
Inherit

Edit Property

Element Id: stream appender

Name: streamType

Source: Local

Value: Events

Ok Cancel

[Setting the streamType](#)

That's it! Our pipeline is configured!

1.3.1.3.3 Test the destination

We can test the XML writer and the streamAppender using the same stepping feature. Make sure you've saved the pipeline and set a **new** stepping session running. If you click on the `stream appender` you'll see something like this:

[1:1:2] ↲ ↳ ↳ ↳ ↳

Source → CSV splitter filter → XSLT filter → XML writer → stream appender

<?xml version="1.1" encoding="UTF-8"?><Events xmlns:stroom="stroom" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><Event><Id>2</Id></Event></Events>

[The final output from the pipeline](#)

1.3.2 Set the pipeline running

Obviously you don't want to step through your data one by one. This all needs automation, and this is what Processors are for. The processor works in the background to take any unprocessed data from a feed and put it through a pipeline. So far everything on our EXAMPLE_IN feed is unprocessed.

1.3.2.1 Create a processor

Processors are created from the *Processors* section of the pipeline:

Pipeline	Tracker Ms	Tracker %	Last Poll Age	Task Count	Priority	Streams

[The processors section of the pipeline](#)

Click the add button and configure the huge dialog. You only need to set the incoming feed and the stream types:



Add Filter

Template:

Folders:

Feeds: + CSV_FEED

Pipelines: Stream Types: Raw Events

Processor Configuration

CSV_FEED x CSV to XML pipeline x CSV to XML pipeline x XSL XSLT x CSV to XML pipeline x CSV to XML pipeline x >2

Settings | Data | Structure | **Processors** | Active Tasks

Processor	Tracker Ms	Tracker %	Last Poll Age	Task Count	Priority
CSV to XML pipeline	10				
CSV to XML pipeline	10				

The new processor

Effective:

Stream Attributes:

Ok Cancel

If you scroll all the way over to the right you'll see the Enabled checkbox:

[Configure the new processor](#)

Processor Status

CSV to XML pipeline x CSV to XML pipeline x XSL XSLT x CSV to XML pipeline x CSV to XML pipeline x >2

Settings | Data | Structure | Processors | Active Tasks

Processor	Tracker Ms	Tracker %	Last Poll Age	Task Count	Priority	Streams	Events	Status	Enabled
CSV to XML pipeline	10								<input type="checkbox"/>
CSV to XML pipeline	10								<input type="checkbox"/>

The enabled checkbox

Check *enabled* for the processor and the filter you've just created. This is it, everything we've done is about to start working on its own, just like it would in a real configuration.

If you keep refreshing this table it will show you the processing status which should change after a few seconds to show that the data you have uploaded is being or has been processed. Once this has happened you should be able to open the destination feed and see the output data (or errors if there were any).

CSV_FEED CSV to XML pipeline CSV to XML pipeline XSLT CSV to XML pipeline CSV to XML pipe

Settings | Data | Active Tasks

Created	Effective	Feed	Type	StreamSize	FileSiz
2016-10-12T13:50:31.287Z	2016-10-12T12:43:07.098Z	CSV_FEED	Events	325 kB	91 kB
2016-10-12T12:43:07.098Z	2016-10-12T12:43:07.098Z	CSV_FEED	Raw Events	151 kB	79 kB

Type	Feed	Pipeline	RecRead	RecWrite	RecInfo
Raw Events	CSV_FEED	CSV to XML pipeline	0	0	0
Events	CSV_FEED				an see all the XML that we've

Data

```

1<?xml version="1.1" encoding="UTF-8"?>
2<Events xmlns:stroom="stroom" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3  <Event>
4    <Id>1</Id>
5    <Guid>10990cde-1084-4006-aaf3-7fe52b62ce06</Guid>
6    <FromIp>159.161.108.105</FromIp>
7    <ToIp>217.151.32.69</ToIp>
8    <Application>Tres-Zap</Application>
9  </Event>
10 <Event>
11   <Id>2</Id>
12   <Guid>633aa1a8-04ff-442d-ad9a-03ce9166a63a</Guid>
13   <FromIp>210.14.34.58</FromIp>
14   <ToIp>133.136.48.23</ToIp>
15   <Application>Sub-Ex</Application>
16 </Event>
17 <Event>
18   <Id>3</Id>
19   <Guid>fabdeb8a-936f-4e1e-a410-3ca5f2ac3ed6</Guid>
20   <FromIp>153.216.143.195</FromIp>
21   <ToIp>152.3.51.83</ToIp>
22   <Application>Otcom</Application>
23 </Event>
24 <Event>
25   <Id>4</Id>
26   <Guid>9481b2d6-f66a-4d21-ae30-26cc6ee8eced</Guid>
27   <FromIp>222.144.34.33</FromIp>

```

[The output of the pipeline](#)

1.4 - Indexing

Indexing the ingested data.

Before you can visualise your data with dashboards you have to index the data; First I opted for creating a specific volume to hold my data, just because I wanted to keep my shards away from the default volumes;

Go to the Tools □ Volumes menu



Once the volumes dialogue opens click the blue plus sign at the top left of the window to add a new one

A screenshot of the 'Volumes' dialog. It shows a table with three columns: 'Node', 'Path', and 'Volume Type'. There are two entries: 'node' with Path '/usr/stroom-app/volumes/defaultIndexVolume' and Volume Type 'Private', and another 'node' with Path '/usr/stroom-app/volumes/defaultStreamVolume' and Volume Type 'Public'. At the bottom of the dialog, there is a link labeled 'Volume list'.

Select the node where you want the volume to be and the path you want to create, (because we are following the quick-start guide we just have one node and limited size, but we do want to set it as active so we can write documents to it and we want it to be public, because we might want other indexes to use it; your needs might be different.

A screenshot of the 'Add Volume' dialog. It contains several input fields: 'Node' (set to 'node'), 'Path' (set to '/usr/stroom-app/volumes/engage-esm'), 'Type' (set to 'Public'), 'Stream Status' (set to 'Active'), 'Index Status' (set to 'Active'), and 'Limit' (set to '30G'). At the bottom are 'Ok' and 'Cancel' buttons. Below the dialog is a link labeled 'Volume edit'.

Click ok and we're good to go.

Then we can create an index by selecting index item in the explorer tree. You do this in the same way you create any of the items. Just select/create a folder that you want to create the new index in and right click, select New Index.

A screenshot of the Stroom Explorer tree. On the left, there is a tree view with nodes like 'Stroom.101', 'Visualisations', and 'XML Schemas'. A context menu is open over a folder named 'engage_idx'. The menu items are: 'New', 'Copy', 'Move', 'Rename', 'Delete', 'Permissions', and 'Index'. The 'Index' item is highlighted with a blue selection bar. At the bottom of the dialog, there is a link labeled 'New index'.

Choose a name for your new index



In the settings tab we need to specify the volume where we will store our shards

Now you need to add fields to this index.

Firstly there are two mandatory fields that need to be added: `StreamId` and `EventId`

Both should be of type `Id`, stored and indexed with the `Keyword` analyser



If you were following the quick-start instruction on ingesting the `mock_stroom_data.csv`, we'll use those fields here.

Open the fields tab then create the following fields:

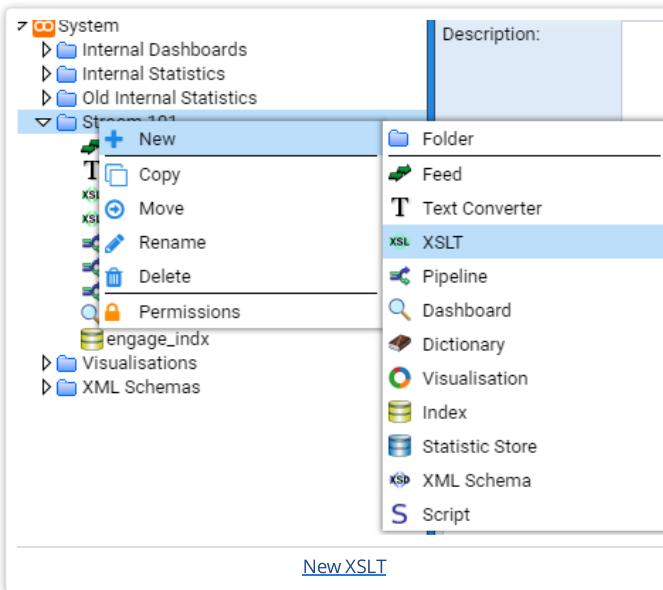
Name	Type	Store	Index	Positions	Analyser	Case Sensitive
StreamId	Id	Yes	Yes	No	Keyword	false
EventId	Id	Yes	Yes	No	Keyword	false
Id	Text	Yes	Yes	No	Keyword	false
Guid	Text	Yes	Yes	No	Alpha numeric	false
FromIp	Text	Yes	Yes	Yes	Keyword	false
ToIp	Text	Yes	Yes	Yes	Keyword	false
Application	Text	Yes	Yes	Yes	Alpha numeric	false

We are creating fields in our index to match the fields we have ingested to provide a place for the data to go that Stroom can reference.

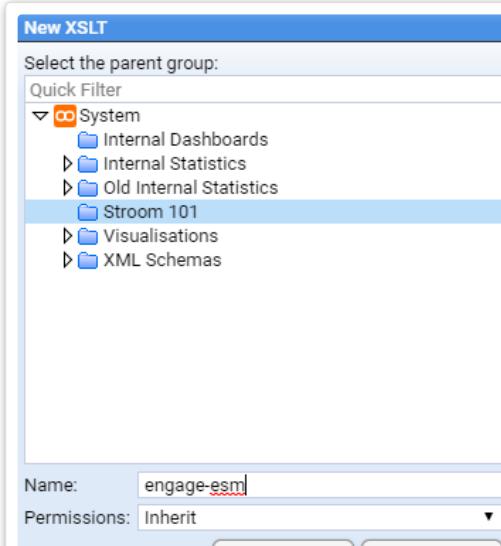
Settings Fields Shards						
Name	Type	Store	Index	Positions	Analyser	Case Sensitive
StreamId	Id	Yes	Yes	No	Keyword	false
EventId	Id	Yes	Yes	No	Keyword	false
Id	Text	Yes	Yes	No	Keyword	false
Guid	Text	Yes	Yes	No	Alpha numeric	false
FromIp	Text	Yes	Yes	Yes	Keyword	false
ToIp	Text	Yes	Yes	Yes	Keyword	false
Application	Text	Yes	Yes	Yes	Alpha numeric	false

Now create a new XSLT. We are going to convert xmi data into something indexable by Stroom.

[Index field list](#)



To make things manageable we create our new XSLT with the same name as the index in the same folder. After you've set the name just save it and close it, we'll add some code in there later.



[XSLT name](#)

Now we get to send data to the index [XSLT settings](#)

Create a new pipeline called Indexing (we are going to make this a template for future indexing requirements).



Edit the structure of the pipeline

Add the following element types with the specified names

Type	Name
XMLParser	parser
SplitFilter	splitFilter
IdEnrichmentFilter	idEnrichmentFilter
XSLTFilter	xsltFilter
IndexingFilter	indexingFilter

So it looks like this (excluding the ReadRecordCountFilter and WriteRecordCountFilter elements)



Once the elements have been added you need to set the following property on the elements:

Element	Property	Value
splitFilter	splitCount	100

To do this we select the element then double click the property value in the property panel which is below it.

Welcome x Indexing x

Settings | Data | Structure | Processors | Active Tasks

Inherit From: None Advanced Mode:

Source → parser → [readRecordCountFilter](#) → [splitFilter](#) → [ID idEnrichmentFilter](#) → [xslFilter](#) → [writeRecordCountFilter](#) → [indexingFilter](#)

View So

Property Name	Value	Source	Inherited Value	Inherited From	Default Value	Description
countRead	true	Inherit	true		true	Is this filter counting records read or records written?

1 to 1 of 1



Save the pipeline, using the top left icon , then close the pipeline tab.

Now create a new pipeline



Which we will base on our new "Indexing" template pipeline

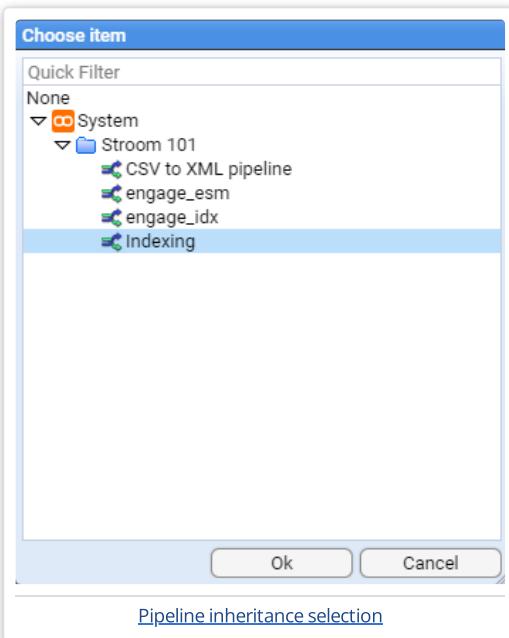
On our structure tab



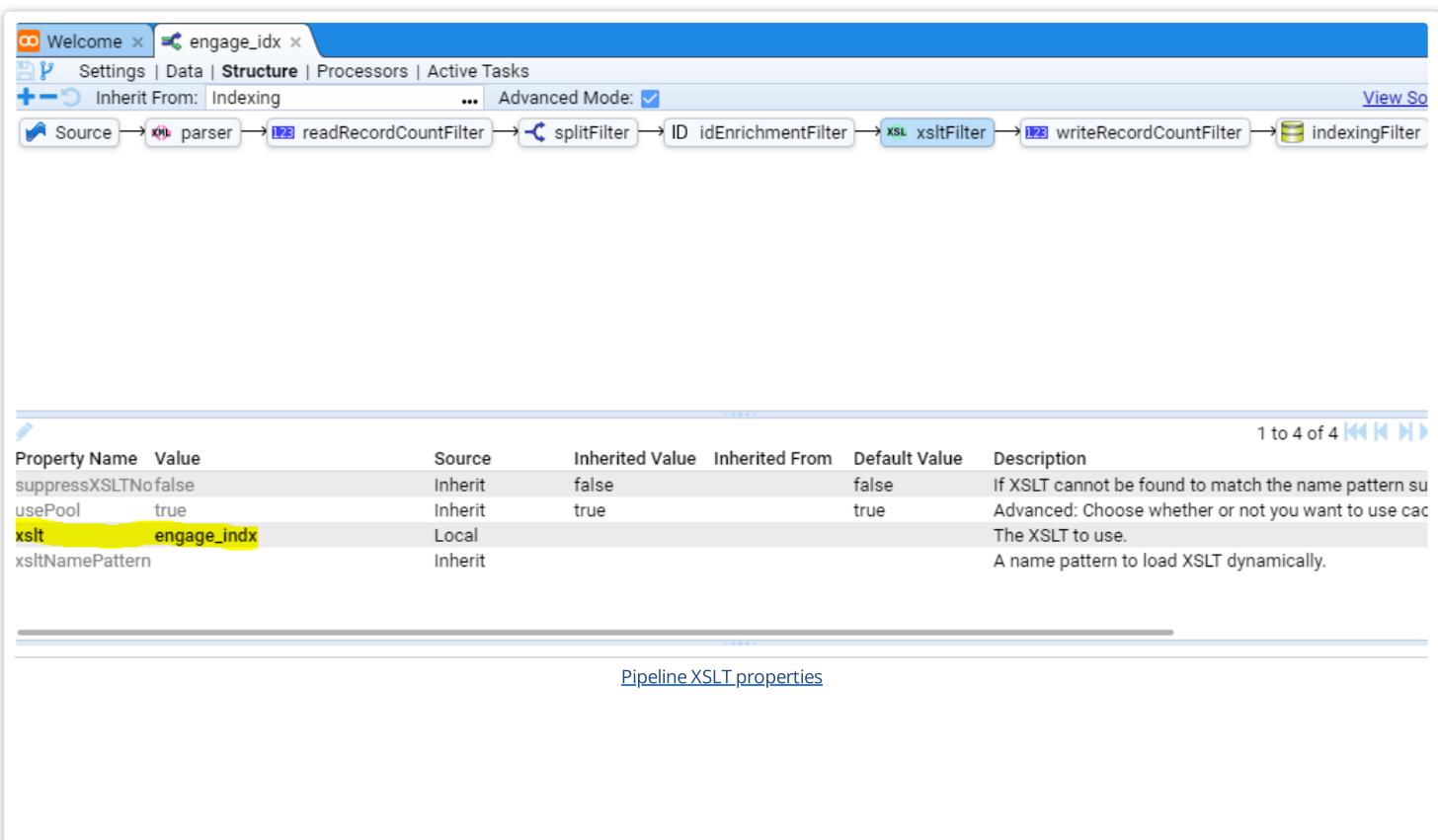
Click in the "Inherit From" window



Select our Indexing pipeline template that we just created



Now we need to set the XSLT property on the `xsltFilter` to point at the XSLT we created earlier and set the index on the `indexFilter` to point to the index we created. This will appear as below (excluding the `ReadRecordCountFilter` and `WriteRecordCountFilter` elements)



Property Name	Value	Source	Inherited Value	Inherited From	Default Value	Description
suppressXSLTNoFile	false	Inherit	false		false	If XSLT cannot be found to match the name pattern specified in the XSLT property, this property controls whether or not to suppress the error.
usePool	true	Inherit	true		true	Advanced: Choose whether or not you want to use cached XSLTs.
xslt	engage_idx	Local				The XSLT to use.
xsltNamePattern		Inherit				A name pattern to load XSLT dynamically.

Pipeline XSLT properties

Once that's done you can save your new pipeline

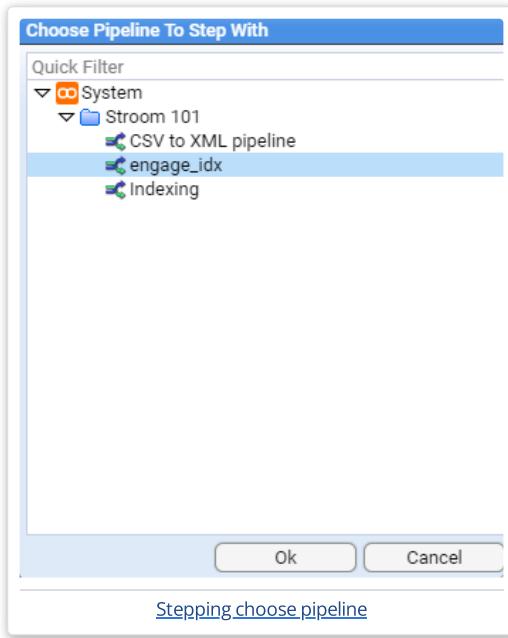
Next we need to create an XSLT that the `IndexingFilter` understands.

Open the feed we created in the quick-start guide if you find some processed data in your feed - i.e. browse the data



Click the stepping button

Select your new pipeline



Paste the following into your `xsltFilter`

```

<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet
    xmlns="records:2" xmlns:stroom="stroom"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    version="2.0">
    <xsl:template match="/Events">
        <records xsi:schemaLocation="records:2 file://records-v2.0.xsd"
            version="2.0">
            <xsl:apply-templates />
        </records>
    </xsl:template>
    <xsl:template match="Event">
        <record>
            <data name="StreamId">
                <xsl:attribute name="value" select="@StreamId" />
            </data>
            <data name="EventId">
                <xsl:attribute name="value" select="@EventId" />
            </data>
            <xsl:apply-templates select="*" />
        </record>
    </xsl:template>
    <!-- Index the Id -->
    <xsl:template match="Id">
        <data name="Id">
            <xsl:attribute name="value" select="text()" />
        </data>
    </xsl:template>
    <!-- Index the Guid -->
    <xsl:template match="Guid">
        <data name="Guid">
            <xsl:attribute name="value" select="text()" />
        </data>
    </xsl:template>
    <!-- Index the FromIp -->
    <xsl:template match="FromIp">
        <data name="FromIp">
            <xsl:attribute name="value" select="text()" />
        </data>
    </xsl:template>
    <!-- Index the ToIp -->
    <xsl:template match="ToIp">
        <data name="ToIp">
            <xsl:attribute name="value" select="text()" />
        </data>
    </xsl:template>
    <!-- Index the Application -->
    <xsl:template match="Application">
        <data name="Application">
            <xsl:attribute name="value" select="text()" />
        </data>
    </xsl:template>
</xsl:stylesheet>

```

Which should look like this

Source → **xsl parser** → ID idEnrichmentFilter → **xsl xsltFilter**

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <xsl:stylesheet
3   xmlns="records:2" xmlns:stroom="stroom"
4   xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   version="2.0">
6   <xsl:template match="/Events">
7     <records xsi:schemaLocation="records:2 file:///records-v2.0.xsd"
8       version="2.0">
9       <xsl:apply-templates />
10    </records>
11  </xsl:template>
12  <xsl:template match="Event">
13    <record>
14      <data name="StreamId">
15        <xsl:attribute name="value" select="@StreamId" />
16      </data>
17      <data name="EventId">
18        <xsl:attribute name="value" select="@EventId" />
19      </data>
20      <xsl:apply-templates select="*" />
21    </record>
22  </xsl:template>
23  <!-- Index the Id -->
24  <xsl:template match="Id">

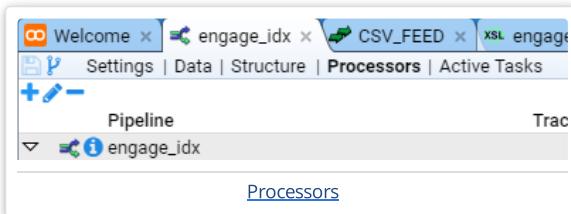
```

[Stepping edit XSLT](#)

What we are trying to do is turn the data into Stroom record format. This is basically name value pairs that we pass to the index. Step through the data using the top right arrows to ensure the XSLT produces correct output.

We're nearly there for indexing the data - you just need to tell the pipeline to pick up all processed data and index it.

Go back to your pipeline and go to the processors tab.



Add a filter using **+** and tell it to process all `Events` data when the filter dialogue opens so it looks like this



Enable the processor and the filter by clicking the enabled tick boxes



Stroom should then index the data, assuming everything is correct

If there are errors you'll see error streams produced in the data browsing page, i.e. where you would normally see your processed and raw data. If no errors have occurred, there will be no rows in the data page.

If it all goes to plan you'll see index shards appear if you open the index you created and click the shards tab.

Settings Fields Shards					
<input type="checkbox"/> Node	Partition	Path	Status	Doc Count	File Size
<input checked="" type="checkbox"/> node	2017-06-15	/usr/stroom-app/volumes/engage	Closed	2000	398K

[Index shards](#)

The document count doesn't update immediately so don't worry if the count is 0. The count is updated on shard flush and happens in the background.

Now that we have finished indexing we can display data on a [dashboard](#).

1.5 - Dashboards

Create a new dashboard the same way you create anything else



On the query pane click the settings button on the top right of the panel.

Choose the index you just created as your data source.



Now add a term to the query to get a handle on the data



For our simple example we're using a wildcard that captures all documents with an Id set.



Within the table panel we now need to set a few defaults. On the table pane click the settings button on the top right of the panel. Extract Values needs to be ticked. If grouping is to be used and the content of the groups is to be viewed then Show Group Detail should also be ticked. The Maximum Results field may also be changed from default if required to limit the results or if more results are expected than the default value. Be aware that setting this value too high may result in excessive memory being used by the query process though.

We now need to select a pipeline to display the results in the table by setting Extraction Pipeline. The simplest way is to create and save a new pipeline based on the existing Search Extraction template pipeline. Within this new pipeline, use either the XSLT used for indexing the data or preferably a copy of this XSLT saved elsewhere. The extraction pipeline and the dashboard itself should then be saved.

In the table panel we can add the fields we are interested in, in this case we wanted to sort the application field and count how many time the application name appears.

The screenshot shows a 'Table' configuration panel. At the top left is a blue plus sign icon with a green arrow pointing down. Below it is a dropdown menu labeled 'Application' with 'Add Field' selected. The main list contains fields: EventId, FromIp, Guid, Id, StreamId, Tolp, Count, Count Groups, and Custom. At the bottom is a link 'Dashboard table fields'.

If at this point, we decide that we'd like to see additional fields in the table extracted from each record then the Extraction Pipeline XSLT can be modified to extract them from the Event:

```
...
<xsl:template match="/xpath/to/usefulField">
  <data name="UsefulField">
    <xsl:attribute name="value" select="text()" />
  </data>
</xsl:template>
...
```

To be able to select this new field from the table drop-down, it needs to be added back into the list of fields in the original index:

Name	Type	Store	Index	Positions	Analyser	Case Sensitive
UsefulField	Text	No	No	No	Keyword	false

If any additions are made at this point, the index must first be saved and then the dashboard closed and reopened. usefulField will then be available as a drop-down option in the table.

Start the query and we should get this

The screenshot shows a table visualization with two columns: 'Application' and 'Count'. The 'Application' column lists various names, and the 'Count' column shows the frequency of each. The row for 'Asoka' is highlighted with a blue background. At the bottom is a link 'Dashboard table'.

Application	Count
Aerified	9
Alpha	17
Alphazap	23
Andalax	24
Asoka	23
Bamity	30
Bigtax	20
Bidex	24
Bitchip	28
Bitwolf	26
Bytcard	30
Cordcard	21

Then we can add an element from the top again and this time use visualisation

The screenshot shows a 'Welcome' dashboard with a search bar containing '* engage_dash'. A modal window is open with a list of options: 'Query', 'Table', 'Text', and 'Visualisation'. 'Visualisation' is highlighted with a blue selection bar. At the bottom is a link 'Dashboard add visualisation'.

In the visualisation panel that has been added to the bottom, click the settings gear button on the top right of the panel.

In our example we have used the Bubble visualisation

Settings

Basic | Data | Bubble

Id: vis-ME6AP

Name: Visualisation

Table: Table (table-II01T)

Visualisation: Bubble

...

Visualisation settings - basic

Settings

Basic | **Data** | Bubble

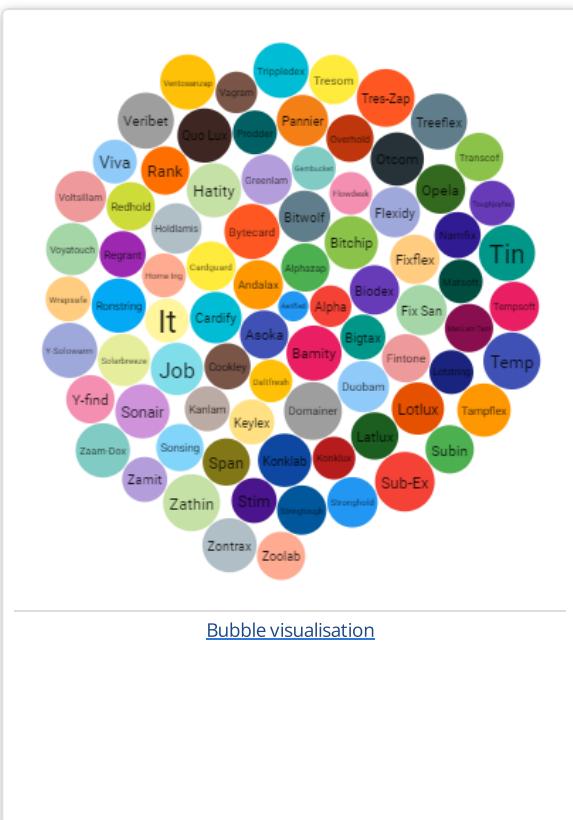
Name:	Application	▼
Value:	Count	▼
Series:	Application	▼
Grid Series:		▼

Ok Cancel

[Visualisation settings data](#)

Visualisation settings data

Which gives us this visualisation when the query is executed



Bubble visualisation

Where you can hover over elements and get a summary of that representation.



Bubble visualisation legends

2 - Installation Guide

This section describes how to install Stroom, its dependencies and related applications.

2.1 - Setup

2.1.1 - Apache Forwarding

Warning

This document refers to v4/5.

Stroom defaults to listening for HTTP on port 8080. It is recommended that Apache is used to listen on the standard HTTP port 80 and forward requests on via the Apache mod_jk module and the AJP protocol (on 8009). Apache can also perform HTTPS on port 443 and pass over requests to Tomcat using the same AJP protocol.

It is additionally recommended that Stroom Proxy is used to front data ingest and so Apache is configured to route traffic to http(s)://server/stroom/datafeed to Stroom Proxy and anything else to Stroom.

2.1.1.1 Prerequisites

- tomcat-connectors-1.2.31-src.tar.gz

2.1.1.2 Setup Apache

- As root
- Patch mod_jk

```
cd ~/tmp
tar -xvzf tomcat-connectors-1.2.31-src.tar.gz
cd tomcat-connectors-1.2.31-src/native
./configure --with-apxs=/usr/sbin/apxs
make
sudo cp apache-2.0/mod_jk.so /etc/httpd/modules/
cd
```

- Put the web server cert, private key, and CA cert into the web servers conf directory /etc/httpd/conf. E.g.

```
[user@node1 stroom-doc]$ ls -al /etc/httpd/conf
...
-rw-r--r-- 1 root root 1729 Aug 27 2013 host.crt
-rw-r--r-- 1 root root 1675 Aug 27 2013 host.key
-rw-r--r-- 1 root root 1289 Aug 27 2013 CA.crt
...
```

- Make changes to /etc/http/conf.d/ssl.conf as per below

```
JkMount /stroom* local
JkMount /stroom/remoting/cluster* local
```

```
JkOptions +ForwardKeySize +ForwardURICompat +ForwardSSLCertChain -ForwardDirectories  
  
SSLCertificateFile /etc/httpd/conf/[YOUR SERVER].crt  
SSLCertificateKeyFile /etc/httpd/conf/[YOUR SERVER].key  
SSLCertificateChainFile /etc/httpd/conf/[YOUR CA].crt  
SSLCACertificateFile /etc/httpd/conf/[YOUR CA APPENDED LIST].crt  
  
SSLOptions +ExportCertData
```

- Remove /etc/httpd/conf.d/nss.conf to avoid a 8443 port clash

```
rm /etc/httpd/conf.d/nss.conf
```

- Create a /etc/httpd/conf.d/mod_jk.conf configuration

```
LoadModule jk_module modules/mod_jk.so  
JkWorkersFile conf/workers.properties  
JkLogFile logs/mod_jk.log  
JkLogLevel info  
JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"  
JkOptions +ForwardKeySize +ForwardURICompat +ForwardSSLCertChain -ForwardDirectories  
JkRequestLogFormat "%w %V %T"
```

```
JkMount /stroom* local  
JkMount /stroom/remoting/cluster* local
```

```
JkShmFile logs/jk.shm  
<Location /jkstatus/>  
    JkMount status  
    Order deny,allow  
    Deny from all  
    Allow from 127.0.0.1  
</Location>
```

- Setup stroom-setup/cluster.txt, generate the workers file and copy into Apache. (as root and replace stroomuser with your processing user)

```
/home/stroomuser/stroom-setup/workers.properties.sh --cluster=/home/stroomuser/cluster.txt > /etc/httpd/conf/workers.properties
```

- Inspect /etc/httpd/conf/workers.properties to make sure it looks as you expect for your cluster

```
worker.list=loadbalancer,local,status  
worker.stroom_1.port=8009  
worker.stroom_1.host=localhost  
worker.stroom_1.type=ajp13  
worker.stroom_1.lbfactor=1  
worker.stroom_1.max_packet_size=65536  
....  
....  
worker.loadbalancer.type=lb  
worker.loadbalancer.balance_workers=stroom_1,stroom_2  
worker.loadbalancer.sticky_session=1  
worker.local.type=lb  
worker.local.balance_workers=stroom_1  
worker.local.sticky_session=1  
worker.status.type=status
```

- Create a simple redirect page to the stroom web app for the root URL (e.g. DocumentRoot "/var/www/html", index.html)

```
&lt;html&ampgt&lt;head&ampgt&lt;meta http-equiv="Refresh" content="0; URL=stroom"&gt;&lt;/head&ampgt&lt;/html&ampgt
```

- Restart Apache and then test default http / https access.

```
sudo /etc/init.d/httpd restart
```

2.1.1.3 Advanced Forwarding

Typically Stroom is setup so that traffic sent to /stroom* is routed to Stroom and /stroom/datafeed to Stroom Proxy. It is possible to setup an extra 1 level of datafeed routing so that based on the URL this traffic can be routed differently.

For example to route traffic directly to Stroom under the URL /stroom/datafeed/direct (avoiding any aggregation) the following mod_jk setting could be used.

```
JkMount /stroom/datafeed/direct* loadbalancer
```

2.1.2 - Java Key Store Setup

In order that the java process communicates over https (for example Stroom Proxy forwarding onto Stroom) the JVM requires relevant keystore's setting up.

As the processing user copy the following files to a directory stroom-jks in the processing user home directory :

- CA.crt - Certificate Authority
- SERVER.crt - Server certificate with client authentication attributes
- SERVER.key - Server private key

As the processing user perform the following:

- First turn your keys into der format:

```
cd ~/stroom-jks

SERVER=<SERVER crt/key PREFIX>
AUTHORITY=CA

openssl x509 -in ${SERVER}.crt -inform PEM -out ${SERVER}.crt.der -outform DER
openssl pkcs8 -topk8 -nocrypt -in ${SERVER}.key -inform PEM -out ${SERVER}.key.der -outform DER
```

- Import Keys into the Key Stores:

```
Stroom_UTIL_JAR=`find ~/*app -name 'stroom-util*.jar' -print | head -1`

java -cp ${Stroom_UTIL_JAR} stroom.util.cert.ImportKey keystore=${SERVER}.jks keypass=${SERVER} alias=${SERVER} keyfile=${SERVER}
keytool -import -noprompt -alias ${AUTHORITY} -file ${AUTHORITY}.crt -keystore ${AUTHORITY}.jks -storepass ${AUTHORITY}
```

- Update Processing User Global Java Settings:

```
PWD=`pwd`
echo "export JAVA_OPTS=\"-Djavax.net.ssl.trustStore=${PWD}/${AUTHORITY}.jks -Djavax.net.ssl.trustStorePassword=${AUTHORITY} -Dja
```

Any Stroom or Stroom Proxy instance will now additionally pickup the above JAVA_OPTS settings.

2.1.3 - MySQL Setup

2.1.3.1 Prerequisites

- MySQL 5.5.y server installed (e.g. yum install mysql-server)
- Processing User Setup

A single MySQL database is required for each Stroom instance. You do not need to setup a MySQL instance per node in your cluster.

2.1.3.2 Check Database installed and running

```
[root@stroomdb ~]# /sbin/chkconfig --list mysqld
mysqld      0:off  1:off  2:on   3:on   4:on   5:on   6:off
[root@stroomdb ~]# mysql --user=root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
...
mysql> quit
```

The following commands can be used to auto start mysql if required:

```
[root@stroomdb ~]# /sbin/chkconfig --level 345 mysqld on
[root@stroomdb ~]# /sbin/service httpd start
```

2.1.3.3 Overview

MySQL configuration can be simple to complex depending on your requirements.

For a very simple configuration you simply need an out-of-the-box mysql install and create a database user account.

Things get more complicated when considering:

- Security
- Master Slave Replication
- Tuning memory usage
- Running Stroom Stats in a different database to Stroom
- Performance Monitoring

2.1.3.4 Simple Install

Ensure the database is running, create the database and access to it

```
[stroomuser@host stroom-setup]$ mysql --user=root
Welcome to the MySQL monitor. Commands end with ; or \g.
...
mysql> create database stroom;
Query OK, 1 row affected (0.02 sec)

mysql> grant all privileges on stroom.* to 'stroomuser'@'host' identified by 'password';
Query OK, 0 rows affected (0.00 sec)

mysql> create database stroom_stats;
Query OK, 1 row affected (0.02 sec)

mysql> grant all privileges on stroom_stats.* to 'stroomuser'@'host' identified by 'password';
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

2.1.3.5 Advanced Security

It is recommended to run /usr/bin/mysql_secure_installation to remove test database and accounts.

./stroom-setup/mysql_grant.sh is a utility script that creates accounts for you to use within a cluster (or single node setup). Run to see the options:

```
[stroomuser@host stroom-setup]$ ./mysql_grant.sh
usage : --name=<instance name (defaults to my for /etc/my.cnf)>
        --user=<the stroom user for the db>
        --password=<the stroom password for the db>
        --cluster=<the file with a line per node in the cluster>
--user=<db user> Must be set
```

N.B. name is used when multiple mysql instances are setup (see below).

You need to create a file cluster.txt with a line for each member of your cluster (or single line in the case of a one node Stroom install). Then run the utility script to lock down the server access.

```
[stroomuser@host ~]$ hostname >> cluster.txt
[stroomuser@host ~]$ ./stroom-setup/mysql_grant.sh --name=mysql156_dev --user=stroomuser --password= --cluster=cluster.txt
Enter root mysql password :
-----
flush privileges
-----
-----
delete from mysql.user where user = 'stroomuser'
-----
...
...
-----
flush privileges
-----
[stroomuser@host ~]$
```

2.1.3.6 Advanced Install

The below example uses the utility scripts to create 3 custom mysql server instances on 2 servers:

- server1 - master stroom,
- server2 - slave stroom, stroom_stats

As root on server1:

```
yum install "mysql56-mysql-server"
```

Create the master database:

```
[root@node1 stroomuser]# ./stroom-setup/mysql_instance.sh --name=mysql56_stroom --port=3106 --server=mysql56 --os=rhel6
--master not set ... assuming master database
Wrote base files in tmp (You need to move them as root). cp /tmp/mysql56_stroom /etc/init.d/mysql56_stroom; cp /tmp/mysql56_
Run mysql client with mysql --defaults-file=/etc/mysql56_stroom.cnf

[root@node1 stroomuser]# cp /tmp/mysql56_stroom /etc/init.d/mysql56_stroom; cp /tmp/mysql56_stroom.cnf /etc/mysql56_stroom.cnf
[root@node1 stroomuser]# /etc/init.d/mysql56_stroom start

Initializing MySQL database: Installing MySQL system tables...
OK
Filling help tables...
...
Starting mysql56-mysqld: [ OK ]
```

Check Start up Settings Correct

```
[root@node2 stroomuser]# chkconfig mysqld off
[root@node2 stroomuser]# chkconfig mysql56-mysqld off
[root@node1 stroomuser]# chkconfig --add mysql56_stroom
[root@node1 stroomuser]# chkconfig mysql56_stroom on

[root@node2 stroomuser]# chkconfig --list | grep mysql
mysql56-mysqld 0:off 1:off 2:off 3:off 4:off 5:off 6:off
mysqld 0:off 1:off 2:off 3:off 4:off 5:off 6:off
mysql56_stroom 0:off 1:off 2:on 3:on 4:on 5:on 6:off
mysql56_stats 0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

Create a text file will all members of the cluster:

```
[root@node1 stroomuser]# vi cluster.txt

node1.my.org
node2.my.org
node3.my.org
node4.my.org
```

Create the grants:

```
[root@node1 stroomuser]# ./stroom-setup/mysql_grant.sh --name=mysql56_stroom --user=stroombuser --password=password --cluster=cl
```

As root on server2:

```
[root@node2 stroomuser]# yum install "mysql56-mysql-server"
```

```
[root@node2 stroomuser]# ./stroom-setup/mysqlld_instance.sh --name=mysqlld56_stroom --port=3106 --server=mysqlld56 --os=rhel6 --mas  
--master set ... assuming slave database  
Wrote base files in tmp (You need to move them as root). cp /tmp/mysqlld56_stroom /etc/init.d/mysqlld56_stroom; cp /tmp/mysqlld56_...  
Run mysql client with mysql --defaults-file=/etc/mysqlld56_stroom.cnf  
  
[root@node2 stroomuser]# cp /tmp/mysqlld56_stroom /etc/init.d/mysqlld56_stroom; cp /tmp/mysqlld56_stroom.cnf /etc/mysqlld56_stroom.cnf  
[root@node1 stroomuser]# /etc/init.d/mysqlld56_stroom start  
  
Initializing MySQL database: Installing MySQL system tables...  
OK  
Filling help tables...  
...  
...  
Starting mysql56-mysqld: [ OK ]
```

Check Start up Settings Correct

```
[root@node2 stroomuser]# chkconfig mysqld off  
[root@node2 stroomuser]# chkconfig mysql56-mysqld off  
[root@node1 stroomuser]# chkconfig --add mysqlld56_stroom  
[root@node1 stroomuser]# chkconfig mysqlld56_stroom on  
  
[root@node2 stroomuser]# chkconfig --list | grep mysql  
mysql56-mysqld 0:off 1:off 2:off 3:off 4:off 5:off 6:off  
mysqld 0:off 1:off 2:off 3:off 4:off 5:off 6:off  
mysqlld56_stroom 0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

Create the grants:

```
[root@node1 stroomuser]# ./stroom-setup/mysql_grant.sh --name=mysqlld56_stroom --user=stroomuser --password=password --cluster=cl
```

Make the slave database start to follow:

```
[root@node2 stroomuser]# cat /etc/mysqlld56_stroom.cnf | grep "change master"  
# change master to MASTER_HOST='node1.my.org', MASTER_PORT=3106, MASTER_USER='stroomuser', MASTER_PASSWORD='password';  
  
[root@node2 stroomuser]# mysql --defaults-file=/etc/mysqlld56_stroom.cnf  
  
mysql> change master to MASTER_HOST='node1.my.org', MASTER_PORT=3106, MASTER_USER='stroomuser', MASTER_PASSWORD='password';  
mysql> start slave;
```

As processing user on server1:

```
[stroomuser@node1 ~]$ mysql --defaults-file=/etc/mysqlld56_stroom.cnf --user=stroomuser --password=password  
  
mysql> create database stroom;  
Query OK, 1 row affected (0.00 sec)  
  
mysql> use stroom;  
Database changed  
  
mysql> create table test (a int);  
Query OK, 0 rows affected (0.05 sec)
```

As processing user on server2 check server replicating OK:

```
[stroomuser@node2 ~]$ mysql --defaults-file=/etc/myql56_stroom.cnf --user=stroomuser --password=password

mysql> show create table test;
+-----+-----+
| Table | Create Table |
+-----+-----+
| test | CREATE TABLE `test` (`a` int(11) DEFAULT NULL ) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
+-----+-----+
1 row in set (0.00 sec)
```

As root on server2:

```
[root@node2 stroomuser]# /home/stroomuser/stroom-setup/myql56_instance.sh --name=myql56_stats --port=3206 --server=myql56 --
[root@node2 stroomuser]# cp /tmp/myql56_stats /etc/init.d/myql56_stats; cp /tmp/myql56_stats.cnf /etc/myql56_stats.cnf
[root@node2 stroomuser]# /etc/init.d/myql56_stats start
[root@node2 stroomuser]# chkconfig myql56_stats on
```

Create the grants:

```
[root@node2 stroomuser]# ./stroom-setup/mysql_grant.sh --name=myql56_stats --database=stats --user=stroomstats --password=pass
```

As processing user create the database:

```
[stroomuser@node2 ~]$ mysql --defaults-file=/etc/myql56_stats.cnf --user=stroomstats --password=password
Welcome to the MySQL monitor. Commands end with ; or \g.
....
mysql> create database stats;
Query OK, 1 row affected (0.00 sec)
```

2.1.4 - Processing Users

2.1.4.1 Processing User Setup

Stroom / Stroom Proxy should be run under a processing user (we assume stroomuser below).

- Setup this user

```
/usr/sbin/adduser --system stroomuser
```

- You may want to allow normal accounts to sudo to this account for maintenance (visudo)
- Create a service script to start/stop on server startup (as root).

```
vi /etc/init.d/stroomuser

#!/bin/bash
#
# stroomuser      This shell script takes care of starting and stopping
#                  the stroomuser subsystem (tomcat6, etc)
#
# chkconfig: - 86 14
# description: stroomuser is the stroomuser sub system

Stroom_USER=stroomuser

case $1 in
start)
/bin/su ${Stroom_USER} /home/${Stroom_USER}/stroom-deploy/start.sh
;;
stop)
/bin/su ${Stroom_USER} /home/${Stroom_USER}/stroom-deploy/stop.sh
;;
restart)
/bin/su ${Stroom_USER} /home/${Stroom_USER}/stroom-deploy/stop.sh
/bin/su ${Stroom_USER} /home/${Stroom_USER}/stroom-deploy/start.sh
;;
esac
exit 0
```

- Initialise Script

```
/bin/chmod +x /etc/init.d/stroomuser
/sbin/chkconfig --level 345 stroomuser on
```

2.1.4.2 Install Java 8

```
yum install java-1.8.0-openjdk.x86_64
yum install java-1.8.0-openjdk-devel.x86_64
```

2.1.4.3 Setup Deployment Scripts

- As the processing user unpack the stroom-deploy-X-Y-Z-bin.zip generic deployment scripts in the processing users home directory.

```
unzip stroom-deploy-5.0.beta1-bin.zip
```

- Setup env.sh to include JAVA_HOME to point to the installed directory of the JDK (this will be platform specific). vi ~/env.sh

```
# User specific aliases and functions
export JAVA_HOME=/usr/lib/jvm/java-1.8.0
export PATH=${JAVA_HOME}/bin:${PATH}
```

- Setup users profile to include the same. vi ~/.bashrc

```
# User specific aliases and functions
. ~/env.sh
```

- Check that java is installed OK

```
[stroomuser@node1 ~]$ . .bashrc
[stroomuser@node1 ~]$ which java
/usr/lib/jvm/java-1.8.0/bin/java

[stroomuser@node1 ~]$ which javac
/usr/lib/jvm/java-1.8.0/bin/javac

[stroomuser@node1 ~]$ java -version
openjdk version "1.8.0_65"
OpenJDK Runtime Environment (build 1.8.0_65-b17)
OpenJDK 64-Bit Server VM (build 25.65-b01, mixed mode)
```

- Setup auto deployment crontab script as below (crontab -e)

```
[stroomuser@node1 ~]$ crontab -l
# Deploy Script
0,5,10,15,20,25,30,35,40,45,50,55 * * * * /home/stroomuser/stroom-deploy/deploy.sh >> /home/stroomuser/stroom-deploy.log
59 0 * * * rm -f /home/stroomuser/stroom-deploy.log
# Clean system
0 0 * * * /home/stroomuser/stroom-deploy/clean.sh > /dev/null
```

2.1.5 - Securing Stroom

NOTE This document was written for stroom v4/5. Some parts may not be applicable for v6+.

2.1.5.1 Firewall

The following firewall configuration is recommended:

- Outside cluster drop all access except ports HTTP 80, HTTPS 443, and any other system ports you require SSH, etc
- Within cluster allow all access

This will enable nodes within the cluster to communicate on:

- Native tomcat HTTP 8080, 9080
- Tomcat AJP 8009, 9009
- MySQL 3006

2.1.5.2 MySQL

- It is recommended that you run mysql_secure_installation to set a root password and remove test database:

```
mysql_secure_installation (provide a root password)
- Set root password? [Y/n] Y
- Remove anonymous users? [Y/n] Y
- Disallow root login remotely? [Y/n] Y
- Remove test database and access to it? [Y/n] Y
- Reload privilege tables now? [Y/n] Y
```

- stroom-setup includes a version of this script designed to be run on instances created using mysqld_instance.sh (i.e. non standard or multiple instances of mysql)

```
[stroomuser@stroom_1 stroom-setup]$ ./mysql_secure_installation.sh --name=mysqld_ref1m
```

2.2 - Stroom 5 Installation

Warning

This document was written for stroom v4/5. It is not applicable for v6+.

2.2.1 Prerequisites

- Install file 'stroom-app-distribution-X-Y-Z-bin.zip'. All the pre-built binaries are [available on GitHub \(external link\)](#)
- MySQL Server 5.5
- JDK8
- Temporarily allow port 8080, if not relying on Apache Forwarding.

2.2.2 Installing Stroom

Unpack the distribution `stroom-app-distribution-X-Y-Z-bin.zip` :

```
unzip stroom-app-distribution-X-Y-Z-bin.zip
```

In `bin` are scripts for configuring and starting and stopping Stroom.

2.2.2.1 Configuring

The `setup.sh` script will ask a series of questions to help you configure Stroom.

```
./bin/setup.sh
```

This script asks a series of questions about configuration parameters. These parameters are:

- **TEMP_DIR** - This is where Stroom will write some temporary files, e.g. imports/exports. Only change this if you do not want to use '/tmp'.
- **NODE** - Each Stroom instance in the cluster needs a unique name, if this is a reinstall ensure you use the previous deployment. **This name needs match the name used in your worker.properties (e.g. 'node1' in the case 'node1.my.org')**
- **RACK** - Used to group nodes together (so for example nodes near each other process near data)
- **PORT_PREFIX** - By default Stroom will run on port 8080
- **JDBC_CLASSNAME, JDBC_URL, DB_USERNAME, DB_PASSWORD** - MySQL connection details for the **stroom** database
- **JPA DIALECT** - Leave blank to use MySQL
- **JAVA_OPTS** - By default this is '-Xms1g -Xmx8g'. Stroom performs better if you use most of the servers memory so change the maximum memory setting (Xmx) accordingly, e.g. -Xmx40g will use 40 GB.
- **STROOM_STATISTICS_SQL_JDBC_CLASSNAME, STROOM_STATISTICS_SQL_JDBC_URL, STROOM_STATISTICS_SQL_DB_USERNAME, STROOM_STATISTICS_SQL_DB_PASSWORD** - MySQL connection details for the **statistics** database

2.2.2.2 Running

Start the configured instance:

```
./bin/start.sh
```

Inspect the logs:

```
tail -f instance/logs/stroom.log
```

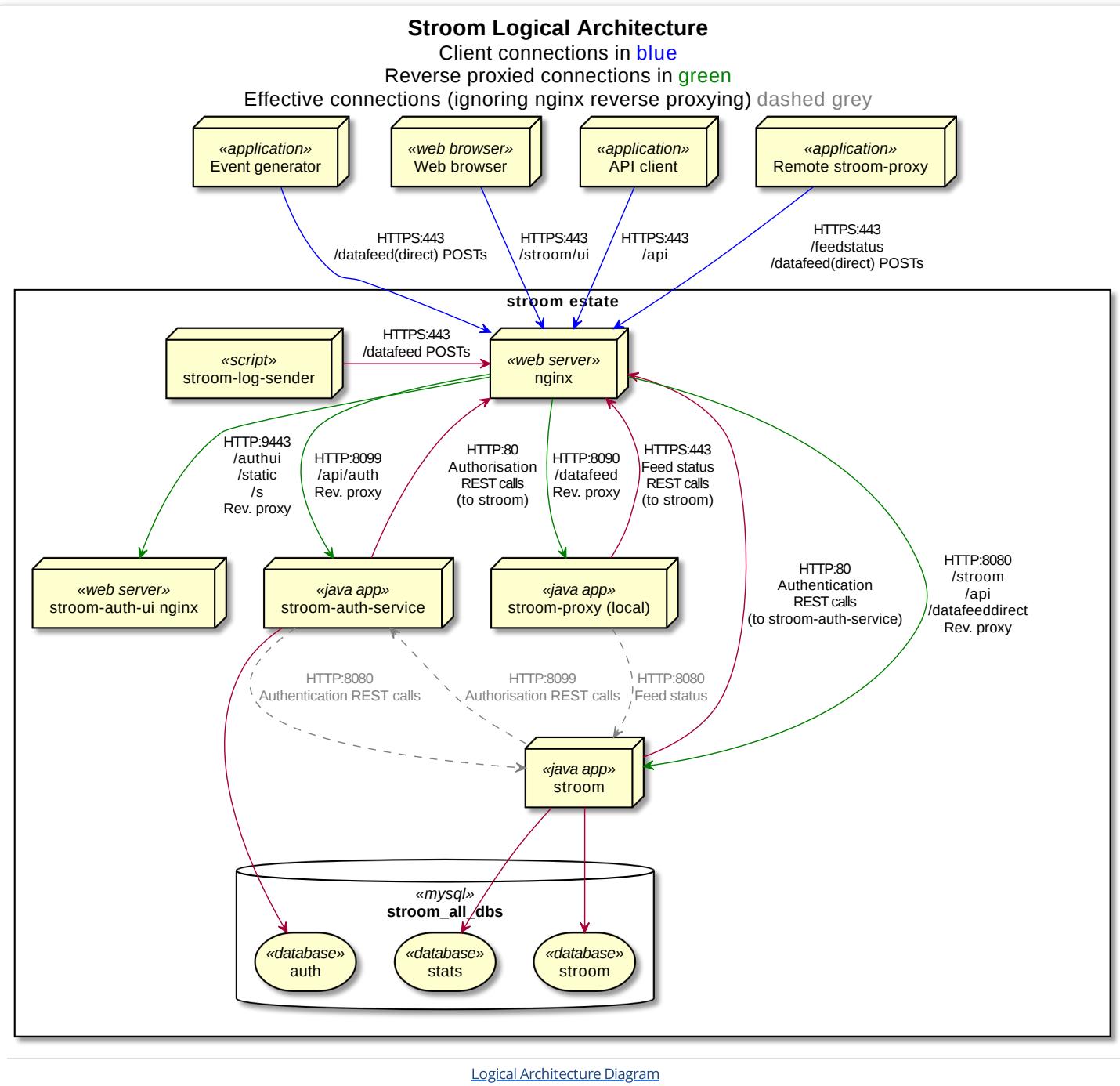
2.2.2.3 Other things to configure:

You might want to configure some of the following:

- [Processing User Setup](#)
- [MySQL Server Setup](#)
- [Java Key Store Setup](#)
- [Apache Forwarding](#)
- [Securing Stroom](#)

2.3 - Stroom 6 Architecture & Deployment

The diagram below shows the logical architecture of Stroom v6. It is not concerned with how/where the various services are deployed. This page describes represents a reference architecture and deployment for stroom but it is possible to deploy the various services in many different ways, e.g. using a different web server to Nginx or introducing load balancers.



2.3.1 Nginx

In stroom v6, a central Nginx is key to the whole architecture. It acts in the following capacities:

- A reverse proxy to abstract clients from the multiple service instances.
- An API gateway for all service traffic.
- The termination point for client SSL traffic.

2.3.1.1 Reverse Proxy

Nginx is used to reverse proxy all client connections (even those from within the estate) to the various services that sit behind it. For example, a client request to `https://nginx-host/stroom` will be reverse proxied to `http://a-stroom-host:8080/stroom`. Nginx is responsible for selecting the upstream server to reverse proxy to. It is possible to use multiple instances of Nginx for redundancy or improved performance, however care needs to be taken to ensure all requests for a session go to the same Nginx instance, i.e. sticky sessions. Some requests are stateful and some are stateless but the Nginx config will reverse proxy them accordingly.

2.3.1.2 API Gateway

Nginx is also used as an API gateway. This means all inter-service calls go via the Nginx gateway so each service only needs to know the location of the Nginx gateway. Nginx will then reverse proxy all requests to the appropriate instance of an upstream service.

The grey dashed lines on the diagram attempt to show the effective inter-service connections that are being made if you ignore the Nginx reverse proxying.

2.3.1.3 SSL Termination

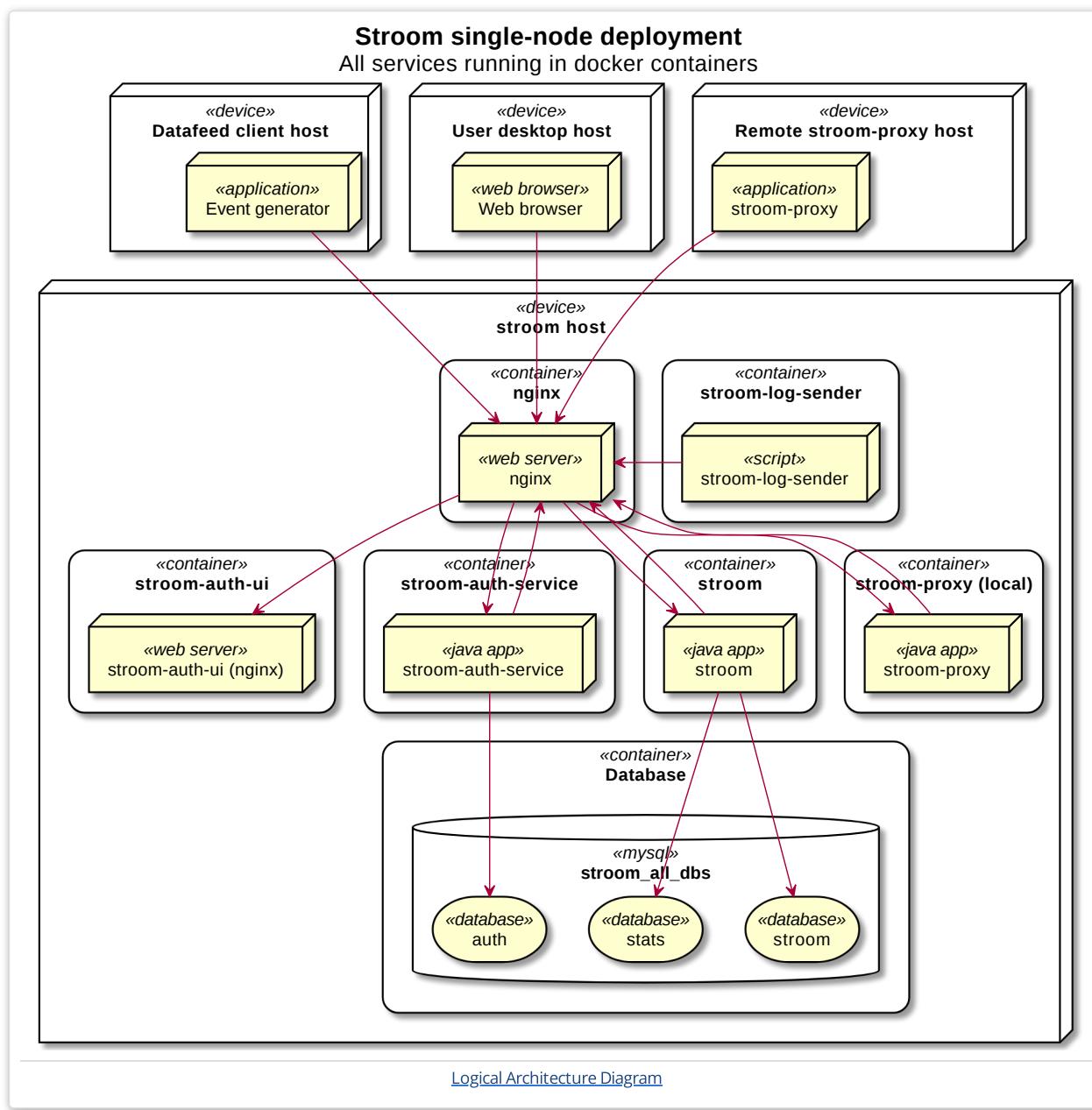
All SSL termination is handled by Nginx. Nginx holds the server and certificate authority certificate and will authenticate the client requests if the client has a certificate. Any client certificate details will be passed on to the service that is being reverse proxied.

2.3.2 Physical Deployment

2.3.2.1 Single Node Docker Deployment

The simplest deployment of stroom is where all services are on a single host and each service runs in its own docker container. Such a deployment can be achieved by following [these instructions](#).

The following diagram shows how a single node deployment would look.



2.3.2.2 Multi Node Mixed Deployment

The typical deployment for a large scale stroom is where stroom is run on multiple hosts to scale out the processing. In this deployment stroom and MySQL are run directly on the host OS, i.e. without docker. This approach was taken to gradually introduce docker into the stroom deployment strategies.

The following diagram shows how a multi node deployment would look.



[Logical Architecture Diagram](#)

2.3.2.3 Multi Node All docker Deployment

The aim in future is to run all services in docker in a multi node deployment. Such a deployment is still under development and will likely involve kubernetes for container orchestration.

2.4 - Stroom 6 Installation

We would welcome feedback on this documentation.

2.4.1 Running on a single box

2.4.1.1 Running a release

Download a [release \(external link\)](#), for example [Stroom Core v6.0 Beta 3 \(external link\)](#), unpack it, and run the `start.sh` script. When you've given it some time to start up go to `http://localhost/stroom`. There's a `README.md` file inside the tar.gz with more information.

2.4.2 Post-install hardening

2.4.2.1 Before first run

2.4.2.1.1 Change database passwords

If you don't do this before the first run of Stroom then the passwords will already be set and you'll have to change them on the database manually, and then change the `.env`.

This change should be made in the `.env` configuration file. If the values are not there then this service is not included in your Stroom stack and there is nothing to change.

- `STROOM_DB_PASSWORD`
- `STROOM_DB_ROOT_PASSWORD`
- `STROOM_STATS_DB_ROOT_PASSWORD`
- `STROOM_STATS_DB_PASSWORD`
- `STROOM_AUTH_DB_PASSWORD`
- `STROOM_AUTH_DB_ROOT_PASSWORD`
- `STROOM_ANNOTATIONS_DB_PASSWORD`
- `STROOM_ANNOTATIONS_DB_ROOT_PASSWORD`

2.4.2.2 On first run

2.4.2.2.1 Create yourself an account

After first logging in as `admin` you should create yourself a normal account (using your email address) and add yourself to the `Administrators` group. You should then log out of `admin`, log in with your new administrator account and then disable the `admin` account.

If you decide to use the `admin` account as your normal account you might find yourself locked out. The `admin` account has no associated email address, so the Reset Password feature will not work if your account is locked. It might become locked if you enter your password incorrectly too many times.

2.4.2.2.2 Delete un-used users and API keys

- If you're not using stats you can delete or disable the following:
 - the user `statsServiceUser`
 - the API key for `statsServiceUser`

2.4.2.2.3 Change the API keys

First generate new API keys. You can generate a new API key using Stroom, under `Tools -> API Keys`. The following need to be changed:

- `STROOM_SECURITY_API_TOKEN`
 - This is the API token for user `stroomServiceUser`.

Then stop Stroom and update the API key in the `.env` configuration file with the new value.

2.4.3 Troubleshooting

2.4.3.1 I'm trying to use certificate logins (PKI) but I keep being prompted for the username and password!

You need to be sure of several things:

- When a user arrives at Stroom the first thing Stroom does is redirect the user to the authentication service. This is when the certificate is checked. If this redirect doesn't use HTTPS then nginx will not get the cert and will not send it onwards to the authentication service. Remember that all of this stuff, apart from back-channel/service-to-service chatter, goes through nginx. The env var that needs to use HTTPS is `STROOM_AUTHENTICATION_SERVICE_URL`. Note that this is the var Stroom looks for, not the var as set in the stack, so you'll find it in the stack YAML.
- Are your certs configured properly? If nginx isn't able to decode the incoming cert for some reason then it won't pass anything on to the service.
- Is your browser sending certificates?

2.5 - Stroom Proxy Installation

There are 2 versions of the stroom software available for building a proxy server.

There is an 'app' version that runs stroom as a Java ARchive (jar) file locally on the server and has settings contained in a configuration file that controls access to the stroom server and database.

The other version runs stroom proxy within docker containers and also has a settings configuration file that controls access to the stroom server and database.

The document will cover the installation and configuration of the stroom proxy software for both the docker and 'app' versions.

2.5.1 Assumptions

The following assumptions are used in this document.

- the user has reasonable RHEL/CentOS System administration skills.
- installation is on a fully patched minimal CentOS 7 instance.
- the Stroom database has been created and resides on the host `stroomdb0.strmdev00.org` listening on port 3307.
- the Stroom database user is `stroomuser` with a password of `Stroompassword1@`.
- the application user `stroomuser` has been created.
- the user is or has deployed the two node Stroom cluster described [here](#).
- the user has set up the Stroom processing user as described [here](#).
- the prerequisite software has been installed.
- when a screen capture is documented, data entry is identified by the data surrounded by '<' >'. This excludes enter/return presses.

2.5.2 Stroom Remote Proxy (docker version)

The build of a stroom proxy where the stroom applications are running in docker containers.

The operating system (OS) build for a 'dockerised' stroom proxy is minimal RHEL/CentOS 7 plus the docker-ce & docker-compose packages.

Neither of the pre-requisites are available from the CentOS distribution.

It will also be necessary to open additional ports on the system firewall (where appropriate).

2.5.2.1 Download and install docker

To download and install - docker-ce - from the internet, a new 'repo' file is downloaded first, that provides access to the docker.com repository. e.g. as *root* user:

- wget <https://download.docker.com/linux/centos/docker-ce.repo> -O /etc/yum.repos.d/docker-ce.repo
- yum install docker-ce.x86_64

The packages - docker-ce docker-ce-cli & containerd.io - will be installed

The docker-compose software can be downloaded from github e.g. as *root* user to download docker-compose version 1.25.4 and save it to - /usr/local/bin/docker-compose

- curl -L https://github.com/docker/compose/releases/download/1.25.4/docker-compose-Linux-x86_64 -o /usr/local/bin/docker-compose
- chmod 755 /usr/local/bin/docker-compose

2.5.2.2 Firewall Configuration

If you have a firewall running additional ports will need to be opened, to allow the Docker containers to talk to each other. Currently these ports are:

- 3307
- 8080

- 8081
- 8090
- 8091
- 8543
- 5000
- 2888
- 443
- 80

For example on a RHEL/CentOS server using `firewalld` the commands would be as *root* user:

```
firewall-cmd --zone=public --permanent --add-port=3307/tcp
firewall-cmd --zone=public --permanent --add-port=8080/tcp
firewall-cmd --zone=public --permanent --add-port=8081/tcp
firewall-cmd --zone=public --permanent --add-port=8090/tcp
firewall-cmd --zone=public --permanent --add-port=8091/tcp
firewall-cmd --zone=public --permanent --add-port=8099/tcp
firewall-cmd --zone=public --permanent --add-port=5000/tcp
firewall-cmd --zone=public --permanent --add-port=2888/tcp
firewall-cmd --zone=public --permanent --add-port=443/tcp
firewall-cmd --zone=public --permanent --add-port=80/tcp
firewall-cmd --reload
```

2.5.2.3 Download and install Stroom v7 (docker version)

The installation example below is for stroom version 7.0.beta.45 - but is applicable to other stroom v7 versions.
As a suitable stroom user e.g. stroomuser - download and unpack the stroom software.

- wget https://github.com/gchq/stroom-resources/releases/download/stroom-stacks-v7.0-beta.41/stroom_proxy-v7.0-beta.45.tar.gz
- tar zxf stroom-stacks.....

For a stroom proxy, the configuration file - `stroom_proxy/stroom_proxy-v7.0-beta.45/stroom_proxy.env` needs to be edited, with the connection details of the stroom server that data files will be sent to.

The default network port for connection to the stroom server is 8080 The values that need to be set are:

`STROOM_PROXY_REMOTE_FEED_STATUS_API_KEY`
`STROOM_PROXY_REMOTE_FEED_STATUS_URL`
`STROOM_PROXY_REMOTE_FORWARD_URL`

The 'API key' is generated on the stroom server and is related to a specific user e.g. `proxyServiceUser` The 2 URL values also refer to the stroom server and can be a fully qualified domain name (fqdn) or the IP Address.

e.g. if the stroom server was - `stroom-serve.somewhere.co.uk` - the URL lines would be:

```
export STROOM_PROXY_REMOTE_FEED_STATUS_URL="http://stroom-serve.somewhere.co.uk:8080/api/feedStatus/v1"
export STROOM_PROXY_REMOTE_FORWARD_URL="http://stroom-serve.somewhere.co.uk:8080/stroom/datafeed"
```

2.5.2.4 To Start Stroom Proxy

As the stroom user, run the 'start.sh' script found in the stroom install:

- cd ~/stroom_proxy/stroom_proxy-v7.0-beta.45/
- ./start.sh

The first time the script is ran it will download from github the docker containers for a stroom proxy
these are - `stroom-proxy-remote`, `stroom-log-sender` and `nginx`.

Once the script has completed the stroom proxy server should be running. There are additional scripts - `status.sh` - that will show the status of the docker containers (`stroom-proxy-remote`, `stroom-log-sender` and `nginx`) and - `logs.sh` - that will tail all of the stroom message files to the screen.

2.5.3 Stroom Remote Proxy (app version)

The build of a stroom proxy server, where the stroom application is running locally as a Java ARchive (jar) file.
The operating system (OS) build for an 'application' stroom proxy is minimal RHEL/CentOS 7 plus Java.

The Java version required for stroom v7 is 12+ This version of Java is not available from the RHEL/CentOS distribution.

The version of Java used below is the 'openJDK' version as opposed to Oracle's version.

This can be downloaded from the internet.

Version 12.0.1

wget https://download.java.net/java/GA/jdk12.0.1/69cfe15208a647278a19ef0990eea691/12/GPL/openjdk-12.0.1_lin_ux-x64_bin.tar.gz Or
version 14.0.2 https://download.java.net/java/GA/jdk14.0.2/205943a0976c4ed48cb16f1043c5c647/12/GPL/openjdk-14.0.2_linux-x64_bin.tar.gz

The gzipped tar file needs to be untarred and moved to a suitable location.

- tar xvf openjdk-12.0.1_linux-x64_bin.tar.gz
- mv jdk-12.0.1 /opt/

Create a shell script that will define the Java variables OR add the statements to .bash_profile e.g. vi /etc/profile.d/jdk12.sh

export JAVA_HOME=/opt/jdk-12.0.1

export PATH=\$PATH:\$JAVA_HOME/bin

- source /etc/profile.d/jdk12.sh
- echo \$JAVA_HOME
/opt/jdk-12.0.1
- java -version *openjdk version "12.0.1" 2019-04-16
OpenJDK Runtime Environment (build 12.0.1+12)
OpenJDK 64-Bit Server VM (build 12.0.1+12, mixed mode, sharing)*

**Disable selinux to avoid issues with access and file permissions. **

2.5.3.1 Firewall Configuration

If you have a firewall running additional ports will need to be opened, to allow the Docker containers to talk to each other.
Currently these ports are:

- 3307
- 8080
- 8081
- 8090
- 8091
- 8543
- 5000
- 2888
- 443
- 80

For example on a RHEL/CentOS server using `firewalld` the commands would be as *root* user:

```

firewall-cmd --zone=public --permanent --add-port=3307/tcp
firewall-cmd --zone=public --permanent --add-port=8080/tcp
firewall-cmd --zone=public --permanent --add-port=8081/tcp
firewall-cmd --zone=public --permanent --add-port=8090/tcp
firewall-cmd --zone=public --permanent --add-port=8091/tcp
firewall-cmd --zone=public --permanent --add-port=8099/tcp
firewall-cmd --zone=public --permanent --add-port=5000/tcp
firewall-cmd --zone=public --permanent --add-port=2888/tcp
firewall-cmd --zone=public --permanent --add-port=443/tcp
firewall-cmd --zone=public --permanent - -add-port=80/tcp
firewall-cmd --reload

```

2.5.3.2 Download and install Stroom v7 (app version)

The installation example below is for stroom version 7.0.beta.45 - but is applicable to other stroom v7 versions.
As a suitable stroom user e.g. stroomuser - download and unpack the stroom software.

```

wget https://github.com/gchq/stroom/releases/download/v7.0-beta.45/stroom-proxy-app-v7.0-beta.45.zip
unzip stroom-proxy-app.....
```

The configuration file – `stroom-proxy/config/config.yml` – is the principal file to be edited, as it contains

- connection details to the stroom server
- the locations of the proxy server log files
- the directory on the proxy server, where data files will be stored prior to forwarding on to stroom
- the location of the PKI Java keystore (jks) files

The log file locations are changed to be relative to where stroom is started i.e. `~stroomuser/stroom-proxy/logs/...`

```

currentLogFilename: logs/events/event.log
archivedLogFilenamePattern: logs/events/event-%d{yyyy-MM-dd'T'HH:mm}.log
currentLogFilename: logs/events/event.log
archivedLogFilenamePattern: logs/events/event-%d{yyyy-MM-dd'T'HH:mm}.log
currentLogFilename: logs/send/send.log
archivedLogFilenamePattern: logs/send/send-%d{yyyy-MM-dd'T'HH:mm}.log.gz
currentLogFilename: logs/app/app.log
archivedLogFilenamePattern: logs/app/app-%d{yyyy-MM-dd'T'HH:mm}.log.gz
```

An API key created on the stroom server for a special proxy user is added to the configuration file. The API key is used to validate access to the application

```

proxyConfig:
  useDefaultOpenIdCredentials: **false**
  proxyContentDir: "/stroom-proxy/content"

  #If you want to use a receipt policy then the RuleSet must exist
  #in Stroom and have the UUID as specified below in receiptPolicyUuid
  #proxyRequestConfig:
  #receiptPolicyUuid: "${RECEIPT_POLICY_UUID:-}"

  feedStatus:
    url: ***"http://stroomserver.somewhere.co.uk:8080/api/feedStatus/v1}"**
    apiKey: *** eyJhbGciOiJSUz .....ScdPX0qai5Uw1BA"**
```

`forwardStreamConfig:`

`forwardingEnabled: true`

The location of the jks files has to be set, or comment all of the lines that have **sslConfig:** & **tls:** sections out to not use jks checking.

Stroom also needs the client & ca 'jks' files and by default are located in - **/stroom-proxy/certs/ca.jks & client.jks** Their location can be changed in the config.yml

```
keyStorePath: "/stroom-proxy/certs/client.jks"
trustStorePath: "/stroom-proxy/certs/ca.jks"
keyStorePath: "/stroom-proxy/certs/client.jks"
trustStorePath: "/stroom-proxy/certs/ca.jks"
```

Could be changed to.....

```
keyStorePath: "/home/stroomuser/stroom-proxy/certs/client.jks"
trustStorePath: "/home/stroomuser/stroom-proxy/certs/ca.jks"
keyStorePath: "/home/stroomuser/stroom-proxy/certs/client.jks"
trustStorePath: "/home/stroomuser/stroom-proxy/certs/ca.jks"
```

Create a directory - **/stroom-proxy** – and ensure that stroom can write to it

This is where the proxy data files are stored - **/stroom-proxy/repo**

```
proxyRepositoryConfig:
  storingEnabled: true
  repoDir: **"/stroom-proxy/repo"**
  format: "${executionUuid}/${year}-${month}-${day}/${feed}/${pathId}/${id}"
```

2.6 - Stroom Upgrades

2.6.1 Stroom v6+

Stroom is designed to detect the version of the database schema present and to run any migrations necessary to bring it up to the version begin deployed.

2.6.1.1 Docker stack deployments

TODO

2.6.1.2 Non-docker deployments

TODO

2.6.1.3 Major version upgrades

The following notes are specific for these major version upgrades

- [v6 => v7](#)

2.6.2 Stroom v5

The cleanest way to upgrade or patch is to simply remove the installed content and reinstall. For example:

```
./stroom-deploy/stop.sh
rm -fr stroom-app*

<unzip new builds as per install instructions>
<run setup.sh as per install instructions>

./stroom-deploy/start.sh
```

It is extremely important that you enter the configuration parameters correctly. In particular the node name should match the current node name otherwise Stroom will create a new node in the system thinking it is part of a cluster. It is recommended that you copy the original parameters file values used in the original installation to help with this (e.g. cp stroom-app/bin/~setup.xml /tmp/orig-stroom-app-setup.xml).

You should remove and reinstall all components you originally installed i.e. stroom-deploy-X-Y-Z, stroom-app-X-Y-X as required.

You should temporary disable the cron auto deploy script if you have it running during the above.

2.6.2.1 Patching

You can choose for a minor patch (1-2-X) to simply copy the new WAR file into relevant lib directory and run the deploy.sh script (which you may have running on a cron tab). However this would not patch any potential script or tomcat setting changes.

2.7 - Upgrades

2.7.1 - v6 to v7 Upgrade

This document describes the process for upgrading a Stroom single node docker stack from v6.x to v7.x.

Warning

Before commencing an upgrade to v7 you should upgrade Stroom to the latest minor and patch version of v6.

2.7.1.1 Differences between v6 and v7

Stroom v7 has significant differences to v6 which make the upgrade process a little more complicated.

- v6 handled authentication using a separate application, *stroom-auth-service*, with its own database. In v7 authentication is handled either internally in stroom (the default) or by an external identity provider such as google or AWS Cognito.
- v6 used a *stroom.conf* file or environment variables for configuration. In v7 stroom uses a *config.yml* file for its configuration (see [Properties](#))
- v6 used upper case and heavily abbreviated names for its tables. In v7 clearer and lower case table names are used. As a result ALL v6 tables get renamed with the prefix `OLD_`, the new tables created and any content copied over. As the database will be holding two copies of most data you need to ensure you have space to accomodate it.

2.7.1.2 Pre-Upgrade tasks

The following steps are required to be performed before migrating from v6 to v7.

2.7.1.2.1 Download migration scripts

Download the migration SQL scripts from <https://github.com/gchq/stroom/blob/STROOM VERSION/scripts> e.g.
<https://github.com/gchq/stroom/blob/v7.0-beta.133/scripts>

These scripts will be used in the steps below.

2.7.1.2.2 Pre-migration database checks

Run the pre-migration checks script on the running database.

```
docker exec -i stroom-all-dbs mysql --table -u"stroomuser" -p"stroompassword1" stroom < v7_db_pre_migration_checks.sql
```

This will produce a report of items that will not be migrated or need attention before migration.

2.7.1.2.3 Stop processing

Before shutting stroom down it is wise to turn off stream processing and let all outstanding server tasks complete.

TODO clarify steps for this.

2.7.1.2.4 Stop the stack

Stop the stack (stroom and the database) then start up the database. Do this using the v6 stack. This ensures that stroom is not trying to access the database.

```
./stop.sh  
./start.sh stroom-all-dbs
```

2.7.1.2.5 Backup the databases

Backup all the databases for the different components. Typically these will be `stroom`, `stats` and `auth`.

If you are running in a docker stack then you can run the `./backup_databases.sh` script.

2.7.1.2.6 Stop the database

Stop the database using the v6 stack.

```
./stop.sh
```

2.7.1.2.7 Deploy and configure v7

Deploy the v7 stack. *TODO* - more detail

Verify the database connection configuration for the `stroom` and `stats` databases. Ensure that there is NOT any configuration for a separate `auth` database as this will now be in `stroom`.

2.7.1.2.8 Running `mysql_upgrade`

`Stroom v6 ran on mysql v5.6. Stroom v7 runs on mysql v8.` The upgrade path for MySQL is `5.6 => 5.7.33 => 8.x`

To ensure the database is up to date `mysql_upgrade` needs to be run using the 5.7.33 binaries, [see \(external link\)](#).

This is the process for upgrading the database. All of these commands are using the v7 stack.

```
# Set the version of the MySQL docker image to use
export MYSQL_TAG=5.7.33

# Start MySQL at v5.7, this will recreate the container
./start.sh stroom-all-dbs

# Run the upgrade from 5.6 => 5.7.33
docker exec -it stroom-all-dbs mysql_upgrade -u"root" -p"my-secret-pw"

# Stop MySQL
./stop.sh

# Unset the tag variable so that it now uses the default from the stack (8.x)
unset MYSQL_TAG

# Start MySQL at v8.x, this will recreate the container and run the upgrade from 5.7.33=>8
./start.sh stroom-all-dbs

./stop.sh
```

2.7.1.2.9 Rename legacy `stroom-auth` tables

Run this command to connect to the `auth` database and run the pre-migration SQL script.

```
docker exec -i stroom-all-dbs mysql --table -u"authuser" -p"stroompassword1" auth < v7_auth_db_table_rename.sql
```

This will rename all but one of the tables in the `auth` database.

2.7.1.2.10 Copy the `auth` database content to `stroom`

Having run the table rename perform another backup of just the `auth` database.

```
./backup_databases.sh . auth
```

Now restore this backup into the `stroom` database. You can use the v7 stack scripts to do this.

```
./restore_database.sh stroom auth_20210312143513.sql.gz
```

You should now see the following tables in the `stroom` database:

```
OLD_AUTH_json_web_key  
OLD_AUTH_schema_version  
OLD_AUTH_token_types  
OLD_AUTH_tokens  
OLD_AUTH_users
```

This can be checked by running the following in the v7 stack.

```
echo 'select table_name from information_schema.tables where table_name like "OLD_AUTH%"' | ./database_shell.sh
```

2.7.1.2.11 Drop unused databases

There may be a number of databases that are no longer used that can be dropped prior to the upgrade. Note the use of the `--force` argument so it copes with users that are not there.

```
docker exec -i stroom-all-dbs mysql --force -u"root" -p"my-secret-pw" < v7_drop_unused_databases.sql
```

Verify it worked with:

```
echo 'show databases;' | docker exec -i stroom-all-dbs mysql -u"root" -p"my-secret-pw"
```

2.7.1.3 Performing the upgrade

To perform the stroom schema upgrade to v7 run the `migrate` command which will migrate the database then exit. For a large upgrade like this it is preferable to run the `migrate` command rather than just starting stroom as stroom will only migrate the parts of the schema as it needs to use them. Running `migrate` ensures all parts of the migration are completed when the command is run and no other parts of stroom will be started.

```
./migrate.sh
```

2.7.1.4 Post-Upgrade tasks

TODO remove auth* containers,images,volumes

2.8 - Configuration

Version Information: Created with Stroom v7.0

Last Updated: 2021-06-23

Stroom and its associated services can be deployed in many ways (single node docker stack, non-docker cluster, kubernetes, etc). This document will cover two types of deployment:

- Single node stroom_core docker stack.
- A mixed deployment with nginx in docker and stroom, stroom-proxy and the database not in docker.

This document will explain how each application/service is configured and where its configuration files live.

2.8.1 Application Configuration

The following sections provide links to how to configure each application.

- [Stroom Configuration](#)
- [Stroom Proxy Configuration](#)
- [MySQL Configuration](#)
- [Nginx Configuration](#)
- [Stroom log sender Configuration](#)

2.8.2 General configuration of docker stacks

2.8.2.1 Environment variables

The stroom docker stacks have a single env file `<stack name>.env` that acts as a single point to configure some aspects of the stack. Setting values in the env file can be useful when the value is shared between multiple containers. This env file sets environment variables that are then used for variable substitution in the docker compose YAML files, e.g.

```
environment:  
  - MYSQL_ROOT_PASSWORD=${STROOM_DB_ROOT_PASSWORD:-my-secret-pw}
```

In this example the environment variable `STROOM_DB_ROOT_PASSWORD` is read and used to set the environment variable `MYSQL_ROOT_PASSWORD` in the docker container. If `STROOM_DB_ROOT_PASSWORD` is not set then the value `my-secret-pw` is used instead.

The environment variables set in the env file are *NOT* automatically visible *inside* the containers. Only those environment variables defined in the `environment` section of the docker-compose YAML files are visible. These `environment` entries can either be hard coded values or use environment variables from *outside* the container. In some case the names in the env file and the names of the environment variables set in the containers are the same, in some they are different.

The environment variables set in the containers can then be used by the application running in each container to set its configuration. For example, stroom's `config.yml` file also uses variable substitution, e.g.

```
appConfig:  
  commonDbDetails:  
    connection:  
      jdbcDriverClassName: "${STROOM_JDBC_DRIVER_CLASS_NAME:-com.mysql.cj.jdbc.Driver}"
```

In this example `jdbcDriverUrl` will be set to the value of environment variable `STROOM_JDBC_DRIVER_CLASS_NAME` or `com.mysql.cj.jdbc.Driver` if that is not set.

The following example shows how setting `MY_ENV_VAR=123` means `myProperty` will ultimately get a value of `123` and not its default of `789`.

```

env file (stroom<stack name>.env) - MY_ENV_VAR=123
|
|
| environment variable substitution
|
|
v
docker compose YAML (01_stroom.yml) - STR00M_ENV_VAR=${MY_ENV_VAR:-456}
|
|
| environment variable substitution
|
|
v
Stroom configuration file (config.yml) - myProperty: "${STR00M_ENV_VAR:-789}"

```

Note that environment variables are only set into the container on start. Any changes to the env file will not take effect until the container is (re)started.

2.8.2.2 Configuration files

The following shows the basic structure of a stack with respect to the location of the configuration files:

```

-- stroom_core_test-vx.Y.Z
  └── config           [stack env file and docker compose YAML files]
    └── volumes
      └── <service>
        └── conf/config  [service specific configuration files]

```

Some aspects of configuration do not lend themselves to environment variable substitution, e.g. deeply nested parts of stroom's config.yml . In these instances it may be necessary to have static configuration files that have no connection to the env file or only use environment variables for some values.

2.8.2.3 Bind mounts

Everything in the stack volumes directory is bind-mounted into the named docker container but is mounted read-only to the container. This allows configuration files to be read by the container but not modified.

Typically the bind mounts mount a directory into the container, though in the case of the stroom-all-dbs.cnf file, the file is mounted. The mounts are done using the inode of the file/directory rather than the name, so docker will mount whatever the inode points to even if the name changes. If for instance the stroom-all-dbs.cnf file is renamed to stroom-all-dbs.cnf.old then copied to stroom-all-dbs.cnf and then the new version modified, the container would still see the old file.

2.8.2.4 Docker managed volumes

When stroom is running various forms of data are persisted, e.g. stroom's stream store, stroom-all-dbs database files, etc. All this data is stored in docker managed volumes. By default these will be located in /var/lib/docker/volumes/<volume name>/_data and root/sudo access will be needed to access these directories.

2.8.2.4.1 Docker data root

IMPORTANT

By default Docker stores all its images, container layers and managed volumes in its default data root directory which defaults to /var/lib/docker . It is typical in server deployments for the root file system to be kept fairly small and this is likely to result in the root file system running out of space due to the growth in docker images/layers/volumes in /var/lib/docker . It is therefore strongly recommended to move the docker data root to another location with more space.

There are various options for achieving this. In all cases the docker daemon should be stopped prior to making the changes, e.g. service docker stop , then started afterwards.

- **Symlink** - One option is to move the var/lib/docker directory to a new location then create a symlink to it. For example:

```
ln -s /large_mount/docker_data_root /var/lib/docker
```

This has the advantage that anyone unaware that the data root has moved will be able to easily find it if they look in the default location.

- **Configuration** - The location can be changed by adding this key to the file `/etc/docker/daemon.json` (or creating this file if it doesn't exist).

```
{  
  "data-root": "/mnt/docker"  
}
```

- **Mount** - If your intention is to use a whole storage device for the docker data root then you can mount that device to `/var/lib/docker`. You will need to make a copy of the `/var/lib/docker` directory prior to doing this then copy it mount once created. The process for setting up this mount will be OS dependent and is outside the scope of this document.

2.8.2.5 Active services

Each stroom docker stack comes pre-built with a number of different services, e.g. the `stroom_core` stack contains the following:

- `stroom`
- `stroom-proxy-local`
- `stroom-all-dbs`
- `nginx`
- `stroom-log-sender`

While you can pass a set of service names to the commands like `start.sh` and `stop.sh`, it may sometimes be required to configure the stack instance to only have a set of services active. You can set the active services like so:

```
./set_services.sh stroom stroom-all-dbs nginx
```

In the above example and subsequent use of commands like `start.sh` and `stop.sh` with no named services would only act upon the active services set by `set_services.sh`. This list of active services is held in `ACTIVE_SERVICES.txt` and the full list of available services is held in `ALL_SERVICES.txt`.

2.8.2.6 Certificates

A number of the services in the docker stacks will make use of SSL certificates/keys in various forms. The certificate/key files are typically found in the directories `volumes/<service>/certs/`.

The stacks come with a set of client/server certificates that can be used for demo/test purposes. For production deployments these should be replaced with the actual certificates/keys for your environment.

In general the best approach to configuring the certificates/keys is to replace the existing files with symlinks to the actual files. For example in the case of the server certificates for nginx (found in `volumes/nginx/certs/`) the directory would look like:

```
ca.pem.crt -> /some/path/to/certificate_authority.pem.crt  
server.pem.crt -> /some/path/to/host123.pem.crt  
server.unencrypted.key -> /some/path/to/host123.key
```

This approach avoids the need to change any configuration files to reference differently named certificate/key files and avoids having to copy your real certificates/keys into multiple places.

For examples of how to create certificates, keys and keystores see [creatCerts.sh \(external link\)](#)

2.8.1 - Nginx Configuration

Nginx is the standard web server used by stroom. Its primary role is SSL termination and reverse proxying for stroom and stroom-proxy that sit behind it. It can also load balance incoming requests and ensure traffic from the same source is always route to the same upstream instance. Other web servers can be used if required but their installation/configuration is out of the scope of this documentation.

Version Information: Created with Stroom v7.0

Last Updated: 2021-06-07

See Also: [Nginx documentation \(external link\)](#).

2.8.1.1 Without Docker

The standard way of deploying Nginx with stroom running without docker involves running Nginx as part of the *services* stack. See below for details of how to configure it. If you want to deploy Nginx without docker then you can but that is outside the scope of this documentation.

2.8.1.2 As part of a docker stack

Nginx is included in all the stroom docker stacks. Nginx is configured using multiple configuration files to aid clarity and allow reuse of sections of configuration. The main file for configuring Nginx is `nginx.conf.template` and this makes use of other files via `include` statements.

The purpose of the various files is as follows:

- `nginx.conf.template` - Top level configuration file that orchestrates the other files.
- `logging.conf.template` - Configures the logging output, its content and format.
- `server.conf.template` - Configures things like SSL settings, timeouts, ports, buffering, etc.
- Upstream configuration
 - `upstreams.stroom.ui.conf.template` - Defines the upstream host(s) for stroom node(s) that are dedicated to serving the user interface.
 - `upstreams.stroom.processing.conf.template` - Defines the upstream host(s) for stroom node(s) that are dedicated to stream processing and direct data receipt.
 - `upstreams.proxy.conf.template` - Defines the upstream host(s) for local stroom-proxy node(s).
- Location configuration
 - `locations_defaults.conf.template` - Defines some default directives (e.g. headers) for configuring stroom paths.
 - `proxy_location_defaults.conf.template` - Defines some default directives (e.g. headers) for configuring stroom-proxy paths.
 - `locations.proxy.conf.template` - Defines the various paths (e.g/ `/datafeed`) that will be reverse proxied to stroom-proxy hosts.
 - `locations.stroom.conf.template` - Defines the various paths (e.g/ `/datafeeeddirect`) that will be reverse proxied to stroom hosts.

2.8.1.2.1 Templating

The nginx container has been configured to support using environment variables passed into it to set values in the Nginx configuration files. It should be noted that recent versions of Nginx have templating support built in. The templating mechanism used in stroom's Nginx container was set up before this existed but achieves the same result.

All non-default configuration files for Nginx should be placed in `volumes/nginx/conf/` and named with the suffix `.template` (even if no templating is needed). When the container starts any variables in these templates will be substituted and the resulting files will be copied into `/etc/nginx`. The result of the template substitution is logged to help with debugging.

The files can contain templating of the form:

```
ssl_certificate /stroom-nginx/certs/<<<NGINX_SSL_CERTIFICATE>>>;
```

In this example `<<<NGINX_SSL_CERTIFICATE>>>` will be replaced with the value of environment variable `NGINX_SSL_CERTIFICATE` when the container starts.

2.8.1.2.2 Upstreams

When configuring a multi node cluster you will need to configure the upstream hosts. Nginx acts as a reverse proxy for the applications behind it so the lists of hosts for each application need to be configured.

For example if you have a 10 node cluster and 2 of those nodes are dedicated for user interface use then the configuration would look like:

upstreams.stroom.ui.conf.template

```
server node1.stroomhosts:<<<STROOM_PORT>>>
server node2.stroomhosts:<<<STROOM_PORT>>>
```

upstreams.stroom.processing.conf.template

```
server node3.stroomhosts:<<<STROOM_PORT>>>
server node4.stroomhosts:<<<STROOM_PORT>>>
server node5.stroomhosts:<<<STROOM_PORT>>>
server node6.stroomhosts:<<<STROOM_PORT>>>
server node7.stroomhosts:<<<STROOM_PORT>>>
server node8.stroomhosts:<<<STROOM_PORT>>>
server node9.stroomhosts:<<<STROOM_PORT>>>
server node10.stroomhosts:<<<STROOM_PORT>>>
```

upstreams.proxy.conf.template

```
server node3.stroomhosts:<<<STROOM_PORT>>>
server node4.stroomhosts:<<<STROOM_PORT>>>
server node5.stroomhosts:<<<STROOM_PORT>>>
server node6.stroomhosts:<<<STROOM_PORT>>>
server node7.stroomhosts:<<<STROOM_PORT>>>
server node8.stroomhosts:<<<STROOM_PORT>>>
server node9.stroomhosts:<<<STROOM_PORT>>>
server node10.stroomhosts:<<<STROOM_PORT>>>
```

In the above example the port is set using templating as it is the same for all nodes. Nodes 1 and 2 will receive all UI and REST API traffic. Nodes 8-10 will serve all datafeed(direct) requests.

2.8.1.2.3 Certificates

The stack comes with a default server certificate/key and CA certificate for demo/test purposes. The files are located in `volumes/nginx/certs/`. For a production deployment these will need to be changed, see [Certificates](#)

2.8.1.2.4 Log rotation

The Nginx container makes use of logrotate to rotate Nginx's log files after a period of time so that rotated logs can be sent to stroom. Logrotate is configured using the file `volumes/stroom-log-sender/logrotate.conf.template`. This file is templated in the same way as the Nginx configuration files, see [above](#). The number of rotated files that should be kept before deleting them can be controlled using the line.

```
rotate 100
```

This should be set in conjunction with the frequency that logrotate is called, which is controlled by `volumes/stroom-log-sender/crontab.txt`. This crontab file drives the lograte process and by default is set to run every minute.

2.8.2 - Stroom Configuration

Version Information: Created with Stroom v7.0

Last Updated: 2021-06-23

See Also: [Properties](#).

2.8.2.1 General configuration

The Stroom application is essentially just an executable [JAR \(external link\)](#) file that can be run when provided with a configuration file, `config.yml`. This config file is common to all forms of deployment.

2.8.2.1.1 config.yml

This file, sometimes known as the DropWizard configuration file (as DropWizard is the java framework on which Stroom runs) is the primary means of configuring stroom. As a minimum this file should be used to configure anything that needs to be set before stroom can start up, e.g. database connection details or is specific to a node in a stroom cluster. If you are using some form of scripted deployment, e.g. ansible then it can be used to set all stroom properties for the environment that stroom runs in. If you are not using scripted deployments then you can maintain stroom's node agnostic configuration properties via the user interface.

For more details on the structure of the file, data types and property precedence see [Properties](#).

Stroom operates on a configuration by exception basis so all configuration properties will have a sensible default value and a property only needs to be explicitly configured if the default value is not appropriate, e.g. for tuning a large scale production deployment or where values are environment specific. As a result `config.yml` only contains a minimal set of properties. The full tree of properties can be seen in `./config/config-defaults.yml` and a schema for the configuration tree (along with descriptions for each property) can be found in `./config/config-schema.yml`. These two files can be used as a reference when configuring stroom.

2.8.2.1.1 Key Configuration Properties

The following are key properties that would typically be changed for a production deployment. All configuration branches are relative to the `appConfig` root.

The database name(s), hostname(s), port(s), usernames(s) and password(s) should be configured using these properties. Typically stroom is configured to keep its statistics data in a separate database to the main stroom database, as is configured below.

```
commonDbDetails:  
  connection:  
    jdbcDriverUrl: "jdbc:mysql://localhost:3307/stroom?useUnicode=yes&characterEncoding=UTF-8"  
    jdbcDriverUsername: "stroomuser"  
    jdbcDriverPassword: "stroompassword1"  
  statistics:  
    sql:  
      db:  
        connection:  
          jdbcDriverUrl: "jdbc:mysql://localhost:3307/stats?useUnicode=yes&characterEncoding=UTF-8"  
          jdbcDriverUsername: "statsuser"  
          jdbcDriverPassword: "stroompassword1"
```

In a clustered deployment each node must be given a node name that is unique within the cluster. This is used to identify nodes in the Nodes screen. It could be the hostname of the node or follow some other naming convention.

```
node:  
  name: "node1a"
```

Each node should have its identity on the network configured so that it uses the appropriate FQDNs. The `nodeuri` hostname is the FQDN of each node and used by nodes to communicate with each other, therefore it can be private to the cluster of nodes. The `publicUri` hostname is the public facing FQDN for stroom, i.e. the address of a load balancer or Nginx. This is the address that users will use in their browser.

```

nodeUri:
  hostname: "localhost" # e.g. node5.stroomnodes.somedomain
publicUri:
  hostname: "localhost" # e.g. stroom.somedomain

```

2.8.2.2 Deploying without Docker

Stroom running without docker has two files to configure it. The following locations are relative to the stroom home directory, i.e. the root of the distribution zip.

- `./config/config.yml` - Stroom configuration YAML file
- `./config/scripts.env` - Stroom scripts configuration env file

The distribution also includes these files which are helpful when it comes to configuring stroom.

- `./config/config-defaults.yml` - Full version of the config.yml file containing all branches/leaves with default values set. Useful as a reference for the structure and the default values.
- `./config/config-schema.yml` - The schema defining the structure of the `config.yml` file.

2.8.2.2.1 scripts.env

This file is used by the various shell scripts like `start.sh`, `stop.sh`, etc. This file should not need to be unless you want to change the locations where certain log files are written to or need to change the java memory settings.

In a production system it is highly likely that you will need to increase the java heap size as the default is only 2G. The heap size settings and any other java command line options can be set by changing:

```
JAVA_OPTS="-Xms512m -Xmx2048m"
```

2.8.2.3 As part of a docker stack

When stroom is run as part of one of our docker stacks, e.g. `stroom_core` there are some additional layers of configuration to take into account, but the configuration is still primarily done using the `config.yml` file.

Stroom's `config.yml` file is found in the stack in `./volumes/stroom/config/` and this is the primary means of configuring Stroom.

The stack also ships with a default `config.yml` file baked into the docker image. This minimal fallback file (located in `/stroom/config-fallback/` inside the container) will be used in the absence of one provided in the docker stack configuration (`./volumes/stroom/config/`).

The default `config.yml` file uses [environment variable substitution](#) so some configuration items will be set by environment variables set into the container by the stack `env` file and the docker-compose YAML. This approach is useful for configuration values that need to be used by multiple containers, e.g. the public FQDN of Nginx, so it can be configured in one place.

If you need to further customise the stroom configuration then it is recommended to edit the `./volumes/stroom/config/config.yml` file. This can either be a simple file with hard coded values or one that uses environment variables for some of its configuration items.

The configuration works as follows:

```

env file (stroom<stack name>.env)
  |
  |
  | environment variable substitution
  |
  v
docker compose YAML (01_stroom.yml)
  |
  |
  | environment variable substitution
  |
  v
Stroom configuration file (config.yml)

```

2.8.2.3.1 Ansible

If you are using Ansible to deploy a stack then it is recommended that all of stroom's configuration properties are set directly in the `config.yml` file using a templated version of the file and to NOT use any environment variable substitution. When using Ansible, the Ansible inventory is the single source of truth for your configuration so not using environment variable substitution for stroom simplifies the configuration and makes it clearer when looking at deployed configuration files.

[Stroom-ansible](#) has an example inventory for a single node stroom stack deployment. The [group_vars/all](#) file shows how values can be set into the `env` file.

2.8.3 - Stroom Proxy Configuration

Version Information: Created with Stroom v7.0

Last Updated: 2021-06-23

See Also: [Stroom Application Configuration](#)

See Also: [Properties](#).

TODO: This needs updating for v7.1

The configuration of Stroom-proxy is very much the same as for Stroom with the only difference being the structure of the `config.yml` file. Stroom-proxy has a `proxyConfig` key in the YAML while Stroom has `appConfig`. It is recommended to first read [Stroom Application Configuration](#) to understand the general mechanics of the stroom configuration as this will largely apply to stroom-proxy.

2.8.3.1 General configuration

The Stroom-proxy application is essentially just an executable [JAR \(external link\)](#) file that can be run when provided with a configuration file, `config.yml`. This configuration file is common to all forms of deployment.

2.8.3.1.1 config.yml

Stroom-proxy does not have a user interface so the `config.yml` file is the only way of configuring stroom-proxy. As with stroom, the `config.yml` file is split into three sections using these keys:

- `server` - Configuration of the web server, e.g. ports, paths, request logging.
- `logging` - Configuration of application logging
- `proxyConfig` - Configuration of stroom-proxy

See also [Properties](#) for more details on structure of the config.yml file and supported data types.

Stroom-proxy operates on a configuration by exception basis so all configuration properties will have a sensible default value and a property only needs to be explicitly configured if the default value is not appropriate, e.g. for tuning a large scale production deployment or where values are environment specific. As a result `config.yml` only contains a minimal set of properties. The full tree of properties can be seen in `./config/config-defaults.yml` and a schema for the configuration tree (along with descriptions for each property) can be found in `./config/config-schema.yml`. These two files can be used as a reference when configuring stroom.

2.8.3.1.1.1 Key Configuration Properties

Stroom-proxy has two main functions, storing and forwarding. It can be configured to do either or both of these functions. These functions are enabled/disabled using:

```
proxyConfig:  
  
  forwardStreamConfig:  
    forwardingEnabled: true  
  
  proxyRepositoryConfig:  
    storingEnabled: true
```

Stroom-proxy should be configured to check the receipt status of feeds on receipt of data. This is done by configuring the end point of a downstream stroom-proxy or stroom.

```
feedStatus:  
  url: "http://stroom:8080/api/feedStatus/v1"  
  apiKey: ""
```

The `url` should be the url for the feed status API on the downstream stroom(-proxy). If this is on the same host then you can use the http endpoint, however if it is on a remote host then you should use https and the host of its nginx, e.g. `https://downstream-instance/api/feedStatus/v1`.

In order to use the API, the proxy must have a configured `apiKey`. The API key must be created in the downstream stroom instance and then copied into this configuration.

If the proxy is configured to forward data then the forward destination(s) should be set. This is the `datafeed` endpoint of the downstream stroom-proxy or stroom instance that data will be forwarded to. This may also be the address of a load balancer or similar that is fronting a cluster of stroom-proxy or stroom instances. See also [Feed status certificate configuration](#).

```
forwardStreamConfig:  
  forwardDestinations:  
    - forwardUrl: "https://nginx/stroom/datafeed"
```

`forwardUrl` specifies the URL of the `datafeed` endpoint on the destination host. Each forward location can use a different key/trust store pair. See also [Forwarding certificate configuration](#).

If the proxy is configured to store then it is the location of the proxy repository may need to be configured if it needs to be in a different location to the proxy home directory, e.g. on another mount point.

2.8.3.2 Deploying without Docker

Apart from the structure of the `config.yml` file, the configuration in a non-docker environment is the same as for [stroom](#)

2.8.3.3 As part of a docker stack

The way stroom-proxy is configured is essentially the same as for [stroom](#) with the only real difference being the structure of the `config.yml` file as note [above](#). As with stroom the docker stack comes with a `./volumes/stroom-proxy-*/config/config.yml` file that will be used in the absence of a provided one. Also as with stroom, the `config.yml` file supports environment variable substitution so can make use of environment variables set in the stack env file and passed down via the docker-compose YAML files.

2.8.3.3.1 Certificates

Stroom-proxy makes use of client certificates for two purposes:

- Communicating with a downstream stroom/stroom-proxy in order to establish the receipt status for the feeds it has received data for.
- When forwarding data to a downstream stroom/stroom-proxy

The stack comes with the following files that can be used for demo/test purposes.

```
volumes/stroom-proxy-*/certs/ca.jks  
volumes/stroom-proxy-*/certs/client.jks
```

For a production deployment these will need to be changed, see [Certificates](#)

2.8.3.3.1.1 Feed status certificate configuration

The configuration of the client certificates for feed status checks is done using:

```
proxyConfig:
```

```
jerseyClient:  
  timeout: "10s"  
  connectionTimeout: "10s"  
  timeToLive: "1h"  
  cookiesEnabled: false  
  maxConnections: 1024  
  maxConnectionsPerRoute: "1024"  
  keepAlive: "0ms"  
  retries: 0  
  tls:  
    verifyHostname: true  
    keyStorePath: "/stroom-proxy/certs/client.jks"  
    keyStorePassword: "password"  
    keyStoreType: "JKS"  
    trustStorePath: "/stroom-proxy/certs/ca.jks"  
    trustStorePassword: "password"  
    trustStoreType: "JKS"  
    trustSelfSignedCertificates: false
```

This configuration is also used for making any other REST API calls.

2.8.3.3.1.2 Forwarding certificate configuration

Stroom-proxy can forward to multiple locations. The configuration of the certificate(s) for the forwarding locations is as follows:

```
proxyConfig:
```

```
forwardStreamConfig:  
  forwardingEnabled: true  
  forwardDestinations:  
    # If you want multiple forward destinations then you will need to edit this file directly  
    # instead of using env var substitution  
    - forwardUrl: "https://nginx/stroom/datafeed"  
      sslConfig:  
        keyStorePath: "/stroom-proxy/certs/client.jks"  
        keyStorePassword: "password"  
        keyStoreType: "JKS"  
        trustStorePath: "/stroom-proxy/certs/ca.jks"  
        trustStorePassword: "password"  
        trustStoreType: "JKS"  
        hostnameVerificationEnabled: true
```

`forwardUrl` specifies the URL of the *datafeed* endpoint on the destination host. Each forward location can use a different key/trust store pair.

2.8.4 - Stroom Log Sender Configuration

Version Information: Created with Stroom v7.0

Last Updated: 2021-06-14

Stroom log sender is a docker image used for sending application logs to stroom. It is essentially just a combination of the [send_to_stroom.sh \(external link\)](#) script and a set of crontab entries to call the script at intervals.

2.8.4.1 Deploying without Docker

When deploying without docker stroom and stroom-proxy nodes will need to be configured to send their logs to stroom. This can be done using the `./bin/send_to_stroom.sh` script in the stroom and stroom-proxy zip distributions and some crontab configuration.

The crontab file for the user account running stroom should be edited (`crontab -e`) and set to something like:

```
# stroom logs
* * * * * STROOM_HOME=<path to stroom home> ${STROOM_HOME}/bin/send_to_stroom.sh ${STROOM_HOME}/logs/access STROOM-ACCESS-EVENTS
* * * * * STROOM_HOME=<path to stroom home> ${STROOM_HOME}/bin/send_to_stroom.sh ${STROOM_HOME}/logs/app STROOM-APP-EVENTS
* * * * * STROOM_HOME=<path to stroom home> ${STROOM_HOME}/bin/send_to_stroom.sh ${STROOM_HOME}/logs/user STROOM-USER-EVENTS

# stroom-proxy logs
* * * * * PROXY_HOME=<path to proxy home> ${PROXY_HOME}/bin/send_to_stroom.sh ${PROXY_HOME}/logs/access STROOM_PROXY-ACCESS-EVE
* * * * * PROXY_HOME=<path to proxy home> ${PROXY_HOME}/bin/send_to_stroom.sh ${PROXY_HOME}/logs/app STROOM_PROXY-APP-EVENTS
* * * * * PROXY_HOME=<path to proxy home> ${PROXY_HOME}/bin/send_to_stroom.sh ${PROXY_HOME}/logs/send STROOM_PROXY-SEND-EVENT
* * * * * PROXY_HOME=<path to proxy home> ${PROXY_HOME}/bin/send_to_stroom.sh ${PROXY_HOME}/logs/receive STROOM_PROXY-RECEIVE-EV
```

where the environment specific values are:

- <path to stroom home> - The absolute path to the stroom home, i.e. the location of the `start.sh` script.
- <path to proxy home> - The absolute path to the stroom-proxy home, i.e. the location of the `start.sh` script.
- <datafeed URL> - The URL that the logs will be sent to. This will typically be the nginx host or load balancer and the path will typically be `https://host/datafeeddirect` to bypass the proxy for faster access to the logs.
- <environment> - The environment name that the stroom/proxy is deployed in, e.g. OPS, REF, DEV, etc.
- <key file> - The absolute path to the SSL key file used by curl.
- <cert file> - The absolute path to the SSL certificate file used by curl.
- <CA cert file> - The absolute path to the SSL certificate authority file used by curl.
- <path to log> - The absolute path to a log file to log all the `send_to_stroom.sh` output to.

If your implementation of cron supports environment variables then you can define some of the common values at the top of the crontab file and use them in the entries. `cronie` as used by Centos does not support environment variables in the crontab file but variables can be defined at the line level as has been shown with `STROOM_HOME` and `PROXY_HOME`.

The above crontab entries assume that stroom and stroom-proxy are running on the same host. If there are not then the entries can be split across the hosts accordingly.

2.8.4.1.1 Service host(s)

When deploying stroom/stroom-proxy without stroom you may still be deploying the service stack (nginx and stroom-log-sender) to a host. In this case see [As part of a docker stack](#) below for details of how to configure stroom-log-sender to send the nginx logs.

2.8.4.2 As part of a docker stack

2.8.4.2.1 Crontab

The docker stacks include the stroom-log-sender docker image for sending the logs of all the other containers to stroom. Stroom-log-sender is configured using the crontab file `volumes/stroom-log-sender/conf/crontab.txt`. When the container starts this file will be read. Any variables in it will be substituted with the values from the corresponding environment variables that are present in the container. These common values can be set in the `config/<stack name>.env` file.

As the variables are substituted on container start you will need to restart the container following any configuration change.

2.8.4.2.2 Certificates

The directory `volumes/stroom-log-sender/certs` contains the default client certificates used for the stack. These allow stroom-log-sender to send the log files over SSL which also provides stroom with details of the sender. These will need to be replaced in a production environment.

```
volumes/stroom-log-sender/certs/ca.pem.crt  
volumes/stroom-log-sender/certs/client.pem.crt  
volumes/stroom-log-sender/certs/client.unencrypted.key
```

For a production deployment these will need to be changed, see [Certificates](#)

2.8.5 - MySQL Configuration

Version Information: Created with Stroom v7.0

Last Updated: 2021-06-07

See Also: [MySQL Server Setup](#)

See Also: [MySQL Server Administration \(external link\)](#)

2.8.5.1 General configuration

MySQL is configured via the `.cnf` file which is typically located in one of these locations:

- `/etc/my.cnf`
- `/etc/mysql/my.cnf`
- `$MYSQL_HOME/my.cnf`
- `<data dir>/my.cnf`
- `~/.my.cnf`

2.8.5.1.1 Key configuration properties

- `lower_case_table_names` - This property controls how the tables are stored on the filesystem and the case-sensitivity of table names in SQL. A value of `0` means tables are stored on the filesystem in the case used in CREATE TABLE and sql is case sensitive. This is the default in linux and is the preferred value for deployments of stroom of v7+. A value of `1` means tables are stored on the filesystem in lowercase but sql is case insensitive. [See also \(external link\)](#)
- `max_connections` - The maximum permitted number of simultaneous client connections. For a clustered deployment of stroom, the default value of 151 will typically be too low. Each stroom node will hold a pool of open database connections for its use, therefore with a large number of stroom nodes and a big connection pool the total number of connections can be very large. This property should be set taking into account the values of the stroom properties of the form `*.db.connectionPool.maxPoolSize`. [See also \(external link\)](#)
- `innodb_buffer_pool_size / innodb_buffer_pool_instances` - Controls the amount of memory available to MySQL for caching table/index data. Typically this will be set to 80% of available RAM, assuming MySQL is running on a dedicated host and the total amount of table/index data is greater than 80% of available RAM. Note: `innodb_buffer_pool_size` must be set to a value that is equal to or a multiple of `innodb_buffer_pool_chunk_size * innodb_buffer_pool_instances`. [See also \(external link\)](#)

TODO - Add additional key configuration items

2.8.5.2 Deploying without Docker

When MySQL is deployed without a docker stack then MySQL should be installed and configured according to the MySQL documentation. How MySQL is deployed and configured will depend on the requirements of the environment, e.g. clustered, primary/standby, etc.

2.8.5.3 As part of a docker stack

Where a stroom docker stack includes stroom-all-dbs (MySQL) the MySQL instance is configured via the `.cnf` file. The `.cnf` file is located in `volumes/stroom-all-dbs/conf/stroom-all-dbs.cnf`. This file is read-only to the container and will be read on container start.

2.8.5.3.1 Database initialisation

When the container is started for the first time the database be initialised with the root user account. It will also then run any scripts found in `volumes/stroom-all-dbs/init/stroom`. The scripts in here will be run in alphabetical order. Scripts of the form `.sh`, `.sql`, `.sql.gz` and `.sql.template` are supported.

`.sql.template` files are proprietary to stroom stacks and are just templated `.sql` files. They can contain tags of the form `<<ENV_VAR_NAME>>` which will be replaced with the value of the named environment variable that has been set in the container.

If you need to add additional database users then either add them to `volumes/stroom-all-dbs/init/stroom/001_create_databases.sql.template` or create additional scripts/templates in that directory.

The script that controls this templating is `volumes/stroom-all-dbs/init/000_stroom_init.sh`. This script MUST not have its executable bit set else it will be executed rather than being sourced by the MySQL entry point scripts and will then not work.

3 - User Guide

Reference documentation for how to use Stroom.

3.1 - Application Programming Interfaces (API)

Stroom's public APIs for querying and interacting with all aspects of Stroom.

Stroom has a number of public APIs to allow other systems to interact with Stroom. The APIs are defined by a *Swagger* spec that can be viewed as [json \(external link\)](#) or [yaml \(external link\)](#). A dynamic [user interface \(external link\)](#) is also available for viewing the APIs.

3.1.1 - Query API

An API to allow other systems to query the data held in Stroom.

The Query API uses common request/response models and end points for querying each type of data source held in Stroom. The request/response models are defined in [stroom-query \(external link\)](#).

Currently Stroom exposes a set of query endpoints for the following data source types. Each data source type will have its own endpoint due to differences in the way the data is queried and the restrictions imposed on the query terms. However they all share the same API definition.

- [stroom-index \(external link\)](#) (the Lucene based event index)
- [sqlstatistics \(external link\)](#) (Stroom's own statistics store)

The detailed documentation for the request/responses is contained in the *Swagger* definition linked to above.

3.1.1.1 Common endpoints

The standard query endpoints are

- [/datasource](#)
- [/search](#)
- [/destroy](#)

3.1.1.1.1 datasource

The data source endpoint is used to query Stroom for the details of a data source with a given *docRef*. The details will include such things as the fields available and any restrictions on querying the data.

3.1.1.1.2 search

The search endpoint is used to initiate a search against a data source or to request more data for an active search. A search request can be made using iterative mode, where it will perform the search and then only return the data it has immediately available. Subsequent requests for the same *queryKey* will also return the data immediately available, expecting that more results will have been found by the query. Requesting a search in non-iterative mode will result in the response being returned when the query has completed and all known results have been found.

The SearchRequest model is fairly complicated and contains not only the query terms but also a definition of how the data should be returned. A single SearchRequest can include multiple ResultRequest sections to return the queried data in multiple ways, e.g. as flat data and in an alternative aggregated form.

3.1.1.1.2.1 Stroom as a query builder

Stroom is able to export the json form of a SearchRequest model from its dashboards. This makes the dashboard a useful tool for building a query and the table settings to go with it. You can use the dashboard to define the data source, define the query terms tree and build a table definition (or definitions) to describe how the data should be returned. The, clicking the download icon on the query pane of the dashboard will generate the SearchRequest json which can be immediately used with the /search API or modified to suit.

3.1.1.1.3 destroy

This endpoint is used to kill an active query by supplying the *queryKey* for query in question.

3.2 - Concepts

Describes a number of core concepts involved in using Stroom.

3.2.1 - Streams

The unit of data that Stroom operates on, essentially a bounded stream of data.

Streams can either be created when data is directly POSTed in to Stroom or during the proxy aggregation process. When data is directly POSTed to Stroom the content of the POST will be stored as one Stream. With proxy aggregation multiple files in the proxy repository will/can be aggregated together into a single Stream.

3.2.1.1 Anatomy of a Stream

A Stream is made up of a number of parts of which the raw or cooked data is just one. In addition to the data the Stream can contain a number of other child stream types, e.g. Context and Meta Data.

The hierarchy of a stream is as follows:

- Stream nnn
 - Part [1 to *]
 - Data [1-1]
 - Context [0-1]
 - Meta Data [0-1]

Although all streams conform to the above hierarchy there are three main types of Stream that are used in Stroom:

- Non-segmented Stream - Raw events, Raw Reference
- Segmented Stream - Events, Reference
- Segmented Error Stream - Error

Segmented means that the data has been demarcated into segments or records.

3.2.1.1.1 Child Stream Types

3.2.1.1.1.1 Data

This is the actual data of the stream, e.g. the XML events, raw CSV, JSON, etc.

3.2.1.1.1.2 Context

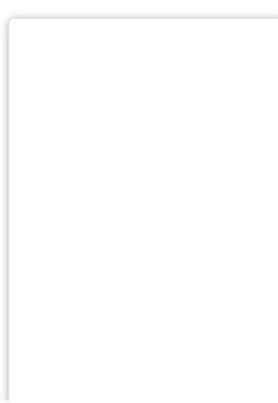
This is additional contextual data that can be sent with the data. Context data can be used for reference data lookups.

3.2.1.1.1.3 Meta Data

This is the data about the Stream (e.g. the feed name, receipt time, user agent, etc.). This meta data either comes from the HTTP headers when the data was POSTed to Stroom or is added by Stroom or Stroom-Proxy on receipt/processing.

3.2.1.1.2 Non-Segmented Stream

The following is a representation of a non-segmented stream with three parts, each with Meta Data and Context child streams.



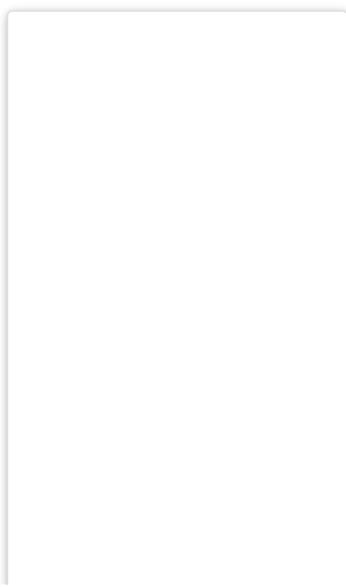


Raw Events and Raw Reference streams contain non-segmented data, e.g. a large batch of CSV, JSON, XML, etc. data. There is no notion of a record/event/segment in the data, it is simply data in any form (including malformed data) that is yet to be processed and demarcated into records/events, for example using a Data Splitter or an XML parser.

The Stream may be single-part or multi-part depending on how it is received. If it is the product of proxy aggregation then it is likely to be multi-part. Each part will have its own context and meta data child streams, if applicable.

3.2.1.1.3 Segmented Stream

The following is a representation of a segmented stream that contains three records (i.e events) and has Meta Data and Context child streams.

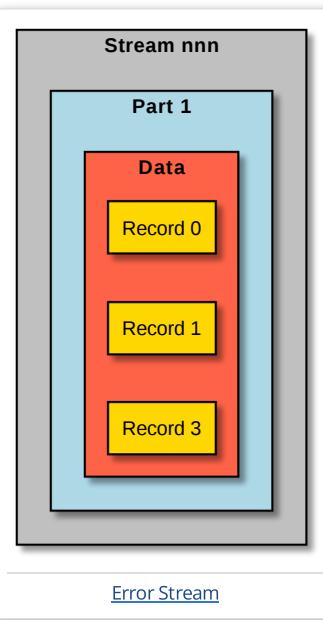




[Segmented Stream](#)

Cooked Events and Reference data are forms of segmented data. The raw data has been parsed and split into records/events and the resulting data is stored in a way that allows Stroom to know where each record/event starts/ends. These streams only have a single part.

3.2.1.1.4 Error Stream



[Error Stream](#)

Error streams are similar to segmented Event/Reference streams in that they are single-part and have demarcated records (where each error/warning/info message is a record). Error streams do not have any Meta Data or Context child streams.

3.3 - Dashboards

The means of displaying and visualising the results of queries.

3.3.1 - Dashboard Expressions

Expression language used to manipulate data on Stroom Dashboards.

Expressions can be used to manipulate data on Stroom Dashboards.

Each function has a name, and some have additional aliases.

In some cases, functions can be nested. The return value for some functions being used as the arguments for other functions.

The arguments to functions can either be other functions, literal values, or they can refer to fields on the input data using the field reference \${val} syntax.

Aggregate Functions	String Functions	Mathematics Functions	Type Checking Functions
<ul style="list-style-type: none">• Average• Count• Count Groups• Count Unique• Joining• Max• Min• Standard Deviation• Sum• Variance	<ul style="list-style-type: none">• Concat• CurrentUser• Decode• DecodeUrl• EncodeUrl• Exclude• Hash• Include• Index Of• Last Index Of• Lower Case• Match• Query Param• Query Params• Replace• String Length• Substring• Substring After• Substring Before• Upper Case	<ul style="list-style-type: none">• Add• Average• Divide• Max• Min• Modulo• Multiply• Negate• Power• Random• Subtract• Sum	<ul style="list-style-type: none">• Is Boolean• Is Double• Is Error• Is Integer• Is Long• Is Null• Is Number• Is String• Is Value• Type Of
Link Functions	Cast Functions	Date Functions	Logic Functions
<ul style="list-style-type: none">• Annotation• Dashboard• Data• Link• Stepping	<ul style="list-style-type: none">• To Boolean• To Double• To Integer• To Long• ToString	<ul style="list-style-type: none">• Format Date• Parse Date• Ceiling Functions• Floor Functions• Round Functions	<ul style="list-style-type: none">• Equals• Greater Than• Greater Than or Equal To• If• Less Than• Less Than or Equal To• Not
Rounding Functions	Selection Functions	URI Functions	Value Functions
<ul style="list-style-type: none">• Ceiling• Floor	<ul style="list-style-type: none">• Any• Bottom	<ul style="list-style-type: none">• extractAuthorityFromUri	<ul style="list-style-type: none">• Err• False

- [Round](#)
- [First](#)
- [Last](#)
- [Nth](#)
- [Top](#)
- [extractFragmentFromUri](#)
- [extractHostFromUri](#)
- [extractPathFromUri](#)
- [extractPortFromUri](#)
- [extractQueryFromUri](#)
- [extractSchemeFromUri](#)
- [extractSchemeSpecificPartFromUri](#)
- [extractUserInfoFromUri](#)
- [Null](#)
- [True](#)

3.3.1.1 - Aggregate Functions

Functions that produce aggregates over multiple data points.

3.3.1.1.1 Average

Takes an average value of the arguments

```
average(arg)  
mean(arg)
```

Examples

```
average(${val})  
${val} = [10, 20, 30, 40]  
> 25  
  
mean(${val})  
${val} = [10, 20, 30, 40]  
> 25
```

3.3.1.1.2 Count

Counts the number of records that are passed through it. Doesn't take any notice of the values of any fields.

```
count()
```

Examples

```
Supplying 3 values...  
  
count()  
> 3
```

3.3.1.1.3 Count Groups

This is used to count the number of unique values where there are multiple group levels. For Example, a data set grouped as follows

1. Group by Name
2. Group by Type

A groupCount could be used to count the number of distinct values of 'type' for each value of 'name'

3.3.1.1.4 Count Unique

This is used to count the number of unique values passed to the function where grouping is used to aggregate values in other columns. For Example, a data set grouped as follows

1. Group by Name
2. Group by Type

countUnique() could be used to count the number of distinct values of 'type' for each value of 'name'

Examples

```
countUnique(${val})
${val} = ['bill', 'bob', 'fred', 'bill']
> 3
```

3.3.1.1.5 Joining

Concatenates all values together into a single string. If a delimiter is supplied then the delimiter is placed between each concatenated string. If a limit is supplied then it will only concatenate up to limit values.

```
joining(values)
joining(values, delimiter)
joining(values, delimiter, limit)
```

Examples

```
joining(${val}, ', ')
${val} = ['bill', 'bob', 'fred', 'bill']
> 'bill, bob, fred, bill'
```

3.3.1.1.6 Max

Determines the maximum value given in the args

```
max(arg)
```

Examples

```
max(${val})
${val} = [100, 30, 45, 109]
> 109

# They can be nested
max(max(${val}), 40, 67, 89)
${val} = [20, 1002]
> 1002
```

3.3.1.1.7 Min

Determines the minimum value given in the args

```
min(arg)
```

Examples

```
min(${val})
${val} = [100, 30, 45, 109]
> 30
```

They can be nested

```
min(max(${val}), 40, 67, 89)
${val} = [20, 1002]
> 20
```

3.3.1.1.8 Standard Deviation

Calculate the standard deviation for a set of input values.

```
stDev(arg)
```

Examples

```
round(stDev(${val}))
${val} = [600, 470, 170, 430, 300]
> 147
```

3.3.1.1.9 Sum

Sums all the arguments together

```
sum(arg)
```

Examples

```
sum(${val})
${val} = [89, 12, 3, 45]
> 149
```

3.3.1.1.10 Variance

Calculate the variance of a set of input values.

```
variance(arg)
```

Examples

```
variance(${val})
${val} = [600, 470, 170, 430, 300]
> 21704
```

3.3.1.2 - Cast Functions

A set of functions for converting between different data types or for working with data types.

3.3.1.2.1 To Boolean

Attempts to convert the passed value to a *boolean* data type.

```
toBoolean(arg1)
```

Examples:

```
toBoolean(1)
> true
toBoolean(0)
> false
toBoolean('true')
> true
toBoolean('false')
> false
```

3.3.1.2.2 To Double

Attempts to convert the passed value to a *double* data type.

```
toDouble(arg1)
```

Examples:

```
toDouble('1.2')
> 1.2
```

3.3.1.2.3 To Integer

Attempts to convert the passed value to a *integer* data type.

```
toInteger(arg1)
```

Examples:

```
toInteger('1')
> 1
```

3.3.1.2.4 To Long

Attempts to convert the passed value to a *long* data type.

```
toLong(arg1)
```

Examples:

```
toLong('1')
> 1
```

3.3.1.2.5 To String

Attempts to convert the passed value to a *string* data type.

```
toString(arg1)
```

Examples:

```
toString(1.2)
> '1.2'
```

3.3.1.3 - Date Functions

Functions for manipulating dates and times.

3.3.1.3.1 Parse Date

Parse a date and return a long number of milliseconds since the epoch.

```
parseDate(aString)
parseDate(aString, pattern)
parseDate(aString, pattern, timeZone)
```

Example

```
parseDate('2014 02 22', 'yyyy MM dd', '+0400')
> 1393012800000
```

3.3.1.3.2 Format Date

Format a date supplied as milliseconds since the epoch.

```
formatDate(aLong)
formatDate(aLong, pattern)
formatDate(aLong, pattern, timeZone)
```

Example

```
formatDate(1393071132888, 'yyyy MM dd', '+1200')
> '2014 02 23'
```

3.3.1.3.3 Ceiling Year/Month/Day/Hour/Minute/Second

```
ceilingYear(args...)
ceilingMonth(args...)
ceilingDay(args...)
ceilingHour(args...)
ceilingMinute(args...)
ceilingSecond(args...)
```

Examples

```
ceilingSecond("2014-02-22T12:12:12.888Z"
> "2014-02-22T12:12:13.000Z"
ceilingMinute("2014-02-22T12:12:12.888Z"
> "2014-02-22T12:13:00.000Z"
ceilingHour("2014-02-22T12:12:12.888Z"
> "2014-02-22T13:00:00.000Z"
ceilingDay("2014-02-22T12:12:12.888Z"
> "2014-02-23T00:00:00.000Z"
ceilingMonth("2014-02-22T12:12:12.888Z"
> "2014-03-01T00:00:00.000Z"
ceilingYear("2014-02-22T12:12:12.888Z"
> "2015-01-01T00:00:00.000Z"
```

3.3.1.3.4 Floor Year/Month/Day/Hour/Minute/Second

```
floorYear(args...)
floorMonth(args...)
floorDay(args...)
floorHour(args...)
floorMinute(args...)
floorSecond(args...)
```

Examples

```
floorSecond("2014-02-22T12:12:12.888Z"
> "2014-02-22T12:12:12.000Z"
floorMinute("2014-02-22T12:12:12.888Z"
> "2014-02-22T12:12:00.000Z"
floorHour("2014-02-22T12:12:12.888Z"
> 2014-02-22T12:00:00.000Z"
floorDay("2014-02-22T12:12:12.888Z"
> "2014-02-22T00:00:00.000Z"
floorMonth("2014-02-22T12:12:12.888Z"
> "2014-02-01T00:00:00.000Z"
floorYear("2014-02-22T12:12:12.888Z"
> "2014-01-01T00:00:00.000Z"
```

3.3.1.3.5 Round Year/Month/Day/Hour/Minute/Second

```
roundYear(args...)
roundMonth(args...)
roundDay(args...)
roundHour(args...)
roundMinute(args...)
roundSecond(args...)
```

Examples

```
roundSecond("2014-02-22T12:12:12.888Z")
> "2014-02-22T12:12:13.000Z"
roundMinute("2014-02-22T12:12:12.888Z")
> "2014-02-22T12:12:00.000Z"
roundHour("2014-02-22T12:12:12.888Z"
> "2014-02-22T12:00:00.000Z"
roundDay("2014-02-22T12:12:12.888Z"
> "2014-02-23T00:00:00.000Z"
roundMonth("2014-02-22T12:12:12.888Z"
> "2014-03-01T00:00:00.000Z"
roundYear("2014-02-22T12:12:12.888Z"
> "2014-01-01T00:00:00.000Z"
```

3.3.1.4 - Link Functions

Functions for linking to other screens in Stroom and/or to particular sets of data.

3.3.1.4.1 Annotation

A helper function to make forming links to annotations easier than using [Link](#). The Annotation function allows you to create a link to open the Annotation editor, either to view an existing annotation or to begin creating one with pre-populated values.

```
annotation(text, annotationId)
annotation(text, annotationId, [streamId, eventId, title, subject, status, assignedTo, comment])
```

If you provide just the *text* and an *annotationId* then it will produce a link that opens an existing annotation with the supplied ID in the Annotation Edit dialog.

Example

```
annotation('Open annotation', ${annotation:id})
> [Open annotation](?annotationId=1234){annotation}
annotation('Create annotation', '', ${StreamId}, ${EventId})
> [Create annotation](?annotationId=&streamId=1234&eventId=45){annotation}
annotation('Escalate', '', ${StreamId}, ${EventId}, 'Escalation', 'Triage required')
> [Escalate](?annotationId=&streamId=1234&eventId=45&title=Escalation&subject=Triage%20required){annotation}
```

If you don't supply an *annotationId* then the link will open the Annotation Edit dialog pre-populated with the optional arguments so that an annotation can be created. If the *annotationId* is not provided then you must provide a *streamId* and an *eventId*. If you don't need to pre-populate a value then you can use '' or null() instead.

Example

```
annotation('Create suspect event annotation', null(), 123, 456, 'Suspect Event', null(), 'assigned', 'jbloggs')
> [Create suspect event annotation](?streamId=123&eventId=456&title=Suspect%20Event&assignedTo=jbloggs){annotation}
```

3.3.1.4.2 Dashboard

A helper function to make forming links to dashboards easier than using [Link](#).

```
dashboard(text, uuid)
dashboard(text, uuid, params)
```

Example

```
dashboard('Click Here', 'e177cf16-da6c-4c7d-a19c-09a201f5a2da')
> [Click Here](?uuid=e177cf16-da6c-4c7d-a19c-09a201f5a2da){dashboard}
dashboard('Click Here', 'e177cf16-da6c-4c7d-a19c-09a201f5a2da', 'userId=user1')
> [Click Here](?uuid=e177cf16-da6c-4c7d-a19c-09a201f5a2da&params=userId%3Duser1){dashboard}
```

3.3.1.4.3 Data

Creates a link to open a source for data for viewing.

```
data(text, id, partNo, [recordNo, lineFrom, colFrom, lineTo, colTo, viewType, displayType])
```

viewType can be one of:

- preview : Display the data as a formatted preview of a limited portion of the data.

- `source` : Display the un-formatted data in its original form with the ability to navigate around all of the data source.

`displayType` can be one of:

- `dialog` : Open as a modal popup dialog.
- `tab` : Open as a top level tab within the Stroom browser tab.

Example

```
data('Quick View', ${StreamId}, 1)
> [Quick View]?id=1234&&partNo=1)
```

3.3.1.4.4 Link

Create a string that represents a hyperlink for display in a dashboard table.

```
link(url)
link(text, url)
link(text, url, type)
```

Example

```
link('http://www.somehost.com/somepath')
> [http://www.somehost.com/somepath](http://www.somehost.com/somepath)
link('Click Here', 'http://www.somehost.com/somepath')
> [Click Here](http://www.somehost.com/somepath)
link('Click Here', 'http://www.somehost.com/somepath', 'dialog')
> [Click Here](http://www.somehost.com/somepath){dialog}
link('Click Here', 'http://www.somehost.com/somepath', 'dialog|Dialog Title')
> [Click Here](http://www.somehost.com/somepath){dialog|Dialog Title}
```

Type can be one of:

- `dialog` : Display the content of the link URL within a stroom popup dialog.
- `tab` : Display the content of the link URL within a stroom tab.
- `browser` : Display the content of the link URL within a new browser tab.
- `dashboard` : Used to launch a stroom dashboard internally with parameters in the URL.

If you wish to override the default title or URL of the target link in either a tab or dialog you can. Both `dialog` and `tab` types allow titles to be specified after a `|`, e.g. `dialog|My Title`.

3.3.1.4.5 Stepping

Open the *Stepping* tab for the requested data source.

```
stepping(text, id)
stepping(text, id, partNo)
stepping(text, id, partNo, recordNo)
```

Example

```
stepping('Click here to step', ${StreamId})
> [Click here to step]{?id=1}
```

3.3.1.5 - Logic Funtions

3.3.1.5.1 Equals

Evaluates if arg1 is equal to arg2

```
arg1 = arg2
equals(arg1, arg2)
```

Examples

```
'foo' = 'bar'
> false
'foo' = 'foo'
> true
51 = 50
> false
50 = 50
> true

equals('foo', 'bar')
> false
equals('foo', 'foo')
> true
equals(51, 50)
> false
equals(50, 50)
> true
```

Note that `equals` cannot be applied to `null` and `error` values, e.g. `x=null()` or `x=err()`. The [`isNull\(\)`](#) and [`isError\(\)`](#) functions must be used instead.

3.3.1.5.2 Greater Than

Evaluates if arg1 is greater than to arg2

```
arg1 > arg2
greaterThan(arg1, arg2)
```

Examples

```
51 > 50
> true
50 > 50
> false
49 > 50
> false

greaterThan(51, 50)
> true
greaterThan(50, 50)
> false
greaterThan(49, 50)
> false
```

3.3.1.5.3 Greater Than or Equal To

Evaluates if arg1 is greater than or equal to arg2

```
arg1 >= arg2
greaterThanOrEqualTo(arg1, arg2)
```

Examples

```
51 >= 50
> true
50 >= 50
> true
49 >= 50
> false

greaterThanOrEqualTo(51, 50)
> true
greaterThanOrEqualTo(50, 50)
> true
greaterThanOrEqualTo(49, 50)
> false
```

3.3.1.5.4 If

Evaluates the supplied boolean condition and returns one value if true or another if false

```
if(expression, trueReturnValue, falseReturnValue)
```

Examples

```
if(5 < 10, 'foo', 'bar')
> 'foo'
if(5 > 10, 'foo', 'bar')
> 'bar'
if(isNull(null()), 'foo', 'bar')
> 'foo'
```

3.3.1.5.5 Less Than

Evaluates if arg1 is less than to arg2

```
arg1 < arg2
lessThan(arg1, arg2)
```

Examples

```
51 < 50
> false
50 < 50
> false
49 < 50
> true

lessThan(51, 50)
> false
lessThan(50, 50)
> false
lessThan(49, 50)
> true
```

3.3.1.5.6 Less Than or Equal To

Evaluates if arg1 is less than or equal to arg2

```
arg1 <= arg2
lessThanOrEqualTo(arg1, arg2)
```

Examples

```
51 <= 50
> false
50 <= 50
> true
49 <= 50
> true

lessThanOrEqualTo(51, 50)
> false
lessThanOrEqualTo(50, 50)
> true
lessThanOrEqualTo(49, 50)
> true
```

3.3.1.5.7 Not

Inverts boolean values making true, false etc.

```
not(booleanValue)
```

Examples

```
not(5 > 10)
> true
not(5 = 5)
> false
not(false())
> true
```

3.3.1.6 - Mathematics Functions

Standard mathematical functions, such as add subtract, multiple, etc.

3.3.1.6.1 Add

```
arg1 + arg2
```

Or reduce the args by successive addition

```
add(args...)
```

Examples

```
34 + 9  
> 43  
add(45, 6, 72)  
> 123
```

3.3.1.6.2 Average

Takes an average value of the arguments

```
average(args...)  
mean(args...)
```

Examples

```
average(10, 20, 30, 40)  
> 25  
mean(8.9, 24, 1.2, 1008)  
> 260.525
```

3.3.1.6.3 Divide

Divides arg1 by arg2

```
arg1 / arg2
```

Or reduce the args by successive division

```
divide(args...)
```

Examples

```
42 / 7  
> 6  
divide(1000, 10, 5, 2)  
> 10  
divide(100, 4, 3)  
> 8.33
```

3.3.1.6.4 Max

Determines the maximum value given in the args

```
max(args...)
```

Examples

```
max(100, 30, 45, 109)
> 109

# They can be nested
max(max(${val}), 40, 67, 89)
${val} = [20, 1002]
> 1002
```

3.3.1.6.5 Min

Determines the minimum value given in the args

```
min(args...)
```

Examples

```
min(100, 30, 45, 109)
> 30
```

They can be nested

```
min(max(${val}), 40, 67, 89)
${val} = [20, 1002]
> 20
```

3.3.1.6.6 Modulo

Determines the modulus of the dividend divided by the divisor.

```
modulo(dividend, divisor)
```

Examples

```
modulo(100, 30)
> 10
```

3.3.1.6.7 Multiply

Multiples arg1 by arg2

```
arg1 * arg2
```

Or reduce the args by successive multiplication

```
multiply(args...)
```

Examples

```
4 * 5  
> 20  
multiply(4, 5, 2, 6)  
> 240
```

3.3.1.6.8 Negate

Multiplies arg1 by -1

```
negate(arg1)
```

Examples

```
negate(80)  
> -80  
negate(23.33)  
> -23.33  
negate(-9.5)  
> 9.5
```

3.3.1.6.9 Power

Raises arg1 to the power arg2

```
arg1 ^ arg2
```

Or reduce the args by successive raising to the power

```
power(args...)
```

Examples

```
4 ^ 3  
> 64  
power(2, 4, 3)  
> 4096
```

3.3.1.6.10 Random

Generates a random number between 0.0 and 1.0

```
random()
```

Examples

```
random()  
> 0.78  
random()  
> 0.89  
...you get the idea
```

3.3.1.6.11 Subtract

```
arg1 - arg2
```

Or reduce the args by successive subtraction

```
subtract(args...)
```

Examples

```
29 - 8  
> 21  
subtract(100, 20, 34, 2)  
> 44
```

3.3.1.6.12 Sum

Sums all the arguments together

```
sum(args...)
```

Examples

```
sum(89, 12, 3, 45)  
> 149
```

3.3.1.7 - Rounding Functions

Functions for rounding data to a set precision.

These functions require a value, and an optional decimal places. If the decimal places are not given it will give you nearest whole number.

3.3.1.7.1 Ceiling

```
ceiling(value, decimalPlaces<optional>)
```

Examples

```
ceiling(8.4234)
> 9
ceiling(4.56, 1)
> 4.6
ceiling(1.22345, 3)
> 1.223
```

3.3.1.7.2 Floor

```
floor(value, decimalPlaces<optional>)
```

Examples

```
floor(8.4234)
> 8
floor(4.56, 1)
> 4.5
floor(1.2237, 3)
> 1.223
```

3.3.1.7.3 Round

```
round(value, decimalPlaces<optional>)
```

Examples

```
round(8.4234)
> 8
round(4.56, 1)
> 4.6
round(1.2237, 3)
> 1.224
```

3.3.1.8 - Selection Functions

Functions for selecting a sub-set of a set of data.

Selection functions are a form of aggregate function operating on grouped data.

3.3.1.8.1 Any

Selects the first value found in the group that is not `null()` or `err()`. If no explicit ordering is set then the value selected is indeterminate.

```
any(${val})
```

Examples

```
any(${val})
${val} = [10, 20, 30, 40]
> 10
```

3.3.1.8.2 Bottom

Selects the bottom N values and returns them as a delimited string in the order they are read.

```
bottom(${val}, delimiter, limit)
```

Examples

```
bottom(${val}, ', ', 2)
${val} = [10, 20, 30, 40]
> '30, 40'
```

3.3.1.8.3 First

Selects the first value found in the group even if it is `null()` or `err()`. If no explicit ordering is set then the value selected is indeterminate.

```
first(${val})
```

Examples

```
first(${val})
${val} = [10, 20, 30, 40]
> 10
```

3.3.1.8.4 Last

Selects the last value found in the group even if it is `null()` or `err()`. If no explicit ordering is set then the value selected is indeterminate.

```
last(${val})
```

Examples

```
last(${val})  
${val} = [10, 20, 30, 40]  
> 40
```

3.3.1.8.5 Nth

Selects the Nth value in a set of grouped values. If there is no explicit ordering on the field selected then the value returned is indeterminate.

```
nth(${val}, position)
```

Examples

```
nth(${val}, 2)  
${val} = [20, 40, 30, 10]  
> 40
```

3.3.1.8.6 Top

Selects the top N values and returns them as a delimited string in the order they are read.

```
top(${val}, delimiter, limit)
```

Examples

```
top(${val}, ', ', 2)  
${val} = [10, 20, 30, 40]  
> '10, 20'
```

3.3.1.9 - String Functions

Functions for manipulating strings (text data).

3.3.1.9.1 Concat

Appends all the arguments end to end in a single string

```
concat(args...)
```

Example

```
concat('this ', 'is ', 'how ', 'it ', 'works')
> 'this is how it works'
```

3.3.1.9.2 Current User

Returns the username of the user running the query.

```
currentUser()
```

Example

```
currentUser()
> 'jbloggs'
```

3.3.1.9.3 Decode

The arguments are split into 3 parts

1. The input value to test
2. Pairs of regex matchers with their respective output value
3. A default result, if the input doesn't match any of the regexes

```
decode(input, test1, result1, test2, result2, ... testN, resultN, otherwise)
```

It works much like a Java Switch/Case statement

Example

```
decode(${val}, 'red', 'rgb(255, 0, 0)', 'green', 'rgb(0, 255, 0)', 'blue', 'rgb(0, 0, 255)', 'rgb(255, 255, 255)')
${val}='blue'
> rgb(0, 0, 255)
${val}='green'
> rgb(0, 255, 0)
${val}='brown'
> rgb(255, 255, 255) // falls back to the 'otherwise' value
```

in Java, this would be equivalent to

```

String decode(value) {
    switch(value) {
        case "red":
            return "rgb(255, 0, 0)"
        case "green":
            return "rgb(0, 255, 0)"
        case "blue":
            return "rgb(0, 0, 255)"
        default:
            return "rgb(255, 255, 255)"
    }
}

```

```

decode('red')
> 'rgb(255, 0, 0)'

```

3.3.1.9.4 DecodeUrl

Decodes a URL

```

decodeUrl('userId%3Duser1')
> userId=user1

```

3.3.1.9.5 EncodeUrl

Encodes a URL

```

encodeUrl('userId=user1')
> userId%3Duser1

```

3.3.1.9.6 Exclude

If the supplied string matches one of the supplied match strings then return null, otherwise return the supplied string

```

exclude(aString, match...)

```

Example

```

exclude('hello', 'hello', 'hi')
> null
exclude('hi', 'hello', 'hi')
> null
exclude('bye', 'hello', 'hi')
> 'bye'

```

3.3.1.9.7 Hash

Cryptographically hashes a string

```

hash(value)
hash(value, algorithm)
hash(value, algorithm, salt)

```

Example

```
hash(${val}, 'SHA-512', 'mysalt')
> A hashed result...
```

If not specified the `hash()` function will use the `SHA-256` algorithm. Supported algorithms are determined by Java runtime environment.

3.3.1.9.8 Include

If the supplied string matches one of the supplied match strings then return it, otherwise return null

```
include(aString, match...)
```

Example

```
include('hello', 'hello', 'hi')
> 'hello'
include('hi', 'hello', 'hi')
> 'hi'
include('bye', 'hello', 'hi')
> null
```

3.3.1.9.9 Index Of

Finds the first position of the second string within the first

```
indexOf(firstString, secondString)
```

Example

```
indexOf('aa-bb-cc', '-')
> 2
```

3.3.1.9.10 Last Index Of

Finds the last position of the second string within the first

```
lastIndexOf(firstString, secondString)
```

Example

```
lastIndexOf('aa-bb-cc', '-')
> 5
```

3.3.1.9.11 Lower Case

Converts the string to lower case

```
lowerCase(aString)
```

Example

```
lowerCase('Hello DeVeLoPER')
> 'hello developer'
```

3.3.1.9.12 Match

Test an input string using a regular expression to see if it matches

```
match(input, regex)
```

Example

```
match('this', 'this')
> true
match('this', 'that')
> false
```

3.3.1.9.13 Query Param

Returns the value of the requested query parameter.

```
queryParam(paramKey)
```

Examples

```
queryParam('user')
> 'jbloggs'
```

3.3.1.9.14 Query Params

Returns all query parameters as a space delimited string.

```
queryParams()
```

Examples

```
queryParams()
> 'user=jbloggs site=HQ'
```

3.3.1.9.15 Replace

Perform text replacement on an input string using a regular expression to match part (or all) of the input string and a replacement string to insert in place of the matched part

```
replace(input, regex, replacement)
```

Example

```
replace('this', 'is', 'at')
> 'that'
```

3.3.1.9.16 String Length

Takes the length of a string

```
stringLength(aString)
```

Example

```
stringLength('hello')  
> 5
```

3.3.1.9.17 Substring

Take a substring based on start/end index of letters

```
substring(aString, startIndex, endIndex)
```

Example

```
substring('this', 1, 2)  
> 'h'
```

3.3.1.9.18 Substring After

Get the substring from the first string that occurs after the presence of the second string

```
substringAfter(firstString, secondString)
```

Example

```
substringAfter('aa-bb', '-')  
> 'bb'
```

3.3.1.9.19 Substring Before

Get the substring from the first string that occurs before the presence of the second string

```
substringBefore(firstString, secondString)
```

Example

```
substringBefore('aa-bb', '-')  
> 'aa'
```

3.3.1.9.20 Upper Case

Converts the string to upper case

```
upperCase(aString)
```

Example

```
upperCase('Hello DeVeLoPER')
> 'HELLO DEVELOPER'
```

3.3.1.10 - Type Checking Functions

Functions for evaluating the type of a value.

3.3.1.10.1 Is Boolean

Checks if the passed value is a *boolean* data type.

```
isBoolean(arg1)
```

Examples:

```
isBoolean(toBoolean('true'))  
> true
```

3.3.1.10.2 Is Double

Checks if the passed value is a *double* data type.

```
isDouble(arg1)
```

Examples:

```
isDouble(toDouble('1.2'))  
> true
```

3.3.1.10.3 Is Error

Checks if the passed value is an error caused by an invalid evaluation of an expression on passed values, e.g. some values passed to an expression could result in a divide by 0 error. Note that this method must be used to check for `error` as error equality using `x==err()` is not supported.

```
isError(arg1)
```

Examples:

```
isError(toLong('1'))  
> false  
isError(err())  
> true
```

3.3.1.10.4 Is Integer

Checks if the passed value is an *integer* data type.

```
isInteger(arg1)
```

Examples:

```
isInteger(toInteger('1'))  
> true
```

3.3.1.10.5 Is Long

Checks if the passed value is a *long* data type.

```
isLong(arg1)
```

Examples:

```
isLong(toLong('1'))  
> true
```

3.3.1.10.6 Is Null

Checks if the passed value is `null`. Note that this method must be used to check for `null` as null equality using `x=null()` is not supported.

```
isNull(arg1)
```

Examples:

```
isNull(toLong('1'))  
> false  
isNull(null())  
> true
```

3.3.1.10.7 Is Number

Checks if the passed value is a numeric data type.

```
isNumber(arg1)
```

Examples:

```
isNumber(toLong('1'))  
> true
```

3.3.1.10.8 Is String

Checks if the passed value is a *string* data type.

```
isString(arg1)
```

Examples:

```
isString(toString(1.2))  
> true
```

3.3.1.10.9 Is Value

Checks if the passed value is a value data type, e.g. not `null` or `error`.

```
isValue(arg1)
```

Examples:

```
isValue(toLong('1'))  
> true  
isValue(null())  
> false
```

3.3.1.10.10 Type Of

Returns the data type of the passed value as a string.

```
typeOf(arg1)
```

Examples:

```
typeOf('abc')  
> string  
typeOf(toInteger(123))  
> integer  
typeOf(err())  
> error  
typeOf(null())  
> null  
typeOf(toBoolean('false'))  
> false
```

3.3.1.11 - URI Functions

Functions for extracting parts from a Uniform Resource Identifier (URI).

Fields containing a Uniform Resource Identifier (URI) in string form can be queried to extract the URI's individual components of `authority`, `fragment`, `host`, `path`, `port`, `query`, `scheme`, `schemeSpecificPart` and `userInfo`. See either [RFC 2306: Uniform Resource Identifiers \(URI\): Generic Syntax](#) or Java's `java.net.URI` Class for details regarding the components. If any component is not present within the passed URI, then an empty string is returned.

The extraction functions are

- `extractAuthorityFromUri()` - extract the Authority component
- `extractFragmentFromUri()` - extract the Fragment component
- `extractHostFromUri()` - extract the Host component
- `extractPathFromUri()` - extract the Path component
- `extractPortFromUri()` - extract the Port component
- `extractQueryFromUri()` - extract the Query component
- `extractSchemeFromUri()` - extract the Scheme component
- `extractSchemeSpecificPartFromUri()` - extract the Scheme specific part component
- `extractUserInfoFromUri()` - extract the UserInfo component

If the URI is `http://foo:bar@w1.superman.com:8080/very/long/path.html?p1=v1&p2=v2#more-details` the table below displays the extracted components

Expression	Extraction
<code>extractAuthorityFromUri(\${URI})</code>	<code>foo:bar@w1.superman.com:8080</code>
<code>extractFragmentFromUri(\${URI})</code>	<code>more-details</code>
<code>extractHostFromUri(\${URI})</code>	<code>w1.superman.com</code>
<code>extractPathFromUri(\${URI})</code>	<code>/very/long/path.html</code>
<code>extractPortFromUri(\${URI})</code>	<code>8080</code>
<code>extractQueryFromUri(\${URI})</code>	<code>p1=v1&p2=v2</code>
<code>extractSchemeFromUri(\${URI})</code>	<code>http</code>
<code>extractSchemeSpecificPartFromUri(\${URI})</code>	<code>//foo:bar@w1.superman.com:8080/very/long/path.html?p1=v1&p2=v2</code>
<code>extractUserInfoFromUri(\${URI})</code>	<code>foo:bar</code>

3.3.1.11.1 extractAuthorityFromUri

Extracts the Authority component from a URI

```
extractAuthorityFromUri(uri)
```

Example

```
extractAuthorityFromUri('http://foo:bar@w1.superman.com:8080/very/long/path.html?p1=v1&p2=v2#more-details')
> 'foo:bar@w1.superman.com:8080'
```

3.3.1.11.2 extractFragmentFromUri

Extracts the Fragment component from a URI

```
extractFragmentFromUri(uri)
```

Example

```
extractFragmentFromUri('http://foo:bar@w1.superman.com:8080/very/long/path.html?p1=v1&p2=v2#more-details')
> 'more-details'
```

3.3.1.11.3 extractHostFromUri

Extracts the Host component from a URI

```
extractHostFromUri(uri)
```

Example

```
extractHostFromUri('http://foo:bar@w1.superman.com:8080/very/long/path.html?p1=v1&p2=v2#more-details')
> 'w1.superman.com'
```

3.3.1.11.4 extractPathFromUri

Extracts the Path component from a URI

```
extractPathFromUri(uri)
```

Example

```
extractPathFromUri('http://foo:bar@w1.superman.com:8080/very/long/path.html?p1=v1&p2=v2#more-details')
> '/very/long/path.html'
```

3.3.1.11.5 extractPortFromUri

Extracts the Port component from a URI

```
extractPortFromUri(uri)
```

Example

```
extractPortFromUri('http://foo:bar@w1.superman.com:8080/very/long/path.html?p1=v1&p2=v2#more-details')
> '8080'
```

3.3.1.11.6 extractQueryFromUri

Extracts the Query component from a URI

```
extractQueryFromUri(uri)
```

Example

```
extractQueryFromUri('http://foo:bar@w1.superman.com:8080/very/long/path.html?p1=v1&p2=v2#more-details')
> 'p1=v1&p2=v2'
```

3.3.1.11.7 extractSchemeFromUri

Extracts the Scheme component from a URI

```
extractSchemeFromUri(uri)
```

Example

```
extractSchemeFromUri('http://foo:bar@w1.superman.com:8080/very/long/path.html?p1=v1&p2=v2#more-details')
> 'http'
```

3.3.1.11.8 extractSchemeSpecificPartFromUri

Extracts the SchemeSpecificPart component from a URI

```
extractSchemeSpecificPartFromUri(uri)
```

Example

```
extractSchemeSpecificPartFromUri('http://foo:bar@w1.superman.com:8080/very/long/path.html?p1=v1&p2=v2#more-details')
> '//foo:bar@w1.superman.com:8080/very/long/path.html?p1=v1&p2=v2'
```

3.3.1.11.9 extractUserInfoFromUri

Extracts the UserInfo component from a URI

```
extractUserInfoFromUri(uri)
```

Example

```
extractUserInfoFromUri('http://foo:bar@w1.superman.com:8080/very/long/path.html?p1=v1&p2=v2#more-details')
> 'foo:bar'
```

3.3.1.12 - Value Functions

Functions that return a static value.

3.3.1.12.1 Err

Returns *err*

```
err()
```

3.3.1.12.2 False

Returns boolean *false*

```
false()
```

3.3.1.12.3 Null

Returns *null*

```
null()
```

3.3.1.12.4 True

Returns boolean *true*

```
true()
```

3.3.2 - Dictionaries

3.3.2.1 Creating

Right click on a folder in the explorer tree that you want to create a dictionary in. Choose 'New/Dictionary' from the popup menu:

TODO: Fix image

Call the dictionary something like 'My Dictionary' and click OK.

TODO: Fix image

Now just add any search terms you want to the newly created dictionary and click save.

TODO: Fix image

You can add multiple terms.

- Terms on separate lines act as if they are part of an 'OR' expression when used in a search.
- Terms on a single line separated by spaces act as if they are part of an 'AND' expression when used in a search.

3.3.2.2 Using

To perform a search using your dictionary, just choose the newly created dictionary as part of your search expression:

TODO: Fix image

3.3.3 - Direct URLs

Navigating directly to a specific *Stroom* dashboard using a direct URL.

It is possible to navigate directly to a specific *Stroom* dashboard using a direct URL. This can be useful when you have a dashboard that needs to be viewed by users that would otherwise not be using the *Stroom* user interface.

3.3.3.1 URL format

The format for the URL is as follows:

```
https://<HOST>/stroom/dashboard?type=Dashboard&uuid=<DASHBOARD UUID>[&title=<DASHBOARD TITLE>][&params=<DASHBOARD PARAMETERS>]
```

Example:

```
https://localhost/stroom/dashboard?type=Dashboard&uuid=c7c6b03c-5d47-4b8b-b84e-e4dfc6c84a09&title=My%20Dash&params=userId%3DFred%20Bloggs
```

3.3.3.1.1 Host and path

The host and path are typically `https://<HOST>/stroom/dashboard` where `<HOST>` is the hostname/IP for *Stroom*.

3.3.3.1.2 type

`type` is a required parameter and must always be `Dashboard` since we are opening a dashboard.

3.3.3.1.3 uuid

`uuid` is a required parameter where `<DASHBOARD UUID>` is the UUID for the dashboard you want a direct URL to, e.g. `uuid=c7c6b03c-5d47-4b8b-b84e-e4dfc6c84a09`

The UUID for the dashboard that you want to link to can be found by right clicking on the dashboard icon in the explorer tree and selecting Info.

The Info dialog will display something like this and the UUID can be copied from it:

```
DB ID: 4
UUID: c7c6b03c-5d47-4b8b-b84e-e4dfc6c84a09
Type: Dashboard
Name: Stroom Family App Events Dashboard
Created By: INTERNAL
Created On: 2018-12-10T06:33:03.275Z
Updated By: admin
Updated On: 2018-12-10T07:47:06.841Z
```

3.3.3.1.4 title (Optional)

`title` is an optional URL parameter where `<DASHBOARD TITLE>` allows the specification of a specific title for the opened dashboard instead of the default dashboard name.

The inclusion of `${name}` in the title allows the default dashboard name to be used and appended with other values, e.g.

```
'title=${name}%20-%20' + param.name
```

3.3.3.1.5 params (Optional)

`params` is an optional URL parameter where `<DASHBOARD PARAMETERS>` includes any parameters that have been defined for the dashboard in any of the expressions, e.g. `params=userId%3DFred%20Bloggs`

3.3.3.2 Permissions

In order for a user to view a dashboard they will need the necessary permission on the various entities that make up the dashboard.

For a Lucene index query and associated table the following permissions will be required:

- *Read* permission on the *Dashboard* entity.
- *Use* permission on any *Indexe* entities being queried in the dashboard.
- *Use* permission on any *Pipeline* entities set as search extraction *Pipelines* in any of the dashboard's tables.
- *Use* permission on any *XSLT* entities used by the above search extraction *Pipeline* entites.
- *Use* permission on any ancestor pipelines of any of the above search extraction *Pipeline* entites (if applicable).
- *Use* permission on any *Feed* entities that you want the user to be able to see data for.

For a SQL Statistics query and associated table the following permissions will be required:

- *Read* permission on the *Dashboard* entity.
- *Use* permission on the *StatisticStore* entity being queried.

For a visualisation the following permissions will be required:

- *Read* permission on any *Visualiation* entities used in the dashboard.
- *Read* permission on any *Script* entities used by the above *Visualiation* entities.
- *Read* permission on any *Script* entities used by the above *Script* entities.

3.3.4 - Queries

How to query the data in Stroom.

Dashboard queries are created with the query expression builder. The expression builder allows for complex boolean logic to be created across multiple index fields. The way in which different index fields may be queried depends on the type of data that the index field contains.

3.3.4.1 Date Time Fields

Time fields can be queried for times equal, greater than, greater than or equal, less than, less than or equal or between two times.

Times can be specified in two ways:

- Absolute times
- Relative times

3.3.4.1.1 Absolute Times

An absolute time is specified in ISO 8601 date time format, e.g. 2016-01-23T12:34:11.844Z

3.3.4.1.2 Relative Times

In addition to absolute times it is possible to specify times using expressions. Relative time expressions create a date time that is relative to the execution time of the query. Supported expressions are as follows:

- now() - The current execution time of the query.
- second() - The current execution time of the query rounded down to the nearest second.
- minute() - The current execution time of the query rounded down to the nearest minute.
- hour() - The current execution time of the query rounded down to the nearest hour.
- day() - The current execution time of the query rounded down to the nearest day.
- week() - The current execution time of the query rounded down to the first day of the week (Monday).
- month() - The current execution time of the query rounded down to the start of the current month.
- year() - The current execution time of the query rounded down to the start of the current year.

3.3.4.1.3 Adding/Subtracting Durations

With relative times it is possible to add or subtract durations so that queries can be constructed to provide for example, the last week of data, the last hour of data etc.

To add/subtract a duration from a query term the duration is simply appended after the relative time, e.g.

`now() + 2d`

Multiple durations can be combined in the expression, e.g.

`now() + 2d - 10h`

`now() + 2w - 1d10h`

Durations consist of a number and duration unit. Supported duration units are:

- s - Seconds
- m - Minutes
- h - Hours
- d - Days
- w - Weeks
- M - Months

- y - Years

Using these durations a query to get the last weeks data could be as follows:

```
between now() - 1w and now()
```

Or midnight a week ago to midnight today:

```
between day() - 1w and day()
```

Or if you just wanted data for the week so far:

```
greater than week()
```

Or all data for the previous year:

```
between year() - 1y and year()
```

Or this year so far:

```
greater than year()
```

3.4 - Data Retention

Controlling the purging/retention of old data.

By default Stroom will retain all the data it ingests and creates forever. It is likely that storage constraints/costs will mean that data needs to be deleted after a certain time. It is also likely that certain types of data may need to be kept for longer than other types.

3.4.1 Rules

Stroom allows for a set of data retention policy rules to be created to control at a fine grained level what data is deleted and what is retained.

The data retention rules are accessible by selecting *Data Retention* from the *Tools* menu. On first use the *Rules* tab of the *Data Retention* screen will show a single rule named *Default Retain All Forever Rule*. This is the implicit rule in stroom that retains all data and is always in play unless another rule overrides it. This rule cannot be edited, moved or removed.

3.4.1.1 Rule Precedence

Rules have a precedence, with a lower rule number being a higher priority. When running the data retention job, Stroom will look at each stream held on the system and the retention policy of the first rule (starting from the lowest numbered rule) that matches that stream will apply. Once a matching rule is found all other rules with higher rule numbers (lower priority) are ignored. For example if rule 1 says to retain streams from feed `x-EVENTS` for 10 years and rule 2 says to retain streams from feeds `*-EVENTS` for 1 year then rule 1 would apply to streams from feed `x-EVENTS` and they would be kept for 10 years, but rule 2 would apply to feed `y-EVENTS` and they would only be kept for 1 year. Rules are re-numbered as new rules are added/deleted/moved.

3.4.1.2 Creating a Rule

To create a rule do the following:

1. Click the  icon to add a new rule.
2. Edit the expression to define the data that the rule will match on.
3. Provide a name for the rule to help describe what its purpose is.
4. Set the retention period for data matching this rule, i.e *Forever* or a set time period.

The new rule will be added at the top of the list of rules, i.e. with the highest priority. The  and  icons can be used to change the priority of the rule.

Rules can be enabled/disabled by clicking the checkbox next to the rule.

Changes to rules will not take effect until the  icon is clicked.

Rules can also be deleted () and copied ()

3.4.2 Impact Summary

When you have a number of complex rules it can be difficult to determine what data will actually be deleted next time the *Policy Based Data Retention* job runs. To help with this, Stroom has the *Impact Summary* tab that acts as a dry run for the active rules. The impact summary provides a count of the number of streams that will be deleted broken down by rule, stream type and feed name. On large systems with lots of data or complex rules, this query may take a long time to run.

The impact summary operates on the current state of the rules on the *Rules* tab whether saved or un-saved. This allows you to make a change to the rules and test its impact before saving it.

3.5 - Data Splitter

Data Splitter was created to transform text into XML. The XML produced is basic but can be processed further with XSLT to form any desired XML output.

Data Splitter works by using regular expressions to match a region of content or tokenizers to split content. The whole match or match group can then be output or passed to other expressions to further divide the matched data.

The root `<dataSplitter>` element controls the way content is read and buffered from the source. It then passes this content on to one or more child expressions that attempt to match the content. The child expressions attempt to match content one at a time in the order they are specified until one matches. The matching expression then passes the content that it has matched to other elements that either emit XML or apply other expressions to the content matched by the parent.

This process of content supply, match, (supply, match)*, emit is best illustrated in a simple CSV example. Note that the elements and attributes used in all examples are explained in detail in the [element reference](#).

3.5.1 - Simple CSV Example

The following CSV data will be split up into separate fields using Data Splitter.

```
01/01/2010,00:00:00,192.168.1.100,SOMEHOST.SOMEWHERE.COM,user1,logon,  
01/01/2010,00:01:00,192.168.1.100,SOMEHOST.SOMEWHERE.COM,user1,create,c:\test.txt  
01/01/2010,00:02:00,192.168.1.100,SOMEHOST.SOMEWHERE.COM,user1,logoff,
```

The first thing we need to do is match each record. Each record in a CSV file is delimited by a new line character. The following configuration will split the data into records using '\n' as a delimiter:

```
<?xml version="1.0" encoding="UTF-8"?>  
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3  
!-- Match each line using a new line character as the delimiter -->  
<split delimiter="\n"/>  
</dataSplitter>
```

In the above example the 'split' tokenizer matches all of the supplied content up to the end of each line ready to pass each line of content on for further treatment.

We can now add a `<group>` element within `<split>` to take content matched by the tokenizer.

```
<?xml version="1.0" encoding="UTF-8"?>  
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3  
!-- Match each line using a new line character as the delimiter -->  
<split delimiter="\n">  
  
    <!-- Take the matched line (using group 1 ignores the delimiters,  
        without this each match would include the new line character) -->  
    <group value="$1">  
  
        </group>  
    </split>  
</dataSplitter>
```

The `<group>` within the `<split>` chooses to take the content from the `<split>` without including the new line '\n' delimiter by using match group 1, see [expression match references](#) for details.

```
01/01/2010,00:00:00,192.168.1.100,SOMEHOST.SOMEWHERE.COM,user1,logon,
```

The content selected by the `<group>` from its parent match can then be passed onto sub expressions for further matching:

```

<?xml version="1.0" encoding="UTF-8"?>
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3

<!-- Match each line using a new line character as the delimiter -->
<split delimiter="\n">

    <!-- Take the matched line (using group 1 ignores the delimiters,
        without this each match would include the new line character) -->
    <group value="$1">

        <!-- Match each value separated by a comma as the delimiter -->
        <split delimiter="," ">

            </split>
        </group>
    </split>
</dataSplitter>

```

In the above example the additional `<split>` element within the `<group>` will match the content provided by the group repeatedly until it has used all of the group content.

The content matched by the inner `<split>` element can be passed to a `<data>` element to emit XML content.

```

<?xml version="1.0" encoding="UTF-8"?>
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3

<!-- Match each line using a new line character as the delimiter -->
<split delimiter="\n">

    <!-- Take the matched line (using group 1 ignores the delimiters,
        without this each match would include the new line character) -->
    <group value="$1">

        <!-- Match each value separated by a comma as the delimiter -->
        <split delimiter="," ">

            <!-- Output the value from group 1 (as above using group 1
                ignores the delimiters, without this each value would include
                the comma) -->
            <data value="$1" />
        </split>
    </group>
</split>
</dataSplitter>

```

In the above example each match from the inner `<split>` is made available to the inner `<data>` element that chooses to output content from match group 1, see [expression match references](#) for details.

The above configuration results in the following XML output for the whole input:

```
<?xml version="1.0" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="records:2 file:///records-v2
<record>
<data value="01/01/2010" />
<data value="00:00:00" />
<data value="192.168.1.100" />
<data value="SOMEHOST.SOMEWHERE.COM" />
<data value="user1" />
<data value="logon" />
</record>
<record>
<data value="01/01/2010" />
<data value="00:01:00" />
<data value="192.168.1.100" />
<data value="SOMEHOST.SOMEWHERE.COM" />
<data value="user1" />
<data value="create" />
<data value="c:\test.txt" />
</record>
<record>
<data value="01/01/2010" />
<data value="00:02:00" />
<data value="192.168.1.100" />
<data value="SOMEHOST.SOMEWHERE.COM" />
<data value="user1" />
<data value="logoff" />
</record>
</records>
```

3.5.2 - Simple CSV example with heading

In addition to referencing content produced by a parent element it is often desirable to store content and reference it later. The following example of a CSV with a heading demonstrates how content can be stored in a variable and then referenced later on.

3.5.2.1 Input

This example will use a similar input to the one in the previous CSV example but also adds a heading line.

```
Date,Time,IPAddress,HostName,User,EventType,Detail  
01/01/2010,00:00:00,192.168.1.100,SOMEHOST.SOMEWHERE.COM,user1,logon,  
01/01/2010,00:01:00,192.168.1.100,SOMEHOST.SOMEWHERE.COM,user1,create,c:\test.txt  
01/01/2010,00:02:00,192.168.1.100,SOMEHOST.SOMEWHERE.COM,user1,logoff,
```

3.5.2.2 Configuration

```
<?xml version="1.0" encoding="UTF-8"?>  
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3  
!-- Match heading line (note that maxMatch="1" means that only the  
first line will be matched by this splitter) -->  
<split delimiter="\n" maxMatch="1">  
  
    <!-- Store each heading in a named list -->  
    <group>  
        <split delimiter="," />  
        <var id="heading" />  
    </split>  
    </group>  
</split>  
  
    <!-- Match each record -->  
<split delimiter="\n">  
  
        <!-- Take the matched line -->  
        <group value="$1">  
  
            <!-- Split the line up -->  
            <split delimiter="," />  
  
            <!-- Output the stored heading for each iteration and the value  
            from group 1 -->  
            <data name="$heading$1" value="$1" />  
        </split>  
    </group>  
</split>  
</dataSplitter>
```

3.5.2.3 Output

```
<?xml version="1.0" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="records:2 file:///records-v2
<record>
<data name="Date" value="01/01/2010" />
<data name="Time" value="00:00:00" />
<data name="IPAddress" value="192.168.1.100" />
<data name="HostName" value="SOMEHOST.SOMEWHERE.COM" />
<data name="User" value="user1" />
<data name="EventType" value="logon" />
</record>
<record>
<data name="Date" value="01/01/2010" />
<data name="Time" value="00:01:00" />
<data name="IPAddress" value="192.168.1.100" />
<data name="HostName" value="SOMEHOST.SOMEWHERE.COM" />
<data name="User" value="user1" />
<data name="EventType" value="create" />
<data name="Detail" value="c:\test.txt" />
</record>
<record>
<data name="Date" value="01/01/2010" />
<data name="Time" value="00:02:00" />
<data name="IPAdress" value="192.168.1.100" />
<data name="HostName" value="SOMEHOST.SOMEWHERE.COM" />
<data name="User" value="user1" />
<data name="EventType" value="logoff" />
</record>
</records>
```

3.5.3 - Complex example with regex and user defined names

The following example uses a real world Apache log and demonstrates the use of regular expressions rather than simple 'split' tokenizers. The usage and structure of regular expressions is outside of the scope of this document but Data Splitter uses Java's standard regular expression library that is POSIX compliant and documented in numerous places.

This example also demonstrates that the names and values that are output can be hard coded in the absence of field name information to make XSLT conversion easier later on. Also shown is that any match can be divided into further fields with additional expressions and the ability to nest data elements to provide structure if needed.

3.5.3.1 Input

```
192.168.1.100 - "-" [12/Jul/2012:11:57:07 +0000] "GET /doc.htm HTTP/1.1" 200 4235 "-" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; en-US; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; .NET CLR 1.1.4322)"  
192.168.1.100 - "-" [12/Jul/2012:11:57:07 +0000] "GET /default.css HTTP/1.1" 200 3494 "http://some.server:8080/doc.htm" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/28.0.1500.72 Safari/537.36"
```

3.5.3.2 Configuration

```

<?xml version="1.0" encoding="UTF-8"?>
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3

<!--
Standard Apache Format

%h - host name should be ok without quotes
%l - Remote logname (from identd, if supplied). This will return a dash unless IdentityCheck is set On.
\"%u\" - user name should be quoted to deal with DNS
%t - time is added in square brackets so is contained for parsing purposes
\"%r\" - URL is quoted
%>s - Response code doesn't need to be quoted as it is a single number
%b - The size in bytes of the response sent to the client
\"%{Referer}i\" - Referrer is quoted so that's ok
\"%{User-Agent}i\" - User agent is quoted so also ok

LogFormat "%h %l \"%u\" %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
-->

<!-- Match line -->
<split delimiter="\n">
  <group value="$1">

    <!-- Provide a regular expression for the whole line with match
    groups for each field we want to split out -->
    <regex pattern="^([^\s]+) ([^\s]+)\s+([^\s;]+);([^\s;]+)\s+([^\s;]+);([^\s;]+)\s+([^\s;]+);([^\s;]+)\s+([^\s;]+)">
      <data name="host" value="$1" />
      <data name="log" value="$2" />
      <data name="user" value="$3" />
      <data name="time" value="$4" />
      <data name="url" value="$5">

      <!-- Take the 5th regular expression group and pass it to
      another expression to divide into smaller components -->
      <group value="$5">
        <regex pattern="^([^\s]+) ([^\s]+) ([^\s]*)/([^\s]*)">
          <data name="httpMethod" value="$1" />
          <data name="url" value="$2" />
          <data name="protocol" value="$3" />
          <data name="version" value="$4" />
        </regex>
      </group>
    </data>
    <data name="response" value="$6" />
    <data name="size" value="$7" />
    <data name="referrer" value="$8" />
    <data name="userAgent" value="$9" />
  </regex>
</group>
</split>
</dataSplitter>

```

3.5.3.3 Output

```
<?xml version="1.0" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="records:2 file:///records-v2
<record>
  <data name="host" value="192.168.1.100" />
  <data name="log" value="-" />
  <data name="user" value="-" />
  <data name="time" value="12/Jul/2012:11:57:07 +0000" />
  <data name="url" value="GET /doc.htm HTTP/1.1">
    <data name="httpMethod" value="GET" />
    <data name="url" value="/doc.htm" />
    <data name="protocol" value="HTTP" />
    <data name="version" value="1.1" />
  </data>
  <data name="response" value="200" />
  <data name="size" value="4235" />
  <data name="referrer" value="-" />
  <data name="userAgent" value="Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)" />
</record>
<record>
  <data name="host" value="192.168.1.100" />
  <data name="log" value="-" />
  <data name="user" value="-" />
  <data name="time" value="12/Jul/2012:11:57:07 +0000" />
  <data name="url" value="GET /default.css HTTP/1.1">
    <data name="httpMethod" value="GET" />
    <data name="url" value="/default.css" />
    <data name="protocol" value="HTTP" />
    <data name="version" value="1.1" />
  </data>
  <data name="response" value="200" />
  <data name="size" value="3494" />
  <data name="referrer" value="http://some.server:8080/doc.htm" />
  <data name="userAgent" value="Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)" />
</record>
</records>
```

3.5.4 - Multi Line Example

Example multi line file where records are split over many lines. There are various ways this data could be treated but this example forms a record from data created when some fictitious query starts plus the subsequent query results.

3.5.4.1 Input

```
09/07/2016 14:49:36 User = user1
09/07/2016 14:49:36 Query = some query

09/07/2016 16:34:40 Results:
09/07/2016 16:34:40 Line 1: result1
09/07/2016 16:34:40 Line 2: result2
09/07/2016 16:34:40 Line 3: result3
09/07/2016 16:34:40 Line 4: result4

09/07/2009 16:35:21 User = user2
09/07/2009 16:35:21 Query = some other query

09/07/2009 16:45:36 Results:
09/07/2009 16:45:36 Line 1: result1
09/07/2009 16:45:36 Line 2: result2
09/07/2009 16:45:36 Line 3: result3
09/07/2009 16:45:36 Line 4: result4
```

3.5.4.2 Configuration

```

<?xml version="1.0" encoding="UTF-8"?>
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3

<!-- Match each record. We want to treat the query and results as a single event so match the two sets of data separated by a
<regex pattern="\n*(.*\n)+?\n(.*\n)+?\n)|\n*(.*\n?)+">

<group>

    <!-- Split the record into query and results -->
    <regex pattern="(.*?)\n\n(.*)" dotAll="true">

        <!-- Create a data element to output query data -->
        <data name="query">
            <group value="$1">

                <!-- We only want to output the date and time from the first line. -->
                <regex pattern="([^\t]*)\t([^\t]*)[\t]*([^\=:]*[=:]*(.*)" maxMatch="1">
                    <data name="date" value="$1" />
                    <data name="time" value="$2" />
                    <data name="$3" value="$4" />
                </regex>

                <!-- Output all other values -->
                <regex pattern="([^\t]*)\t([^\t]*)[\t]*([^\=:]*[=:]*(.*)">
                    <data name="$3" value="$4" />
                </regex>
            </group>
        </data>

        <!-- Create a data element to output result data -->
        <data name="results">
            <group value="$2">

                <!-- We only want to output the date and time from the first line. -->
                <regex pattern="([^\t]*)\t([^\t]*)[\t]*([^\=:]*[=:]*(.*)" maxMatch="1">
                    <data name="date" value="$1" />
                    <data name="time" value="$2" />
                    <data name="$3" value="$4" />
                </regex>

                <!-- Output all other values -->
                <regex pattern="([^\t]*)\t([^\t]*)[\t]*([^\=:]*[=:]*(.*)">
                    <data name="$3" value="$4" />
                </regex>
            </group>
        </data>

    </regex>
</group>
</regex>
</dataSplitter>

```

3.5.4.3 Output

```
<?xml version="1.0" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="records:2 file://records-v2
</record>
<data name="query">
<data name="date" value="09/07/2016" />
<data name="time" value="14:49:36" />
<data name="User" value="user1" />
<data name="Query" value="some query" />
</data>
<data name="results">
<data name="date" value="09/07/2016" />
<data name="time" value="16:34:40" />
<data name="Results" />
<data name="Line 1" value="result1" />
<data name="Line 2" value="result2" />
<data name="Line 3" value="result3" />
<data name="Line 4" value="result4" />
</data>
</record>
<record>
<data name="query">
<data name="date" value="09/07/2016" />
<data name="time" value="16:35:21" />
<data name="User" value="user2" />
<data name="Query" value="some other query" />
</data>
<data name="results">
<data name="date" value="09/07/2016" />
<data name="time" value="16:45:36" />
<data name="Results" />
<data name="Line 1" value="result1" />
<data name="Line 2" value="result2" />
<data name="Line 3" value="result3" />
<data name="Line 4" value="result4" />
</data>
</record>
</records>
```

3.5.5 - Element Reference

There are various elements used in a Data Splitter configuration to control behaviour. Each of these elements can be categorised as one of the following:

3.5.5.1 - Content Providers

Content providers take some content from the input source or elsewhere (see [fixed strings](#)) and provide it to one or more expressions. Both the root element `<dataSplitter>` and `<group>` elements are content providers.

3.5.5.1.1 Root element `<dataSplitter>`

The root element of a Data Splitter configuration is `<dataSplitter>`. It supplies content from the input source to one or more expressions defined within it. The way that content is buffered is controlled by the root element and the way that errors are handled as a result of child expressions not matching all of the content it supplies.

3.5.5.1.1.1 Attributes

The following attributes can be added to the `<dataSplitter>` root element:

- [ignoreErrors](#)
- [bufferSize](#)

3.5.5.1.1.1.1 ignoreErrors

Data Splitter generates errors if not all of the content is matched by the regular expressions beneath the `<dataSplitter>` or within `<group>` elements. The error messages are intended to aid the user in writing good Data Splitter configurations. The intent is to indicate when the input data is not being matched fully and therefore possibly skipping some important data. Despite this, in some cases it is laborious to have to write expressions to match all content. In these cases it is preferable to add this attribute to ignore these errors. However it is often better to write expressions that capture all of the supplied content and discard unwanted characters. This attribute also affects errors generated by the use of the `minMatch` attribute on `<regex>` which is described later on.

Take the following example input:

```
Name1,Name2,Name3
value1,value2,value3 # a useless comment
value1,value2,value3 # a useless comment
```

This could be matched with the following configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3
<regex id="heading" pattern=".+" maxMatch="1">
...
</regex>
<regex id="body" pattern="\n[^#]+>
...
</regex>
</dataSplitter>
```

The above configuration would only match up to a comment for each record line, e.g.

```
Name1,Name2,Name3
value1,value2,value3 # a useless comment
value1,value2,value3 # a useless comment
```

This may well be the desired functionality but if there was useful content within the comment it would be lost. Because of this Data Splitter warns you when expressions are failing to match all of the content presented so that you can make sure that you aren't missing anything important. In the above example it is obvious that this is the required behaviour but in more complex cases you might be otherwise unaware that your expressions were losing data.

To maintain this assurance that you are handling all content it is usually best to write expressions to explicitly match all content even though you may do nothing with some matches, e.g.

```
<?xml version="1.0" encoding="UTF-8"?>
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3
  <regex id="heading" pattern=".+" maxMatch="1">
  ...
  </regex>
  <regex id="body" pattern="\n([^\#]+)\#.+">
  ...
  </regex>
</dataSplitter>
```

The above example would match all of the content and would therefore not generate warnings. Sub-expressions of 'body' could use match group 1 and ignore the comment.

However as previously stated it might often be difficult to write expressions that will just match content that is to be discarded. In these cases ignoreErrors can be used to suppress errors caused by unmatched content.

3.5.5.1.1.1.2 bufferSize (Advanced)

This is an optional attribute used to tune the size of the character buffer used by Data Splitter. The default size is 20000 characters and should be fine for most translations. The minimum value that this can be set to is 20000 characters and the maximum is 1000000000. The only reason to specify this attribute is when individual records are bigger than 10000 characters which is rarely the case.

3.5.5.1.2 Group element <group>

Groups behave in a similar way to the root element in that they provide content for one or more inner expressions to deal with, e.g.

```
<group value="$1">
  <regex pattern="([^\t]*)\t([^\t]*)[\t]*([^\t]*)[=:]*(.*)" maxMatch="1">
  ...
  <regex pattern="([^\t]*)\t([^\t]*)[\t]*([^\t]*)[=:]*(.*)">
  ...

```

3.5.5.1.2.1 Attributes

As the <group> element is a content provider it also includes the same 'ignoreErrors' attribute which behaves in the same way. The complete list of attributes for the <group> element is as follows:

- [id](#)
- [value](#)
- [ignoreErrors](#)
- [matchOrder](#)
- [reverse](#)

3.5.5.1.2.1.1 id

When Data Splitter reports errors it outputs an XPath to describe the part of the configuration that generated the error, e.g.

```
DSParser [2:1] ERROR: Expressions failed to match all of the content provided by group: regex[0]/group[0]/regex[3]/group[1] : <c
```

It is often a little difficult to identify the configuration element that generated the error by looking at the path and the element description, particularly when multiple elements are the same, e.g. many <group> elements without attributes. To make identification easier you can add an 'id' attribute to any element in the configuration resulting in error descriptions as follows:

```
DSParser [2:1] ERROR: Expressions failed to match all of the content provided by group: regex[0]/group[0]/regex[3]/group[1] : <c
```

3.5.5.1.2.1.2 value

This attribute determines what content to present to child expressions. By default the entire content matched by a group's parent expression is passed on by the group to child expressions. If required, content from a specific match group in the parent expression can be passed to child expressions using the value attribute, e.g. `value="$1"`. In addition to this content can be composed in the same way as it is for data names and values. see [match references](#) for a full description of match references.

3.5.5.1.2.1.3 ignoreErrors

This behaves in the same way as for the root element.

3.5.5.1.2.1.4 matchOrder

This is an optional attribute used to control how content is consumed by expression matches. Content can be consumed in sequence or in any order using `matchOrder="sequence"` or `matchOrder="any"`. If the attribute is not specified, Data Splitter will default to matching in sequence.

When matching in sequence, each match consumes some content and the content position is moved beyond the match ready for the subsequent match. However, in some cases the order of these constructs is not predictable, e.g. we may sometimes be presented with:

```
Value1=1 Value2=2
```

... or sometimes with:

```
Value2=2 Value1=1
```

Using a sequential match order the following example would work to find both values in `Value1=1 Value2=2`

```
<group>
<regex pattern="Value1=([^\n]*)">
...
<regex pattern="Value2=([^\n]*)">
...
```

... but this example would skip over Value2 and only find the value of Value1 if the input was `Value2=2 Value1=1`.

To be able to deal with content that contains these constructs in either order we need to change the match order to `any`.

When matching in any order, each match removes the matched section from the content rather than moving the position past the match so that all remaining content can be matched by subsequent expressions. In the following example the first expression would match and remove `Value1=1` from the supplied content and the second expression would be presented with `Value2=2` which it could also match.

```
<group matchOrder="any">
<regex pattern="Value1=([^\n]*)">
...
<regex pattern="Value2=([^\n]*)">
...
```

If the attribute is omitted by default the match order will be sequential. This is the default behaviour as tokens are most often in sequence and consuming content in this way is more efficient as content does not need to be copied by the parser to chop out sections as is required for matching in any order. It is only necessary to use this feature when fields that are identifiable with a specific match can occur in any order.

3.5.5.1.2.1.5 reverse

Occasionally it is desirable to reverse the content presented by a group to child expressions. This is because it is sometimes easier to form a pattern by matching content in reverse.

Take the following example content of name, value pairs delimited by `=` but with no spaces between names, multiple spaces between values and only a space between subsequent pairs:

```
ipAddress=123.123.123.123 zones=Zone 1, Zone 2, Zone 3 location=loc1 A user=An end user serverName=bigserver
```

We could write a pattern that matches each name value pair by matching up to the start of the next name, e.g.

```
<regex pattern="([^\=]+)=(.+?)( [^\=]+)">
```

This would match the following:

```
ipAddress=123.123.123.123 zones=
```

Here we are capturing the name and value for each pair in separate groups but the pattern has to also match the name from the next name value pair to find the end of the value. By default Data Splitter will move the content buffer to the end of the match ready for subsequent matches so the next name will not be available for matching.

In addition to matching too much content the above example also uses a reluctant qualifier `.+?`. Use of reluctant qualifiers almost always impacts performance so they are to be avoided if at all possible.

A better way to match the example content is to match the input in reverse, reading characters from right to left.

The following example demonstrates this:

```
<group reverse="true">
  <regex pattern="([^\=]+)=([^\ ]+)">
    <data name="$2" value="$1" />
  </regex>
</group>
```

Using the `reverse` attribute on the parent group causes content to be supplied to all child expressions in reverse order. In the above example this allows the pattern to match values followed by names which enables us to cope with the fact that values have multiple spaces but names have no spaces.

Content is only presented to child regular expressions in reverse. When referencing values from match groups the content is returned in the correct order, e.g. the above example would return:

```
<data name="ipAddress" value="123.123.123.123" />
<data name="zones" value="Zone 1, Zone 2, Zone 3" />
<data name="location" value="loc1" />
<data name="user" value="An end user" />
<data name="serverName" value="bigserver" />
```

The reverse feature isn't needed very often but there are a few cases where it really helps produce the desired output without the complexity and performance overhead of a reluctant match.

An alternative to using the `reverse` attribute is to use the original reluctant expression example but tell Data Splitter to make the subsequent name available for the next match by not advancing the content beyond the end of the previous value. This is done by using the [advance attribute](#) on the `<regex>`. However, the `reverse` attribute represents a better way to solve this particular problem and allows a simpler and more efficient regular expression to be used.

3.5.5.2 - Expressions

Expressions match some data supplied by a parent content provider. The content matched by an expression depends on the type of expression and how it is configured.

The `<split>`, `<regex>` and `<all>` elements are all expressions and match content as described below.

3.5.5.2.1 The `<split>` element

The `<split>` element directs Data Splitter to break up content using a specified character sequence as a delimiter. In addition to this it is possible to specify characters that are used to escape the delimiter as well as characters that contain or “quote” a value that may include the delimiter sequence but allow it to be ignored.

3.5.5.2.1.1 Attributes

The `<split>` element has the following attributes:

- [id](#)
- [delimiter](#)
- [escape](#)
- [containerStart](#)
- [containerEnd](#)
- [maxMatch](#)
- [minMatch](#)
- [onlyMatch](#)

3.5.5.2.1.1.1 id

Optional attribute used to debug the location of expressions causing errors, see [id](#).

3.5.5.2.1.1.2 delimiter

A required attribute used to specify the character string that will be used as a delimiter to split the supplied content unless it is preceded by an escape character or within a container if specified. Several of the previous examples use this attribute.

3.5.5.2.1.1.3 escape

An optional attribute used to specify a character sequence that is used to escape the delimiter. Many delimited text formats have an escape character that is used to tell any parser that the following delimiter should be ignored, e.g. often a character such as “ is used to escape the character that follows it so that it is not treated as a delimiter. When specified this escape sequence also applies to any container characters that may be specified.

3.5.5.2.1.1.4 containerStart

An optional attribute used to specify a character sequence that will make this expression ignore the presence of delimiters until an end container is found. If the character is preceded by the specified escape sequence then this container sequence will be ignored and the expression will continue matching characters up to a delimiter.

If used `containerEnd` must also be specified. If the container characters are to be ignored from the match then match group 1 must be used instead of 0.

3.5.5.2.1.1.5 containerEnd

An optional attribute used to specify a character sequence that will make this expression stop ignoring the presence of delimiters if it believes it is currently in a container. If the character is preceded by the specified escape sequence then this container sequence will be ignored and the expression will continue matching characters while ignoring the presence of any delimiter.

If used `containerStart` must also be specified. If the container characters are to be ignored from the match then match group 1 must be used instead of 0.

3.5.5.2.1.1.6 maxMatch

An optional attribute used to specify the maximum number of times this expression is allowed to match the supplied content. If you do not supply this attribute then the Data Splitter will keep matching the supplied content until it reaches the end. If specified Data Splitter will stop matching the supplied content when it has matched it the specified number of times.

This attribute is used in the '[CSV with header line](#)' example to ensure that only the first line is treated as a header line.

3.5.5.2.1.1.7 minMatch

An optional attribute used to specify the minimum number of times this expression should match the supplied content. If you do not supply this attribute then Data Splitter will not enforce that the expression matches the supplied content. If specified Data Splitter will generate an error if the expression does not match the supplied content at least as many times as specified.

Unlike `maxMatch`, `minMatch` does not control the matching process but instead controls the production of error messages generated if the parser is not seeing the expected input.

3.5.5.2.1.1.8 onlyMatch

Optional attribute to use this expression only for specific instances of a match of the parent expression, e.g. on the 4th, 5th and 8th matches of the parent expression specified by '4,5,8'. This is used when this expression should only be used to subdivide content from certain parent matches.

3.5.5.2.2 The <regex> element

The `<regex>` element directs Data Splitter to match content using the specified regular expression pattern. In addition to this the same match control attributes that are available on the `<split>` element are also present as well as attributes to alter the way the pattern works.

3.5.5.2.2.1 Attributes

The `<regex>` element has the following attributes:

- [id](#)
- [pattern](#)
- [dotAll](#)
- [caseInsensitive](#)
- [maxMatch](#)
- [minMatch](#)
- [onlyMatch](#)
- [advance](#)

3.5.5.2.2.1.1 id

Optional attribute used to debug the location of expressions causing errors, see [id](#).

3.5.5.2.2.1.2 pattern

This is a required attribute used to specify a regular expression to use to match on the supplied content. The pattern is used to match the content multiple times until the end of the content is reached while the maxMatch and onlyMatch conditions are satisfied.

3.5.5.2.2.1.3 dotAll

An optional attribute used to specify if the use of '' in the supplied pattern matches all characters including new lines. If 'true' '' will match all characters including new lines, if 'false' it will only match up to a new line. If this attribute is not specified it defaults to 'false' and will only match up to a new line.

This attribute is used in many of the multiline examples above.

3.5.5.2.2.1.4 caseInsensitive

An optional attribute used to specify if the supplied pattern should match content in a case insensitive way. If 'true' the expression will match content in a case insensitive manner, if 'false' it will match the content in a case sensitive manner. If this attribute is not specified it defaults to 'false' and will match the content in a case sensitive manner.

3.5.5.2.2.1.5 maxMatch

This is used in the same way it is on the `<split>` element, see [maxMatch](#).

3.5.5.2.2.1.6 minMatch

This is used in the same way it is on the `<split>` element, see [minMatch](#).

3.5.5.2.2.1.7 onlyMatch

This is used in the same way it is on the `<split>` element, see [onlyMatch](#).

3.5.5.2.2.1.8 advance

After an expression has matched content in the buffer, the buffer start position is advanced so that it moves to the end of the entire match. This means that subsequent expressions operating on the content buffer will not see the previously matched content again. This is normally required behaviour, but in some cases some of the content from a match is still required for subsequent matches. Take the following example of name value pairs:

```
name1=some value 1 name2=some value 2 name3=some value 3
```

The first name value pair could be matched with the following expression:

```
<regex pattern="([^\=]+)=(.+?) [^\= ]+=">
```

The above expression would match as follows:

```
name1=some value 1 name2=some value 2 name3=some value 3
```

In this example we have had to do a reluctant match to extract the value in group 2 and not include the subsequent name. Because the reluctant match requires us to specify what we are reluctantly matching up to, we have had to include an expression after it that matches the next name.

By default the parser will move the character buffer to the end of the entire match so the next expression will be presented with the following:

```
some value 2 name3=some value 3
```

Therefore `name2` will have been lost from the content buffer and will not be available for matching.

This behaviour can be altered by telling the expression how far to advance the character buffer after matching. This is done with the `advance` attribute and is used to specify the match group whose end position should be treated as the point the content buffer should advance to, e.g.

```
<regex pattern="([^\=]+)=(.+?) [^\= ]+=" advance="2">
```

In this example the content buffer will only advance to the end of match group 2 and subsequent expressions will be presented with the following content:

```
name2=some value 2 name3=some value 3
```

Therefore `name2` will still be available in the content buffer.

It is likely that the advance feature will only be useful in cases where a reluctant match is performed. Reluctant matches are discouraged for performance reasons so this feature should rarely be used. A better way to tackle the above example would be to present the content in [reverse](#), however this is only possible if the expression is within a group, i.e. is not a root expression. There may also be more complex cases where reversal is not an option and the use of a reluctant match is the only option.

3.5.5.2.3 The `<all>` element

The `<all>` element matches the entire content of the parent group and makes it available to child groups or `<data>` elements. The purpose of `<all>` is to act as a catch all expression to deal with content that is not handled by a more specific expression, e.g. to output some other unknown, unrecognised or unexpected data.

```
<group>
  <regex pattern="^\s*([^=]+)=([^=+])\s*">
    <data name="$1" value="$2" />
  </regex>

  <!-- Output unexpected data -->
  <all>
    <data name="unknown" value="$" />
  </all>
</group>
```

The `<all>` element provides the same functionality as using `.*` as a pattern in a `<regex>` element and where `dotAll` is set to true, e.g. `<regex pattern=".*" dotAll="true">`. However it performs much faster as it doesn't require pattern matching logic and is therefore always preferred.

3.5.5.2.3.1 Attributes

The `<all>` element has the following attributes:

- [id](#)

3.5.5.2.3.1.1 id

Optional attribute used to debug the location of expressions causing errors, see [id](#).

3.5.5.3 - Output

As with all other aspects of Data Splitter, output XML is determined by adding certain elements to the Data Splitter configuration.

3.5.5.3.1 The <data> element

Output is created by Data Splitter using one or more `<data>` elements in the configuration. The first `<data>` element that is encountered within a matched expression will result in parent `<record>` elements being produced in the output.

3.5.5.3.1.1 Attributes

The `<data>` element has the following attributes:

- [id](#)
- [name](#)
- [value](#)

3.5.5.3.1.1.1 id

Optional attribute used to debug the location of expressions causing errors, see [id](#).

3.5.5.3.1.1.2 name

Both the name and value attributes of the `<data>` element can be specified using [match references](#).

3.5.5.3.1.1.3 value

Both the name and value attributes of the `<data>` element can be specified using [match references](#).

3.5.5.3.1.1.4 Single <data> element example

The simplest example that can be provided uses a single `<data>` element within a `<split>` expression.

Given the following input:

```
This is line 1
This is line 2
This is line 3
```

... and the following configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3
<split delimiter="\n" >
  <data value="$1"/>
</split>
</dataSplitter>
```

... you would get the following output:

```

<?xml version="1.0" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="records:2 file:///records-v2"
<record>
  <data value="This is line 1" />
</record>
<record>
  <data value="This is line 2" />
</record>
<record>
  <data value="This is line 3" />
</record>
</records>

```

3.5.5.3.1.1.5 Multiple <data> element example

You could also output multiple <data> elements for the same <record> by adding multiple elements within the same expression:

Given the following input:

```

ip=1.1.1.1 user=user1
ip=2.2.2.2 user=user2
ip=3.3.3.3 user=user3

```

... and the following configuration:

```

<?xml version="1.0" encoding="UTF-8"?>
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3
<regex pattern="ip=([^\ ]+) user=([^\ ]+)\s*>
  <data name="ip" value="$1"/>
  <data name="user" value="$2"/>
</split>
</dataSplitter>

```

... you would get the following output:

```

<?xml version="1.0" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="records:2 file:///records-v2"
<record>
  <data name="ip" value="1.1.1.1" />
  <data name="user" value="user1" />
</record>
<record>
  <data name="ip" value="2.2.2.2" />
  <data name="user" value="user2" />
</record>
<record>
  <data name="ip" value="3.3.3.3" />
  <data name="user" value="user3" />
</record>
</records>

```

3.5.5.3.1.2 Multi level <data> elements

As long as all data elements occur within the same parent/ancestor expression, all data elements will be output within the same record.

Given the following input:

```

ip=1.1.1.1 user=user1
ip=2.2.2.2 user=user2
ip=3.3.3.3 user=user3

```

... and the following configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3
<split delimiter="\n" >
  <data name="line" value="$1"/>

  <group value="$1">
    <regex pattern="ip=([^ ]+) user=([^ ]+)">
      <data name="ip" value="$1"/>
      <data name="user" value="$2"/>
    </regex>
  </group>
</split>
</dataSplitter>
```

... you would get the following output:

```
<?xml version="1.0" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="records:2 file://records-v2
<record>
  <data name="line" value="ip=1.1.1.1 user=user1" />
  <data name="ip" value="1.1.1.1" />
  <data name="user" value="user1" />
</record>
<record>
  <data name="line" value="ip=2.2.2.2 user=user2" />
  <data name="ip" value="2.2.2.2" />
  <data name="user" value="user2" />
</record>
<record>
  <data name="line" value="ip=3.3.3.3 user=user3" />
  <data name="ip" value="3.3.3.3" />
  <data name="user" value="user3" />
</record>
</records>
```

3.5.5.3.1.3 Nesting <data> elements

Rather than having <data> elements all appear as children of <record> it is possible to nest them either as direct children or within child groups.

3.5.5.3.1.3.1 Direct children

Given the following input:

```
ip=1.1.1.1 user=user1
ip=2.2.2.2 user=user2
ip=3.3.3.3 user=user3
```

... and the following configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3
<regex pattern="ip=([^ ]+) user=([^ ]+)\s*>
  <data name="line" value="$">
    <data name="ip" value="$1"/>
    <data name="user" value="$2"/>
  </data>
</split>
</dataSplitter>
```

... you would get the following output:

```
<?xml version="1.0" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="records:2 file:///records-v2"
<record>
  <data name="line" value="ip=1.1.1.1 user=user1">
    <data name="ip" value="1.1.1.1" />
    <data name="user" value="user1" />
  </data>
</record>
<record>
  <data name="line" value="ip=2.2.2.2 user=user2">
    <data name="ip" value="2.2.2.2" />
    <data name="user" value="user2" />
  </data>
</record>
<record>
  <data name="line" value="ip=3.3.3.3 user=user3">
    <data name="ip" value="3.3.3.3" />
    <data name="user" value="user3" />
  </data>
</record>
</records>
```

3.5.5.3.1.3.2 Within child groups

Given the following input:

```
ip=1.1.1.1 user=user1
ip=2.2.2.2 user=user2
ip=3.3.3.3 user=user3
```

... and the following configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3
<split delimiter="\n" >
  <data name="line" value="$1">
    <group value="$1">
      <regex pattern="ip=([^ ]+) user=([^ ]+)">
        <data name="ip" value="$1"/>
        <data name="user" value="$2"/>
      </regex>
    </group>
  </data>
</split>
</dataSplitter>
```

... you would get the following output:

```

<?xml version="1.0" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="records:2 file://records-v2
<record>
  <data name="line" value="ip=1.1.1.1 user=user1">
    <data name="ip" value="1.1.1.1" />
    <data name="user" value="user1" />
  </data>
</record>
<record>
  <data name="line" value="ip=2.2.2.2 user=user2">
    <data name="ip" value="2.2.2.2" />
    <data name="user" value="user2" />
  </data>
</record>
<record>
  <data name="line" value="ip=3.3.3.3 user=user3">
    <data name="ip" value="3.3.3.3" />
    <data name="user" value="user3" />
  </data>
</record>
</records>

```

The above example produces the same output as the previous but could be used to apply much more complex expression logic to produce the child `<data>` elements, e.g. the inclusion of multiple child expressions to deal with different types of lines.

3.5.5.4 - Variables

A variable is added to Data Splitter using the `<var>` element. A variable is used to store matches from a parent expression for use in a reference elsewhere in the configuration, see [variable reference](#).

The most recent matches are stored for use in local references, i.e. references that are in the same match scope as the variable. Multiple matches are stored for use in references that are in a separate match scope. The concept of different variable scopes is described in [scopes](#).

3.5.5.4.1 The `<var>` element

The `<var>` element is used to tell Data Splitter to store matches from a parent expression for use in a reference.

3.5.5.4.1.1 Attributes

The `<var>` element has the following attributes:

- [id](#)

3.5.5.4.1.1.1 id

Mandatory attribute used to uniquely identify it within the configuration (see [id](#)) and is the means by which a variable is referenced, e.g. `VAR_ID`.

3.5.6 - Match References, Variables and Fixed Strings

The `<group>` and `<data>` elements can reference match groups from parent expressions or from stored matches in variables. In the case of the `<group>` element, referenced values are passed on to child expressions whereas the `<data>` element can use match group references for name and value attributes. In the case of both elements the way of specifying references is the same.

3.5.6.1 - Concatenation of references

It is possible to concatenate multiple fixed strings and match group references using the + character. As with all references and fixed strings this can be done in <group> value and <data> name and value attributes. However concatenation does have some performance overhead as new buffers have to be created to store concatenated content.

A good example of concatenation is the production of ISO8601 date format from data in the previous example:

```
01/01/2010,00:00:00
```

Here the following <regex> could be used to extract the relevant date, time groups:

```
<regex pattern="(\d{2})/(\d{2})/(\d{4}),(\d{2}):(\d{2}):(\d{2})">
```

The match groups from this expression can be concatenated with the following value output pattern in the data element:

```
<data name="dateTime" value="$3+'-'+$2+'-'+$1+'-'+'T'+$4+':'+$5+':'+$6+'.000Z'" />
```

Using the original example, this would result in the output:

```
<data name="dateTime" value="2010-01-01T00:00:00.000Z" />
```

Note that the value output pattern wraps all fixed strings in single quotes. This is necessary when concatenating strings and references so that Data Splitter can determine which parts are to be treated as fixed strings. This also allows fixed strings to contain \$ and + characters.

As single quotes are used for this purpose, a single quote needs to be escaped with another single quote if one is desired in a fixed string, e.g.

```
'this ''is quoted text'''
```

will result in:

```
this 'is quoted text'
```

3.5.6.2 - Expression match references

Referencing matches in expressions is done using `$`. In addition to this a match group number may be added to just retrieve part of the expression match. The applicability and effect that this has depends on the type of expression used.

3.5.6.2.1 References to <split> Match Groups

In the following example a line matched by a parent `<split>` expression is referenced by a child `<data>` element.

```
<split delimiter="\n" >
  <data name="line" value="$"/>
</split>
```

A `<split>` element matches content up to and including the specified [delimiter](#), so the above reference would output the entire line plus the delimiter. However there are various match groups that can be used by child `<group>` and `<data>` elements to reference sections of the matched content.

To illustrate the content provided by each match group, take the following example:

```
"This is some text\, that we wish to match", "This is the next text"
```

And the following `<split>` element:

```
<split delimiter="," escape="\\">
```

The match groups are as follows:

- `$` or `$0`: The entire content that is matched including the specified delimiter at the end

```
"This is some text\, that we wish to match",
```

- `$1`: The content up to the specified delimiter at the end

```
"This is some text\, that we wish to match"
```

- `$2`: The content up to the specified delimiter at the end and filtered to remove escape characters (more expensive than `$1`)

```
"This is some text, that we wish to match"
```

In addition to this behaviour match groups 1 and 2 will omit outermost whitespace and container characters if specified, e.g. with the following content:

```
" This is some text\, that we wish to match " , "This is the next text"
```

And the following `<split>` element:

```
<split delimiter="," escape="\\" containerStart="\" containerEnd="\">
```

The match groups are as follows:

- `$` or `$0`: The entire content that is matched including the specified delimiter at the end

```
" This is some text\, that we wish to match " ,
```

- `$1`: The content up to the specified delimiter at the end and strips outer containers.

```
This is some text\, that we wish to match
```

- `$2`: The content up to the specified delimiter at the end and strips outer containers and filtered to remove escape characters (more computationally expensive than `$1`)

```
This is some text, that we wish to match
```

3.5.6.2.2 References to <regex> Match Groups

Like the `<split>` element various match groups can be referenced in a `<regex>` expression to retrieve portions of matched content. This content can be used as values for `<group>` and `<data>` elements.

Given the following input:

```
ip=1.1.1.1 user=user1
```

And the following `<regex>` element:

```
<regex pattern="ip=([^\ ]+) user=([^\ ]+)">
```

The match groups are as follows:

- `$` or `$0`: The entire content that is matched by the expression

```
ip=1.1.1.1 user=user1
```

- `$1`: The content of the first match group

```
1.1.1.1
```

- `$2`: The content of the second match group

```
user1
```

Match group numbers in regular expressions are determined by the order that their open bracket appears in the expression.

3.5.6.2.3 References to <any> Match Groups

The `<any>` element does not have any match groups and always returns the entire content that was passed to it when referenced with `$`.

3.5.6.3 - Use of fixed strings

Any `<group>` value or `<data>` name and value can use references to matched content, but in addition to this it is possible just to output a known string, e.g.

```
<data name="somename" value="$" />
```

The above example would output `somename` as the `<data>` name attribute. This can often be useful where there are no headings specified in the input data but we want to associate certain names with certain values.

Given the following data:

```
01/01/2010,00:00:00,192.168.1.100,SOMEHOST.SOMEWHERE.COM,user1,logon,
```

We could provide useful headings with the following configuration:

```
<regex pattern="([^\n]*),([^\n]*),([^\n]*),([^\n]*),([^\n]*),([^\n]*),">
<data name="date" value="$1" />
<data name="time" value="$2" />
<data name="ipAddress" value="$3" />
<data name="hostName" value="$4" />
<data name="user" value="$5" />
<data name="action" value="$6" />
</regex>
```

3.5.6.4 - Variable reference

Variables are added to Data Splitter configuration using the `<var>` element, see [variables](#). Each variable must have a unique id so that it can be referenced. References to variables have the form `$VARIABLE_ID$`, e.g.

```
<data name="$heading$" value="$" />
```

3.5.6.4.1 Identification

Data Splitter validates the configuration on load and ensures that all element ids are unique and that referenced ids belong to a variable.

A variable will only store data if it is referenced so variables that are not referenced will do nothing. In addition to this a variable will only store data for match groups that are referenced, e.g. if `$heading$1` is the only reference to a variable with an id of 'heading' then only data for match group 1 will be stored for reference lookup.

3.5.6.4.2 Scopes

Variables have two scopes which affect how data is retrieved when referenced:

- [Local scope](#)
- [Remote scope](#)

3.5.6.4.2.1 Local Scope

Variables are local to a reference if the reference exists as a descendant of the variables parent expression, e.g.

```
<split delimiter="\n" >
  <var id="line" />

  <group value="$1">
    <regex pattern="ip=([^ ]+) user=([^ ]+)">
      <data name="line" value="$line$"/>
      <data name="ip" value="$1"/>
      <data name="user" value="$2"/>
    </regex>
  </group>
</split>
```

In the above example, matches for the outermost `<split>` expression are stored in the variable with the id of `line`. The only reference to this variable is in a data element that is a descendant of the variables parent expression `<split>`, i.e. it is nested within `split/group/regex`.

Because the variable is referenced locally only the most recent parent match is relevant, i.e. no retrieval of values by [iteration](#), [iteration offset](#) or [fixed position](#) is applicable. These features only apply to remote variables that store multiple values.

3.5.6.4.2.2 Remote Scope

The [CSV example with a heading](#) is an example of a variable being referenced from a remote scope.

```

<?xml version="1.0" encoding="UTF-8"?>
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3

<!-- Match heading line (note that maxMatch="1" means that only the first line will be matched by this splitter) -->
<split delimiter="\n" maxMatch="1">

    <!-- Store each heading in a named list -->
    <group>
        <split delimiter="," ">
            <var id="heading" />
        </split>
    </group>
</split>

<!-- Match each record -->
<split delimiter="\n">

    <!-- Take the matched line -->
    <group value="$1">

        <!-- Split the line up -->
        <split delimiter="," ">

            <!-- Output the stored heading for each iteration and the value from group 1 -->
            <data name="$heading$1" value="$1" />
        </split>
    </group>
</split>
</dataSplitter>

```

In the above example the parent expression of the variable is not the ancestor of the reference in the `<data>` element. This makes the `<data>` elements reference to the variable a remote one. In this situation the variable knows that it must store multiple values as the remote reference `<data>` may retrieve one of many values from the variable based on:

1. The match count of the parent expression.
2. The match count of the parent expression, plus or minus an offset.
3. A fixed position in the variable store.

3.5.6.4.2.2.1 Retrieval of value by iteration

In the above example the first line is taken then repeatedly matched by delimiting with commas. This results in multiple values being stored in the 'heading' variable. Once this is done subsequent lines are matched and then also repeatedly matched by delimiting with commas in the same way the heading is.

Each time a line is matched the internal match count of all sub expressions, (e.g. the `<split>` expression that is delimited by comma) is reset to 0. Every time the sub `<split>` expression matches up to a comma delimiter the match count is incremented. Any references to remote variables will, by default, use the current match count as an index to retrieve one of the many values stored in the variable. This means that the `<data>` element in the above example will retrieve the corresponding heading for each value as the match count of the values will match the storage position of each heading.

3.5.6.4.2.2.2 Retrieval of value by iteration offset

In some cases there may be a mismatch between the position where a value is stored in a variable and the match count applicable when remotely referencing the variable.

Take the following input:

```

BAD,Date,Time,IPAddress,HostName,User,EventType,Detail
01/01/2010,00:00:00,192.168.1.100,SOMEHOST.SOMEWHERE.COM,user1,logon,

```

In the above example we can see that the first heading 'BAD' is not correct for the first value of every line. In this situation we could either adjust the way the heading line is parsed to ignore 'BAD' or just adjust the way the heading variable is referenced.

To make this adjustment the reference just needs to be told what offset to apply to the current match count to correctly retrieve the stored value. In the above example this would be done like this:

```
<data name="$heading$1[+1]" value="$1" />
```

The above reference just uses the match count plus 1 to retrieve the stored value. Any integral offset plus or minus may be used, e.g. [+4] or [-10]. Offsets that result in a position that is outside of the storage range for the variable will not return a value.

3.5.6.4.2.2.3 Retrieval of value by fixed position

In addition to retrieval by offset from the current match count, a stored value can be returned by a fixed position that has no relevance to the current match count.

In the following example the value retrieved from the 'heading' variable will always be 'IPAddress' as this is the fourth value stored in the 'heading' variable and the position index starts at 0.

```
<data name="$heading$1[3]" value="$1" />
```

3.6 - Editing and Viewing Data

How to view data and edit content.

3.6.1 Viewing Data

The data viewer is shown on the *Data* tab when you open (by double clicking) one of these items in the explorer tree:

- Feed - to show all data for that feed.
- Folder - to show all data for all feeds that are descendants of the folder.
- System Root Folder - to show all data for all feeds that are ancestors of the folder.

In all cases the data shown is dependant on the permissions of the user performing the action and any permissions set on the feeds/folders being viewed.

The Data Viewer screen is made up of the following three parts which are shown as three panes split horizontally.

3.6.1.1 Stream List

This shows all streams within the opened entity (feed or folder). The streams are shown in reverse chronological order. By default *Deleted* and *Locked* streams are filtered out. The filtering can be changed by clicking on the  Filter icon. This will show all stream types by default so may be a mixture of *Raw events*, *Events*, *Errors*, etc. depending on the feed/folder in question.

3.6.1.2 Related Stream List

This list only shows data when a stream is selected in the streams list above it. It shows all streams related to the currently selected stream. It may show streams that are 'ancestors' of the selected stream, e.g. showing the *Raw Events* stream for an *Events* stream, or show descendants, e.g. showing the *Errors* stream which resulted from processing the selected *Raw Events* stream.

3.6.1.3 Content Viewer Pane

This pane shows the contents of the stream selected in the Related Streams List. The content of a stream will differ depending on the type of stream selected and the child stream types in that stream. For more information on the anatomy of streams, see [Streams](#). This pane is split into multiple sub tabs depending on the different types of content available.

3.6.1.3.1 Info Tab

This sub-tab shows the information for the stream, such as creation times, size, physical file location, state, etc.

3.6.1.3.2 Error Tab

This sub-tab is only visible for an Error stream. It shows a table of errors and warnings with associated messages and locations in the stream that it relates to.

3.6.1.3.3 Data Preview Tab

This sub-tab shows the content of the data child stream, formatted if it is XML. It will only show a limited amount of data so if the data child stream is large then it will only show the first n characters.

If the stream is multi-part then you will see Part navigation controls to switch between parts. For each part you will be shown the first n character of that part (formatted if applicable).

If the stream is a [Segmented stream](#) stream then you will see the Record navigation controls to switch between records. Only one record is shown at once. If a record is very large then only the first n characters of the record will be shown.

This sub-tab is intended for seeing a quick preview of the data in a form that is easy to read, i.e. formatted. If you want to see the full data in its original form then click on the *View Source* link at the top right of the sub-tab.

The Data Preview tab shows a '[progress](#)' bar to indicate what portion of the content is visible in the editor.

3.6.1.3.4 Context Tab

This sub-tab is only shown for non-segmented streams, e.g. *Raw Events* and *Raw_Reference* that have an associated context data child stream. For more details of context streams, see [Context](#) This sub-tab works in exactly the same way as the Data Preview sub-tab except that it shows a different child stream.

3.6.1.3.5 Meta Tab

This sub-tab is only shown for non-segmented streams, e.g. *Raw Events* and *Raw_Reference* that have an associated meta data child stream. For more details of meta streams, see [Meta](#) This sub-tab works in exactly the same way as the Data Preview sub-tab except that it shows a different child stream.

3.6.1.4 Source View

The source view is accessed by clicking the *View Source* link on the *Data Preview* sub-tab or from the `data()` dashboard column function. Its purpose is to display the selected child stream (data, context, meta, etc) or record in the form in which it was received, i.e un-formatted.

The Data Preview tab shows a ‘[progress](#)’ bar to indicate what portion of the content is visible in the editor.

In order to navigate through the data you have three options

- Click on the ‘progress bar’ to show a porting of the data starting from the position clicked on.
- Page through the data using the navigation controls.
- Select a source range to display using the Set Source Range dialog which is accessed by clicking on the *Lines* or *Chars* links. This allows you to precisely select the range to display. You can either specify a range with a just start point or a start point and some form of size/position limit. If no limit is specified then Stroom will limit the data shown to the configured maximum (`stroom.ui.source.maxCharactersPerFetch`). If a range is entered that is too big to display Stroom will limit the data to its maximum.

3.6.1.4.1 A Note About Characters

Stroom does not know the size of a stream in terms of character lines/cols, it only knows the size in bytes. Due to the way character data is encoded into bytes it is not possible to say how many characters are in a stream based on its size in bytes. Stroom can only provide an estimate based on the ratio of characters to bytes seen so far in the stream.

3.6.1.5 Data Progress Bar

Stroom often handles very large streams of data and it is not feasible to show all of this data in the editor at once. Therefore Stroom will show a limited amount of the data in the editor at a time. The ‘progress’ bar at the top of the Data Preview and Source View screens shows what percentage of the data is visible in the editor and where in the stream the visible portion is located. If all of the data is visible in the editor (which includes scrolling down to see it) the bar will be green and will occupy the full width. If only some of the data is visible then the bar will be blue and the coloured part will only occupy part of the width.

3.6.2 Editor

Stroom uses the Ace editor for editing and viewing text, such as XSLTs, raw data, cooked events, stepping, etc.

3.6.2.1 Keyboard shortcuts

The following are some useful keyboard shortcuts in the editor:

- `ctrl-z` - Undo last action.
- `ctrl-shift-z` - Redo previously undone action.
- `ctrl-/` - Toggle commenting of current line/selection. Applies when editing XML, XSLT or Javascript.
- `alt-up` / `alt-down` - Move line/selection up/down respectively
- `ctrl-d` - Delete current line.
- `ctrl-f` - Open find dialog.
- `ctrl-h` - Open find/replace dialog.
- `ctrl-k` - Find next match.
- `ctrl-shift-k` - Find previous match.
- `tab` - Indent selection.
- `shift-tab` - Outdent selection.
- `ctrl-u` - Make selection upper-case.

See [here \(external link\)](#) for more.

3.6.2.1.1 Vim key bindings

If you are familiar with the Vi/Vim text editors then it is possible to enable Vim key bindings in Stroom. This is currently done by enabling Vim bindings in the editor context menu in each editor screen. In future versions of Stroom it will be possible to store these preferences on a per user basis.

The Ace editor does not support all features of Vim however the core navigation/editing key bindings are present. The key supported features of Vim are:

- Visual mode and visual block mode.
- Searching with / (javascript flavour regex)
- Search/replace with commands like :%s/foo/bar/g
- Incrementing/decrementing numbers with `ctrl-a` / `ctrl-x`
- Code (un-)folding with `zo` , `zc` , etc.
- Text objects, e.g. `>` , `)` , `]` , `'` , `"` , `p` paragraph, `w` word.
- Repetition with the `.` command.
- Jumping to a line with `:<line no>` .

Notable features not supported by the Ace editor:

- The following text objects are not supported
 - `b` - Braces, i.e { or [.
 - `t` - Tags, i.e. XML tags `<value>` .
 - `s` - Sentence.
- The `g` command mode command, i.e. `:g/foo/d`
- Splits

For a list of useful Vim key bindings see this [cheat sheet \(external link\)](#), though not all bindings will be available in Stroom's Ace editor.

3.6.2.2 Auto-Completion And Snippets

The editor supports a number of different types of auto-completion of text. Completion suggestions are triggered by the following mechanisms:

- `ctrl-space` - when live auto-complete is disabled.
- Typing - when live auto-complete is enabled.

When completion suggestions are triggered the follow types of completion may be available depending on the text being edited.

- *Local* - any word/token found in the existing text. Useful if you have typed a long word and need to type it again.
- *Keyword* - A word/token that has been defined in the syntax highlighting rules for the text type, i.e. `function` is a keyword when editing Javascript.
- *Snippet* - A block of text that has been defined as a snippet for the editor mode (XML, Javascript, etc.).

3.6.2.2.1 Snippets

Snippets allow you to quickly entry pre-defined blocks of common text. For example when editing an XSLT you may want to insert a call-template with parameters. To do this using snippets you can do the following:

- Type `call` then hit `ctrl-space` .
- In the list of options use the cursor keys to select `call-template with-param` then hit `enter` or `tab` to insert the snippet. The snippet will look like

```
<xsl:call-template name="template">
  <xsl:with-param name="param"></xsl:with-param>
</xsl:call-template>
```

- The cursor will be positioned on the first tab stop (the template name) with the tab stop text selected.
- At this point you can type in your template name, e.g. `MyTemplate` , then hit `tab` to advance to the next tab stop (the param name)
- Now type the name of the param, e.g. `MyParam` , then hit `tab` to advance to the last tab stop positioned within the `<with-param>` ready to enter the param value.

Snippets can be disabled from the list of suggestions by selecting the option in the editor context menu.

3.7 - Event Feeds

##Event Feeds

In order for Stroom to be able to handle the various data types as described in the previous section, Stroom must be told what the data is when it is received. This is achieved using Event Feeds. Each feed has a unique name within the system.

Events Feeds can be related to one or more Reference Feed. Reference Feeds are used to provide look up data for a translation. E.g. look up a computer name by its IP address.

Feeds can also have associated context data. Context data used to provide look up information that is only applicable for the events file it relates to. E.g. if the events file is missing information relating to the computer it was generated on, and you don't want to create separate feeds for each computer, an associated context file could be used to provide this information.

##Feed Identifiers

Feed identifiers must be unique within the system. Identifiers should be in the following format

```
<SYSTEM>-<ENVIRONMENT>-<TYPE>-<EVENTS/REFERENCE>-<VERSION>
```

If feeds in a certain site need different reference data then the site can be broken down further.

_ may be used as a space.

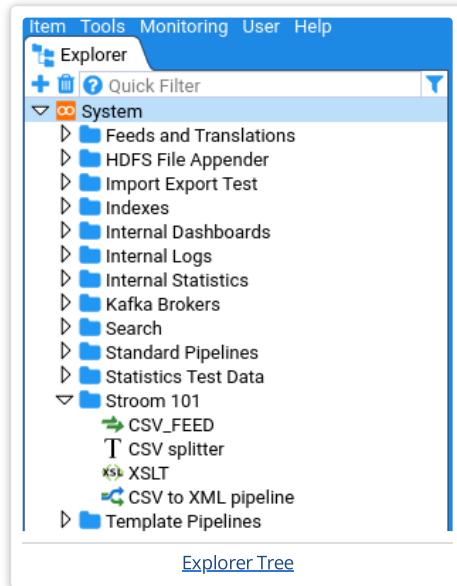
3.8 - Finding Things

How to find things in Stroom, for example content, simple string values, etc.

Version Information: Created with Stroom v7.0

Last Updated: 01 Jun 2020

3.8.1 Explorer Tree

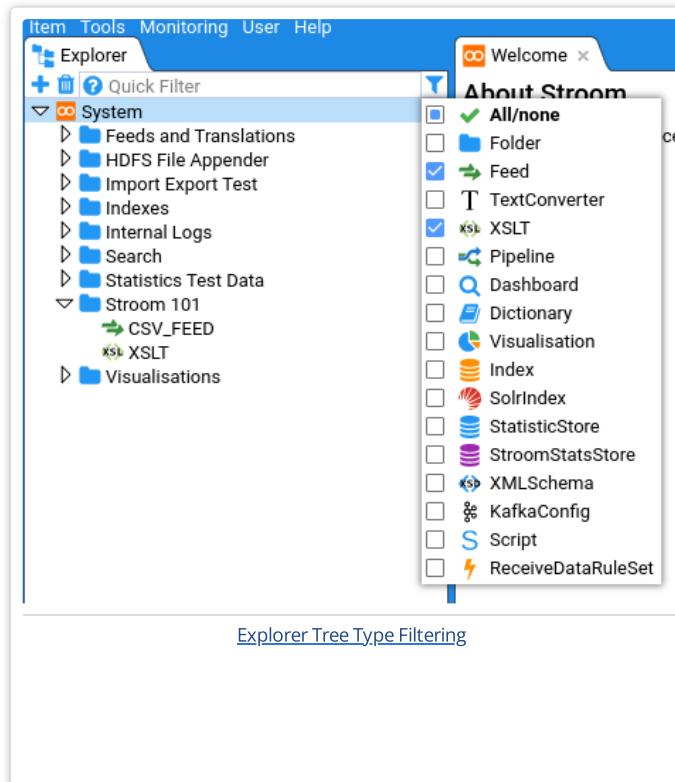


The *Explorer Tree* in stroom is the primary means of finding user created content, for example *Feeds*, *XSLTs*, *Pipelines*, etc.

Branches of the *Explorer Tree* can be expanded and collapsed to reveal/hide the content at different levels.

3.8.1.1 Filtering by Type

The *Explorer Tree* can be filtered by the type of content, e.g. to display only *Feeds*, or only *Feeds* and *XSLTs*. This is done by clicking the filter icon . The following is an example of filtering by *Feeds* and *XSLTs*.

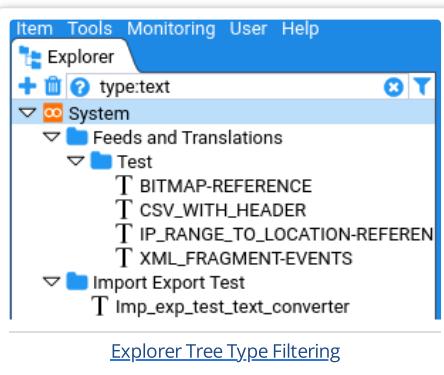


Clicking *All/None* toggles between all types selected and no types selected.

Filtering by type can also be achieved using the Quick Filter by entering the type name (or a partial form of the type name), prefixed with type: . i.e:

```
type:feed
```

For example:



NOTE: If both the type picker and the Quick Filter are used to filter on type then the two filters will be combined as an AND.

3.8.1.2 Filtering by Name

The *Explorer Tree* can be filtered by the name of the entity. This is done by entering some text in the *Quick Filter* field. The tree will then be updated to only show entities matching the *Quick Filter*. The way the matching works for entity names is described in [Common Fuzzy Matching](#)

3.8.1.3 Filtering by UUID

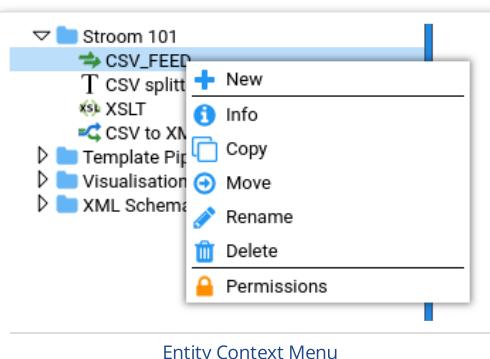
3.8.1.3.1 What is a UUID?

The *Explorer Tree* can be filtered by the UUID of the entity. The UUID [Universally unique identifier \(external link\)](#) is an identifier that can be relied on to be unique both within the system and universally across all other systems. Stroom uses UUIDs as the primary identifier for all content (Feeds, XSLTs, Pipelines, etc.) created in Stroom. An entity's UUID is generated randomly by Stroom upon creation and is fixed for the life of that entity.

When an entity is exported it is exported with its UUID and if it is then imported into another instance of Stroom the same UUID will be used. The name of an entity can be changed within Stroom but its UUID remains un-changed.

With the exception of *Feeds*, Stroom allows multiple entities to have the same name. This is because entities may exist that a user does not have access to see so restricting their choice of names based on existing invisible entities would be confusing. Where there are multiple entities with the same name the UUID can be used to distinguish between them.

The UUID of an entity can be viewed using the context menu for the entity. The context menu is accessed by right-clicking on the entity.



Clicking *Info* displays the entities UUID.



ctrl-c .

3.8.1.3.2 UUID Quick Filter Matching

[EntityInfo](#)

In the *Explorer Tree Quick Filter* you can filter by UUIDs in the following ways:

To show the entity matching a UUID, enter the full UUID value (with dashes) prefixed with the field qualifier `uuid` , e.g. `uuid:a95e5c59-2a3a-4f14-9b26-2911c6043028` .

To filter on part of a UUID you can do `uuid:/2a3a` to find an entity whose UUID contains `2a3a` or `uuid:^2a3a` to find an entity whose UUID starts with `2a3a` .

3.8.2 Quick Filters

Quick Filter controls are used in a number of screens in *Stroom*. The most prominent use of a Quick Filter is in the Explorer Tree as described above. Quick filters allow for quick searching of a data set or a list of items using a text based query language. The basis of the query language is described in [Common Fuzzy Matching](#).

A number of the Quick Filters are used for filter tables of data that have a number of fields. The quick filter query language supports matching in specified fields. Each Quick Filter will have a number of named fields that it can filter on. The field to match on is specified by prefixing the match term with the name of the field followed by a `:` , i.e. `type:` . Multiple field matches can be used, each separate by a space. E.g:

```
name:^xml name:events$ type:feed
```

In the above example the filter will match on items with a name beginning `xml` , a name ending `events` and a type partially matching `feed` .

All the match terms are combined together with an AND operator. The same field can be used multiple times in the match. The list of filterable fields and their qualifier names (sometimes a shortened form) are listed by clicking on the help icon .

One or more of the fields will be defined as *default* fields. This means the if no qualifier is entered the match will be applied to all *default* fields using an OR operator. Sometimes all fields may be considered *default* which means a match term will be tested against all fields and an item will be included in the results if one or more of those fields match.

For example if the Quick Filter has fields `Name` , `Type` and `Status` , of which `Name` and `Type` are *default*:

```
feed status:ok
```

The above would match items where the Name OR Type fields match `feed` AND the Status field matches `ok` .

3.8.2.1 Match Negation

Each match item can be negated using the `!` prefix. This is also described in [Common Fuzzy Matching](#). The prefix is applied **after** the field qualifier. E.g:

```
name:xml source:!-/default
```

In the above example it would match on items where the Name field matched `xml` and the Source field does NOT match the regex pattern `default` .

3.8.2.2 Spaces and Quotes

If your match term contains a space then you can surround the match term with double quotes. Also if your match term contains a double quote you can escape it with a \ character. The following would be valid for example.

```
"name:csv splitter" "default field match" "symbol:\\""
```

3.8.2.3 Multiple Terms

If multiple terms are provided for the same field then an AND is used to combine them. This can be useful where you are not sure of the order of words within the items being filtered.

For example:

User input: spain plain rain

Will match:

```
The rain in spain stays mainly in the plain
 ^^^^^^      ^^^^^^          ^^^^^^
rainsplainplain
 ^^^^^^^^^^^^^^

spain plain rain
 ^^^^^^ ^^^^^^ ^^^^

raining spain plain
 ^^^^^^ ^^^^^^ ^^^^
```

Won't match: sprain , rain , spain

3.8.2.4 OR Logic

There is no support for combining terms with an OR. However you can achieve this using a single regular expression term. For example

User input: status:/(disabled|locked)

Will match:

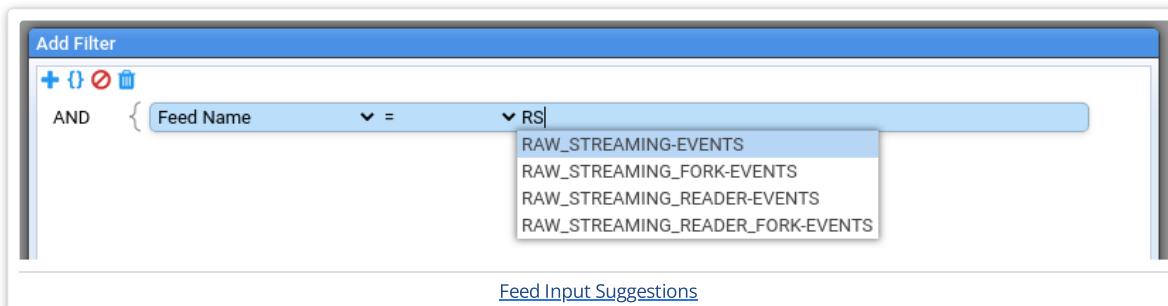
```
Locked
 ^^^^^^

Disabled
 ^^^^^^
```

Won't match: A MAN , HUMAN

3.8.3 Suggestion Input Fields

Stroom uses a number of suggestion input fields, such as when selecting Feeds, Pipelines, types, status values, etc. in the pipeline processor filter screen.



These fields will typically display the full list of values or a truncated list where the total number of values is too large. Entering text in one of these fields will use the fuzzy matching algorithm to partially/fully match on values. See [Common Fuzzy Matching](#) below for details of how the matching works.

3.8.4 Common Fuzzy Matching

A common fuzzy matching mechanism is used in a number of places in *Stroom*. It is used for partially matching the user input to a list of a list of possible values.

In some instances, the list of matched items will be truncated to a more manageable size with the expectation that the filter will be refined.

The fuzzy matching employs a number of approaches that are attempted in the following order:

NOTE: In the following examples the ^ character is used to indicate which characters have been matched.

3.8.4.1 No Input

If no input is provided all items will match.

3.8.4.2 Contains (Default)

If no prefixes or suffixes are used then all characters in the user input will need to be contained as a whole somewhere within the string being tested. The matching is case insensitive.

User input: bad

Will match:

```
bad angry dog
^^^
BAD
^^^
very badly
^^^
Very bad
^^^
```

Won't match: dab , ba d , ba

3.8.4.3 Characters Anywhere Matching

If the user input is prefixed with a ~ (tilde) character then characters anywhere matching will be employed. The matching is case insensitive.

User input: bad

Will match:

```
Big Angry Dog
^ ^ ^
bad angry dog
^^^
BAD
^^^
badly
^^^
Very bad
^^^
b a d
^ ^ ^
bbaadd
^ ^ ^
```

Won't match: dab , ba

3.8.4.4 Word Boundary Matching

If the user input is prefixed with a `?` character then word boundary matching will be employed. This approach uses upper case letters to denote the start of a *word*. If you know the some or all of *words* in the item you are looking for, and their order, then condensing those *words* down to their first letters (capitalised) makes this a more targeted way to find what you want than the characters anywhere matching above. Words can either be separated by characters like `_ - ()[].`, or be distinguished with `lowerCamelCase` or `upperCamelCase` format. An upper case letter in the input denotes the beginning of a *word* and any subsequent lower case characters are treated as contiguously following the character at the start of the word.

User input: ?TheiMa

Will match:

```
the cat sat on their mat
^ ^^^^ ^^

ON THEIR MAT
^ ^^^^ ^^

of their magic
^ ^^^^ ^^

o thei ma
^ ^^^^ ^^

onTheirMat
^ ^^^^ ^^

OnTheirMat
^ ^^^^ ^^
```

Won't match: On the mat , the cat sat on there mat , On their moat

User input: ?MFN

Will match:

```
MY_FEED_NAME
^ ^ ^

MY FEED NAME
^ ^ ^

MY_FEED_OTHER_NAME
^ ^ ^

THIS_IS_MY_FEED_NAME_TOO
^ ^ ^

myFeedName
^ ^ ^

MyFeedName
^ ^ ^

also-my-feed-name
^ ^ ^

MFN
^ ^ ^

stroom.something.somethingElse.maxFileNumber
^ ^ ^
```

Won't match: myfeedname , MY FEEDNAME

3.8.4.5 Regular Expression Matching

If the user input is prefixed with a `/` character then the remaining user input is treated as a Java syntax regular expression. A string will be considered a match if any part of it matches the regular expression pattern. The regular expression operates in case insensitive mode. For more details on the syntax of java regular expressions see this internet link

<https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/util/regex/Pattern.html>.

User input: /(^|wo)man

Will match:

```
MAN
^^^
A WOMAN
^^^^^
Manly
^^^
Womanly
^^^^^
```

Won't match: A MAN , HUMAN

3.8.4.6 Exact Match

If the user input is prefixed with a ^ character and suffixed with a \$ character then a case-insensitive exact match will be used. E.g:

User input: ^xml-events\$

Will match:

```
xml-events
^^^^^^^^^
XML-EVENTS
^^^^^^^^^
```

Won't match: xslt-events , XML EVENTS , SOME-XML-EVENTS , AN-XML-EVENTS-PIPELINE

Note: Despite the similarity in syntax, this is NOT regular expression matching.

3.8.4.7 Starts With

If the user input is prefixed with a ^ character then a case-insensitive starts with match will be used. E.g:

User input: ^events

Will match:

```
events
^^^^^
EVENTS_FEED
^^^^^
events-xslt
^^^^^
```

Won't match: xslt-events , JSON_EVENTS

Note: Despite the similarity in syntax, this is NOT regular expression matching.

3.8.4.8 Ends With

If the user input is suffixed with a \$ character then a case-insensitive ends with match will be used. E.g:

User input: events\$

Will match:

```
events
^^^^^
xslt-events
^^^^^
JSON_EVENTS
^^^^^
```

Won't match: EVENTS_FEED , events-xslt

3.8.4.9 Wild-Carded Case Sensitive Exact Matching

If one or more * characters are found in the user input then this form of matching will be used.

This form of matching is to support those fields that accept wild-carded values, e.g. a wild-carded feed name expression term. In this instance you are NOT picking a value from the suggestion list but entering a wild-carded value that will be evaluated when the expression/filter is actually used. The user may want an expression term that matches on all feeds starting with XML_, in which case they would enter XML_* . To give an indication of what it would match on if the list of feeds remains the same, the list of suggested items will reflect the wild-carded input.

User input: XML_*

Will match:

```
XML_
^^^
XML_EVENTS
^^^
```

Won't match: BAD_XML_EVENTS , XML-EVENTS , xml_events

User input: XML_*EVENTS*

Will match:

```
XML_EVENTS
^^^^^^^^^
XML_SEC_EVENTS
^^^      ^^^^
XML_SEC_EVENTS_FEED
^^^      ^^^^
```

Won't match: BAD_XML_EVENTS , xml_events

3.8.4.10 Match Negation

A match can be negated, ie. the NOT operator using the prefix ! . This prefix can be applied before all the match prefixes listed above. E.g:

```
!/(error|warn)
```

In the above example it will match everything except those matched by the regex pattern (error|warn) .

3.9 - Nodes

Configuring the nodes in a Stroom cluster.

All nodes in an Stroom cluster must be configured correctly for them to communicate with each other.

3.9.1 Configuring nodes

Open Monitoring/Nodes from the top menu. The nodes screen looks like this:

TODO

Screenshot

You need to edit each line by selecting it and then clicking the edit  icon at the bottom. The URL for each node needs to be set as above but obviously substituting in the host name of the individual node, e.g. `http://<HOST_NAME>:8080/stroom/clustercall.rpc`

Nodes are expected communicate with each other on port 8080 over http. Ensure you have configured your firewall to allow nodes to talk to each other over this port. You can configure the URL to use a different port and possibly HTTPS but performance will be better with HTTP as no SSL termination is required.

Once you have set the URLs of each node you should also set the master assignment priority for each node to be different to all of the others. In the image above the priorities have been set in a random fashion to ensure that node3 assumes the role of master node for as long as it is enabled. You also need to check all of the nodes are enabled that you want to take part in processing or any other jobs.

Keep refreshing the table until all nodes show healthy pings as above. If you do not get ping results for each node then they are not configured correctly.

Once a cluster is configured correctly you will get proper distribution of processing tasks and search will be able to access all nodes to take part in a distributed query.

3.10 - Pipelines

Pipelines are the mechanism for processing and transforming ingested data.

Pipelines

Every feed has an associated translation. The translation is used to convert the input text or XML into event logging XML format.

XSLT is used to translate from XML to event logging XML.

3.10.1 - File Output

When outputting files with Stroom, the output file names and paths can include various substitution variables to form the file and path names.

3.10.1.1 Context Variables

The following replacement variables are specific to the current processing context.

- `#{feed}` - The name of the feed that the stream being processed belongs to
- `#{pipeline}` - The name of the pipeline that is producing output
- `#{sourceId}` - The id of the input data being processed
- `#{partNo}` - The part number of the input data being processed where data is in aggregated batches
- `#{searchId}` - The id of the batch search being performed. This is only available during a batch search
- `#{node}` - The name of the node producing the output

3.10.1.2 Time Variables

The following replacement variables can be used to include aspects of the current time in UTC.

- `#{year}` - Year in 4 digit form, e.g. 2000
- `#{month}` - Month of the year padded to 2 digits
- `#{day}` - Day of the month padded to 2 digits
- `#{hour}` - Hour padded to 2 digits using 24 hour clock, e.g. 22
- `#{minute}` - Minute padded to 2 digits
- `#{second}` - Second padded to 2 digits
- `#{millis}` - Milliseconds padded to 3 digits
- `#{ms}` - Milliseconds since the epoch

3.10.1.3 System (Environment) Variables

System variables (environment variables) can also be used, e.g. `#{TMP}`.

3.10.1.4 File Name References

rolledFileName in RollingFileAppender can use references to the fileName to incorporate parts of the non rolled file name.

- `#{fileName}` - The complete file name
- `#{fileStem}` - Part of the file name before the file extension, i.e. everything before the last ''
- `#{fileExtension}` - The extension part of the file name, i.e. everything after the last ''

3.10.1.5 Other Variables

- `#{uuid}` - A randomly generated UUID to guarantee unique file names

3.10.2 - Parser

Parsing input data.

The following capabilities are available to parse input data:

- XML - XML input can be parsed with the XML parser.
- [XML Fragment](#) - Treat input data as an XML fragment, i.e. XML that does not have an XML declaration or root elements.
- [Data Splitter](#) - Delimiter and regular expression based language for turning non XML data into XML (e.g. CSV)

3.10.2.1 - Context Data

##Context File

###Input File:

```
<?xml version="1.0" encoding="UTF-8"?>
<SomeData>
    <SomeEvent>
        <SomeTime>01/01/2009:12:00:01</SomeTime>
        <SomeAction>OPEN</SomeAction>
        <SomeUser>userone</SomeUser>
        <SomeFile>D:\TranslationKit\example\VerySimple\OpenFileEvents.txt</SomeFile>
    </SomeEvent>
</SomeData>
```

###Context File:

```
<?xml version="1.0" encoding="UTF-8"?>
<SomeContext>
    <Machine>MyMachine</Machine>
</SomeContext>
```

###Context XSLT:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet
    xmlns="reference-data:2"
    xmlns:evt="event-logging:3"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">

    <xsl:template match="SomeContext">
        <referenceData
            xsi:schemaLocation="event-logging:3 file://event-logging-v3.0.0.xsd reference-data:2 file://reference-data-v
            version="2.0.1">

            <xsl:apply-templates/>
        </referenceData>
    </xsl:template>

    <xsl:template match="Machine">
        <reference>
            <map>CONTEXT</map>
            <key>Machine</key>
            <value><xsl:value-of select=". "/></value>
        </reference>
    </xsl:template>
</xsl:stylesheet>
```

###Context XML Translation:

```

<?xml version="1.0" encoding="UTF-8"?>
<referenceData xmlns:evt="event-logging:3"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="reference-data:2"
    xsi:schemaLocation="event-logging:3 file://event-logging-v3.0.0.xsd reference-data:2 file://refe
    version="2.0.1">
    <reference>
        <map>CONTEXT</map>
        <key>Machine</key>
        <value>MyMachine</value>
    </reference>
</referenceData>

```

###Input File:

```

<?xml version="1.0" encoding="UTF-8"?>
<SomeData>
    <SomeEvent>
        <SomeTime>01/01/2009:12:00:01</SomeTime>
        <SomeAction>OPEN</SomeAction>
        <SomeUser>userone</SomeUser>
        <SomeFile>D:\TranslationKit\example\VerySimple\OpenFileEvents.txt</SomeFile>
    </SomeEvent>
</SomeData>

```

###Main XSLT (Note the use of the context lookup):

```

<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet
    xmlns="event-logging:3"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    version="2.0">

    <xsl:template match="SomeData">
        <Events xsi:schemaLocation="event-logging:3 file://event-logging-v3.0.0.xsd" Version="3.0.0">
            <xsl:apply-templates/>
        </Events>
    </xsl:template>
    <xsl:template match="SomeEvent">
        <xsl:if test="SomeAction = 'OPEN'">
            <Event>
                <EventTime>
                    <TimeCreated>
                        <xsl:value-of select="s:format-date(SomeTime, 'dd/MM/yyyy:hh:mm:ss')"/>
                    </TimeCreated>
                </EventTime>
                <EventSource>
                    <System>Example</System>
                    <Environment>Example</Environment>
                    <Generator>Very Simple Provider</Generator>
                    <Device>
                        <IPAddress>182.80.32.132</IPAddress>
                        <Location>
                            <Country>UK</Country>
                            <Site><xsl:value-of select="s:lookup('CONTEXT', 'Machine')"/></Site>
                            <Building>Main</Building>
                            <Floor>1</Floor>
                            <Room>1aaa</Room>
                        </Location>
                    </Device>
                    <User><Id><xsl:value-of select="SomeUser"/></Id></User>
                </EventSource>
                <EventDetail>
                    <View>
                        <Document>
                            <Title>UNKNOWN</Title>
                            <File>
                                <Path><xsl:value-of select="SomeFile"/></Path>
                            </File>
                        </Document>
                    </View>
                </EventDetail>
            </Event>
        </xsl:if>
    </xsl:template>
</xsl:stylesheet>

```

###Main Output XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<Events xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="event-logging:3"
         xsi:schemaLocation="event-logging:3 file://event-logging-v3.0.0.xsd"
         Version="3.0.0">
    <Event Id="6:1">
        <EventTime>
            <TimeCreated>2009-01-01T00:00:01.000Z</TimeCreated>
        </EventTime>
        <EventSource>
            <System>Example</System>
            <Environment>Example</Environment>
            <Generator>Very Simple Provider</Generator>
            <Device>
                <IPAddress>182.80.32.132</IPAddress>
                <Location>
                    <Country>UK</Country>
                    <Site>MyMachine</Site>
                    <Building>Main</Building>
                    <Floor>1</Floor>
                    <Room>1aaa</Room>
                </Location>
            </Device>
            <User>
                <Id>userone</Id>
            </User>
        </EventSource>
        <EventDetail>
            <View>
                <Document>
                    <Title>UNKNOWN</Title>
                    <File>
                        <Path>D:\TranslationKit\example\VerySimple\OpenFileEvents.txt</Path>
                    </File>
                </Document>
            </View>
        </EventDetail>
    </Event>
</Events>
```

3.10.2.2 - XML Fragments

Handling XML data without root level elements.

Some input XML data may be missing an XML declaration and root level enclosing elements. This data is not a valid XML document and must be treated as an XML fragment. To use XML fragments the input type for a translation must be set to 'XML Fragment'. A fragment wrapper must be defined in the XML conversion that tells Stroom what declaration and root elements to place around the XML fragment data.

Here is an example:

```
<?xml version="1.1" encoding="UTF-8"?>
<!DOCTYPE records [
<!ENTITY fragment SYSTEM "fragment">
]>
<records
  xmlns="records:2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="records:2 file://records-v2.0.xsd"
  version="2.0">
  &fragment;
</records>
```

During conversion Stroom replaces the fragment text entity with the input XML fragment data. Note that XML fragments must still be well formed so that they can be parsed correctly.

3.10.3 - Reference Data

Performing temporal reference data lookups to decorate event data.

Version Information: Created with Stroom v7.0

Last Updated: 15 September 2020

See Also:

- [HOWTO - Creating a Simple Reference Feed](#)
- [XSLT Functions](#)

In Stroom reference data is primarily used to decorate events using `stroom:lookup()` calls in XSLTs. For example you may have reference data feed that associates the FQDN of a device to the physical location. You can then perform a `stroom:lookup()` in the XSLT to decorate an event with the physical location of a device by looking up the FQDN found in the event.

Reference data is time sensitive and each stream of reference data has an Effective Date set against it. This allows reference data lookups to be performed using the date of the event to ensure the reference data that was actually effective at the time of the event is used.

Using reference data involves the following steps/processes:

- Ingesting the raw reference data into Stroom.
- Creating (and processing) a pipeline to transform the raw reference into `reference-data:2` format XML.
- Creating a reference loader pipeline with a Reference Data Filter element to load *cooked* reference data into the reference data store.
- Adding reference pipeline/feeds to an XSLT Filter in your event pipeline.
- Adding the lookup call to the XSLT.
- Processing the raw events through the event pipeline.

The process of creating a reference data pipeline is described in the HOWTO linked at the top of this document.

3.10.3.1 Reference Data Structure

A reference data entry essentially consists of the following:

- **Effective time** - The data/time that the entry was effective from, i.e the time the raw reference data was received.
- **Map name** - A unique name for the key/value map that the entry will be stored in. The name only needs to be unique within all map names that may be loaded within an XSLT Filter. In practice it makes sense to keep map names globally unique.
- **Key** - The text that will be used to lookup the value in the reference data map. Mutually exclusive with **Range**.
- **Range** - The inclusive range of integer keys that the entry applies to. Mutually exclusive with **Key**.
- **Value** - The value can either be simple text, e.g. an IP address, or an XML fragment that can be inserted into another XML document. XML values must be correctly namespaced.

The following is an example of some reference data that has been converted from its raw form into `reference-data:2` XML. In this example the reference data contains three entries that each belong to a different map. Two of the entries are simple text values and the last has an XML value.

```

<?xml version="1.1" encoding="UTF-8"?>
<referenceData
    xmlns="reference-data:2"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:stroom="stroom"
    xmlns:evt="event-logging:3"
    xsi:schemaLocation="reference-data:2 file://reference-data-v2.0.xsd"
    version="2.0.1">

    <!-- A simple string value -->
    <reference>
        <map>FQDN_TO_IP</map>
        <key>stroomnode00.strmdev00.org</key>
        <value>
            <IPAddress>192.168.2.245</IPAddress>
        </value>
    </reference>

    <!-- A simple string value -->
    <reference>
        <map>IP_TO_FQDN</map>
        <key>192.168.2.245</key>
        <value>
            <HostName>stroomnode00.strmdev00.org</HostName>
        </value>
    </reference>

    <!-- A key range -->
    <reference>
        <map>USER_ID_TO_COUNTRY_CODE</map>
        <range>
            <from>1</from>
            <to>1000</to>
        </range>
        <value>GBR</value>
    </reference>

    <!-- An XML fragment value -->
    <reference>
        <map>FQDN_TO_LOC</map>
        <key>stroomnode00.strmdev00.org</key>
        <value>
            <evt:Location>
                <evt:Country>GBR</evt:Country>
                <evt:Site>Bristol-S00</evt:Site>
                <evt:Building>GZero</evt:Building>
                <evt:Room>R00</evt:Room>
                <evt:TimeZone>+00:00/+01:00</evt:TimeZone>
            </evt:Location>
        </value>
    </reference>
</referenceData>

```

3.10.3.1.1 Reference Data Namespaces

When XML reference data values are created, as in the example XML above, the XML values must be qualified with a namespace to distinguish them from the `reference-data:2` XML that surrounds them. In the above example the XML fragment will become as follows when injected into an event:

```

<evt:Location xmlns:evt="event-logging:3" >
  <evt:Country>GBR</evt:Country>
  <evt:Site>Bristol-S00</evt:Site>
  <evt:Building>GZero</evt:Building>
  <evt:Room>R00</evt:Room>
  <evt:TimeZone>+00:00/+01:00</evt:TimeZone>
</evt:Location>

```

Even if `evt` is already declared in the XML being injected into it, if it has been declared for the reference fragment then it will be explicitly declared in the destination. While duplicate namespace may appear odd it is valid XML.

The namespace can also be achieved like this:

```

<?xml version="1.1" encoding="UTF-8"?>
<referenceData
  xmlns="reference-data:2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:stroom="stroom"
  xsi:schemaLocation="reference-data:2 file://reference-data-v2.0.xsd"
  version="2.0.1">

  <!-- An XML value -->
  <reference>
    <map>FQDN_TO_LOC</map>
    <key>stroomnode00.strmdev00.org</key>
    <value>
      <Location xmlns="event-logging:3">
        <Country>GBR</Country>
        <Site>Bristol-S00</Site>
        <Building>GZero</Building>
        <Room>R00</Room>
        <TimeZone>+00:00/+01:00</TimeZone>
      </Location>
    </value>
  </reference>
</referenceData>

```

This reference data will be injected into event XML exactly as it, i.e.:

```

<Location xmlns="event-logging:3">
  <Country>GBR</Country>
  <Site>Bristol-S00</Site>
  <Building>GZero</Building>
  <Room>R00</Room>
  <TimeZone>+00:00/+01:00</TimeZone>
</Location>

```

3.10.3.2 Reference Data Storage

Reference data is stored in two different places on a Stroom node. All reference data is only visible to the node where it is located. Each node that is performing reference data lookups will need to load and store its own reference data. While this will result in duplicate data being held by nodes it makes the storage of reference data and its subsequent lookup very performant.

3.10.3.2.1 On-Heap Store

The On-Heap store is the reference data store that is held in memory in the Java Heap. This store is volatile and will be lost on shut down of the node. The On-Heap store is only used for storage of context data.

3.10.3.2.2 Off-Heap Store

The Off-Heap store is the reference data store that is held in memory outside of the Java Heap and is persisted to local disk. As the store is also persisted to local disk it means the reference data will survive the shutdown of the Stroom instance. Storing the data off-heap means Stroom can run with a much smaller Java Heap size.

The Off-Heap store is based on the Lightning Memory-Mapped Database (LMDB). LMDB makes use of the Linux page cache to ensure that hot portions of the reference data are held in the page cache (making use of all available free memory). Infrequently used portions of the reference data will be evicted from the page cache by the Operating System. Given that LMDB utilises the page cache for holding reference data in memory the more free memory the host has the better as there will be less shifting of pages in/out of the OS page cache. When storing large amounts of data you may experience the OS reporting very little free memory as a large amount will be in use by the page cache. This is not an issue as the OS will evict pages when memory is needed for other applications, e.g. the Java Heap.

3.10.3.2.2.1 Local Disk

The Off-Heap store is intended to be located on local disk on the Stroom node. The location of the store is set using the property `stroom.pipeline.referenceData.localDir`. Using LMDB on remote storage is NOT advised, see <http://www.lmdb.tech/doc>.

If you are running stroom on AWS EC2 instances then you will need to attach some local instance storage to the host, e.g. SSD, to use for the reference data store. In tests EBS storage was found to be VERY slow. It should be noted that AWS instance storage is not persistent between instance stops, terminations and hardware failure. However any loss of the reference data store will mean that the next time Stroom boots a new store will be created and reference data will be loaded on demand as normal.

3.10.3.2.2.2 Transactions

LMDB is a transactional database with ACID semantics. All interaction with LMDB is done within a read or write transaction. There can only be one write transaction at a time so if there are a number of concurrent reference data loads then they will have to wait in line. Read transactions, i.e. lookups, are not blocked by each other or by write transactions so once the data is loaded and is in memory lookups can be performed very quickly.

3.10.3.2.2.3 Read-Ahead Mode

When data is read from the store, if the data is not already in the page cache then it will be read from disk and added to the page cache by the OS. Read-ahead is the process of speculatively reading ahead to load more pages into the page cache than were requested. This is on the basis that future requests for data may need the pages speculatively read into memory. If the reference data store is very large or is larger than the available memory then it is recommended to turn read-ahead off as the result will be to evict hot reference data from the page cache to make room for speculative pages that may not be needed. It can be tuned off with the system property `stroom.pipeline.referenceData.readAheadEnabled`.

3.10.3.2.2.4 Key Size

When reference data is created care must be taken to ensure that the *Key* used for each entry is less than 507 bytes. For simple ASCII characters then this means less than 507 characters. If non-ASCII characters are in the key then these will take up more than one byte per character so the length of the key in characters will be less. This is a limitation inherent to LMDB.

3.10.3.2.2.5 Commit intervals

The property `stroom.pipeline.referenceData.maxPutsBeforeCommit` controls the number of entries that are put into the store between each commit. As there can be only one transaction writing to the store at a time, committing periodically allows other process to jump in and make writes. There is a trade off though as reducing the number of entries put between each commit can seriously affect performance. For the fastest single process performance a value of zero should be used which means it will not commit mid-load. This however means all other processes wanting to write to the store will need to wait.

3.10.3.2.2.6 Cloning The Off Heap Store

If you are provisioning a new stroom node it is possible to copy the off heap store from another node. Stroom should not be running on the node being copied from. Simply copy the contents of `stroom.pipeline.referenceData.localDir` into the same configured location on the other node. The new node will use the copied store and have access to its reference data.

3.10.3.2.2.7 Store Size & Compaction

Due to the way LMDB works the store can only grow in size, it will never shrink, even if data is deleted. Deleted data frees up space for new writes to the store so will be reused but never freed. If there is a regular process of purging old data and adding new reference data then this should not be an issue as the new reference data will use the space made available by the purged data.

If store size becomes an issue then it is possible to compact the store. `lmdb-utils` is available on some package managers and this has an `mdb_copy` command that can be used with the `-c` switch to copy the LMDB environment to a new one, compacting it in the process. This should be done when Stroom is down to avoid writes happening to the store while the copy is happening.

Given that the store is essentially a cache and all data can be re-loaded another option is to delete the contents of `stroom.pipeline.referenceData.localDir` when Stroom is not running. On boot Stroom will create a brand new store and reference data will be re-loaded as required.

3.10.3.3 The Loading Process

Reference data is loaded into the store on demand during the processing of a `stroom:lookup()` method call. Reference data will only be loaded if it does not already exist in the store, however it is always loaded as a complete stream, rather than entry by entry.

The test for existence in the store is based on the following criteria:

- The UUID of the reference loader pipeline.
- The version of the reference loader pipeline.
- The Stream ID for the stream of reference data that has been deemed effective for the lookup.
- The Stream Number (in the case of multi part streams).

If a reference stream has already been loaded matching the above criteria then no additional load is required.

IMPORTANT: It should be noted that as the version of the reference data pipeline forms part of the criteria, if the reference loader pipeline is changed, for whatever reason, then this will invalidate ALL existing reference data associated with that reference loader pipeline.

Typically the reference loader pipeline is very static so this should not be an issue.

Standard practice is to convert raw reference data into `reference:2` XML on receipt using a pipeline separate to the reference loader. The reference loader is then only concerned with reading cooked `reference:2` into the Reference Data Filter.

In instances where reference data streams are infrequently used it may be preferable to not convert the raw reference on receipt but instead to do it in the reference loader pipeline.

3.10.3.3.1 Duplicate Keys

The Reference Data Filter pipeline element has a property `overrideExistingValues` which if set to `true` means if an entry is found in an effective stream with the same key as an entry already loaded then it will overwrite the existing one. Entries are loaded in the order they are found in the `reference:2` XML document. If set to `false` then the existing entry will be kept. If `warnOnDuplicateKeys` is set to `true` then a warning will be logged for any duplicate keys, whether an overwrite happens or not.

3.10.3.3.2 Value De-Duplication

Only unique values are held in the store to reduce the storage footprint. This is useful given that typically, reference data updates may be received daily and each one is a full snapshot of the whole reference data. As a result this can mean many copies of the same value being loaded into the store. The store will only hold the first instance of duplicate values.

3.10.3.4 Querying the Reference Data Store

The reference data store can be queried within a Dashboard in Stroom by selecting `Reference Data Store` in the data source selection pop-up. Querying the store is currently an experimental feature and is mostly intended for use in fault finding. Given the localised nature of the reference data store the dashboard can currently only query the store on the node that the user interface is being served from. In a multi-node environment where some nodes are UI only and most are processing only, the UI nodes will have no reference data in their store.

3.10.3.5 Purging Old Reference Data

Reference data loading and purging is done at the level of a reference stream. Whenever a reference lookup is performed the last accessed time of the reference stream in the store is checked. If it is older than one hour then it will be updated to the current time. This last access time is used to determine reference streams that are no longer in active use and thus can be purged.

The Stroom job `Ref Data Off-heap Store Purge` is used to perform the purge operation on the Off-Heap reference data store. No purge is required for the On-Heap store as that only holds transient data. When the purge job is run it checks the time since each reference stream was accessed against the purge cut-off age. The purge age is configured via the property `stroom.pipeline.referenceData.purgeAge`. It is advised to schedule this job for quiet times when it is unlikely to conflict with reference loading operations as they will fight for access to the single write transaction.

3.10.3.6 Lookups

Lookups are performed in XSLT Filters using the XSLT functions. In order to perform a lookup one or more reference feeds must be specified on the XSLT Filter pipeline element. Each reference feed is specified along with a reference loader pipeline that will ingest the specified feed (optional convert it into `reference:2 XML` if it is not already) and pass it into a Reference Data Filter pipeline element.

3.10.3.6.1 Reference Feeds & Loaders

In the XSLT Filter pipeline element multiple combinations of feed and reference loader pipeline can be specified. There must be at least one in order to perform lookups. If there are multiple then when a lookup is called for a given time, the effective stream for each feed/loader combination is determined. The effective stream for each feed/loader combination will be loaded into the store, unless it is already present.

When the actual lookup is performed Stroom will try to find the key in each of the effective streams that have been loaded and that contain the map in the lookup call. If the lookup is unsuccessful in the effective stream for the first feed/loader combination then it will try the next, and so on until it has tried all of them. For this reason if you have multiple feed/loader combinations then order is important. It is possible for multiple effective streams to contain the same map/key so a feed/loader combination higher up the list will trump one lower down with the same map/key. Also if you have some lookups that may not return a value and others that should always return a value then the feed/loader for the latter should be higher up the list so it is searched first.

3.10.3.6.2 Standard Key/Value Lookups

Standard key/value lookups consist of a simple string key and a value that is either a simple string or an XML fragment. Standard lookups are performed using the various forms of the [`stroom:lookup\(\)`](#) XSLT function.

3.10.3.6.3 Range Lookups

Range lookups consist of a key that is an integer and a value that is either a simple string or an XML fragment. For more detail on range lookups see the XSLT function [`stroom:lookup\(\)`](#).

3.10.3.6.4 Nested Map Lookups

Nested map lookups involve chaining a number of lookups with the value of each map being used as the key for the next. For more detail on nested lookups see the XSLT function [`stroom:lookup\(\)`](#).

3.10.3.6.5 Bitmap Lookups

A bitmap lookup is a special kind of lookup that actually performs a lookup for each enabled bit position of the passed bitmap value. For more detail on bitmap lookups see the XSLT function [`stroom:bitmap-lookup\(\)`](#).

Values can either be a simple string or an XML fragment.

3.10.3.6.6 Context data lookups

Some event streams have a Context stream associated with them. Context streams allow the system sending the events to Stroom to supply an additional stream of data that provides context to the raw event stream. This can be useful when the system sending the events has no control over the event content but needs to supply additional information. The context stream can be used in lookups as a reference source to decorate events on receipt. Context reference data is specific to a single event stream so is transient in nature, therefore the On Heap Store is used to hold it for the duration of the event stream processing only.

Typically the reference loader for a context stream will include a translation step to convert the raw context data into `reference:2 XML`.

3.10.3.7 Reference Data API

The reference data store has an API to allow other systems to access the reference data store. The `lookup` endpoint requires the caller to provide details of the reference feed and loader pipeline so if the effective stream is not in the store it can be loaded prior to performing the lookup.

Below is an example of a lookup request.

```
{  
  "mapName": "USER_ID_TO_LOCATION",  
  "effectiveTime": "2020-12-02T08:37:02.772Z",  
  "key": "jbloggs",  
  "referenceLoaders": [  
    {  
      "loaderPipeline" : {  
        "name" : "Reference Loader",  
        "uuid" : "da1c7351-086f-493b-866a-b42dbe990700",  
        "type" : "Pipeline"  
      },  
      "referenceFeed" : {  
        "name": "USER_ID_TOLOCATION-REFERENCE",  
        "uuid": "60f9f51d-e5d6-41f5-86b9-ae866b8c9fa3",  
        "type" : "Feed"  
      }  
    }  
  ]  
}
```

3.10.4 - XSLT Conversion

Using Extensible Stylesheet Language Transformations (XSLT) to transform data.

Once the text file has been converted into Intermediary XML (or the feed is already XML), XSLT is used to translate the XML into event logging XML format.

Event Feeds must be translated into the events schema and Reference into the reference schema. You can browse documentation relating to the schemas within the application.

Here is an example XSLT:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet
    xmlns="event-logging:3"
    xmlns:s="stroom"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    version="2.0">

    <xsl:template match="SomeData">
        <Events
            xsi:schemaLocation="event-logging:3 file://event-logging-v3.0.0.xsd"
            Version="3.0.0">
            <xsl:apply-templates/>
        </Events>
    </xsl:template>
    <xsl:template match="SomeEvent">
        <xsl:variable name="dateTime" select="SomeTime"/>
        <xsl:variable name="formattedDateTime" select="s:format-date($dateTime, 'dd/MM/yyyyhh:mm:ss')"/>

        <xsl:if test="SomeAction = 'OPEN'">
            <Event>
                <EventTime>
                    <TimeCreated>
                        <xsl:value-of select="$formattedDateTime"/>
                    </TimeCreated>
                </EventTime>
                <EventSource>
                    <System>Example</System>
                    <Environment>Example</Environment>
                    <Generator>Very Simple Provider</Generator>
                    <Device>
                        <IPAddress>3.3.3.3</IPAddress>
                    </Device>
                    <User>
                        <Id><xsl:value-of select="SomeUser"/></Id>
                    </User>
                </EventSource>
                <EventDetail>
                    <View>
                        <Document>
                            <Title>UNKNOWN</Title>
                            <File>
                                <Path><xsl:value-of select="SomeFile"/></Path>
                                <File>
                            </Document>
                        </View>
                    </EventDetail>
                </Event>
            </xsl:if>
        </xsl:template>
    </xsl:stylesheet>
```

3.10.4.1 - XSLT Functions

Custom XSLT functions available in Stroom.

By including the following namespace:

```
xmlns:s="stroom"
```

E.g.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet
    xmlns="event-logging:3"
    xmlns:s="stroom"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    version="2.0">
```

The following functions are available to aid your translation:

- [bitmap-lookup\(String map, String key\)](#) - Bitmap based look up against reference data map using the period start time
- [bitmap-lookup\(String map, String key, String time\)](#) - Bitmap based look up against reference data map using a specified time, e.g. the event time
- [bitmap-lookup\(String map, String key, String time, Boolean ignoreWarnings\)](#) - Bitmap based look up against reference data map using a specified time, e.g. the event time, and ignore any warnings generated by a failed lookup
- [bitmap-lookup\(String map, String key, String time, Boolean ignoreWarnings, Boolean trace\)](#) - Bitmap based look up against reference data map using a specified time, e.g. the event time, and ignore any warnings generated by a failed lookup and get trace information for the path taken to resolve the lookup.
- [classification\(\)](#) - The classification of the feed for the data being processed
- [col-from\(\)](#) - The column in the input that the current record begins on (can be 0).
- [col-to\(\)](#) - The column in the input that the current record ends at.
- [current-time\(\)](#) - The current system time
- [current-user\(\)](#) - The current user logged into Stroom (only relevant for interactive use, e.g. search)
- [decode-url\(String encodedUrl\)](#) - Decode the provided url.
- [dictionary\(String name\)](#) - Loads the contents of the named dictionary for use within the translation
- [encode-url\(String url\)](#) - Encode the provided url.
- [feed-name\(\)](#) - Name of the feed for the data being processed
- [format-date\(String date, String pattern\)](#) - Format a date that uses the specified pattern using the default time zone
- [format-date\(String date, String pattern, String timeZone\)](#) - Format a date that uses the specified pattern with the specified time zone
- [format-date\(String date, String patternIn, String timeZoneIn, String patternOut, String timeZoneOut\)](#) - Parse a date with the specified input pattern and time zone and format the output with the specified output pattern and time zone
- [format-date\(String milliseconds\)](#) - Format a date that is specified as a number of milliseconds since a standard base time known as "the epoch", namely January 1, 1970, 00:00:00 GMT
- [get\(String key\)](#) - Returns the value associated with a key that has been stored using put()
- [hash\(String value\)](#) - Hash a string value using the default SHA-256 algorithm and no salt
- [hash\(String value, String algorithm, String salt\)](#) - Hash a string value using the specified hashing algorithm and supplied salt value. Supported hashing algorithms include SHA-256, SHA-512, MD5.
- [hex-to-dec\(String hex\)](#) - Convert hex to dec representation
- [hex-to-oct\(String hex\)](#) - Convert hex to oct representation
- [json-to-xml\(String json\)](#) - Returns an XML representation of the supplied JSON value for use in XPath expressions
- [line-from\(\)](#) - The line in the input that the current record begins on (1 based).
- [line-to\(\)](#) - The line in the input that the current record ends at.
- [link\(String url\)](#) - Creates a stroom dashboard table link.
- [link\(String title, String url\)](#) - Creates a stroom dashboard table link.
- [link\(String title, String url, String type\)](#) - Creates a stroom dashboard table link.
- [log\(String severity, String message\)](#) - Logs a message to the processing log with the specified severity

- `lookup(String map, String key)` - Look up a reference data map using the period start time
- `lookup(String map, String key, String time)` - Look up a reference data map using a specified time, e.g. the event time
- `lookup(String map, String key, String time, Boolean ignoreWarnings)` - Look up a reference data map using a specified time, e.g. the event time, and ignore any warnings generated by a failed lookup
- `lookup(String map, String key, String time, Boolean ignoreWarnings, Boolean trace)` - Look up a reference data map using a specified time, e.g. the event time, ignore any warnings generated by a failed lookup and get trace information for the path taken to resolve the lookup.
- `meta(String key)` - Lookup a meta data value for the current stream using the specified key. The key can be `Feed`, `StreamType`, `CreatedTime`, `EffectiveTime`, `Pipeline` or any other attribute supplied when the stream was sent to Stroom, e.g. `meta('System')`.
- `numeric-ip(String ipAddress)` - Convert an IP address to a numeric representation for range comparison
- `part-no()` - The current part within a multi part aggregated input stream (AKA the substream number) (1 based)
- `parse-uri(String URI)` - Returns an XML structure of the URI providing `authority`, `fragment`, `host`, `path`, `port`, `query`, `scheme`, `schemeSpecificPart`, and `userInfo` components if present.
- `random()` - Get a system generated random number between 0 and 1.
- `record-no()` - The current record number within the current part (substream) (1 based).
- `search-id()` - Get the id of the batch search when a pipeline is processing as part of a batch search
- `source()` - Returns an XML structure with the `stroom-meta` namespace detailing the current source location.
- `source-id()` - Get the id of the current input stream that is being processed
- `stream-id()` - An alias for `source-id` included for backward compatibility.
- `pipeline-name()` - Name of the current processing pipeline using the XSLT
- `put(String key, String value)` - Store a value for use later on in the translation

3.10.4.1.1 bitmap-lookup()

The `bitmap-lookup()` function looks up a bitmap key from reference or context data a value (which can be an XML node set) for each set bit position and adds it to the resultant XML.

```
bitmap-lookup(String map, String key)
bitmap-lookup(String map, String key, String time)
bitmap-lookup(String map, String key, String time, Boolean ignoreWarnings)
bitmap-lookup(String map, String key, String time, Boolean ignoreWarnings, Boolean trace)
```

- `map` - The name of the reference data map to perform the lookup against.
- `key` - The bitmap value to lookup. This can either be represented as a decimal integer (e.g. `14`) or as hexadecimal by prefixing with `0x` (e.g. `0xE`).
- `time` - Determines which set of reference data was effective at the requested time. If no reference data exists with an effective time before the requested time then the lookup will fail. Time is in the format `yyyy-MM-dd'T'HH:mm:ss.SSSXX`, e.g. `2010-01-01T00:00:00.000Z`.
- `ignoreWarnings` - If true, any lookup failures will be ignored, else they will be reported as warnings.
- `trace` - If true, additional trace information is output as INFO messages.

If the look up fails no result will be returned.

The key is a bitmap expressed as either a decimal integer or a hexidecimal value, e.g. `14` / `0xE` is `1110` as a binary bitmap. For each bit position that is set, (i.e. has a binary value of `1`) a lookup will be performed using that bit position as the key. In this example, positions `1`, `2` & `3` are set so a lookup would be performed for these bit positions. The result of each lookup for the bitmap are concatenated together in bit position order, separated by a space.

If `ignoreWarnings` is true then any lookup failures will be ignored and it will return the value(s) for the bit positions it was able to lookup.

This function can be useful when you have a set of values that can be represented as a bitmap and you need them to be converted back to individual values. For example if you have a set of additive account permissions (e.g Admin, ManageUsers, PerformExport, etc.), each of which is associated with a bit position, then a user's permissions could be defined as a single decimal/hex bitmap value. Thus a bitmap lookup with this value would return all the permissions held by the user.

For example the reference data store may contain:

Key (Bit position)	Value
--------------------	-------

0	Administrator
---	---------------

Key (Bit position)	Value
1	Manage_Users
2	Perform_Export
3	View_Data
4	Manage_Jobs
5	Delete_Data
6	Manage_Volumes

The following are example lookups using the above reference data:

Lookup Key (decimal)	Lookup Key (Hex)	Bitmap	Result
0	0x0	0000000	-
1	0x1	0000001	Administrator
74	0x4A	1001010	Manage_Users View_Data Manage_Volumes
2	0x2	0000010	Manage_Users
96	0x60	1100000	Delete_Data Manage_Volumes

3.10.4.1.2 dictionary()

The dictionary() function gets the contents of the specified dictionary for use during translation. The main use for this function is to allow users to abstract the management of a set of keywords from the XSLT so that it is easier for some users to make quick alterations to a dictionary that is used by some XSLT, without the need for the user to understand the complexities of XSLT.

3.10.4.1.3 format-date()

The format-date() function takes a Pattern and optional TimeZone arguments and replaces the parsed contents with an XML standard Date Format. The pattern must be a Java based SimpleDateFormat. If the optional TimeZone argument is present the pattern must not include the time zone pattern tokens (z and Z). A special time zone value of "GMT/BST" can be used to guess the time based on the date (BST during British Summer Time).

E.g. Convert a GMT date time "2009/12/01 12:34:11"

```
<xsl:value-of select="s:format-date('2009/08/01 12:34:11', 'yyyy/MM/dd HH:mm:ss')"/>
```

E.g. Convert a GMT or BST date time "2009/08/01 12:34:11"

```
<xsl:value-of select="s:format-date('2009/08/01 12:34:11', 'yyyy/MM/dd HH:mm:ss', 'GMT/BST')"/>
```

E.g. Convert a GMT+1:00 date time "2009/08/01 12:34:11"

```
<xsl:value-of select="s:format-date('2009/08/01 12:34:11', 'yyyy/MM/dd HH:mm:ss', 'GMT+1:00')"/>
```

E.g. Convert a date time specified as milliseconds since the epoch "1269270011640"

```
<xsl:value-of select="s:format-date('1269270011640')"/>
```

3.10.4.1.4 link()

Create a string that represents a hyperlink for display in a dashboard table.

```
link(url)
link(title, url)
link(title, url, type)
```

Example

```
link('http://www.somehost.com/somepath')
> [http://www.somehost.com/somepath](http://www.somehost.com/somepath)
link('Click Here', 'http://www.somehost.com/somepath')
> [Click Here](http://www.somehost.com/somepath)
link('Click Here', 'http://www.somehost.com/somepath', 'dialog')
> [Click Here](http://www.somehost.com/somepath){dialog}
link('Click Here', 'http://www.somehost.com/somepath', 'dialog|Dialog Title')
> [Click Here](http://www.somehost.com/somepath){dialog|Dialog Title}
```

Type can be one of:

- dialog : Display the content of the link URL within a stroom popup dialog.
- tab : Display the content of the link URL within a stroom tab.
- browser : Display the content of the link URL within a new browser tab.
- dashboard : Used to launch a stroom dashboard internally with parameters in the URL.

If you wish to override the default title or URL of the target link in either a tab or dialog you can. Both `dialog` and `tab` types allow titles to be specified after a `|`, e.g. `dialog|My Title`.

3.10.4.1.5 log()

The `log()` function writes a message to the processing log with the specified severity. Severities of INFO, WARN, ERROR and FATAL can be used. Severities of ERROR and FATAL will result in records being omitted from the output if a RecordOutputFilter is used in the pipeline. The counts for RecWarn, RecError will be affected by warnings or errors generated in this way therefore this function is useful for adding business rules to XML output.

E.g. Warn if a SID is not the correct length.

```
<xsl:if test="string-length($sid) != 7">
  <xsl:value-of select="s:log('WARN', concat($sid, ' is not the correct length'))"/>
</xsl:if>
```

3.10.4.1.6 lookup()

The `lookup()` function looks up from reference or context data a value (which can be an XML node set) and adds it to the resultant XML.

```
lookup(String map, String key)
lookup(String map, String key, String time)
lookup(String map, String key, String time, Boolean ignoreWarnings)
lookup(String map, String key, String time, Boolean ignoreWarnings, Boolean trace)
```

- map - The name of the reference data map to perform the lookup against.
- key - The key to lookup. The key can be a simple string, an integer value in a numeric range or a nested lookup key.
- time - Determines which set of reference data was effective at the requested time. If no reference data exists with an effective time before the requested time then the lookup will fail. Time is in the format `yyyy-MM-dd'T'HH:mm:ss.SSSXX`, e.g. `2010-01-01T00:00:00.000Z`.
- ignoreWarnings - If true, any lookup failures will be ignored, else they will be reported as warnings.

- trace - If true, additional trace information is output as INFO messages.

If the look up fails no result will be returned. By testing the result a default value may be output if no result is returned.

E.g. Look up a SID given a PF

```

<xsl:variable name="pf" select="PFNumber"/>
<xsl:if test="$pf">
    <xsl:variable name="sid" select="s:lookup('PF_TO_SID', $pf, $formattedDateTime)"/>

    <xsl:choose>
        <xsl:when test="$sid">
            <User>
                <Id><xsl:value-of select="$sid"/></Id>
            </User>
        </xsl:when>
        <xsl:otherwise>
            <data name="PFNumber">
                <xsl:attribute name="Value"><xsl:value-of select="$pf"/></xsl:attribute>
            </data>
        </xsl:otherwise>
    </xsl:choose>
</xsl:if>
```

3.10.4.1.6.1 Range lookups

Reference data entries can either be stored with single string key or a key range that defines a numeric range, e.g 1-100. When a lookup is preformed the passed key is looked up as if it were a normal string key. If that lookup fails Stroom will try to convert the key to an integer (long) value. If it can be converted to an integer than a second lookup will be performed against entries with key ranges to see if there is a key range that includes the requested key.

Range lookups can be used for looking up an IP address where the reference data values are associated with ranges of IP addresses. In this use case, the IP address must first be converted into a numeric value using `numeric-ip()`, e.g:

```
stroom:lookup('IP_TO_LOCATION', numeric-ip($ipAddress))
```

Similarly the reference data must be stored with key ranges whose bounds were created using this function.

3.10.4.1.6.2 Nested Maps

The lookup function allows you to perform chained lookups using nested maps. For example you may have a reference data map called `USER_ID_TO_LOCATION` that maps user IDs to some location information for that user and a map called `USER_ID_TO_MANAGER` that maps user IDs to the user ID of their manager. If you wanted to decorate a user's event with the location of their manager you could use a nested map to achieve the lookup chain. To perform the lookup set the `map` argument to the list of maps in the lookup chain, separated by a `/`, e.g. `USER_ID_TO_MANAGER/USER_ID_TO_LOCATION`.

This will perform a lookup against the first map in the list using the requested key. If a value is found the value will be used as the key in a lookup against the next map. The value from each map lookup is used as the key in the next map all the way down the chain. The value from the last lookup is then returned as the result of the `lookup()` call. If no value is found at any point in the chain then that results in no value being returned from the function.

In order to use nested map lookups each intermediate map must contain simple string values. The last map in the chain can either contain string values or XML fragment values.

3.10.4.1.7 put() and get()

You can put values into a map using the `put()` function. These values can then be retrieved later using the `get()` function. Values are stored against a key name so that multiple values can be stored. These functions can be used for many purposes but are most commonly used to count a number of records that meet certain criteria.

An example of how to count records is shown below:

```

<!-- Get the current record count -->
<xsl:variable name="currentCount" select="number(s:get('count'))" />

<!-- Increment the record count -->
<xsl:variable name="count">
  <xsl:choose>
    <xsl:when test="$currentCount">
      <xsl:value-of select="$currentCount + 1" />
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="1" />
    </xsl:otherwise>
  </xsl:choose>
</xsl:variable>

<!-- Store the count for future retrieval -->
<xsl:value-of select="s:put('count', $count)" />

<!-- Output the new count -->
<data name="Count">
  <xsl:attribute name="Value" select="$count" />
</data>

```

3.10.4.1.8 parse-uri()

The `parse-uri()` function takes a Uniform Resource Identifier (URI) in string form and returns an XML node with a namespace of `uri` containing the URI's individual components of `authority`, `fragment`, `host`, `path`, `port`, `query`, `scheme`, `schemeSpecificPart` and `userInfo`. See either [RFC 2306: Uniform Resource Identifiers \(URI\): Generic Syntax](#) or Java's `java.net.URI` Class for details regarding the components.

The following xml

```

<!-- Display and parse the URI contained within the text of the rURI element -->
<xsl:variable name="u" select="s:parseUri(rURI)" />

<URI>
  <xsl:value-of select="rURI" />
</URI>
<URIDetail>
  <xsl:copy-of select="$v"/>
</URIDetail>

```

given the `rURI` text contains

```
http://foo:bar@w1.superman.com:8080/very/long/path.html?p1=v1&p2=v2#more-details
```

would provide

```

<URL>http://foo:bar@w1.superman.com:8080/very/long/path.html?p1=v1&p2=v2#more-details</URL>
<URIDetail>
  <authority xmlns="uri">foo:bar@w1.superman.com:8080</authority>
  <fragment xmlns="uri">more-details</fragment>
  <host xmlns="uri">w1.superman.com</host>
  <path xmlns="uri">/very/long/path.html</path>
  <port xmlns="uri">8080</port>
  <query xmlns="uri">p1=v1&p2=v2</query>
  <scheme xmlns="uri">http</scheme>
  <schemeSpecificPart xmlns="uri">//foo:bar@w1.superman.com:8080/very/long/path.html?p1=v1&p2=v2</schemeSpecificPart>
  <userInfo xmlns="uri">foo:bar</userInfo>
</URIDetail>

```

3.10.4.2 - XSLT Includes

Using an XSLT import to include XSLT from another translation.

You can use an XSLT import to include XSLT from another translation. E.g.

```
<xsl:import href="ApacheAccessCommon" />
```

This would include the XSLT from the ApacheAccessCommon translation.

3.11 - Properties

Configuration of Stroom's application properties.

This documentation applies to Stroom 7.0 or higher

Properties are the means of configuring the Stroom application and are typically maintained by the Stroom system administrator. The value of some properties are required in order for Stroom to function, e.g. database connection details, and thus need to be set prior to running Stroom. Some properties can be changed at runtime to alter the behaviour of Stroom.

3.11.1 Sources

Property values can be defined in the following locations.

3.11.1.1 System Default

The system defaults are hard-coded into the Stroom application code by the developers and can't be changed. These represent reasonable defaults, where applicable, but may need to be changed, e.g. to reflect the scale of the Stroom system or the specific environment.

The default property values can either be viewed in the Stroom user interface (*Tools -> Properties*) or in the file `config/config-defaults.yml` in the Stroom distribution.

3.11.1.2 Global Database Value

Global database values are property values stored in the database that are global across the whole cluster.

The global database value is defined as a record in the `config` table in the database. The database record will only exist if a database value has explicitly been set. The database value will apply to all nodes in the cluster, overriding the default value, unless a node also has a value set in its YAML configuration.

Database values can be set from the Stroom user interface using *Tools -> Properties*. Some properties are marked *Read Only* which means they cannot have a database value set for them. These properties can only be altered via the YAML configuration file on each node. Such properties are typically used to configure values required for Stroom to be able to boot, so it does not make sense for them to be configurable from the User Interface.

3.11.1.3 YAML Configuration file

Stroom is built on top of a framework called Drop Wizard. Drop Wizard uses a YAML configuration file on each node to configure the application. This is typically `config.yml` and is located in the `config` directory.

This file contains both the Drop Wizard configuration settings (settings for ports, paths and application logging) and the Stroom specific properties configuration. The file is in YAML format and the Stroom properties are located under the `appConfig` key. For details of the Drop Wizard configuration structure, see [here \(external link\)](#).

The file is split into three sections using these keys:

- `server` - Configuration of the web server, e.g. ports, paths, request logging.
- `logging` - Configuration of application logging
- `appConfig` - The stroom configuration properties

The following is an example of the YAML configuration file:

```

# Drop Wizard configuration section
server:
  # e.g. ports and paths
logging:
  # e.g. logging levels/appenders

# Stroom properties configuration section
appConfig:
  commonDbDetails:
    connection:
      jdbcDriverClassName: ${STROOM_JDBC_DRIVER_CLASS_NAME:-com.mysql.cj.jdbc.Driver}
      jdbcDriverUrl: ${STROOM_JDBC_DRIVER_URL:-jdbc:mysql://localhost:3307/stroom?useUnicode=yes&characterEncoding=UTF-8}
      jdbcDriverUsername: ${STROOM_JDBC_DRIVER_USERNAME:-stroomuser}
      jdbcDriverPassword: ${STROOM_JDBC_DRIVER_PASSWORD:-stroompassword1}
    contentPackImport:
      enabled: true
...

```

In the Stroom user interface properties are named with a dot notation key, e.g. `stroom.contentPackImport.enabled`. Each part of the dot notation property name represents a key in the YAML file, e.g. for this example, the location in the YAML would be:

```

appConfig:
  contentPackImport:
    enabled: true  # stroom.contentPackImport.enabled

```

The `stroom` part of the dot notation name is replaced with `appConfig`.

3.11.1.3.1 Variable Substitution

The YAML configuration file supports Bash style variable substitution in the form of:

```

${ENV_VAR_NAME:-value_if_not_set}

```

This allows values to be set either directly in the file or via an environment variable, e.g.

```

jdbcDriverClassName: ${STROOM_JDBC_DRIVER_CLASS_NAME:-com.mysql.cj.jdbc.Driver}

```

In the above example, if the `STROOM_JDBC_DRIVER_CLASS_NAME` environment variable is not set then the value `com.mysql.cj.jdbc.Driver` will be used instead.

3.11.1.3.2 Typed Values

YAML supports typed values rather than just strings, see <https://yaml.org/refcard.html>. YAML understands booleans, strings, integers, floating point numbers, as well as sequences/lists and maps. Some properties will be represented differently in the user interface to the YAML file. This is due to how values are stored in the database and how the current user interface works. This will likely be improved in future versions. For details of how different types are represented in the YAML and the UI, see [Data Types](#).

3.11.2 Source Precedence

The three sources (*Default*, *Database* & *YAML*) are listed in increasing priority, i.e *YAML* trumps *Database*, which trumps *Default*.

For example, in a two node cluster, this table shows the effective value of a property on each node. A `-` indicates the value has not been set in that source. `NULL` indicates that the value has been explicitly set to `NULL`.

Source	Node1	Node2
Default	red	red
Database	-	-

Source	Node1	Node2
YAML	-	blue
Effective	red	blue

Or where a Database value is set.

Source	Node1	Node2
Default	red	red
Database	green	green
YAML	-	blue
Effective	green	blue

Or where a YAML value is explicitly set to `NULL`.

Source	Node1	Node2
Default	red	red
Database	green	green
YAML	-	NULL
Effective	green	NULL

3.11.3 Data Types

Stroom property values can be set using a number of different data types. Database property values are currently set in the user interface using the string form of the value. For each of the data types defined below, there will be an example of how the data type is recorded in its string form.

Data Type	Example UI String Forms	Example YAML form
Boolean	true false	true false
String	This is a string	"This is a string"
Integer/Long	123	123
Float	1.23	1.23
Stroom Duration	P30D P1DT12H PT30S 30d 30s 30000	"P30D" "P1DT12H" "PT30S" "30d" "30s" "30000" See Stroom Duration Data Type .
List	#red#Green#Blue ,1,2,3	See List Data Type
Map	,=red=FF0000,Green=00FF00,Blue=0000FF	See Map Data Type
DocRef	,docRef(MyType,a56ff805-b214-4674-a7a7-a8fac288be60,My DocRef name)	See DocRef Data Type
Enum	HIGH LOW	"HIGH" "LOW"
Path	/some/path/to/a/file	"/some/path/to/a/file"

Data Type	Example UI String Forms	Example YAML form
ByteSize	32 , 512Kib	32 , 512Kib See Byte Size Data Type

3.11.3.1 Stroom Duration Data Type

The *Stroom Duration* data type is used to specify time durations, for example the time to live of a cache or the time to keep data before it is purged. *Stroom Duration* uses a number of string forms to support legacy property values.

3.11.3.1.1 ISO 8601 Durations

Stroom Duration can be expressed using [ISO 8601 \(external link\)](#) duration strings. It does NOT support the full *ISO 8601* format, only D , H , M and s . For details of how the string is parsed to a Stroom Duration, see [Duration \(external link\)](#)

The following are examples of *ISO 8601* duration strings:

- P30D - 30 days
- P1DT12H - 1 day 12 hours (36 hours)
- PT30S - 30 seconds
- PT0.5S - 500 milliseconds

3.11.3.1.2 Legacy Stroom Durations

This format was used in versions of Stroom older than v7 and is included to support legacy property values.

The following are examples of legacy duration strings:

- 30d - 30 days
- 12h - 12 hours
- 10m - 10 minutes
- 30s - 30 seconds
- 500 - 500 milliseconds

Combinations such as 1m30s are not supported.

3.11.3.2 List Data Type

This type supports ordered lists of items, where an item can be of any supported data type, e.g. a list of strings or list of integers.

The following is an example of how a property (statusValues) that is List of strings is represented in the YAML:

```
annotation:
  statusValues:
    - "New"
    - "Assigned"
    - "Closed"
```

This would be represented as a string in the User Interface as:

| New|Assigned|Closed .

See [Delimiters in String Conversion](#) for details of how the items are delimited in the string.

The following is an example of how a property (cpu) that is List of DocRefs is represented in the YAML:

```

statistics:
  internal:
    cpu:
      - type: "StatisticStore"
        uuid: "af08c4a7-ee7c-44e4-8f5e-e9c6be280434"
        name: "CPU"
      - type: "StroomStatsStore"
        uuid: "1edfd582-5e60-413a-b91c-151bd544da47"
        name: "CPU"

```

This would be represented as a string in the User Interface as:

```
|,docRef(StatisticStore,af08c4a7-ee7c-44e4-8f5e-e9c6be280434,CPU)|,docRef(StroomStatsStore,1edfd582-5e60-413a-b91c-151bd544da47,CPU)
```

See [Delimiters in String Conversion](#) for details of how the items are delimited in the string.

3.11.3.3 Map Data Type

This type supports a collection of key/value pairs where the key is unique within the collection. The type of the key must be string, but the type of the value can be any supported type.

The following is an example of how a property (mapProperty) that is a map of string => string would be represented in the YAML:

```

mapProperty:
  red: "FF0000"
  green: "00FF00"
  blue: "0000FF"

```

This would be represented as a string in the User Interface as:

```
,=red=FF0000,Green=00FF00,Blue=0000FF
```

The delimiter between pairs is defined first, then the delimiter for the key and value.

See [Delimiters in String Conversion](#) for details of how the items are delimited in the string.

3.11.3.4 DocRef Data Type

A DocRef (or Document Reference) is a type specific to Stroom that defines a reference to an instance of a Document within Stroom, e.g. an XLST, Pipeline, Dictionary, etc. A DocRef consists of three parts, the type, the [UUID \(external link\)](#) and the name of the Document.

The following is an example of how a property (aDocRefProperty) that is a DocRef would be represented in the YAML:

```

aDocRefProperty:
  type: "MyType"
  uuid: "a56ff805-b214-4674-a7a7-a8fac288be60"
  name: "My DocRef name"

```

This would be represented as a string in the User Interface as:

```
,docRef(MyType,a56ff805-b214-4674-a7a7-a8fac288be60,My DocRef name)
```

See [Delimiters in String Conversion](#) for details of how the items are delimited in the string.

3.11.3.5 Byte Size Data Type

The Byte Size data type is used to represent a quantity of bytes using the [IEC\(external link\)](#) standard. Quantities are represented as powers of 1024, i.e. a Kib (Kibibyte) means 1024 bytes.

Examples of Byte Size values in string form are (a YAML value would optionally be surrounded with double quotes):

- 32 , 32b , 32B , 32bytes - 32 bytes
- 32K , 32KB , 32KiB - 32 kibibytes

- 32M , 32MB , 32MiB - 32 mebibytes
- 32G , 32GB , 32GiB - 32 gibibytes
- 32T , 32TB , 32TiB - 32 tebibytes
- 32P , 32PB , 32PiB - 32 pebibytes

The *iB form is preferred as it is more explicit and avoids confusion with SI units.

3.11.3.6 Delimiters in String Conversion

The string conversion used for collection types like *List*, *Map* etc. relies on the string form defining the delimiter(s) to use for the collection. The delimiter(s) are added as the first n characters of the string form, e.g. |red|green|blue or |=red=FF0000|Green=00FF00|Blue=0000FF . It is possible to use a number of different delimiters to allow for delimiter characters appearing in the actual value, e.g. #some text#some text with a | in it The following are the delimiter characters that can be used.

| , : , ; , , ! , / , \ , # , @ , ~ , - , _ , = , + , ?

When Stroom records a property value to the database it may use a delimiter of its own choosing, ensuring that it picks a delimiter that is not used in the property value.

3.11.4 Restart Required

Some properties are marked as requiring a restart. There are two scopes for this:

3.11.4.1 Requires UI Refresh

If a property is marked in UI as requiring a UI refresh then this means that a change to the property requires that the Stroom nodes serving the UI are restarted for the new value to take effect.

3.11.4.2 Requires Restart

If a property is marked in UI as requiring a restart then this means that a change to the property requires that all Stroom nodes are restarted for the new value to take effect.

3.12 - Roles

TODO

Describe application level permissions and how users and groups behave

3.13 - Security

There are many aspects of security that should be considered when installing and running Stroom.

3.13.1 Shared Storage

For most large installations Stroom uses shared storage for its data store. This storage could be a CIFS, NFS or similar shared file system. It is recommended that access to this shared storage is protected so that only the application can access it. This could be achieved by placing the storage and application behind a firewall and by requiring appropriate authentication to the shared storage. It should be noted that NFS is unauthenticated so should be used with appropriate safeguards.

3.13.2 MySQL

3.13.2.1 Accounts

It is beyond the scope of this article to discuss this in detail but all MySQL accounts should be secured on initial install. Official guidance for doing this can be found [here \(external link\)](#).

3.13.2.2 Communication

Communication between MySQL and the application should be secured. This can be achieved in one of the following ways:

- Placing MySQL and the application behind a firewall
- Securing communication through the use of iptables
- Making MySQL and the application communicate over SSL (see [here \(external link\)](#) for instructions)

The above options are not mutually exclusive and may be combined to better secure communication.

3.13.3 Application

3.13.3.1 Node to node communication

In a multi node Stroom deployment each node communicates with the master node. This can be configured securely in one of several ways:

- Direct communication to Tomcat on port 8080 - Secured by being behind a firewall or using iptables
- Direct communication to Tomcat on port 8443 - Secured using SSL and certificates
- Removal of Tomcat connectors other than AJP and configuration of Apache to communicate on port 443 using SSL and certificates

3.13.3.2 Application to Stroom Proxy Communication

The application can be configured to share some information with Stroom Proxy so that Stroom Proxy can decide whether or not to accept data for certain feeds based on the existence of the feed or its reject/accept status. The amount of information shared between the application and the proxy is minimal but could be used to discover what feeds are present within the system. Securing this communication is harder as both the application and the proxy will not typically reside behind the same firewall. Despite this communication can still be performed over SSL thus protecting this potential attack vector.

3.13.3.3 Admin port

Stroom (v6 and above) and its associated family of stroom-* DropWizard based services all expose an admin port (8081 in the case of stroom). This port serves up various health check and monitoring pages as well as a number of restful services for initiating admin tasks. There is currently no authentication on this admin port so it is assumed that access to this port will be tightly controlled using a firewall, iptables or similar.

3.13.3.4 Servlets

There are several servlets in Stroom that are accessible by certain URLs. Considerations should be made about what URLs are made available via Apache and who can access them. The servlets, path and function are described below:

Servlet	Path	Function	Risk
---------	------	----------	------

Servlet	Path	Function	Risk
DataFeed	/datafeed or /datafeed/*	Used to receive data	Possible denial of service attack by posting too much data/noise
RemoteFeedService	/remoting/remotefeedservice.rpc	Used by proxy to ask application about feed status (described in previous section)	Possible to systematically discover which feeds are available. Communication with this service should be secured over SSL discussed above
DynamicCSSServlet	/stroom/dynamic.css	Serves dynamic CSS based on theme configuration	Low risk as no important data is made available by this servlet
DispatchService	/stroom/dispatch.rpc	Service for UI and server communication	All back-end services accessed by this umbrella service are secured appropriately by the application
ImportFileServlet	/stroom/importfile.rpc	Used during configuration upload	Users must be authenticated and have appropriate permissions to import configuration
ScriptServlet	/stroom/script	Serves user defined visualisation scripts to the UI	The visualisation script is considered to be part of the application just as the CSS so is not secured
ClusterCallService	/clustercall.rpc	Used for node to node communication as discussed above	Communication must be secured as discussed above
ExportConfig	/export/*	Servlet used to export configuration data	Servlet access must be restricted with Apache to prevent configuration data being made available to unauthenticated users
Status	/status	Shows the application status including volume usage	Needs to be secured so that only appropriate users can see the application status
Echo	/echo	Block GZIP data posted to the echo servlet is sent back uncompressed. This is a utility servlet for decompression of external data	URL should be secured or not made available
Debug	/debug	Servlet for echoing HTTP header arguments including certificate details	Should be secured in production environments
SessionList	/sessionList	Lists the logged in users	Needs to be secured so that only appropriate users can see who is logged in
SessionResourceStore	/resourcestore/*	Used to create, download and delete temporary files linked to a users session such as data for export	This is secured by using the users session and requiring authentication

3.13.4 HDFS, Kafka, HBase, Zookeeper

Stroom and stroom-stats can integrate with HDFS, Kafka, HBase and Zookeeper. It should be noted that communication with these external services is currently not secure. Until additional security measures (e.g. authentication) are put in place it is assumed that access to these services will be carefully controlled (using a firewall, iptables or similar) so that only stroom nodes can access the open ports.

3.13.5 Content

It may be possible for a user to write XSLT, Data Splitter or other content that may expose data that we do not wish to or to cause the application some harm. At present processing operations are not isolated processes and so it is easy to cripple processing performance with a badly written translation whether written accidentally or on purpose. To mitigate this risk it is recommended that users that are given permission to create XSLT, Data Splitter and Pipeline configurations are trusted to do so.

Visualisations can be completely customised with javascript. The javascript that is added is executed in a clients browser potentially opening up the possibility of XSS attacks, an attack on the application to access data that a user shouldn't be able to access, an attack to destroy data or simply failure/incorrect operation of the user interface. To mitigate this risk all user defined javascript is executed within a separate browser IFrame. In addition all javascript should be examined before being added to a production system unless the author is trusted. This may necessitate the creation of a separate development and testing environment for user content.

3.14 - Stroom Jobs

Managing background jobs.

There are various jobs that run in the background within Stroom. Among these are jobs that control pipeline processing, removing old files from the file system, checking the status of nodes and volumes etc. Each task executes at a different time depending on the purpose of the task. There are three ways that a task can be executed:

1. Scheduled jobs execute periodically according to a cron schedule. These include jobs such as cleaning the file system where Stroom only needs to perform this action once a day and can do so overnight.
2. Frequency controlled jobs are executed every X seconds, minutes, hours etc. Most of the jobs that execute with a given frequency are status checking jobs that perform a short lived action fairly frequently.
3. Distributed jobs are only applicable to stream processing with a pipeline. Distributed jobs are executed by a worker node as soon as a worker has available threads to execute a job and the task distributor has work available.

A list of task types and their execution method can be seen by opening *Monitoring/Jobs* from the main menu.

TODO: image

Expanding each task type allows you to configure how a task behaves on each node:

TODO: image

3.14.1 Account Maintenance

This job checks user accounts on the system and de-activates them under the following conditions:

- An unused account that has been inactive for longer than the age configured by `stroom.security.identity.passwordPolicy.neverUsedAccountDeactivationThreshold`.
- An account that has been inactive for longer than the age configured by `stroom.security.identity.passwordPolicy.unusedAccountDeactivationThreshold`.

3.14.2 Attribute Value Data Retention

Deletes Meta attribute values (additional and less valuable metadata) older than `stroom.data.meta.metaValue.deleteAge`.

3.14.3 Data Delete

Before data is physically removed from the database and file system it is marked as logically deleted by adding a flag to the metadata record in the database. Data can be logically deleted by a user from the UI or via a process such as data retention. Data is deleted logically as it is faster to do than a physical delete (important in the UI), and it also allows for data to be restored (undeleted) from the UI. This job performs the actual physical deletion of data that has been marked logically deleted for longer than the duration configured with `stroom.data.store.deletePurgeAge`. All data files associated with a metadata record are deleted from the file system before the metadata is physically removed from the database.

3.14.4 Data Processor

Processes data by finding data that matches processing filters on each pipeline. When enabled, each worker node asks the master node for data processing tasks. The master node creates tasks based on processing filters added to the `Processors` screen of each pipeline and supplies them to the requesting workers.

3.14.5 Feed Based Data Retention

This job uses the retention property of each feed to logically delete data from the associated feed that is older than the retention period. The recommended way of specifying data retention rules is via the data retention policy feature, but feed based retention still exists for backwards compatibility. Feed based data retention will be removed in a future release and should be considered deprecated.

3.14.6 File System Clean (deprecated)

This is the previous incarnation of the `Data Delete` job. This job scans the file system looking for files that are no longer associated with metadata in the database or where the metadata is marked as deleted and deletes the files if this is the case. The process is slow to run as it has to traverse all stored data files and examine each. However, this version of the data deletion process was created when metadata was deleted immediately, i.e. not marked for future physical deletion, so was the only way to perform this clean up activity at the time. This job will be removed in a future release. The `Data Delete` job should be used instead from now on.

3.14.7 File System Volume Status

Scans your data volumes to ensure they are available and determines how much free space they have. Records this status in the Volume Status table.

3.14.8 Index Searcher Cache Refresh

Refresh references to Lucene index searchers that have been cached for a while.

3.14.9 Index Shard Delete

How frequently index shards that have been logically deleted are physically deleted from the file system.

3.14.10 Index Shard Retention

How frequently index shards that are older than their retention period are logically deleted.

3.14.11 Index Volume Status

Scans your index volumes to ensure they are available and determines how much free space they have. Records this status in the Index Volume Status table.

3.14.12 Index Writer Cache Sweep

How frequently entries in the Index Shard Writer cache are evicted based on the time-to-live, time-to-idle and cache size settings.

3.14.13 Index Writer Flush

How frequently in-memory changes to the index shards are flushed to the file system and committed to the index.

3.14.14 Java Heap Histogram Statistics

How frequently heap histogram statistics will be captured. This can be useful for diagnosing issues or seeing where memory is being used. Each run will result in a JVM pause so care should be taken when running this on a production system.

3.14.15 Node Status

How frequently we try write stats about node status including JVM and memory usage.

3.14.16 Pipeline Destination Roll

How frequently rolling pipeline destinations, e.g. a Rolling File Appender are checked to see if they need to be rolled. This frequency should be at least as short as the most frequent rolling frequency.

3.14.17 Policy Based Data Retention

Run the policy based data retention rules over the data and logically delete and data that should no longer be retained.

3.14.18 Processor Task Queue Statistics

How frequently statistics about the state of the stream processing task queue are captured.

3.14.19 Processor Task Retention

How frequently failed and completed tasks are checked to see if they are older than the delete age threshold set by `stroom.processor.deleteAge`. Any that are older than this threshold are deleted.

3.14.20 Property Cache Reload

Stroom's configuration properties can each be configured globally in the database. This job controls the frequency that each node refreshes the values of its properties cache from the global database values. See also [Properties](#).

3.14.21 Proxy Aggregation

If you front Stroom with an Stroom proxy which is configured to 'store' rather than 'store/forward', then this task when run will pick up all files in the proxy repository dir, aggregate them by feed and bring them into Stroom. It uses the system property `stroom.proxyDir`.

3.14.22 Query History Clean

How frequently items in the query history are removed from the history if their age is older than `stroom.history.daysRetention` or if the number of items in the history exceeds `stroom.history.itemsRetention`.

3.14.23 Ref Data Off-heap Store Purge

Purges all data older than the purge age defined by property `stroom.pipeline.purgeAge`. See also [Reference Data](#).

3.14.24 Solr Index Optimise

How frequently Solr index segments are explicitly optimised by merging them into one.

3.14.25 Solr Index Retention

How frequently a process is run to delete items from the Solr indexes that don't meet the retention rule of that index.

3.14.26 SQL Stats Database Aggregation

This job controls the frequency that the database statistics aggregation process is run. This process takes the entries in `SQL_STAT_VAL_SRC` and merges them into the main statistics tables `SQL_STAT_KEY` and `SQL_STAT_KEY`. As this process is reliant on data flushed by the *SQL Stats In Memory Flush* job it is advisable to schedule it to run after that, leaving some time for the in-memory flush to finish.

3.14.27 SQL Stats In Memory Flush

SQL Statistics are initially held and aggregated in memory. This job controls the frequency that the in memory statistics are flushed from the in memory buffer to the staging table `SQL_STAT_VAL_SRC` in the database.

3.15 - Tools

Various additional tools to assist in administering Stroom and accessing its data.

3.15.1 - Command Line Tools

Command line actions for administering Stroom.

Version Information: Created with Stroom v7.0

Last Updated: 10 September 2020

Stroom has a number of tools that are available from the command line in addition to starting the main application.

The basic structure of the command for starting one of stroom's commands is:

```
java -jar /absolute/path/to/stroom-app.jar COMMAND
```

COMMAND can be a number of values depending on what you want to do. Each command value is described in its own section.

NOTE: These commands are very powerful and potentially dangerous in the wrong hands, e.g. they allow the changing of user's passwords. Access to these commands should be strictly limited. Also, each command will run in its own JVM so are not really intended to be run when Stroom is running on the node.

3.15.1.1 server

```
java -jar /absolute/path/to/stroom-app.jar server path/to/config.yml
```

This is the normal command for starting the Stroom application using the supplied YAML configuration file. The example above will start the application as a foreground process. Stroom would typically be started using the `start.sh` shell script, but the command above is listed for completeness.

When stroom starts it will check the database to see if any migration is required. If migration from an earlier version (including from an empty database) is required then this will happen as part of the application start process.

3.15.1.2 migrate

```
java -jar /absolute/path/to/stroom-app.jar migrate path/to/config.yml
```

There may be occasions where you want to migrate an old version but not start the application, e.g. during migration testing or to initiate the migration before starting up a cluster. This command will run the process that checks for any required migrations and then performs them. On completion of the process it exits. This runs as a foreground process.

3.15.1.3 create_account

```
java -jar /absolute/path/to/stroom-app.jar create_account --u USER --p PASSWORD [OPTIONS] path/to/config.yml
```

Where the named arguments are:

- `-u --user` - The username for the user.
- `-p --password` - The password for the user.
- `-e --email` - The email address of the user.
- `-f --firstName` - The first name of the user.
- `-s --lastName` - The last name of the user.
- `--noPasswordChange` - If set do not require a password change on first login.

- `--neverExpires` - If set, the account will never expire.

This command will create an account in the internal identity provider within Stroom. Stroom is able to use third party OpenID identity providers such as Google or AWS Cognito but by default will use its own. When configured to use its own (the default) it will auto create an admin account when starting up a fresh instance. There are times when you may wish to create this account manually which this command allows.

3.15.1.3.1 Authentication Accounts and Stroom Users

The user account used for authentication is distinct to the Stroom *user* entity that is used for authorisation within Stroom. If an external IDP is used then the mechanism for creating the authentication account will be specific to that IDP. If using the default internal Stroom IDP then an account must be created in order to authenticate, either from within the UI if you are already authenticated as a privileged user or using this command. In either case a Stroom user will need to exist with the same username as the authentication account.

The command will fail if the user already exists. This command should NOT be run if you are using a third party identity provider.

This command will also run any necessary database migrations to ensure it is working with the correct version of the database schema.

3.15.1.4 reset_password

```
java -jar /absolute/path/to/stroom-app.jar reset_password --u USER --p PASSWORD path/to/config.yml
```

Where the named arguments are:

- `-u --user` - The username for the user.
- `-p --password` - The password for the user.

This command is used for changing the password of an existing account in stroom's internal identity provider. It will also reset all locked/inactive/disabled statuses to ensure the account can be logged into. This command should NOT be run if you are using a third party identity provider. It will fail if the account does not exist.

This command will also run any necessary database migrations to ensure it is working with the correct version of the database schema.

3.15.1.5 manage_users

```
java -jar /absolute/path/to/stroom-app.jar manage_users [OPTIONS] path/to/config.yml
```

Where the named arguments are:

- `--createUser USER_NAME` - Creates a Stroom user with the supplied username.
- `--createGroup GROUP_NAME` - Creates a Stroom user group with the supplied group name.
- `--addToGroup USER_OR_GROUP_NAME TARGET_GROUP` - Adds a user/group to an existing group.
- `--removeFromGroup USER_OR_GROUP_NAME TARGET_GROUP` - Removes a user/group from an existing group.
- `--grantPermission USER_OR_GROUP_NAME PERMISSION_NAME` - Grants the named application permission to the user/group.
- `--revokePermission USER_OR_GROUP_NAME PERMISSION_NAME` - Revokes the named application permission from the user/group.
- `--listPermissions` - Lists all the valid permission names.

This command allows you to manage the account permissions within stroom regardless of whether the internal identity provider or a 3rd party one is used. A typical use case for this is when using a third party identity provider. In this instance Stroom has no way of auto creating an admin account when first started so the association between the account on the 3rd party IDP and the stroom user account needs to be made manually. To set up an admin account to enable you to login to stroom you can do:

This command is not intended for automation of user management tasks on a running Stroom instance that you can authenticate with. It is only intended for cases where you cannot authenticate with Stroom, i.e. when setting up a new Stroom with a 3rd party IDP or when scripting the creation of a test environment. If you want to automate actions that can be performed in the UI then you can make use of the REST API that is described at </stroom/noauth/swagger-ui>.

See the [note](#) above about the distinction between authentication accounts and stroom users.

```
java -jar /absolute/path/to/stroom-app.jar manage_users --createUser jbloggs --createGroup Administrators --addToGroup jbloggs A
```

Where *jbloggs* is the user name of the account on the 3rd party IDP.

This command will also run any necessary database migrations to ensure it is working with the correct version of the database schema.

The named arguments can be used as many times as you like so you can create multiple users/groups/grants/etc. Regardless of the order of the arguments, the changes are executed in the following order:

1. Create users
2. Create groups
3. Add users/groups to a group
4. Remove users/groups from a group
5. Grant permissions to users/groups
6. Revoke permissions from users/groups

3.15.2 - Stream Dump Tool

A tool for exporting stream data from Stroom.

Data within Stroom can be exported to a directory using the `StreamDumpTool`. The tool is contained within the core Stroom Java library and can be accessed via the command line, e.g.

```
java -cp "apache-tomcat-7.0.53/lib/*:lib/*:instance/webapps/stroom/WEB-INF/lib/*" stroom.util.StreamDumpTool outputDir=output
```

Note the classpath may need to be altered depending on your installation.

The above command will export all content from Stroom and output it to a directory called `output`. Data is exported to zip files in the same format as zip files in proxy repositories. The structure of the exported data is `${feed}/${pathId}/${id}` by default with a `.zip` extension.

To provide greater control over what is exported and how the following additional parameters can be used:

`feed` - Specify the name of the feed to export data for (all feeds by default).

`streamType` - The single stream type to export (all stream types by default).

`createPeriodFrom` - Exports data created after this time specified in ISO8601 UTC format, e.g. `2001-01-01T00:00:00.000Z` (exports from earliest data by default).

`createPeriodTo` - Exports data created before this time specified in ISO8601 UTC format, e.g. `2001-01-01T00:00:00.000Z` (exports up to latest data by default).

`outputDir` - The output directory to write data to (required).

`format` - The format of the output data directory and file structure (`${feed}/${pathId}/${id}` by default).

3.15.2.1 Format

The format parameter can include several replacement variables:

`feed` - The name of the feed for the exported data.

`streamType` - The data type of the exported data, e.g. `RAW_EVENTS`.

`streamId` - The id of the data being exported.

`pathId` - A incrementing numeric id that creates sub directories when required to ensure no directory ends up containing too many files.

`id` - A incrementing numeric id similar to `pathId` but without sub directories.

3.16 - Volumes

Stroom's logical storage volumes for storing event and index data.

TODO

Describe volumes

4 - How Tos

This is a series of *HOWTOs* that are designed to get one started with Stroom.

TODO

Add the resources dir to /assets and fix all the img links

The *HOWTOs* are broken down into different functional concepts or areas of Stroom.

NOTE: These *HOWTOs* will match the development of Stroom and as a result, various elements will be updated over time, including screen captures. In some instances, screen captures will contain timestamps and so you may note inconsistent date or time movements within a complete *HOWTO*, although if a sequence of captures is contained within a section of a document, they all will be replaced.

4.1 Installation

The [Installation Scenarios](#) *HOWTO* is provided to assist users in setting up a number of different Stroom deployments.

4.2 Event Feed Processing

The [Event Feed Processing](#) *HOWTO* is provided to assist users in setting up Stroom to process inbound event logs and transform them into the Stroom Event Logging XML Schema.

The [Apache HTTPD Event Feed](#) is interwoven into other *HOWTOs* that utilise this feed as a datasource.

4.3 Reference Feeds

Reference Feeds are used to provide look up data for a translation. The reference feed *HOWTOs* illustrate how to create reference feeds and how to use look up reference data maps to enrich the data you are processing.

4.4 Searches and Indexing

This section covers Indexing and Searching for data in Stroom

- [Search using bash](#)
- [Elasticsearch integration](#)
- [Solr integration](#)

4.5 General

[Raw Source Tracking](#) show how to associate a processed Event with the source line that generated it

Other topics in this section are

1. [Feed Management](#).
2. [Tasks](#)

4.1 - Installation

Various How Tos covering installation of Stroom and its dependencies

4.1.1 - Apache Httpd/Mod_JK configuration for Stroom

The following is a HOWTO to assist users in configuring Apache's HTTPD with Mod_JK for Stroom.

4.1.1.1 Assumptions

The following assumptions are used in this document.

- the user has reasonable RHEL/Centos System administration skills
- installations are on Centos 7.3 minimal systems (fully patched)
- the security of the HTTPD deployment should be reviewed for a production environment.

4.1.1.2 Installation of Apache httpd and Mod_JK Software

To deploy Stroom using Apache's httpd web service as a front end (https) and Apache's mod_jk as the interface between Apache and the Stroom tomcat applications, we also need

- apr
- apr-util
- apr-devel
- gcc
- httpd
- httpd-devel
- mod_ssl
- epel-release
- tomcat-native
- apache's mod_jk tomcat connector plugin

Most of the required software are packages available via standard repositories and hence we can simply execute

```
sudo yum -y install apr apr-util apr-devel gcc httpd httpd-devel mod_ssl epel-release
sudo yum -y install tomcat-native
```

The reason for the distinct `tomcat-native` installation is that this package is from the [EPEL \(external link\)](#) repository so it must be installed first.

For the Apache mod_jk Tomcat connector we need to acquire a recent [release \(external link\)](#) and install it. The following commands achieve this for the 1.2.42 release.

```
sudo bash
cd /tmp
V=1.2.42
wget https://www.apache.org/dist/tomcat/tomcat-connectors/jk/tomcat-connectors-${V}-src.tar.gz
tar xf tomcat-connectors-${V}-src.tar.gz
cd tomcat-connectors-*src/native
./configure --with-apxs=/bin/apxs
make && make install
cd /tmp
rm -rf tomcat-connectors-*src
```

Although you could remove the gcc compiler at this point, we leave it installed as one *should* continue to upgrade the Tomcat Connectors to later releases.

4.1.1.3 Configure Apache httpd

We need to configure Apache as the `root` user.

If the Apache httpd service is ‘fronting’ a Stroom user interface, we should create an index file (`/var/www/html/index.html`) on all nodes so browsing to the root of the node will present the Stroom UI. This is not needed if you are deploying a Forwarding or Standalone Stroom proxy.

4.1.1.3.1 Forwarding file for Stroom User Interface deployments

```
F=/var/www/html/index.html
printf '<html>\n' > ${F}
printf '<head>\n' >> ${F}
printf '  <meta http-equiv="Refresh" content="0; URL=stroom"/>\n' >> ${F}
printf '</head>\n' >> ${F}
printf '</html>\n' >> ${F}
chmod 644 ${F}
```

Remember, deploy this file on all nodes running the Stroom Application.

4.1.1.3.2 Httpd.conf Configuration

We modify `/etc/httpd/conf/httpd.conf` on all nodes, but backup the file first with

```
cp /etc/httpd/conf/httpd.conf /etc/httpd/conf/httpd.conf.ORIG
```

Irrespective of the Stroom scenario being deployed - Multi Node Stroom (Application and Proxy), single Standalone Stroom Proxy or single Forwarding Stroom Proxy, the configuration of the `/etc/httpd/conf/httpd.conf` file is the same.

We start by modify the configuration file by, add just before the `ServerRoot` directive the following directives which are designed to make the httpd service more secure.

```
# Stroom Change: Start - Apply generic security directives
ServerTokens Prod
ServerSignature Off
FileETag None
RewriteEngine On
RewriteCond %{THE_REQUEST} !HTTP/1.1$
RewriteRule .* - [F]
Header set X-XSS-Protection "1; mode=block"
# Stroom Change: End
```

That is,

```
...
# Do not add a slash at the end of the directory path. If you point
# ServerRoot at a non-local disk, be sure to specify a local disk on the
# Mutex directive, if file-based mutexes are used. If you wish to share the
# same ServerRoot for multiple httpd daemons, you will need to change at
# least PidFile.
#
ServerRoot "/etc/httpd"

#
# Listen: Allows you to bind Apache to specific IP addresses and/or
...
```

becomes

```

...
# Do not add a slash at the end of the directory path. If you point
# ServerRoot at a non-local disk, be sure to specify a local disk on the
# Mutex directive, if file-based mutexes are used. If you wish to share the
# same ServerRoot for multiple httpd daemons, you will need to change at
# least PidFile.
#
# Stroom Change: Start - Apply generic security directives
ServerTokens Prod
ServerSignature Off
FileETag None
RewriteEngine On
RewriteCond %{THE_REQUEST} !HTTP/1.1$
RewriteRule .* - [F]
Header set X-XSS-Protection "1; mode=block"
# Stroom Change: End
ServerRoot "/etc/httpd"

#
# Listen: Allows you to bind Apache to specific IP addresses and/or
...

```

We now block access to the /var/www directory by commenting out

```

<Directory "/var/www">
    AllowOverride None
    # Allow open access:
    Require all granted
</Directory>

```

that is

```

...
#
# Relax access to content within /var/www.
#
<Directory "/var/www">
    AllowOverride None
    # Allow open access:
    Require all granted
</Directory>

# Further relax access to the default document root:
...

```

becomes

```

...
#
# Relax access to content within /var/www.
#
# Stroom Change: Start - Block access to /var/www
# <Directory "/var/www">
#     AllowOverride None
#     # Allow open access:
#     Require all granted
# </Directory>
# Stroom Change: End

# Further relax access to the default document root:
...

```

then within the /var/www/html directory turn off Indexes FollowSymLinks by commenting out the line

```
Options Indexes FollowSymLinks
```

That is

```
...
# The Options directive is both complicated and important. Please see
# http://httpd.apache.org/docs/2.4/mod/core.html#options
# for more information.
#
Options Indexes FollowSymLinks

#
# AllowOverride controls what directives may be placed in .htaccess files.
# It can be "All", "None", or any combination of the keywords:
...
...
```

becomes

```
...
# The Options directive is both complicated and important. Please see
# http://httpd.apache.org/docs/2.4/mod/core.html#options
# for more information.
#
# Stroom Change: Start - turn off indexes and FollowSymLinks
# Options Indexes FollowSymLinks
# Stroom Change: End

#
# AllowOverride controls what directives may be placed in .htaccess files.
# It can be "All", "None", or any combination of the keywords:
...
...
```

Then finally we add two new log formats and configure the access log to use the new format. This is done within the `<IfModule logio_module>` by adding the two new LogFormat directives

```
LogFormat "%a/{REMOTE_PORT}e %X %t %l \"%u\" \"%r\" %s/%>s %D %I/%O/%B \"%{Referer}i\" \"%{User-Agent}i\" %V/%p" blackboxUser
LogFormat "%a/{REMOTE_PORT}e %X %t %l \"%{SSL_CLIENT_S_DN}x\" \"%r\" %s/%>s %D %I/%O/%B \"%{Referer}i\" \"%{User-Agent}i\" %V/%p" blackboxSSLUser
...
...
```

and replacing the CustomLog directive

```
CustomLog "logs/access_log" combined
...
...
```

with

```
CustomLog logs/access_log blackboxSSLUser
...
...
```

That is

```

...
LogFormat "%h %l %u %t \"%r\" %>s %b" common

<IfModule logio_module>
  # You need to enable mod_logio.c to use %I and %O
  LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %I %O" combinededio
</IfModule>

#
# The location and format of the access logfile (Common Logfile Format).
# If you do not define any access logfiles within a <VirtualHost>
# container, they will be logged here. Contrariwise, if you *do*
# define per-<VirtualHost> access logfiles, transactions will be
# logged therein and *not* in this file.
#
#CustomLog "logs/access_log" common

#
# If you prefer a logfile with access, agent, and referer information
# (Combined Logfile Format) you can use the following directive.
#
CustomLog "logs/access_log" combined
</IfModule>
...

```

becomes

```

...
LogFormat "%h %l %u %t \"%r\" %>s %b" common

<IfModule logio_module>
  # You need to enable mod_logio.c to use %I and %O
  LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %I %O" combinededio
  # Stroom Change: Start - Add new logformats
  LogFormat "%a/%{REMOTE_PORT}e %X %t %l \"%u\" \"%r\" %s/%>s %D %I/%0/%B \"%{Referer}i\" \"%{User-Agent}i\" %V/%p" blackboxUser
  LogFormat "%a/%{REMOTE_PORT}e %X %t %l \"%{SSL_CLIENT_S_DN}x\" \"%r\" %s/%>s %D %I/%0/%B \"%{Referer}i\" \"%{User-Agent}i\" %V/%p" blackboxSSL
  # Stroom Change: End
</IfModule>

# Stroom Change: Start - Add new logformats without the additional byte values
<IfModule !logio_module>
  LogFormat "%a/%{REMOTE_PORT}e %X %t %l \"%u\" \"%r\" %s/%>s %D 0/0/%B \"%{Referer}i\" \"%{User-Agent}i\" %V/%p" blackboxUser
  LogFormat "%a/%{REMOTE_PORT}e %X %t %l \"%{SSL_CLIENT_S_DN}x\" \"%r\" %s/%>s %D 0/0/%B \"%{Referer}i\" \"%{User-Agent}i\" %V/%p" blackboxSSL
</IfModule>
# Stroom Change: End

#
# The location and format of the access logfile (Common Logfile Format).
# If you do not define any access logfiles within a <VirtualHost>
# container, they will be logged here. Contrariwise, if you *do*
# define per-<VirtualHost> access logfiles, transactions will be
# logged therein and *not* in this file.
#
#CustomLog "logs/access_log" common

#
# If you prefer a logfile with access, agent, and referer information
# (Combined Logfile Format) you can use the following directive.
#
# Stroom Change: Start - Make the access log use a new format
# CustomLog "logs/access_log" combined
CustomLog logs/access_log blackboxSSLUser
# Stroom Change: End
</IfModule>
...

```

Remember, deploy this file on all nodes.

4.1.1.3.3 Configuration of ssl.conf

We modify `/etc/httpd/conf.d/ssl.conf` on all nodes, backing up first,

```
cp /etc/httpd/conf.d/ssl.conf /etc/httpd/conf.d/ssl.conf.ORIG
```

The configuration of the `/etc/httpd/conf.d/ssl.conf` **does** change depending on the Stroom scenario deployed. In the following we will indicate differences by tagged sub-headings. If the configuration applies irrespective of scenario, then **All scenarios** is the tag, else the tag indicated the type of Stroom deployment.

4.1.1.3.3.1 ssl.conf: HTTP to HTTPS Redirection - All scenarios

Before the context we add http to https redirection by adding the directives (noting we specify the actual server name)

```
<VirtualHost *:80>
    ServerName stroomp00.strmdev00.org
    Redirect permanent "/" "https://stroomp00.strmdev00.org/"
</VirtualHost>
```

That is, we change

```
...
## SSL Virtual Host Context
##
<VirtualHost _default_:443>
...
```

to

```
...
## SSL Virtual Host Context
##
# Stroom Change: Start - Add http redirection to https
<VirtualHost *:80>
    ServerName stroomp00.strmdev00.org
    Redirect permanent "/" "https://stroomp00.strmdev00.org/"
</VirtualHost>
# Stroom Change: End

<VirtualHost _default_:443>
```

4.1.1.3.3.2 ssl.conf: VirtualHost directives - Multi Node 'Application and Proxy' deployment

Within the context we set the directives, in this case, we use the CNAME `stroomp.strmdev00.org`

```
ServerName stroomp.strmdev00.org
JkMount /stroom* loadbalancer
JkMount /stroom/remoting/cluster* local
JkMount /stroom/datafeed* loadbalancer_proxy
JkMount /stroom/remoting* loadbalancer_proxy
JkMount /stroom/datafeed/direct* loadbalancer
JkOptions +ForwardKeySize +ForwardURICompat +ForwardSSLCertChain -ForwardDirectories
```

That is, we change

```
...
<VirtualHost _default_:443>

# General setup for the virtual host, inherited from global configuration
#DocumentRoot "/var/www/html"
#ServerName www.example.com:443

# Use separate log files for the SSL virtual host; note that LogLevel
# is not inherited from httpd.conf.
...
```

to

```

...
<VirtualHost _default_:443>

# General setup for the virtual host, inherited from global configuration
#DocumentRoot "/var/www/html"
#ServerName www.example.com:443
# Stroom Change: Start - Set servername and mod_jk connectivity
ServerName stroomp.strmdev00.org
JkMount /stroom* loadbalancer
JkMount /stroom/remoting/cluster* local
JkMount /stroom/datafeed* loadbalancer_proxy
JkMount /stroom/remoting* loadbalancer_proxy
JkMount /stroom/datafeed/direct* loadbalancer
JkOptions +ForwardKeySize +ForwardURICompat +ForwardSSLCertChain -ForwardDirectories
# Stroom Change: End

# Use separate log files for the SSL virtual host; note that LogLevel
# is not inherited from httpd.conf.
...

```

4.1.1.3.3.3 ssl.conf: VirtualHost directives - Standalone or Forwarding Proxy deployment

Within the context set the directives, in this case, for a node named say `stroomp0.strmdev00.org`

```

ServerName stroomp0.strmdev00.org
JkMount /stroom/datafeed* local_proxy
JkOptions +ForwardKeySize +ForwardURICompat +ForwardSSLCertChain -ForwardDirectories

```

That is, we change

```

...
<VirtualHost _default_:443>

# General setup for the virtual host, inherited from global configuration
#DocumentRoot "/var/www/html"
#ServerName www.example.com:443

# Use separate log files for the SSL virtual host; note that LogLevel
# is not inherited from httpd.conf.
...

```

to

```

...
<VirtualHost _default_:443>

# General setup for the virtual host, inherited from global configuration
#DocumentRoot "/var/www/html"
#ServerName www.example.com:443
# Stroom Change: Start - Set servername and mod_jk connectivity
ServerName stroomp0.strmdev00.org
JkMount /stroom/datafeed* local_proxy
JkOptions +ForwardKeySize +ForwardURICompat +ForwardSSLCertChain -ForwardDirectories
# Stroom Change: End

# Use separate log files for the SSL virtual host; note that LogLevel
# is not inherited from httpd.conf.
...

```

4.1.1.3.3.4 ssl.conf: VirtualHost directives - Single Node 'Application and Proxy' deployment

Within the context set the directives, in this case, for a node name `stroomp0.strmdev00.org`

```

ServerName stroomp00.strmdev00.org
JkMount /stroom* local
JkMount /stroom/remoting/cluster* local
JkMount /stroom/datafeed* local_proxy
JkMount /stroom/remoting* local_proxy
JkMount /stroom/datafeed/direct* local
JkOptions +ForwardKeySize +ForwardURICompat +ForwardSSLCertChain -ForwardDirectories

```

That is, we change

```

...
<VirtualHost _default_:443>

# General setup for the virtual host, inherited from global configuration
#DocumentRoot "/var/www/html"
#ServerName www.example.com:443

# Use separate log files for the SSL virtual host; note that LogLevel
# is not inherited from httpd.conf.
...

```

to

```

...
<VirtualHost _default_:443>

# General setup for the virtual host, inherited from global configuration
#DocumentRoot "/var/www/html"
#ServerName www.example.com:443
# Stroom Change: Start - Set servername and mod_jk connectivity
ServerName stroomp00.strmdev00.org
JkMount /stroom* local
JkMount /stroom/remoting/cluster* local
JkMount /stroom/datafeed* local_proxy
JkMount /stroom/remoting* local_proxy
JkMount /stroom/datafeed/direct* local
JkOptions +ForwardKeySize +ForwardURICompat +ForwardSSLCertChain -ForwardDirectories
# Stroom Change: End

# Use separate log files for the SSL virtual host; note that LogLevel
# is not inherited from httpd.conf.
...

```

4.1.1.3.3.5 ssl.conf: Certificate file changes - All scenarios

We replace the standard certificate files with the generated certificates. In the example below, we are using the multi node scenario, in that the key file names are `stroomp.crt` and `stroomp.key`. For other scenarios, use the appropriate file names generated. We replace

```
SSLCertificateFile /etc/pki/tls/certs/localhost.crt
```

with

```
SSLCertificateFile /home/stroomuser/stroom-jks/public/stroomp.crt
```

and

```
SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
```

with

```
SSLCertificateKeyFile /home/stroomuser/stroom-jks/private/stroomp.key
```

That is, change

```

...
# pass phrase. Note that a kill -HUP will prompt again. A new
# certificate can be generated using the genkey(1) command.
SSLCertificateFile /etc/pki/tls/certs/localhost.crt

# Server Private Key:
# If the key is not combined with the certificate, use this
# directive to point at the key file. Keep in mind that if
# you've both a RSA and a DSA private key you can configure
# both in parallel (to also allow the use of DSA ciphers, etc.)
SSLCertificateKeyFile /etc/pki/tls/private/localhost.key

# Server Certificate Chain:
# Point SSLCertificateChainFile at a file containing the
...

```

to

```

...
# pass phrase. Note that a kill -HUP will prompt again. A new
# certificate can be generated using the genkey(1) command.
# Stroom Change: Start - Replace with Stroom server certificate
# SSLCertificateFile /etc/pki/tls/certs/localhost.crt
SSLCertificateFile /home/stroomuser/stroom-jks/public/stroomp.crt
# Stroom Change: End

# Server Private Key:
# If the key is not combined with the certificate, use this
# directive to point at the key file. Keep in mind that if
# you've both a RSA and a DSA private key you can configure
# both in parallel (to also allow the use of DSA ciphers, etc.)
# Stroom Change: Start - Replace with Stroom server private key file
# SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
SSLCertificateKeyFile /home/stroomuser/stroom-jks/private/stroomp.key
# Stroom Change: End

# Server Certificate Chain:
# Point SSLCertificateChainFile at a file containing the
...

```

4.1.1.3.3.6 ssl.conf: Certificate Bundle/NO-CA Verification - All scenarios

If you have signed your Stroom server certificate with a Certificate Authority, then change

```
SSLCACertificateFile /etc/pki/tls/certs/ca-bundle.crt
```

to be your own certificate bundle which you should probably store as `~stroomuser/stroom-jks/public/stroomp-ca-bundle.crt`.

Now if you are using a self signed certificate, you will need to set the Client Authentication to have a value of

```
SSLVerifyClient optional_no_ca
```

noting that this may change if you actually use a CA. That is, changing

```

...
# Client Authentication (Type):
# Client certificate verification type and depth. Types are
# none, optional, require and optional_no_ca. Depth is a
# number which specifies how deeply to verify the certificate
# issuer chain before deciding the certificate is not valid.
#SSLVerifyClient require
#SSLVerifyDepth 10

# Access Control:
# With SSLRequire you can do per-directory access control based
...

```

to

```

...
# Client Authentication (Type):
# Client certificate verification type and depth. Types are
# none, optional, require and optional_no_ca. Depth is a
# number which specifies how deeply to verify the certificate
# issuer chain before deciding the certificate is not valid.
#SSLVerifyClient require
#SSLVerifyDepth 10
# Stroom Change: Start - Set optional_no_ca (given we have a self signed certificate)
SSLVerifyClient optional_no_ca
# Stroom Change: End

# Access Control:
# With SSLRequire you can do per-directory access control based
...

```

4.1.1.3.3.7 ssl.conf: Servlet Protection - Single or Multi Node scenarios (not for Standalone/Forwarding Proxy)

We now need to secure certain Stroom Application servlets, to ensure they are only accessed under appropriate control.

This set of servlets will be accessible by all nodes in the subnet 192.168.2 (as well as localhost). We achieve this by adding after the example Location directives

```

<Location ~ "stroom/(status|echo|sessionList|debug)" >
    Require all denied
    Require ip 127.0.0.1 192.168.2
</Location>

```

We further restrict the clustercall and export servlets to just the localhost. If we had multiple Stroom processing nodes, you would specify each node, or preferably, the subnet they are on. In our multi node case this is 192.168.2.

```

<Location ~ "stroom/export/|stroom/remoting/clustercall.rpc" >
    Require all denied
    Require ip 127.0.0.1 192.168.2
</Location>

```

That is, the following

```

...
#           and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
#           and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20      ) \
#           or %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/
#</Location>

#   SSL Engine Options:
#   Set various options for the SSL engine.
#   o FakeBasicAuth:
...

```

changes to

```

...
#           and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
#           and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20      ) \
#           or %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/
```

#</Location>

Stroom Change: Start - Lock access to certain servlets

<Location ~ "stroom/(status|echo|sessionList|debug)" >

Require all denied

Require ip 127.0.0.1 192.168.2

</Location>

Lock these Servlets more securely - to just localhost and processing node(s)

<Location ~ "stroom/export/|stroom/remoting/clustercall.rpc" >

Require all denied

Require ip 127.0.0.1 192.168.2

</Location>

Stroom Change: End

SSL Engine Options:

Set various options for the SSL engine.

o FakeBasicAuth:

...

4.1.1.3.3.8 ssl.conf: Log formats - All scenarios

Finally, as we make use of the Black Box Apache log format, we replace the standard format

```
CustomLog logs/ssl_request_log \
    "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
```

with

```
CustomLog logs/ssl_request_log blackboxSSLUser
```

That is, change

```

...
# Per-Server Logging:
# The home of a custom SSL log file. Use this when you want a
# compact non-error SSL logfile on a virtual host basis.
CustomLog logs/ssl_request_log \
    "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"

</VirtualHost>
```

to

```

...
# Per-Server Logging:
# The home of a custom SSL log file. Use this when you want a
# compact non-error SSL logfile on a virtual host basis.
# Stroom Change: Start - Change ssl_request log to use our BlackBox format
# CustomLog logs/ssl_request_log \
#     "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
CustomLog logs/ssl_request_log blackboxSSLUser
# Stroom Change: End

</VirtualHost>
```

Remember, in the case of Multi node stroom Application servers, deploy this file on all servers.

4.1.1.3.4 Apache Mod_JK configuration

Apache Mod_JK has two configuration files

- /etc/httpd/conf.d/mod_jk.conf - for the http server configuration
- /etc/httpd/conf/workers.properties - to configure the Tomcat workers

In multi node scenarios, `/etc/httpd/conf.d/mod_jk.conf` is the same on all servers, but the `/etc/httpd/conf/workers.properties` file is different. The contents of these two configuration files differ depending on the Stroom deployment. The following provide the various deployment scenarios.

4.1.1.3.4.1 Mod_JK Multi Node Application and Proxy Deployment

For a Stroom Multi node Application and Proxy server,

- we configure `/etc/httpd/conf.d/mod_jk.conf` as per

```
F=/etc/httpd/conf.d/mod_jk.conf
printf 'LoadModule jk_module modules/mod_jk.so\n' > ${F}
printf 'JkWorkersFile conf/workers.properties\n' >> ${F}
printf 'JkLogFile logs/mod_jk.log\n' >> ${F}
printf 'JkLogLevel info\n' >> ${F}
printf 'JkLogStampFormat "[%{a} %{b} %{d} %{H}:{%M}:{%S} %{Y}]\n' >> ${F}
printf 'JkOptions +ForwardKeySize +ForwardURICompat +ForwardSSLCertChain -ForwardDirectories\n' >> ${F}
printf 'JkRequestLogFormat "%{w} %{V} %{T}\n' >> ${F}
printf 'JkMount /stroom* loadbalancer\n' >> ${F}
printf 'JkMount /stroom/remoting/cluster* local\n' >> ${F}
printf 'JkMount /stroom/datafeed* loadbalancer_proxy\n' >> ${F}
printf 'JkMount /stroom/remoting* loadbalancer_proxy\n' >> ${F}
printf 'JkMount /stroom/datafeed/direct* loadbalancer\n' >> ${F}
printf '# Note: Replaced JkShmFile logs/jk.shm due to SELinux issues. Refer to\n' >> ${F}
printf '# https://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=225452\n' >> ${F}
printf '# The following makes use of the existing /run/httpd directory\n' >> ${F}
printf 'JkShmFile run/jk.shm\n' >> ${F}
printf '<Location /jkstatus/>\n' >> ${F}
printf ' JkMount status\n' >> ${F}
printf ' Order deny,allow\n' >> ${F}
printf ' Deny from all\n' >> ${F}
printf ' Allow from 127.0.0.1\n' >> ${F}
printf '</Location>\n' >> ${F}
chmod 640 ${F}
```

- we configure `/etc/httpd/conf/workers.properties` as per

Since we are deploying for a cluster with load balancing, we need a `workers.properties` file per node. Executing the following will result in two files (`workers.properties.stroom00` and `workers.properties.stroom01`) which should be deployed to their respective servers.

```

cd /tmp
# Set the list of nodes
Nodes="stroomp00.strmdev00.org stroomp01.strmdev00.org"
for oN in ${Nodes}; do
    _n=`echo ${oN} | cut -f1 -d\.`
    (
        printf '# Workers.properties for Stroom Cluster member: %s %s\n' ${oN}
        printf 'worker.list=loadbalancer,loadbalancer_proxy,local,local_proxy,status\n'
        L_t=""
        Lp_t=""
        for FQDN in ${Nodes}; do
            N=`echo ${FQDN} | cut -f1 -d\.`
            printf 'worker.%s.port=8009\n' ${N}
            printf 'worker.%s.host=%s\n' ${N} ${FQDN}
            printf 'worker.%s.type=ajp13\n' ${N}
            printf 'worker.%s.lbfactor=1\n' ${N}
            printf 'worker.%s.max_packet_size=65536\n' ${N}
            printf 'worker.%s_proxy.port=9009\n' ${N}
            printf 'worker.%s_proxy.host=%s\n' ${N} ${FQDN}
            printf 'worker.%s_proxy.type=ajp13\n' ${N}
            printf 'worker.%s_proxy.lbfactor=1\n' ${N}
            printf 'worker.%s_proxy.max_packet_size=65536\n' ${N}
            L_t="${L_t}${N},"
            Lp_t="${Lp_t}${N}_proxy,"
        done
        L=`echo $L_t | sed -e 's/.$/\n'`
        Lp=`echo $Lp_t | sed -e 's/.$/\n'`
        printf 'worker.loadbalancer.type=lb\n'
        printf 'worker.loadbalancer.balance_workers=%s\n' $L
        printf 'worker.loadbalancer.sticky_session=1\n'
        printf 'worker.loadbalancer_proxy.type=lb\n'
        printf 'worker.loadbalancer_proxy.balance_workers=%s\n' $Lp
        printf 'worker.loadbalancer_proxy.sticky_session=1\n'
        printf 'worker.local.type=lb\n'
        printf 'worker.local.balance_workers=%s\n' ${_n}
        printf 'worker.local.sticky_session=1\n'
        printf 'worker.local_proxy.type=lb\n'
        printf 'worker.local_proxy.balance_workers=%s_proxy\n' ${_n}
        printf 'worker.local_proxy.sticky_session=1\n'
        printf 'worker.status.type=status\n'
    ) > workers.properties.${_n}
    chmod 640 workers.properties.${_n}
done

```

Now depending in the node you are on, copy the relevant workers.properties.nodename file to /etc/httpd/conf/workers.properties. The following command makes this simple.

```
cp workers.properties.`hostname -s` /etc/httpd/conf/workers.properties
```

If you were to add an additional node to a multi node cluster, say the node stroomp02.strmdev00.org , then you would re-run the above script with

```
Nodes="stroomp00.strmdev00.org stroomp01.strmdev00.org stroomp02.strmdev00.org"
```

then redeploy all three files to the respective servers. Also note, that for the newly created workers.properties files for the existing nodes to take effect you will need to restart the Apache Httpd service on both nodes.

Remember, in multi node cluster deployments, the following files are the same and hence can be created on one node, but copied to the others not forgetting to backup the other node's original files. That is, the files

- /var/www/html/index.html
- /etc/httpd/conf.d/mod_jk.conf

- /etc/httpd/conf/httpd.conf

are to be the same on all nodes. Only the /etc/httpd/conf.d/ssl.conf and /etc/httpd/conf/workers.properties files change.

4.1.1.3.4.2 Mod_JK Standalone or Forwarding Stroom Proxy Deployment

For a Stroom Standalone or Forwarding proxy,

- we configure /etc/httpd/conf.d/mod_jk.conf as per

```
F=/etc/httpd/conf.d/mod_jk.conf
printf 'LoadModule jk_module modules/mod_jk.so\n' > ${F}
printf 'JkWorkersFile conf/workers.properties\n' >> ${F}
printf 'JkLogFile logs/mod_jk.log\n' >> ${F}
printf 'JkLogLevel info\n' >> ${F}
printf 'JkLogStampFormat "[%a %b %d %H:%M:%S %Y]\n" >> ${F}
printf 'JkOptions +ForwardKeySize +ForwardURICcompat +ForwardSSLCertChain -ForwardDirectories\n' >> ${F}
printf 'JkRequestLogFormat "%w %V %T"\n' >> ${F}
printf 'JkMount /stroom/datafeed* local_proxy\n' >> ${F}
printf '# Note: Replaced JkShmFile logs/jk.shm due to SELinux issues. Refer to\n' >> ${F}
printf '# https://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=225452\n' >> ${F}
printf '# The following makes use of the existing /run/httpd directory\n' >> ${F}
printf 'JkShmFile run/jk.shm\n' >> ${F}
printf '<Location /jkstatus/>\n' >> ${F}
printf ' JkMount status\n' >> ${F}
printf ' Order deny,allow\n' >> ${F}
printf ' Deny from all\n' >> ${F}
printf ' Allow from 127.0.0.1\n' >> ${F}
printf '</Location>\n' >> ${F}
chmod 640 ${F}
```

- we configure /etc/httpd/conf/workers.properties as per

The variable **N** in the script below is to be the node name (not FQDN) of your sever (i.e. stroomfp0).

```
N=stroomfp0
FQDN=`hostname -f`
F=/etc/httpd/conf/workers.properties
printf 'worker.list=local_proxy,status\n' > ${F}
printf 'worker.%s_proxy.port=9009\n' ${N} >> ${F}
printf 'worker.%s_proxy.host=%s\n' ${N} ${FQDN} >> ${F}
printf 'worker.%s_proxy.type=ajp13\n' ${N} >> ${F}
printf 'worker.%s_proxy.lbfactor=1\n' ${N} >> ${F}
printf 'worker.local_proxy.type=lb\n' >> ${F}
printf 'worker.local_proxy.balance_workers=%s_proxy\n' ${N} >> ${F}
printf 'worker.local_proxy.sticky_session=1\n' >> ${F}
printf 'worker.status.type=status\n' >> ${F}
chmod 640 ${F}
```

4.1.1.3.4.3 Mod_JK Single Node Application and Proxy Deployment

For a Stroom Single node Application and Proxy server,

- we configure /etc/httpd/conf.d/mod_jk.conf as per

```

F=/etc/httpd/conf.d/mod_jk.conf
printf 'LoadModule jk_module modules/mod_jk.so\n' > ${F}
printf 'JkWorkersFile conf/workers.properties\n' >> ${F}
printf 'JkLogFile logs/mod_jk.log\n' >> ${F}
printf 'JkLogLevel info\n' >> ${F}
printf 'JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"\n' >> ${F}
printf 'JkOptions +ForwardKeySize +ForwardURICompat +ForwardSSLCertChain -ForwardDirectories\n' >> ${F}
printf 'JkRequestLogFormat "%w %V %T"\n' >> ${F}
printf 'JkMount /stroom* local\n' >> ${F}
printf 'JkMount /stroom/remoting/cluster* local\n' >> ${F}
printf 'JkMount /stroom/datafeed* local_proxy\n' >> ${F}
printf 'JkMount /stroom/remoting* local_proxy\n' >> ${F}
printf 'JkMount /stroom/datafeed/direct* local\n' >> ${F}
printf '# Note: Replaced JkShmFile logs/jk.shm due to SELinux issues. Refer to\n' >> ${F}
printf '# https://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=225452\n' >> ${F}
printf '# The following makes use of the existing /run/httpd directory\n' >> ${F}
printf 'JkShmFile run/jk.shm\n' >> ${F}
printf '<Location /jkstatus/>\n' >> ${F}
printf '  JkMount status\n' >> ${F}
printf '  Order deny,allow\n' >> ${F}
printf '  Deny from all\n' >> ${F}
printf '  Allow from 127.0.0.1\n' >> ${F}
printf '</Location>\n' >> ${F}
chmod 640 ${F}

```

- we configure /etc/httpd/conf/workers.properties as per

The variable **N** in the script below is to be the node name (not FQDN) of your sever (i.e. stroomp00).

```

N=stroomp00
FQDN=`hostname -f`
F=/etc/httpd/conf/workers.properties
printf 'worker.list=local,local_proxy,status\n' > ${F}
printf 'worker.%s.port=8009\n' ${N} >> ${F}
printf 'worker.%s.host=%s\n' ${N} ${FQDN} >> ${F}
printf 'worker.%s.type=ajp13\n' ${N} >> ${F}
printf 'worker.%s.lbfactor=1\n' ${N} >> ${F}
printf 'worker.%s.max_packet_size=65536\n' ${N} >> ${F}
printf 'worker.%s_proxy.port=9009\n' ${N} >> ${F}
printf 'worker.%s_proxy.host=%s\n' ${N} ${FQDN} >> ${F}
printf 'worker.%s_proxy.type=ajp13\n' ${N} >> ${F}
printf 'worker.%s_proxy.lbfactor=1\n' ${N} >> ${F}
printf 'worker.%s_proxy.max_packet_size=65536\n' ${N} >> ${F}
printf 'worker.local.type=lb\n' >> ${F}
printf 'worker.local.balance_workers=%s\n' ${N} >> ${F}
printf 'worker.local.sticky_session=1\n' >> ${F}
printf 'worker.local_proxy.type=lb\n' >> ${F}
printf 'worker.local_proxy.balance_workers=%s_proxy\n' ${N} >> ${F}
printf 'worker.local_proxy.sticky_session=1\n' >> ${F}
printf 'worker.status.type=status\n' >> ${F}
chmod 640 ${F}

```

4.1.1.4 Final host configuration and web service enablement

Now tidy up the SELinux context for access on all nodes and files via the commands

```

setsebool -P httpd_enable_homedirs on
setsebool -P httpd_can_network_connect on
chcon --reference /etc/httpd/conf.d/README /etc/httpd/conf.d/mod_jk.conf
chcon --reference /etc/httpd/conf/magic /etc/httpd/conf/workers.properties

```

We also enable both http and https services via the firewall on all nodes. If you don't want to present a http access point, then don't enable it in the firewall setting. This is done with

```
firewall-cmd --zone=public --add-service=http --permanent  
firewall-cmd --zone=public --add-service=https --permanent  
firewall-cmd --reload  
firewall-cmd --zone=public --list-all
```

Finally enable then start the httpd service, correcting any errors. It should be noted that on any errors, the suggestion of a systemctl status or viewing the journal are good, but also review information in the httpd error logs found in `/var/log/httpd/`.

```
systemctl enable httpd.service  
systemctl start httpd.service
```

4.1.2 - Database Installation

This HOWTO describes the installation of the Stroom databases.

Following this HOWTO will produce a simple, minimally secured database deployment. In a production environment consideration needs to be made for redundancy, better security, data-store location, increased memory usage, and the like.

Stroom has two databases. The first, `stroom`, is used for management of Stroom itself and the second, `statistics` is used for the Stroom Statistics capability. There are many ways to deploy these two databases. One could

- have a single database instance and serve both databases from it
- have two database instances on the same server and serve one database per instance
- have two separate nodes, each with its own database instance
- the list goes on.

In this HOWTO, we describe the deployment of two database instances on the one node, each serving a single database. We provide example deployments using either the [MariaDB \(external link\)](#) or [MySQL Community \(external link\)](#) versions of MySQL.

4.1.2.1 Assumptions

- we are installing the MariaDB or MySQL Community RDBMS software.
- the primary database node is 'stroomdb0.stromdev00.org'.
- installation is on a fully patched minimal Centos 7.3 instance.
- we are installing BOTH databases (`stroom` and `statistics`) on the same node - 'stroomdb0.stromdev00.org' but with two distinct database engines. The first database will communicate on port `3307` and the second on `3308`.
- we are deploying with SELinux in enforcing mode.
- any scripts or commands that should run are in code blocks and are designed to allow the user to cut then paste the commands onto their systems.
- in this document, when a textual screen capture is documented, data entry is identified by the data surrounded by '`<>`'. This excludes enter/return presses.

4.1.2.2 Installation of Software

4.1.2.2.1 MariaDB Server Installation

As MariaDB is directly supported by Centos 7, we simply install the database server software and SELinux policy files, as per

```
sudo yum -y install policycoreutils-python mariadb-server
```

4.1.2.2.2 MySQL Community Server Installation

As MySQL is not directly supported by Centos 7, we need to install its repository files prior to installation. We get the current MySQL Community release repository rpm and validate its MD5 checksum against the published value found on the [MySQL Yum Repository \(external link\)](#) site.

```
wget https://repo.mysql.com/mysql57-community-release-el7.rpm  
md5sum mysql57-community-release-el7.rpm
```

On correct validation of the MD5 checksum, we install the repository files via

```
sudo yum -y localinstall mysql57-community-release-el7.rpm
```

NOTE: Stroom currently does not support the latest production MySQL version - 5.7. You will need to install MySQL Version 5.6.

Now since we must use MySQL Version 5.6 you will need to edit the MySQL repo file `/etc/yum.repos.d/mysql-community.repo` to disable the `mysql57-community` channel and enable the `mysql56-community` channel. We start by, backing up the repo file with

```
sudo cp /etc/yum.repos.d/mysql-community.repo /etc/yum.repos.d/mysql-community.repo.ORIG
```

Then edit the file to change

```
...
# Enable to use MySQL 5.6
[mysql56-community]
name=MySQL 5.6 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.6-community/el/7/$basearch/
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql

[mysql57-community]
name=MySQL 5.7 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.7-community/el/7/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
...
```

to become

```
...
# Enable to use MySQL 5.6
[mysql56-community]
name=MySQL 5.6 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.6-community/el/7/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql

[mysql57-community]
name=MySQL 5.7 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.7-community/el/7/$basearch/
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
...
```

Next we install server software and SELinux policy files, as per

```
sudo yum -y install policycoreutils-python mysql-community-server
```

4.1.2.3 Preparing the Database Deployment

4.1.2.3.1 MariaDB Variant

4.1.2.3.1.1 Create and instantiate both database instances

To set up two MariaDB database instances on the one node, we will use `mysql_multi` and `systemd` service templates. The `mysql_multi` utility is a capability that manages multiple MariaDB databases on the same node and `systemd` service templates manage multiple services from one configuration file. A `systemd` service template is unique in that it has an `@` character before the `.service` suffix.

To use this multiple-instance capability, we need to create two data directories for each database instance and also replace the main MariaDB configuration file, `/etc/my.cnf`, with one that includes configuration of key options for each instance. We will name our instances, `mysqld0` and `mysqld1`. We will also create specific log files for each instance.

We will use the directories, `/var/lib/mysql-mysqld0` and `/var/lib/mysql-mysqld1` for the data directories and `/var/log/mariadb/mysql-mysqld0.log` and `/var/log/mariadb/mysql-mysqld1.log` for the log files. Note you should modify `/etc/logrotate.d/mariadb` to manage these log files. Note also, we need to set the appropriate SELinux file contexts on the created directories and any files.

We create the data directories and log files and set their respective SELinux contexts via

```

sudo mkdir /var/lib/mysql-mysqld0
sudo chown mysql:mysql /var/lib/mysql-mysqld0
sudo semanage fcontext -a -t mysqld_db_t "/var/lib/mysql-mysqld0(/.*)?"
sudo restorecon -Rv /var/lib/mysql-mysqld0

sudo touch /var/log/mariadb/mysql-mysqld0.log
sudo chown mysql:mysql /var/log/mariadb/mysql-mysqld0.log
sudo chcon --reference=/var/log/mariadb/mariadb.log /var/log/mariadb/mysql-mysqld0.log

sudo mkdir /var/lib/mysql-mysqld1
sudo chown mysql:mysql /var/lib/mysql-mysqld1
sudo semanage fcontext -a -t mysqld_db_t "/var/lib/mysql-mysqld1(/.*)?"
sudo restorecon -Rv /var/lib/mysql-mysqld1

sudo touch /var/log/mariadb/mysql-mysqld1.log
sudo chown mysql:mysql /var/log/mariadb/mysql-mysqld1.log
sudo chcon --reference=/var/log/mariadb/mariadb.log /var/log/mariadb/mysql-mysqld1.log

```

We now initialise the our two database data directories via

```

sudo mysql_install_db --user=mysql --datadir=/var/lib/mysql-mysqld0
sudo mysql_install_db --user=mysql --datadir=/var/lib/mysql-mysqld1

```

We now replace the MySQL configuration file to set the options for each instance. Note that we will serve `mysqld0` and `mysqld1` via TCP ports 3307 and 3308 respectively. First backup the existing configuration file with

```

sudo cp /etc/my.cnf /etc/my.cnf.ORIG

```

then setup `/etc/my.cnf` as per

```

sudo bash
F=/etc/my.cnf
printf '[mysqld_multi]\n' > ${F}
printf 'mysqld = /usr/bin/mysqld_safe --basedir=/usr\n' >> ${F}
printf '\n' >> ${F}
printf '[mysqld0]\n' >> ${F}
printf 'port=3307\n' >> ${F}
printf 'mysqld = /usr/bin/mysqld_safe --basedir=/usr\n' >> ${F}
printf 'datadir=/var/lib/mysql-mysqld0/mysql.sock\n' >> ${F}
printf 'socket=/var/lib/mysql-mysqld0/mysql.sock\n' >> ${F}
printf 'pid-file=/var/run/mariadb/mysql-mysqld0.pid\n' >> ${F}
printf '\n' >> ${F}
printf 'log-error=/var/log/mariadb/mysql-mysqld0.log\n' >> ${F}
printf '\n' >> ${F}
printf '# Disabling symbolic-links is recommended to prevent assorted security\n' >> ${F}
printf '# risks\n' >> ${F}
printf 'symbolic-links=0\n' >> ${F}
printf '\n' >> ${F}
printf '[mysqld1]\n' >> ${F}
printf 'mysqld = /usr/bin/mysqld_safe --basedir=/usr\n' >> ${F}
printf 'port=3308\n' >> ${F}
printf 'datadir=/var/lib/mysql-mysqld1/\n' >> ${F}
printf 'socket=/var/lib/mysql-mysqld1/mysql.sock\n' >> ${F}
printf 'pid-file=/var/run/mariadb/mysql-mysqld1.pid\n' >> ${F}
printf '\n' >> ${F}
printf 'log-error=/var/log/mariadb/mysql-mysqld1.log\n' >> ${F}
printf '\n' >> ${F}
printf '# Disabling symbolic-links is recommended to prevent assorted security risks\n' >> ${F}
printf 'symbolic-links=0\n' >> ${F}
exit # To exit the root shell

```

We also need to associate the ports with the `mysqld_port_t` SELinux context as per

```
sudo semanage port -a -t mysqld_port_t -p tcp 3307
sudo semanage port -a -t mysqld_port_t -p tcp 3308
```

We next create the systemd service template as per

```
sudo bash
F=/etc/systemd/system/mysqld@.service

printf '# Install in /etc/systemd/system\n' > ${F}
printf '# Enable via systemctl enable mysqld@0 or systemctl enable mysqld@1\n' >> ${F}
printf '[Unit]\n' >> ${F}
printf 'Description=MySQL Multi Server for instance %i\n' >> ${F}
printf 'After=syslog.target\n' >> ${F}
printf 'After=network.target\n' >> ${F}
printf '\n' >> ${F}
printf '[Service]\n' >> ${F}
printf 'User=mysql\n' >> ${F}
printf 'Group=mysql\n' >> ${F}
printf 'Type=forking\n' >> ${F}
printf 'ExecStart=/usr/bin/mysqld_multi start %i\n' >> ${F}
printf 'ExecStop=/usr/bin/mysqld_multi stop %%i\n' >> ${F}
printf 'Restart=always\n' >> ${F}
printf 'PrivateTmp=true\n' >> ${F}
printf '\n' >> ${F}
printf '[Install]\n' >> ${F}
printf 'WantedBy=multi-user.target\n' >> ${F}
chmod 644 ${F}
exit; # to exit the root shell
```

We next enable and start both instances via

```
sudo systemctl enable mysqld@0
sudo systemctl enable mysqld@1
sudo systemctl start mysqld@0
sudo systemctl start mysqld@1
```

At this we should have both instances running. One should check each instance's log file for any errors.

4.1.2.3.1.2 Secure each database instance

We secure each database engine by running the `mysql_secure_installation` script. One should accept all defaults, which means the only entry (aside from pressing returns) is the administrator (root) database password. Make a note of the password you use. In this case we will use `Stroom5User@`. The utility `mysql_secure_installation` expects to find the Linux socket file to access the database it's securing at `/var/lib/mysql/mysql.sock`. Since we have used other locations, we temporarily link the real socket file to `/var/lib/mysql/mysql.sock` for each invocation of the utility. Thus we execute

```
sudo ln /var/lib/mysql-mysqld0/mysql.sock /var/lib/mysql/mysql.sock
sudo mysql_secure_installation
```

to see

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current password for the root user. If you've just installed MariaDB, and you haven't set the root password yet, the password will be blank, so you should just press enter here.

Enter current password for root (enter for none):

OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB root user without the proper authorisation.

Set root password? [Y/n]

New password: <__ Stroom5User@ __>

Re-enter new password: <__ Stroom5User@ __>

Password updated successfully!

Reloading privilege tables..

... Success!

By default, a MariaDB installation has an anonymous user, allowing anyone to log into MariaDB without having to have a user account created for them. This is intended only for testing, and to make the installation go a bit smoother. You should remove them before moving into a production environment.

Remove anonymous users? [Y/n]

... Success!

Normally, root should only be allowed to connect from 'localhost'. This ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n]

... Success!

By default, MariaDB comes with a database named 'test' that anyone can access. This is also intended only for testing, and should be removed before moving into a production environment.

Remove test database and access to it? [Y/n]

- Dropping test database...

... Success!

- Removing privileges on test database...

... Success!

Reloading the privilege tables will ensure that all changes made so far will take effect immediately.

Reload privilege tables now? [Y/n]

... Success!

Cleaning up...

All done! If you've completed all of the above steps, your MariaDB installation should now be secure.

Thanks for using MariaDB!

then we execute

```
sudo rm /var/lib/mysql/mysql.sock
sudo ln /var/lib/mysql-mysqld1/mysql.sock /var/lib/mysql/mysql.sock
sudo mysql_secure_installation
sudo rm /var/lib/mysql/mysql.sock
```

and process as before (for when running `mysql_secure_installation`). At this both database instances should be secure.

4.1.2.3.2 MySQL Community Variant

4.1.2.3.2.1 Create and instantiate both database instances

To set up two MySQL database instances on the one node, we will use `mysql_multi` and `systemd` service templates. The `mysql_multi` utility is a capability that manages multiple MySQL databases on the same node and `systemd` service templates manage multiple services from one configuration file. A `systemd` service template is unique in that it has an `@` character before the `.service` suffix.

To use this multiple-instance capability, we need to create two data directories for each database instance and also replace the main MySQL configuration file, `/etc/my.cnf`, with one that includes configuration of key options for each instance. We will name our instances, `mysqld0` and `mysqld1`. We will also create specific log files for each instance.

We will use the directories, `/var/lib/mysql-mysqld0` and `/var/lib/mysql-mysqld1` for the data directories and `/var/log/mysql-mysqld0.log` and `/var/log/mysql-mysqld1.log` for the log directories. Note you should modify `/etc/logrotate.d/mysql` to manage these log files. Note also, we need to set the appropriate SELinux file context on the created directories and files.

```
sudo mkdir /var/lib/mysql-mysqld0
sudo chown mysql:mysql /var/lib/mysql-mysqld0
sudo semanage fcontext -a -t mysqld_db_t "/var/lib/mysql-mysqld0(/.*)?"
sudo restorecon -Rv /var/lib/mysql-mysqld0

sudo touch /var/log/mysql-mysqld0.log
sudo chown mysql:mysql /var/log/mysql-mysqld0.log
sudo chcon --reference=/var/log/mysql.log /var/log/mysql-mysqld0.log

sudo mkdir /var/lib/mysql-mysqld1
sudo chown mysql:mysql /var/lib/mysql-mysqld1
sudo semanage fcontext -a -t mysqld_db_t "/var/lib/mysql-mysqld1(/.*)?"
sudo restorecon -Rv /var/lib/mysql-mysqld1

sudo touch /var/log/mysql-mysqld1.log
sudo chown mysql:mysql /var/log/mysql-mysqld1.log
sudo chcon --reference=/var/log/mysql.log /var/log/mysql-mysqld1.log
```

We now initialise the our two database data directories via

```
sudo mysql_install_db --user=mysql --datadir=/var/lib/mysql-mysqld0
sudo mysql_install_db --user=mysql --datadir=/var/lib/mysql-mysqld1
```

Disable the default database via

```
sudo systemctl disable mysqld
```

We now modify the MySQL configuration file to set the options for each instance. Note that we will serve `mysqld0` and `mysqld1` via TCP ports 3307 and 3308 respectively. First backup the existing configuration file with

```
sudo cp /etc/my.cnf /etc/my.cnf.ORIG
```

then setup `/etc/my.cnf` as per

```

sudo bash
F=/etc/my.cnf
printf '[mysqld_multi]\n' > ${F}
printf 'mysqld = /usr/bin/mysqld_safe --basedir=/usr\n' >> ${F}
printf '\n' >> ${F}
printf '[mysqld0]\n' >> ${F}
printf 'port=3307\n' >> ${F}
printf 'mysqld = /usr/bin/mysqld_safe --basedir=/usr\n' >> ${F}
printf 'datadir=/var/lib/mysql-mysqld0/\n' >> ${F}
printf 'socket=/var/lib/mysql-mysqld0/mysql.sock\n' >> ${F}
printf 'pid-file=/var/run/mysqld/mysql-mysqld0.pid\n' >> ${F}
printf '\n' >> ${F}
printf 'log-error=/var/log/mysql-mysqld0.log\n' >> ${F}
printf '\n' >> ${F}
printf '# Disabling symbolic-links is recommended to prevent assorted security\n' >> ${F}
printf '# risks\n' >> ${F}
printf 'symbolic-links=0\n' >> ${F}
printf '\n' >> ${F}
printf '[mysqld1]\n' >> ${F}
printf 'mysqld = /usr/bin/mysqld_safe --basedir=/usr\n' >> ${F}
printf 'port=3308\n' >> ${F}
printf 'datadir=/var/lib/mysql-mysqld1/\n' >> ${F}
printf 'socket=/var/lib/mysql-mysqld1/mysql.sock\n' >> ${F}
printf 'pid-file=/var/run/mysqld/mysql-mysqld1.pid\n' >> ${F}
printf '\n' >> ${F}
printf 'log-error=/var/log/mysql-mysqld1.log\n' >> ${F}
printf '\n' >> ${F}
printf '# Disabling symbolic-links is recommended to prevent assorted security risks\n' >> ${F}
printf 'symbolic-links=0\n' >> ${F}
exit # To exit the root shell

```

We also need to associate the ports with the `mysqld_port_t` SELinux context as per

```

sudo semanage port -a -t mysqld_port_t -p tcp 3307
sudo semanage port -a -t mysqld_port_t -p tcp 3308

```

We next create the systemd service template as per

```

sudo bash
F=/etc/systemd/system/mysqld@.service

printf '# Install in /etc/systemd/system\n' > ${F}
printf '# Enable via systemctl enable mysqld@0 or systemctl enable mysqld@1\n' >> ${F}
printf '[Unit]\n' >> ${F}
printf 'Description=MySQL Multi Server for instance %%i\n' >> ${F}
printf 'After=syslog.target\n' >> ${F}
printf 'After=network.target\n' >> ${F}
printf '\n' >> ${F}
printf '[Service]\n' >> ${F}
printf 'User=mysql\n' >> ${F}
printf 'Group=mysql\n' >> ${F}
printf 'Type=forking\n' >> ${F}
printf 'ExecStart=/usr/bin/mysqld_multi start %%i\n' >> ${F}
printf 'ExecStop=/usr/bin/mysqld_multi stop %%i\n' >> ${F}
printf 'Restart=always\n' >> ${F}
printf 'PrivateTmp=true\n' >> ${F}
printf '\n' >> ${F}
printf '[Install]\n' >> ${F}
printf 'WantedBy=multi-user.target\n' >> ${F}
chmod 644 ${F}
exit; # to exit the root shell

```

We next enable and start both instances via

```
sudo systemctl enable mysqld@0
sudo systemctl enable mysqld@1
sudo systemctl start mysqld@0
sudo systemctl start mysqld@1
```

At this we should have both instances running. One should check each instance's log file for any errors.

4.1.2.3.2.2 Secure each database instance

We secure each database engine by running the `mysql_secure_installation` script. One should accept all defaults, which means the only entry (aside from pressing returns) is the administrator (root) database password. Make a note of the password you use. In this case we will use `Stroom5User@`. The utility `mysql_secure_installation` expects to find the Linux socket file to access the database it's securing at `/var/lib/mysql/mysql.sock`. Since we have used other locations, we temporarily link the real socket file to `/var/lib/mysql/mysql.sock` for each invocation of the utility. Thus we execute

```
sudo ln /var/lib/mysql-mysqld0/mysql.sock /var/lib/mysql/mysql.sock
sudo mysql_secure_installation
```

to see

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MySQL
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MySQL to secure it, we'll need the current password for the root user. If you've just installed MySQL, and you haven't set the root password yet, the password will be blank, so you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MySQL root user without the proper authorisation.

Set root password? [Y/n] y
New password: <__ Stroom5User@ __>
Re-enter new password: <__ Stroom5User@ __>
Password updated successfully!
Reloading privilege tables..
... Success!

By default, a MySQL installation has an anonymous user, allowing anyone to log into MySQL without having to have a user account created for them. This is intended only for testing, and to make the installation go a bit smoother. You should remove them before moving into a production environment.

Remove anonymous users? [Y/n]
... Success!

Normally, root should only be allowed to connect from 'localhost'. This ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n]
... Success!

By default, MySQL comes with a database named 'test' that anyone can access. This is also intended only for testing, and should be removed before moving into a production environment.

Remove test database and access to it? [Y/n]
- Dropping test database...
ERROR 1008 (HY000) at line 1: Can't drop database 'test'; database doesn't exist
... Failed! Not critical, keep moving...
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far will take effect immediately.

Reload privilege tables now? [Y/n]
... Success!

All done! If you've completed all of the above steps, your MySQL installation should now be secure.

Thanks for using MySQL!

Cleaning up...

then we execute

```
sudo rm /var/lib/mysql/mysql.sock
sudo ln /var/lib/mysql-mysqld1/mysql.sock /var/lib/mysql/mysql.sock
sudo mysql_secure_installation
sudo rm /var/lib/mysql/mysql.sock
```

and process as before (for when running mysql_secure_installation). At this both database instances should be secure.

4.1.2.4 Create the Databases and Enable access by the Stroom processing users

We now create the `stroom` database within the first instance, `mysqld0` and the `statistics` database within the second instance `mysqld1`. It does not matter which database variant used as all commands are the same for both.

As well as creating the databases, we also need to establish the Stroom processing users that the Stroom processing nodes will use to access each database. For the `stroom` database, we will use the database user `stroomuser` with a password of `Stroompassword1@` and for the `statistics` database, we will use the database user `stroomstats` with a password of `Stroompassword2@`. One identifies a processing user as `<user>@<host>` on a `grant` SQL command.

In the `stroom` database instance, we will grant access for

- `stroomuser@localhost` for local access for maintenance etc.
- `stroomuser@stroomp00.strmdev00.org` for access by processing node `stroomp00.strmdev00.org`
- `stroomuser@stroomp01.strmdev00.org` for access by processing node `stroomp01.strmdev00.org`

and in the `statistics` database instance, we will grant access for

- `stroomstats@localhost` for local access for maintenance etc.
- `stroomstats@stroomp00.strmdev00.org` for access by processing node `stroomp00.strmdev00.org`
- `stroomstats@stroomp01.strmdev00.org` for access by processing node `stroomp01.strmdev00.org`

Thus for the `stroom` database we execute

```
mysql --user=root --port=3307 --socket=/var/lib/mysql-mysqld0/mysql.sock --password
```

and on entering the administrator's password, we arrive at the `MariaDB [(none)]>` or `mysql>` prompt. At this we create the database with

```
create database stroom;
```

and then to establish the users, we execute

```
grant all privileges on stroom.* to stroomuser@localhost identified by 'Stroompassword1@';
grant all privileges on stroom.* to stroomuser@stroomp00.strmdev00.org identified by 'Stroompassword1@';
grant all privileges on stroom.* to stroomuser@stroomp01.strmdev00.org identified by 'Stroompassword1@';
```

then

```
quit;
```

to exit.

And for the `statistics` database

```
mysql --user=root --port=3308 --socket=/var/lib/mysql-mysqld1/mysql.sock --password
```

with

```
create database statistics;
```

and then to establish the users, we execute

```
grant all privileges on statistics.* to stroomstats@localhost identified by 'Stroompassword2@';
grant all privileges on statistics.* to stroomstats@stroomp00.strmdev00.org identified by 'Stroompassword2@';
grant all privileges on statistics.* to stroomstats@stroomp01.strmdev00.org identified by 'Stroompassword2@';
```

then

```
quit;
```

to exit.

Clearly if we need to add more processing nodes, additional `grant` commands would be used. Further, if we were installing the databases in a single node Stroom environment, we would just have the first two pairs of grants .

4.1.2.5 Configure Firewall

Next we need to modify our firewall to allow remote access to our databases which listens on ports 3307 and 3308. The simplest way to achieve this is with the commands

```
sudo firewall-cmd --zone=public --add-port=3307/tcp --permanent
sudo firewall-cmd --zone=public --add-port=3308/tcp --permanent
sudo firewall-cmd --reload
sudo firewall-cmd --zone=public --list-all
```

Note

That this allows ANY node to connect to your databases. You should give consideration to restricting this to only allowing processing node access.

4.1.2.5.1 Debugging of Mariadb for Stroom

If there is a need to debug the Mariadb database and Stroom interaction, one can turn on auditing for the Mariadb service. To do so, log onto the relevant database as the administrative user as per

```
mysql --user=root --port=3307 --socket=/var/lib/mysql-mysqld0/mysql.sock --password
or
mysql --user=root --port=3308 --socket=/var/lib/mysql-mysqld1/mysql.sock --password
```

and at the MariaDB [(none)]> prompt enter

```
install plugin server_audit SONAME 'server_audit';
set global server_audit_file_path='/var/log/mariadb/mysqld-mysqld0_server_audit.log';
or
set global server_audit_file_path='/var/log/mariadb/mysqld-mysqld1_server_audit.log';
set global server_audit_logging=ON;
set global server_audit_file_rotate_size=10485760;
install plugin SQL_ERROR_LOG soname 'sql_errlog';
quit;
```

The above will generate two log files,

- `/var/log/mariadb/mysqld-mysqld0_server_audit.log` or `/var/log/mariadb/mysqld-mysqld1_server_audit.log` which records all commands the respective databases run. We have configured the log file will rotate at 10MB in size.

- `/var/lib/mysql-mysqld0/sql_errors.log` or `/var/lib/mysql-mysqld1/sql_errors.log` which records all erroneous SQL commands. This log file will rotate at 10MB in size. Note we cannot set this filename via the UI, but it will be appear in the data directory.

All files will, by default, generate up to 9 rotated files.

If you wish to rotate a log file manually, log into the database as the administrative user and execute either

- `set global server_audit_file_rotate_now=1;` to rotate the audit log file
- `set global sql_error_log_rotate=1;` to rotate the `sql_errlog` log file

4.1.2.5.2 Initial Database Access

It should be noted that if you monitor the `sql_errors.log` log file on a new Stoom deployment, when the Stoom Application first starts, it's initial access to the `stroom` database will result in the following attempted sql statements.

```
2017-04-16 16:24:50 stroomuser[stroomuser] @ stroomp00.strmdev00.org [192.168.2.126] ERROR 1146: Table 'stroom.schema_version' c
2017-04-16 16:24:50 stroomuser[stroomuser] @ stroomp00.strmdev00.org [192.168.2.126] ERROR 1146: Table 'stroom.STROOM_VER' does
2017-04-16 16:24:50 stroomuser[stroomuser] @ stroomp00.strmdev00.org [192.168.2.126] ERROR 1146: Table 'stroom.FD' doesn't exist
2017-04-16 16:24:50 stroomuser[stroomuser] @ stroomp00.strmdev00.org [192.168.2.126] ERROR 1146: Table 'stroom.FEED' doesn't exi
```

After this access the application will realise the database does not exist and it will initialise the database.

In the case of the `statistics` database you may note the following attempted access

```
2017-04-16 16:25:09 stroomstats[stroomstats] @ stroomp00.strmdev00.org [192.168.2.126] ERROR 1146: Table 'statistics.schema_ver
```

Again, at this point the application will initialise this database.

4.1.3 - Installation

This HOWTO is provided to assist users in setting up a number of different Stroom environments based on Centos 7.3 infrastructure.

4.1.3.1 Assumptions

The following assumptions are used in this document.

- the user has reasonable RHEL/Centos System administration skills.
- installations are on Centos 7.3 minimal systems (fully patched).
- the term 'node' is used to reference the 'host' a service is running on.
- the Stroom Proxy and Application software runs as user 'stroomuser' and will be deployed in this user's home directory
- data will reside in a directory tree referenced via '/stroomdata'. It is up to the user to provision a filesystem here, noting sub-directories of it will be NFS shared in Multi Node Stroom Deployments
- any scripts or commands that should run are in code blocks and are designed to allow the user to cut then paste the commands onto their systems
- in this document, when a textual screen capture is documented, data entry is identified by the data surrounded by '<>'. This excludes enter/return presses.
- better security of password choices, networking, firewalls, data stores, etc. can and should be achieved in various ways, but these HOWTOs are just a quick means of getting a working system, so only limited security is applied
- better configuration of the database (e.g. more memory, redundancy) should be considered in production environments
- the use of self signed certificates is appropriate for test systems, but users should consider appropriate CA infrastructure in production environments
- the user has access to a [Chrome \(external link\)](#) web browser as Stroom is optimised for this browser.

4.1.3.2 Introduction

This HOWTO provides guidance on a variety of simple Stroom deployments.

- [**Multi Node Stroom Cluster \(Proxy and Application\)**](#)

for an environment where multiple nodes are required to handle the processing load.

- [**Forwarding Stroom Proxy**](#)

for extensive networks where one wants to aggregate data through a proxy before sending data to the central Stroom processing systems.

- [**Standalone Stroom Proxy**](#)

for disconnected networks where collected data can be manually transferred to a Stroom processing service.

- [**Addition of a Node to Stroom Cluster**](#)

for when one needs to add an additional node to an existing cluster.

4.1.3.3 Nodename Nomenclature

For simplicity sake, the nodenames used in this HOWTO are geared towards the Multi Node Stroom Cluster deployment. That is,

- the database nodename is `stroomdb0.strmdev00.org`
- the processing nodenames are `stroomp00.strmdev00.org`, `stroomp01.strmdev00.org`, and `stroomp02.strmdev00.org`
- the first node in our cluster, `stroomp00.strmdev00.org`, also has the CNAME `stroomp.strmdev00.org`

In the case of the Proxy only deployments,

- the forwarding Stroom proxy nodename is `stoomfp0.strmdev00.org`
- the standalone nodename will be `stroomp00.strmdev00.org`

4.1.3.4 Storage

Both the Stroom Proxy and Application store data. The typical requirement is

- directory for Stroom proxy to store **inbound data** files
- directory for Stroom application **permanent data** files (events, etc.)
- directory for Stroom application **index data** files
- directory for Stroom application **working files** (temporary files, output, etc.)

Where multiple processing nodes are involved, the application's **permanent data** directories need to be accessible by all participating nodes.

Thus a hierarchy for a Stroom Proxy might by

- /stroomdata/stroom-proxy

and for an Application node

- /stroomdata/stroom-data
- /stroomdata/stroom-index
- /stroomdata/stroom-working

In the following examples, the storage hierarchy proposed will more suited for a multi node Stroom cluster, including the Forwarding or Standalone proxy deployments. This is to simplify the documentation. Thus, the above structure is generalised into

- /stroomdata/stroom-working-p_nn_/proxy

and

- /stroomdata/stroom-data-p_nn_
- /stroomdata/stroom-index-p_nn_
- /stroomdata/stroom-working-p_nn_

where *nn* is a two digit node number. The reason for placing the proxy directory within the *Application* working area will be explained later.

All data should be owned by the Stroom processing user. In this HOWTO, we will use `stroomuser`

4.1.3.5 Multi Node Stroom Cluster (Proxy and Application) Deployment

In this deployment we will install the database on a given node then deploy both the Stroom Proxy and Stroom Application software to both our processing nodes. At this point we will then integrate a web service to run 'in-front' of our Stroom software and then perform the initial configuration of Stroom via the user interface.

4.1.3.6 Database Installation

The Stroom capability requires access to two MySQL/MariaDB databases. The first is for persisting application configuration and metadata information, and the second is for the Stroom Statistics capability. Instructions for installation of the Stroom databases can be found [here](#). Although these instructions describe the deployment of the databases to their own node, there is no reason why one can't just install them both on the first (or only) Stroom node.

4.1.3.7 Prerequisite Software Installation

Certain software packages are required for either the Stroom Proxy or Stroom Application to run.

The core software list is

- java-1.8.0-openjdk
- java-1.8.0-openjdk-devel
- policycoreutils-python
- unzip
- zip
- mariadb or mysql client

Most of the required software are packages available via standard repositories and hence we can simply execute

```
sudo yum -y install java-1.8.0-openjdk java-1.8.0-openjdk-devel policycoreutils-python unzip zip
```

One has a choice of database clients. MariaDB is directly supported by Centos 7 and is simplest to install. This is done via

```
sudo yum -y install mariadb
```

One could deploy the MySQL database software as the alternative.

To do this you need to install the MySQL Community repository files then install the client. Instructions for installation of the MySQL Community repository files can be found [here](#) or on the [MySQL Site \(external link\)](#). Once you have installed the MySQL repository files, install the client via

```
sudo yum -y install mysql-community-client
```

Note that additional software will be required for other integration components (e.g. Apache httpd/mod_jk). This is described in the [Web Service Integration](#) section of this document.

Note also, that Standalone or Forwarding Stroom Proxy deployments do **NOT** need a database client deployed.

4.1.3.7.1 Entropy Issues in Virtual environments

Both the Stroom Application and Stroom Proxy currently run on Tomcat (Version 7) which relies on the Java SecureRandom class to provide random values for any generated session identifiers as well as other components. In some circumstances the Java runtime can be delayed if the entropy source that is used to initialise SecureRandom is short of entropy. The delay is caused by the Java runtime waiting on the blocking entropy souce /dev/random to have sufficient entropy. This quite often occurs in virtual environments were there are few sources that can contribute to a system's entropy.

To view the current available entropy on a Linux system, run the command

```
cat /proc/sys/kernel/random/entropy_avail
```

A reasonable value would be over 2000 and a poor value would be below a few hundred.

If you are deploying Stroom onto systems with low available entropy, the start time for the Stroom Proxy can be as high as 5 minutes and for the Application as high as 15 minutes.

One software based solution would be to install the [haveged \(external link\)](#) service that attempts to provide an easy-to-use, unpredictable random number generator based upon an adaptation of the HAVEGE algorithm. To install execute

```
yum -y install haveged  
systemctl enable haveged  
systemctl start haveged
```

For background reading in this matter, see [this reference \(external link\)](#) or [this reference \(external link\)](#).

4.1.3.8 Storage Scenario

For the purpose of this Installation HOWTO, the following sets up the storage hierarchy for a two node processing cluster. To share our **permanent data** we will use NFS. Accept that the NFS deployment described here is very simple, and in a production deployment, a *lot* more security controls should be used. Further,

Our hierarchy is

- Node: stroomp00.strmdev00.org
- /stroomdata/stroom-data-p00 - location to store Stroom application data files (events, etc.) for this node
- /stroomdata/stroom-index-p00 - location to store Stroom application index files
- /stroomdata/stroom-working-p00 - location to store Stroom application working files (e.g. temporary files, output, etc.) for this node
- /stroomdata/stroom-working-p00/proxy - location for Stroom proxy to store inbound data files
- Node: stroomp01.strmdev00.org

- /stroomdata/stroom-data-p01 - location to store Stroom application data files (events, etc.) for this node
- /stroomdata/stroom-index-p01 - location to store Stroom application index files
- /stroomdata/stroom-working-p01 - location to store Stroom application working files (e.g. temporary files, output, etc.) for this node
- /stroomdata/stroom-working-p01/proxy - location for Stroom proxy to store inbound data files

4.1.3.8.0.1 Creation of Storage Hierarchy

So, we first create processing user on all nodes as per

```
sudo useradd --system stroomuser
```

And the relevant commands to create the above hierarchy would be

- Node: stroomp00.strmdev00.org

```
sudo mkdir -p /stroomdata/stroom-data-p00 /stroomdata/stroom-index-p00 /stroomdata/stroom-working-p00 /stroomdata/stroom-working-p01
sudo mkdir -p /stroomdata/stroom-data-p01 # So that this node can mount stroomp01's data directory
sudo chown -R stroomuser:stroomuser /stroomdata
sudo chmod -R 750 /stroomdata
```

- Node: stroomp01.strmdev00.org

```
sudo mkdir -p /stroomdata/stroom-data-p01 /stroomdata/stroom-index-p01 /stroomdata/stroom-working-p01 /stroomdata/stroom-working-p00
sudo mkdir -p /stroomdata/stroom-data-p00 # So that this node can mount stroomp00's data directory
sudo chown -R stroomuser:stroomuser /stroomdata
sudo chmod -R 750 /stroomdata
```

4.1.3.8.0.2 Deployment of NFS to share Stroom Storage

We will use NFS to cross mount the *permanent data* directories. That is

- node stroomp00.strmdev00.org will mount stroomp01.strmdev00.org:/stroomdata/stroom-data-p01 and,
- node stroomp01.strmdev00.org will mount stroomp00.strmdev00.org:/stroomdata/stroom-data-p00 .

The HOWTO guide to deploy and configure NFS for our Scenario is [here](#)

4.1.3.9 Stroom Installation

4.1.3.9.1 Pre-installation setup

Before installing either the Stroom Proxy or Stroom Application, we need establish various files and scripts within the Stroom Processing user's home directory to support the Stroom services and their persistence. This is setup is described [here](#).

4.1.3.9.2 Stroom Proxy Installation

Instructions for installation of the Stroom Proxy can be found [here](#).

4.1.3.9.3 Stroom Application Installation

Instructions for installation of the Stroom application can be found [here](#).

4.1.3.10 Web Service Integration

One typically 'fronts' either a Stroom Proxy or Stroom Application with a secure web service such as Apache's Httpd or NGINX. In our scenario, we will use SSL to secure the web service and further, we will use Apache's Httpd.

We first need to create certificates for use by the web service. The [following](#) provides instructions for this. The created certificates can then be used when configuration the web service.

This HOWTO is designed to deploy Apache's httpd web service as a front end (https) (to the user) and Apache's mod_jk as the interface between Apache and the Stroom tomcat applications. The instructions to configure this can be found [here](#).

Other Web service capability can be used, for example, [NGINX \(external link\)](#).

4.1.3.11 Installation Validation

We will now check that the installation and web services integration has worked.

4.1.3.11.1 Sanity firewall check

To ensure you have the firewall correctly set up, the following command

```
sudo firewall-cmd --reload  
sudo firewall-cmd --zone=public --list-all
```

should result in

```
public (active)  
target: default  
icmp-block-inversion: no  
interfaces: enp0s3  
sources:  
services: dhcpcv6-client http https nfs ssh  
ports: 8009/tcp 9080/tcp 8080/tcp 9009/tcp  
protocols:  
masquerade: no  
forward-ports:  
sourceports:  
icmp-blocks:  
rich rules:
```

4.1.3.11.2 Test Posting of data to the Stroom service

You can test the data posting service with the command

```
curl -k --data-binary @/etc/group "https://stroomp.strmdev00.org/stroom/datafeed" -H "Feed:TEST-FEED-V1_0" -H "System:EXAMPLE_SY
```

which **WILL** result in an error as we have not configured the Stroom Application as yet. The error should look like

```
<html><head><title>Apache Tomcat/7.0.53 - Error report</title><style><!--H1 {font-family:Tahoma,Arial,sans-serif;color:white;bac
```

If you view the Stroom proxy log, `~/stroom-proxy/instance/logs/stroom.log`, on both processing nodes, you will see on one node, the `datafeed.DataFeedRequestHandler` events running under, in this case, the `ajp-apr-9009-exec-1` thread indicating the failure

```
...  
2017-01-03T03:35:47.366Z WARN [ajp-apr-9009-exec-1] datafeed.DataFeedRequestHandler (DataFeedRequestHandler.java:131) - "handle  
2017-01-03T03:35:47.367Z ERROR [ajp-apr-9009-exec-1] zip.StroomStreamException (StroomStreamException.java:131) - sendErrorRespo  
2017-01-03T03:35:47.368Z INFO [ajp-apr-9009-exec-1] datafeed.DataFeedRequestHandler$1 (DataFeedRequestHandler.java:104) - "doPo  
...
```

Further, if you execute the data posting command (`curl`) multiple times, you will see the loadbalancer working in that, the above WARN/ERROR/INFO logs will swap between the proxy services (i.e. first error will be in `stroomp00.strmdev00.org`'s proxy log file, then second on `stroomp01.strmdev00.org`'s proxy log file, then back to `stroomp00.strmdev00.org` and so on).

4.1.3.12 Stroom Application Configuration

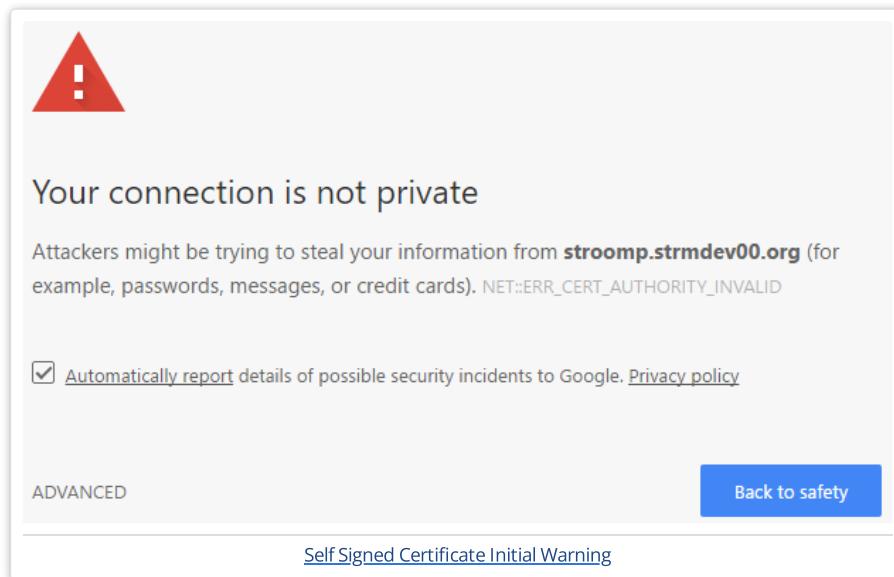
Although we have installed our multi node Stroom cluster, we now need to configure it. We do this via the user interface (UI).

4.1.3.12.1 Logging into the Stroom UI for the first time

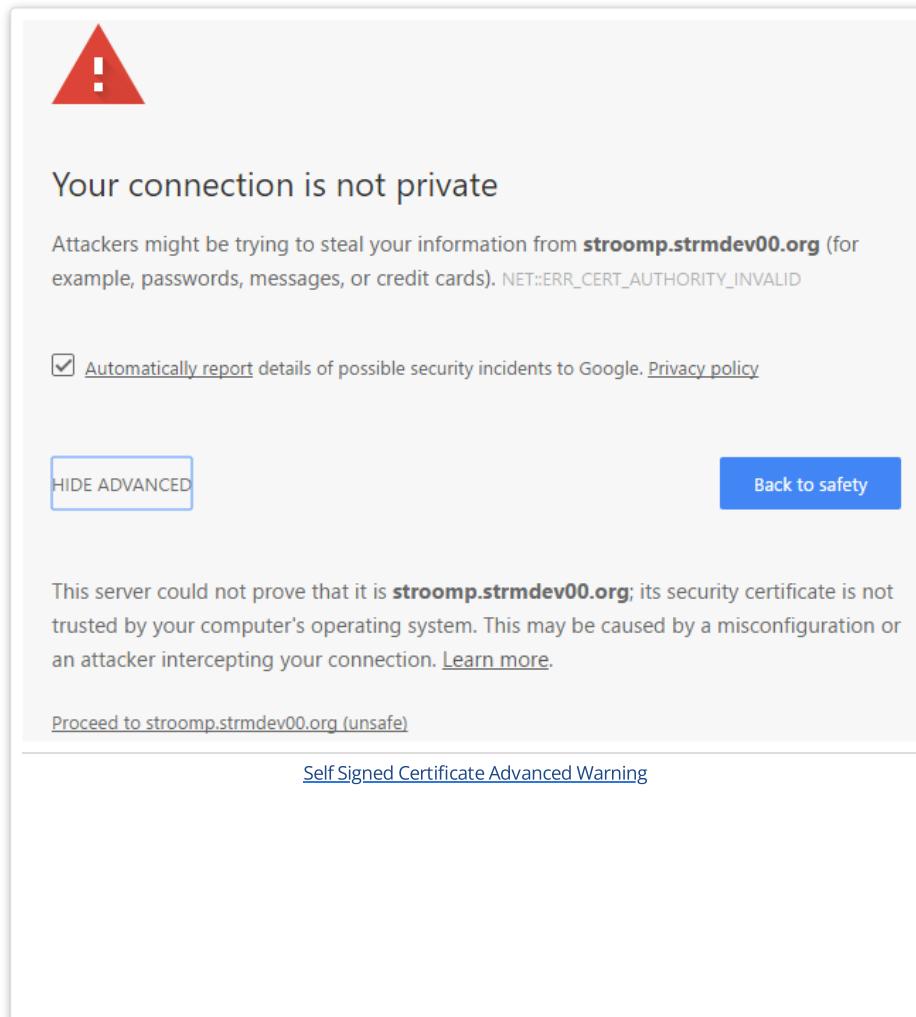
To log into the UI of your newly installed Stroom instance, present the base URL to your [Chrome \(external link\)](#) browser. In this deployment, you should enter the URLs `http://stroomp.strmdev00.org`, or `https://stroomp.strmdev00.org` or `https://stroomp.strmdev00.org/stroom`, noting the first URLs should automatically direct you to the last URL.

If you have personal certificates loaded in your Chrome browser, you may be asked which certificate to use to authenticate yourself to `stroomp.strmdev00.org:443`. As Stroom has not been configured to use user certificates, the choice is not relevant, just choose one and continue.

Additionally, if you are using self-signed certificates, your browser will generate an alert as per



To proceed you need to select the **ADVANCED** hyperlink to see



If you select the **Proceed to stroomp.strmdev00.org (unsafe)** hyper-link you will be presented with the standard Stroom UI login page.



[Stroom UI Login Page](#)

This page has two panels - **About Stroom** and **Login**.

In the **About Stroom** panel we see an introductory description of Stroom in the top left and deployment details in the bottom left of the panel. The deployment details provide

- **Build Version:** - the build version of the Stroom application deployed
- **Build Date:** - the date the version was built
- **Up Date:** - the install date
- **Node Name:** - the node within the Stroom cluster you have connected to

4.1.3.12.2 Login with Stroom default Administrative User

Each new Stroom deployment automatically creates the administrative user `admin` and this user's password is initially set to `admin`. We will [login as this user](#) which also validates that the database and UI is working correctly in that you can login and the password is `admin`.

4.1.3.12.3 Create an Attributed User to perform configuration

We should configure Stroom using an *attributed user* account. That is, we should [create](#) a user, in our case it will be `burn` (the author) and once created, we login with that account then perform the initial configuration activities. You don't have to do this, but it is sound security practice.

Once you have created the user you should [log out](#) of the `admin` account and log back in as our user `burn`.

4.1.3.12.4 Configure the Volumes for our Stroom deployment

Before we can store data within Stroom we need to configure the [volumes](#) we have allocated in our Storage hierarchy. The [Volume Maintenance HOWTO](#) shows how to do this.

4.1.3.12.5 Configure the Nodes for our Stroom deployment

In a Stroom cluster, nodes are expected to communicate with each other on port 8080 over http. Our installation in a multi node environment ensures the firewall will allow this but we also need to configure the nodes. This is achieved via the Stroom UI where we set a Cluster URL for each node. The following [Node Configuration HOWTO](#) demonstrates how to set the Cluster URL.

4.1.3.12.6 Data Stream Processing

To enable Stroom to process data, its [Data Processors](#) need to be enabled. There are NOT enabled by default on installation. The following [section](#) in our [Stroom Tasks HowTo](#) shows how to do this.

4.1.3.13 Testing our Stroom Application and Proxy Installation

To complete the installation process we will test that we can send and ingest data.

4.1.3.13.1 Add a Test Feed

In order for Stroom to be able to handle various data sources, be they Apache HTTPD web access logs, MicroSoft Windows Event logs or Squid Proxy logs, Stroom must be told what the data is when it is received. This is achieved using [Event Feeds](#). Each feed has a unique name within the system.

To test our installation can accept and ingest data, we will [create a test Event feed](#). The 'name' of the feed will be `TEST-FEED-V1_0` . Note that in a production environment it is best that a well defined nomenclature is used for feed 'names'. For our testing purposes `TEST-FEED-V1_0` is sufficient.

4.1.3.13.2 Sending Test Data

NOTE: Before testing our new feed, we should restart both our Stroom application services so that any volume changes are propagated. This can be achieved by simply running

```
sudo -i -u stroomuser  
bin/StopServices.sh  
bin/StartServices.sh
```

on both nodes. It is suggested you first log out of Stroom, if you are currently logged in and you should monitor the Stroom application logs to ensure it has successfully restarted. Remember to use the `tp` and `tp` bash aliases we set up.

For this test, we will send the contents of `/etc/group` to our test feed. We will also send the file from the cluster's database machine. The command to send this file is

```
curl -k --data-binary @/etc/group "https://stroomdev00.org/stroom/datafeed" -H "Feed:TEST-FEED-V1_0" -H "System:EXAMPLE_SY
```

We will test a number of features as part of our installation test. These are

- simple post of data
- simple post of data to validate load balancing is working
- simple post to *direct* feed interface
- simple post to *direct* feed interface to validate load balancing is working
- identify that the Stroom *Proxy Aggregation* is working correctly

As part of our testing will check the presence of the inbound data, as files, within the proxy storage area. Now as the proxy storage area is also the location from which the Stroom application automatically aggregates then ingests the data stored by the proxy, we can either turn off the [Proxy Aggregation](#) task, or attempt to perform our tests noting that proxy aggregation occurs every 10 minutes by default. For simplicity, we will [turn off the Proxy Aggregation task](#).

We can now perform our tests. Follow the steps in the [Data Posting Tests](#) section of the [Testing Stroom Installation HOWTO](#)

4.1.3.14 Forwarding Stroom Proxy Deployment

In this deployment will install a Stroom **Forwarding Proxy** which is designed to aggregate data posted to it for managed forwarding to a central Stroom processing system. This scenario is assuming we are installing on the fully patch Centos 7.3 host, `stroomfp0.stromdev00.org` . Further it assumes we have installed, configured and tested the destination Stroom system we will be forwarding to.

We will first deploy the Stroom Proxy then configure it as a **Forwarding Proxy** then integrate a web service to run 'in-front' of Proxy.

4.1.3.15 Prerequisite Software Installation for Forwarding Proxy

Certain software packages are required for the Stroom Proxy to run.

The core software list is

- `java-1.8.0-openjdk`
- `java-1.8.0-openjdk-devel`
- `policycoreutils-python`
- `unzip`
- `zip`

Most of the required software are packages available via standard repositories and hence we can simply execute

```
sudo yum -y install java-1.8.0-openjdk java-1.8.0-openjdk-devel policycoreutils-python unzip zip
```

Note that additional software will be required for other integration components (e.g. Apache httpd/mod_jk). This is described in the [Web Service Integration for Forwarding Proxy](#) section of this document.

4.1.3.16 Forwarding Proxy Storage

Since we are a proxy that stores data sent to it and forwards it each minute we have only one directory.

- /stroomdata/stroom-working-fp0/proxy - location for Stroom proxy to store inbound data files prior to forwarding

You will note that these HOWTOs use a consistent storage nomenclature for simplicity of documentations.

4.1.3.16.1 Creation of Storage for Forwarding Proxy

We create the processing user, as per

```
sudo useradd --system stroomuser
```

then create the storage hierarchy with the commands

```
sudo mkdir -p /stroomdata/stroom-working-fp0/proxy
sudo chown -R stroomuser:stroomuser /stroomdata
sudo chmod -R 750 /stroomdata
```

4.1.3.17 Stroom Forwarding Proxy Installation

4.1.3.17.1 Pre-installation setup

Before installing the Stroom Forwarding Proxy, we need establish various files and scripts within the Stroom Processing user's home directory to support the Stroom services and their persistence. This is setup is described [here](#). Although this setup HOWTO is orientated towards a complete Stroom Proxy and Application installation, it does provide all the processing user setup requirements for a Stroom Proxy as well.

4.1.3.17.2 Stroom Forwarding Proxy Installation

Instructions for installation of the Stroom Proxy can be found [here](#), noting you should follow the steps for configuring the proxy as a *Forwarding* proxy.

4.1.3.18 Web Service Integration for Forwarding Proxy

One typically 'fronts' a Stroom Proxy with a secure web service such as Apache's Httpd or NGINX. In our scenario, we will use SSL to secure the web service and further, we will use Apache's Httpd.

We first need to create certificates for use by the web service. The [SSL Certificate Generation HOWTO](#) provides instructions for this. The created certificates can then be used when configuration the web service. NOTE also, that for a forwarding proxy we will need to establish Key and Trust stores as well. This is also documented in the SSL Certificate Generation HOWTO [here](#).

This HOWTO is designed to deploy Apache's httpd web service as a front end (https) (to the user) and Apache's mod_jk as the interface between Apache and the Stroom tomcat applications. The instructions to configure this can be found [here](#). Please take note of where a Stroom Proxy configuration item is different to that of a Stroom Application processing node.

Other Web service capability can be used, for example, [NGINX \(external link\)](#).

4.1.3.19 Testing our Forwarding Proxy Installation

To complete the installation process we will test that we can send data to the forwarding proxy and that it forwards the files it receives to the central Stroom processing system. As stated earlier, it is assumed we have installed, configured and tested the destination central Stroom processing system and thus we will have a test [Feed](#) already established - TEST-FEED-V1_0 .

4.1.3.19.1 Sending Test Data

For this test, we will send the contents of /etc/group to our test feed - TEST-FEED-V1_0 . It doesn't matter from which host we send the file from. The command to send file is

```
curl -k --data-binary @/etc/group "https://stroomfp0.strmdev00.org/stroom/datafeed" -H "Feed:TEST-FEED-V1_0" -H "System:EXAMPLE"
```

Before testing, it is recommended you set up to monitor the Stroom proxy logs on the central server as well as on the Forwarding Proxy server.

Follow the steps in the [Forwarding Proxy Data Posting Tests](#) section of the [Testing Stroom Installation HOWTO](#)

4.1.3.20 Standalone Stroom Proxy Deployment

In this deployment will install a Stroom **Standalone Proxy** which is designed to accept and store data posted to it for **manual** forwarding to a central Stroom processing system. This scenario is assuming we are installing on the fully patch Centos 7.3 host, stroomsap0.strmdev00.org .

We will first deploy the Stroom Proxy then configure it as a **Standalone Proxy** then integrate a web service to run 'in-front' of Proxy.

4.1.3.21 Prerequisite Software Installation for Forwarding Proxy

Certain software packages are required for the Stroom Proxy to run.

The core software list is

- java-1.8.0-openjdk
- java-1.8.0-openjdk-devel
- policycoreutils-python
- unzip
- zip

Most of the required software are packages available via standard repositories and hence we can simply execute

```
sudo yum -y install java-1.8.0-openjdk java-1.8.0-openjdk-devel policycoreutils-python unzip zip
```

Note that additional software will be required for other integration components (e.g. Apache httpd/mod_jk). This is described in the [Web Service Integration for Standalone Proxy](#) section of this document.

4.1.3.22 Standalone Proxy Storage

Since we are a proxy that stores data sent to it we have only one directory.

- /stroomdata/stroom-working-sap0/proxy - location for Stroom proxy to store inbound data files

You will note that these HOWTOs use a consistent storage nomenclature for simplicity of documentations.

4.1.3.22.1 Creation of Storage for Standalone Proxy

We create the processing user, as per

```
sudo useradd --system stroomuser
```

then create the storage hierarchy with the commands

```
sudo mkdir -p /stroomdata/stroom-working-sap0/proxy  
sudo chown -R stroomuser:stroomuser /stroomdata  
sudo chmod -R 750 /stroomdata
```

4.1.3.23 Stroom Standalone Proxy Installation

4.1.3.23.1 Pre-installation setup

Before installing the Stroom Standalone Proxy, we need establish various files and scripts within the Stroom Processing user's home directory to support the Stroom services and their persistence. This is setup is described [here](#). Although this setup HOWTO is orientated towards a complete Stroom Proxy and Application installation, it does provide all the processing user setup requirements for a Stroom Proxy as well.

4.1.3.23.2 Stroom Standalone Proxy Installation

Instructions for installation of the Stroom Proxy can be found [here](#), noting you should follow the steps for configuring the proxy as a *Store_NoDB* proxy.

4.1.3.24 Web Service Integration for Standalone Proxy

One typically 'fronts' a Stroom Proxy with a secure web service such as Apache's Httpd or NGINX. In our scenario, we will use SSL to secure the web service and further, we will use Apache's Httpd.

We first need to create certificates for use by the web service. The [SSL Certificate Generation HOWTO](#) provides instructions for this. The created certificates can then be used when configuration the web service. There is no need for Trust or Key stores.

This HOWTO is designed to deploy Apache's httpd web service as a front end (https) (to the user) and Apache's mod_jk as the interface between Apache and the Stroom tomcat applications. The instructions to configure this can be found [here](#). Please take note of where a Stroom Proxy configuration item is different to that of a Stroom Application processing node.

Other Web service capability can be used, for example, [NGINX\(external link\)](#).

4.1.3.25 Testing our Standalone Proxy Installation

To complete the installation process we will test that we can send data to the standalone proxy and it stores it.

4.1.3.25.1 Sending Test Data

For this test, we will send the contents of /etc/group to our test feed - TEST-FEED-V1_0 . It doesn't matter from which host we send the file from. The command to send file is

```
curl -k --data-binary @/etc/group "https://stroomb01.strmdev00.org/stroom/datafeed" -H "Feed:TEST-FEED-V1_0" -H "System:EXAMPLE"
```

Before testing, it is recommended you set up to monitor the Standalone Proxy logs.

Follow the steps in the [Standalone Proxy Data Posting Tests](#) section of the [Testing Stroom Installation HOWTO](#)

4.1.3.26 Addition of a Node to a Stroom Cluster Deployment

In this deployment we will deploy both the Stroom Proxy and Stroom Application software to a new processing node we wish to add to our cluster. Once we have deploy and configured the Stroom software, we will then integrate a web service to run 'in-front' of our Stroom software, and then perform the initial configuration of to add this node via the user interface. The node we will add is stroomp02.strmdev00.org .

4.1.3.27 Grant access to the database for this node

Connect to the Stroom database as the administrative (root) user, via the command

```
sudo mysql --user=root -p
```

and at the MariaDB [(none)]> or mysql> prompt enter

```
grant all privileges on stroom.* to stroomuser@stroomp02.strmdev00.org identified by 'Stroompassword1@';
quit;
```

4.1.3.28 Prerequisite Software Installation

Certain software packages are required for either the Stroom Proxy or Stroom Application to run.

The core software list is

- java-1.8.0-openjdk
- java-1.8.0-openjdk-devel
- policycoreutils-python
- unzip
- zip
- mariadb or mysql client

Most of the required software are packages available via standard repositories and hence we can simply execute

```
sudo yum -y install java-1.8.0-openjdk java-1.8.0-openjdk-devel policycoreutils-python unzip zip
sudo yum -y install mariadb
```

In the above instance, the database client choice is MariaDB as it is directly supported by Centos 7. One could deploy the MySQL database software as the alternative. If you have chosen a different database for the already deployed Stroom Cluster then you should use that one. See [earlier](#) in this document on how to install the MySQL Community client.

Note that additional software will be required for other integration components (e.g. Apache httpd/mod_jk). This is described in the [Web Service Integration](#) section of this document.

4.1.3.28.1 Storage Scenario

To maintain our Storage Scenario them, the scenario for this node is

- Node: stroomp02.strmdev00.org
- /stroomdata/stroom-data-p02 - location to store Stroom application data files (events, etc.) for this node
- /stroomdata/stroom-index-p02 - location to store Stroom application index files
- /stroomdata/stroom-working-p02 - location to store Stroom application working files (e.g. tmp, output, etc.) for this node
- /stroomdata/stroom-working-p02/proxy - location for Stroom proxy to store inbound data files

4.1.3.28.1.1 Creation of Storage Hierarchy

So, we first create processing user on our new node as per

```
sudo useradd --system stroomuser
```

then create the storage via

```
sudo mkdir -p /stroomdata/stroom-data-p02 /stroomdata/stroom-index-p02 /stroomdata/stroom-working-p02 /stroomdata/stroom-working-p02/proxy
sudo mkdir -p /stroomdata/stroom-data-p00 # So that this node can mount stroomp00's data directory
sudo mkdir -p /stroomdata/stroom-data-p01 # So that this node can mount stroomp01's data directory
sudo chown -R stroomuser:stroomuser /stroomdata
sudo chmod -R 750 /stroomdata
```

As we need to share this new nodes **permanent data** directories to the existing nodes in the Cluster, we need to create mount point directories on our existing nodes in addition to deploying NFS.

So we execute on

- Node: stroomp00.strmdev00.org

```
sudo mkdir -p /stroomdata/stroom-data-p02
sudo chmod 750 /stroomdata/stroom-data-p02
sudo chown stroomuser:stroomuser /stroomdata/stroom-data-p02
```

and on

- Node: stroomp01.strmdev00.org

```
sudo mkdir -p /stroomdata/stroom-data-p02
sudo chmod 750 /stroomdata/stroom-data-p02
sudo chown stroomuser:stroomuser /stroomdata/stroom-data-p02
```

4.1.3.28.1.2 Deployment of NFS to share Stroom Storage

We will use NFS to cross mount the *permanent data* directories. That is

- node stroomp00.strmdev00.org will mount
 - stroomp01.strmdev00.org:/stroomdata/stroom-data-p01 and,
 - stroomp02.strmdev00.org:/stroomdata/stroom-data-p02 and,
- node stroomp01.strmdev00.org will mount
 - stroomp00.strmdev00.org:/stroomdata/stroom-data-p00 and
 - stroomp02.strmdev00.org:/stroomdata/stroom-data-p02
- node stroomp02.strmdev00.org will mount
 - stroomp00.strmdev00.org:/stroomdata/stroom-data-p00 and
 - stroomp01.strmdev00.org:/stroomdata/stroom-data-p01

The HOWTO guide to deploy and configure NFS for our Scenario is [here](#).

4.1.3.29 Stroom Installation

4.1.3.29.1 Pre-installation setup

Before installing either the Stroom Proxy or Stroom Application, we need establish various files and scripts within the Stroom Processing user's home directory to support the Stroom services and their persistence. This is setup is described [here](#). Note you should remember to set the **N** bash variable when generating the Environment Variable files to 02.

4.1.3.29.2 Stroom Proxy Installation

Instructions for installation of the Stroom Proxy can be found [here](#). Note you will be deploying a *Store* proxy and during the setup execution ensure you enter the appropriate values for NODE ('stroomp02') and REPO_DIR ('/stroomdata/stroom-working-p02/proxy'). All other values will be the same.

4.1.3.29.3 Stroom Application Installation

Instructions for installation of the Stroom application can be found [here](#). When executing the setup script ensure you enter the appropriate values for TEMP_DIR ('/stroomdata/stroom-working-p02') and NODE ('stroomp02'). All other values will be the same. Note also that you will not have to wait for the 'first' node to initialise the Stroom database as this would have already been done when you first deployed your Stroom Cluster.

4.1.3.30 Web Service Integration

One typically 'fronts' either a Stroom Proxy or Stroom Application with a secure web service such as Apache's Httpd or NGINX. In our scenario, we will use SSL to secure the web service and further, we will use Apache's Httpd.

As we are a cluster, we use the same certificate as the other nodes. Thus we need to gain the certificate package from an existing node.

So, on stroomp00.strmdev00.org , we replicate the directory ~stroomuser/stroom-jks to our new node. That is, tar it up, copy the tar file to stroomp02 and untar it. We can make use of the other node's mounted file system.

```
sudo -i -u stroomuser
cd ~stroomuser
tar cf stroom-jks.tar stroom-jks
mv stroom-jks.tar /stroomdata/stroom-data-p02
```

then on our new node (`stroomp02.strmdev00.org`) we extract the data.

```
sudo -i -u stroomuser
cd ~stroomuser
tar xf /stroomdata/stroom-data-p02/stroom-jks.tar && rm -f /stroomdata/stroom-data-p02/stroom-jks.tar
```

Now ensure protection, ownership and SELinux context for these files by running

```
chmod 700 ~stroomuser/stroom-jks/private ~stroomuser/stroom-jks
chown -R stroomuser:stroomuser ~stroomuser/stroom-jks
chcon -R --reference /etc/pki ~stroomuser/stroom-jks
```

This HOWTO is designed to deploy Apache's httpd web service as a front end (https) (to the user) and Apache's mod_jk as the interface between Apache and the Stroom tomcat applications. The instructions to configure this can be found [here](#). You should pay particular attention to the section on the [Apache Mod_JK configuration](#) as you **MUST** regenerate the `Mod_JK workers.properties` file on the existing cluster nodes as well as generating it on our new node.

Other Web service capability can be used, for example, [NGINX \(external link\)](#).

Note that once you have integrated the web services for our new node, you will need to restart the Apache systemd process on the existing two nodes that the new Mod_JK configuration has taken place.

4.1.3.31 Installation Validation

We will now check that the installation and web services integration has worked. We do this with a simple firewall check and [later](#) perform complete integration tests.

4.1.3.31.1 Sanity firewall check

To ensure you have the firewall correctly set up, the following command

```
sudo firewall-cmd --reload
sudo firewall-cmd --zone=public --list-all
```

should result in

```
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3
  sources:
  services: dhcpcv6-client http https nfs ssh
  ports: 8009/tcp 9080/tcp 8080/tcp 9009/tcp
  protocols:
  masquerade: no
  forward-ports:
  sourceports:
  icmp-blocks:
  rich rules:
```

4.1.3.32 Stroom Application Configuration - New Node

We will need to configure this new node's [volumes](#), set it's Cluster URL and enable it's Stream Processors. We do this by logging into the Stroom User Interface (UI) with an account with Administrator privileges. It is recommended you use a attributed user for this activity. Once you have logged in you can configure this new node.

4.1.3.32.1 Configure the Volumes for our Stroom deployment

Before we can store data on this new Stroom node we need to configure it's [volumes](#) we have allocated in our Storage hierarchy. The section on adding new [volumes](#) in the [Volume Maintenance HOWTO](#) shows how to do this.

4.1.3.32.2 Configure the Nodes for our Stroom deployment

In a Stroom cluster, nodes are expected to communicate with each other on port 8080 over http. Our installation in a multi node environment ensures the firewall will allow this but we also need to configure the new node. This is achieved via the Stroom UI where we set a Cluster URL for our node. The section on Configuring a new node in the [Node Configuration HOWTO](#) demonstrates how do set the Cluster URL.

4.1.3.32.3 Data Stream Processing

To enable Stroom to process data, it's [Data Processors](#) need to be enabled. There are NOT enabled by default on installation. The following [section](#) in our [Stroom Tasks HowTo](#) shows how to do this.

4.1.3.33 Testing our New Node Installation

To complete the installation process we will test that our new node has successfully integrated into our cluster.

First we need to ensure we have restarted the Apache Httpd service (`httpd.service`) on the original nodes so that the `workers.properties` configuration files take effect.

We now test the node integration by running the tests we use to validate a Multi Node Stroom Cluster Deployment found [here](#) noting we should monitor all three nodes proxy and application log files. Basically we are looking to see that this new node participates in the load balancing for the `stroomp.strmdev00.org` cluster.

4.1.4 - Installation of Stroom Application

This HOWTO describes the installation and initial configuration of the Stroom Application.

4.1.4.1 Assumptions

- the user has reasonable RHEL/Centos System administration skills
- installation is on a fully patched minimal Centos 7.3 instance.
- the Stroom `stroom` database has been created and resides on the host `stroomb0.strmdev00.org` listening on port 3307.
- the Stroom `stroom` database user is `stroomuser` with a password of `Stroompassword1@`.
- the Stroom `statistics` database has been created and resides on the host `stroomb0.strmdev00.org` listening on port 3308.
- the Stroom `statistics` database user is `stroomuser` with a password of `Stroompassword2@`.
- the application user `stroomuser` has been created
- the user is or has deployed the two node Stroom cluster described [here](#)
- the user has set up the Stroom processing user as described [here](#)
- the prerequisite software has been installed
- when a screen capture is documented, data entry is identified by the data surrounded by '<' >'. This excludes enter/return presses.

4.1.4.2 Confirm Prerequisite Software Installation

The following command will ensure the prerequisite software has been deployed

```
sudo yum -y install java-1.8.0-openjdk java-1.8.0-openjdk-devel policycoreutils-python unzip zip
sudo yum -y install mariadb
or
sudo yum -y install mysql-community-client
```

4.1.4.3 Test Database connectivity

We need to test access to the Stroom databases on `stroomb0.strmdev00.org`. We do this using the client `mysql` utility. We note that we must enter the `stroomuser` user's password set up in the creation of the database earlier (`Stroompassword1@`) when connecting to the `stroom` database and we must enter the `stroomstats` user's password (`Stroompassword2@`) when connecting to the `statistics` database.

We first test we can connect to the `stroom` database and then set the default database to be `stroom`.

```
[burn@stroomp00 ~]$ mysql --user=stroomuser --host=stroomb0.strmdev00.org --port=3307 --password
Enter password: <__ Stroompassword1@ __>
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 5.5.52-MariaDB MariaDB Server

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use stroom;
Database changed
MariaDB [stroom]> exit
Bye
[burn@stroomp00 ~]$
```

In the case of a MySQL Community deployment you will see

```
[burn@stroomp00 ~]$ mysql --user=stroomuser --host=stroomb0.strmdev00.org --port=3307 --password
Enter password: <__ Stroompassword1@ __>
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 5.7.18 MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use stroom;
Database changed
mysql> quit
Bye
[burn@stroomp00 ~]$
```

We next test connecting to the `statistics` database and verify we can set the default database to be `statistics`.

```
[burn@stroomp00 ~]$ mysql --user=stroomstats --host=stroomb0.strmdev00.org --port=3308 --password
Enter password: <__ Stroompassword2@ __>
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 5.5.52-MariaDB MariaDB Server

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use statistics;
Database changed
MariaDB [stroom]> exit
Bye
[burn@stroomp00 ~]$
```

In the case of a MySQL Community deployment you will see

```
[burn@stroomp00 ~]$ mysql --user=stroomstats --host=stroomb0.strmdev00.org --port=3308 --password
Enter password: <__ Stroompassword2@ __>
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 5.7.18 MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use statistics;
Database changed
mysql> quit
Bye
[burn@stroomp00 ~]$
```

If there are any errors, correct them.

4.1.4.4 Get the Software

The following will gain the identified, in this case release `5.0-beta.18`, Stroom Application software release from github, then deploy it. You should regularly monitor the site for newer releases.

```

sudo -i -u stroomuser
App=5.0-beta.18
wget https://github.com/gchq/stroom/releases/download/v${App}/stroom-app-distribution-${App}-bin.zip
unzip stroom-app-distribution-${App}-bin.zip
chmod 750 stroom-app

```

4.1.4.5 Configure the Software

We install the application via

```
stroom-app/bin/setup.sh
```

during which one is prompted for a number of configuration settings. Use the following

```

TEMP_DIR should be set to '/stroomdata/stroom-working-p00' or '/stroomdata/stroom-working-p01' etc depending on the node we are installing on
NODE to be the hostname (not FQDN) of your host (i.e. 'stroomp00' or 'stroomp01' in our multi node scenario)
RACK can be ignored, just press return
PORT_PREFIX should use the default, just press return
JDBC_CLASSNAME should use the default, just press return
JDBC_URL to 'jdbc:mysql://stroomdb0.strmdev00.org:3307/stroom?useUnicode=yes&characterEncoding=UTF-8'
DB_USERNAME should be our processing user, 'stroomuser'
DB_PASSWORD should be the one we set when creating the stroom database, that is 'Stroompassword1@'
JPA_DIALECT should use the default, just press return
JAVA_OPTS can use the defaults, but ensure you have sufficient memory, either change or accept the default
STROOM_STATISTICS_SQL_JDBC_CLASSNAME should use the default, just press return
STROOM_STATISTICS_SQL_JDBC_URL to 'jdbc:mysql://stroomdb0.strmdev00.org:3308/statistics?useUnicode=yes&characterEncoding=UTF-8'
STROOM_STATISTICS_SQL_DB_USERNAME should be our processing user, 'stroomstats'
STROOM_STATISTICS_SQL_DB_PASSWORD should be the one we set when creating the stroom database, that is 'Stroompassword2@'
STATS_ENGINES should use the default, just press return
CONTENT_PACK_IMPORT_ENABLED should use the default, just press return
CREATE_DEFAULT_VOLUME_ON_START should use the default, just press return

```

At this point, the script will configure the application. There should be no errors, but review the output. If you made an error then just re-run the script.

You will note that **TEMP_DIR** is the same directory we used for our **STROOM_TMP** environment variable when we set up the processing user scripts. Note that if you are deploying a single node environment, where the database is also running on your Stroom node, then the **JDBC_URL** setting can be the default.

4.1.4.6 Start the Application service

Now we start the application. In the case of multi node Stroom deployment, we start the Stroom application on the first node in the cluster, then **wait** until it has initialised the database commenced its Lifecycle task. You will need to monitor the log file to see its completed initialisation.

So as the `stroomuser` start the application with the command

```
stroom-app/bin/start.sh
```

Now monitor `stroom-app/instance/logs` for any errors. Initially you will see the log files `localhost_access_log.YYYY-MM-DD.txt` and `catalina.out`. Check them for errors and correct (or post a question). The log4j warnings in `catalina.out` can be ignored. Eventually the log file `stroom-app/instance/logs/stroom.log` will appear. Again check it for errors and then wait for the application to be initialised. That is, wait for the Lifecycle service thread to start. This is indicated by the message

```
INFO [Thread-11] lifecycle.LifecycleServiceImpl (LifecycleServiceImpl.java:166) - Started Stroom Lifecycle service
```

The directory `stroom-app/instance/logs/events` will also appear with an empty file with the nomenclature `events_YYYY-MM-DDThh:mm:ss.msecz`. This is the directory for storing Stroom's application event logs. We will return to this directory and its content in a later HOWTO.

If you have a multi node configuration, then once the database has initialised, start the application service on all other nodes. Again with

```
stroom-app/bin/start.sh
```

and then monitor the files in its `stroom-app/instance/logs` for any errors. Note that in multi node configurations, you will see `server.UpdateClusterStateTaskHandler` messages in the log file of the form

```
WARN [Stroom P2 #9 - GenericServerTask] server.UpdateClusterStateTaskHandler (UpdateClusterStateTaskHandler.java:150) - discover() - unable to co
```

This is ok as we will establish the cluster URL's later.

4.1.4.6.1 Multi Node Firewall Provision

In the case of a multi node Stroom deployment, you will need to open certain ports to allow Tomcat to communicate to all nodes participating in the cluster. Execute the following on all nodes. Note you will need to drop out of the `stroomuser` shell prior to execution.

```
exit; # To drop out of the stroomuser shell

sudo firewall-cmd --zone=public --add-port=8080/tcp --permanent
sudo firewall-cmd --zone=public --add-port=9080/tcp --permanent
sudo firewall-cmd --zone=public --add-port=8009/tcp --permanent
sudo firewall-cmd --zone=public --add-port=9009/tcp --permanent
sudo firewall-cmd --reload
sudo firewall-cmd --zone=public --list-all
```

In a production environment you would improve the above firewall settings - to perhaps limit the communication to just the Stroom processing nodes.

4.1.5 - Installation of Stroom Proxy

This HOWTO describes the installation and configuration of the Stroom Proxy software.

4.1.5.1 Assumptions

The following assumptions are used in this document.

- the user has reasonable RHEL/Centos System administration skills.
- installation is on a fully patched minimal Centos 7.3 instance.
- the Stroom database has been created and resides on the host `stroomb0.strmdev00.org` listening on port 3307.
- the Stroom database user is `stroomuser` with a password of `Stroompassword1@`.
- the application user `stroomuser` has been created.
- the user is or has deployed the two node Stroom cluster described [here](#).
- the user has set up the Stroom processing user as described [here](#).
- the prerequisite software has been installed.
- when a screen capture is documented, data entry is identified by the data surrounded by '`<'>`' . This excludes enter/return presses.

4.1.5.2 Confirm Prerequisite Software Installation

The following command will ensure the prerequisite software has been deployed

```
sudo yum -y install java-1.8.0-openjdk java-1.8.0-openjdk-devel policycoreutils-python unzip zip
sudo yum -y install mariadb
or
sudo yum -y install mysql-community-client
```

Note that we do **NOT** need the database client software for a Forwarding or Standalone proxy.

4.1.5.3 Get the Software

The following will gain the identified, in this case release `5.1-beta.10` , Stroom Application software release from github, then deploy it. You should regularly monitor the site for newer releases.

```
sudo -i -u stroomuser
Prx=v5.1-beta.10
wget https://github.com/gchq/stroom-proxy/releases/download/${Prx}/stroom-proxy-distribution-${Prx}.zip
unzip stroom-proxy-distribution-${Prx}.zip
```

4.1.5.4 Configure the Software

There are three different types of Stroom Proxy

- Store

A *store* proxy accepts batches of events, as files. It will validate the batch with the database then store the batches as files in a configured directory.

- Store_NoDB

A *store_nodb* proxy accepts batches of events, as files. It has no connectivity to the database, so it assumes all batches are valid, so it stores the batches as files in a configured directory.

- Forwarding

A *forwarding* proxy accepts batches of events, as files. It has indirect connectivity to the database via the destination proxy, so it validates the batches then stores the batches as files in a configured directory until they are periodically forwarded to the configured destination Stroom proxy.

4.1.5.4.1 Store Proxy Configuration

In our *Store* Proxy description below, we will use the multi node deployment scenario. That is we are deploying the *Store* proxy on multiple Stroom nodes (stroomp00, stroomp01) and we have configured our storage as per the [Storage Scenario](#) which means the directories to install the inbound batches of data are /stroomdata/stroom-working-p00/proxy and /stroomdata/stroom-working-p01/proxy depending on the node.

To install a *Store* proxy, we run

```
stroom-proxy/bin/setup.sh store
```

during which one is prompted for a number of configuration settings. Use the following

```
NODE to be the hostname (not FQDN) of your host (i.e. 'stroomp00' or 'stroomp01' depending on the node we are installing on)
PORT_PREFIX should use the default, just press return
REPO_DIR should be set to '/stroomdata/stroom-working-p00/proxy' or '/stroomdata/stroom-working-p01/proxy' depending on the node we are installing
REPO_FORMAT can be left as the default, just press return
JDBC_CLASSNAME should use the default, just press return
JDBC_URL should be set to 'jdbc:mysql://stroomdb0.strmdev00.org:3307/stroom'
DB_USERNAME should be our processing user, 'stroomuser'
DB_PASSWORD should be the one we set when creating the stroom database, that is 'Stroompassword1@'
JAVA_OPTS can use the defaults, but ensure you have sufficient memory, either change or accept the default
```

At this point, the script will configure the proxy. There should be no errors, but review the output. If you make a mistake in the above, just re-run the script.

NOTE: The selection of the REPO_DIR above and the setting of the STROOM_TMP environment variable [earlier](#) ensure that not only inbound files are placed in the REPO_DIR location but the Stroom Application itself will access the same directory when it aggregates inbound data for ingest in its proxy aggregation threads.

4.1.5.4.2 Forwarding Proxy Configuration

In our *Forwarding* Proxy description below, we will deploy on a host named stroomfp0 and it will store the files in /stroomdata/stroom-working-fp0/proxy . Remember, we are being consistent with our Storage hierarchy to make documentation and scripting simpler. Our destination host to periodically forward the files to will be stroomp.strmdev00.org (the CNAME for stroomp00.strmdev00.org).

To install a *Forwarding* proxy, we run

```
stroom-proxy/bin/setup.sh forward
```

during which one is prompted for a number of configuration settings. Use the following

```
NODE to be the hostname (not FQDN) of your host (i.e. 'stroomfp0' in our example)
PORT_PREFIX should use the default, just press return
REPO_DIR should be set to '/stroomdata/stroom-working-fp0/proxy' which we created earlier.
REPO_FORMAT can be left as the default, just press return
FORWARD_SERVER should be set to our stroom server. (i.e. 'stroomp.strmdev00.org' in our example)
JAVA_OPTS can use the defaults, but ensure you have sufficient memory, either change or accept the default
```

At this point, the script will configure the proxy. There should be no errors, but review the output.

4.1.5.4.3 Store No Database Proxy Configuration

In our *Store_NoDB* Proxy description below, we will deploy on a host named stroomsap0 and it will store the files in /stroomdata/stroom-working-sap0/proxy . Remember, we are being consistent with our Storage hierarchy to make documentation and scripting simpler.

To install a *Store_NoDB* proxy, we run

```
stroom-proxy/bin/setup.sh store_nodb
```

during which one is prompted for a number of configuration settings. Use the following

```
NODE to be the hostname (not FQDN) of your host (i.e. 'stroomsap0' in our example)
PORT_PREFIX should use the default, just press return
REPO_DIR should be set to '/stroomdata/stroom-working-sap0/proxy' which we created earlier.
REPO_FORMAT can be left as the default, just press return
JAVA_OPTS can use the defaults, but ensure you have sufficient memory, either change or accept the default
```

At this point, the script will configure the proxy. There should be no errors, but review the output.

4.1.5.5 Apache/Mod_JK change

For all proxy deployments, if we are using Apache's mod_jk then we need to ensure the proxy's AJP connector specifies a 64K packetSize. View the file `stroom-proxy/instance/conf/server.xml` to ensure the Connector element for the AJP protocol has a `packetSize` attribute of `65536`. For example,

```
grep AJP stroom-proxy/instance/conf/server.xml
```

shows

```
<Connector port="9009" protocol="AJP/1.3" connectionTimeout="20000" redirectPort="8443" maxThreads="200" packetSize="65536" />
```

This check is required for earlier releases of the Stroom Proxy. Releases since `v5.1-beta.4` have set the AJP `packetSize`.

4.1.5.6 Start the Proxy Service

We can now manually start our proxy service. Do so as the `stroomuser` with the command

```
stroom-proxy/bin/start.sh
```

Now monitor the directory `stroom-proxy/instance/logs` for any errors. Initially you will see the log files `localhost_access_log.YYYY-MM-DD.txt` and `catalina.out`. Check them for errors and correct (or pose a question to this arena). The context path and unknown version warnings in `catalina.out` can be ignored.

Eventually (about 60 seconds) the log file `stroom-proxy/instance/logs/stroom.log` will appear. Again check it for errors. The proxy will have completely started when you see the messages

```
INFO [localhost-startStop-1] spring.StroomBeanLifeCycleReloadableContextBeanProcessor (StroomBeanLifeCycleReloadableContextBeanProcessor.java:109)
```

and

```
INFO [localhost-startStop-1] spring.StroomBeanLifeCycleReloadableContextBeanProcessor (StroomBeanLifeCycleReloadableContextBeanProcessor.java:109)
```

If you leave it for a while you will eventually see cyclic (10 minute cycle) messages of the form

```
INFO [Repository Reader Thread 1] repo.ProxyRepositoryReader (ProxyRepositoryReader.java:170) - run() - Cron Match at YYYY-MM-DD ...
```

If a proxy takes too long to start, you should read the section on [Entropy issues](#).

4.1.5.7 Proxy Repository Format

A Stroom Proxy stores inbound files in a hierarchical file system whose root is supplied during the proxy setup (`REPO_DIR`) and as files arrive they are given a *repository id* that is a one-up number starting at one (1). The files are stored in a specific *repository format*. The default template is `${pathId}/${id}` and this pattern will produce the following output files under `REPO_DIR` for the given repository id

Repository Id	FilePath
1	000.zip

Repository Id	FilePath
100	100.zip
1000	001/001000.zip
10000	010/010000.zip
100000	100/100000.zip

Since version v5.1-beta.4, this template can be specified during proxy setup via the entry to the `stroom Proxy Repository Format` prompt

```
...
@@REPO_FORMAT@@ : Stroom Proxy Repository Format [${pathId}/${id}] >
...
```

The template uses replacement variables to form the file path. As indicated above, the default template is `${pathId}/${id}` where `${pathId}` is the automatically generated directory for a given *repository id* and `${id}` is the *repository id*.

Other replacement variables can be used to in the template including http header meta data parameters (e.g. ' `${feed}`') and time based parameters (e.g. ' `${year}`'). Replacement variables that cannot be resolved will be output as '_'. You must ensure that all templates include the ' `${id}`' replacement variable at the start of the file name, failure to do this will result in an invalid repository.

Available time based parameters are based on the file's time of processing and are zero filled (excluding `ms`).

Parameter	Description
year	four digit year
month	two digit month
day	two digit day
hour	two digit hour
minute	two digit minute
second	two digit second
millis	three digit milliseconds value
ms	milliseconds since Epoch value

4.1.5.7.1 Proxy Repository Template Examples

For each of the following templates applied to a Store NoDB Proxy, the resultant proxy directory tree is shown after three posts were sent to the test feed `TEST-FEED-V1_0` and two posts to the test feed `FEED-NOVALUE-V9_0`

4.1.5.7.1.1 Example A - The default - `${pathId}/${id}`

```
[stroomuser@stroomb0 ~]$ find /stroomdata/stroom-working-sap0/proxy/
/stroomdata/stroom-working-sap0/proxy/
/stroomdata/stroom-working-sap0/proxy/001.zip
/stroomdata/stroom-working-sap0/proxy/002.zip
/stroomdata/stroom-working-sap0/proxy/003.zip
/stroomdata/stroom-working-sap0/proxy/004.zip
/stroomdata/stroom-working-sap0/proxy/005.zip
[stroomuser@stroomb0 ~]$
```

4.1.5.7.1.2 Example B - A feed orientated structure - `${feed}/${year}/${month}/${day}/${pathId}/${id}`

```
[stroomuser@stroombucket ~]$ find /stroomdata/stroom-working-sap0/proxy/
/stroomdata/stroom-working-sap0/proxy/
/stroomdata/stroom-working-sap0/proxy/TEST-FEED-V1_0
/stroomdata/stroom-working-sap0/proxy/TEST-FEED-V1_0/2017
/stroomdata/stroom-working-sap0/proxy/TEST-FEED-V1_0/2017/07
/stroomdata/stroom-working-sap0/proxy/TEST-FEED-V1_0/2017/07/23
/stroomdata/stroom-working-sap0/proxy/TEST-FEED-V1_0/2017/07/23/001.zip
/stroomdata/stroom-working-sap0/proxy/TEST-FEED-V1_0/2017/07/23/002.zip
/stroomdata/stroom-working-sap0/proxy/TEST-FEED-V1_0/2017/07/23/003.zip
/stroomdata/stroom-working-sap0/proxy/FEED-NOVALUE-V9_0
/stroomdata/stroom-working-sap0/proxy/FEED-NOVALUE-V9_0/2017
/stroomdata/stroom-working-sap0/proxy/FEED-NOVALUE-V9_0/2017/07
/stroomdata/stroom-working-sap0/proxy/FEED-NOVALUE-V9_0/2017/07/23
/stroomdata/stroom-working-sap0/proxy/FEED-NOVALUE-V9_0/2017/07/23/004.zip
/stroomdata/stroom-working-sap0/proxy/FEED-NOVALUE-V9_0/2017/07/23/005.zip
[stroomuser@stroombucket ~]$
```

4.1.5.7.1.3 Example C - A date orientated structure - \${year}/\${month}/\${day}/\${pathId}/\${id}

```
[stroomuser@stroombucket ~]$ find /stroomdata/stroom-working-sap0/proxy/
/stroomdata/stroom-working-sap0/proxy/
/stroomdata/stroom-working-sap0/proxy/2017
/stroomdata/stroom-working-sap0/proxy/2017/07
/stroomdata/stroom-working-sap0/proxy/2017/07/23
/stroomdata/stroom-working-sap0/proxy/2017/07/23/001.zip
/stroomdata/stroom-working-sap0/proxy/2017/07/23/002.zip
/stroomdata/stroom-working-sap0/proxy/2017/07/23/003.zip
/stroomdata/stroom-working-sap0/proxy/2017/07/23/004.zip
/stroomdata/stroom-working-sap0/proxy/2017/07/23/005.zip
[stroomuser@stroombucket ~]$
```

4.1.5.7.1.4 Example D - A feed orientated structure, but with a bad parameter - \${feed}/\${badparam}/\${day}/\${pathId}/\${id}

```
[stroomuser@stroombucket ~]$ find /stroomdata/stroom-working-sap0/proxy/
/stroomdata/stroom-working-sap0/proxy/
/stroomdata/stroom-working-sap0/proxy/TEST-FEED-V1_0
/stroomdata/stroom-working-sap0/proxy/TEST-FEED-V1_0/_/
/stroomdata/stroom-working-sap0/proxy/TEST-FEED-V1_0/_/23
/stroomdata/stroom-working-sap0/proxy/TEST-FEED-V1_0/_/23/001.zip
/stroomdata/stroom-working-sap0/proxy/TEST-FEED-V1_0/_/23/002.zip
/stroomdata/stroom-working-sap0/proxy/TEST-FEED-V1_0/_/23/003.zip
/stroomdata/stroom-working-sap0/proxy/FEED-NOVALUE-V9_0
/stroomdata/stroom-working-sap0/proxy/FEED-NOVALUE-V9_0/_/
/stroomdata/stroom-working-sap0/proxy/FEED-NOVALUE-V9_0/_/23
/stroomdata/stroom-working-sap0/proxy/FEED-NOVALUE-V9_0/_/23/004.zip
/stroomdata/stroom-working-sap0/proxy/FEED-NOVALUE-V9_0/_/23/005.zip
[stroomuser@stroombucket ~]$
```

and one would also see a warning for each post in the proxy's log file of the form

```
WARN [ajp-apr-9009-exec-4] repo.StroomFileNameUtil (StroomFileNameUtil.java:133) - Unused variables found: [badparam]
```

4.1.6 - NFS Installation and Configuration

The following is a HOWTO to assist users in the installation and set up of NFS to support the sharing of directories in a two node Stroom cluster or add a new node to an existing cluster.

4.1.6.1 Assumptions

The following assumptions are used in this document.

- the user has reasonable RHEL/Centos System administration skills
- installations are on Centos 7.3 minimal systems (fully patched)
- the user is or has deployed the example two node Stroom cluster storage hierarchy described [here](#)
- the configuration of this NFS is NOT secure. It is highly recommended to improve it's security in a production environment. This could include improved firewall configuration to limit NFS access, NFS4 with Kerberos etc.

4.1.6.2 Installation of NFS software

We install NFS on each node, via

```
sudo yum -y install nfs-utils
```

and enable the relevant services, via

```
sudo systemctl enable rpcbind
sudo systemctl enable nfs-server
sudo systemctl enable nfs-lock
sudo systemctl enable nfs-idmap
sudo systemctl start rpcbind
sudo systemctl start nfs-server
sudo systemctl start nfs-lock
sudo systemctl start nfs-idmap
```

4.1.6.3 Configuration of NFS exports

We now export the node's /stroomdata directory (in case you want to share the working directories) by configuring /etc/exports. For simplicity sake, we will allow all nodes with the hostname nomenclature of stroomp*.strmdev00.org to mount the /stroomdata directory. This means the same configuration applies to all nodes.

```
# Share Stroom data directory
/stroomdata    stroomp*.strmdev00.org(rw, sync, no_root_squash)
```

This can be achieved with the following on both nodes

```
sudo su -c "printf '# Share Stroom data directory\n' >> /etc/exports"
sudo su -c "printf '/stroomdata\tstroomp*.strmdev00.org(rw, sync, no_root_squash)\n' >> /etc/exports"
```

On both nodes restart the NFS service to ensure the above export takes effect via

```
sudo systemctl restart nfs-server
```

So that our nodes can offer their filesystems, we need to enable NFS access on the firewall. This is done via

```
sudo firewall-cmd --zone=public --add-service=nfs --permanent
sudo firewall-cmd --reload
sudo firewall-cmd --zone=public --list-all
```

4.1.6.4 Test Mounting

You should do test mounts on each node.

- Node: stroomp00.strmdev00.org

```
sudo mount -t nfs4 stroomp01.strmdev00.org:/stroomdata/stroom-data-p01 /stroomdata/stroom-data-p01
```

- Node: stroomp01.strmdev00.org

```
sudo mount -t nfs4 stroomp00.strmdev00.org:/stroomdata/stroom-data-p00 /stroomdata/stroom-data-p00
```

If you are concerned you can't see the mount with a `df` try a `df --type=nfs4 -a` or a `sudo df`. Irrespective, once the mounting works, make the mounts permanent by adding the following to each node's `/etc/fstab` file.

- Node: stroomp00.strmdev00.org

```
stroomp01.strmdev00.org:/stroomdata/stroom-data-p01 /stroomdata/stroom-data-p01 nfs4 soft,bg
```

achieved with

```
sudo su -c "printf 'stroomp01.strmdev00.org:/stroomdata/stroom-data-p01 /stroomdata/stroom-data-p01 nfs4 soft,bg\n' >> /etc/fstab"
```

- Node: stroomp01.strmdev00.org

```
stroomp00.strmdev00.org:/stroomdata/stroom-data-p00 /stroomdata/stroom-data-p00 nfs4 soft,bg
```

achieved with

```
sudo su -c "printf 'stroomp00.strmdev00.org:/stroomdata/stroom-data-p00 /stroomdata/stroom-data-p00 nfs4 soft,bg\n' >> /etc/fstab"
```

At this point reboot all processing nodes to ensure the directories mount automatically. You may need to give the nodes a minute to do this.

4.1.6.5 Addition of another Node

If one needs to add another node to the cluster, lets say, `stroomp02.strmdev00.org`, on which `/stroomdata` follows the same storage hierarchy as the existing nodes and all nodes have added mount points (directories) for this new node, you would take the following steps *in order*.

- Node: stroomp02.strmdev00.org
 - Install NFS software as [above](#)
 - Configure the exports file as per

```
sudo su -c "printf '# Share Stroom data directory\n' >> /etc/exports"
sudo su -c "printf '/stroomdata\tstroomp*.strmdev00.org(rw,sync,no_root_squash)\n' >> /etc/exports"
```

- Restart the NFS service and make the firewall enable NFS access as per

```
sudo systemctl restart nfs-server
sudo firewall-cmd --zone=public --add-service=nfs --permanent
sudo firewall-cmd --reload
sudo firewall-cmd --zone=public --list-all
```

- Test mount the existing node file systems

```
sudo mount -t nfs4 stroomp00.strmdev00.org:/stroomdata/stroom-data-p00 /stroomdata/stroom-data-p00
sudo mount -t nfs4 stroomp01.strmdev00.org:/stroomdata/stroom-data-p01 /stroomdata/stroom-data-p01
```

- Once the test mounts work, we make them permanent by adding the following to the /etc/fstab file.

```
stroomp00.strmdev00.org:/home/stroomdata/stroom-data-p00 /home/stroomdata/stroom-data-p00 nfs4 soft,bg
stroomp01.strmdev00.org:/home/stroomdata/stroom-data-p01 /home/stroomdata/stroom-data-p01 nfs4 soft,bg
```

achieved with

```
sudo su -c "printf 'stroomp00.strmdev00.org:/stroomdata/stroom-data-p00 /stroomdata/stroom-data-p00 nfs4 soft,bg\\n' >> /etc/fstab"
sudo su -c "printf 'stroomp01.strmdev00.org:/stroomdata/stroom-data-p01 /stroomdata/stroom-data-p01 nfs4 soft,bg\\n' >> /etc/fstab"
```

- Node: stroomp00.strmdev00.org **and** stroomp01.strmdev00.org
 - Test mount the new node's filesystem as per

```
sudo mount -t nfs4 stroomp02.strmdev00.org:/stroomdata/stroom-data-p02 /stroomdata/stroom-data-p02
```

- Once the test mount works, make the mount permanent by adding the following to the /etc/fstab file

```
stroomp02.strmdev00.org:/stroomdata/stroom-data-p02 /stroomdata/stroom-data-p02 nfs4 soft,bg
```

achieved with

```
sudo su -c "printf 'stroomp02.strmdev00.org:/stroomdata/stroom-data-p02 /stroomdata/stroom-data-p02 nfs4 soft,bg\\n' >> /etc/fstab"
```

4.1.7 - Node Cluster URL Setup

Configuring Stroom cluster URLs

In a Stroom cluster, [Nodes](#) are expected to communicate with each other on port 8080 over http. To facilitate this, we need to set each node's Cluster URL and the following demonstrates this process.

4.1.7.1 Assumptions

- an account with the [Administrator Application Permission](#) is currently logged in.
- we have a multi node Stroom cluster with two nodes, `stroomp00` and `stroomp01`
- appropriate firewall configurations have been made
- in the scenario of adding a new node to our multi node deployment, the node added will be `stroomp02`

4.1.7.2 Configure Two Nodes

To configure the nodes, move to the `Monitoring` item of the **Main Menu** and select it to bring up the `Monitoring` sub-menu.



then move down and select the `Nodes` sub-item to be presented with the `Nodes` configuration tab as seen below.

1 to 2 of 2

Stroom UI Node Management - management tab

To set `stroomp00`'s Cluster URL, move the its line in the display and select it. It will be highlighted.

Stroom UI Node Management - select first node

Then move the cursor to the *Edit Node* icon in the top left of the `Nodes` tab and select it. On selection the `Edit Node` configuration window will be displayed and into the **Cluster URL:** entry box, enter the first node's URL of
`http://stroomp00.strmdev00.org:8080/stroom/clustercall.rpc`

Name	Cluster URL	Ping	Master	Priority	Enabled
stroomp00		2.0 ms	<input checked="" type="checkbox"/>	1	<input checked="" type="checkbox"/>
stroomp01		No cluster call URL has been set		1	<input checked="" type="checkbox"/>

Edit Node

Name: stroomp00
Cluster URL: http://stroomp00.strmdev00.org:8080/stroom/clustercall.rpc

Ok Cancel

[Stroom UI OkButton](#)

at which we see the **Cluster URL** has been set for the first node's clustercall url for first node

Name	Cluster URL	Ping	Master	Priority	Enabled
stroomp00	http://stroomp00.strmdev00.org:8080/stroom/clustercall.rpc	2.0 ms	<input checked="" type="checkbox"/>	1	<input checked="" type="checkbox"/>
stroomp01		No cluster call URL has been set		1	<input checked="" type="checkbox"/>

[Stroom UI Node Management - set clustercall url on first node](#)

We next select the second node

Name	Cluster URL	Ping	Master	Priority	Enabled
stroomp00	http://stroomp00.strmdev00.org:8080/stroom/clustercall.rpc	1.0 ms	<input checked="" type="checkbox"/>	1	<input checked="" type="checkbox"/>
stroomp01		No cluster call URL has been set		1	<input checked="" type="checkbox"/>

[Stroom UI Node Management - select second node](#)

then move the cursor to the **Edit Node** icon in the top left of the **Nodes** tab and select it. On selection the **Edit Node** configuration window will be displayed and into the **Cluster URL:** entry box, enter the second node's URL of <http://stroomp01.strmdev00.org:8080/stroom/clustercall.rpc>

Name	Cluster URL	Ping	Master	Priority	Enabled
stroomp00	http://stroomp00.strmdev00.org:8080/stroom/clustercall.rpc	1.0 ms	<input checked="" type="checkbox"/>	1	<input checked="" type="checkbox"/>
stroomp01		No cluster call URL has been set		1	<input checked="" type="checkbox"/>

Edit Node

Name: stroomp01
Cluster URL: http://stroomp01.strmdev00.org:8080/stroom/clustercall.rpc

Ok Cancel

[Stroom UI Node Management - set clustercall url for second node](#)

then press the

Ok

[Stroom UI OkButton](#)

At this we will see both nodes have the **Cluster URLs** set.

Name	Cluster URL	Ping	Master	Priority	Enabled
stroomp00	http://stroomp00.strmdev00.org:8080/stroom/clustercall.rpc	2.0 ms	<input checked="" type="checkbox"/>	1	<input checked="" type="checkbox"/>
stroomp01	http://stroomp01.strmdev00.org:8080/stroom/clustercall.rpc	No cluster call URL has been set		1	<input checked="" type="checkbox"/>

[Stroom UI Node Management - both nodes setup](#)

You may need to press the **Refresh** icon found at top left of **Nodes** configuration tab, until both nodes show healthy pings.

Name	Cluster URL	Ping	Master	Priority	Enabled
stroomp00	http://stroomp00.strmdev00.org:8080/stroom/clustercall.rpc	2.0 ms	<input checked="" type="checkbox"/>	1	<input checked="" type="checkbox"/>
stroomp01	http://stroomp01.strmdev00.org:8080/stroom/clustercall.rpc	14 ms		1	<input checked="" type="checkbox"/>

[Stroom UI Node Management - both nodes ping](#)

If you do not get ping results for each node, then they are not configured correctly. In that situation, review all log files and processes that you have performed.

Once you have set the Cluster URLs of each node you should also set the *master assignment priority* for each node to be different to all of the others. In the image above both have been assigned equal priority - 1. We will change stroomp00 to have a different priority - 3. You should note that the node with the highest priority gains the `Master` node status.

Name	Cluster URL	Ping	Master	Priority	Enabled
stroomp00	http://stroomp00.strmdev00.org:8080/stroom/clustercall.rpc	2.0 ms	<input checked="" type="checkbox"/>	3	<input checked="" type="checkbox"/>
stroomp01	http://stroomp01.strmdev00.org:8080/stroom/clustercall.rpc	16 ms		1	<input checked="" type="checkbox"/>

[Stroom UI Node Management - set node priorities](#)

4.1.7.3 Configure New Node

When one expands a Multi Node Stroom cluster deployment, after the installation of the Stroom Proxy and Application software and services on the new node, one has to configure the new node's Cluster URL.

To configure the new node, move to the `Monitoring` item of the **Main Menu** and select it to bring up the `Monitoring` sub-menu.



[Stroom UI Monitoring sub-menu](#)

then move down and select the `Nodes` sub-item to be presented with the `Nodes` configuration tab as seen below.

Name	Cluster URL	Ping	Master	Priority	Enabled
stroomp00	http://stroomp00.strmdev00.org:8080/stroom/clustercall.rpc	1.0 ms	<input checked="" type="checkbox"/>	3	<input checked="" type="checkbox"/>
stroomp01	http://stroomp01.strmdev00.org:8080/stroom/clustercall.rpc	5.0 ms		1	<input checked="" type="checkbox"/>
stroomp02		No cluster call URL has been set for node: stroomp02		1	<input checked="" type="checkbox"/>

[Stroom UI New Node Management - management tab](#)

To set `stroomp02`'s Cluster URL, move the its line in the display and select it. It will be highlighted.

Name	Cluster URL	Ping	Master	Priority	Enabled
stroomp00	http://stroomp00.strmdev00.org:8080/stroom/clustercall.rpc	1.0 ms	✓	3	✓
stroomp01	http://stroomp01.strmdev00.org:8080/stroom/clustercall.rpc	12 ms		1	✓
stroomp02		No cluster call URL has been set for node: stroomp02		1	✓

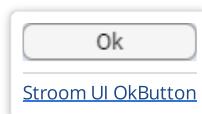
Edit Node

Name: stroomp02
Cluster URL: http://stroomp02.strmdev00.org:8080/stroom/clustercall.rpc

Ok Cancel

[Stroom UI New Node Management - select new node](#)

then press the



[Stroom UI OkButton](#)

button at which we see the *Cluster URL* has been set for the first node as per

Name	Cluster URL	Ping	Master	Priority	Enabled
stroomp00	http://stroomp00.strmdev00.org:8080/stroom/clustercall.rpc	1.0 ms	✓	3	✓
stroomp01	http://stroomp01.strmdev00.org:8080/stroom/clustercall.rpc	300 ms		1	✓
stroomp02	http://stroomp02.strmdev00.org:8080/stroom/clustercall.rpc	No cluster call URL has been set for node: stroomp02		1	✓

[Stroom UI New Node Management - set clustercall url on new node](#)

You need to press the Refresh icon found at top left of Nodes configuration tab, until the new node shows a healthy ping.

Name	Cluster URL	Ping	Master	Priority	Enabled
stroomp00	http://stroomp00.strmdev00.org:8080/stroom/clustercall.rpc	2.0 ms	✓	3	✓
stroomp01	http://stroomp01.strmdev00.org:8080/stroom/clustercall.rpc	7.0 ms		1	✓
stroomp02	http://stroomp02.strmdev00.org:8080/stroom/clustercall.rpc	13 ms		1	✓

[Stroom UI New Node Management - all nodes ping](#)

If you do not get a ping results for the new node, then it is not configured correctly. In that situation, review all log files and processes that you have performed.

Once you have set the Cluster URL you should also set the *master assignment priority* for each node to be different to all of the others. In the image above both stroomp01 and the new node, stroomp02, have been assigned equal priority - 1. We will change stroomp01 to have a different priority - 2. You should note that the node with the highest priority maintains the Master node status.

Name	Cluster URL	Ping	Master	Priority	Enabled
stroomp00	http://stroomp00.strmdev00.org:8080/stroom/clustercall.rpc	4.0 ms	✓	3	✓
stroomp01	http://stroomp01.strmdev00.org:8080/stroom/clustercall.rpc	18 ms		2	✓
stroomp02	http://stroomp02.strmdev00.org:8080/stroom/clustercall.rpc	18 ms		1	✓

[Stroom UI New Node Management - set node priorities](#)

4.1.8 - Processing User setup

This HOWTO demonstrates how to set up various files and scripts that the Stroom processing user requires.

4.1.8.1 Assumptions

- the user has reasonable RHEL/Centos System administration skills
- installation is on a fully patched minimal Centos 7.3 instance.
- the application user `stroomuser` has been created
- the user is deploying for either
- the example two node Stroom cluster whose storage is described [here](#)
- a simple Forwarding or Standalone Proxy
- adding a node to an existing Stroom cluster

4.1.8.2 Set up the Stroom processing user's environment

To automate the running of a Stroom Proxy or Application service under our Stroom processing user, `stroomuser`, there are a number of configuration files and scripts we need to deploy.

We first become the `stroomuser`

```
sudo -i -u stroomuser
```

4.1.8.2.1 Environment Variable files

When either a Stroom Proxy or Application starts, it needs predefined environment variables. We set these up in the `stroomuser` home directory. We need two files for this. The first is for the Stroom processes themselves and the second is for the Stroom systemd service we deploy. The difference is that for the Stroom processes, we need to `export` the environment variables whereas the Stroom systemd service file just needs to read them.

The `JAVA_HOME` and `PATH` variables are to support Java running the Tomcat instances. The `STROOM_TMP` variable is set to a working area for the Stroom Application to use. The application accesses this environment variable internally via the `$(stroom_tmp)` context variable. Note that we only need the `STROOM_TMP` variable for Stroom Application deployments, so one could remove it from the files for a Forwarding or Standalone proxy deployment.

With respect to the working area, we will make use of the [Storage Scenario](#) we have defined and hence use the directory `/stroomdata/stroom-working-p_nn_` where `nn` is the hostname node number (i.e 00 for host `stroomp00`, 01 for host `stroomp01`, etc).

So, for the first node, `00`, we run

```
N=00
F=~/env.sh
printf '# Environment variables for Stroom services\n' > ${F}
printf 'export JAVA_HOME=/usr/lib/jvm/java-1.8.0\n' >> ${F}
printf 'export PATH=${JAVA_HOME}/bin:${PATH}\n' >> ${F}
printf 'export STROOM_TMP=/stroomdata/stroom-working-p%${N}\n' >> ${F}
chmod 640 ${F}

F=~/env_service.sh
printf '# Environment variables for Stroom services, executed out of systemd service\n' > ${F}
printf 'JAVA_HOME=/usr/lib/jvm/java-1.8.0\n' >> ${F}
printf 'PATH=${JAVA_HOME}/bin:${PATH}\n' >> ${F}
printf 'STROOM_TMP=/stroomdata/stroom-working-p%${N}\n' >> ${F}
chmod 640 ${F}
```

then we can change the `N` variable on each successive node and run the above.

Alternately, for a Stroom Forwarding or Standalone proxy, the following would be sufficient

```

F=~/env.sh
printf '# Environment variables for Stroom services\n' > ${F}
printf 'export JAVA_HOME=/usr/lib/jvm/java-1.8.0\n' >> ${F}
printf 'export PATH=${JAVA_HOME}/bin:${PATH}\n' >> ${F}
chmod 640 ${F}

F=~/env_service.sh
printf '# Environment variables for Stroom services, executed out of systemd service\n' > ${F}
printf 'JAVA_HOME=/usr/lib/jvm/java-1.8.0\n' >> ${F}
printf 'PATH=${JAVA_HOME}/bin:${PATH}\n' >> ${F}
chmod 640 ${F}

```

And we integrate the environment into our bash instantiation script as well as setting up useful bash functions. This is the same for all nodes. Note that the `T` and `Tp` aliases are always installed whether they are of use or not. IE a Standalone or Forwarding Stroom Proxy could make no use of the `T` shell alias.

```

F=~/bashrc
printf '. ~/env.sh\n\n' >> ${F}
printf '# Simple functions to support Stroom\n' >> ${F}
printf '# T - continually monitor (tail) the Stroom application log\n' >> ${F}
printf '# Tp - continually monitor (tail) the Stroom proxy log\n' >> ${F}
printf 'function T {\n  tail --follow=name ~/stroom-app/instance/logs/stroom.log\n}\n' >> ${F}
printf 'function Tp {\n  tail --follow=name ~/stroom-proxy/instance/logs/stroom.log\n}\n' >> ${F}

```

And test it has set up correctly

```

. ./bashrc
which java

```

which should return `/usr/lib/jvm/java-1.8.0/bin/java`

4.1.8.2.2 Establish Simple Start/Stop Scripts

We create some simple start/stop scripts that start, or stop, all the available Stroom services. At this point, it's just the Stroom application and proxy.

```

if [ ! -d ~/bin ]; then mkdir ~/bin; fi
F=~/bin/StartServices.sh
#!/bin/bash\n' > ${F}
printf '# Start all Stroom services\n' >> ${F}
printf '# Set list of services\n' >> ${F}
printf 'Services="stroom-proxy stroom-app"\n' >> ${F}
printf 'for service in ${Services}; do\n' >> ${F}
printf '  if [ -f ${service}/bin/start.sh ]; then\n' >> ${F}
printf '    bash ${service}/bin/start.sh\n' >> ${F}
printf '  fi\n' >> ${F}
printf 'done\n' >> ${F}
chmod 750 ${F}

F=~/bin/StopServices.sh
#!/bin/bash\n' > ${F}
printf '# Stop all Stroom services\n' >> ${F}
printf '# Set list of services\n' >> ${F}
printf 'Services="stroom-proxy stroom-app"\n' >> ${F}
printf 'for service in ${Services}; do\n' >> ${F}
printf '  if [ -f ${service}/bin/stop.sh ]; then\n' >> ${F}
printf '    bash ${service}/bin/stop.sh\n' >> ${F}
printf '  fi\n' >> ${F}
printf 'done\n' >> ${F}
chmod 750 ${F}

```

4.1.8.3 Establish and Deploy Systemd services

4.1.8.3.1 Processing or Proxy node

For a standard Stroom Processing or Proxy nodes, we can use the following service script. (Noting this is done as root)

```
sudo bash
F=/etc/systemd/system/stroom-services.service
printf '# Install in /etc/systemd/system\n' > ${F}
printf '# Enable via systemctl enable stroom-services.service\n\n' >> ${F}
printf '[Unit]\n' >> ${F}
printf '# Who we are\n' >> ${F}
printf 'Description=Stroom Service\n' >> ${F}
printf '# We want the network and httpd up before us\n' >> ${F}
printf 'Requires=network-online.target httpd.service\n' >> ${F}
printf 'After= httpd.service network-online.target\n\n' >> ${F}
printf '[Service]\n' >> ${F}
printf '# Source our environment file so the Stroom service start/stop scripts work\n' >> ${F}
printf 'EnvironmentFile=/home/stroomuser/env_service.sh\n' >> ${F}
printf 'Type=oneshot\n' >> ${F}
printf 'ExecStart=/bin/su --login stroomuser /home/stroomuser/bin/StartServices.sh\n' >> ${F}
printf 'ExecStop=/bin/su --login stroomuser /home/stroomuser/bin/StopServices.sh\n' >> ${F}
printf 'RemainAfterExit=yes\n\n' >> ${F}
printf '[Install]\n' >> ${F}
printf 'WantedBy=multi-user.target\n' >> ${F}
chmod 640 ${F}
```

4.1.8.3.2 Single Node Scenario with local database

Should you only have a deployment where the database is on a processing node, use the following service script. The only difference is the Stroom dependency on the database. The database dependency below is for the MariaDB database. If you had installed the MySQL Community database, then replace `mariadb.service` with `mysqld.service`. (Noting this is done as root)

```
sudo bash
F=/etc/systemd/system/stroom-services.service
printf '# Install in /etc/systemd/system\n' > ${F}
printf '# Enable via systemctl enable stroom-services.service\n\n' >> ${F}
printf '[Unit]\n' >> ${F}
printf '# Who we are\n' >> ${F}
printf 'Description=Stroom Service\n' >> ${F}
printf '# We want the network, httpd and Database up before us\n' >> ${F}
printf 'Requires=network-online.target httpd.service mariadb.service\n' >> ${F}
printf 'After=mariadb.service httpd.service network-online.target\n\n' >> ${F}
printf '[Service]\n' >> ${F}
printf '# Source our environment file so the Stroom service start/stop scripts work\n' >> ${F}
printf 'EnvironmentFile=/home/stroomuser/env_service.sh\n' >> ${F}
printf 'Type=oneshot\n' >> ${F}
printf 'ExecStart=/bin/su --login stroomuser /home/stroomuser/bin/StartServices.sh\n' >> ${F}
printf 'ExecStop=/bin/su --login stroomuser /home/stroomuser/bin/StopServices.sh\n' >> ${F}
printf 'RemainAfterExit=yes\n\n' >> ${F}
printf '[Install]\n' >> ${F}
printf 'WantedBy=multi-user.target\n' >> ${F}
chmod 640 ${F}
```

4.1.8.3.3 Enable the service

Now we enable the Stroom service, but we **DO NOT** start it as we will manually start the Stroom services as part of the installation process.

```
systemctl enable stroom-services.service
```

4.1.9 - SSL Certificate Generation

A HOWTO to assist users in setting up various SSL Certificates to support a Web interface to Stroom.

4.1.9.1 Assumptions

The following assumptions are used in this document.

- the user has reasonable RHEL/Centos System administration skills
- installations are on Centos 7.3 minimal systems (fully patched)
- either a Stroom Proxy or Stroom Application has already been deployed
- processing node names are 'stroomp00.strmdev00.org' and 'stroomp01.strmdev00.org'
- the first node, 'stroomp00.strmdev00.org' also has a CNAME 'stroomp.strmdev00.org'
- in the scenario of a Stroom Forwarding Proxy, the node name is 'stroomfp0.strmdev00.org'
- in the scenario of a Stroom Standalone Proxy, the node name is 'stroomsap0.strmdev00.org'
- stroom runs as user 'stroomuser'
- the use of self signed certificates is appropriate for test systems, but users should consider appropriate CA infrastructure in production environments
- in this document, when a screen capture is documented, data entry is identified by the data surrounded by '<>'. This excludes enter/return presses.

4.1.9.2 Create certificates

The first step is to establish a self signed certificate for our Stroom service. If you have a certificate server, then certainly gain an appropriately signed certificate. For this HOWTO, we will stay with a self signed solution and hence no certificate authorities are involved. If you are deploying a cluster, then you will only have one certificate for all nodes. We achieve this by setting up an alias for the first node in the cluster and then use that alias for addressing the cluster. That is, we have set up a CNAME, `stroomp.strmdev00.org` for `stroomp00.strmdev00.org`. This means within the web service we deploy, the ServerName will be `stroomp.strmdev00.org` on each node. Since it's one certificate we only need to set it up on one node then deploy the certificate key files to other nodes.

As the certificates will be stored in the `stroomuser`'s home directory, we become the stroom user

```
sudo -i -u stroomuser
```

4.1.9.2.1 Use host variable

To make things simpler in the following bash extracts, we establish the bash variable `H` to be used in filename generation. The variable name is set to the name of the host (or cluster alias) your are deploying the certificates on. In our multi node HOWTO example we are using, we would use the host CNAME `stroomp`. Thus we execute

```
export H=stroomp
```

Note in our the Stroom Forwarding Proxy HOWTO we would use the name `stroomfp0`. In the case of our Standalone Proxy we would use `stroomsap0`.

We set up a directory to house our certificates via

```
cd ~stroomuser
rm -rf stroom-jks
mkdir -p stroom-jks/stroom-jks/public stroom-jks/stroom-jks/private
cd stroom-jks
```

Create a server key for Stroom service (enter a password when prompted for both initial and verification prompts)

```
openssl genrsa -des3 -out private/$H.key 2048
```

as per

```
Generating RSA private key, 2048 bit long modulus
.....+++  
.....+++  
e is 65537 (0x10001)  
Enter pass phrase for private/stroomp.key: <__ENTER_SERVER_KEY_PASSWORD__>  
Verifying - Enter pass phrase for private/stroomp.key: <__ENTER_SERVER_KEY_PASSWORD__>
```

Create a signing request. The two important prompts are the password and Common Name. All the rest can use the defaults offered. The requested password is for the server key and you should use the host (or cluster alias) your are deploying the certificates on for the Common Name. In the output below we will assume a multi node cluster certificate is being generated, so will use `stroomp.strmdev00.org`.

```
openssl req -sha256 -new -key private/$H.key -out $H.csr
```

as per

```
Enter pass phrase for private/stroomp.key: <__ENTER_SERVER_KEY_PASSWORD__>  
You are about to be asked to enter information that will be incorporated  
into your certificate request.  
What you are about to enter is what is called a Distinguished Name or a DN.  
There are quite a few fields but you can leave some blank  
For some fields there will be a default value,  
If you enter '.', the field will be left blank.  
----  
Country Name (2 letter code) [XX]:  
State or Province Name (full name) []:  
Locality Name (eg, city) [Default City]:  
Organization Name (eg, company) [Default Company Ltd]:  
Organizational Unit Name (eg, section) []:  
Common Name (eg, your name or your server's hostname) []:<__ stroomp.strmdev00.org __>  
Email Address []:  
  
Please enter the following 'extra' attributes  
to be sent with your certificate request  
A challenge password []:  
An optional company name []:
```

We now self sign the certificate (again enter the server key password)

```
openssl x509 -req -sha256 -days 720 -in $H.csr -signkey private/$H.key -out public/$H.crt
```

as per

```
Signature ok  
subject=/C=XX/L=Default City/O=Default Company Ltd/CN=stroomp.strmdev00.org  
Getting Private key  
Enter pass phrase for private/stroomp.key: <__ENTER_SERVER_KEY_PASSWORD__>
```

and noting the subject will change depending on the host name used when generating the signing request.

Create insecure version of private key for Apache autoboot (you will again need to enter the server key password)

```
openssl rsa -in private/$H.key -out private/$H.key.insecure
```

as per

```
Enter pass phrase for private/stroomp.key: <__ENTER_SERVER_KEY_PASSWORD__>  
writing RSA key
```

and then move the insecure keys as appropriate

```
mv private/$H.key private/$H.key.secure  
chmod 600 private/$H.key.secure  
mv private/$H.key.insecure private/$H.key
```

We have now completed the creation of our certificates and keys.

4.1.9.2.2 Replication of Keys Directory to other nodes

If you are deploying a multi node Stroom cluster, then you would replicate the directory `~stroomuser/stroom-jks` to each node in the cluster. That is, tar it up, copy the tar file to the other node(s) then untar it. We can make use of the other node's mounted file system for this process. That is one could execute the commands on the first node, where we created the certificates

```
cd ~stroomuser  
tar cf stroom-jks.tar stroom-jks  
mv stroom-jks.tar /stroomdata/stroom-data-p01
```

then on the another node, say `stroomp01.strmdev00.org`, as the `stroomuser` we extract the data.

```
sudo -i -u stroomuser  
cd ~stroomuser  
tar xf /stroomdata/stroom-data-p01/stroom-jks.tar && rm -f /stroomdata/stroom-data-p01/stroom-jks.tar
```

4.1.9.2.3 Protection, Ownership and SELinux Context

Now ensure protection, ownership and SELinux context for these key files on **ALL** nodes via

```
chmod 700 ~stroomuser/stroom-jks/private ~stroomuser/stroom-jks  
chown -R stroomuser:stroomuser ~stroomuser/stroom-jks  
chcon -R --reference /etc/pki ~stroomuser/stroom-jks
```

4.1.9.3 Stroom Proxy to Proxy Key and Trust Stores

In order for a Stroom Forwarding Proxy to communicate to a central Stroom proxy over https, the JVM running the forwarding proxy needs relevant keystores set up.

One would set up a Stroom's forwarding proxy SSL certificate as per [above](#), with the change that the hostname would be different. That is, in the initial setup, we would set the hostname variable `H` to be the hostname of the forwarding proxy. Lets say it is `stroompfp0` thus we would set

```
export H=stroompfp0
```

and then proceed as [above](#).

Note that you also need the public key of the central Stroom server you will be connecting to. For the following, we will assume the central Stroom proxy is the `stroomp.strmdev00.org` server and its public key is stored in the file `stroomp.crt`. We will store this file on the forwarding proxy in `~stroomuser/stroom-jks/public/stroomp.crt`.

So once you have created the forwarding proxy server's SSL keys and have deployed the central proxy's public key, we next need to convert the proxy server's SSL keys into DER format. This is done by executing the following.

```
cd ~stroomuser/stroom-jks  
export H=stroompfp0  
export S=stroomp  
rm -f ${H}_k.jks ${S}_t.jks  
H_k=${H}  
S_k=${S}  
# Convert public key  
openssl x509 -in public/$H.crt -inform PEM -out public/$H.crt.der -outform DER
```

When you convert the local server's private key, you will be prompted for the server key password.

```
# Convert the local server's Private key
openssl pkcs8 -topk8 -nocrypt -in private/$H.key.secure -inform PEM -out private/$H.key.der -outform DER
```

as per

```
Enter pass phrase for private/stroomfp0.key.secure: <__ENTER_SERVER_KEY_PASSWORD__>
```

We now import these keys into our Key Store. As part of the Stroom Proxy release, an Import Keystore application has been provisioned. We identify where it's found with the command

```
find ~stroomuser/*proxy -name 'stroom*util*.jar' -print | head -1
```

which should return `/home/stroomuser/stroom-proxy/lib/stroom-proxy-util-v5.1-beta.10.jar` or similar depending on the release version. To make execution simpler, we set this as a shell variable as per

```
Stroom_UTIL_JAR=`find ~/*proxy -name 'stroom*util*.jar' -print | head -1`
```

We now create the keystore and import the proxy's server key

```
java -cp ${Stroom_UTIL_JAR} stroom.util.cert.ImportKey keystore=${H}_k.jks keypass=$H alias=$H keyfile=private/$H.key.der certfi
```

as per

```
One certificate, no chain
```

We now import the destination server's public key

```
keytool -import -noprompt -alias ${S} -file public/${S}.crt -keystore ${S}_k.jks -storepass ${S}
```

as per

```
Certificate was added to keystore
```

We now add the key and trust store location and password arguments to our Stroom proxy environment files.

```
PWD=`pwd`
echo "export JAVA_OPTS=\"-Djavax.net.ssl.trustStore=${PWD}/${S}_k.jks -Djavax.net.ssl.trustStorePassword=${S} -Djavax.net.ssl.keyStore=${PWD}/${S}_k.jks -Djavax.net.ssl.keyStorePassword=${S}" > ./env.sh
```

At this point you should restart the proxy service. Using the commands

```
cd ~stroomuser
source ./env.sh
stroom-proxy/bin/stop.sh
stroom-proxy/bin/start.sh
```

then check the logs to ensure it started correctly.

4.1.10 - Testing Stroom Installation

This HOWTO will demonstrate various ways to test that your Stroom installation has been successful.

4.1.10.1 Assumptions

- Stroom Single or Multi Node Cluster Testing
- the [Multi Node Stroom Cluster \(Proxy and Application\)](#) has been deployed
- a [Test Feed](#), TEST-FEED-V1_0 has been added
- Proxy aggregation has been turned off on all Stroom *Store* Proxies
- the Stroom Proxy Repository Format (REPO_FORMAT) chosen was the default - \${pathId}/\${id}
- Stroom Forwarding Proxy Testing
- the [Multi Node Stroom Cluster \(Proxy and Application\)](#) has been deployed
- the [Stroom Forwarding Proxy](#) has been deployed
- a [Test Feed](#), TEST-FEED-V1_0 has been added
- the Stroom Proxy Repository Format (REPO_FORMAT) chosen was the default - \${pathId}/\${id}
- Stroom Standalone Proxy Testing
- the [Stroom Standalone Proxy](#) has been deployed
- the Stroom Proxy Repository Format (REPO_FORMAT) chosen was the default - \${pathId}/\${id}

4.1.10.2 Stroom Single or Multi Node Cluster Testing

4.1.10.2.1 Data Post Tests

4.1.10.2.1.1 Simple Post tests

These tests are to ensure the Stroom *Store* proxy and it's connection to the database is working along with the Apache mod_jk loadbalancer. We will send a file to the load balanced `stroomb0.strmdev00.org` node (really `stroomp00.strmdev00.org`) and each time we send the file, it's receipt should be managed by alternate proxy nodes. As a number of elements can effect load balancing, it is not always guaranteed to alternate every time but for the most part it will.

Perform the following

- Log onto the Stroom database node (`stroomb0.strmdev00.org`) as any user.
- Log onto both Stroom nodes and become the `stroomuser` and monitor each node's Stroom proxy service using the `Tp` bash macro. That is, on each node, run

```
sudo -i -u stroomuser
Tp
```

You will note events of the form from `stroomp00.strmdev00.org`:

```
...
2017-01-14T06:22:26.672Z INFO [ProxyProperties refresh thread 0] datafeed.ProxyHandlerFactory$1 (ProxyHandlerFactory.java:96) -
2017-01-14T06:30:00.993Z INFO [Repository Reader Thread 1] handler.ProxyRepositoryReader (ProxyRepositoryReader.java:143) - rur
2017-01-14T06:40:00.245Z INFO [Repository Reader Thread 1] handler.ProxyRepositoryReader (ProxyRepositoryReader.java:143) - rur
```

and from `stroomp01.strmdev00.org`:

```
...
2017-01-14T06:22:26.828Z INFO [ProxyProperties refresh thread 0] datafeed.ProxyHandlerFactory$1 (ProxyHandlerFactory.java:96) -
2017-01-14T06:30:00.066Z INFO [Repository Reader Thread 1] handler.ProxyRepositoryReader (ProxyRepositoryReader.java:143) - rur
2017-01-14T06:40:00.318Z INFO [Repository Reader Thread 1] handler.ProxyRepositoryReader (ProxyRepositoryReader.java:143) - rur
```

- On the Stroom database node, execute the command

```
curl -k --data-binary @/etc/group "https://stroomp.strmdev00.org/stroom/datafeed" -H "Feed:TEST-FEED-V1_0" -H "System:EXAMPLE_SY
```

If you are monitoring the proxy log of `stroomp00.strmdev00.org` you would see two new logs indicating the successful arrival of the file

```
2017-01-14T06:46:06.411Z INFO [ajp-apr-9009-exec-1] handler.LogRequestHandler (LogRequestHandler.java:37) - log() - guid=54dc00  
2017-01-14T06:46:06.449Z INFO [ajp-apr-9009-exec-1] datafeed.DataFeedRequestHandler$1 (DataFeedRequestHandler.java:104) - "doPo
```

- On the Stroom database node, again execute the command

```
curl -k --data-binary @/etc/group "https://stroomp.strmdev00.org/stroom/datafeed" -H "Feed:TEST-FEED-V1_0" -H "System:EXAMPLE_SY
```

If you are monitoring the proxy log of `stroomp01.strmdev00.org` you should see a new log. As foreshadowed, we didn't as the time delay resulted in the first node getting the file. That is `stroomp00.strmdev00.org` log file gained the two entries

```
2017-01-14T06:47:26.642Z INFO [ajp-apr-9009-exec-2] handler.LogRequestHandler (LogRequestHandler.java:37) - log() - guid=941d29  
2017-01-14T06:47:26.645Z INFO [ajp-apr-9009-exec-2] datafeed.DataFeedRequestHandler$1 (DataFeedRequestHandler.java:104) - "doPo
```

- Again on the database node, execute the command and this time we see that node `stroomp01.strmdev00.org` received the file as per

```
2017-01-14T06:47:30.782Z INFO [ajp-apr-9009-exec-1] handler.LogRequestHandler (LogRequestHandler.java:37) - log() - guid=2cef6e  
2017-01-14T06:47:30.816Z INFO [ajp-apr-9009-exec-1] datafeed.DataFeedRequestHandler$1 (DataFeedRequestHandler.java:104) - "doPo
```

- Running the curl post command in quick succession shows the loadbalancer working ... four executions result in seeing our pair of logs appearing on alternate proxies.

`stroomp00` :

```
2017-01-14T06:52:09.815Z INFO [ajp-apr-9009-exec-3] handler.LogRequestHandler (LogRequestHandler.java:37) - log() - guid=bf0bc3  
2017-01-14T06:52:09.817Z INFO [ajp-apr-9009-exec-3] datafeed.DataFeedRequestHandler$1 (DataFeedRequestHandler.java:104) - "doPo
```

`stroomp01` :

```
2017-01-14T06:52:11.139Z INFO [ajp-apr-9009-exec-2] handler.LogRequestHandler (LogRequestHandler.java:37) - log() - guid=1088fc  
2017-01-14T06:52:11.150Z INFO [ajp-apr-9009-exec-2] datafeed.DataFeedRequestHandler$1 (DataFeedRequestHandler.java:104) - "doPo
```

`stroomp00` :

```
2017-01-14T06:52:12.284Z INFO [ajp-apr-9009-exec-4] handler.LogRequestHandler (LogRequestHandler.java:37) - log() - guid=def94a  
2017-01-14T06:52:12.289Z INFO [ajp-apr-9009-exec-4] datafeed.DataFeedRequestHandler$1 (DataFeedRequestHandler.java:104) - "doPo
```

`stroomp01` :

```
2017-01-14T06:52:13.374Z INFO [ajp-apr-9009-exec-3] handler.LogRequestHandler (LogRequestHandler.java:37) - log() - guid=55ddaa  
2017-01-14T06:52:13.378Z INFO [ajp-apr-9009-exec-3] datafeed.DataFeedRequestHandler$1 (DataFeedRequestHandler.java:104) - "doPo
```

At this point we will see what the proxies have received.

- On each node run the command

```
ls -l /stroomdata/stroom-working*/proxy
```

On `stroomp00` we see

```
[stroomuser@stroomp00 ~]$ ls -l /stroomdata/stroom-working*/proxy
total 16
-rw-rw-r--. 1 stroomuser stroomuser 785 Jan 14 17:46 001.zip
-rw-rw-r--. 1 stroomuser stroomuser 783 Jan 14 17:47 002.zip
-rw-rw-r--. 1 stroomuser stroomuser 784 Jan 14 17:52 003.zip
-rw-rw-r--. 1 stroomuser stroomuser 783 Jan 14 17:52 004.zip
[stroomuser@stroomp00 ~]$
```

and on stroomp01 we see

```
[stroomuser@stroomp01 ~]$ ls -l /stroomdata/stroom-working*/proxy
total 12
-rw-rw-r--. 1 stroomuser stroomuser 785 Jan 14 17:47 001.zip
-rw-rw-r--. 1 stroomuser stroomuser 783 Jan 14 17:52 002.zip
-rw-rw-r--. 1 stroomuser stroomuser 784 Jan 14 17:52 003.zip
[stroomuser@stroomp01 ~]$
```

which corresponds to the seven posts of data and the associated events in the proxy logs. To see the contents of one of these files we execute on either node, the command

```
unzip -c /stroomdata/stroom-working*/proxy/001.zip
```

to see

```

Archive: /stroomdata/stroom-working-p00/proxy/001.zip
inflating: 001.dat
root:x:0:
bin:x:1:
daemon:x:2:
sys:x:3:
adm:x:4:
tty:x:5:
disk:x:6:
lp:x:7:
mem:x:8:
kmem:x:9:
wheel:x:10:burn
cdrom:x:11:
mail:x:12:postfix
man:x:15:
dialout:x:18:
floppy:x:19:
games:x:20:
tape:x:30:
video:x:39:
ftp:x:50:
lock:x:54:
audio:x:63:
nobody:x:99:
users:x:100:
utmp:x:22:
utempter:x:35:
input:x:999:
systemd-journal:x:190:
systemd-bus-proxy:x:998:
systemd-network:x:192:
dbus:x:81:
polkitd:x:997:
ssh_keys:x:996:
dip:x:40:
tss:x:59:
sshd:x:74:
postdrop:x:90:
postfix:x:89:
chrony:x:995:
burn:x:1000:burn
mysql:x:27:

    inflating: 001.meta
content-type:application/x-www-form-urlencoded
Environment:EXAMPLE_ENVIRONMENT
Feed:TEST-FEED-V1_0
GUID:54dc0da2-f35c-4dc2-8a98-448415ffc76b
host:stroomp.strmdev00.org
ReceivedTime:2017-01-14T06:46:05.883Z
RemoteAddress:192.168.2.144
RemoteHost:192.168.2.144
StreamSize:527
System:EXAMPLE_SYSTEM
user-agent:curl/7.29.0

[stroomuser@stroomp00 ~]$
```

Checking the /etc/group file on `stroomdb0.strmdev00.org` confirms the above contents. For the present, ignore the metadata file present in the zip archive.

If you execute the same command on the other files, all that changes is the value of the `ReceivedTime:` attribute in the `.meta` file.

For those curious about the file size differences, this is a function of the compression process within the proxy. Using `stroomp01`'s files and extracting them manually and renaming them results in the six files

```
[stroombuser@stroomp01 xx]$ ls -l
total 24
-rw-rw-r--. 1 stroombuser stroombuser 527 Jan 14 17:47 A_001.dat
-rw-rw-r--. 1 stroombuser stroombuser 321 Jan 14 17:47 A_001.meta
-rw-rw-r--. 1 stroombuser stroombuser 527 Jan 14 17:52 B_001.dat
-rw-rw-r--. 1 stroombuser stroombuser 321 Jan 14 17:52 B_001.meta
-rw-rw-r--. 1 stroombuser stroombuser 527 Jan 14 17:52 C_001.dat
-rw-rw-r--. 1 stroombuser stroombuser 321 Jan 14 17:52 C_001.meta
[stroombuser@stroomp01 xx]$ cmp A_001.dat B_001.dat
[stroombuser@stroomp01 xx]$ cmp B_001.dat C_001.dat
[stroombuser@stroomp01 xx]$
```

We have effectively tested the receipt of our data and the load balancing of the Apache mod_jk installation.

4.1.10.2.1.2 Simple Direct Post tests

In this test we will use the direct feed interface of the Stroom application, rather than sending data via the proxy. One would normally use this interface for time sensitive data which shouldn't aggregate in a proxy waiting for the Stroom application to collect it. In this situation we use the command

```
curl -k --data-binary @/etc/group "https://stroomp.strmdev00.org/stroom/datafeed/direct" -H "Feed:TEST-FEED-V1_0" -H "System:EXA
```

To prepare for this test, we monitor the Stroom application log using the `tail` bash alias on each node. So on each node run the command

```
sudo -i -u stroombuser
tail
```

On each node you should see *LifecycleTask* events, for example,

```
2017-01-14T07:42:08.281Z INFO [Stroom P2 #7 - LifecycleTask] spring.StroomBeanMethodExecutable (StroomBeanMethodExecutable.java:20)
2017-01-14T07:42:18.284Z INFO [Stroom P2 #2 - LifecycleTask] spring.StroomBeanMethodExecutable (StroomBeanMethodExecutable.java:20)
2017-01-14T07:42:18.284Z INFO [Stroom P2 #10 - LifecycleTask] spring.StroomBeanMethodExecutable (StroomBeanMethodExecutable.java:20)
2017-01-14T07:42:18.285Z INFO [Stroom P2 #7 - LifecycleTask] spring.StroomBeanMethodExecutable (StroomBeanMethodExecutable.java:20)
```

To perform the test, on the database node, run the posting command a number of times in rapid succession. This will result in `server.DataFeedServiceImpl` events in both log files. The Stroom application log is quite busy, you may have to look for these logs.

In the following we needed to execute the posting command three times before seeing the data arrive on both nodes. Looking at the arrival times, the file turned up on the second node twice before appearing on the first node. `stroomp00`:

```
2017-01-14T07:43:09.394Z INFO [ajp-apr-8009-exec-6] server.DataFeedServiceImpl (DataFeedServiceImpl.java:133) - handleRequest r
```

and on `stroomp01`:

```
2017-01-14T07:43:05.614Z INFO [ajp-apr-8009-exec-1] server.DataFeedServiceImpl (DataFeedServiceImpl.java:133) - handleRequest r
2017-01-14T07:43:06.821Z INFO [ajp-apr-8009-exec-2] server.DataFeedServiceImpl (DataFeedServiceImpl.java:133) - handleRequest r
```

To confirm this data arrived, we need to view the **Data** pane of our `TEST-FEED-V1_0` tab. To do this, log onto the Stroom UI then move the cursor to the `TEST-FEED-V1_0` entry in the `Explorer` tab and select the item with a left click

The screenshot shows the Stroom UI Jobs page. The left sidebar has 'System' expanded, showing 'Internal Statistics' and 'TEST-FEED-V1_0'. The main area displays a table of jobs:

Job	Enabled	Description
Disable Unused Accounts	<input checked="" type="checkbox"/>	Job to disable unused accounts
File System Clean	<input checked="" type="checkbox"/>	Job to process a volume deleting files that are no longer indexed (maybe the rete)
Node Status	<input checked="" type="checkbox"/>	Job to record status of node (CPU and Memory usage)
Proxy Aggregation	<input type="checkbox"/>	Job to pick up the data written by the proxy and store it in Stroom
Stream Processor	<input checked="" type="checkbox"/>	.Job to process streams matching stream processor filters with their associated n

Below the table is a link: [Stroom UI Test Feed - Open Feed](#).

and double click on the entry to see our TEST-FEED-V1_0 tab.

The screenshot shows the Stroom UI Settings page for the TEST-FEED-V1_0 feed. The left sidebar has 'System' expanded, showing 'Internal Statistics' and 'TEST-FEED-V1_0'. The main area shows the feed's attributes:

- Description: Feed for installation validation only. No data value.
- Classification: (empty)
- Reference Feed: (empty)
- Feed Status: Receive
- Stream Type: Raw Events
- Data Encoding: UTF-8
- Context Encoding: UTF-8
- Retention Period: Forever

Below the form is a link: [Stroom UI Test Feed - Opened Feed](#).

and it is noted that we are viewing the Feed's attributes as we can see the **Setting** hyper-link highlighted. As we want to see the Data we have received for this feed, move the cursor to the **Data** hyper-link and select it to see

The screenshot shows the Stroom UI Data view for the TEST-FEED-V1_0 feed. The left sidebar has 'System' expanded, showing 'Internal Statistics' and 'TEST-FEED-V1_0'. The main area displays a table of received data entries:

Created	Effective	Feed	Type	StreamSize	FileSize	RecRead	RecWrite	RecInfo	RecWarn	RecError	RecFatal
2017-01-14T07:43:08.945Z	2017-01-14T07:43:08.945Z	TEST-FEED-V1_0	Raw Events	527 bytes	383 bytes				1 to 3 of 3	RecWarn	RecError
2017-01-14T07:43:06.574Z	2017-01-14T07:43:06.574Z	TEST-FEED-V1_0	Raw Events	527 bytes	383 bytes						
2017-01-14T07:43:05.425Z	2017-01-14T07:43:05.425Z	TEST-FEED-V1_0	Raw Events	527 bytes	383 bytes						

Below the table are two smaller tables:

Type	Feed	Pipeline	RecRead	RecWrite	RecInfo	RecWarn	RecError	RecFatal

StreamSize	FileSize

Below the tables is a link: [Stroom UI Test Feed - Opened Feed view Data](#).

These three entries correspond to the three posts we performed.

We have successfully tested direct posting to a Stroom feed and that the Apache mod_jk loadbalancer also works for this posting method.

4.1.10.2.1.3 Test Proxy Aggregation is Working

To test that the Proxy Aggregation is working, we need to [enable](#) on each node.

By enabling the Proxy Aggregation process, both nodes immediately performed the task as indicated by each node's Stroom application logs as per `stroomp00`:

```
2017-01-14T07:58:58.752Z INFO [Stroom P2 #3 - LifecycleTask] server.ProxyAggregationExecutor (ProxyAggregationExecutor.java:138)
2017-01-14T07:58:58.937Z INFO [Stroom P2 #2 - GenericServerTask] server.ProxyAggregationExecutor$2 (ProxyAggregationExecutor.java:138)
2017-01-14T07:58:59.045Z INFO [Stroom P2 #2 - GenericServerTask] server.ProxyAggregationExecutor$2 (ProxyAggregationExecutor.java:138)
2017-01-14T07:58:59.101Z INFO [Stroom P2 #3 - LifecycleTask] server.ProxyAggregationExecutor (ProxyAggregationExecutor.java:152)
```

and stroom01:

```
2017-01-14T07:59:16.687Z INFO [Stroom P2 #10 - LifecycleTask] server.ProxyAggregationExecutor (ProxyAggregationExecutor.java:13)
2017-01-14T07:59:16.799Z INFO [Stroom P2 #5 - GenericServerTask] server.ProxyAggregationExecutor$2 (ProxyAggregationExecutor.java:13)
2017-01-14T07:59:16.909Z INFO [Stroom P2 #5 - GenericServerTask] server.ProxyAggregationExecutor$2 (ProxyAggregationExecutor.java:13)
2017-01-14T07:59:16.997Z INFO [Stroom P2 #10 - LifecycleTask] server.ProxyAggregationExecutor (ProxyAggregationExecutor.java:13)
```

And on refreshing the top pane of the TEST-FEED-V1_0 tab we see that two more batches of data have arrived.

The screenshot shows the Stroom UI interface with the 'TEST-FEED-V1_0' tab selected. The main area displays a table of data batches. The columns are: Created, Effective, Feed, Type, StreamSize, FileSize, RecRead, RecWrite, RecInfo, RecWarn, and RecErr. There are five rows of data, each with a checkbox and a timestamp indicating when the data was created and effective, the feed name, the type (Raw Events), and the file size.

Created	Effective	Feed	Type	StreamSize	FileSize	RecRead	RecWrite	RecInfo	1 to 5 of 5	RecWarn	RecErr
2017-01-14T07:59:16.800Z	2017-01-14T07:59:16.800Z	TEST-FEED-V1_0	Raw Events	1.5 kB	401 bytes						
2017-01-14T07:58:58.939Z	2017-01-14T07:58:58.939Z	TEST-FEED-V1_0	Raw Events	2.0 kB	405 bytes						
2017-01-14T07:43:08.945Z	2017-01-14T07:43:08.945Z	TEST-FEED-V1_0	Raw Events	527 bytes	383 bytes						
2017-01-14T07:43:06.574Z	2017-01-14T07:43:06.574Z	TEST-FEED-V1_0	Raw Events	527 bytes	383 bytes						
2017-01-14T07:43:05.425Z	2017-01-14T07:43:05.425Z	TEST-FEED-V1_0	Raw Events	527 bytes	383 bytes						

Stroom UI Test Feed - Proxy Aggregated data arrival

This demonstrates that Proxy Aggregation is working.

4.1.10.3 Stroom Forwarding Proxy Testing

4.1.10.3.1 Data Post Tests

4.1.10.3.1.1 Simple Post tests

This test is to ensure the Stroom *Forwarding* proxy and it's connection to the central Stroom Processing system is working.

We will send a file to our *Forwarding* proxy (`stroomfp0.strmdev00.org`) and monitor this nodes' proxy log files as well as all the destination nodes proxy log files. The reason for monitoring all the destination system's proxy log files is that the destination system is probably load balancing and hence the forwarded file may turn up on any of the destination nodes.

Perform the following

- Log onto any host where you will perform the `curl` post
- Monitor all proxy log files
- Log onto the Forwarding Proxy node and become the `stroomuser` and monitor the Stroom proxy service using the `Tp` bash macro.
- Log onto the destination Stroom nodes and become the `stroomuser` and monitor each node's Stroom proxy service using the `Tp` bash macro. That is, on each node, run

```
sudo -i -u stroomuser
Tp
```

- On the 'posting' node, run the command

```
curl -k --data-binary @/etc/group "https://stroomfp0.strmdev00.org/stroom/datafeed" -H "Feed:TEST-FEED-V1_0" -H "System:EXAMPLE..."
```

In the Stroom Forwarding proxy log, `~/stroom-proxy/instance/logs/stroom.log`, you will see the arrival of the file as per the `datafeed.DataFeedRequestHandler$1` event running under, in this case, the `ajp-apr-9009-exec-1` thread.

```
...
2017-01-01T23:17:00.240Z INFO [Repository Reader Thread 1] handler.ProxyRepositoryReader (ProxyRepositoryReader.java:143) - rur
2017-01-01T23:18:00.275Z INFO [Repository Reader Thread 1] handler.ProxyRepositoryReader (ProxyRepositoryReader.java:143) - rur
2017-01-01T23:18:12.367Z INFO [ajp-apr-9009-exec-1] datafeed.DataFeedRequestHandler$1 (DataFeedRequestHandler.java:104) - "doPo...
```

And then at the next periodic interval (60 second intervals) this file will be forwarded to the main stroom proxy server `stroomp.strmdev00.org` as shown by the `handler.ForwardRequestHandler` events running under the `pool-10-thread-2` thread.

```
2017-01-01T23:19:00.304Z INFO [Repository Reader Thread 1] handler.ProxyRepositoryReader (ProxyRepositoryReader.java:143) - rur
2017-01-01T23:19:00.586Z INFO [pool-10-thread-2] handler.ForwardRequestHandler (ForwardRequestHandler.java:109) - handleHeader()
2017-01-01T23:19:00.990Z INFO [pool-10-thread-2] handler.ForwardRequestHandler (ForwardRequestHandler.java:89) - handleFooter()
2017-01-01T23:20:00.064Z INFO [Repository Reader Thread 1] handler.ProxyRepositoryReader (ProxyRepositoryReader.java:143) - rur
...
...
```

On one of the central processing nodes, when the file is send by the Forwarding Proxy, you will see the file's arrival as per the `datafeed.DataFeedRequestHandler$1` event in the `ajp-apr-9009-exec-3` thread.

```
...
2017-01-01T23:00:00.236Z INFO [Repository Reader Thread 1] handler.ProxyRepositoryReader (ProxyRepositoryReader.java:143) - rur
2017-01-01T23:10:00.473Z INFO [Repository Reader Thread 1] handler.ProxyRepositoryReader (ProxyRepositoryReader.java:143) - rur
2017-01-01T23:19:00.787Z INFO [ajp-apr-9009-exec-3] handler.LogRequestHandler (LogRequestHandler.java:37) - log() - guid=b5722e
2017-01-01T23:19:00.981Z INFO [ajp-apr-9009-exec-3] datafeed.DataFeedRequestHandler$1 (DataFeedRequestHandler.java:104) - "doPoC"
2017-01-01T23:20:00.771Z INFO [Repository Reader Thread 1] handler.ProxyRepositoryReader (ProxyRepositoryReader.java:143) - rur
...
...
```

4.1.10.4 Stroom Standalone Proxy Testing

4.1.10.4.1 Data Post Tests

4.1.10.4.1.1 Simple Post tests

This test is to ensure the Stroom *Store NODB* or *Standalone* proxy is working.

We will send a file to our *Standalone* proxy (`stroomsap0.strmdev00.org`) and monitor this nodes' proxy log files as well the directory the received files are meant to be stored in.

Perform the following

- Log onto any host where you will perform the `curl` post
- Log onto the Standalone Proxy node and become the `stroomuser` and monitor the Stroom proxy service using the `Tp` bash macro. That is run

```
sudo -i -u stroomuser
Tp
```

- On the 'posting' node, run the command

```
curl -k --data-binary @/etc/group "https://stroomsap0.strmdev00.org/stroom/datafeed" -H "Feed:TEST-FEED-V1_0" -H "System:EXAMPLE"
```

In the stroom proxy log, `~/stroom-proxy/instance/logs/stroom.log`, you will see the arrival of the file via both the `handler.LogRequestHandler` and `datafeed.DataFeedRequestHandler$1` events running under, in this case, the `ajp-apr-9009-exec-1` thread.

```
...
2017-01-02T02:10:00.325Z INFO [Repository Reader Thread 1] handler.ProxyRepositoryReader (ProxyRepositoryReader.java:143) - rur
2017-01-02T02:11:34.501Z INFO [ajp-apr-9009-exec-1] handler.LogRequestHandler (LogRequestHandler.java:37) - log() - guid=ebd112
2017-01-02T02:11:34.528Z INFO [ajp-apr-9009-exec-1] datafeed.DataFeedRequestHandler$1 (DataFeedRequestHandler.java:104) - "doPoC"
...
...
```

Further, if you check the proxy's storage directory, you will see the file `001.zip`. The file names number upwards from 001.

```
ls -l /stroomdata/stroom-working-sap0/proxy
```

shows

```
[stroomuser@stroomb0 ~]$ ls -l /stroomdata/stroom-working-sap0/proxy  
total 4  
-rw-rw-r--. 1 stroomuser stroomuser 1107 Jan  2 13:11 001.zip  
[stroomuser@stroomb0 ~]$
```

On viewing the contents of this file we see both a .dat and .meta file.

```
[stroomuser@stroomb0 ~]$ (cd /stroomdata/stroom-working-sap0/proxy; unzip 001.zip)  
Archive: 001.zip  
  inflating: 001.dat  
  inflating: 001.meta  
[stroomuser@stroomb0 ~]$
```

The .dat file holds the content of the file we posted - /etc/group .

```
[stroomuser@stroomb0 ~]$ (cd /stroomdata/stroom-working-sap0/proxy; head -5 001.dat)  
root:x:0:  
bin:x:1:bin,daemon  
daemon:x:2:bin,daemon  
sys:x:3:bin,adm  
adm:x:4:adm,daemon  
[stroomuser@stroomb0 ~]$
```

The .meta file is generated by the proxy and holds information about the posted file

```
[stroomuser@stroomb0 ~]$ (cd /stroomdata/stroom-working-sap0/proxy; cat 001.meta)  
content-type:application/x-www-form-urlencoded  
Environment:EXAMPLE_ENVIRONMENT  
Feed:TEST-FEED-V1_0  
GUID:ebd11215-7d4c-4be6-a524-358015e2ac38  
host:stroomb0.strmdev00.org  
ReceivedTime:2017-01-02T02:11:34.501Z  
RemoteAddress:192.168.2.220  
RemoteHost:192.168.2.220  
StreamSize:1051  
System:EXAMPLE_SYSTEM  
user-agent:curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.21 Basic ECC zlib/1.2.3 libidn/1.18 libssh2/1.4.2  
[stroomuser@stroomb0 ~]$ (cd /stroomdata/stroom-working-sap0/proxy; rm 001.meta 001.dat)  
[stroomuser@stroomb0 ~]$
```

4.1.11 - Volume Maintenance

How to maintain Stroom's data and index volumes.

Stroom stores data in [volumes](#). These are the logical link to the Storage hierarchy we setup on the operating system. This HOWTO will demonstrate how one first sets up volumes and also how to add additional volumes if one expanded an existing Stroom cluster.

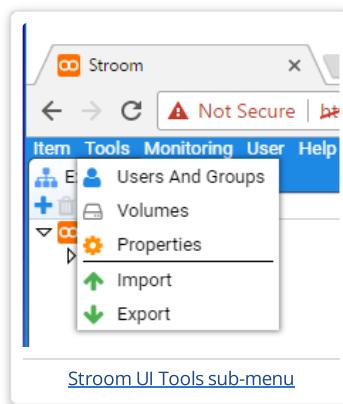
4.1.11.1 Assumptions

- an account with the [Administrator Application Permission](#) is currently logged in.
- we will add volumes as per the Multi Node Stroom deployment Storage hierarchy

4.1.11.2 Configure the Volumes

We need to configure the volumes for Stroom. The follow demonstrates adding the volumes for two nodes, but demonstrates the process for a single node deployment as well the volume maintenance needed when expanding a Multi Node Cluster when adding in a new node.

To configure the volumes, move to the `Tools` item of the **Main Menu** and select it to bring up the `Tools` sub-menu.



then move down and select the `Volumes` sub-item to be presented with the `Volumes` configuration window as seen below.

Volumes										
				Volume Type	Stream Status	Index Status	Usage Date	Limit	Used	Free
Node	Path									
stroomp00	/home/stroomuser/stroom-app/volumes/defaultIndexVolume			Private	Inactive	Active	2017-02-02T08:34:52.591Z	12 GB	2.2 GB	11 GB
stroomp00	/home/stroomuser/stroom-app/volumes/defaultStreamVolume			Public	Active	Inactive	2017-02-02T08:34:52.635Z	12 GB	2.2 GB	11 GB

Stroom UI Volumes - configuration window

The attributes we see for each volume are

- Node - the processing node the volume resides on (this is just the node name entered when configuration the Stroom application)
- Path - the path to the volume
- Volume Type - The type of volume
- Public - to indicate that all nodes would access this volume
- Private - to indicate that only the local node will access this volume
- Stream Status
- Active - to store data within the volume
- Inactive - to **NOT** store data within the volume
- Closed - had stored data within the volume, but now no more data can be stored
- Index Status
- Active - to store index data within the volume
- Inactive - to **NOT** store index data within the volume
- Closed - had stored index data within the volume, but now no more index data can be stored
- Usage Date - the date and time the volume was last used
- Limit - the maximum amount of data the system will store on the volume
- Used - the amount of data in use on the volume

- Free - the amount of available storage on the volume
- Use% - the usage percentage

If you are setting up Stroom for the first time and you had accepted the default for the **CREATE_DEFAULT_VOLUME_ON_START** configuration option (*true*) when configuring the Stroom service application, you will see two default volumes have already been created. Had you set this option to *false* then the window would be empty.

4.1.11.2.0.1 Add Volumes

Now from our two node Stroom Cluster example, our storage hierarchy was

- Node: `stroomp00.strmdev00.org`
- `/stroomdata/stroom-data-p00` - location to store Stroom application data files (events, etc.) for this node
- `/stroomdata/stroom-index-p00` - location to store Stroom application index files
- `/stroomdata/stroom-working-p00` - location to store Stroom application working files (e.g. temporary files, output, etc.) for this node
- `/stroomdata/stroom-working-p00/proxy` - location for Stroom proxy to store inbound data files
- Node: `stroomp01.strmdev00.org`
- `/stroomdata/stroom-data-p01` - location to store Stroom application data files (events, etc.) for this node
- `/stroomdata/stroom-index-p01` - location to store Stroom application index files
- `/stroomdata/stroom-working-p01` - location to store Stroom application working files (e.g. temporary files, output, etc.) for this node
- `/stroomdata/stroom-working-p01/proxy` - location for Stroom proxy to store inbound data files

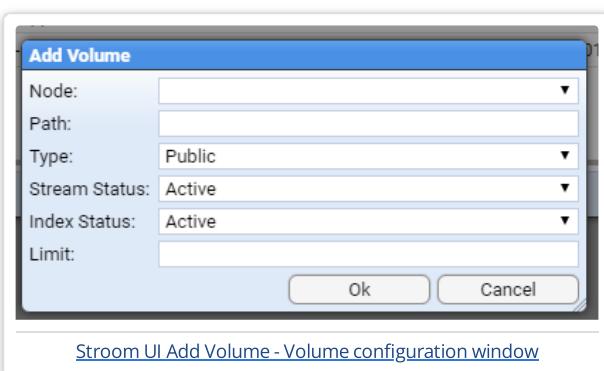
From this we need to create four volumes. On `stroomp00.strmdev00.org` we create

- `/stroomdata/stroom-data-p00` - location to store Stroom application data files (events, etc.) for this node
- `/stroomdata/stroom-index-p00` - location to store Stroom application index files

and on `stroomp01.strmdev00.org` we create

- `/stroomdata/stroom-data-p01` - location to store Stroom application data files (events, etc.) for this node
- `/stroomdata/stroom-index-p01` - location to store Stroom application index files

So the first step to configure a volume is to move the cursor to the *New* icon  in the top left of the `Volumes` window and select it. This will bring up the `Add Volume` configuration window

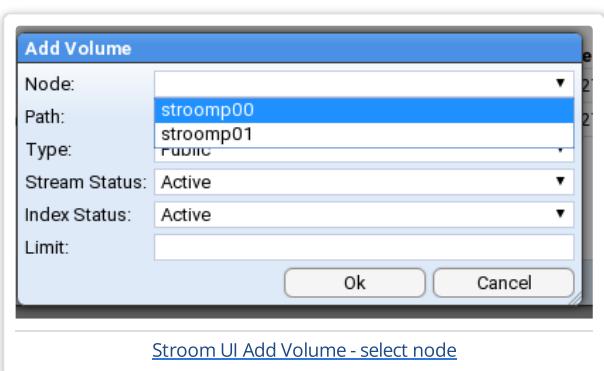


[Stroom UI Add Volume - Volume configuration window](#)

As you can see, the entry box titles reflect the attributes of a volume. So we will add the first nodes *data* volume

- `/stroomdata/stroom-data-p00` - location to store Stroom application data files (events, etc.) for this node for node `stroomp00`.

If you move the the *Node* drop down entry box and select it you will be presented with a choice of available nodes - in this case `stroomp00` and `stroomp01` as we have a two node cluster with these node names.



[Stroom UI Add Volume - select node](#)

By selecting the node `stroomp00` we see

The dialog box has a blue header bar with the title "Add Volume". Below it are five input fields with dropdown menus: "Node" (set to "stroomp00"), "Path" (empty), "Type" (set to "Public"), "Stream Status" (set to "Active"), and "Index Status" (set to "Active"). At the bottom are two buttons: "Ok" and "Cancel".

[Stroom UI Add Volume - selected node](#)

To configure the rest of the attributes for this volume, we:

- enter the *Path* to our first node's *data* volume
- select a *Volume Type* of *Public* as this is a data volume we want all nodes to access
- select a *Stream Status* of *Active* to indicate we want to store data on it
- select an *Index Status* of *Inactive* as we do **NOT** want index data stored on it
- set a *Limit* of 12GB for allowed storage

The dialog box has a blue header bar with the title "Add Volume". Below it are five input fields with dropdown menus: "Node" (set to "stroomp00"), "Path" (set to "/stroomdata/stroom-data-p00"), "Type" (set to "Public"), "Stream Status" (set to "Active"), and "Index Status" (set to "Inactive"). The "Limit" field contains "12GB". At the bottom are two buttons: "Ok" and "Cancel".

[Stroom UI Add Volume - adding first data volume](#)

and on selection of the



[Stroom UI OkButton](#)

we see the changes in the **Volumes** configuration window

The table has a blue header bar with the title "Volumes". It includes icons for adding, editing, and deleting. The columns are: Node, Path, Volume Type, Stream Status, Index Status, Usage Date, Limit, Used, and Free. There are three rows of data:

Node	Path	Volume Type	Stream Status	Index Status	Usage Date	Limit	Used	Free
stroomp00	/home/stroomuser/stroom-app/volumes/defaultIndexVolume	Private	Inactive	Active	2017-02-02T09:30:55.954Z	12 GB	2.2 GB	11 GB
stroomp00	/home/stroomuser/stroom-app/volumes/defaultStreamVolume	Public	Active	Inactive	2017-02-02T09:30:55.959Z	12 GB	2.2 GB	11 GB
stroomp00	/stroomdata/stroom-data-p00	Public	Active	Inactive			?	?

[Stroom UI Add Volume - added first data volume](#)

We next add the first node's index volume, as per

The dialog box has a blue header bar with the title "Add Volume". Below it are five input fields with dropdown menus: "Node" (set to "stroomp00"), "Path" (set to "/stroomdata/stroom-index-p00"), "Type" (set to "Private"), "Stream Status" (set to "Inactive"), and "Index Status" (set to "Active"). The "Limit" field contains "12GB". At the bottom are two buttons: "Ok" and "Cancel".

[Stroom UI Add Volume - adding first index volume](#)

And after adding the second node's volumes we are finally presented with our configured volumes

Volumes								
Node	Path	Volume Type	Stream Status	Index Status	Usage Date	Limit	Used	Free
stroomp00	/home/stroomuser/stroom-app/volumes/defaultIndexVolume	Private	Inactive	Active	2017-02-02T09:35:56.145Z	12 GB	2.2 GB	11 GB
stroomp00	/home/stroomuser/stroom-app/volumes/defaultStreamVolume	Public	Active	Inactive	2017-02-02T09:35:56.197Z	12 GB	2.2 GB	11 GB
stroomp00	/stroomdata/stroom-data-p00	Public	Active	Inactive	2017-02-02T09:35:56.198Z		32 MB	15 GB
stroomp00	/stroomdata/stroom-index-p00	Private	Inactive	Active	2017-02-02T09:35:56.199Z		32 MB	15 GB
stroomp01	/stroomdata/stroom-data-p01	Public	Active	Inactive			?	?
stroomp01	/stroomdata/stroom-index-p01	Private	Inactive	Active			?	?

[Stroom UI Add Volume - all volumes added](#)

4.1.11.2.0.1 Delete Default Volumes

We now need to deal with our default volumes. We want to delete them.

stroomp00	/home/stroomuser/stroom-app/volumes/defaultIndexVolume	Private	Inactive	Active	2017-02-02T09:35:56.145Z	12 GB	2.2 GB
stroomp00	/home/stroomuser/stroom-app/volumes/defaultStreamVolume	Public	Active	Inactive	2017-02-02T09:35:56.197Z	12 GB	2.2 GB

[Stroom UI Delete Default - display default](#)

So we move the cursor to the first volume's line (`stroomp00 /home/stroomuser/stroom-app/volumes/defaultIndexVolume ...`) and select the line then move the cursor to the *Delete* icon  in the top left of the `Volumes` window and select it. On selection you will be given a confirmation request



[Stroom UI Delete Default - confirm deletion](#)

at which we press the

Ok

[Stroom UI OkButton](#)

button to see the first default volume has been deleted

Volumes								
Node	Path	Volume Type	Stream Status	Index Status	Usage Date	Limit	Used	Free
stroomp00	/home/stroomuser/stroom-app/volumes/defaultStreamVolume	Public	Active	Inactive	2017-02-02T09:35:56.197Z	12 GB	2.2 GB	11 GB
stroomp00	/stroomdata/stroom-data-p00	Public	Active	Inactive	2017-02-02T09:35:56.198Z		32 MB	15 GB
stroomp00	/stroomdata/stroom-index-p00	Private	Inactive	Active	2017-02-02T09:35:56.199Z		32 MB	15 GB
stroomp01	/stroomdata/stroom-data-p01	Public	Active	Inactive			?	?
stroomp01	/stroomdata/stroom-index-p01	Private	Inactive	Active			?	?

[Stroom UI Delete Default - first volume deleted](#)

and after we select then delete the second default volume(`stroomp00 /home/stroomuser/stroom-app/volumes/defaultStreamVolume ...`), we are left with

Volumes								
Node	Path	Volume Type	Stream Status	Index Status	Usage Date	Limit	Used	Free
stroomp00	/stroomdata/stroom-data-p00	Public	Active	Inactive	2017-02-02T09:35:56.198Z	32 MB	15 GB	
stroomp00	/stroomdata/stroom-index-p00	Private	Inactive	Active	2017-02-02T09:35:56.199Z	32 MB	15 GB	
stroomp01	/stroomdata/stroom-data-p01	Public	Active	Inactive	2017-02-02T09:40:18.431Z	32 MB	15 GB	
	/stroom-index-p01	Private	Inactive	Active	2017-02-02T09:40:18.466Z	32 MB	15 GB	

Close

[Stroom UI CloseButton](#)

[Stroom UI Delete Default - all deleted](#)

button.

NOTE: At the time of writing there is an issue regarding volumes

4.1.11.2.0.0.1 Stroom Github Issue 84 -

Due to [Issue 84 \(external link\)](#), if we delete volumes in a multi node environment, the deletion is not propagated to all other nodes in a cluster. Thus if we attempted to use the volumes we would get a database error. The current *workaround* is to restart all the Stroom applications which will cause a reload of all volume information. This **MUST** be done before sending any data to your multi-node Stroom cluster.

4.1.11.3 Adding new Volumes

When one expands a Multi Node Stroom cluster deployment, after the installation of the Stroom Proxy and Application software and services on the new node, one has to configure the new volumes that are on the new node. The following demonstrates this assuming we are adding

- the new node is `stroomp02`
- the storage hierarchy for this node is
- `/stroomdata/stroom-data-p02` - location to store Stroom application data files (events, etc.) for this node
- `/stroomdata/stroom-index-p02` - location to store Stroom application index files
- `/stroomdata/stroom-working-p02` - location to store Stroom application working files (e.g. tmp, output, etc.) for this node
- `/stroomdata/stroom-working-p02/proxy` - location for Stroom proxy to store inbound data files

From this we need to create two volumes on `stroomp02`

- `/stroomdata/stroom-data-p02` - location to store Stroom application data files (events, etc.) for this node
- `/stroomdata/stroom-index-p02` - location to store Stroom application index files

To configure the volumes, move to the `Tools` item of the **Main Menu** and select it to bring up the `Tools` sub-menu.



then move down and select the `Volumes` sub-item to be presented with the `Volumes` configuration window as. We then move the cursor to the `New` icon **+** in the top left of the `Volumes` window and select it. This will bring up the `Add Volume` configuration window where we select our volume's node `stroomp02`.

Volumes

Node	Path	Volume Type	Stream Status	Index Status	Usage Date	Limit	Used	Free
stroomp00	/stroomdata/stroom-data-p00	Public	Active	Inactive	2017-01-14T08:36:09.534Z	12 GB	2.0 GB	11 GB
stroomp00	/stroomdata/stroom-index-p00	Private	Inactive	Active	2017-01-14T08:36:09.553Z	12 GB	2.0 GB	11 GB
stroomp01	/stroomdata/stroom-data-p01	Public	Active	Inactive	2017-01-14T08:32:27.348Z	12 GB	2.0 GB	11 GB
stroomp01	/stroomdata/stroom-index-p01	Private	Inactive	Active	2017-01-14T08:32:27.349Z	12 GB	2.0 GB	11 GB

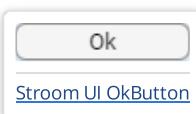
Add Volume

Node: stroomp00
Path: /stroomdata/stroom-data-p02
Type: Public
Stream Status: Active
Index Status: Inactive
Limit: 12GB

Ok Cancel

[Stroom UI Volumes - New Node configuration window start data volume](#)

then press the



button.

We then add another volume for the *index* volume for this node with attributes as per

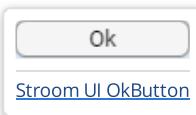
Add Volume

Node: stroomp02
Path: /stroomdata/stroom-index-02
Type: Private
Stream Status: Inactive
Index Status: Active
Limit: 12GB

Ok Cancel

[Stroom UI Volumes - New Node configuration window index volume added](#)

And on pressing the



button we see our two new volumes for this node have been added.

Volumes

Node	Path	Volume Type	Stream Status	Index Status	Usage Date	Limit	Used	Free
stroomp00	/stroomdata/stroom-data-p00	Public	Active	Inactive	2017-01-14T08:36:09.534Z	12 GB	2.0 GB	11 GB
stroomp00	/stroomdata/stroom-index-p00	Private	Inactive	Active	2017-01-14T08:36:09.553Z	12 GB	2.0 GB	11 GB
stroomp01	/stroomdata/stroom-data-p01	Public	Active	Inactive	2017-01-14T08:37:27.474Z	12 GB	2.0 GB	11 GB
stroomp01	/stroomdata/stroom-index-p01	Private	Inactive	Active	2017-01-14T08:37:27.488Z	12 GB	2.0 GB	11 GB
stroomp02	/stroomdata/stroom-data-p02	Public	Active	Inactive		12 GB	?	?
stroomp02	/stroomdata/stroom-index-p02	Private	Inactive	Active		12 GB	?	?

1 to 6 of 6

Close

[Stroom UI Volumes - New Node configuration window volumes added](#)

At this one can close the `volumes` configuration window by pressing the



button.

4.2 - Reference Feeds

4.2.1 - Create a Simple Reference Feed

How to create a reference feed for decorating event data using reference data lookups.

A **Reference Feed** is a temporal set of data that a pipeline's translation can look up to gain additional information to decorate the subject data of the translation. For example, an XML Event.

A Reference Feed is temporal, in that, each time a new set of reference data is loaded into Stroom, the effective date (for the data) is also recorded. Thus by using a timestamp field with the subject data, the appropriate batch of reference data can be accessed.

A typical reference data set to support the Stroom XML Event schema might be one that relates to devices. Such a data set can contain the device logical identifiers such as fully qualified domain name and ip address and their geographical location information such as country, site, building, room and timezone.

The following example will describe how to create a reference feed for such device data. we will call the reference feed `GeoHost-V1.0-REFERENCE`.

4.2.1.1 Reference Data

Our reference data will be supplied in a separated file containing the fields

- the device Fully Qualified Domain Name
- the device IP Address
- the device Country location (using ISO 3166-1 alpha-3 codes)
- the device Site location
- the device Building location
- the device TimeZone location (both standard then daylight timezone offsets from UTC)

For simplicity, our example will use a file with just 3 entries

```
FQDN    IPAddress   Country Site     Building      Room      TimeZones
stroomnode00.strmdev00.org 192.168.2.245 GBR Bristol-S00 GZero    R00 +00:00/+01:00
stroomnode01.strmdev01.org 192.168.3.117 AUS Sydney-S04 R6      5-134    +10:00/+11:00
host01.company4.org 192.168.4.220 USA LosAngeles-S19 ILM C5-54-2 -08:00/-07:00
```

A copy of this sample data source can be found [here](#). Save a copy of this data to your local environment for use later in this HOWTO. Save this file as a text document with ANSI encoding.

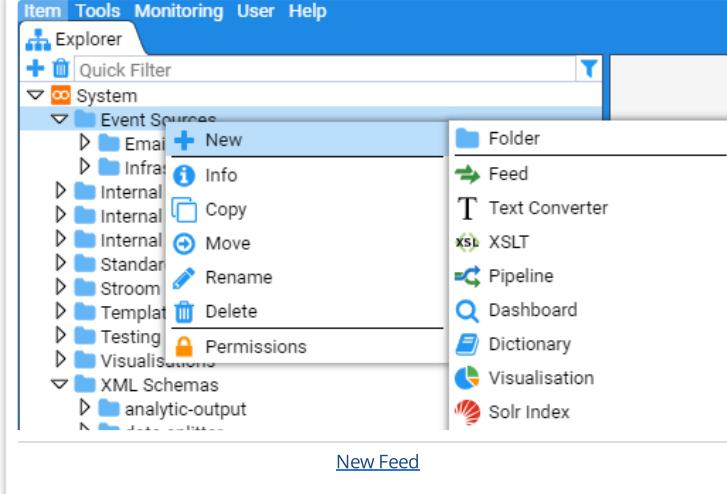
4.2.1.2 Creation

To create our Reference Event stream we need to create:

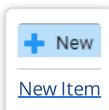
- the **Feed**
- a **Pipeline** to automatically process and store the Reference data
- a **Text Parser** to convert the text file into simple XML record format, and
- a **Translation** to create reference data maps

4.2.1.2.1 Create Feed

First, within the Explorer pane, and with the cursor having selected the Event Sources group, right click the mouse to have the object context menu appear.

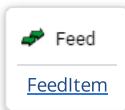


If you hover over the

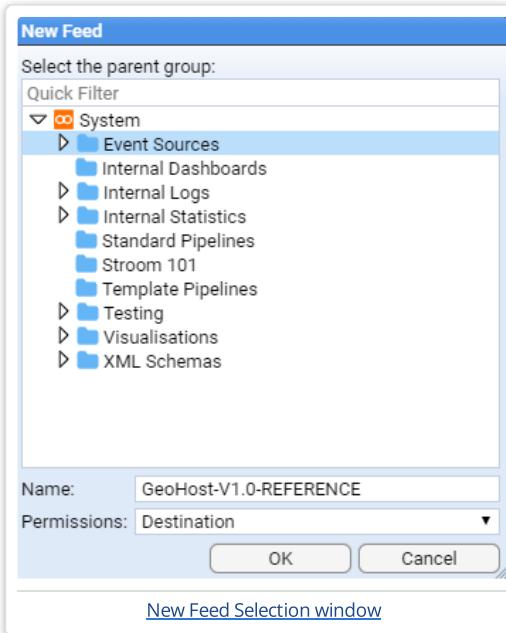


icon then the New sub-context menu will be revealed.

Now hover the mouse over the

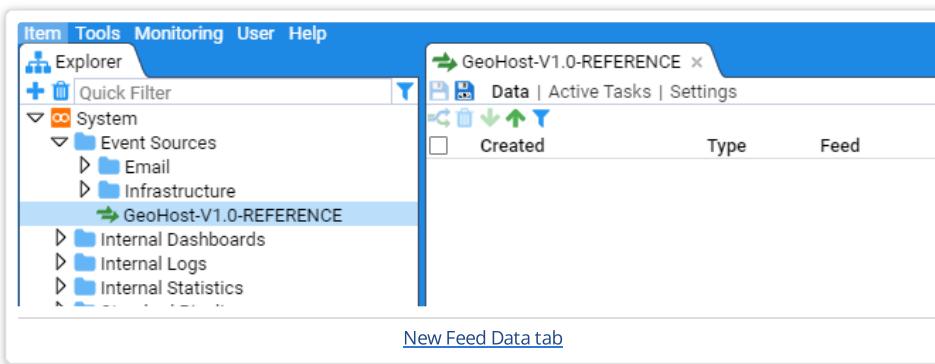


icon and right click to select.



[New Feed Selection window](#)

When the **New Feed** selection windows comes up, navigate to the `Event Sources` system group. Then enter the name of the reference feed `GeoHost -V1.0-REFERENCE` onto the **Name:** text entry box. On pressing the OK button we will see the following Feed configuration tab appear.



[New Feed Data tab](#)

Click on the **Settings** sub-item in the **GeoHost-V1.0-REFERENCE** Feed tab to populate the initial Settings configuration. Enter an appropriate description, classification and click on the **Reference Feed** check box

The screenshot shows the 'Settings' tab of the 'GeoHost-V1.0-REFERENCE' feed configuration. The 'Description' field contains the text: 'Device Logical and Geographic reference feed holding FQDN, IPAddress, Country, Site, Building, Room, and TimeZone'. The 'Classification' field is set to 'UNCLASSIFIED'. The 'Reference Feed' checkbox is checked. Other configuration options include: 'Feed Status' (Receive), 'Stream Type' (Raw Events), 'Data Encoding' (UTF-8), 'Context Encoding' (UTF-8), and 'Retention Period' (Forever). A link at the bottom of the window says 'New Feed Settings tab'.

and we then use the Stream Type drop-down menu to set the stream type as Raw Reference. At this point we save our configuration so far, by clicking on the



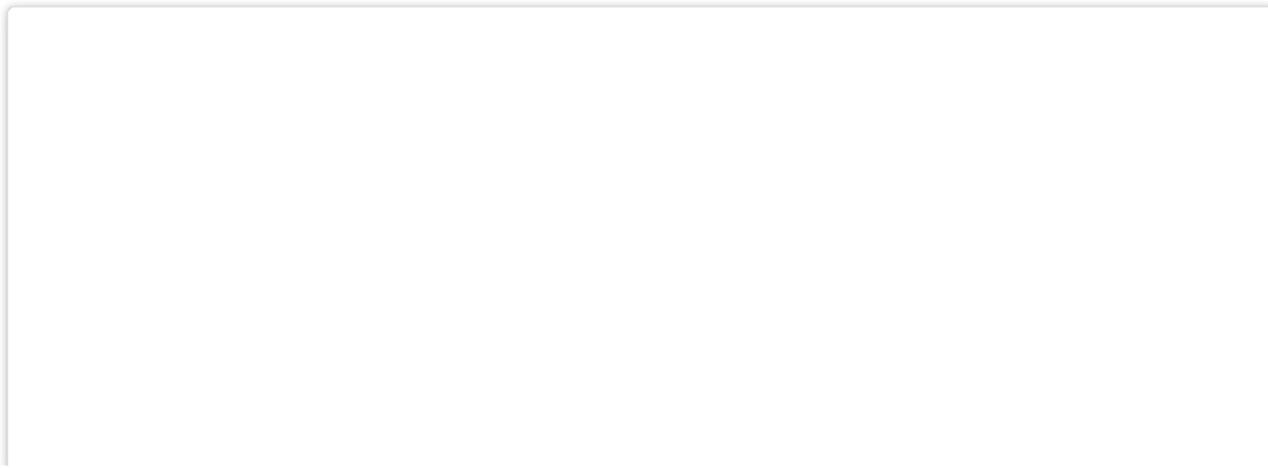
icon. The save icon becomes ghosted and our feed configuration has been saved.

The screenshot shows the 'Settings' tab of the 'GeoHost-V1.0-REFERENCE' feed configuration. The configuration is identical to the previous screenshot, with the 'Stream Type' set to 'Raw Events'. A link at the bottom of the window says 'New Feed Settings window configuration'.

4.2.1.2.2 Load sample Reference data

At this point we want to load our sample reference data, in order to develop our reference feed. We can do this two ways - posting the file to our Stroom web server, or directly upload the data using the user interface. For this example we will use Stroom's user interface upload facility.

First, open the **Data** sub-item in the **GeoHost-V1.0-REFERENCE** feed configuration tab to reveal

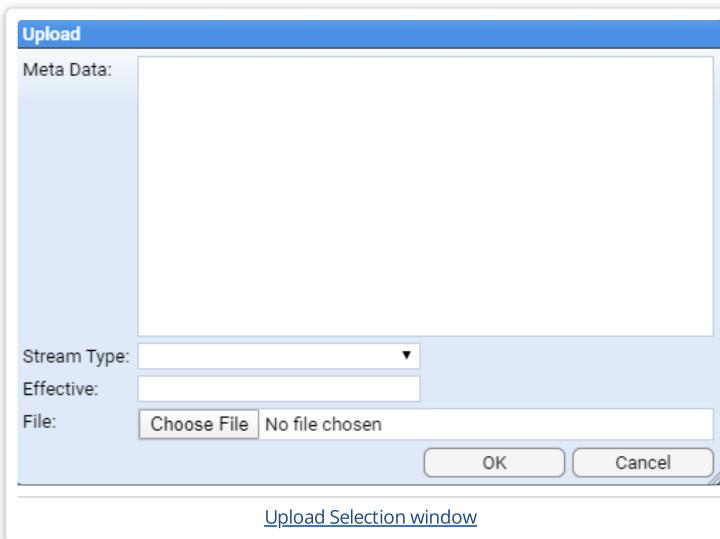


The screenshot shows the GeoHost-V1.0-REFERENCE application interface. At the top, there's a header bar with icons for back, forward, and search, followed by 'Data | Active Tasks | Settings'. Below the header are three tabs: 'Created' (selected), 'Type', and 'Feed'. To the right of these tabs is a 'Pipeline' section. A vertical blue bar on the left indicates a scroll position. At the bottom of the main pane, there's another set of tabs: 'Created', 'Type', 'Feed', and 'Pipeline'. Below these tabs, the text 'Reference Data configuration tab' is displayed.

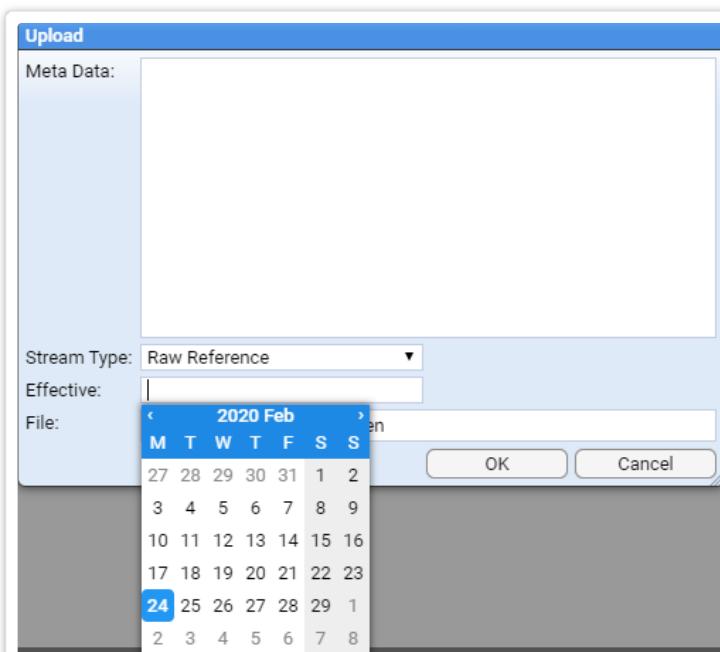
Note the Upload icon



in the bottom left of the **Stream table** (top pane). On clicking the Upload icon, we are presented with the data upload selection window.



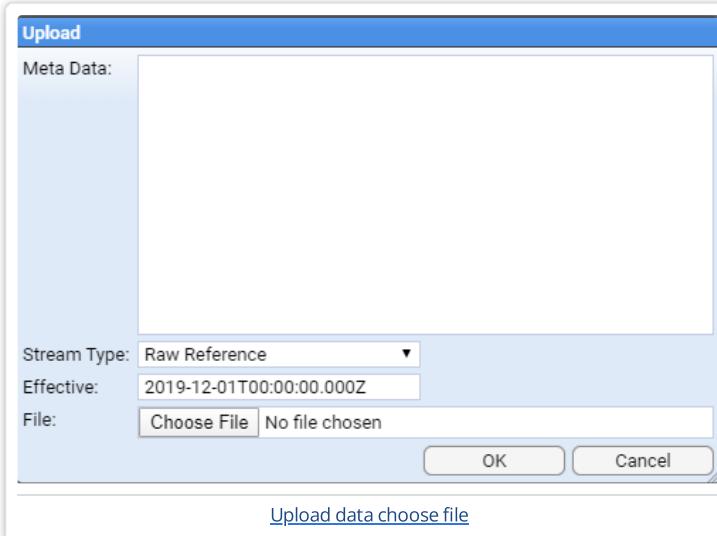
Naturally, as this is a reference feed we are creating and this is raw data we are uploading, we select a **Stream Type:** of Raw Reference. We need to set the **Effective:** date (really a timestamp) for this specific *stream* of reference data. Clicking in the **Effective:** entry box will cause a calendar selection window to be displayed (initially set to the current date).



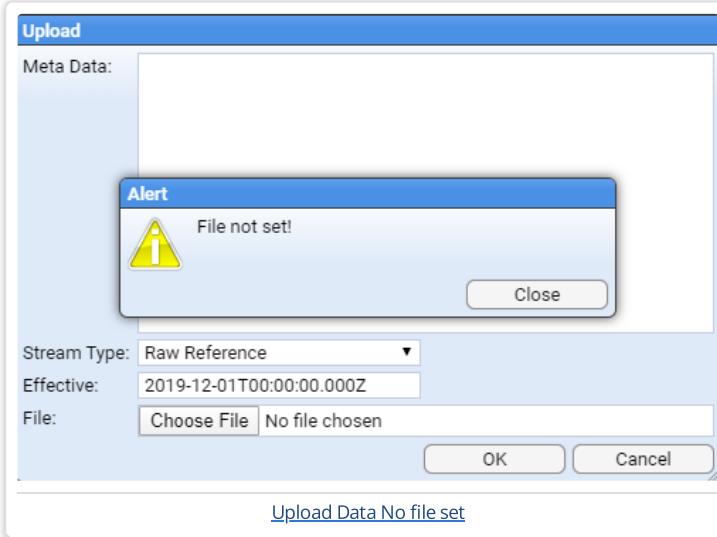
We are going to set the effective date to be late in 2019. Normally, you would choose a time stamp that matches the generation of the reference data. Click on the blue Previous Month icon (a less than symbol <) on the Year/Month line to move back to December 2019.



Select the 1st (clicking on 1) at which point the calendar selection window will disappear and a time of 2019-12-01T00:00:00.000Z is displayed. This is the default whenever using the calendar selection window in Stroom - the resultant timestamp is that of the day selected at 00:00:00 (Zulu time). To get the calendar selection window to disappear, click anywhere outside of the timestamp entry box.



Note, if you happen to click on the **OK** button before selecting the **File** (or Stream Type for that matter), an appropriate Alert dialog box will be displayed



We don't need to set **Meta Data** for this stream of reference data, but we (obviously) need to select the file. For the purposes of this example, we will utilise the file **GeoHostReference.log** you downloaded earlier in the **Reference Data** section of this document. This file contains a header and three lines of reference data as per

FQDN	IP Address	Country	Site	Building	Room	Time Zones
stroomnode00.strmdev00.org	192.168.2.245	GBR	Bristol-S00	GZero	R00	+00:00/+01:00
stroomnode01.strmdev01.org	192.168.3.117	AUS	Sydney-S04	R6	5-134	+10:00/+11:00
host01.company4.org	192.168.4.220	USA	Los Angeles-S19	ILM	C5-54-2	-08:00/-07:00

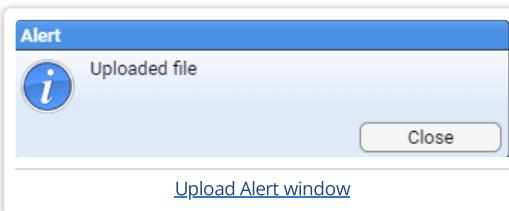
When we construct the pipeline for this reference feed, we will see how to make use of the header line.

So, click on the **Choose File** button to bring up a file selector window. Navigate within the selector window to the location on your location machine where you have saved the GeoHostReference.log file. On clicking **Open** we return to the **Upload** window with the file selected.



[Upload Reference Data - File chosen](#)

On clicking **OK** we get an Alert dialog window to advise a file has been uploaded.



at which point we press **Close**.

At this point, the **Upload** selection window closes, and we see our file displayed in the **GeoHost-V1.0-REFERENCE Data** stream table.

GeoHost-V1.0-REFERENCE			Pipeline
Data Active Tasks Settings			1 to 1 of 1
Created	Type	Feed	
2020-03-10T00:29:46.899Z	Raw Reference	GeoHost-V1.0-REFERENCE	

[Upload Display raw reference stream](#)

When we click on the newly up-loaded stream in the *Stream Table* pane we see the other two panes fill with information.

The screenshot shows the Stroom interface with three main panes:

- Pipeline Pane:** Shows a single item: "2020-03-10T00:29:46.899Z Raw Reference" of type "Raw Reference" from "GeoHost-V1.0-REFERENCE".
- Data Pane:** Shows a single item: "2020-03-10T00:29:46.899Z Raw Reference" of type "Raw Reference" from "GeoHost-V1.0-REFERENCE".
- Data | Meta Pane:** Displays the raw reference data and its metadata. The data table shows:

	FQDN	IPAddress	Country	Site	Building	Room	TimeZones
1	stroomnode00.strmdev00.org	192.168.2.245	GBR	Bristol-S00	GZero	R00	+00:00/+01:00
2	stroomnode01.strmdev01.org	192.168.3.117	AUS	Sydney-S04	R6	S-134	+10:00/+11:00
3	host01.company4.org	192.168.4.220	USA	LosAngeles-S19	ILM	C5-54-2	-08:00/-07:00
4							
5							

The middle pane shows the selected or *Specific* feed and any linked streams. A linked stream could be the resultant Reference data set generated from a Raw Reference stream. If errors occur during processing of the stream, then a linked stream could be an Error stream.

The bottom pane displays the selected stream's data or meta-data. If we click on the **Meta** link at the top of this pane, we will see the *Metadata* associated with this stream. We also note that the **Meta** link at the bottom of the pane is now embolden.

The screenshot shows the Stroom interface with three main panes:

- Pipeline Pane:** Shows a single item: "2020-03-10T00:29:46.899Z Raw Reference" of type "Raw Reference" from "GeoHost-V1.0-REFERENCE".
- Data Pane:** Shows a single item: "2020-03-10T00:29:46.899Z Raw Reference" of type "Raw Reference" from "GeoHost-V1.0-REFERENCE".
- Data | Meta Pane:** Displays the raw reference data and its metadata. The data table shows:

	EffectiveTime	Feed	ReceivedTime	RemoteFile	StreamSize	User-Agent
1	2019-12-01T00:00:00.000Z	GeoHost-V1.0-REFERENCE	2020-03-10T00:29:46.861Z	C:\fakepath\GeoHostReference.log	295	STROOM-UI
2						
3						
4						
5						
6						
7						

We can see the metadata we set - the EffectiveTime, and implicitly, the Feed but we also see additional fields that Stroom has added that provide more detail about the data and its delivery to Stroom such as how and when it was received. We now need to switch back to the Data display as we need to author our reference feed translation.

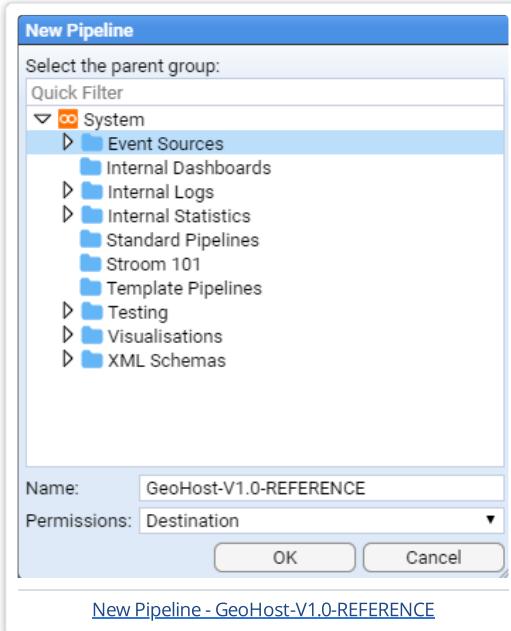
4.2.1.2.3 Create Pipeline

We now need to create the pipeline for our reference feed so that we can create our translation and hence create reference data for our feed.

Within the Explorer pane, and having selected the **Event Sources** system group, right click to bring up the object context menu, then the **New** sub-context menu. Move to the



and left click to select. When the *New Pipeline* selection window appears, navigate to, then select the **Feeds and Translations** system group then enter the name of the reference feed, GeoHost-V1.0-REFERENCE in the **Name:** text entry box.



On pressing the **OK** button you will be presented with the new pipeline's configuration tab



Within **Settings**, enter an appropriate description as per



We now need to select the structure this pipeline will use. We need to move from the **Settings** sub-item on the pipeline configuration tab to the **Structure** sub-item. This is done by clicking on the **Structure** link, at which we will see

As this pipeline will be processing reference data, we would use a `Reference Data` pipeline. This is done by inheriting it from a defined set of Standard Pipelines. To do this, click on the menu selection icon



to the right of the **Inherit From:** test display box.

When the **Choose item** selection window appears, navigate to `Template Pipelines` system group (if not already displayed), and select (left click) the



`Reference Data` pipeline

New Pipeline - Reference Data pipeline inherited

then press **OK**. At this we will see the inherited pipeline structure of

New Pipeline - Inherited set



to save, which results in



This ends the first stage of the pipeline creation. We need to author the feed's translation.

4.2.1.2.4 Create Text Converter

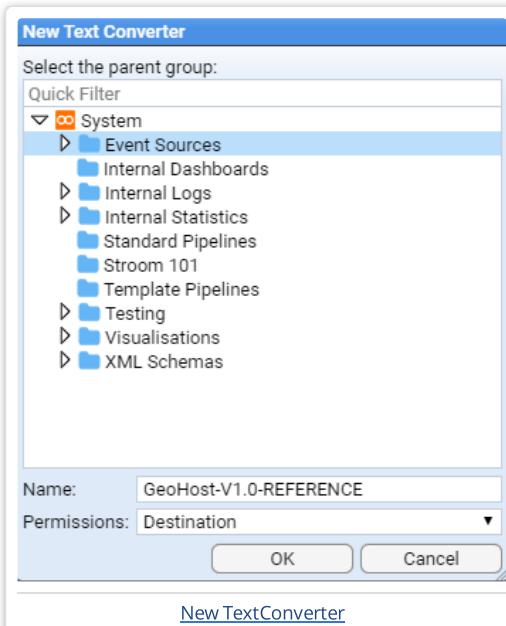
To turn our tab delimited data in Stroom reference data, we first need to convert the text into simple XML. We do this using a *Text Converter*. *Text Converters* use a *Stroom Data Splitter* to convert text into simple XML.

Within the Explorer pane, and having selected the `Event Sources` system group, right click to bring up the object context menu. Navigate to the

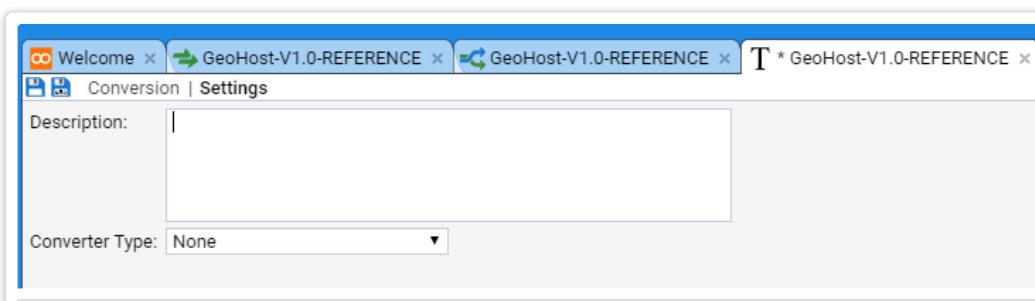


item and left click to select.

When the *New Text Converter* selection window comes up, navigate to and select `Event Sources` system group, then enter the name of the feed, `GeoHost-V1.0-REFERENCE` into the **Name:** text entry box as per



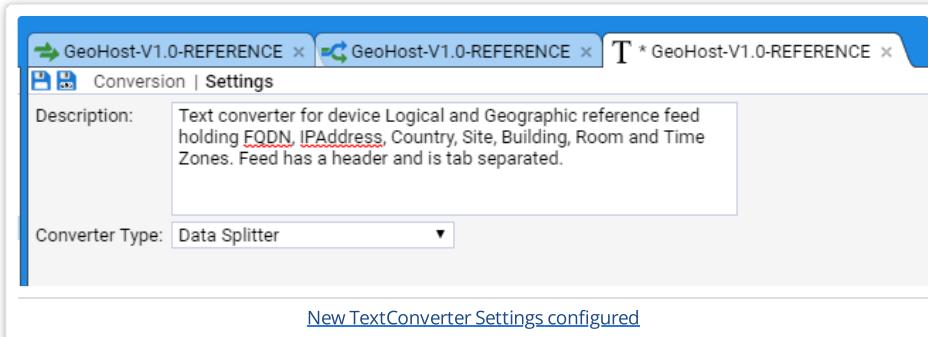
On pressing the **OK** button we see the next text converter's configuration tab displayed.



Enter an appropriate description into the **Description:** text entry box, for instance

Text converter for device Logical and Geographic reference feed holding FQDN, IPAddress, Country, Site, Building, Room and Time Zones. Feed has a header and is tab separated.

Set the **Converter Type:** to be **Data Splitter** from the drop-down menu.



We next press the **Conversion** sub-item on the TextConverter tab to bring up the *Data Splitter* editing window.

The following is our Data Splitter code (see **Data Splitter** documentation for more complete details)

```

<?xml version="1.1" encoding="UTF-8"?>
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3
<!--
GEOHOST REFERENCE FEED:

CHANGE HISTORY
v1.0.0 - 2020-02-09 John Doe

This is a reference feed for device Logical and Geographic data.

The feed provides for each device
* the device FQDN
* the device IP Address
* the device Country location (using ISO 3166-1 alpha-3 codes)
* the device Site location
* the device Building location
* the device Room location
*the device TimeZone location (both standard then daylight timezone offsets from UTC)

The data is a TAB delimited file with the first line providing headings.

Example data:

FQDN    IPAddress    Country Site      Building     Room      TimeZones
stroomnode00.strmdev00.org 192.168.2.245   GBR Bristol-S00 GZero    R00 +00:00/+01:00
stroomnode01.strmdev01.org 192.168.3.117   AUS Sydney-S04 R6      5-134    +10:00/+11:00
host01.company4.org 192.168.4.220   USA LosAngeles-S19 ILM C5-54-2 -08:00/-07:00

-->

<!-- Match the heading line - split on newline and match a maximum of one line -->
<split delimiter="\n" maxMatch="1">

<!-- Store each heading and note we split fields on the TAB (&#9;) character -->
<group>
  <split delimiter="#&#9;">
    <var id="heading"/>
  </split>
</group>
</split>

<!-- Match all other data lines - splitting on newline -->
<split delimiter="\n">
<group>
  <!-- Store each field using the column heading number for each column ($heading$1) and note we split fields on the TAB (&#9;)
  <split delimiter="#&#9;">
    <data name="$heading$1" value="$1"/>
  </split>
</group>
</split>
</dataSplitter>
```

At this point we want to save our Text Converter, so click on the

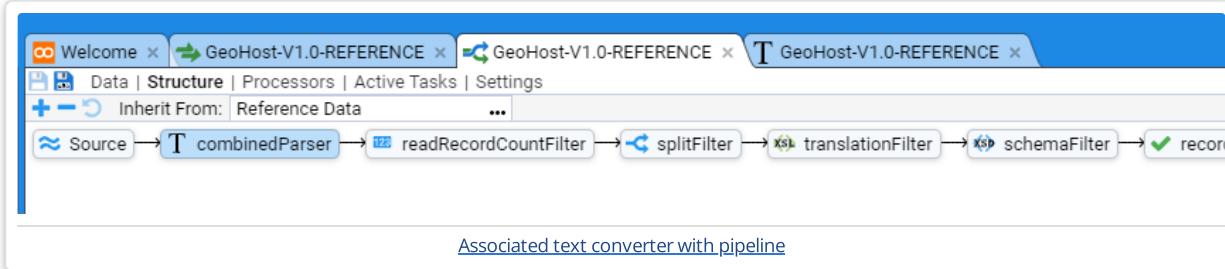


icon.

A copy of this data splitter can be found [here](#).

4.2.1.2.5 Assign Text Converter to Pipeline

To test our Text Converter, we need to modify our `GeoHost-V1.0-REFERENCE` pipeline to use it. Select the `GeoHost-V1.0-REFERENCE` pipeline tab and then select the **Structure** sub-item



To associate our new Text Converter with the pipeline, click on the



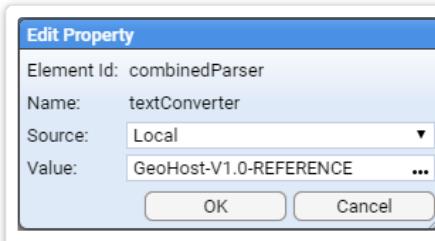
icon then move the cursor to the *Property* (middle) pane then double click on the **textConverter** Property Name to allow you to edit the property as per



We leave the Property **Source**: as *Inherit* but we need to change the Property **Value**: from *None* to be our newly created `GeoHost-V1.0-REFERENCE` text Converter



then press **OK**. At this we will see the Property *Value* set



[textConverter set Property Value](#)

Again press **OK** to finish editing this property and we then see that the **textConverter** property has been set to GeoHost-V1.0-REFERENCE. Similarly set the **type** property *Value* to "Data Splitter".

At this point, we should save our changes, by clicking on the highlighted



icon. The combined Parser window panes should now look like

The screenshot shows the combined Parser window with two main sections. At the top, a pipeline diagram is displayed with nodes: Source → combinedParser → readRecordCountFilter → splitFilter → translationFilter → schemaFilter → recordOutputFilter → writeRecordCountFilter. Below the pipeline is a table of properties:

Property Name	Value	Source	Inherited Value	Inherited From	Default Value	Description
fixInvalidChars	false	Inherit	false		false	Fix invalid XML characters from the input stream.
textConverter	GeoHost-V1.0-REFERENCE	Local				The text converter configuration that should be used to parse the input.
type		Inherit				The parser type, e.g. 'JSON', 'XML', 'Data Splitter'.

At the bottom of the window, there is a link: [textConverter set Property Value type](#).

4.2.1.2.6 Test Text Converter

To test our Text Converter, we select the GeoHost-V1.0-REFERENCE **Feed** tab



then click on our uploaded stream in the *Stream Table* pane, then click the check box of the *Raw Reference* stream in the *Specific Stream* table (middle pane)

The screenshot shows the Feed tab with the specific stream table selected. The middle pane shows the selected row:

Created	Type	Feed	Pipeline
2020-03-10T00:29:46.899Z	Raw Reference	GeoHost-V1.0-REFERENCE	

At the bottom, the Data | Meta pane displays the raw reference data:

FQDN	IPAddress	Country	Site	Building	Room	TimeZones
stroomnode00.strmdev00.org	192.168.2.245	GBR	Bristol-S00	GZero	R00	+00:00/+01:00
stroomnode01.strmdev01.org	192.168.3.117	AUS	Sydney-S04	R6	5-134	+10:00/+11:00
host01.company4.org	192.168.4.220	USA	LosAngeles-S19	ILM	C5-54-2	-08:00/-07:00
5						

At the bottom of the window, there is a link: [textConverter - select raw reference data](#).

We now want to step our data through the Text Converter. We enter Stepping Mode by pressing the stepping button

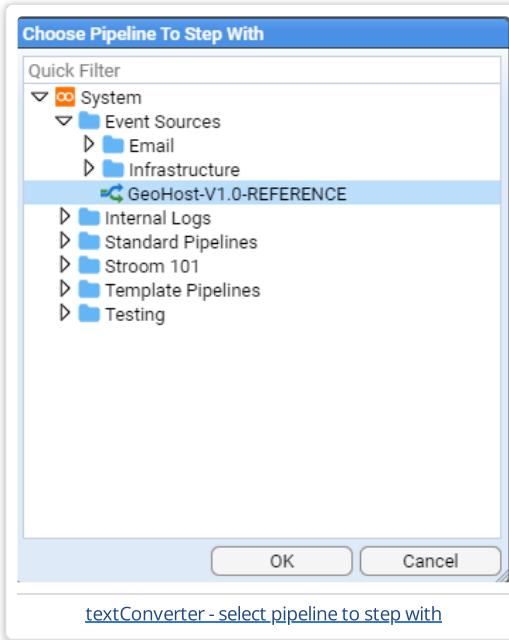




[Enter Stepping](#)

found at the bottom of the right of the stream *Raw Data* display.

You will then be requested to choose a pipeline to step with, at which, you should navigate to the `GeoHost-V1.0-REFERENCE` pipeline as per



then press **OK**.

At this point we enter the pipeline Stepping tab



which initially displays the Raw Reference data from our stream.

We click on the



icon, to display.

GeoHost-V1.0-REFERENCE x GeoHost-V1.0-REFERENCE x GeoHost-V1.0-REFERENCE x

Source → T combinedParser → translationFilter → schemaFilter → xmlWriter → streamAppender

```

1 <?xml version="1.1" encoding="UTF-8"?>
2 <dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3 file:///data-splitter.xsd">
3   <!--
4     GEOHOST REFERENCE FEED:
5
6     CHANGE HISTORY
7     v1.0.0 - 2020-02-09 John Doe
8
9     This is a reference feed for device Logical and Geographic data.
10
11    The feed provides for each device
12      * the device FQDN
13      * the device IP Address
14      * the device Country location (using ISO 3166-1 alpha-3 codes)
15      * the device Site location
16      * the device Building location
17      * the device Room location
18      *the device TimeZone location (both standard then daylight timezone offsets from UTC)
19
20    The data is a TAB delimited file with the first line providing headings.
21
22    Example data:
23
24      FQDN  IPAddress Country Site  Building  Room  TimeZone

```

UNCLASSIFIED

[textConverter - stepping.editor workspace](#)

This *stepping* window is divided into three sub-panes. the top one is the Text Converter editor and it will allow you to adjust the text conversion should you wish too. The bottom left window displays the *input* to the Text Converter. The bottom right window displays the *output* from the Text Converter for the given input.

We now click on the pipeline Step Forward button



to single step the Raw reference data throughout text converter. We see that the Stepping function has displayed the heading and first data line of our raw reference data in the *input* sub-pane and the resultant simple *records* XML (adhering to the Stroom **records v2.0** schema) in the *output* pane.

GeoHost-V1.0-REFERENCE x GeoHost-V1.0-REFERENCE x T GeoHost-V1.0-REFERENCE x H GeoHost-V1.0-REFERENCE x [168533:1:1] 

Source → T combinedParser → xsl translationFilter → xs schemaFilter → xsl xmlWriter → streamAppender

```

1 <?xml version="1.1" encoding="UTF-8"?>
2 <dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3 file://data-splitter->
3 <!--
4 GEOHOST REFERENCE FEED:
5
6 CHANGE HISTORY
7 v1.0.0 - 2020-02-09 John Doe
8
9 This is a reference feed for device Logical and Geographic data.
10
11 The feed provides for each device
12 * the device FQDN
13 * the device IP Address
14 * the device Country location (using ISO 3166-1 alpha-3 codes)
15 * the device Site location
16 * the device Building location
17 * the device Room location
18 *the device TimeZone location (both standard then daylight timezone offsets from UTC)
19
20 The data is a TAB delimited file with the first line providing headings.
21

```

FQDN	IPAddress	Country	Site	Building	Room	TimeZones
stroombnode00.strmdev00.org	192.168.2.245	GBR	Bristol-S00	GZero	R00	+00:00/+00:00/+01:00

```

<?xml version="1.1" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<record>
<data name="FQDN" value="stroombnode00.strmdev00.org" />
<data name="IPAddress" value="192.168.2.245" />
<data name="Country" value="GBR" />
<data name="Site" value="Bristol-S00" />
<data name="Building" value="GZero" />
<data name="Room" value="R00" />
<data name="TimeZones" value="+00:00/+01:00" />
</record>
</records>

```

UNCLASSIFIED

[textConverter - pipeline stepping - 1st record](#)

If we again press the



[Step Forward](#)

button we see the second line in our raw reference data in the *input* sub-pane and the resultant simple *records* XML in the *output* pane.

stroombnode01.strmdev01.org	192.168.3.117	AUS	Sydney-S04	R6	5-134	+10:00/+11:00
stroombnode01.strmdev01.org	192.168.3.117	AUS	Sydney-S04	R6	5-134	+10:00/+11:00

```

<?xml version="1.1" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<record>
<data name="FQDN" value="stroombnode01.strmdev01.org" />
<data name="IPAddress" value="192.168.3.117" />
<data name="Country" value="AUS" />
<data name="Site" value="Sydney-S04" />
<data name="Building" value="R6" />
<data name="Room" value="5-134" />
<data name="TimeZones" value="+10:00/+11:00" />
</record>
</records>

```

UNCLASSIFIED

[textConverter - pipeline stepping - 2nd record](#)

Pressing the Step Forward button



[Step Forward](#)

again displays our third and last line of our raw and converted data.

host01.company4.org 192.168.4.220 USA LosAngeles-S19 ILM C5-54-2 -08:00/-07:

```
<?xml version="1.1" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<record>
<data name="FQDN" value="host01.company4.org" />
<data name="IPAddress" value="192.168.4.220" />
<data name="Country" value="USA" />
<data name="Site" value="LosAngeles-S19" />
<data name="Building" value="ILM" />
<data name="Room" value="C5-54-2" />
<data name="TimeZones" value="-08:00/-07:00" />
</record>
</records>
```

UNCLASSIFIED

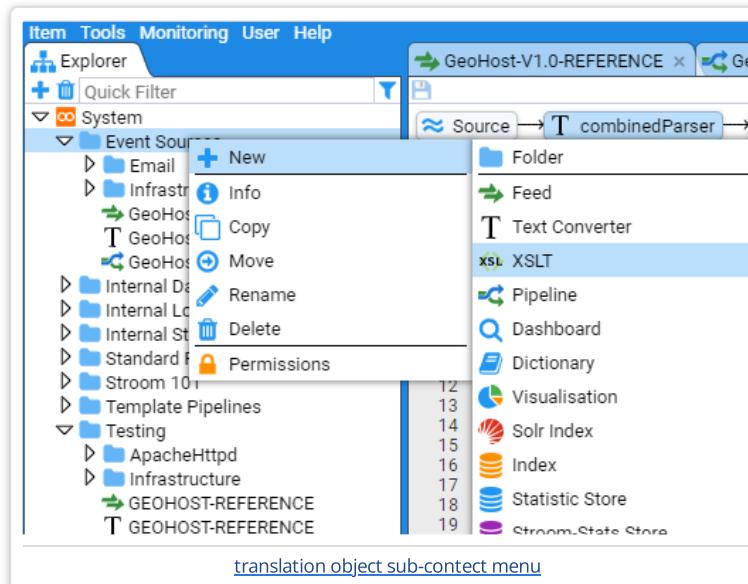
textConverter - pipeline stepping - 3rd record

We have now successfully tested the Text Converter for our reference feed. Our next step is to author our translation to generate reference data records that Stroom can use.

4.2.1.2.7 Create XSLT Translation

We now need to create our translation. This XSLT translation will convert simple *records* XML data into *ReferenceData* records - see the Stroom **reference-data v2.0.1** Schema for details.

We first need to create an XSLT translation for our feed. Move back to the Explorer window, and with the cursor having selected the Event Sources system group, right click the mouse to display the object context menu, select **New**



and then move the cursor to the

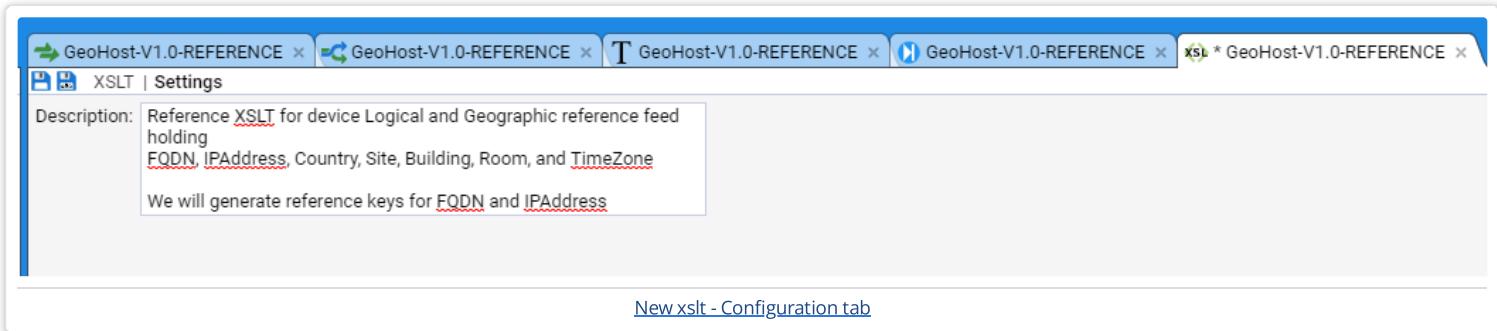


item, then left click to select.

When the **New XSLT** selection window comes up, navigate to the Event Sources system group and enter the name of the reference feed - GeoHost-V1.0-REFERENCE into the **Name:** text entry box as per



On pressing the **OK** button we see the XSL tab for our translation and as previously, we enter an appropriate description before selecting the **XSLT** sub-item.



On selection of the **XSLT** sub-item, we are presented with the XSLT editor window



At this point, rather than edit the translation in this editor and then assign this translation to the GeoHost-V1.0-REFERENCE pipeline, we will first make the assignment in the pipeline and then develop the translation whilst stepping through the raw data. This is to demonstrate there are a number of ways to *develop a translation*.

So, to start, save the XSLT by clicking on the



tab to raise the GeoHost-V1.0-REFERENCE pipeline. Then select the **Structure** sub-item followed by selecting the **XSL translationFilter** icon. We now see the **XSL translationFilter** Property Table for our pipeline in the middle pane.

GeoHost-V1.0-REFERENCE x GeoHost-V1.0-REFERENCE x T GeoHost-V1.0-REFERENCE x H GeoHost-V1.0-REFERENCE x xsl * GeoHost-V1.0-REFERENCE x

Data | Structure | Processors | Active Tasks | Settings

Inherit From: Reference Data ... View Source

Source → T combinedParser → readRecordCountFilter → splitFilter → xsl translationFilter → schemaFilter → recordOutputFilter → writeRecordCountFilter

Property Name	Value	Source	Inherited Value	Inherited From	Default Value	Description
suppressXSLTNotFoundWarnings	false	Inherit	false		false	If XSLT cannot be found to match the name pattern.
usePool	true	Inherit	true		true	Advanced: Choose whether or not you want to use a pool.
xslt		Inherit				The XSLT to use.
xsltNamePattern		Inherit				A name pattern to load XSLT dynamically.

1 to 4 of 4 << << >> >>

[xsl translation element - property pane](#)

To associate our new translation with the pipeline, move the cursor to the *Property Table*, click on the grayed out *xslt* Property Name and then click on the Edit Property



[Edit Property](#)

icon to allow you to edit the property as per

GeoHost-V1.0-REFERENCE x GeoHost-V1.0-REFERENCE x T GeoHost-V1.0-REFERENCE x H GeoHost-V1.0-REFERENCE x xsl * GeoHost-V1.0-REFERENCE x

Data | Structure | Processors | Active Tasks | Settings

Inherit From: Reference Data ...

Source → T combinedParser → readRecordCountFilter → splitFilter → xsl translationFilter

Property Name	Value	Source	Inherited Value	Inherited From
suppressXSLTNotFoundWarnings	false	Inherit	false	
usePool	true	Inherit	true	
xslt		Inherit		
xsltNamePattern		Inherit		

Edit Property

Element Id: translationFilter

Name:

Source:

Value:

[xsl -property editor](#)

We leave the Property **Source**: as *Inherit* and we need to change the Property **Value**: from *None* to be our newly created GeoHost-V1.0-REFERENCE XSL translation. To do this, position the cursor over the menu selection icon

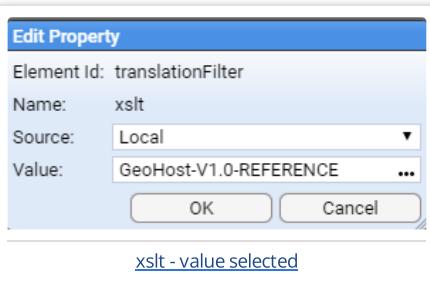


[Menu Selection](#)

of the **Value**: chooser and right click, at which the **Choose item** selection window appears. Navigate to the **Event Sources** system group then select the GeoHost-V1.0-REFERENCE xsl translation.



then press **OK**. At this point we will see the property **Value:** set



Again press **OK** to finish editing this property and we see that the *xs/t* property has been set to GeoHost-V1.0-REFERENCE.

Property Name	Value	Source	Inherited Value	Inherited From	Default Value
suppressXSLTNotFoundWarnings	false	Inherit	false		false
usePool	true	Inherit	true		true
xs/t	GeoHost-V1.0-REFERENCE	Local			
xs/tNamePattern		Inherit			

[xs/t - property set](#)

At this point, we should save our changes, by clicking on the highlighted



icon.

4.2.1.2.8 Test XSLT Translation

We now go back to the GeoHost-V1.0-REFERENCE **Feed** tab



then click on our uploaded stream in the *Stream Table* pane. Next click the check box of the *Raw Reference* stream in the *Specific Stream* table (middle pane) as per

GeoHost-V1.0-REFERENCE x GeoHost-V1.0-REFERENCE x T GeoHost-V1.0-REFERENCE

Data | Active Tasks | Settings

Created	Type	Feed	Pipeline
<input type="checkbox"/> 2020-03-10T00:29:46.899Z	Raw Reference	GeoHost-V1.0-REFERENCE	

Created	Type	Feed	Pipeline
<input checked="" type="checkbox"/> > <input type="checkbox"/> 2020-03-10T00:29:46.899Z	Raw Reference	GeoHost-V1.0-REFERENCE	

Data | Meta

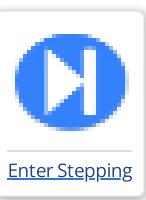
```

1 FQDN IPAddress Country Site Building Room TimeZones
2 stroombnode00.strmdev00.org 192.168.2.245 GBR Bristol-S00 GZero R00 +00:00/+01:00
3 stroombnode01.strmdev01.org 192.168.3.117 AUS Sydney-S04 R6 5-134 +10:00/+11:00
4 host01.company4.org 192.168.4.220 USA LosAngeles-S19 ILM C5-54-2 -08:00/-07:00
5

```

[GeoHost-V1.0-REFERENCE feedTab - Specific Stream](#)

We now want to step our data through the xslt Translation. We enter Stepping Mode by pressing the stepping button



found at the bottom of the right of the stream *Raw Data* display.

You will then be requested to choose a pipeline to step with, at which, you should navigate to the GeoHost-V1.0-REFERENCE pipeline as per

Choose Pipeline To Step With

Quick Filter

- System
 - Event Sources
 - Email
 - Infrastructure
 - GeoHost-V1.0-REFERENCE
 - Internal Logs
 - Standard Pipelines
 - Stroom 101
 - Template Pipelines
 - Testing

OK Cancel

[xslt Translation - select.pipeline to step with](#)

then press **OK**.

At this point we enter the pipeline through the Stepping tab

(Blank area for Stepping tab)

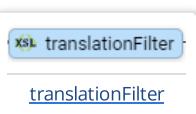
The screenshot shows a pipeline configuration window with the following steps: Source → combinedParser → translationFilter → schemaFilter → xmlWriter → streamAppender. The 'Data' tab is selected, displaying raw reference data in a table:

	FQDN	IPAddress	Country	Site	Building	Room	TimeZones
1	stroomnode00.strmdev00.org	192.168.2.245	GBR	Bristol-S00	GZero	R00	+00:00/+01:00
2	stroomnode01.strmdev01.org	192.168.3.117	AUS	Sydney-S04	R6	5-134	+10:00/+11:00
3	host01.company4.org	192.168.4.220	USA	LosAngeles-S19	ILM	C5-54-2	-08:00/-07:00
4							
5							

[xsIt Translation - stepping tab](#)

which initially displays the Raw Reference data from our stream.

We click on the



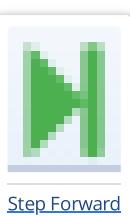
icon to enter the xsIt Translation stepping window and all panes are empty.

The screenshot shows the 'xsIt Translation - editor' window. The top pane is empty, indicating no XSLT translation has been entered. The bottom right pane is also empty.

[xsIt Translation - editor](#)

As for the Text Converter, this translation *stepping* window is divided into three sub-panes. The top one is the XSLT Translation. The bottom right window displays the *output* from the XSLT Translation for the given input.

We now click on the pipeline Step Forward button



to single step the Raw reference data through our translation. We see that the Stepping function has displayed the first *records* XML entry in the *input* sub-pane and the same data is displayed in the *output* sub-pane.

GeoHost-V1.0-REFERENCE x GeoHost-V1.0-REFERENCE x T GeoHost-V1.0-REFERENCE x xsl * GeoHost-V1.0-REFERENCE x GeoHost-V1.0-REFERENCE x [168533:1:1] ↻ ↺ ↺

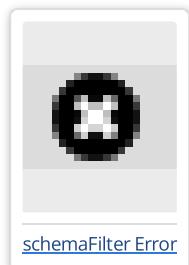
Source → T combinedParser → xsl translationFilter → xsp schemaFilter → xpi xmlWriter → ≈ streamAppender

```
<?xml version="1.1" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<record>
  <data name="FQDN" value="stroomnode00.strmdev00.org" />
  <data name="IPAddress" value="192.168.2.245" />
  <data name="Country" value="GBR" />
  <data name="Site" value="Bristol-S00" />
  <data name="Building" value="GZero" />
  <data name="Room" value="R00" />
  <data name="TimeZones" value="+00:00/+01:00" />
</record>
</records>
```

```
<?xml version="1.1" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<record>
  <data name="FQDN" value="stroomnode00.strmdev00.org" />
  <data name="IPAddress" value="192.168.2.245" />
  <data name="Country" value="GBR" />
  <data name="Site" value="Bristol-S00" />
  <data name="Building" value="GZero" />
  <data name="Room" value="R00" />
  <data name="TimeZones" value="+00:00/+01:00" />
</record>
</records>
```

xslt Translation - editor 1st record

But we also note if we move along the pipeline structure to the



[schemaFilter Error](#)

GeoHost-V1.0-REFERENCE x GeoHost-V1.0-REFERENCE x T GeoHost-V1.0-REFERENCE x xsl * GeoHost-V1.0-REFERENCE x

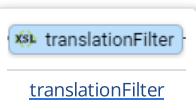
Source → T combinedParser → xsl translationFilter → xsp schemaFilter → xpi xmlWriter → ≈ streamAppender

```
1  <?xml version="1.1" encoding="UTF-8"?>
2  <records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="file://reference-data/v2.0.1.xsd">
3  <record>
4    <data name="Country" value="GBR" />
5    <data name="Site" value="Bristol-S00" />
6    <data name="Building" value="GZero" />
7    <data name="Room" value="R00" />
8    <data name="TimeZones" value="+00:00/+01:00" />
9  </record>
10 </records>
```

xslt Translation - schema fault

In essence, since the *translation* has done nothing, and the data is simple *records* XML, the system is indicating that it expects the *output* data to be in the *reference-data v2.0.1* format.

We can correct this by adding the skeleton xslt translation for reference data into our translationFilter. Move back to the



icon on the pipeline structure and add the following to the xsl window

```

<?xml version="1.1" encoding="UTF-8" ?>
<xsl:stylesheet xpath-default-namespace="records:2"
xmlns="reference-data:2"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:stroom="stroom"
xmlns:evt="event-logging:3"
version="2.0">

<xsl:template match="records">
<referenceData xmlns="reference-data:2"
xsi:schemaLocation="reference-data:2 file://reference-data-v2.0.xsd" version="2.0.1">
<!-- MAIN TEMPLATE -->
<xsl:template match="record">
<reference>
<map></map>
<key></key>
<value></value>
</reference>
</xsl:template>
</xsl:template>

```

And on pressing the refresh button



[Step Refresh Button](#)

we see that the *output* window is an empty ReferenceData element.

Screenshot of a software interface showing the XSLT translation process and output.

The top navigation bar shows multiple tabs: `GeoHost-V1.0-REFERENCE`, `* GeoHost-V1.0-REFERENCE`, `T GeoHost-V1.0-REFERENCE`, `XSL * GeoHost-V1.0-REFERENCE`, and `* GeoHost-V1.0-REFERENCE`. The status bar indicates the current step: [168533:1:1].

The main window displays the XSLT code and its execution flow:

```

Source → combinedParser → xsl translationFilter → xsl schemaFilter → xsl xmlWriter → streamAppender

```

The XSLT code is as follows:

```

1 <?xml version="1.1" encoding="UTF-8" ?>
2 <xsl:stylesheet xpath-default-namespace="records:2"
3   xmlns="reference-data:2"
4   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
5   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:stroom="stroom"
6   xmlns:evt="event-logging:3"
7   version="2.0">
8
9 <xsl:template match="records">
10 <referenceData xmlns="reference-data:2"
11   xsi:schemaLocation="reference-data:2 file://reference-data-v2.0.xsd" version="2.0.1">
12 <!-- MAIN TEMPLATE -->
13 <xsl:template match="record">
14   <reference>
15     <map></map>
16     <key></key>
17     <value></value>
18   </reference>
19 </xsl:template>
20 </xsl:template>
21 </xsl:template>
22 </xsl:stylesheet>

```

The output window shows the generated XML:

```

<?xml version="1.1" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<record>
<data name="FQDN" value="stroomnode00.strmdev00.org" />
<data name="IPAddress" value="192.168.2.245" />
<data name="Country" value="GBR" />
<data name="Site" value="Bristol-S00" />
<data name="Building" value="GZero" />
<data name="Room" value="R00" />
<data name="TimeZones" value="+00:00/+01:00" />
</record>
</records>

```

The right side of the output window shows the generated ReferenceData XML:

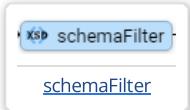
```

<xsl:stylesheet version="1.1" encoding="UTF-8">
<referenceData xmlns="reference-data:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<map />
<key />
<value />
</reference>
</referenceData>

```

The bottom status bar indicates the classification: **UNCLASSIFIED**.

Also note that if we move to the



icon on the pipeline structure, we no longer have an “Invalid Schema Location” error.

We next extend the translation to actually generate reference data. The translation will now look like

```

<?xml version="1.1" encoding="UTF-8" ?>
<xsl:stylesheet xpath-default-namespace="records:2"
xmlns="reference-data:2"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:stroom="stroom"
xmlns:evt="event-logging:3"
version="2.0">

<!--
GEOHOST REFERENCE FEED:

CHANGE HISTORY
v1.0.0 - 2020-02-09 John Doe

This is a reference feed for device Logical and Geographic data.

The feed provides for each device
* the device FQDN
* the device IP Address
* the device Country location (using ISO 3166-1 alpha-3 codes)
* the device Site location
* the device Building location
* the device Room location
*the device TimeZone location (both standard then daylight timezone offsets from UTC)

The reference maps are
FQDN_TO_IP - Fully Qualified Domain Name to IP Address
IP_TO_FQDN - IP Address to FQDN (HostName)
FQDN_TO_LOC - Fully Qualified Domain Name to Location element
-->

<xsl:template match="records">
<referenceData xmlns="reference-data:2"
xsi:schemaLocation="reference-data:2 file://reference-data-v2.0.xsd" version="2.0.1">
<xsl:apply-templates/>
</referenceData>
</xsl:template>

<!-- MAIN TEMPLATE -->
<xsl:template match="record">
<!-- FQDN_TO_IP map -->
<reference>
<map>FQDN_TO_IP</map>
<key>
<xsl:value-of select="lower-case(data[@name='FQDN']/@value)" />
</key>
<value>
<IPAddress>
<xsl:value-of select="data[@name='IPAddress']/@value" />
</IPAddress>
</value>
</reference>

<!-- IP_TO_FQDN map -->
<reference>
<map>IP_TO_FQDN</map>
<key>
<xsl:value-of select="lower-case(data[@name='IPAddress']/@value)" />
</key>
<value>
<HostName>
<xsl:value-of select="data[@name='FQDN']/@value" />
</HostName>
</value>
</reference>

```

```
</reference>
</xsl:template>
</xsl:stylesheet>
```

and when we refresh, by pressing the *Refresh Current Step* button



[Step Refresh Button](#)

we see that the *output* window now has *Reference* elements within the parent *ReferenceData* element

The screenshot shows the XSLT transformation process. The top navigation bar lists five tabs: 'GeoHost-V1.0-REFERENCE' (active), 'GeoHost-V1.0-REFERENCE', 'GeoHost-V1.0-REFERENCE', 'xsl * GeoHost-V1.0-REFERENCE', and 'H * GeoHost-V1.0-REFERENCE'. The status bar indicates '[168533:1:1]'. Below the tabs, a sequence of steps is shown: 'Source' → 'combinedParser' → 'xsl translationFilter' → 'schemaFilter' → 'xmlWriter' → 'streamAppender'. The 'xsl translationFilter' step displays the XSLT code:

```

<!-- IP_TO_FQDN map -->
<reference>
  <map>IP_TO_FQDN</map>
  <key>
    <xsl:value-of select="lower-case(data[@name='IPAddress'])/@value" />
  </key>
  <value>
    <HostName>
      <xsl:value-of select="data[@name='FQDN']/@value" />
    </HostName>
  </value>
</reference>

```

The 'xmlWriter' step shows the resulting XML output:

```

<?xml version="1.1" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <record>
    <data name="FQDN" value="stroomnode00.strmdev00.org" />
    <data name="IPAddress" value="192.168.2.245" />
    <data name="Country" value="GBR" />
    <data name="Site" value="Bristol-S00" />
    <data name="Building" value="GZero" />
    <data name="Room" value="R00" />
    <data name="TimeZones" value="+00:00/+01:00" />
  </record>
</records>

```

The 'streamAppender' step shows the final output:

```

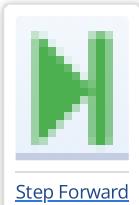
<?xml version="1.1" encoding="UTF-8"?>
<referenceData xmlns="reference-data:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <reference>
    <map>FQDN_TO_IP</map>
    <key>stroomnode00.strmdev00.org</key>
    <value>
      <IPAddress>192.168.2.245</IPAddress>
    </value>
  </reference>
  <reference>
    <map>IP_TO_FQDN</map>
    <key>192.168.2.245</key>
    <value>
      <HostName>stroomnode00.strmdev00.org</HostName>
    </value>
  </reference>
</referenceData>

```

The bottom status bar says 'UNCLASSIFIED'.

[xslt Translation - basic translation](#)

If we press the Step Forward button



[Step Forward](#)

we see the second *record* of our raw reference data in the *input* sub-pane and the resultant *Reference* elements

Source → T combinedParser → xsl translationFilter → xsl schemaFilter → xsl xmlWriter → streamAppender

```

47   <xsl:value-of select="lower-case(data[@name='FQDN']/@value)" />
48   </key>
49   <value>
50     <IPAddress>
51       <xsl:value-of select="data[@name='IPAddress']/@value" />
52     </IPAddress>
53   </value>
54 </reference>
55   <!-- IP_TO_FQDN map -->
56   <reference>
57     <map>IP_TO_FQDN</map>
58   <key>
59     <xsl:value-of select="lower-case(data[@name='IPAddress']/@value)" />
60   </key>
61   <value>
62     <HostName>
63       <xsl:value-of select="data[@name='FQDN']/@value" />

```

```

<?xml version="1.1" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<record>
<data name="FQDN" value="stroomnode01.strmdev01.org" />
<data name="IPAddress" value="192.168.3.117" />
<data name="Country" value="AUS" />
<data name="Site" value="Sydney-S04" />
<data name="Building" value="R6" />
<data name="Room" value="5-134" />
<data name="TimeZones" value="+10:00/+11:00" />
</record>
</records>

```

```

<?xml version="1.1" encoding="UTF-8"?>
<referenceData xmlns="reference-data:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<reference>
<map>FQDN_TO_IP</map>
<key>stroomnode01.strmdev01.org</key>
<value>
<IPAddress>192.168.3.117</IPAddress>
</value>
</reference>
<reference>
<map>IP_TO_FQDN</map>
<key>192.168.3.117</key>
<value>
<HostName>stroomnode01.strmdev01.org</HostName>
</value>
</reference>
</referenceData>

```

UNCLASSIFIED

[xslt Translation - basic translation next record](#)

At this point it would be wise to save our translation. This is done by clicking on the highlighted



icon in the top left-hand area of the window under the tabs.

We can now further our Reference by adding a Fully Qualified Domain Name to Location reference - FQDN_TO_LOC and so now the translation looks like

```

<?xml version="1.1" encoding="UTF-8" ?>
<xsl:stylesheet xpath-default-namespace="records:2"
xmlns="reference-data:2"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:stroom="stroom"
xmlns:evt="event-logging:3"
version="2.0">

<!--
GEOHOST REFERENCE FEED:

CHANGE HISTORY
v1.0.0 - 2020-02-09 John Doe

This is a reference feed for device Logical and Geographic data.

The feed provides for each device
* the device FQDN
* the device IP Address
* the device Country location (using ISO 3166-1 alpha-3 codes)
* the device Site location
* the device Building location
* the device Room location
*the device TimeZone location (both standard then daylight timezone offsets from UTC)

The reference maps are
FQDN_TO_IP - Fully Qualified Domain Name to IP Address
IP_TO_FQDN - IP Address to FQDN (HostName)
FQDN_TO_LOC - Fully Qualified Domain Name to Location element
-->

<xsl:template match="records">
<referenceData xmlns="reference-data:2"
xsi:schemaLocation="reference-data:2 file://reference-data-v2.0.xsd" version="2.0.1">
<xsl:apply-templates/>
</referenceData>
</xsl:template>

<!-- MAIN TEMPLATE -->
<xsl:template match="record">
<!-- FQDN_TO_IP map -->
<reference>
<map>FQDN_TO_IP</map>
<key>
<xsl:value-of select="lower-case(data[@name='FQDN']/@value)" />
</key>
<value>
<IPAddress>
<xsl:value-of select="data[@name='IPAddress']/@value" />
</IPAddress>
</value>
</reference>

<!-- IP_TO_FQDN map -->
<reference>
<map>IP_TO_FQDN</map>
<key>
<xsl:value-of select="lower-case(data[@name='IPAddress']/@value)" />
</key>
<value>
<HostName>
<xsl:value-of select="data[@name='FQDN']/@value" />
</HostName>
</value>
</reference>

```

```

</reference>

<!-- FQDN_TO_LOC map -->
<reference>
  <map>FQDN_TO_LOC</map>
  <key>
    <xsl:value-of select="lower-case(data[@name='FQDN']/@value)" />
  </key>
  <value>
    <!--
      Note, when mapping to a XML node set, we make use of the Event namespace - i.e. evt:
      defined on our stylesheet element. This is done, so that, when the node set is returned,
      it is within the correct namespace.
    -->
    <evt:Location>
      <evt:Country>
        <xsl:value-of select="data[@name='Country']/@value" />
      </evt:Country>
      <evt:Site>
        <xsl:value-of select="data[@name='Site']/@value" />
      </evt:Site>
      <evt:Building>
        <xsl:value-of select="data[@name='Building']/@value" />
      </evt:Building>
      <evt:Room>
        <xsl:value-of select="data[@name='Room']/@value" />
      </evt:Room>
      <evt:TimeZone>
        <xsl:value-of select="data[@name='TimeZones']/@value" />
      </evt:TimeZone>
    </evt:Location>
  </value>
</reference>
</xsl:template>
</xsl:stylesheet>

```

and our second ReferenceData element would now look like

```

<?xml version="1.0" encoding="UTF-8"?>
<referenceData xmlns="reference-data:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <reference>
    <map>FQDN_TO_IP</map>
    <key>stroomnode01.strmdev01.org</key>
    <value>
      <IPAddress>192.168.3.117</IPAddress>
    </value>
  </reference>
  <reference>
    <map>IP_TO_FQDN</map>
    <key>192.168.3.117</key>
    <value>
      <HostName>stroomnode01.strmdev01.org</HostName>
    </value>
  </reference>
  <reference>
    <map>FQDN_TO_LOC</map>
    <key>stroomnode01.strmdev01.org</key>
    <value>
      <evt:Location>
        <evt:Country>AUS</evt:Country>
        <evt:Site>Sydney-S04</evt:Site>
        <evt:Building>R6</evt:Building>
        <evt:Room>5-134</evt:Room>
        <evt:TimeZone>+10:00/+11:00</evt:TimeZone>
      </evt:Location>
    </value>
  </reference>
</referenceData>

```

[xslt Translation - complete translation 2nd record](#)

We have completed the translation and have hence completed the development of our GeoHost-V1.0-REFERENCE reference feed.

At this point, the reference feed is set up to accept Raw Reference data, but it will not automatically process the raw data and hence it will not place reference data into the reference data store. To have Stroom automatically process Raw Reference streams, you will need to enable *Processors* for this pipeline.

4.2.1.2.9 Enabling the Reference Feed Processors

We now create the pipeline Processors for this feed, so that the raw reference data will be transformed into Reference Data on ingest and save to Reference Data stores.

Open the reference feed pipeline by selecting the GeoHost-V1.0-REFERENCE pipeline



tab to raise the GeoHost-V1.0-REFERENCE pipeline. Then select the **Processors** sub-item to show

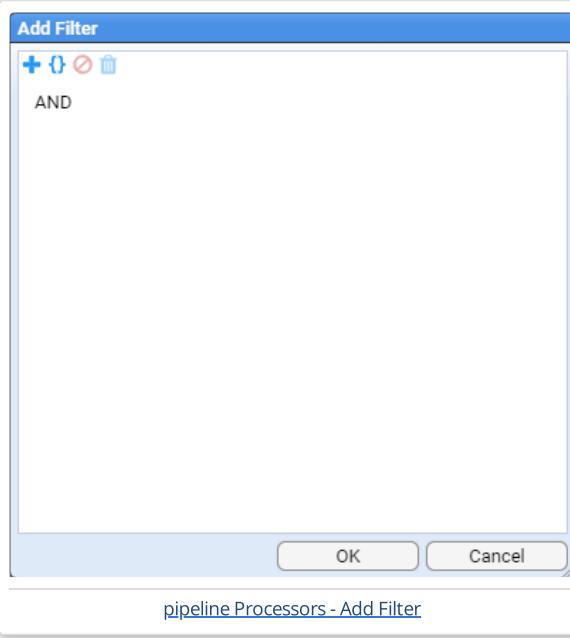
A screenshot of a software interface titled 'GeoHost-V1.0-REFERENCE'. The main area shows a table with columns: Pipeline, Tracker Ms, Tracker %, Last Poll Age, Task Count, Priority, Streams, and Events. The first row is highlighted with a blue background. Below the table, there is a link labeled 'pipeline Processors'.

This configuration tab is divided into two panes. The top pane shows the current enabled Processors and any recently processed streams and the bottom pane provides meta-data about each Processor or recently processed streams.

First, move the mouse to the Add Processor



icon at the top left of the top pane. Select by left clicking this icon to have displayed the **Add Filter** selection window



This selection window allows us to *filter* what set of data streams we want our Processor to process. As our intent is to enable processing for all GeoHost-V1.0-REFERENCE streams, both already received and yet to be received, then our filtering criteria is just to process all Raw Reference for this feed, ignoring all other conditions.

To do this, first click on the **Add Term**



icon to navigate to the desired feed name (GeoHost-V1.0-REFERENCE) object



[pipeline Processors - Choose Feed name](#)

and press **OK** to make the selection.

Next, we select the required *stream type*. To do this click on the **Add Term**



icon again. Click on the down arrow to change the Term selection from *Feed* to *Type*. Click in the **Value** position on the highlighted line (it will be currently empty). Once you have clicked here a drop-down box will appear as per



[pipeline Processors - Choose Stream Type](#)

at which point, select the *Stream Type* of **Raw Reference** and then press **OK**. At this we return to the **Add Processor** selection window to see that the *Raw Reference* stream type has been added.

GeoHost-V1.0-REFERENCE x GeoHost-V1.0-REFERENCE x

Data | Structure | Processors | Active Tasks | Settings

Pipeline	Tracker Ms	Tracker %	Last Poll Age	Task Count	Priority	Streams	Events	Status	Enabled
GeoHost-V1.0-REFERENCE									<input type="checkbox"/>
GeoHost-V1.0-REFERENCE									<input type="checkbox"/>

AND { Feed is GeoHost-V1.0-REFERENCE
Type = Raw Reference }

[pipeline Processors - pipeline criteria set](#)

Note the Processor has been added but it is in a **disabled** state. We **enable** both pipeline processor and the processor filter

GeoHost-V1.0-REFERENCE x GeoHost-V1.0-REFERENCE x

Data | Structure | Processors | Active Tasks | Settings

Pipeline	Tracker Ms	Tracker %	Last Poll Age	Task Count	Priority	Streams	Events	Status	Enabled
GeoHost-V1.0-REFERENCE									<input checked="" type="checkbox"/>
GeoHost-V1.0-REFERENCE									<input checked="" type="checkbox"/>

AND { Feed is GeoHost-V1.0-REFERENCE
Type = Raw Reference }

[pipeline Processors - Enable](#)

Note - if this is the first time you have set up pipeline processing on your Stroom instance you may need to check that the **Stream Processor** job is enabled on your Stroom instance. To do this go to the Stroom main menu and select *Monitoring>Jobs>* Check the status of the Stream Processor job and enable if required. If you need to enable the job also ensure you enable the job on the individual nodes as well (go to the bottom window pane and select the enable box on the far right)

Item Tools Monitoring User Help

Explorer Database Tables GeoHost-V1.0-REFERENCE x GeoHost-V1.0-REFERENCE x Jobs x

Quick Jobs

System

- Nodes
- Volume Clean
- Enabled
- Description
- Job to process a volume deleting files that are no longer indexed
- Status
- Enabled
- Job to record status of node (CPU and Memory usage)
- Aggregation
- Enabled
- Job to pick up the data written by the proxy and store it in Stroom
- GeoHost-V1.0-REFERENCE
- Query History Clean
- Enabled
- Job to clean up old query history items
- Stream Processor
- Enabled
- Job to process streams matching stream processor filters with t
- Volume Status
- Enabled
- Update the usage status of volumes owned by the node
- Internal Dashboards
- Data Retention
- Enabled
- Delete data that exceeds the retention period specified by data re
- Internal Logs
- Index Shard Delete
- Enabled
- Job to delete index shards from disk that have been marked as d
- Internal Statistics
- Index Shard Retention
- Enabled
- Job to set index shards to have a status of deleted that have pas
- Standard Pipelines
- Index Writer Cache Sweep
- Enabled
- Job to remove old index shard writers from the cache
- Stroom 101
- Index Writer Flush
- Enabled
- Job to flush index shard data to disk
- Template Pipelines
- Java Heap Histogram Statistics
- Enabled
- Generate Java heap map histogram and record statistic events f
- Testing
- Pipeline Destination Roll
- Enabled
- Roll any destinations based on their roll settings
- ApacheHttpd
- Property Cache Reload
- Enabled
- Reload properties in the cluster
- Infrastructure
- Solr Index Optimise
- Enabled
- Optimise Solr indexes
- GEOHOST-REFERENCE
- Solr Index Retention
- Enabled
- Logically delete indexed documents in Solr indexes based on the
- GEOHOST-REFERENCE
- SQL Stats Database Aggregation
- Enabled
- Run SQL stats database aggregation
- GEOHOST-REFERENCE
- SOL Stats In Memory Flush
- Enabled
- SOL Stats In Memory Flush (Cache to DR)

[pipeline Processors - Enable node processing](#)

GeoHost-V1.0-REFERENCE x GeoHost-V1.0-REFERENCE x Jobs x 1 to 25 of 25

Job	Enabled	Description
File System Clean	<input checked="" type="checkbox"/>	Job to process a volume deleting files that are no longer indexed (maybe the retention period has past or they have been deleted)
Node Status	<input checked="" type="checkbox"/>	Job to record status of node (CPU and Memory usage)
Proxy Aggregation	<input checked="" type="checkbox"/>	Job to pick up the data written by the proxy and store it in Stroom
Query History Clean	<input checked="" type="checkbox"/>	Job to clean up old query history items
Stream Processor	<input checked="" type="checkbox"/>	Job to process streams matching stream processor filters with their associated pipelines
Volume Status	<input checked="" type="checkbox"/>	Update the usage status of volumes owned by the node
Data Retention	<input checked="" type="checkbox"/>	Delete data that exceeds the retention period specified by data retention policy
Index Shard Delete	<input checked="" type="checkbox"/>	Job to delete index shards from disk that have been marked as deleted
Index Shard Retention	<input checked="" type="checkbox"/>	Job to set index shards to have a status of deleted that have past their retention period
Index Writer Cache Sweep	<input checked="" type="checkbox"/>	Job to remove old index shard writers from the cache

Job	Node	Type	Max	Cur	Last Executed	Enabled
Stream Processor	node1a	Distributed	20	0	2020-03-10T07:07:07.689Z	<input checked="" type="checkbox"/>

1 to 1 of 1

[pipeline Processors - Enable](#)

Returning to the

GeoHost-V1.0-REFERENCE

[GeoHost-V1.0-REFERENCE Pipeline](#)

tab, **Processors** sub-item, if everything is working on your Stroom instance you should now see that Raw Reference streams are being processed by your processor - the **Streams** count is incrementing and the **Tracker%** is incrementing (when the Tracker% is 100% then all streams you selected (Filtered for) have been processed)

GeoHost-V1.0-REFERENCE x GeoHost-V1.0-REFERENCE x Data | Structure | Processors | Active Tasks | Settings 1 to 2 of 2

Pipeline	Tracker Ms	Tracker %	Last Poll Age	Task Count	Priority	Streams	Events	Status	Enabled
GeoHost-V1.0-REFERENCE	2020-03-10T07:07:09.353Z	100	5.1m	0	10	10	1	OK	<input checked="" type="checkbox"/>
GeoHost-V1.0-REFERENCE	2020-03-10T07:07:09.353Z	100	5.1m	0	10	10	1	OK	<input checked="" type="checkbox"/>

[pipeline Processors - Enable](#)

Navigating back to the **Data** sub-item and clicking on the reference feed stream in the *Stream Table* we see

GeoHost-V1.0-REFERENCE x GeoHost-V1.0-REFERENCE x Data | Structure | Processors | Active Tasks | Settings

[Save As](#)

Type	Feed	Pipeline
Reference	GeoHost-V1.0-REFERENCE	GeoHost-V1.0-REFERENCE

Created	Type	Feed	Pipeline
2020-03-10T07:07:08.829Z	Raw Reference	GeoHost-V1.0-REFERENCE	GeoHost-V1.0-REFERENCE
2020-03-10T07:07:08.829Z	Reference	GeoHost-V1.0-REFERENCE	GeoHost-V1.0-REFERENCE

Data

```

1 <?xml version="1.1" encoding="UTF-8"?>
2 <referenceData xmlns="reference-data:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:stroom="stroom" xmlns:evt="e
3   <reference>
4     <map>FQDN_TO_IP</map>
5     <key>stroomnode00.strmdev00.org</key>
6     <value>
7       <IPAddress>192.168.2.245</IPAddress>
8     </value>
9   </reference>
10  <reference>
11    <map>IP_TO_FQDN</map>
12    <key>192.168.2.245</key>
13    <value>
14      <HostName>stroomnode00.strmdev00.org</HostName>
15    </value>
16  </reference>
17  <reference>
18    <map>FQDN_TO_LOC</map>
19    <key>stroomnode00.strmdev00.org</key>
20    <value>
21      <evt:Location>
22        <evt:Country>GBR</evt:Country>
23        <evt:Site>Bristol-S00</evt:Site>
24        <evt:Building>GZero</evt:Building>
25

```

UNCLASSIFIED

In the top pane, we see the *Streams table* as per normal, but in the *Specific stream* table we see that we have both a **Raw Reference** stream and its child **Reference** stream. By clicking on and highlighting the **Reference** stream we see its content in the bottom pane.

The complete ReferenceData for this stream is

```
<?xml version="1.1" encoding="UTF-8"?>
<referenceData xmlns="reference-data:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:stroom="stroom" xmlns:evt="e
<reference>
<map>FQDN_TO_IP</map>
<key>stroomnode00.strmdev00.org</key>
<value>
<IPAddress>192.168.2.245</IPAddress>
</value>
</reference>
<reference>
<map>IP_TO_FQDN</map>
<key>192.168.2.245</key>
<value>
<HostName>stroomnode00.strmdev00.org</HostName>
</value>
</reference>
<reference>
<map>FQDN_TO_LOC</map>
<key>stroomnode00.strmdev00.org</key>
<value>
<evt:Location>
<evt:Country>GBR</evt:Country>
<evt:Site>Bristol-S00</evt:Site>
<evt:Building>GZero</evt:Building>
<evt:Room>R00</evt:Room>
<evt:TimeZone>+00:00/+01:00</evt:TimeZone>
</evt:Location>
</value>
</reference>
<reference>
<map>FQDN_TO_IP</map>
<key>stroomnode01.strmdev01.org</key>
<value>
<IPAddress>192.168.3.117</IPAddress>
</value>
</reference>
<reference>
<map>IP_TO_FQDN</map>
<key>192.168.3.117</key>
<value>
<HostName>stroomnode01.strmdev01.org</HostName>
</value>
</reference>
<reference>
<map>FQDN_TO_LOC</map>
<key>stroomnode01.strmdev01.org</key>
<value>
<evt:Location>
<evt:Country>AUS</evt:Country>
<evt:Site>Sydney-S04</evt:Site>
<evt:Building>R6</evt:Building>
<evt:Room>5-134</evt:Room>
<evt:TimeZone>+10:00/+11:00</evt:TimeZone>
</evt:Location>
</value>
</reference>
<reference>
<map>FQDN_TO_IP</map>
<key>host01.company4.org</key>
<value>
<IPAddress>192.168.4.220</IPAddress>
</value>
</reference>
<reference>
<map>IP_TO_FQDN</map>
```

```

<key>192.168.4.220</key>
<value>
  <HostName>host01.company4.org</HostName>
</value>
</reference>
<reference>
  <map>FQDN_TO_LOC</map>
  <key>host01.company4.org</key>
  <value>
    <evt:Location>
      <evt:Country>USA</evt:Country>
      <evt:Site>LosAngeles-S19</evt:Site>
      <evt:Building>ILM</evt:Building>
      <evt:Room>C5-54-2</evt:Room>
      <evt:TimeZone>-08:00/-07:00</evt:TimeZone>
    </evt:Location>
  </value>
</reference>
</referenceData>

```

If we go back to the reference feed itself (and click on the



button on the far right of the top and middle panes), we now see both the *Reference* and *Raw Reference* streams in the *Streams Table* pane.

Created	Type	Feed	Pipeline
2020-03-10T07:07:08.829Z	Reference	GeoHost-V1.0-REFERENCE	GeoHost-V1.0-REFERENCE
2020-03-10T00:29:46.899Z	Raw Reference	GeoHost-V1.0-REFERENCE	

Created	Type	Feed	Pipeline
> 2020-03-10T00:29:46.899Z	Raw Reference	GeoHost-V1.0-REFERENCE	
> 2020-03-10T07:07:08.829Z	Reference	GeoHost-V1.0-REFERENCE	GeoHost-V1.0-REFERENCE

[reference feed - Data tab](#)

Selecting the *Reference* stream in the *Stream Table* will result in the *Specific stream* pane displaying the *Raw Reference* and its child *Reference* stream (highlighted) and the actual *ReferenceData* output in the *Data* pane at the bottom.

GeoHost-V1.0-REFERENCE x GeoHost-V1.0-REFERENCE x

Data | Active Tasks | Settings

Created Type Feed Pipeline

2020-03-10T07:07:08.829Z Reference GeoHost-V1.0-REFERENCE GeoHost-V1.0-REFERENCE

2020-03-10T00:29:46.899Z Raw Reference GeoHost-V1.0-REFERENCE GeoHost-V1.0-REFERENCE

2020-03-10T00:29:46.899Z Raw Reference GeoHost-V1.0-REFERENCE Pipeline

2020-03-10T07:07:08.829Z Reference GeoHost-V1.0-REFERENCE GeoHost-V1.0-REFERENCE

Data

```

1  <?xml version="1.1" encoding="UTF-8"?>
2  <referenceData xmlns="reference-data:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:stroom="stroom" xmlns:evt="ev
3   <reference>
4     <map>FQDN_TO_IP</map>
5     <key>stroomnode00.strmdev00.org</key>
6     <value>
7       <IPAddress>192.168.2.245</IPAddress>
8     </value>
9   </reference>
10  <reference>
11    <map>IP_TO_FQDN</map>
12    <key>192.168.2.245</key>
13    <value>
14      <HostName>stroomnode00.strmdev00.org</HostName>
15    </value>
16  </reference>
17  <reference>
18    <map>FQDN_TO_LOC</map>
19    <key>stroomnode00.strmdev00.org</key>
20    <value>
21      <evt:Location>
22        <evt:Country>GBR</evt:Country>
23        <evt:Site>Bristol-S00</evt:Site>
24

```

UNCLASSIFIED

[reference feed - Select reference](#)

Selecting the *Raw Reference* stream in the *Streams Table* will result in the *Specific stream* pane displaying the *Raw Reference* and its child *Reference* stream as before, but with the *Raw Reference* stream highlighted and the actual Raw Reference input data displayed in the *Data* pane at the bottom.

GeoHost-V1.0-REFERENCE x GeoHost-V1.0-REFERENCE x

Data | Active Tasks | Settings

Created Type Feed Pipeline

2020-03-10T07:07:08.829Z Reference GeoHost-V1.0-REFERENCE GeoHost-V1.0-REFERENCE

2020-03-10T00:29:46.899Z Raw Reference GeoHost-V1.0-REFERENCE GeoHost-V1.0-REFERENCE

2020-03-10T00:29:46.899Z Raw Reference GeoHost-V1.0-REFERENCE Pipeline

2020-03-10T07:07:08.829Z Reference GeoHost-V1.0-REFERENCE GeoHost-V1.0-REFERENCE

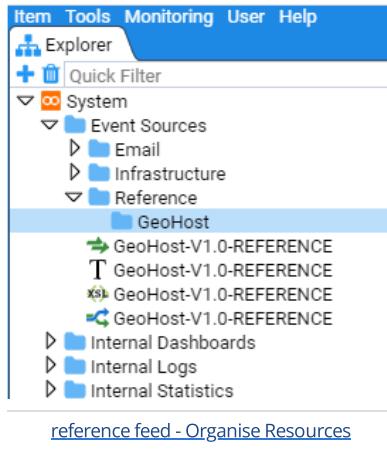
Data | Meta

FQDN	IPAddress	Country	Site	Building	Room	TimeZones
stroomnode00.strmdev00.org	192.168.2.245	GBR	Bristol-S00	GZero	R00	+00:00/+01:00
stroomnode01.strmdev01.org	192.168.3.117	AUS	Sydney-S04	R6	5-134	+10:00/+11:00
host01.company4.org	192.168.4.220	USA	LosAngeles-S19	ILM	C5-54-2	-08:00/-07:00
5						

[reference feed - Select raw reference](#)

The creation of the *Raw Reference* is now complete.

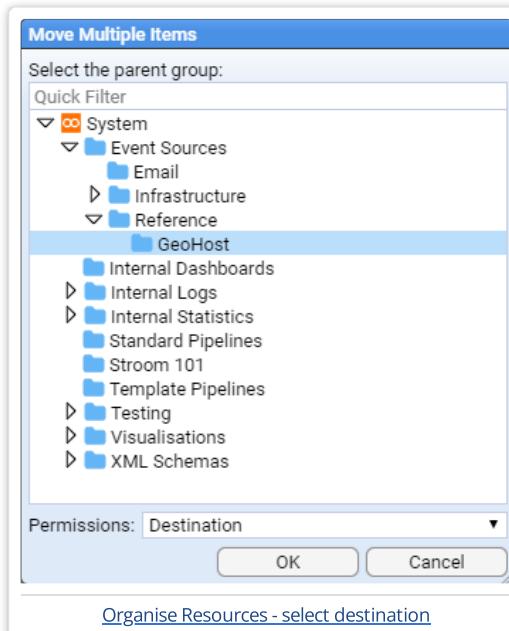
At this point you may wish to organise the resources you have created within the Explorer pane to a more appropriate location such as *Reference/GeoHost*. Because Stroom Explorer is a flat structure you can move resources around to reorganise the content without any impact on directory paths, configurations etc.



Now you have created the new folder structure you can move the various GeoHost resources to this location. Select all four resources by using the mouse right-click button while holding down the *Shift* key. Then right click on the highlighted group to display the action menu



Select **move** and the *Move Multiple Items* window will display. Navigate to the `Reference/GeoHost` folder to move the items to this destination.



The final structure is seen below





4.3 - Search

4.3.1 - Elasticsearch integration

How to integrate Stroom with Elastic Search

4.3.1.1 Introduction

Stroom v6.1 can pass data to Elasticsearch for indexing. Indices created using this process (i.e. those containing a `StreamId` and `EventId` corresponding to a particular Stroom instance) are searchable via a Stroom *dashboard*, much like a Stroom Lucene index.

This integration provides operators with the flexibility to utilise the additional capabilities of Elasticsearch, (like clustering and replication) and expose indexed data for consumption by external analytic or processing tools.

This guide will take you through creating an Elasticsearch index, setting up an indexing pipeline, activating a stream processor and searching the indexed data in both Stroom and Elasticsearch.

4.3.1.1.1 Assumptions

1. You have created an Elasticsearch cluster. For test purposes, you can quickly create a single-node cluster using Docker by following the steps in the [Elasticsearch Docs \(external link\)](#).
2. The Elasticsearch cluster is reachable via HTTP/S from all Stroom nodes participating in [stream processing](#).
3. Elasticsearch security is disabled.
4. You have a feed containing `Event` data.

4.3.1.1.2 Key differences

1. Unlike with [Solr indexing](#), Elasticsearch field mappings are managed outside of Stroom, usually via the [REST API \(external link\)](#).
2. Aside from creating the mandatory `StreamId` and `EventId` field mappings, explicitly defining mappings for other fields is optional. It is however, considered good practice to define these mappings, to ensure each field's data type is correctly parsed and represented. For text fields, it also pays to ensure that the [appropriate mapping parameters are used \(external link\)](#), in order to satisfy your search and analysis requirements - and meet system resource constraints.
3. Unlike both Solr and Lucene indexing, it is not necessary to mark a field as `stored` (i.e. storing its raw value in the inverted index). This is because Elasticsearch stores the content of the original document in the [_source field \(external link\)](#), which is retrieved when populating search results. Provided the `_source` field is enabled (as it is by default), a field is treated as `stored` in Stroom and its value doesn't need to be retrieved via an *extraction pipeline*.

4.3.1.2 Indexing data

4.3.1.2.1 Creating an index in Elasticsearch

The following cURL command creates an index named `stroom_test` in Elasticsearch cluster `http://localhost:9200` consisting of the following fields:

1. `StreamId` (mandatory, must be of data type `long`)
2. `EventId` (mandatory, must also be `long`)
3. `Name` (text). Uses the default analyzer, which tokenizes the text for matching on terms. `fielddata` is enabled, which allows for [aggregating on these terms \(external link\)](#).
4. `State` (keyword). Supports exact matching.

The created index consists of 5 shards. Note that the shard count cannot be changed after index creation, without a reindex. See [this guide \(external link\)](#) on shard sizing.

```

curl -X PUT "http://localhost:9200/stroom_test?pretty" -H 'Content-Type: application/json' -d'
{
  "settings": {
    "number_of_shards": 5
  },
  "mappings": {
    "properties": {
      "StreamId": {
        "type": "long"
      },
      "EventId": {
        "type": "long"
      },
      "Name": {
        "type": "text",
        "fielddata": true
      },
      "State": {
        "type": "text",
        "fielddata": true
      }
    }
  }
}
'

```

After creating the index, you can add additional field mappings. Note the [limitations \(external link\)](#) in doing so, particularly the fact that it will not cause existing documents to be re-indexed. It is worthwhile to test index mappings on a subset of data before committing to indexing a large event feed, to ensure the resulting search experience meets your requirements.

4.3.1.2.2 Registering the index in Stroom

This step creates an *Elasticsearch Index* in the *Stroom Tree* and tells Stroom how to connect to your Elasticsearch cluster and index. Note that this process needs to be repeated for each index you create.

4.3.1.2.2.1 Steps

1. Right-click on the folder in the *Explorer Tree* where you wish to create the index
2. Select `New / Elasticsearch Index`
3. Enter a valid name for the index. It is a good idea to choose one that reflects either the feed name being indexed, or if indexing multiple feeds, the nature of data they represent.
4. In the index tab that just opened:
 1. Select the `Settings` tab
 2. Set the `Index` to the name of the index in Elasticsearch (e.g. `stroom_test` from the previous example)
 3. Set the `Connection URLs` to one or more Elasticsearch node URLs. If multiple, separate each URL with `,`. For example, a URL like `http://data-0.elastic:9200,http://data-1.elastic:9200` will balance requests to two data nodes within an Elasticsearch cluster. See [this document](#) for guidance on node roles.
 4. Click `Test Connection`. If the connection succeeds, and the index is found, a dialog is shown indicating the test was successful. Otherwise, an error message is displayed.
 5. If the test succeeded, click the `save` button in the top-left. The `Fields` tab will now be populated with fields from the Elasticsearch index.

Note

The field mappings list is only updated when index settings are changed, or a Stroom indexing or search task begins. The refresh button in the `Fields` tab does not have any effect.

4.3.1.2.3 Setting index retention

As with Solr indexing, index document retention is determined by defining a Stroom query.

Setting a retention query is optional and by default, documents will be retained in an index indefinitely.

It is recommended for indices containing events spanning long periods of time, that [Elasticsearch Index Lifecycle Management \(external link\)](#) be used instead. The capabilities provided, such as automatic rollover to warm or cold storage tiers, are well worth considering, especially in high-volume production clusters.

4.3.1.2.3.1 Considerations when implementing ILM

1. It is recommended that [data streams](#) are used when indexing data. These allow easier rollover and work well with ILM policies. A *data stream* is essentially a container for multiple date-based indices and to a search client such as Stroom, appears and is searchable like a normal Elasticsearch index.
2. Use of *data streams* requires that a `@timestamp` field of type `date` be defined for each document (instead of say, `EventTime`).
3. Implementing ILM policies requires careful capacity planning, including anticipating search and retention requirements.

4.3.1.2.4 Creating an indexing pipeline

As with Lucene and Solr indexing pipelines, indexing data using Elasticsearch uses a pipeline filter. This filter accepts `<record>` elements and for each, sends a document to Elasticsearch for indexing.

Each `<data>` element contained within a `<record>` sets the document field name and value. You should ensure the `name` attribute of each `<data>` element exactly matches the mapping property of the Elasticsearch index you created.

4.3.1.2.4.1 Steps

1. Create a pipeline inheriting from the built-in `Indexing` template.
2. Modify the `xsltFilter` pipeline stage to output the correct `<records>` XML (see the [Quick-Start Guide](#)).
3. Delete the default `indexingFilter` and in its place, create an `ElasticIndexingFilter` (see screenshot below).
4. Review and set the following properties:
 1. `batchSize` (default: `10,000`). Number of documents to send in a single request to the Elasticsearch [Bulk API\(external link\)](#). Should usually be set to `1,000` or more. The higher the number, the more memory is required by both Stroom and Elasticsearch when sending or receiving the request.
 2. `index` (required). Set this to the target Elasticsearch index in the Stroom *Explorer Tree*.
 3. `refreshAfterEachBatch` (default: `false`). Refreshes the Elasticsearch index after each batch has finished processing. This makes any documents ingested in the batch available for searching. Unless search results are needed in near-real-time, it is recommended this be set to `false` and the index refresh interval be set to an appropriate value. See [this document \(external link\)](#) for guidance on optimising indexing performance.

The screenshot shows the Stroom Explorer interface with the 'Indexing' pipeline selected. The left sidebar shows various pipelines and feeds. The main panel displays the pipeline structure and its properties.

Pipeline Structure:

```
graph LR; Source --> xslxmlParser[xsl xmlParser]; xslxmlParser --> readRecordCountFilter[readRecordCountFilter]; readRecordCountFilter --> splitFilter[splitFilter]; splitFilter --> idEnrichmentFilter[idEnrichmentFilter]; idEnrichmentFilter --> xsltFilter[xsltFilter]; xsltFilter --> elasticIndexingFilter[elasticIndexingFilter]
```

Properties Table:

Property Name	Value	Source	Inherited Value	Inherited From	Default Value	Description
batchSize	10000	Inherit	10000		10000	How many documents to send to the index in a single post
index	TEST	Local				The index to send records to
refreshAfterEachBatch	true	Inherit	true		true	Refresh the index after each batch is processed, making the

Bottom Navigation:

1 to 3 of 3 | << < > >> | 1 to 1 of 0 | << < > >> |

4.3.1.2.5 Creating and activating a stream processor

Follow the steps as in [this guide](#).

4.3.1.2.6 Checking data has been indexed

Query Elasticsearch, checking the fields you expect are there, and of the correct data type:

The following query displays five results:

```
curl -X GET "http://localhost:9200/stroom_test/_search?size=5"
```

You can also get an exact document count, to ensure this matches the number of events you are expecting:

```
curl -X GET "http://localhost:9200/stroom_test/_count"
```

For more information, see the Elasticsearch [Search API documentation \(external link\)](#).

4.3.1.2.7 Reindexing data

By default, the original document values are stored in an Elasticsearch index and may be used later on to re-index data (such as when a change is made to field mappings). This is done via the [Reindex API\(external link\)](#). Provided these values have not changed, it would likely be more efficient to use this API to perform a re-index, instead of processing data from scratch using a Stroom stream processor.

On the other hand, if the content of documents being output to Elasticsearch has changed, the Elasticsearch index will need to be re-created and the stream re-processed. Examples of where this would be required include:

1. A new field is added to the indexing filter, which previously didn't exist. That field needs to be searchable for all historical events.
2. A field is renamed
3. A field data type is changed

If a field is omitted from the indexing translation, there is no need for a re-index, unless you wish to reclaim the space occupied by that field.

4.3.1.2.7.1 Reindexing using a pipeline processor

1. Delete the index. While it is possible to [delete by query\(external link\)](#), it is more efficient to drop the index. Additionally, deleting by query doesn't actually remove data from disk, until segments are merged.

```
curl -X DELETE "http://localhost:9200/stroom_test"
```

2. Re-create the index (as shown earlier)
3. Create a new pipeline processor to index the documents

4.3.1.3 Searching

Once indexed in Elasticsearch, you can search either using the *Stroom Dashboard* user interface, or directly against the Elasticsearch cluster.

The advantage of using Stroom to search is that it allows access to the raw source data (i.e. it is not limited to what's stored in the index). It can also use *extraction pipelines* to enrich search results for export in a table.

Elasticsearch on the other hand, provides a rich [Search REST API\(external link\)](#) with powerful [aggregations](#) that can be used to generate reports and discover patterns and anomalies. It can also be readily queried using third-party tools.

4.3.1.3.1 Stroom

See the [Dashboard](#) page in the Quick-Start Guide.

Instead of selecting a Lucene index, set the target *data source* to the desired Elasticsearch index in the Stroom *Explorer Tree*.

Once the target *data source* has been set, the Dashboard can be used as with a Lucene or Solr index *data source*.

4.3.1.3.2 Elasticsearch

Elasticsearch queries can be performed directly against the cluster using the [Search API](#)(external link).

Alternatively, there are tools that make search and discovery easier and more intuitive, like [Kibana](#)(external link).

4.3.1.4 Security

It is important to note that Elasticsearch data is not encrypted at rest, unless this feature is enabled and the relevant [licensing tier](#)(external link) is purchased. Therefore, appropriate measures should be taken to control access to Elasticsearch user data at the file level.

For production clusters, the Elasticsearch [security guidelines](#)(external link) should be followed, in order to control access and ensure requests are audited.

You might want to consider implementing [role-based access control](#)(external link) to prevent unauthorised users of the native Elasticsearch API or tools like Kibana, from creating, modifying or deleting data within sensitive indices.

4.3.2 - Search API

Stroom v6 introduced an API that allows a user to perform queries against Stroom resources such as indices and statistics. This is a guide to show how to perform a Stroom Query directly from bash using Stroom v7.

1. Create an API Key for yourself, this will allow the API to authenticate as you and run the query with your privileges.
2. Create a Dashboard that extracts the data you are interested in. You should create a Query and Table.
3. Download the JSON for your Query. Press the download icon in the Query Pane to generate a file containing the JSON. Save the JSON to a file named *query.json*.
4. Use curl to send the query to Stroom.

```
API_KEY='<put your API Key here'
URI=stroom.host/api/searchable/v2/search
curl -s --request POST ${URL} -o response.out -H "Authorization:Bearer ${API_KEY}" -H "Content-Type: application/json" --data @query.json
```

5. The query response should be in a file named *response.out*.

6. Optional step: reformat the response to csv using `jq`.

```
cat response.out | jq '.results[0].rows[].values | @csv'
```

4.3.3 - Solr integration

This document will show how to use Solr from within Stroom. A single Solr node will be used running in a docker container.

4.3.3.1 Assumptions

1. You are familiar with Lucene indexing within Stroom
2. You have some data to index

4.3.3.2 Points to note

1. A Solr core is the home for exactly one Stroom index.
2. Cores must initially be created in Solr.
3. It is good practice to name your Solr core the same as your Stroom Index.

4.3.3.3 Method

1. Start a docker container for a single solr node.

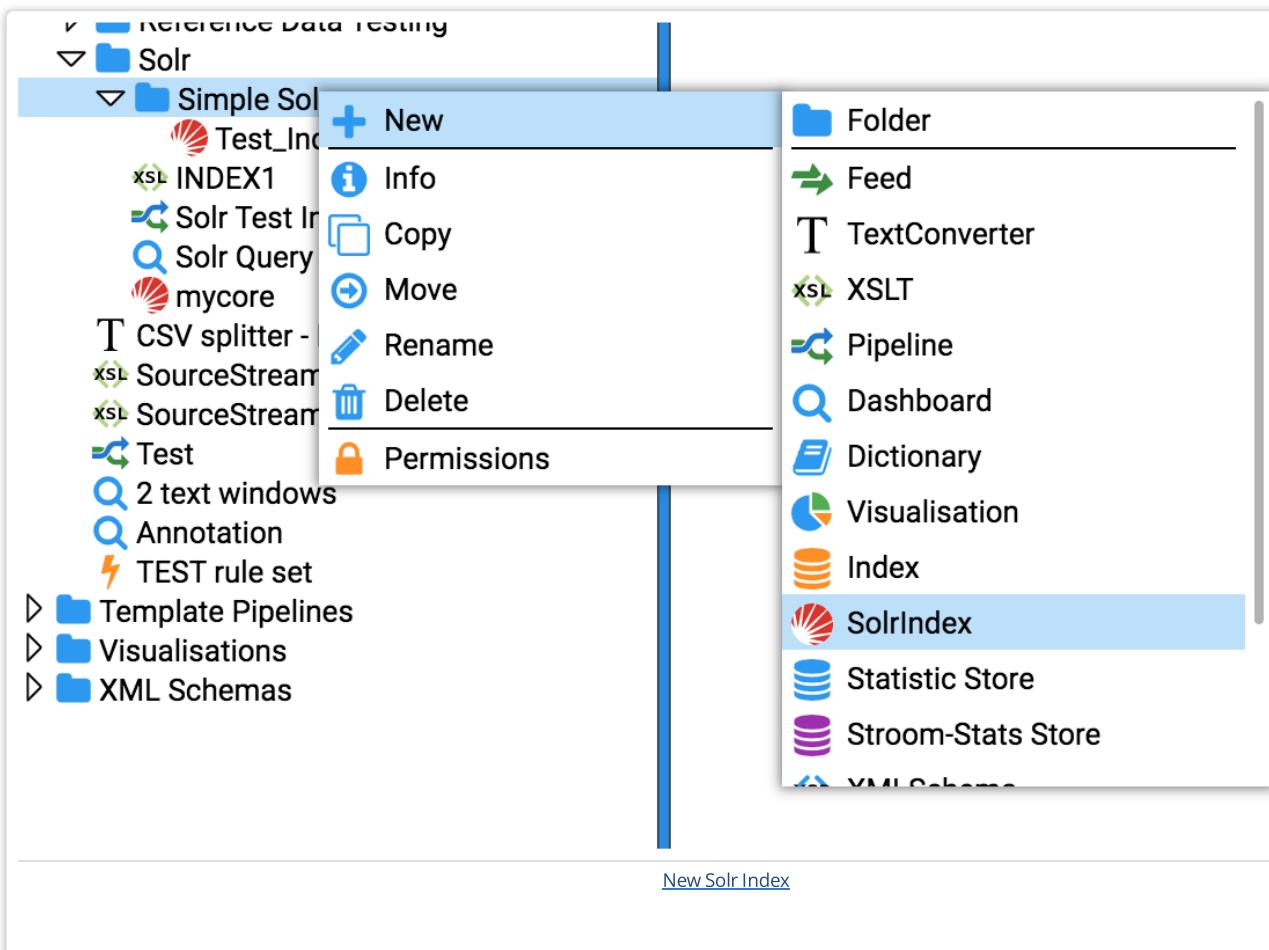
```
docker run -d -p 8983:8983 --name my_solr solr
```

2. Check your Solr node. Point your browser at http://yourSolrHost:8983

3. Create a core in Solr using the CLI.

```
docker exec -it my_solr solr create_core -c test_index
```

4. Create a SolrIndex in Stroom



5. Update settings for your new Solr Index in Stroom then press "Test Connection". If successful then press Save. Note the "Solr URL" field is a reference to the newly created Solr core.

The screenshot shows the Stroom interface with the 'Fields' tab selected. On the left, there's a configuration panel for a Solr index named 'Test_Index'. It includes fields for 'Collection' (set to 'Single Node'), 'Solr URLs' (containing the URL 'http://ec2-18-215-111-1.compute.amazonaws.com:8983/solr/test_index'), and options for 'Use ZK' and 'ZK Hosts'. Below these are sections for 'Data Retention Expression' and 'Solr Index Settings'. On the right, a modal window titled 'Alert' is open, displaying a success message with detailed log information. The log details the request and response headers and parameters. At the bottom right of the modal is a 'Close' button.

6. Add some Index fields. e.g. EventTime, UserId

7. Retention is different in Solr, you must specify an expression that matches data that can be deleted.

The screenshot shows the 'Data Retention Expression' editor. It displays the expression 'EventTime < month()-3M' under the 'AND' condition. Above the expression are icons for adding, removing, and clearing filters. Below the expression is a link labeled 'Solr Retention'.

8. Your Solr Index can now be used as per a Stroom Lucene Index. However, your Indexing pipeline must use a SolrIndexingFilter instead of an IndexingFilter.

4.4 - Administration

4.4.1 - System Properties

This HOWTO is provided to assist users in managing Stroom **System Properties** via the User Interface.

4.4.1.1 Assumptions

The following assumptions are used in this document.

- the user successfully logged into Stroom with the appropriate administrative privilege (**Manage Properties**).

4.4.1.2 Introduction

Certain Stroom **System Properties** can be edited via the Stroom User Interface.

4.4.1.3 Editing a System Property

To edit a **System Property** select the **Tools** item of the **Main Menu** and select to bring up the **Tools** sub-menu.



Then move down and select the **Properties** sub-item to be presented with **System Properties** configuration window as seen below.

Name	Value	Source	Description
stroom.aboutHTML	<h1>About Stroom</h1><p>Stroom is designed to receive and process data from various sources and destinations. It provides a flexible architecture for building complex data processing pipelines. This page serves as a starting point for learning more about Stroom's features and how to use it.</p>	Default	Set to true if users should be prompted to choose an activity on login.
stroom.activity.chooseOnStartup	false	Default	Set to true if users should be prompted to choose an activity on login.
stroom.activity.editorBody	Activity Code: <input type="text" name="code"></input>	Default	The HTML to display in the activity editor popup.
stroom.activity.editorTitle	Edit Activity	Default	The title of the activity editor popup.
stroom.activity.enabled	false	Default	If you would like users to be able to record some info about the activity they are performing set this property to true.
stroom.activity.managerTitle	Choose Activity	Default	The title of the activity manager popup.
stroom.benchmark.concurrentWriters	10	Default	Set the number of threads to use concurrently to write test streams
stroom.benchmark.recordCount	10000	Default	Set the number of records to be created for each stream during a benchmark test
stroom.benchmark.streamCount	1000	Default	Set the number of streams to be created during a benchmark test
stroom.bufferSize	4096	Default	If set the default buffer size to use
stroom.clusterCallIgnoreSSLHostnameVerifier	true	Default	If cluster calls are using SSL then choose if we want to ignore host name verification
stroom.clusterCallReadTimeout	30s	Default	Time in ms (but can be specified as 10s, 1m) before throwing read timeout
stroom.clusterCallUseLocal	true	Default	Do local calls when calling our own local services (true is an optimisation)
stroom.clusterResponseTimeout	30s	Default	Time in ms (but can be specified as 10s, 1m) before giving up on cluster results
stroom.contentPackImportEnabled	true	Default	If true any content packs found in \${CATALINA_BASE}/contentPackImport/ will be imported into Stroom
stroom.databaseMultiInsertMaxBatchSize	500	Default	The maximum number of rows to insert in a single multi insert statement, e.g. INSERT INTO X VALUE:
stroom.daysToAccountExpiry	90	Default	Number of days before we disable an account
stroom.daysToPasswordExpiry	90	Default	Number of days before passwords expire
stroom.daysToUnusedAccountExpiry	30	Default	Number of days before we disable an unused account
stroom.db.connectionPool.acquireIncrement	3	Default	Determines how many connections to list.add to the pool in one go.
stroom.db.connectionPool.acquireRetryAttempts	30	Default	Determines how many attempts we make to acquire a connection.
stroom.db.connectionPool.acquireRetryDelay	1000	Default	Determines how long we wait (in milliseconds) between attempts to acquire a connection.
stroom.db.connectionPool.checkoutTimeout	0	Default	Determines how long (in seconds) a connection can be checked out from the pool before it is discarded.
stroom.db.connectionPool.idleConnectionTestPeriod	0	Default	If this is a number greater than 0, we will test all idle, pooled but uncheck-out connections, every thi
stroom.db.connectionPool.initialPoolSize	3	Default	The initial size of the DB connection pool
stroom.db.connectionPool.maxConnectionAge	0	Default	Determines how long in seconds a connection will remain in the pool before being discarded.
stroom.db.connectionPool.maxIdleTime	0	Default	The seconds a connection can remain pooled but unused before being discarded. Zero means idle co
stroom.db.connectionPool.maxIdleTimeExcessConnections	60	Default	Determines how long (in seconds) excess connections remain in the pool before being discarded.

Stroom UI Tools System Properties

Using the Scrollbar to the right of the **System Properties** configuration window and scroll down to the line where the property one wants to modify is displayed then select (*left click*) the line. In the example below we have selected the `stroom.maxStreamSize` property.

System Properties			
Name	Value	Source	Description
stroom.mail.userName	StroomUser@swtf.dyndns.org	DB	Email service username, e.g. XXXXX@gmail.com
stroom.maintenance.message		Default	Provide a warning message to users about an outage or other significant event.
stroom.maintenance.preventLogin	false	Default	Prevent new logins to the system. This is useful if the system is scheduled to have an outage.
stroom.maxAggregation	10000	Default	This stops the aggregation after a certain size / nested streams
stroom.maxAggregationScan	100000	Default	The limit of files to inspect before aggregation begins (should be bigger than maxAggregation)
stroom.maxStreamSize	1G	Default	This stops the aggregation after a certain size / nested streams
stroom.namePattern	^[\w\W-Z0-9_-\.\(\)](1)\$	Default	The regex pattern for entity names
stroom.node	tba	Default	Should only be set per node in application property file
stroom.node.status.heapHistogram.classNameMatchRe	*stroom\..*	Default	A single regex that will be used to filter classes from the jmap histogram internal statistic based on their n
stroom.node.status.heapHistogram.jMapExecutable	jmap	Default	The jmap executable name if it is available on the PATH or a fully qualified form
stroom.pageTitle	Stroom	Default	The page title for Stroom shown in the browser tab

[Stroom UI Tools System Properties - Selected Property](#)

Now bring up the editing window by *double clicking* on the selected line. At this we will be presented with the Application Property - `stroom.maxStreamSize` editing window.

System Properties

Quick Filter

Name	Value	Source	Description
stroom.mail.userName	StroomUser@swtf.dyndns.org	DB	Email service user
stroom.maintenance.message		Default	Provide a warning
stroom.maintenance.preventLogin	false	Default	Prevent new logins
stroom.maxAggregation	10000	Default	This stops the agg
stroom.maxAggregationScan	100000	Default	The limit of files to
stroom.maxStreamSize	1G	Default	This stops the agg
stroom.namePattern	^[\w\W-Z0-9_-\.\(\)](1)\$	Default	The regex pattern f
stroom.node			Only be set
stroom.node.status.heapHistogram.classNameMatchRe			Regex that
stroom.node.status.heapHistogram.jMapExecutable			executab
stroom.pageTitle			title for S
stroom.pipeline.appenders.maxActiveDest			um num
stroom.pipeline.parser.secureProcessing			the imple
stroom.pipeline.xslt.maxElements			num num
stroom.proxyDir			look for S
stroom.proxyThreads			of threads
stroom.query.history.daysRetention			per of day
stroom.query.history.itemsRetention			minimum num
stroom.query.infoPopup.enabled			would like us
stroom.query.infoPopup.title			of the que
stroom.query.infoPopup.validationRegex			used to val
stroom.rack			only be set
stroom.search.extraction.maxThreads			olute maxi
stroom.search.extraction.maxThreadsPer			mum num
stroom.search.maxBooleanClauseCount			mum num
stroom.search.maxResults			The maximum num

Application Property - stroom.maxStreamSize

Name: stroom.maxStreamSize
Description: This stops the aggregation after a certain size / nested streams
Value: 1G
Source: Default
Default: 1G
Requires UI Refresh:
Requires Restart:

Ok **Cancel**

[Stroom UI Tools System Properties - Editing Property](#)

Now edit the property, by *double clicking* the string in the Value entry box. In this case we select the `1G` value to see

Application Property - stroom.maxStreamSize

Name: stroom.maxStreamSize
Description: This stops the aggregation after a certain size / nested streams
Value: 1G
Source: Default
Default: 1G
Requires UI Refresh:
Requires Restart:

Ok **Cancel**

[Stroom UI Tools System Properties - Editing Property - Value selected](#)

Now change the selected `1G` value to the value we want. In this example, we are changing the value to `512M`.

Application Property - stroom.maxStreamSize

Name:	stroom.maxStreamSize
Description:	This stops the aggregation after a certain size / nested streams
Value:	512M
Source:	Default
Default:	1G
Requires UI Refresh:	<input type="checkbox"/>
Requires Restart:	<input type="checkbox"/>

Ok **Cancel**

[Stroom UI Tools System Properties - Editing Property - Value changed](#)

At this, press the

Ok

[Stroom UI OkButton](#)

to see the new value updated in the **System Properties** configuration window

stroom.maxAggregationScan	100000	Default	This limit the aggregation after a certain size / nested streams
stroom.maxStreamSize	512M	DB	This stops the aggregation after a certain size / nested streams
stroom.namePattern	^([a-zA-Z0-9_\\-\\.\\()]{1})\$	Default	The regex pattern for entity names
stroom.nodes	the	Default	Should only be set per node in application properties file

[Stroom UI Tools System Properties - Value changed](#)

4.5 - Authentication

4.5.1 - Create a user

This HOWTO provides the steps to create a user via the Stroom User Interface.

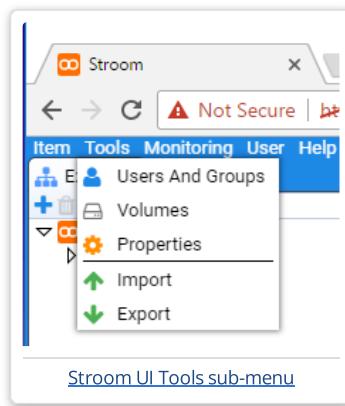
4.5.1.1 Assumptions

The following assumptions are used in this document.

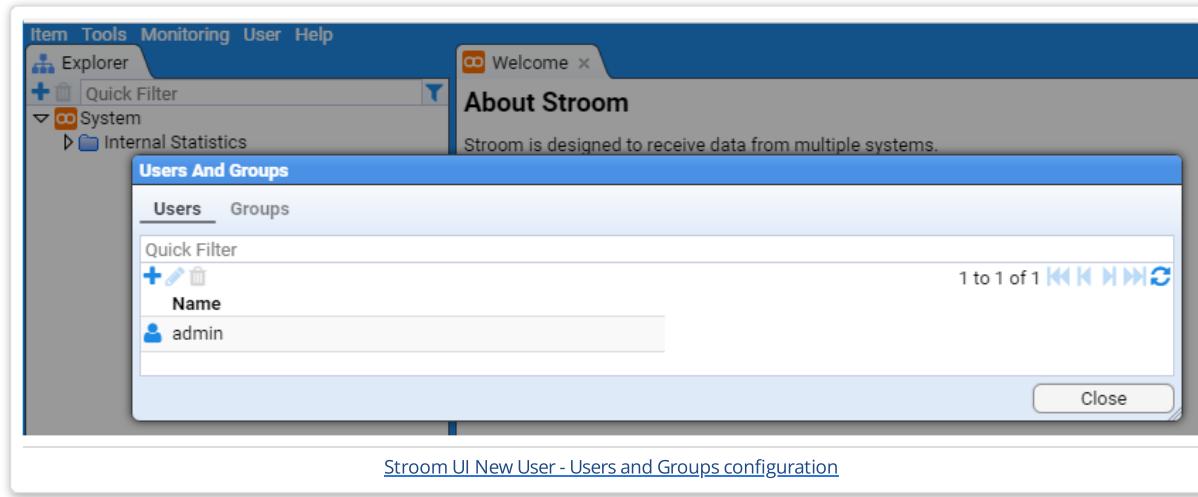
- An account with the `Administrator` Application [Permission](#) is currently logged in.
- We will be adding the user `burn`
- We will make this user an `Administrator`

4.5.1.2 Add a new user

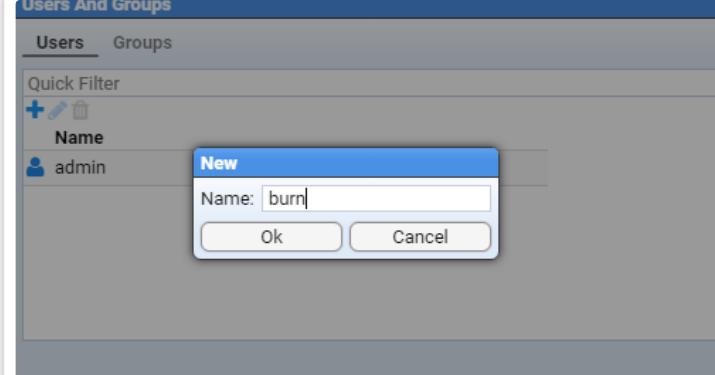
To add a new user, move your cursor to the `Tools` item of the **Main Menu** and select to bring up the `Tools` sub-menu.



then move down and select the `Users and Groups` sub-item to be presented with the `Users and Groups` configuration window as seen below.

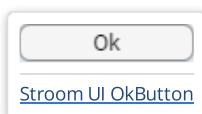


To add the user, move the cursor to the `New` icon in the top left and select it. On selection you will be prompted for a user name. In our case we will enter the user `burn`.

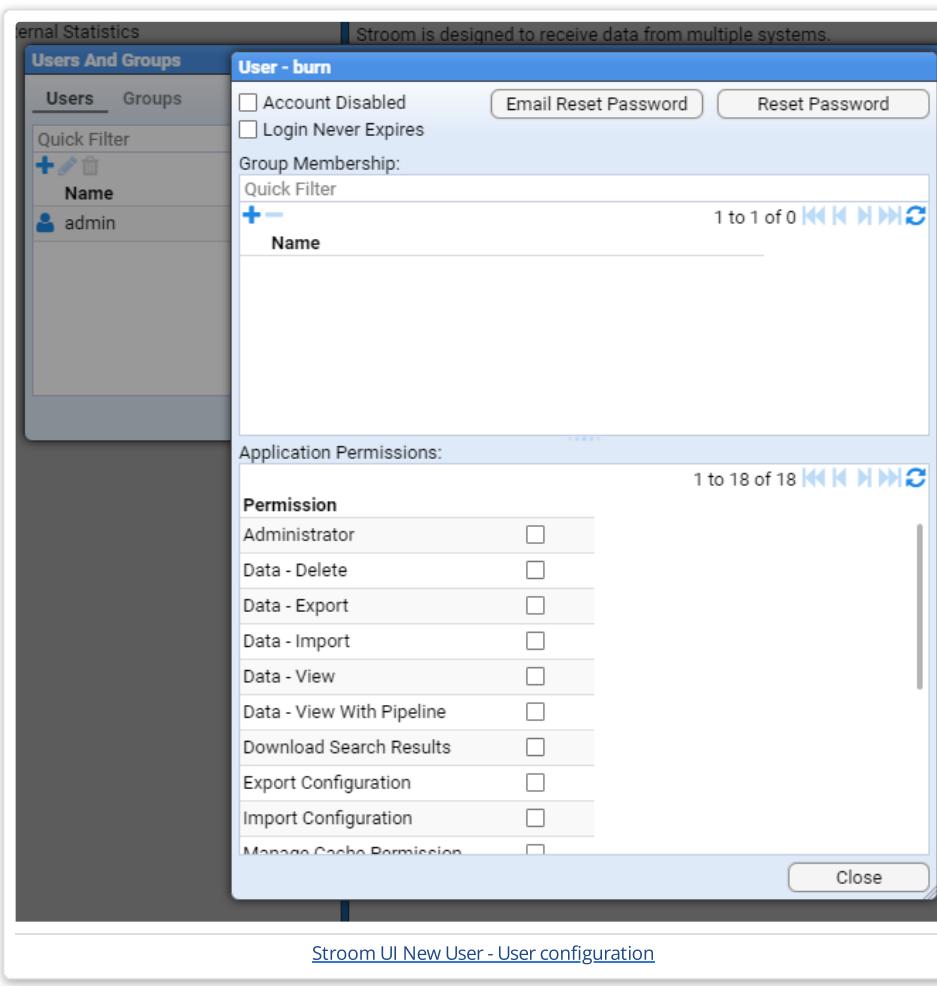


[Stroom UI New User - Add User](#)

and on pressing



will be presented with the User configuration window.



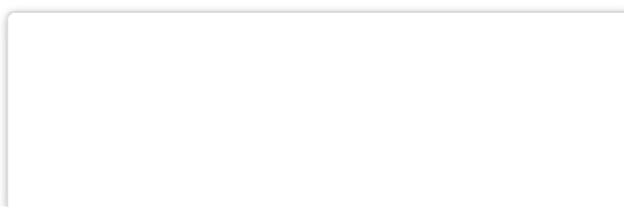
[Stroom UI New User - User configuration](#)

4.5.1.2.1 Set the User Application Permissions

See [Permissions](#) for an explanation of the various Application Permissions a user can have.

4.5.1.2.1.1 Assign an Administrator Permission

As we want the user to be an administrator, select the **Administrator** Permission check-box



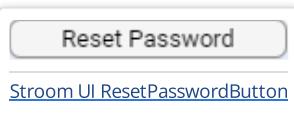
Application Permissions:

Permission	
Administrator	<input checked="" type="checkbox"/>
Data - Delete	<input type="checkbox"/>

[Stroom UI New User - User configuration - set administrator permission](#)

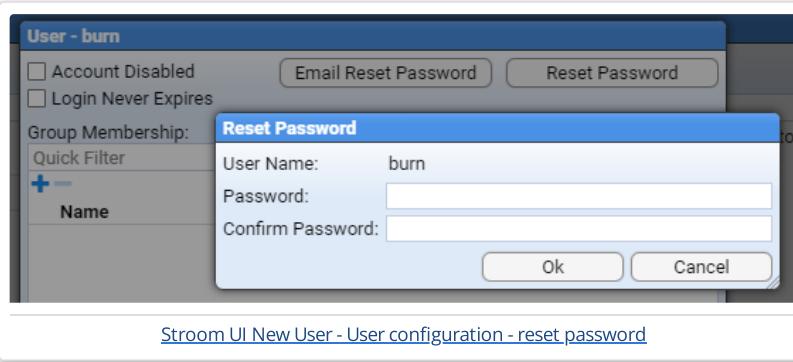
4.5.1.2.1.2 Set User's Password

We need to set `burn`'s password (which he will need to reset on first login). So, select the



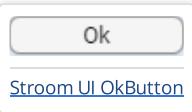
[Stroom UI ResetPasswordButton](#)

button to gain the Reset Password window



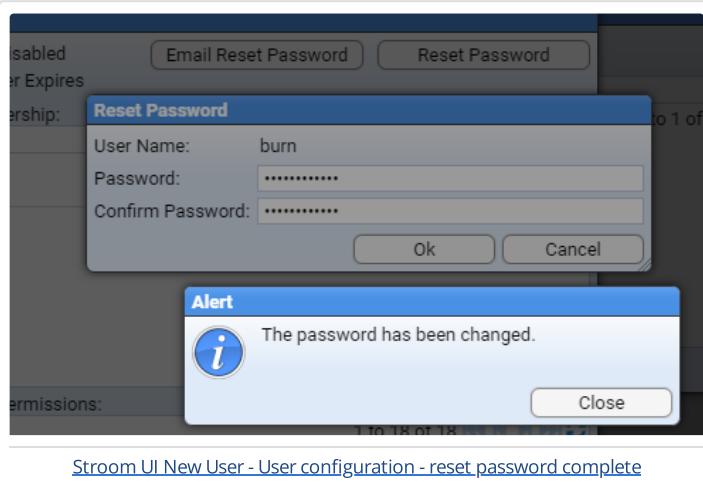
[Stroom UI New User - User configuration - reset password](#)

After setting a password and pressing the



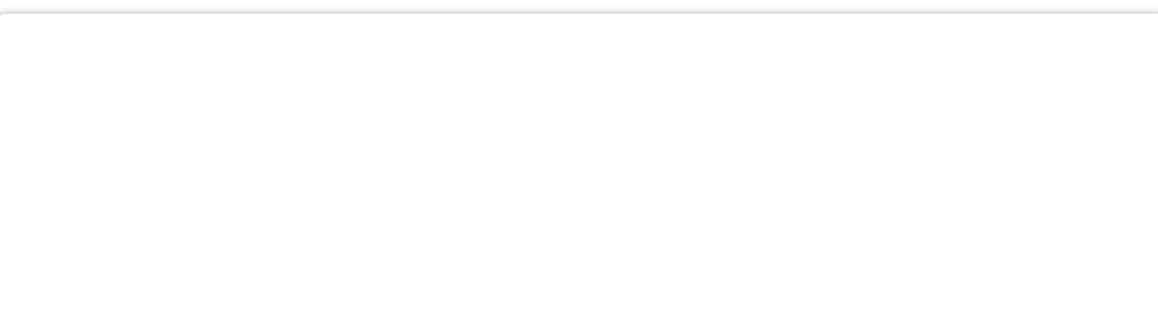
[Stroom UI OkButton](#)

button we get the informational Alert window as per



[Stroom UI New User - User configuration - reset password complete](#)

and on close of the Alert we are presented again with the user configuration window.





[Stroom UI New User - User configuration - user added](#)

We should close this window by pressing the

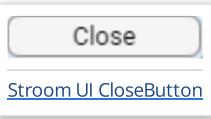


[Stroom UI CloseButton](#)

button to be presented with the **Users and Groups** window with the new user **burn** added.



At this, one can close the **Users and Groups** configuration window by pressing the



[Stroom UI CloseButton](#)

button at the bottom right of the window.

4.5.2 - Login

This HOWTO shows how to log into the Stroom User Interface.

4.5.2.1 Assumptions

The following assumptions are used in this document.

- for manual login, we will log in as the user `admin` whose password is set to `admin` and the password is pre-expired
- for PKI Certificate login, the Stroom deployment would have been configured to accept PKI Logins

4.5.2.2 Manual Login

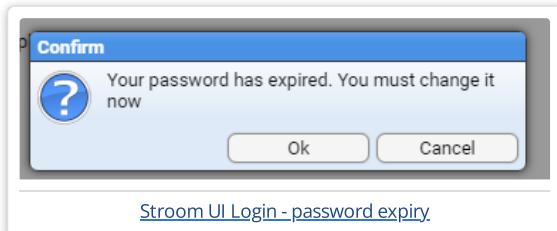
Within the **Login** panel, enter `admin` into the *User Name:* entry box and `admin` into the *Password:* entry box as per



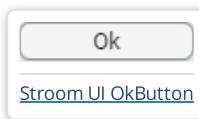
When you press the



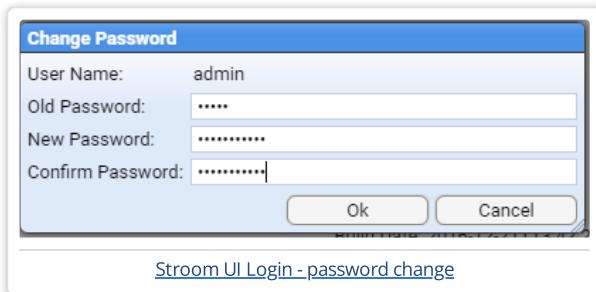
button, you are advised that your user's password has expired and you need to change it.



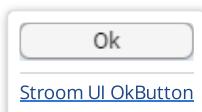
Press the



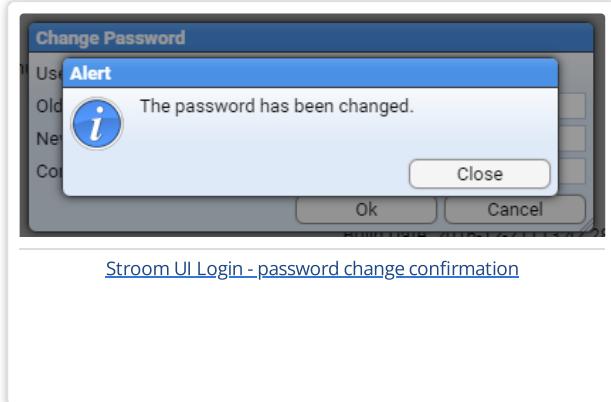
button and enter the old password `admin` and a new password with confirmation in the appropriate entry boxes.



Again press the



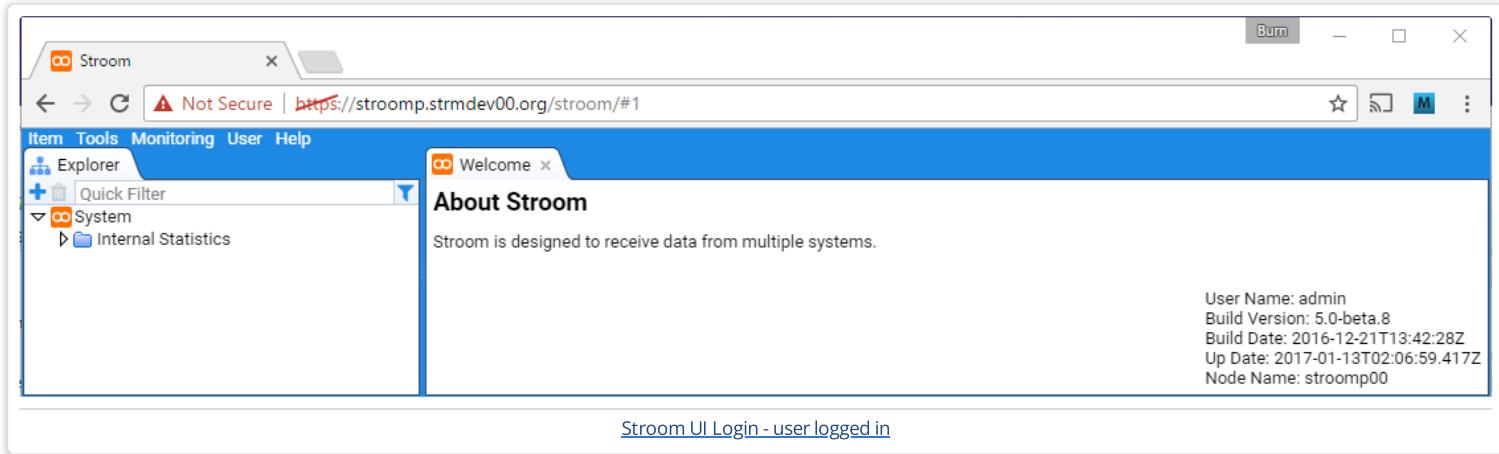
button to see the confirmation that the password has changed.



On pressing



you will be logged in as the **admin** user and you will be presented with the **Main Menu** (**Item Tools Monitoring User Help**), and the **Explorer** and **Welcome** panels (or tabs).



We have now successfully logged on as the `admin` user.

The next time you login with this account, you will not be prompted to change the password until the password expiry period has been met.

4.5.2.3 PKI Certificate Login

To login using a PKI Certificate, a user should have their Personal PKI certificate loaded in the browser (and selected if you have multiple certificates) and the user just needs to go to the Stroom UI URL, and providing you have an account, you will be automatically logged in.

4.5.3 - Logout

This HOWTO shows how to log out of the Stroom User Interface.

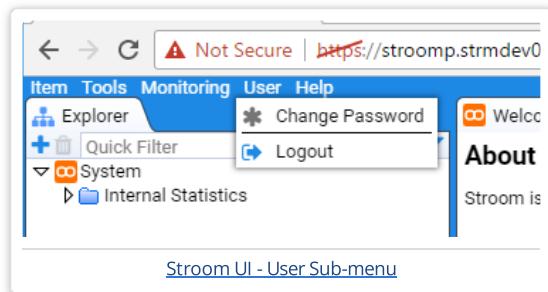
4.5.3.1 Assumptions

The following assumptions are used in this document.

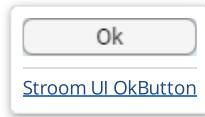
- the user `admin` is currently logged in

4.5.3.2 Log out of UI

To log out of the UI, select the `User` item of the **Main Menu** and to bring up the `User` sub-menu.



and select the `Logout` sub-item and confirm you wish to log out by selecting the



button.



This will return you to the Login page



4.6 - Event Feeds

4.6.1 - Apache HTTPD Event Feed

The following will take you through the process of creating an Event Feed in Stroom.

In this example, the logs are in a well-defined, line based, text format so we will use a Data Splitter parser to transform the logs into simple record-based XML and then a XSLT translation to normalise them into the Event schema.

A separate document will describe the method of automating the storage of normalised events for this feed. Further, we will not Decorate these events. Again, Event Decoration is described in another document.

Author: John Doe

Last Updated: 7 Mar 2020

Recommended Additional Documentation: [HOWTO - Enabling Processors for a Pipeline](#)

4.6.1.1 Event Log Source

For this example, we will use logs from an Apache HTTPD Web server. In fact, the web server in front of Stroom.

To get the optimal information from the Apache HTTPD access logs, we define our log format based on an extension of the BlackBox format. The format is described and defined below. This is an extract from a httpd configuration file (`/etc/httpd/conf/httpd.conf`)

```

# Stroom - Black Box Auditing configuration

#
# %a - Client IP address (not hostname (%h) to ensure ip address only)
# When logging the remote host, it is important to log the client IP address, not the
# hostname. We do this with the '%a' directive. Even if HostnameLookups are turned on,
# using '%a' will only record the IP address. For the purposes of BlackBox formats,
# reversed DNS should not be trusted

# %{REMOTE_PORT}e - Client source port
# Logging the client source TCP port can provide some useful network data and can help
# one associate a single client with multiple requests.
# If two clients from the same IP address make simultaneous connections, the 'common log'
# file format cannot distinguish between those clients. Otherwise, if the client uses
# keep-alives, then every hit made from a single TCP session will be associated by the same
# client port number.
# The port information can indicate how many connections our server is handling at once,
# which may help in tuning server TCP/OP settings. It will also identify which client ports
# are legitimate requests if the administrator is examining a possible SYN-attack against a
# server.
# Note we are using the REMOTE_PORT environment variable. Environment variables only come
# into play when mod_cgi or mod_cgid is handling the request.

# %X - Connection status (use %c for Apache 1.3)
# The connection status directive tells us detailed information about the client connection.
# It returns one of three flags:
# x if the client aborted the connection before completion,
# + if the client has indicated that it will use keep-alives (and request additional URLs),
# - if the connection will be closed after the event
# Keep-Alive is a HTTP 1.1. directive that informs a web server that a client can request multiple
# files during the same connection. This way a client doesn't need to go through the overhead
# of re-establishing a TCP connection to retrieve a new file.

# %t - time - or [%{%d/%b/%Y:%T}t.%{msec_frac}t %{%z}t] for Apache 2.4
# The %t directive records the time that the request started.
# NOTE: When deployed on an Apache 2.4, or better, environment, you should use
# strftime format in order to get microsecond resolution.

# %l - remote logname

# %u - username [in quotes]
# The remote user (from auth; This may be bogus if the return status (%s) is 401
# for non-ssl services)
# For SSL services, user names need to be delivered as DNS to deliver PKI user details
# in full. To pass through PKI certificate properties in the correct form you need to
# add the following directives to your Apache configuration:
# SSLUserName SSL_CLIENT_S_DN
# SSLOptions +StdEnvVars
# If you cannot, then use %{SSL_CLIENT_S_DN}x in place of %u and use blackboxSSLUser
# LogFormat nickname

# %r - first line of text sent by web client [in quotes]
# This is the first line of text send by the web client, which includes the request
# method, the full URL, and the HTTP protocol.

# %s - status code before any redirection
# This is the status code of the original request.

# %>s - status code after any redirection has taken place

# This is the final status code of the request, after any internal redirections may
# have taken place.

# %D - time in microseconds to handle the request
# This is the number of microseconds the server took to handle the request in microseconds

```

```

# %I - incoming bytes
# This is the bytes received, include request and headers. It cannot, by definition be zero.

# %O - outgoing bytes
# This is the size in bytes of the outgoing data, including HTTP headers. It cannot, by definition be zero.

# %B - outgoing content bytes
# This is the size in bytes of the outgoing data, EXCLUDING HTTP headers. Unlike %b, which records '-' for zero bytes transferred, %B will record '0'.

# %{Referer}i - Referrer HTTP Request Header [in quotes]
# This is typically the URL of the page that made the request. If linked from e-mail or direct entry this value will be empty. Note, this can be spoofed or turned off

# %{User-Agent}i - User agent HTTP Request Header [in quotes]
# This is the identifying information the client (browser) reports about itself.
# It can be spoofed or turned off

# %V - the server name according to the UseCanonicalName setting
# This identifies the virtual host in a multi host webservice

# %p - the canonical port of the server servicing the request

# Define a variation of the Black Box logs
#
# Note, you only need to use the 'blackboxSSLUser' nickname if you cannot set the following directives for any SSL configurations
# SSLUserName SSL_CLIENT_S_DN
# SSLOptions +StdEnvVars
# You will also note the variation for no logio module. The logio module supports the %I and %O formatting directive
#
<IfModule mod_logio.c>
    LogFormat "%a/%{REMOTE_PORT}e %X %t %l \".\" \"%r\" %s/%>s %D %I/%0/%B \"%{Referer}i\" \"%{User-Agent}i\" %V/%p" blackboxUser
    LogFormat "%a/%{REMOTE_PORT}e %X %t %l \".\" \"%{SSL_CLIENT_S_DN}.\\" \"%r\" %s/%>s %D %I/%0/%B \"%{Referer}i\" \"%{User-Agent}i\" %V/%p" blackboxUser
</IfModule>
<IfModule !mod_logio.c>
    LogFormat "%a/%{REMOTE_PORT}e %X %t %l \".\" \"%r\" %s/%>s %D 0/0/%B \"%{Referer}i\" \"%{User-Agent}i\" %V/$p" blackboxUser
    LogFormat "%a/%{REMOTE_PORT}e %X %t %l \".\" \"%{SSL_CLIENT_S_DN}.\\" \"%r\" %s/%>s %D 0/0/%B \"%{Referer}i\" \"%{User-Agent}i\" %V/$p" blackboxUser
</IfModule>
```

A copy of this configuration can be found [here](#).

As Stroom can use PKI for login, you can configure Stroom's Apache to make use of the blackboxSSLUser log format. A sample set of logs in this format appear below.

```

192.168.4.220/61801 - [18/Jan/2020:12:39:04 -0800] - "/C=USA/ST=CA/L=Los Angeles/O=Default Company Ltd/CN=Burn Frank (burn)" "POST /index.html"
192.168.4.220/61854 - [18/Jan/2020:12:40:04 -0800] - "/C=USA/ST=CA/L=Los Angeles/O=Default Company Ltd/CN=Burn Frank (burn)" "POST /index.html"
192.168.4.220/61909 - [18/Jan/2020:12:41:04 -0800] - "/C=USA/ST=CA/L=Los Angeles/O=Default Company Ltd/CN=Burn Frank (burn)" "POST /index.html"
192.168.4.220/61962 - [18/Jan/2020:12:42:04 -0800] - "/C=USA/ST=CA/L=Los Angeles/O=Default Company Ltd/CN=Burn Frank (burn)" "POST /index.html"
192.168.3.117/62015 - [18/Jan/2020:12:43:04 -1000] - "/C=AUS/ST=NSW/L=Sydney/O=Default Company Ltd/CN=Max Bergman (maxb)" "POST /index.html"
192.168.3.117/62092 - [18/Jan/2020:12:44:04 -1000] - "/C=AUS/ST=NSW/L=Sydney/O=Default Company Ltd/CN=Max Bergman (maxb)" "POST /index.html"
192.168.3.117/62147 - [18/Jan/2020:12:44:04 -1000] - "/C=AUS/ST=NSW/L=Sydney/O=Default Company Ltd/CN=Max Bergman (maxb)" "POST /index.html"
192.168.3.117/62202 - [18/Jan/2020:12:44:04 -1000] - "/C=AUS/ST=NSW/L=Sydney/O=Default Company Ltd/CN=Max Bergman (maxb)" "POST /index.html"
192.168.3.117/62294 - [18/Jan/2020:12:44:04 -1000] - "/C=AUS/ST=NSW/L=Sydney/O=Default Company Ltd/CN=Max Bergman (maxb)" "POST /index.html"
192.168.3.117/62349 - [18/Jan/2020:12:44:04 -1000] - "/C=AUS/ST=NSW/L=Sydney/O=Default Company Ltd/CN=Max Bergman (maxb)" "POST /index.html"
192.168.2.245/62429 - [18/Jan/2020:12:50:04 +0200] - "/C=GBR/ST=GLOUCESTERSHIRE/L=Bristol/O=Default Company Ltd/CN=Kostas Kosta" "POST /index.html"
192.168.2.245/62495 - [18/Jan/2020:12:51:04 +0200] - "/C=GBR/ST=GLOUCESTERSHIRE/L=Bristol/O=Default Company Ltd/CN=Kostas Kosta" "POST /index.html"
192.168.2.245/62549 - [18/Jan/2020:12:52:04 +0200] - "/C=GBR/ST=GLOUCESTERSHIRE/L=Bristol/O=Default Company Ltd/CN=Kostas Kosta" "POST /index.html"
192.168.2.245/62626 - [18/Jan/2020:12:52:04 +0200] - "/C=GBR/ST=GLOUCESTERSHIRE/L=Bristol/O=Default Company Ltd/CN=Kostas Kosta" "POST /index.html"
```

4.6.1.2 Create the Feed and its Pipeline

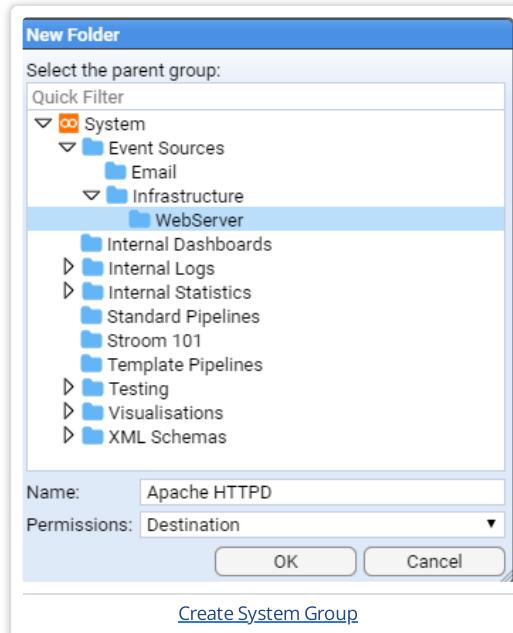
To reflect the source of these Accounting Logs, we will name our feed and its pipeline Apache-SSLBlackBox-V2.0-EVENTS and it will be stored in the system group Apache HTTPD under the main system group - Event Sources .

4.6.1.2.1 Create System Group

To create the system group Apache HTTPD, navigate to the *Event Sources/Infrastructure/WebServer* system group within the Explorer pane (if this system group structure does not already exist in your Stroom instance then refer to the **HOWTO Stroom Explorer Management** for guidance). Left click to highlight the *WebServer* system group then right click to bring up the object context menu. Navigate to the *New* icon, then the *Folder* icon to reveal the *New Folder* selection window.



In the New Folder window enter Apache HTTPD into the **Name:** text entry box.



The click on **OK** at which point you will be presented with the Apache HTTPD system group configuration tab. Also note, the *WebServer* system group within the Explorer pane has automatically expanded to display the *Apache HTTPD* system group.



Close the Apache HTTPD system group configuration tab by clicking on the close item icon on the right-hand side of the tab

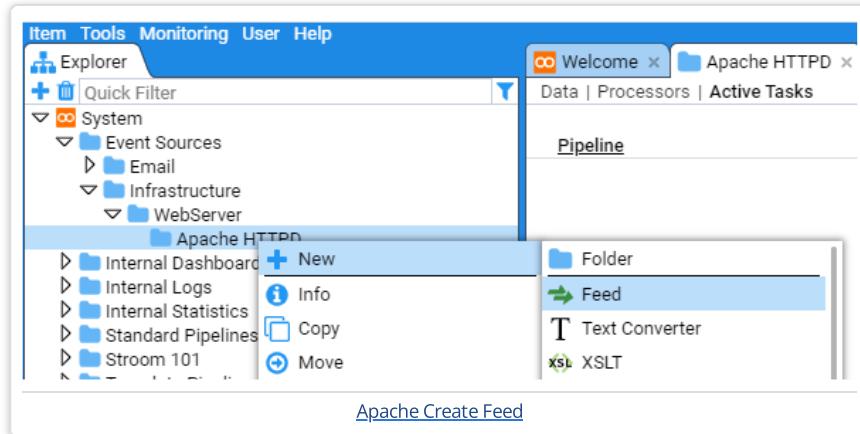


. We now need to create, in order

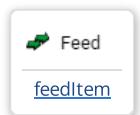
- the Feed,
- the Text Parser,
- the Translation and finally,
- the Pipeline.

4.6.1.2.2 Create Feed

Within the Explorer pane, and having selected the Apache HTTPD group, right click to bring up object context menu. Navigate to New, Feed



Select the Feed icon



, when the **New Feed** selection window comes up, ensure the **Apache HTTPD** system group is selected or navigate to it. Then enter the name of the feed, Apache-SSLBlackBox-V2.0-EVENTS, into the **Name:** text entry box the press **OK**.

It should be noted that the default Stroom FeedName pattern will not accept this name. One needs to modify the `stroom.feedNamePattern` stroom property to change the default pattern to `^[a-zA-Z0-9_-\.\]{4,}\$`. See the [HOWTO on System Properties](#) document to see how to make this change.



At this point you will be presented with the new feed's configuration tab and the feed's Explorer object will automatically appear in the Explorer pane within the **Apache HTTPD** system group.

Select the **Settings** tab on the feed's configuration tab. Enter an appropriate description into the **Description:** text entry box, for instance:

"Apache HTTPD events for BlackBox Version 2.0. These events are from a Secure service (https)."

In the **Classification**: text entry box, enter a Classification of the data that the event feed will contain - that is the classification or sensitivity of the accounting log's content itself.

As this is not a Reference Feed, leave the **Reference Feed**: check box unchecked.

We leave the **Feed Status**: at *Receive*.

We leave the **Stream Type**: as *Raw Events* as this we will be sending batches (streams) of raw event logs.

We leave the **Data Encoding**: as UTF-8 as the raw logs are in this form.

We leave the **Context Encoding**: as UTF-8 as there no context events for this feed.

We leave the **Retention Period**: at *Forever* as we do not want to delete the raw logs.

This results in

Apache-SSLBlackBox-V2.0-EVENTS

Description: "Apache HTTPD events for BlackBox Version 2.0. These events are from a Secure service (https)."

Classification: UNCLASSIFIED

Reference Feed:

Feed Status: Receive

Stream Type: Raw Events

Data Encoding: UTF-8

Context Encoding: UTF-8

Retention Period: Forever

New Feed tab

Save the feed by clicking on the



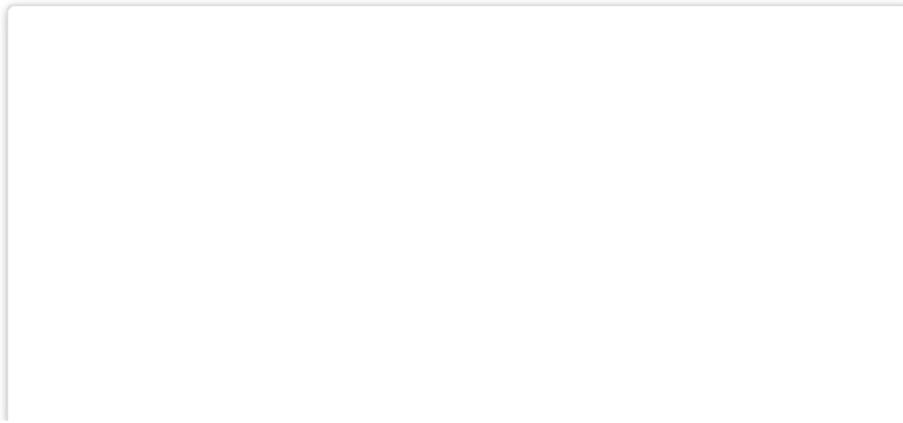
icon.

4.6.1.2.3 Create Text Converter

Within the Explorer pane, and having selected the `Apache HTTPD` system group, right click to bring up object context menu, then navigate to the

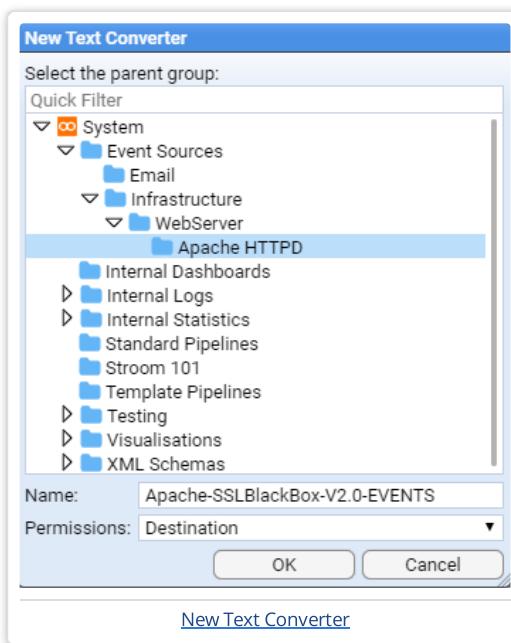


Text Converter item

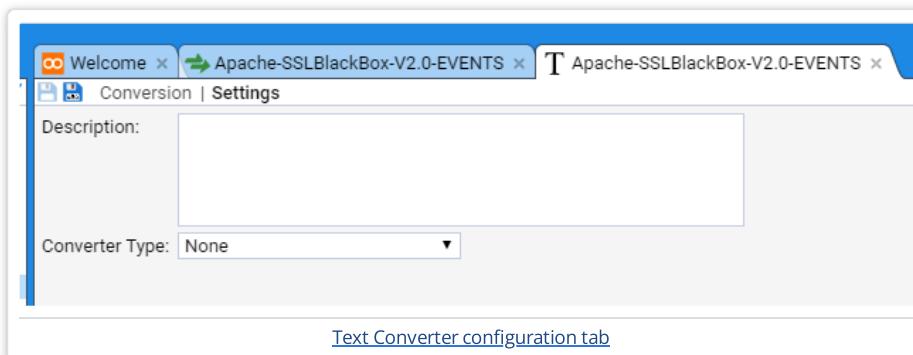




and left click to select. When the **New Text Converter**



selection window comes up enter the name of the feed, Apache-SSLBlackBox-V2.0-EVENTS, into the **Name:** text entry box then press **OK**. At this point you will be presented with the new text converter's configuration tab.



Enter an appropriate description into the **Description:** text entry box, for instance

"Apache HTTPD events for BlackBox Version 2.0 - text converter. See Conversion for complete documentation."

Set the **Converter Type:** to be Data Splitter from drop down menu.

The screenshot shows the Apache-SSLBlackBox-V2.0-EVENTS interface. In the top navigation bar, there are tabs for 'Welcome', 'Apache-SSLBlackBox-V2.0-EVENTS', and 'T * Apache-SSLBlackBox-V2.0-EVENTS'. Below the tabs, the title 'Conversion | Settings' is displayed. A 'Description' field contains the text: 'Apache HTTPD events for BlackBox Version 2.0 - text converter. See Conversion for complete documentation.' A 'Converter Type' dropdown is set to 'Data Splitter'. At the bottom of the panel, there is a link 'Text Converter configuration settings'.

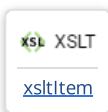
Save the text converter by clicking on the



icon.

4.6.1.2.4 Create XSLT Translation

Within the Explorer pane, and having selected the Apache HTTPD system group, right click to bring up object context menu, then navigate to the New icon to reveal the New sub-context menu. Next, navigate to the



item and left click to select.

The screenshot shows the Apache-SSLBlackBox-V2.0-EVENTS interface with the 'Explorer' tab selected. In the left sidebar, under the 'System' group, the 'Apache HTTPD' folder is expanded, showing various sub-items like 'Internal Dashboards', 'Internal Logs', etc. A right-click context menu is open over the 'Apache HTTPD' folder, with the 'New' option highlighted. A sub-menu titled 'New XSLT' is displayed, containing options: 'Folder', 'Feed', 'Text Converter', 'XSLT' (which is selected and highlighted in blue), 'Pipeline', and 'Dashboard'. The 'New XSLT' label is visible at the bottom of this sub-menu.

When the **New XSLT** selection window comes up,

The screenshot shows the 'New XSLT' configuration dialog box. At the top, it says 'Select the parent group:' followed by a 'Quick Filter' search bar. Below that is a tree view of the system structure, with 'Apache HTTPD' selected as the parent group. The tree includes 'System', 'Event Sources', 'Email', 'Infrastructure', 'WebServer', and their sub-items. At the bottom of the dialog, there are fields for 'Name:' (set to 'Apache-SSLBlackBox-V2.0-EVENTS') and 'Permissions:' (set to 'Destination'). There are also 'OK' and 'Cancel' buttons at the bottom.

enter the name of the feed, Apache-SSLBlackBox-V2.0-EVENTS, into the **Name:** text entry box then press **OK**. At this point you will be presented with the new XSLT's configuration tab.



Enter an appropriate description into the **Description:** text entry box, for instance

"Apache HTTPD events for BlackBox Version 2.0 - translation. See Translation for complete documentation."



Save the XSLT by clicking on the



icon.

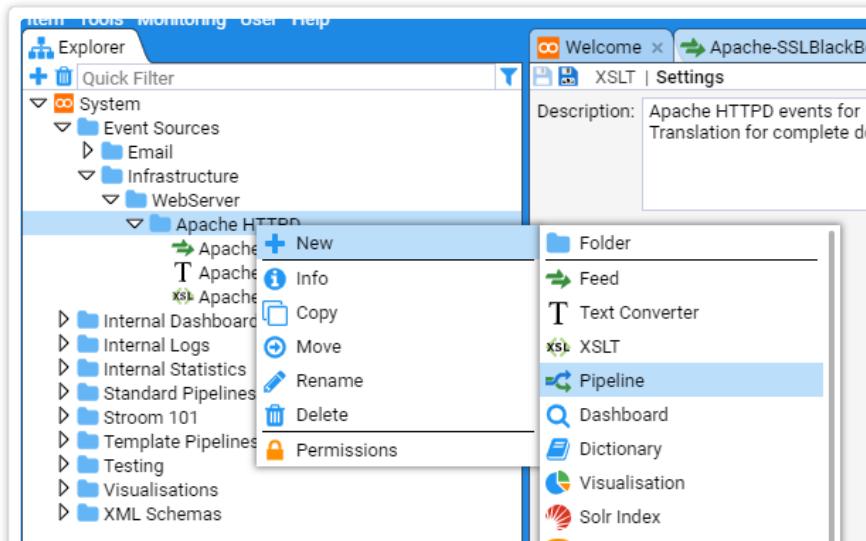
4.6.1.2.5 Create Pipeline

In the process of creating this pipeline we have assumed that the **Template Pipeline** content pack has been loaded, so that we can *Inherit* a pipeline structure from this content pack and configure it to support this specific feed.

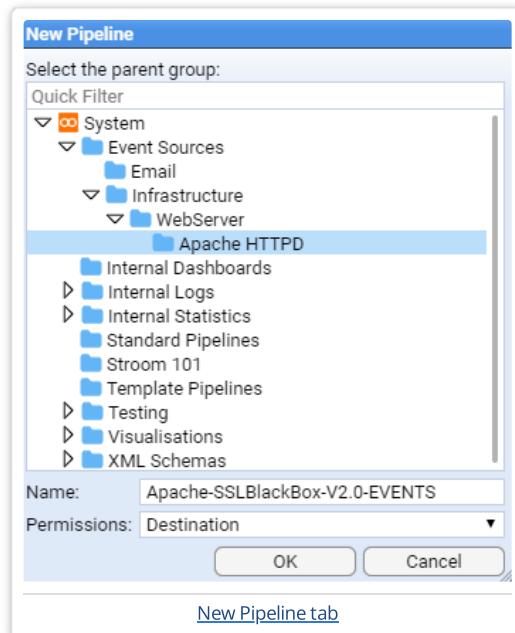
Within the Explorer pane, and having selected the Apache HTTPD system group, right click to bring up object context menu, then the New sub-context menu. Navigate to the



Pipeline and left click to select. When the **New Pipeline**

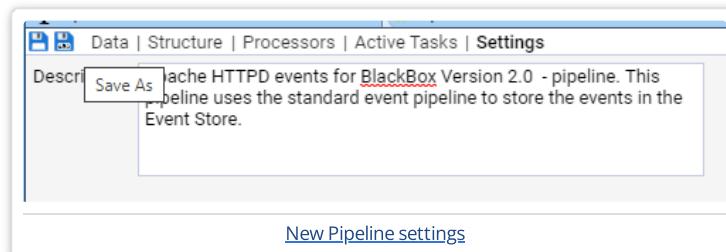


selection window comes up, navigate to, then select the Apache HTTPD system group and then enter the name of the pipeline, Apache-SSLBlackBox-V2.0-EVENTS into the **Name:** text entry box then press **OK**. At this you will be presented with the new pipeline's configuration tab



As usual, enter an appropriate **Description:**

"Apache HTTPD events for BlackBox Version 2.0 - pipeline. This pipeline uses the standard event pipeline to store the events in the Event Store."



Save the pipeline by clicking on the



icon.

We now need to select the structure this pipeline will use. We need to move from the **Settings** sub-item on the pipeline configuration tab to the **Structure** sub-item. This is done by clicking on the **Structure** link, at which we see



Apache-SSLBlackBox-V2.0-EVENTS x Apache-SSLBlackBox-V2.0-EVENTS x Apache-SSLBlackBox-V2.0-EVENTS x

Data | Structure | Processors | Active Tasks | Settings
Inherit From: None ... View Source

Source

Property Name	Value	Source	Inherited Value	Inherited From	Default Value	Description
---------------	-------	--------	-----------------	----------------	---------------	-------------

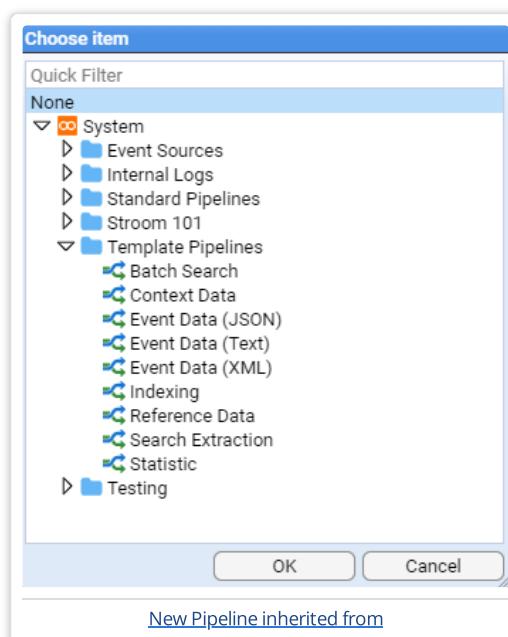
+ - Pipeline	Feed	Stream Type	Inherited From
--------------	------	-------------	----------------

1 to 1 of ?

[New Pipeline Structure](#)

Next we will choose an Event Data pipeline. This is done by inheriting it from a defined set of Template Pipelines. To do this, click on the menu selection icon to the right of the Inherit From: text display box.

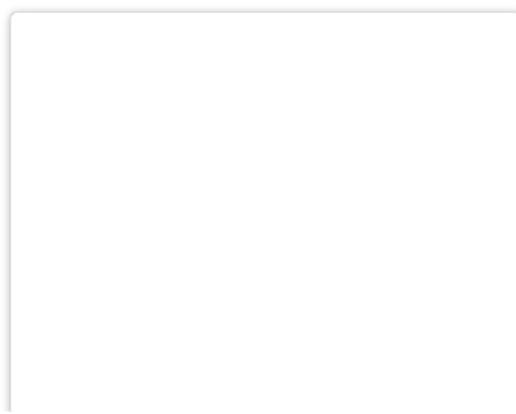
When the **Choose item**



selection window appears, select from the `Template Pipelines` system group. In this instance, as our input data is text, we select (left click) the



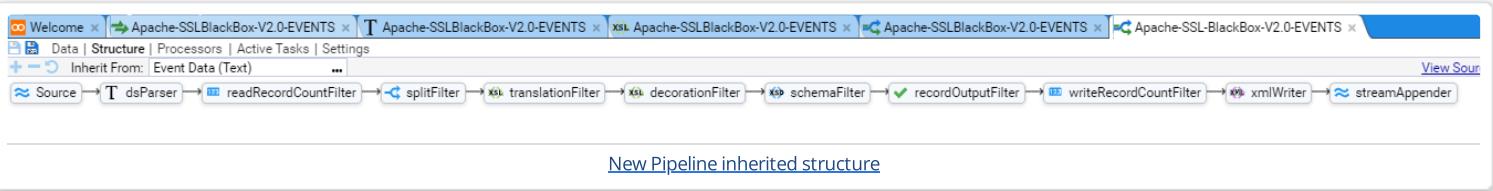
Event Data (Text) pipeline





[New Pipeline inherited selection](#)

then press **OK**. At this we see the inherited pipeline structure of



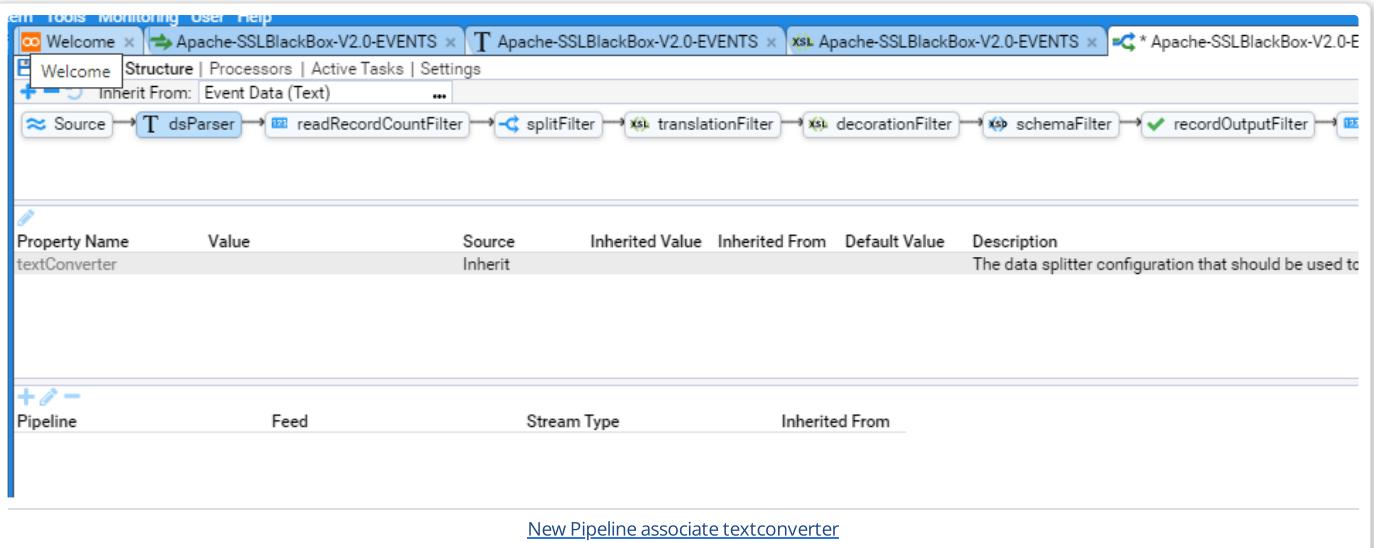
[New Pipeline inherited structure](#)

For the purpose of this HOWTO, we are only interested in two of the eleven (11) elements in this pipeline

- the Text Converter labelled *dsParser*
- the XSLT Translation labelled *translationFilter*

We now need to associate our Text Converter and Translation with the pipeline so that we can pass raw events (logs) through our pipeline in order to save them in the Event Store.

To associate the Text Converter, select the Text Converter icon, to display.



[New Pipeline associate textconverter](#)

Now identify to the **Property** pane (the middle pane of the pipeline configuration tab), then and double click on the *textConverter* Property Name to display the **Edit Property** selection window that allows you to edit the given property

The screenshot shows the Apache-SSLBlackBox-V2.0-EVENTS interface with the 'Edit Property' dialog open. The 'Element Id: dsParser' is selected. The 'Name:' field is set to 'textConverter', 'Source:' is set to 'Inherit', and the 'Value:' field is set to 'None'. A tooltip 'New Pipeline textconverter association' is displayed at the bottom.

We leave the Property **Source:** as Inherit but we need to change the Property **Value:** from *None* to be our newly created Apache-SSLBlackBox-V2.0-EVENTS Text Converter.

To do this, position the cursor over the menu selection



icon to the right of the **Value:** text display box and click to select. Navigate to the **Apache HTTPD** system group then select the Apache-SSLBlackBox-V2.0-EVENTS text Converter

The screenshot shows the 'Choose item' dialog. The 'System' group is expanded, showing 'Event Sources', 'Infrastructure', 'WebServer', and 'Apache HTTPD'. 'Apache-SSLBlackBox-V2.0-EVENTS' is selected under 'Apache HTTPD'. A tooltip 'New Pipeline textconverter association' is displayed at the bottom.

then press **OK**. At this we will see the Property *Value* set

The screenshot shows the 'Edit Property' dialog. The 'Element Id: dsParser' is selected. The 'Name:' field is set to 'textConverter', 'Source:' is set to 'Local', and the 'Value:' field is set to 'Apache-SSLBlackBox-V2.0-EVENTS'. A tooltip 'New Pipeline textconverter association' is displayed at the bottom.

Again press **OK** to finish editing this property and we see that the *textConverter* Property has been set to **Apache-SSLBlackBox-V2.0-EVENTS**

Apache-SSLBlackBox-V2.0-EVENTS

dsParser → readRecordCountFilter → splitFilter → xsl:translationFilter → xsl:decorationFilter → xsl:schemaFilter → recordOutputFilter → writeRecordCountFilter → xmlWriter → streamAppender

Property Name	Value	Source	Inherited Value	Inherited From	Default Value	Description
textConverter	Apache-SSLBlackBox-V2.0-EVENTS	Local				The data splitter configuration that should be used to parse the input data.

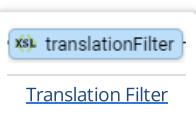
1 to 1 of 1

Pipeline Feed Stream Type Inherited From

[New Pipeline textconverter association](#)

We perform the same actions to associate the translation.

First, we select the translation Filter's



icon and then within translation Filter's **Property** pane we double click on the **xs:t** Property Name to bring up the **Property Editor**. As before, bring up the **Choose item** selection window, navigate to the **Apache HTTPD** system group and select the **Apache-SSLBlackBox-V2.0-EVENTS xslt Translation**.

splitFilter → xsl:translationFilter → xsl:decorationFilter → xsl:schemaFilter → recordOutputFilter

Choose item

Quick Filter: None

- System
 - Event Sources
 - Email
 - Infrastructure
 - WebServer
 - Apache HTTPD
 - xsl:Apache-SSLBlackBox-V2.0-EVENTS

OK Cancel

Source	Inherit
Inherit	false
Inherit	true
Inherit	
Inherit	

Edit Property

Element Id: translationFilter
Name: xs:t
Source: Inherit
Value: None

OK

[New Pipeline Translation association](#)

We leave the remaining properties in the translation Filter's **Property** pane at their default values. The result is the assignment of our translation to the **xs:t** Property.

Apache-SSLBlackBox-V2.0-EVENTS

dsParser → readRecordCountFilter → splitFilter → xsl:translationFilter → xsl:decorationFilter → xsl:schemaFilter → recordOutputFilter

Property Name	Value	Source	Inherited Value	Inherited From	Default Value	Description
suppressXSLTNotFoundWarnings	false	Inherit	false		false	If XSLT cannot be found, do not log an error message.
usePool	true	Inherit	true		true	Advanced: Choose true to use a pool of XSLT documents.
xs:t	Apache-SSLBlackBox-V2.0-EVENTS	Local				The XSLT to use.
xs:tNamePattern		Inherit				A name pattern to use for XSLT files.

Pipeline Feed Stream Type Inherited From

For the moment, we will not associate a decoration filter.

Save the pipeline by clicking on its

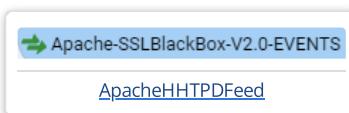


icon.

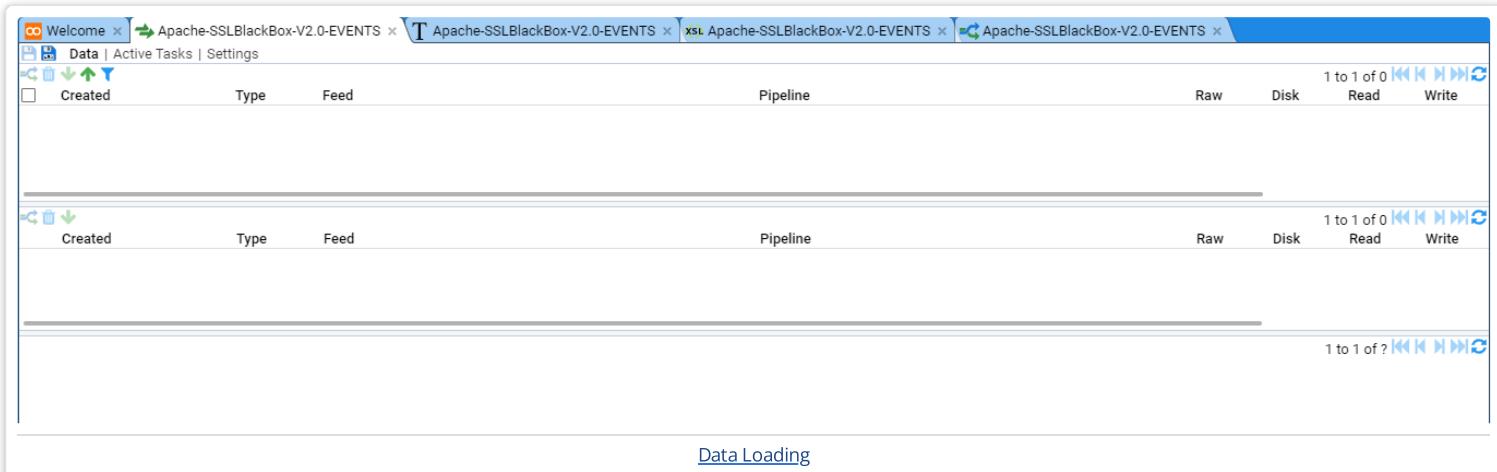
4.6.1.2.6 Manually load Raw Event test data

Having established the pipeline, we can now start authoring our text converter and translation. The first step is to load some Raw Event test data. Previously in the **Event Log Source** of this HOWTO you saved a copy of the file *sampleApacheBlackBox.log* to your local environment. It contains only a few events as the content is consistently formatted. We could feed the test data by posting the file to Stroom's accounting/datafeed url, but for this example we will manually load the file. Once developed, raw data is posted to the web service.

Select the

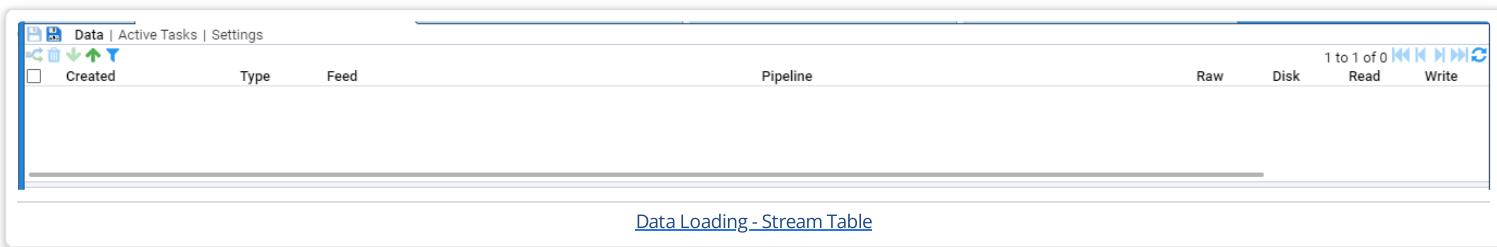


configuration tab and select the **Data** sub-item to display



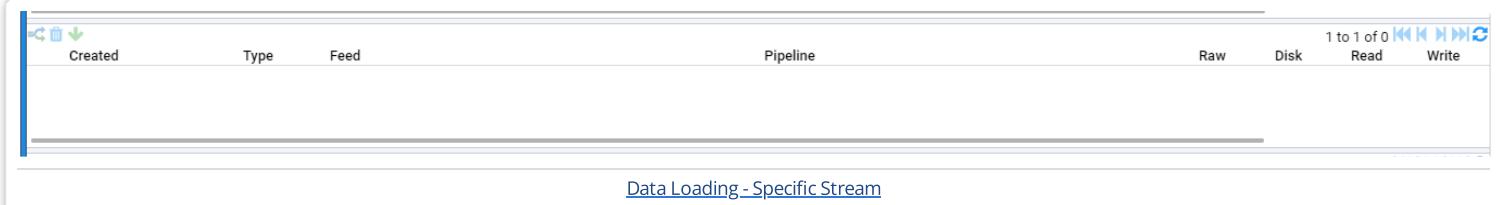
This window is divided into three panes.

The top pane displays the *Stream Table*, which is a table of the latest streams that belong to the feed (clearly it's empty).



Note that a Raw Event *stream* is made up of data from a single file of data or aggregation of multiple data files and also meta-data associated with the data file(s). For example, file names, file size, etc.

The middle pane displays a *Specific* feed and any linked streams. To display a *Specific* feed, you select it from the *Stream Table* above.

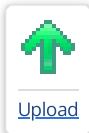


The bottom pane displays the selected stream's data or meta-data.

1 to 1 of ?

[Data Loading - Data/Metadata](#)

Note the Upload icon



[Upload](#)

in the top left of the Stream table pane. On clicking the Upload icon, we are presented with the data **Upload** selection window.

[Data Loading - Upload Data](#)

As stated earlier, raw event data is normally posted as a file to the Stroom web server. As part of this posting action, a set of well-defined *HTTP extra headers* are sent as part of the post. These headers, in the form of key value pairs, provide additional context associated with the system sending the logs. These standard headers become Stroom *feed attributes* available to the Stroom translation. Common attributes are

- System - the name of the System providing the logs
- Environment - the environment of the system (Production, Quality Assurance, Reference, Development)
- Feed - the feedname itself
- MyHost - the fully qualified domain name of the system sending the logs
- MyIPaddress - the IP address of the system sending the logs
- MyNameServer - the name server the system resolves names through

Since our translation will want these feed attributes, we will set them in the Meta Data text entry box of the **Upload** selection window. Note we can skip *Feed* as this will automatically be assigned correctly as part of the upload action (setting it to Apache-SSLBlackBox-V2.0-EVENTS obviously). Our **Meta Data:** will have

- System:LinuxWebServer
- Environment:Production
- MyHost:stroomnode00.strmdev01.org
- MyIPaddress:192.168.2.245
- MyNameServer:192.168.2.254

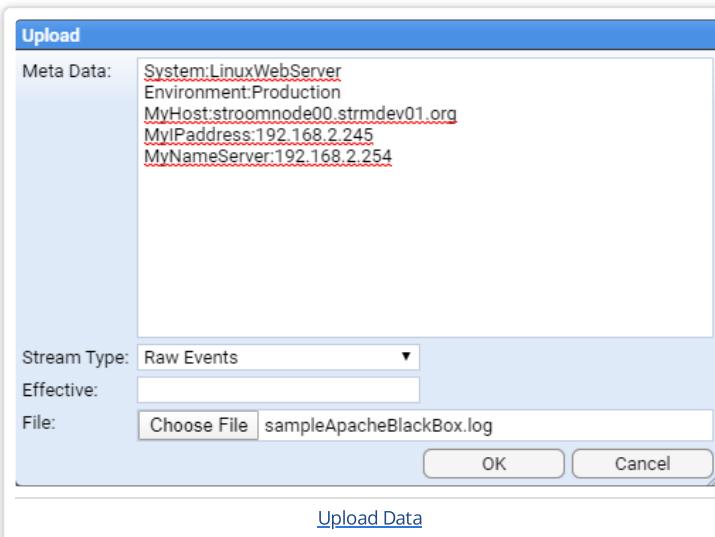
We select a **Stream Type:** of *Raw Events* as this data is for an *Event Feed*. As this is not a *Reference Feed* we ignore the **Effective:** entry box (a date/time selector).



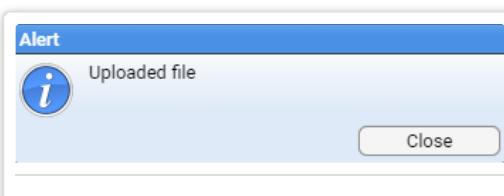
We now click the **Choose File** button, then navigate to the location of the raw log file you downloaded earlier, *sampleApacheBlackBox.log*



then click **Open** to return to the **Upload** selection window where we can then press **OK** to perform the upload.



An Alert dialog window is presented



which should be **closed**.

The stream we have just loaded will now be displayed in the *Streams Table* pane. Note that the *Specific Stream* and *Data/Meta-data* panes are still blank.

The screenshot shows the StreamInsight interface with the Streams Table pane open. There are two streams listed:

- Created**: Type Raw Events, Feed Apache-SSLBlackBox-V2.0-EVENTS. Pipeline status: 1 to 1 of 1 Raw, Disk, Read.
- 2020-03-07T22:48:51.109Z Raw Events**: Type Raw Events, Feed Apache-SSLBlackBox-V2.0-EVENTS. Pipeline status: 1 to 1 of 1 Raw, Disk, Read.

Below the table, there is a link: [Data Loading - Streams Table](#).

If we select the stream by clicking anywhere along its line, the stream is highlighted and the *Specific Stream* and *Data/Meta-data* panes now display data.

The screenshot shows the StreamInsight interface with the Streams Table pane open. One stream is selected:

- Created**: Type Raw Events, Feed Apache-SSLBlackBox-V2.0-EVENTS. Pipeline status: 1 to 1 of 1 Raw, Disk, Read.

The **Data | Meta** pane displays the raw event data:

	Created	Type	Feed	Pipeline
1	2020-03-08T11:04:31.403Z	Raw Events	Apache-SSLBlackBox-V2.0-EVENTS	1 to 1 of 1 Raw, Disk, Read
2	2020-03-08T11:04:31.403Z	Raw Events	Apache-SSLBlackBox-V2.0-EVENTS	1 to 1 of 1 Raw, Disk, Read
3	2020-03-08T11:04:31.403Z	Raw Events	Apache-SSLBlackBox-V2.0-EVENTS	1 to 1 of 1 Raw, Disk, Read
4	2020-03-08T11:04:31.403Z	Raw Events	Apache-SSLBlackBox-V2.0-EVENTS	1 to 1 of 1 Raw, Disk, Read
5	2020-03-08T11:04:31.403Z	Raw Events	Apache-SSLBlackBox-V2.0-EVENTS	1 to 1 of 1 Raw, Disk, Read
6	2020-03-08T11:04:31.403Z	Raw Events	Apache-SSLBlackBox-V2.0-EVENTS	1 to 1 of 1 Raw, Disk, Read
7	2020-03-08T11:04:31.403Z	Raw Events	Apache-SSLBlackBox-V2.0-EVENTS	1 to 1 of 1 Raw, Disk, Read
8	2020-03-08T11:04:31.403Z	Raw Events	Apache-SSLBlackBox-V2.0-EVENTS	1 to 1 of 1 Raw, Disk, Read
9	2020-03-08T11:04:31.403Z	Raw Events	Apache-SSLBlackBox-V2.0-EVENTS	1 to 1 of 1 Raw, Disk, Read
10	2020-03-08T11:04:31.403Z	Raw Events	Apache-SSLBlackBox-V2.0-EVENTS	1 to 1 of 1 Raw, Disk, Read
11	2020-03-08T11:04:31.403Z	Raw Events	Apache-SSLBlackBox-V2.0-EVENTS	1 to 1 of 1 Raw, Disk, Read
12	2020-03-08T11:04:31.403Z	Raw Events	Apache-SSLBlackBox-V2.0-EVENTS	1 to 1 of 1 Raw, Disk, Read
13	2020-03-08T11:04:31.403Z	Raw Events	Apache-SSLBlackBox-V2.0-EVENTS	1 to 1 of 1 Raw, Disk, Read
14	2020-03-08T11:04:31.403Z	Raw Events	Apache-SSLBlackBox-V2.0-EVENTS	1 to 1 of 1 Raw, Disk, Read

At the bottom of the Data/Meta-data pane, there is a link: [UNCLASSIFIED](#).

The *Specific Stream* pane only displays the Raw Event stream and the *Data/Meta-data* pane displays the content of the log file just uploaded (the **Data** link). If we were to click on the **Meta** link at the top of the *Data/Meta-data* pane, the log data is replaced by this stream's meta-data.

```

1 Environment:Production
2 Feed:Apache-SSLBlackBox-V2.0-EVENTS
3 MyHost:stroomnode00.strmdev01.org
4 MyIpAddress:192.168.2.245
5 MyNameServer:192.168.2.254
6 ReceivedTime:2020-03-07T22:48:51.065Z
7 RemoteFile:C:\fakepath\sampleApacheBlackBox.log
8 StreamSize:5367
9 System:LinuxWebServer
10 user-agent:STROOM-UI
11

```



UNCLASSIFIED

[Data Loading - Meta-data](#)

Note that, in addition to the feed attributes we set, the upload process added additional feed attributes of

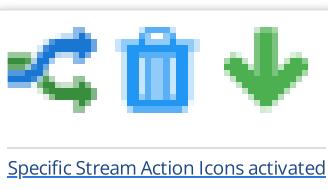
- Feed - the feed name
- ReceivedTime - the time the feed was received by Stroom
- RemoteFile - the name of the file loaded
- StreamSize - the size, in bytes, of the loaded data within the stream
- user-agent - the user agent used to present the stream to Stroom - in this case, the Stroom user Interface

We now have data that will allow us to develop our text converter and translation.

4.6.1.2.7 Step data through Pipeline - Source

We now need to step our data through the pipeline.

To do this, set the check-box on the *Specific Stream* pane and we note that the previously grayed out action icons



Created	Type	Feed	Pipeline
<input checked="" type="checkbox"/> > i 2020-03-07T22:48:51.109Z	Raw Events	Apache-SSLBlackBox-V2.0-EVENTS	

Select Stream to Step

We now want to step our data through the first element of the pipeline, the Text Converter. We enter Stepping Mode by pressing the stepping button



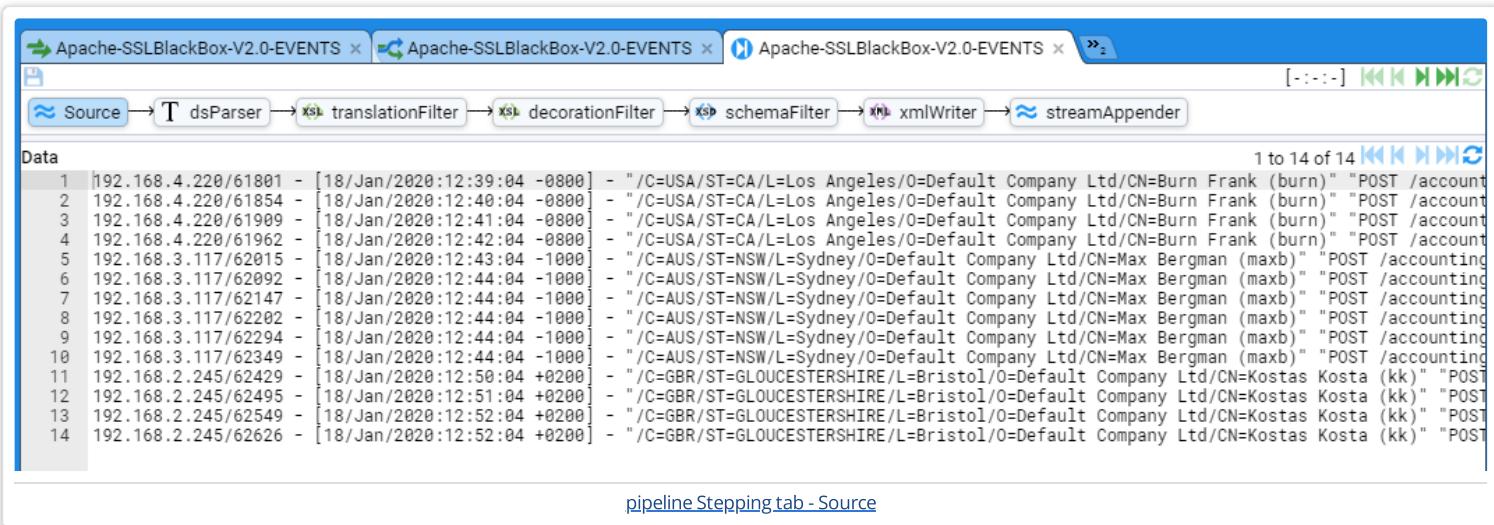
found at the bottom right corner of the *Data/Meta-data* pane.

We will then be requested to choose a pipeline to step with, at which, you should navigate to the Apache-SSLBlackBox-V2.0-EVENTS pipeline as per



then press **OK**.

At this point, we enter the pipeline Stepping tab



which, initially displays the Raw Event data from our stream. This is the Source display for the Event Pipeline.

4.6.1.2.8 Step data through Pipeline - Text Converter

We click on the

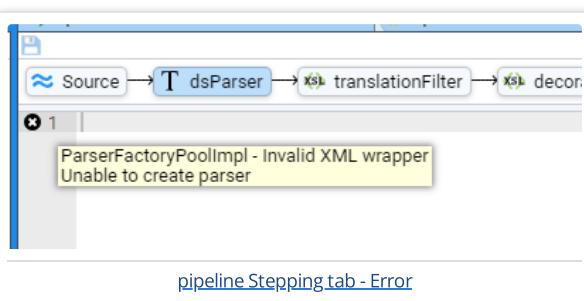


icon to enter the Text Converter stepping window.



This *stepping* tab is divided into three sub-panes. The top one is the Text Converter editor and it will allow you to edit the text conversion. The bottom left window displays the *input* to the Text Converter. The bottom right window displays the *output* from the Text Converter for the given input.

We also note an error indicator - that of an error in the editor pane as indicated by the black back-grounded x and rectangular black boxes to the right of the editor's scroll bar.



In essence, this means that we have no text converter to pass the Raw Event data through.

To correct this, we will author our text converter using the Data Splitter *language*. Normally this is done incrementally to more easily develop the parser. The minimum text converter contains

```
<?xml version="1.1" encoding="UTF-8"?>
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3
    <split delimiter="\n">
        <group>
            <regex pattern="^(.*)$">
                <data name="rest" value="$1" />
            </regex>
        </group>
    </split>
</dataSplitter>
```

If we now press the Step First



icon the error will disappear and the stepping window will show.

Apache-SSLBlackBox-V2.0-EVENTS x Apache-SSLBlackBox-V2.0-EVENTS x * Apache-SSLBlackBox-V2.0-EVENTS x [151842:1:1] 

Source → T dsParser → translationFilter → decorationFilter → schemaFilter → xmlWriter → streamAppender

```

1  <?xml version="1.1" encoding="UTF-8"?>
2  <dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3 file://da
3  <split delimiter="\n">
4  <group>
5  <regex pattern="^(.*$)">
6  <data name="rest" value="$1" />
7  </regex>
8  </group>
9  </split>
10 </dataSplitter>

```

```

[192.168.4.220/61801 - [18/Jan/2020:12:39:04 -0800] - "/C=U
<?xml version="1.1" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="records:2 file://da
<record>
  <data name="rest" value="192.168.4.220/61801 - [18/Jan/2020:12:39:04 -0800] - "/C=U
</record>
</records>

```

[pipeline Stepping tab - Text Converter Simple A](#)

As we can see, the first line of our Raw Event is displayed in the *input* pane and the *output* window holds the converted XML output where we just have a single *data* element with a *name* attribute of *rest* and a *value* attribute of the complete raw event as our regular expression matched the entire line.

The next incremental step in the parser, would be to *parse out* additional *data* elements. For example, in this next iteration we extract the client ip address, the client port and hold the rest of the Event in the rest data element.

With the text converter containing

```

<?xml version="1.1" encoding="UTF-8"?>
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3
  <split delimiter="\n">
    <group>
      <regex pattern="^([^\n]+)([^\n]+)(.*)$">
        <data name="clientip" value="$1" />
        <data name="clientport" value="$2" />
        <data name="rest" value="$3" />
      </regex>
    </group>
  </split>
</dataSplitter>

```

and a click on the Refresh Current Step



icon we will see the *output* pane contain

```

<?xml version="1.1" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="records:2 file://da
<record>
  <data name="clientip" value="192.168.4.220" />
  <data name="clientport" value="61801" />
  <data name="rest" value="- [18/Jan/2020:12:39:04 -0800] - &#34;/C=USA/ST=CA/L=Los Angeles/O=Stroo
</record>
</records>

```

[Text Converter Simple B](#)

We continue this incremental parsing until we have our complete parser.

The following is our complete Text Converter which generates xml records as defined by the Stroom **records v3.0** schema.

```

<?xml version="1.1" encoding="UTF-8"?>
<dataSplitter xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3

<!-- CLASSIFICATION: UNCLASSIFIED -->

<!-- Release History:
Release 20131001, 1 Oct 2013 - Initial release

General Notes:
This data splitter takes audit events for the Stroom variant of the Black Box Apache Auditing.

Event Format: The following is extracted from the Configuration settings for the Stroom variant of the Black Box Apache Auditing

# STROOM - Black Box Auditing configuration
#
# %a - Client IP address (not hostname (%h) to ensure ip address only)
# When logging the remote host, it is important to log the client IP address, not the
# hostname. We do this with the '%a' directive. Even if HostnameLookups are turned on,
# using '%a' will only record the IP address. For the purposes of BlackBox formats,
# reversed DNS should not be trusted

# %{REMOTE_PORT}e - Client source port
# Logging the client source TCP port can provide some useful network data and can help
# one associate a single client with multiple requests.
# If two clients from the same IP address make simultaneous connections, the 'common log'
# file format cannot distinguish between those clients. Otherwise, if the client uses
# keep-alives, then every hit made from a single TCP session will be associated by the same
# client port number.
# The port information can indicate how many connections our server is handling at once,
# which may help in tuning server TCP/OP settings. It will also identify which client ports
# are legitimate requests if the administrator is examining a possible SYN-attack against a
# server.
# Note we are using the REMOTE_PORT environment variable. Environment variables only come
# into play when mod_cgi or mod_cgid is handling the request.

# %X - Connection status (use %c for Apache 1.3)
# The connection status directive tells us detailed information about the client connection.
# It returns one of three flags:
# x if the client aborted the connection before completion,
# + if the client has indicated that it will use keep-alives (and request additional URLs),
# - if the connection will be closed after the event
# Keep-Alive is a HTTP 1.1. directive that informs a web server that a client can request multiple
# files during the same connection. This way a client doesn't need to go through the overhead
# of re-establishing a TCP connection to retrieve a new file.

# %t - time - or [%{d/b/Y:%T}t.%{msec_frac}t %{%z}t] for Apache 2.4
# The %t directive records the time that the request started.
# NOTE: When deployed on an Apache 2.4, or better, environment, you should use
# strftime format in order to get microsecond resolution.

# %l - remote logname
#
# %u - username [in quotes]
# The remote user (from auth; This may be bogus if the return status (%s) is 401
# for non-ssl services)
# For SSL services, user names need to be delivered as DNs to deliver PKI user details
# in full. To pass through PKI certificate properties in the correct form you need to
# add the following directives to your Apache configuration:
# SSLUserName SSL_CLIENT_S_DN
# SSLOptions +StdEnvVars
# If you cannot, then use %{SSL_CLIENT_S_DN}x in place of %u and use blackboxSSLUser
# LogFormat nickname

# %r - first line of text sent by web client [in quotes]

```

```

# This is the first line of text send by the web client, which includes the request
# method, the full URL, and the HTTP protocol.

# %s - status code before any redirection
# This is the status code of the original request.

# %>s - status code after any redirection has taken place
# This is the final status code of the request, after any internal redirections may
# have taken place.

# %D - time in microseconds to handle the request
# This is the number of microseconds the server took to handle the request in microseconds

# %I - incoming bytes
# This is the bytes received, include request and headers. It cannot, by definition be zero.

# %O - outgoing bytes
# This is the size in bytes of the outgoing data, including HTTP headers. It cannot, by
# definition be zero.

# %B - outgoing content bytes
# This is the size in bytes of the outgoing data, EXCLUDING HTTP headers. Unlike %b, which
# records '-' for zero bytes transferred, %B will record '0'.

# %{Referer}i - Referrer HTTP Request Header [in quotes]
# This is typically the URL of the page that made the request. If linked from
# e-mail or direct entry this value will be empty. Note, this can be spoofed
# or turned off

# %{User-Agent}i - User agent HTTP Request Header [in quotes]
# This is the identifying information the client (browser) reports about itself.
# It can be spoofed or turned off

# %V - the server name according to the UseCanonicalName setting
# This identifies the virtual host in a multi host webservice

# %p - the canonical port of the server servicing the request

# Define a variation of the Black Box logs
#
# Note, you only need to use the 'blackboxSSLUser' nickname if you cannot set the
# following directives for any SSL configurations
# SSLUserName SSL_CLIENT_S_DN
# SSLOptions +StdEnvVars
# You will also note the variation for no logio module. The logio module supports
# the %I and %O formatting directive
#
<IfModule mod_logio.c>
LogFormat "%a/%{REMOTE_PORT}e %X %t %l \"%u\" \"%r\" %s/%>s %D I/0/0/B \"%{Referer}i\" \"%{User-Agent}i\" %V/%p" blackboxUser
LogFormat "%a/%{REMOTE_PORT}e %X %t %l \"%{SSL_CLIENT_S_DN}x\" \"%r\" %s/%>s %D %I/0/0/B \"%{Referer}i\" \"%{User-Agent}i\" %V/%p"
</IfModule>
<IfModule !mod_logio.c>
LogFormat "%a/%{REMOTE_PORT}e %X %t %l \"%u\" \"%r\" %s/%>s %D 0/0/0/B \"%{Referer}i\" \"%{User-Agent}i\" %V/$p" blackboxUser
LogFormat "%a/%{REMOTE_PORT}e %X %t %l \"%{SSL_CLIENT_S_DN}x\" \"%r\" %s/%>s %D 0/0/0/B \"%{Referer}i\" \"%{User-Agent}i\" %V/$p"
</IfModule> -->

<!-- Match line -->
<split delimiter="\n">
  <group>
    <regex pattern="^([/]+)([^ ]+)([^ ]+)([([([^\n]+)])] ([^ ]+) &#34;([^\n]+)&#34; &#34;([^\n]+)&#34; (\d+)/(\d+) (\d+)">
      <data name="clientip" value="$1" />
      <data name="clientport" value="$2" />
      <data name="constatus" value="$3" />
      <data name="time" value="$4" />
      <data name="remotelname" value="$5" />
      <data name="user" value="$6" />
    </group>
  </split>
</IfModule>
```

```

<data name="url" value="$7">
    <group value="$7" ignoreErrors="true">
        <!--
            Special case the "GET /" url string as opposed to the more standard "method url protocol/protocol_version".
            Also special case a url of "-" which occurs on some errors (eg 408)
        -->
        <regex pattern="^-$">
            <data name="url" value="error" />
        </regex>
        <regex pattern="^([^\ ]+) (/)$">
            <data name="httpMethod" value="$1" />
            <data name="url" value="$2" />
        </regex>
        <regex pattern="^([^\ ]+) ([^\ ]+) ([^\ /]*)([^\ ]*)">
            <data name="httpMethod" value="$1" />
            <data name="url" value="$2" />
            <data name="protocol" value="$3" />
            <data name="version" value="$4" />
        </regex>
    </group>
</data>
<data name="responseB" value="$8" />
<data name="response" value="$9" />
<data name="timeM" value="$10" />
<data name="bytesIn" value="$11" />
<data name="bytesOut" value="$12" />
<data name="bytesOutContent" value="$13" />
<data name="referer" value="$14" />
<data name="userAgent" value="$15" />
<data name="vserver" value="$16" />
<data name="vserverport" value="$17" />
</regex>
</group>
</split>
</dataSplitter>

```

A copy of this Data Splitter can be found [here](#).

If we now press the Step First



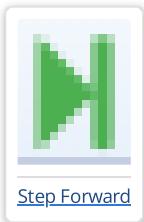
icon we will see the complete parsed record

```

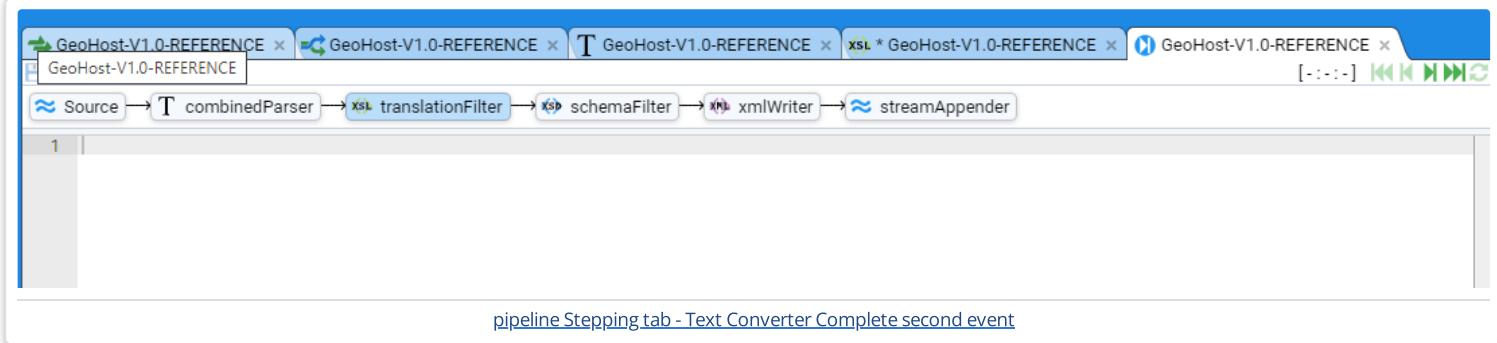
<?xml version="1.1" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="records:2 file://recor
<record>
    <data name="clientip" value="192.168.4.220" />
    <data name="clientport" value="61801" />
    <data name="constatus" value="-" />
    <data name="time" value="18/Jan/2020:12:39:04 -0800" />
    <data name="remotelname" value="-" />
    <data name="user" value="/C=USA/ST=CA/L=Los Angeles/O=Default Company Ltd/CN=Burn Frank (burn)" />
    <data name="url" value="POST /accounting/ui/dispatch.rpc HTTP/1.1">
        <data name="httpMethod" value="POST" />
        <data name="url" value="/accounting/ui/dispatch.rpc" />
        <data name="protocol" value="HTTP" />
        <data name="version" value="1.1" />
    </data>
    <data name="responseB" value="200" />
    <data name="response" value="200" />
    <data name="timeM" value="21221" />
    <data name="bytesIn" value="2289" />
    <data name="bytesOut" value="415" />
    <data name="bytesOutContent" value="14" />
    <data name="referer" value="https://host01.company4.org/accounting/" />
    <data name="userAgent" value="Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.21
    <data name="vserver" value="host01.company4.org" />
    <data name="vserverport" value="443" />
</record>
</records>

```

If we click on the Step Forward



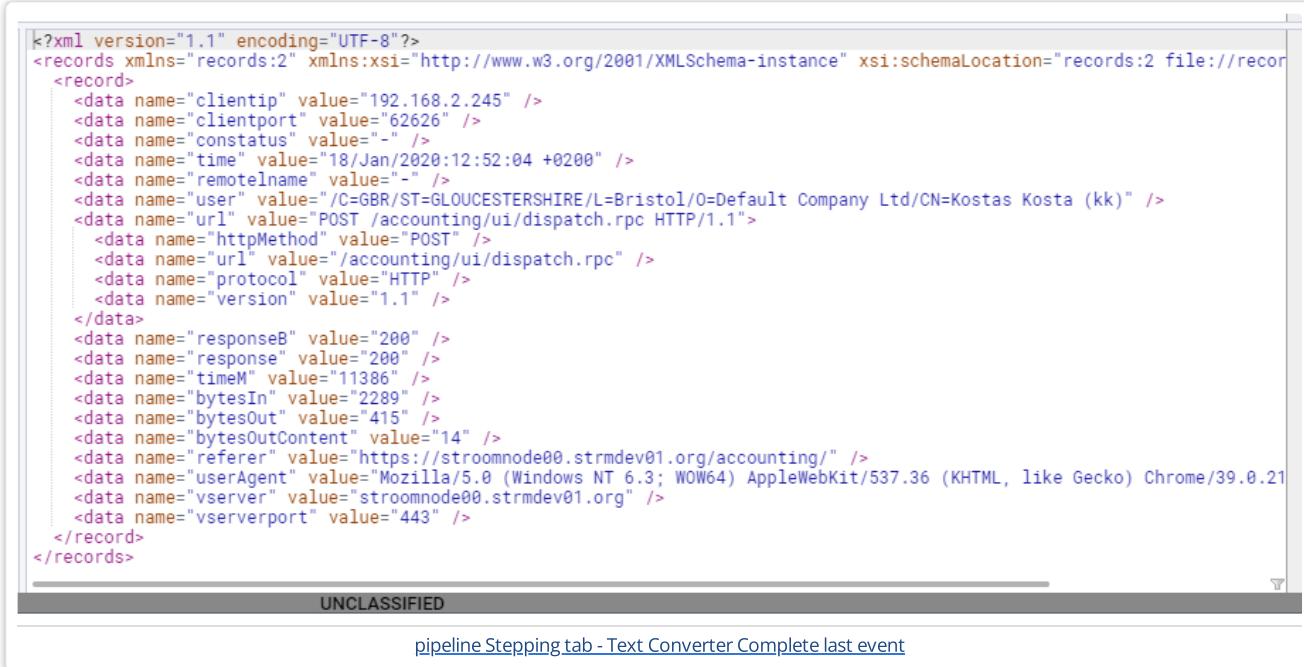
icon we will see the next event displayed in both the *input* and *output* panes.



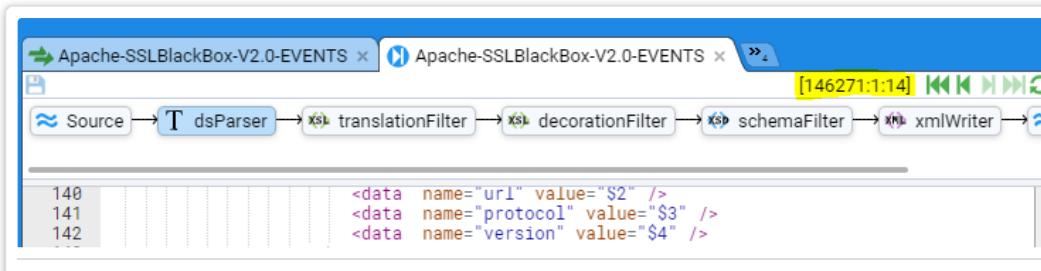
we click on the Step Last



icon we will see the last event displayed in both the *input* and *output* panes.



You should take note of the stepping key that has been displayed in each stepping window. The stepping key are the numbers enclosed in square brackets e.g. [146271:1:14] found in the top right-hand side of the stepping window next to the stepping icons



The form of these keys is [streamId ":" subStreamId ":" recordNo]

where

- **streamId** - is the stream ID and won't change when stepping through the selected stream.
- **subStreamId** - is the sub stream ID. When Stroom processes event streams it aggregates multiple input files and this is the file number.
- **recordNo** - is the record number within the sub stream.

One can double click on either the **subStreamId** or **recordNo** numbers and enter a new number. This allows you to 'step' around a stream rather than just relying on first, previous, next and last movement.

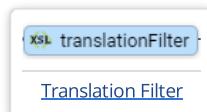
Note, you should now Save



your edited Text Converter.

4.6.1.2.9 Step data through Pipeline - Translation

To start authoring the xslt Translation Filter, press the



icon which steps us to the xsl Translation Filter pane.

[pipeline Stepping tab - Translation Initial](#)

As for the *Text Converter* stepping tab, this tab is divided into three sub-panes. The top one is the xslt translation editor and it will allow you to edit the xslt translation. The bottom left window displays the *input* to the xslt translation (which is the output from the *Text Converter*). The bottom right window displays the *output* from the xslt Translation filter for the given input.

We now click on the pipeline Step Forward button



to single step the Text Converter *records* element data through our xslt Translation. We see no change as an empty translation will just perform a copy of the input data.

To correct this, we will author our xslt translation. Like the Data Splitter this is also authored incrementally. A minimum xslt translation might contain

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xpath-default-namespace="records:2" xmlns="event-logging:3" xmlns:stroom="stroom" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <!-- Ingest the records tree -->
    <xsl:template match="records">
        <Events xsi:schemaLocation="event-logging:3 file://event-logging-v3.2.3.xsd" Version="3.2.3">
            <xsl:apply-templates />
        </Events>
    </xsl:template>

    <!-- Only generate events if we have an url on input -->
    <xsl:template match="record[data[@name = 'url']]">
        <Event>
            <xsl:apply-templates select=". " mode="eventTime" />
            <xsl:apply-templates select=". " mode="eventSource" />
            <xsl:apply-templates select=". " mode="eventDetail" />
        </Event>
    </xsl:template>

    <xsl:template match="node()" mode="eventTime">
        <EventTime>
            <TimeCreated/>
        </EventTime>
    </xsl:template>

    <xsl:template match="node()" mode="eventSource">
        <EventSource>
            <System>
                <Name />
                <Environment />
            </System>
            <Generator />
            <Device />
            <Client />
            <Server />
            <User>
                <Id />
            </User>
        </EventSource>
    </xsl:template>

    <xsl:template match="node()" mode="eventDetail">
        <EventDetail>
            <View>
                <WebPage>
                    <Type>WebPage</Type>
                </WebPage>
            </View>
        </EventDetail>
    </xsl:template>
</xsl:stylesheet>
```

Sols Monitoring User Help

Apache-SSLBlackBox-V2.0-EVENTS xsl Apache-SSLBlackBox-V2.0-EVENTS x Apache-SSLBlackBox-V2.0-EVENTS x * Apache-SSLBlackBox-V2.0-EVENTS x [151842:1:14]

The screenshot shows the Fiddler interface with several tabs open. The main pane displays an XSLT transformation process:

```

<xsl:template match="node()" mode="eventDetail">
    <EventDetail>
        <View>
            <WebPage>
                <Type>WebPage</Type>
    
```

Below this is the raw XML log:

```

<?xml version="1.1" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="records:2">
    <record>
        <data name="clientip" value="192.168.2.245" />
        <data name="clientport" value="62626" />
        <data name="constatus" value=" " />
        <data name="time" value="18/Jan/2020:12:52:04 +0200" />
        <data name="remoteName" value=" " />
        <data name="url" value="/C=GBR/ST=GLOUCESTERSHIRE/L=Bristol/O=Default Company Ltd/CN=...>
        <data name="httpMethod" value="POST" />
        <data name="url" value="/accounting/ui/dispatch.rpc" />
        <data name="protocol" value="HTTP" />
        <data name="version" value="1.1" />
    </data>
    <data name="responseB" value="200" />
    <data name="response" value="200" />
    <data name="timeM" value="11386" />
    <data name="bytesIn" value="2289" />
    <data name="bytesOut" value="415" />
    <data name="bytesOutContent" value="14" />
    <data name="referer" value="https://stroomnode00.strmdev01.org/accounting/" />
    <data name="userAgent" value="Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.147 Safari/537.36" />
    <data name="vserver" value="stroomnode00.strmdev01.org" />
    <data name="vserverPort" value="443" />
  </record>
</records>

```

To the right, the transformed XML output is shown:

```

<?xml version="1.1" encoding="UTF-8"?>
<Events xmlns="event-logging:3" xmlns:stroom="stroom" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="event-logging:3 stroom">
    <Event>
        <EventTime>
            <TimeCreated>2020-01-18T20:39:04.000Z</TimeCreated>
        </EventTime>
        <EventSource>
            <System>
                <Name />
                <Environment />
            </System>
            <Generator />
            <Device />
            <Client />
            <Server />
            <User>
                <Id />
            </User>
        </EventSource>
        <EventDetail>
            <View>
                <WebPage>
                    <Type>WebPage</Type>
                </WebPage>
            </View>
        </EventDetail>
    </Event>
</Events>

```

At the bottom, the status is labeled "UNCLASSIFIED".

Clearly this doesn't generate useful events. Our first iterative change might be to generate the TimeCreated element value. The change would be

```

<xsl:template match="node()" mode="eventTime">
    <EventTime>
        <TimeCreated>
            <xsl:value-of select="stroom:format-date(data[@name = 'time']/@value, 'dd/MMM/yyyy:HH:mm:ss XX')"/>
        </TimeCreated>
    </EventTime>
</xsl:template>

```

The screenshot shows the Fiddler interface with the XSLT transformation updated to include the TimeCreated element:

```

<xsl:template match="node()" mode="eventTime">
    <EventTime>
        <TimeCreated>2020-01-18T20:39:04.000Z</TimeCreated>
    </EventTime>

```

Below this is the resulting XML event:

```

<?xml version="1.1" encoding="UTF-8"?>
<Events xmlns="event-logging:3" xmlns:stroom="stroom" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="event-logging:3 stroom">
    <Event>
        <EventTime>
            <TimeCreated>2020-01-18T20:39:04.000Z</TimeCreated>
        </EventTime>
        <EventSource>
            <System>
                <Name />
                <Environment />
            </System>
            <Generator />
            <Device />
            <Client />
            <Server />
            <User>
                <Id />
            </User>
        </EventSource>
        <EventDetail>
            <View>
                <WebPage>
                    <Type>WebPage</Type>
                </WebPage>
            </View>
        </EventDetail>
    </Event>
</Events>

```

At the bottom, the status is labeled "UNCLASSIFIED".

Adding in the EventSource elements (without ANY error checking!) as per

```

<xsl:template match="node()" mode="eventSource">
  <EventSource>
    <System>
      <Name>
        <xsl:value-of select="stroom:feed-attribute('System')"/>
      </Name>
      <Environment>
        <xsl:value-of select="stroom:feed-attribute('Environment')"/>
      </Environment>
    </System>
    <Generator>Apache HTTPD</Generator>
    <Device>
      <HostName>
        <xsl:value-of select="stroom:feed-attribute('MyHost')"/>
      </HostName>
      <IPAddress>
        <xsl:value-of select="stroom:feed-attribute('MyIPAddress')"/>
      </IPAddress>
    </Device>
    <Client>
      <IPAddress>
        <xsl:value-of select="data[@name = 'clientip']/@value"/>
      </IPAddress>
      <Port>
        <xsl:value-of select="data[@name = 'clientport']/@value"/>
      </Port>
    </Client>
    <Server>
      <HostName>
        <xsl:value-of select="data[@name = 'vserver']/@value"/>
      </HostName>
      <Port>
        <xsl:value-of select="data[@name = 'vserverport']/@value"/>
      </Port>
    </Server>
    <User>
      <Id>
        <xsl:value-of select="data[@name='user']/@value"/>
      </Id>
    </User>
  </EventSource>
</xsl:template>

```

And after a Refresh Current Step



[Step Refresh](#)

we see our output event 'grow' to



```
<?xml version="1.1" encoding="UTF-8"?>
<Events xmlns="event-logging:3" xmlns:stroom="stroom" xmlns:xsi="http://www.w3.org/2001/XMLSchema">
<Event>
  <EventTime>
    <TimeCreated>2020-01-18T20:39:04.000Z</TimeCreated>
  </EventTime>
  <EventSource>
    <System>
      <Name>LinuxWebServer</Name>
      <Environment>Production</Environment>
    </System>
    <Generator>Apache HTTPD</Generator>
    <Device>
      <HostName>stroomnode00.strmdev01.org</HostName>
      <IPAddress>192.168.2.245</IPAddress>
    </Device>
    <Client>
      <IPAddress>192.168.4.220</IPAddress>
      <Port>61801</Port>
    </Client>
    <Server>
      <HostName>host01.company4.org</HostName>
      <Port>443</Port>
    </Server>
    <User>
      <Id>/C=USA/ST=CA/L=Los Angeles/O=Default Company Ltd/CN=Burn Frank (burn)</Id>
    </User>
  </EventSource>
  <EventDetail>
    <View>
      <WebPage>
        <Type>WebPage</Type>
      </WebPage>
    </View>
  </EventDetail>
</Event>
```

INCLASSIFIED

[Translation Minimal++](#)

We now complete our translation by expanding the *EventDetail* elements to have the completed translation of (again with limited error checking and non-existent documentation!)

```

<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xpath-default-namespace="records:2" xmlns="event-logging:3" xmlns:stroom="stroom" xmlns:xsl="http://www.w3.org/1

<!-- Ingest the records tree -->
<xsl:template match="records">
  <Events xsi:schemaLocation="event-logging:3 file://event-logging-v3.2.3.xsd" Version="3.2.3">
    <xsl:apply-templates />
  </Events>
</xsl:template>

<!-- Only generate events if we have an url on input -->
<xsl:template match="record[data[@name = 'url']]">
  <Event>
    <xsl:apply-templates select=".." mode="eventTime" />
    <xsl:apply-templates select=".." mode="eventSource" />
    <xsl:apply-templates select=".." mode="eventDetail" />
  </Event>
</xsl:template>

<xsl:template match="node()" mode="eventTime">
  <EventTime>
    <TimeCreated>
      <xsl:value-of select="stroom:format-date(data[@name = 'time']/@value, 'dd/MMM/yyyy:HH:mm:ss XX')"/>
    </TimeCreated>
  </EventTime>
</xsl:template>

<xsl:template match="node()" mode="eventSource">
  <EventSource>
    <System>
      <Name>
        <xsl:value-of select="stroom:feed-attribute('System')"/>
      </Name>
      <Environment>
        <xsl:value-of select="stroom:feed-attribute('Environment')"/>
      </Environment>
    </System>
    <Generator>Apache HTTPD</Generator>
    <Device>
      <HostName>
        <xsl:value-of select="stroom:feed-attribute('MyHost')"/>
      </HostName>
      <IPAddress>
        <xsl:value-of select="stroom:feed-attribute('MyIPAddress')"/>
      </IPAddress>
    </Device>
    <Client>
      <IPAddress>
        <xsl:value-of select="data[@name = 'clientip']/@value"/>
      </IPAddress>
      <Port>
        <xsl:value-of select="data[@name = 'clientport']/@value"/>
      </Port>
    </Client>
    <Server>
      <HostName>
        <xsl:value-of select="data[@name = 'vserver']/@value"/>
      </HostName>
      <Port>
        <xsl:value-of select="data[@name = 'vserverport']/@value"/>
      </Port>
    </Server>
    <User>
      <Id>
        <xsl:value-of select="data[@name='user']/@value"/>
      </Id>
    </User>
  </EventSource>
</xsl:template>

```

```

</Id>
</User>
</EventSource>
</xsl:template>

<!-- -->
<xsl:template match="node()" mode="eventDetail">
<EventDetail>
<View>
<WebPage>
<Type>WebPage</Type>
<URL>
<xsl:value-of select="data[@name = 'url']/data[@name = 'url']/@value" />
</URL>
<xsl:if test="data[@name = 'referer']/@value != '-'">
<Referrer>
<xsl:value-of select="data[@name = 'referer']/@value" />
</Referrer>
</xsl:if>
<HTTPMethod>
<xsl:value-of select="data[@name = 'url']/data[@name = 'httpMethod']/@value" />
</HTTPMethod>
<UserAgent>
<xsl:value-of select="data[@name = 'userAgent']/@value" />
</UserAgent>
<ResponseCode>
<xsl:value-of select="data[@name = 'response']/@value" />
</ResponseCode>

<!-- Protocol -->
<Data Name="Protocol">
<xsl:attribute name="Value" select="data[@name = 'url']/data[@name = 'protocol']/@value" />
</Data>

<!-- Protocol Version -->
<Data Name="Version">
<xsl:attribute name="Value" select="data[@name = 'url']/data[@name = 'version']/@value" />
</Data>

<!-- Response Code Before -->
<Data Name="ResponseCodeBefore">
<xsl:attribute name="Value" select="data[@name = 'responseB']/@value" />
</Data>

<!-- Connection Status -->
<Data Name="ConnectionStatus">
<xsl:attribute name="Value" select="data[@name = 'constatus']/@value" />
</Data>

<!-- Bytes transferred -->
<xsl:if test="data[@name = 'bytesIn']/@value != '0' and data[@name = 'bytesIn']/@value != '-'">
<Data Name="BytesIn">
<xsl:attribute name="Value" select="data[@name = 'bytesIn']/@value" />
</Data>
</xsl:if>
<xsl:if test="data[@name = 'bytesOut']/@value != '0' and data[@name = 'bytesOut']/@value != '-'">
<Data Name="BytesOut">
<xsl:attribute name="Value" select="data[@name = 'bytesOut']/@value" />
</Data>
</xsl:if>
<xsl:if test="data[@name = 'bytesOutContent']/@value != '0'">
<Data Name="BytesOutContentOnly">
<xsl:attribute name="Value" select="data[@name = 'bytesOutContent']/@value" />
</Data>
</xsl:if>

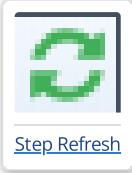
```

```

<!-- Time to serve Request -->
<xsl:if test="data[@name = 'timeM']/@value != '0'>
<Data Name="TotalRequestTimeMicroseconds">
    <xsl:attribute name="Value" select="data[@name = 'timeM']/@value" />
</Data>
</xsl:if>
</WebPage>
</View>
</EventDetail>
</xsl:template>
</xsl:stylesheet>

```

And after a Refresh Current Step Refresh



we see our complete output event

```

<?xml version="1.1" encoding="UTF-8"?>
<Events xmlns="event-logging:3" xmlns:stroom="stroom" xmlns:xsi="http://www.w3.org/2001/XMLSchema"
<Event>
    <EventTime>
        <TimeCreated>2020-01-18T20:39:04.000Z</TimeCreated>
    </EventTime>
    <EventSource>
        <System>
            <Name>LinuxWebServer</Name>
            <Environment>Production</Environment>
        </System>
        <Generator>Apache HTTPD</Generator>
        <Device>
            <HostName>stroomnode00.strmdev01.org</HostName>
            <IPAddress>192.168.2.245</IPAddress>
        </Device>
        <Client>
            <IPAddress>192.168.4.220</IPAddress>
            <Port>61801</Port>
        </Client>
        <Server>
            <HostName>host01.company4.org</HostName>
            <Port>443</Port>
        </Server>
        <User>
            <Id>/C=USA/ST=CA/L=Los Angeles/O=Default Company Ltd/CN=Burn Frank (burn)</Id>
        </User>
    </EventSource>
    <EventDetail>
        <View>
            <WebPage>
                <Type>WebPage</Type>
            </WebPage>
        </View>
    </EventDetail>
</Event>
</Events>

```

[Translation Complete](#)

Note, you should now Save



your edited xslt Translation.

A copy of this XSLT Translation can be found [here](#).

We have completed the translation and have completed developing our Apache-SSLBlackBox-V2.0-EVENTS event feed.

At this point, this event feed is set up to accept Raw Event data, but it will not automatically process the raw data and hence it will not place events into the Event Store. To have Stroom automatically process Raw Event streams, you will need to enable Processors for this pipeline.

4.6.2 - Event Processing

This HOWTO is provided to assist users in setting up Stroom to process inbound raw event logs and transform them into the Stroom Event Logging XML Schema.

4.6.2.1 Assumptions

The following assumptions are used in this document.

- the user successfully deployed Stroom
- the following Stroom content packages have been installed
 - Template Pipelines
 - XML Schemas

4.6.2.2 Introduction

This HOWTO will demonstrate the process by which an Event Processing pipeline for a given Event Source is developed and deployed.

The sample event source used will be based on BlueCoat Proxy logs. An extract of BlueCoat logs were sourced from [log-sharing.dreamhosters.com \(external link\)](#) (a Public Security Log Sharing Site) but modified to add sample user attribution.

Template pipelines are being used to simplify the establishment of this processing pipeline.

The sample BlueCoat Proxy log will be transformed into an intermediate simple XML key value pair structure, then into the [Stroom Event Logging XML Schema \(external link\)](#) format.

4.6.2.3 Event Source

As mentioned, we will use BlueCoat Proxy logs as a sample event source. Although BlueCoat logs can be customised, the default is to use the W2C Extended Log File Format (ELF). Our sample data set looks like

```
#Software: SGOS 3.2.4.28
#Version: 1.0
#Date: 2005-04-27 20:57:09
#Fields: date time time-taken c-ip sc-status s-action sc-bytes cs-bytes cs-method cs-uri-scheme cs-host cs-uri-path cs-uri-query
2005-05-04 17:16:12 1 45.110.2.82 200 TCP_HIT 941 729 GET http www.inmabus.com /wcm/assets/images/imagefileicon.gif - george DIF
2005-05-04 17:16:12 2 45.110.2.82 200 TCP_HIT 941 729 GET http www.inmabus.com /wcm/assets/images/imagefileicon.gif - george DIF
...
...
```

A copy of this sample data source can be found [here](#). Later in this HOWTO, one will be required to upload this file. If you save this file now, ensure the file is saved as a text document with ANSI encoding.

4.6.2.4 Establish the Processing Pipeline

We will create the components that make up the processing pipeline for transforming these raw logs into the Stroom Event Logging XML Schema. They will be placed a folder appropriately named **BlueCoat** in the path **System/Event Sources/Proxy**. See [Folder Creation](#) for details on creating such a folder.

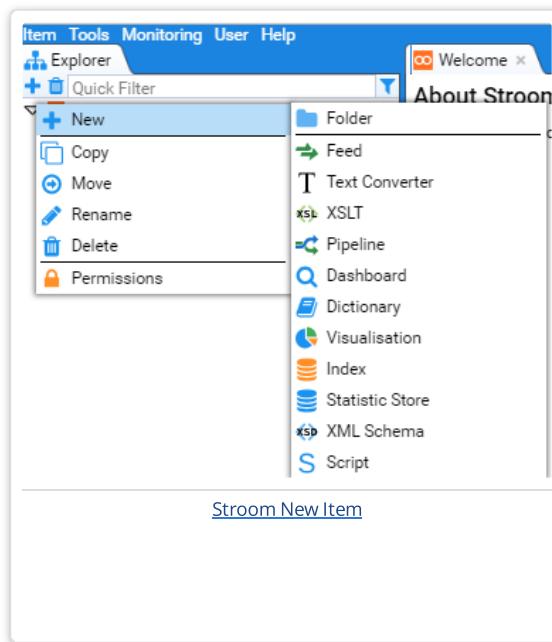
There will be four components

- the Event Feed to group the BlueCoat log files
- the Text Converter to convert the BlueCoat raw logs files into simple XML
- the XSLT Translation to translate the simple XML formed by the Text Converter into the Stroom Event Logging XML form, and
- the Processing pipeline which manages how the processing is performed.

All components will have the same Name **BlueCoat-Proxy-V1.0-EVENTS**. It should be noted that the default Stroom FeedName pattern will not accept this name. One needs to modify the `stroom.feedNamePattern` stroom property to change the default pattern to `^[a-zA-Z0-9_-\.\.]{3,}\$`. See the [HOWTO on System Properties](#) document to see how to make this change.

4.6.2.4.1 Create the Event Feed

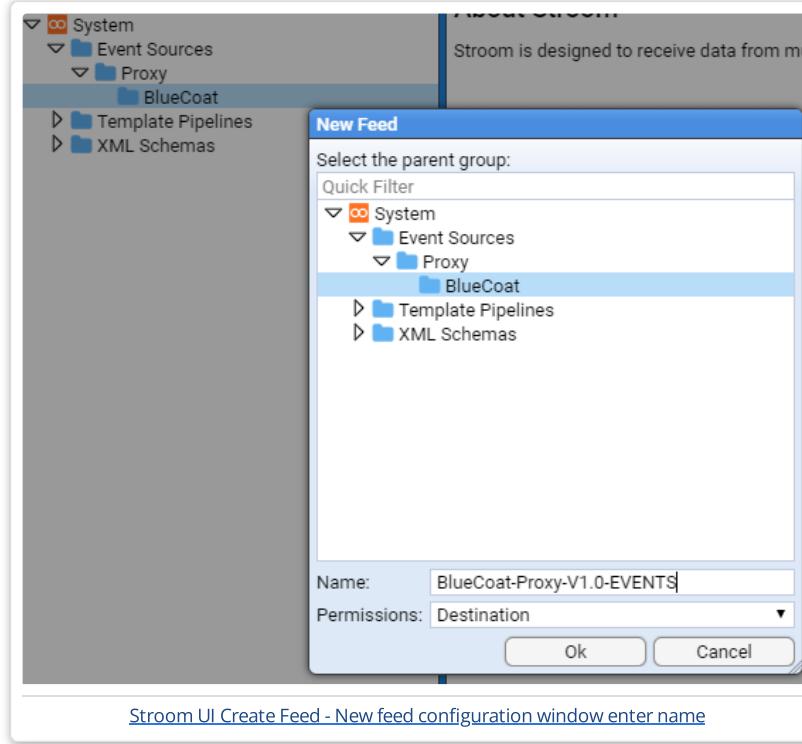
We first select (with a *left click*) the **System/Event Sources/Proxy/BlueCoat** folder in the **Explorer** tab then *right click* to bring up the **New Item** selection sub-menu.



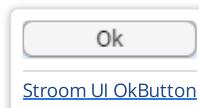
As we are creating an Event Feed, select the



item to have the **New Feed** configuration window into which we enter **BlueCoat-Proxy-V1.0-EVENTS** into the **Name:** entry box



and press



to see the new Event Feed tab

The screenshot shows the 'Settings' tab of the 'BlueCoat-Proxy-V1.0-EVENTS' feed configuration. The 'Description' field is empty. Other settings are as follows:

- Classification: Unchecked
- Reference Feed: Unchecked
- Feed Status: Receive
- Stream Type: Raw Events
- Data Encoding: UTF-8
- Context Encoding: UTF-8
- Retention Period: Forever

and its corresponding reference in the `Explorer` display.

The configuration items for a Event Feed are

- Description - a description of the feed
- Classification - the classification or sensitivity of the Event Feed data
- Reference Feed Flag - to indicate if this is a Reference Feed or not
- Feed Status - which indicates if we accept data, reject it or silently drop it
- Stream Type - to indicate if the Feed contains raw log data or reference data
- Data Encoding - the character encoding of the data being sent to the Feed
- Context Encoding - the character encoding of context data associated with this Feed
- Retention Period - the amount of time to retain the Event data

In our example, we will set the above to

- Description - *BlueCoat Proxy log data sent in W2C Extended Log File Format (ELFF)*
- Classification - We will leave this blank
- Reference Feed Flag - We leave the check-box unchecked as this is not a *Reference Feed*
- Feed Status - We set to *Receive*
- Stream Type - We set to *Raw Events* as we will be sending batches (streams) of raw event logs
- Data Encoding - We leave at the default of *UTF-8* as this is the proposed character encoding
- Context Encoding - We leave at the default of *UTF-8* as there are no Context Events for this Feed
- Retention Period - We leave at *Forever* as we do not want to delete any collected BlueCoat event data.

The screenshot shows the 'Settings' tab of the 'BlueCoat-Proxy-V1.0-EVENTS' feed configuration. The 'Description' field now contains the value "BlueCoat Proxy log data sent in W2C Extended Log File Format (ELFF)". Other settings are identical to the previous screenshot.

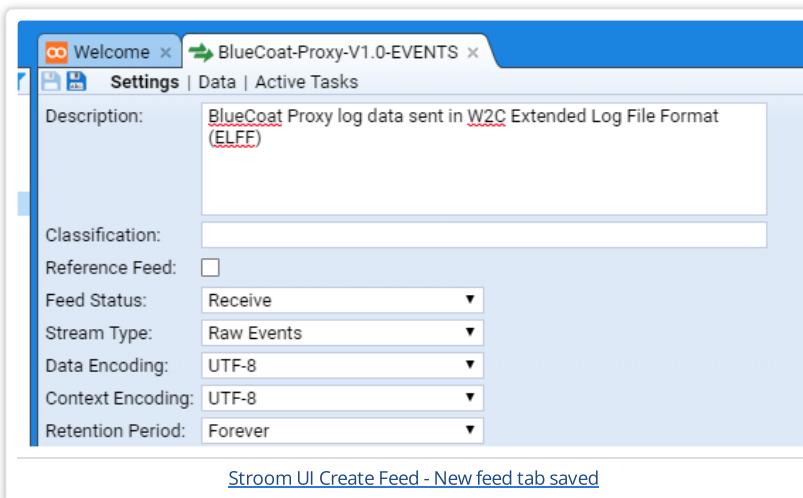
One should note that the `Feed` tab has been marked as having unsaved changes. This is indicated by the asterisk character * between the `Feed` icon



and the name of the feed **BlueCoat-Proxy-V1.0-EVENTS**. We can save the changes to our feed by pressing the *Save* icon



in the top left of the **BlueCoat-Proxy-V1.0-EVENTS** tab. At this point one should notice two things, the first is that the asterisk has disappeared from the `Feed` tab and the second is that the *Save* icon



4.6.2.4.2 Create the Text Converter

We now create the Text Converter for this `Feed` in a similar fashion to the `Event Feed`. We first select (with a *left click*) the **System/Event Sources/Proxy/BlueCoat** folder in the `Explorer` tab then *right click* to bring up the `New Item` selection sub-menu. As we are creating a Text Converter, select the



item to have the `New Text Converter` configuration window.

Enter **BlueCoat-Proxy-V1.0-EVENTS** into the `Name:` entry box and press the



which results in the creation of the Text Converter tab

Item Tools Monitoring User Help

Explorer

+ Quick Filter

System

- Event Sources
 - Proxy
 - BlueCoat
 - BlueCoat-Proxy-V1.0-EVENTS

T BlueCoat-Proxy-V1.0-EVENTS

We set the configuration for this Text Converter to be

[Stroom UI Create Feed - New TextConverter tab](#)

- Description - Simple XML transform for BlueCoat Proxy log data sent in W2C Extended Log File Format (ELFF)
- Converter Type - We set to Data Splitter as we will be using the Stroom Data Splitter facility to convert the raw log data into simple XML.

Again, press the Save icon



to save the configuration items.

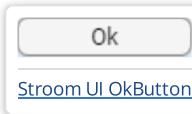
4.6.2.4.3 Create the XSLT Translation

We now create the XSLT translation for this Feed in a similar fashion to the Event Feed or Text Converter. We first select (with a left click) the **System/Event Sources/Proxy/BlueCoat** folder in the Explorer tab then right click to bring up the New Item selection sub-menu. As we are creating a XSLT Translation, select the



item to have the **New XSLT** configuration window.

Enter **BlueCoat-Proxy-V1.0-EVENTS** into the Name: entry box and press the



which results in the creation of the XSLT Translation tab

Item Tools Monitoring User Help

Explorer

+ Quick Filter

System

- Event Sources
 - Proxy
 - BlueCoat
 - BlueCoat-Proxy-V1.0-EVENTS
 - T BlueCoat-Proxy-V1.0-EVENTS
 - xsl BlueCoat-Proxy-V1.0-EVENTS

T BlueCoat-Proxy-V1.0-EVENTS

xsl BlueCoat-Proxy-V1.0-EVENTS

[Stroom UI Create Feed - New Translation tab](#)

and its corresponding reference in the Explorer display.

We set the configuration for this XSLT Translation to be

- Description - Transform simple XML of BlueCoat Proxy log data into Stroom Event Logging XML form

Again, press the Save icon



to save the configuration items.

4.6.2.4.4 Create the Pipeline

We now create the Pipeline for this Feed in a similar fashion to the Event Feed , Text Converter or XSLT Translation . We first select (with a *left click*) the **System/Event Sources/Proxy/BlueCoat** folder in the Explorer tab then *right click* to bring up the New Item selection sub-menu. As we are creating a Pipeline, select the



item to have the **New Pipeline** configuration window.

Enter **BlueCoat-Proxy-V1.0-EVENTS** into the **Name:** entry box and press the



which results in the creation of the Pipeline tab



and it's corresponding reference in the **Explorer** display.

We set the configuration for this **Pipeline** to be

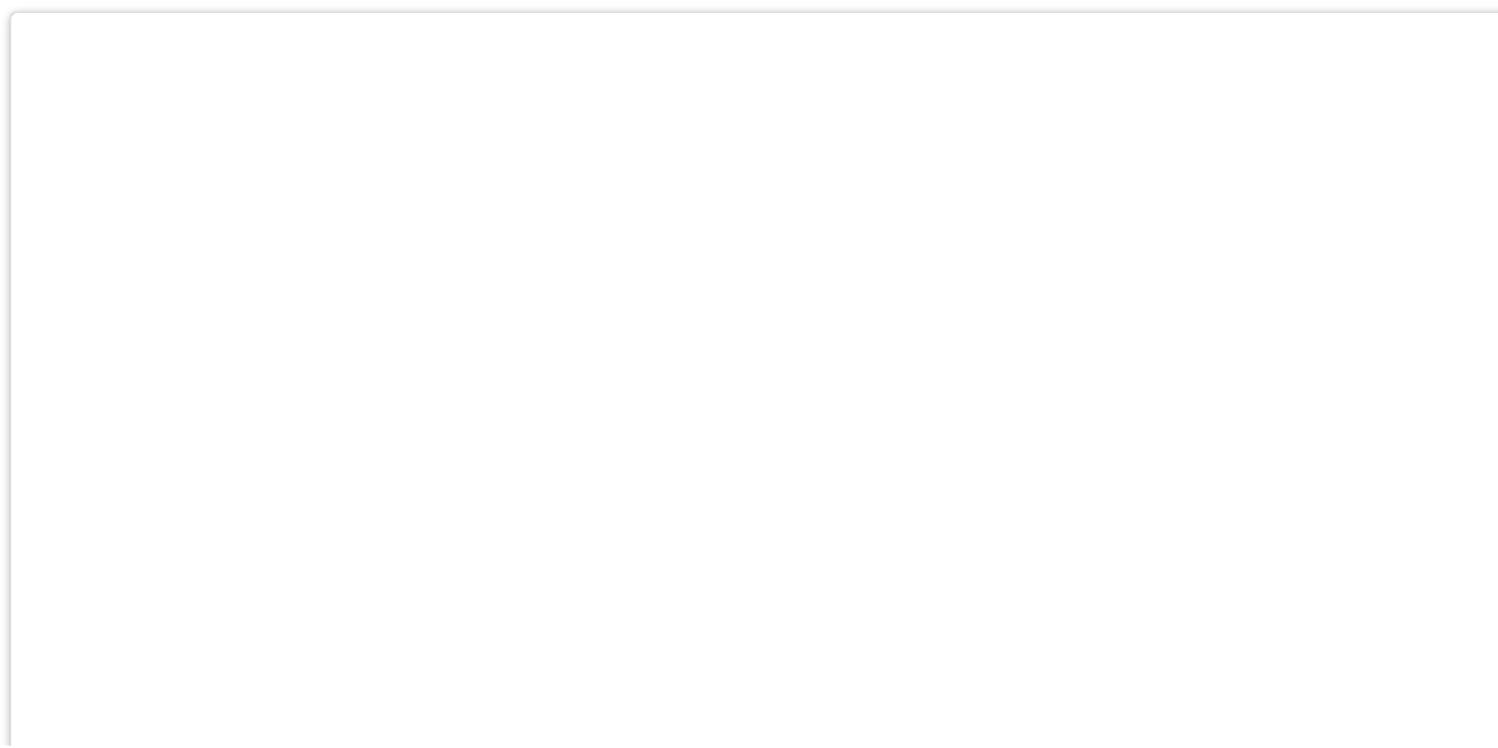
- Description - *Processing of XML of BlueCoat Proxy log data into Stroom Event Logging XML*
- Type - We leave as *Event Data* as this is an *Event Data* pipeline

4.6.2.4.4.1 Configure Pipeline Structure

We now need to configure the *Structure* of this **Pipeline** .

We do this by selecting the *structure* hyper-link of the **BlueCoat-Proxy-V1.0-EVENTS* Pipeline tab.

At this we see the **Pipeline Structure** configuration tab



The screenshot shows the Stroom UI interface for creating a feed. At the top, there are tabs for 'Welcome', 'BlueCoat-Proxy-V1.0-EVENTS', 'Structure' (which is selected), 'Processors', and 'Active Tasks'. Below the tabs, there are buttons for 'Inherit From' (set to 'None') and 'Source'. On the right, there is a 'View Source' link.

The main area contains two tables:

- Properties Table:** Headers: Property Name, Value, Source, Inherited Value, Inherited From, Default Value, Description. There are no rows in this table.
- Pipelines Table:** Headers: Pipeline, Feed, Stream Type, Inherited From. There are no rows in this table.

At the bottom of the interface, there are navigation links: '1 to 1 of ?' followed by icons for back, forward, and search.

[Stroom UI Create Feed - Pipeline Structure tab](#)

As noted in the Assumptions at the start, we have loaded the **Template Pipeline** content pack, so that we can *Inherit* a pipeline structure from this content pack and configure it to support this specific feed.

We find a template by selecting the **Inherit From:** entry box labeled

A dropdown menu with the following options:

- None
- ...

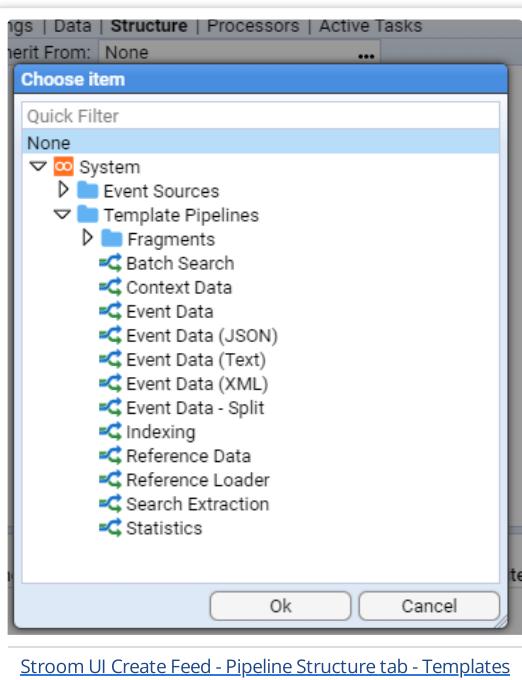
Below the dropdown, the text [Stroom UI NoneEntryBox](#) is displayed.

to reveal a **Choose Item** configuration item window.



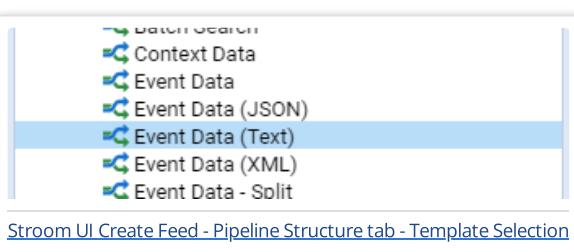
[Stroom UI Open Folder](#)

icon to the left of the folder to reveal the choice of available templates.



[Stroom UI Create Feed - Pipeline Structure tab - Templates](#)

For our BlueCoat feed we will select the `Event Data (Text)` template. This is done by moving the cursor to the relevant line and select via a *left click*



[Stroom UI Create Feed - Pipeline Structure tab - Template Selection](#)

then pressing



[Stroom UI OkButton](#)

to see the inherited pipeline structure

Stroom UI Create Feed - Pipeline Structure tab - Template Selected

4.6.2.4.4.2 Configure Pipeline Elements

For the purpose of this HOWTO, we are only interested in two of the eleven (11) elements in this pipeline

- the Text Converter labeled *dsParser*
- the XSLT Translation labeled *translationFilter*

We need to assign our BlueCoat-Proxy-V1.0-EVENTS Text Converter and XSLT Translation to these elements respectively.

4.6.2.4.4.2.1 Text Converter Configuration

We do this by first selecting (*left click*) the *dsParser* element at which we see the *Property* sub-window displayed

Property Name	Value	Source	Inherited Value	Inherited From	Default Value	Description
textConverter	Inherit					The data splitter configuration that should be used to parse the input data.

Stroom UI Create Feed - Pipeline Structure tab - dsParser

We then select (*left click*) the *textConverter* **Property Name**

Property Name	Value	Source	Inherited Value	Inherited From	Default Value	Description
textConverter	Inherit					The data splitter configuration that should be used to parse the input data.

Stroom UI Create Feed - Pipeline Structure tab - dsParser selected Property

then press the **Edit** button



Stroom UI EditButton

. At this, the **Edit Property** configuration window is displayed.

Property Name	Value	Source
textConverter	Inherit	

Edit Property

Element Id: dsParser

Name: textConverter

Source: Inherit

Value: None

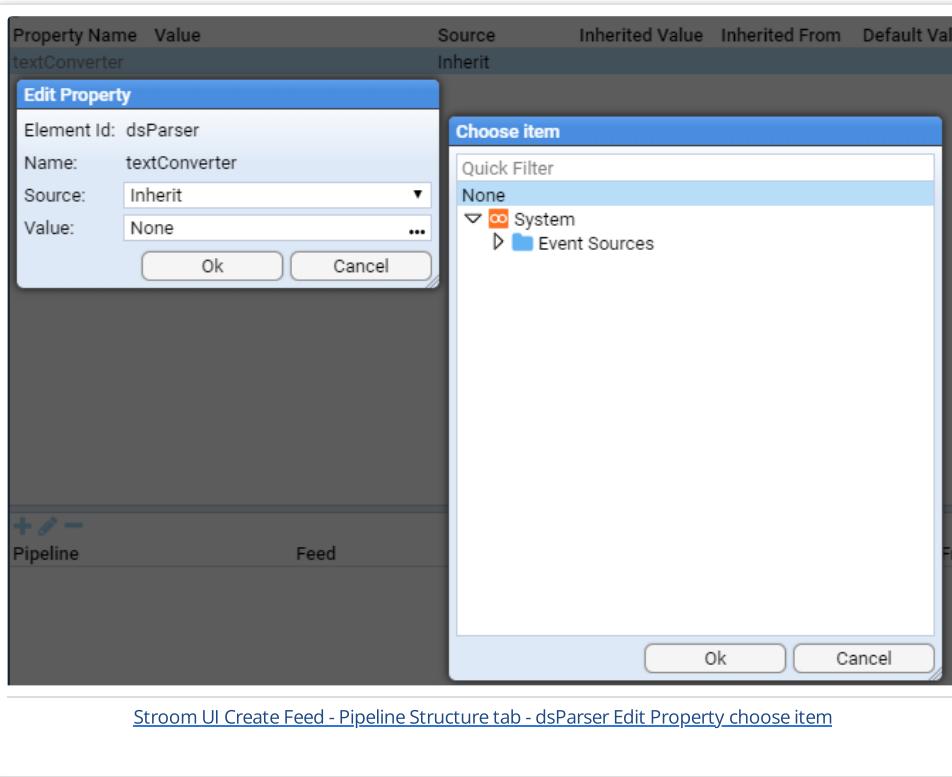
Ok Cancel

Stroom UI Create Feed - Pipeline Structure tab - dsParser Edit Property

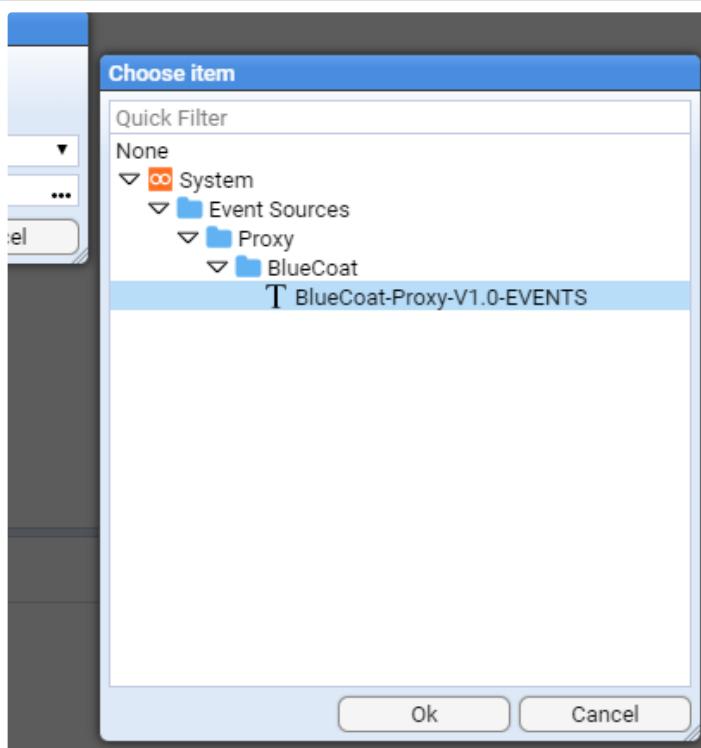
We select the **Value:** entry box labeled



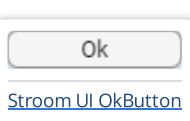
to reveal a **Choose Item** configuration item window.



We traverse the folder structure until we can select the **BlueCoat-Proxy-V1.0-EVENTS** Text Converter as per



and then press the

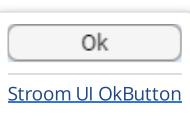


to see that the Property **Value:** has been selected.



[Stroom UI Create Feed - Pipeline Structure tab - dsParser set Property chosen item](#)

and pressing the



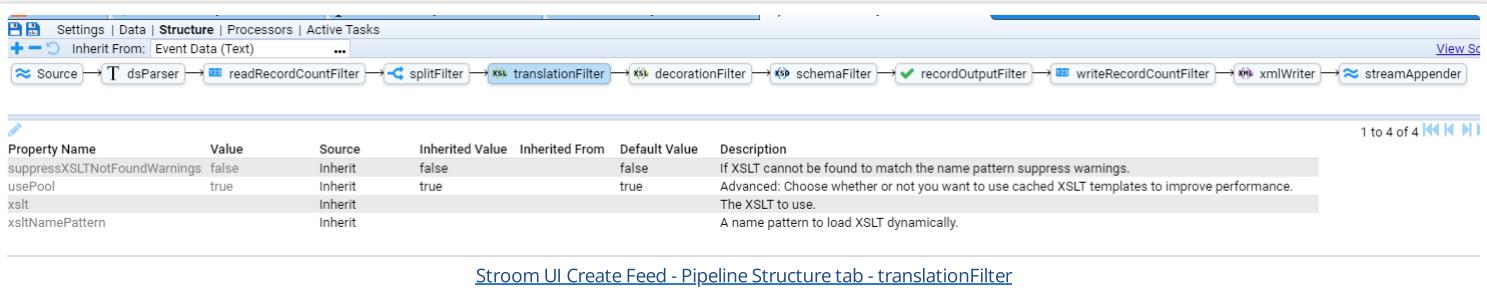
button of the **Edit Property** configuration window results in the pipelines **dsParser** property being set.



[Stroom UI Create Feed - Pipeline Structure tab - dsParser set Property](#)

4.6.2.4.4.2.2 XSLT Translation Configuration

We do this by first selecting (*left click*) the *translationFilter* element at which we see the *Property* sub-window displayed



[Stroom UI Create Feed - Pipeline Structure tab - translationFilter](#)

We then select (*left click*) the **xslt** **Property Name**



[Stroom UI Create Feed - Pipeline Structure tab - xslt selected Property](#)

and following the same steps as for the Text Converter property selection, we assign the **BlueCoat-Proxy-V1.0-EVENTS** XSLT Translation to the xslt property.

Property Name	Value	Source	Inherited Value	Inherited From	Default Value	Description
suppressXSLTNotFoundWarnings	false	Inherit	false		false	If XSLT cannot be found to match the name
usePool	true	Inherit	true		true	Advanced: Choose whether or not you want
xslt	BlueCoat-Proxy-V1.0-EVENTS	Local				The XSLT to use.
xslnNamePattern		Inherit				A name pattern to load XSLT dynamically.

[Stroom UI Create Feed - Pipeline Structure tab - xslt selected Property](#)

At this point, we save these changes by pressing the Save icon



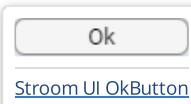
4.6.2.5 Authoring the Translation

We are now ready to author the translation. Close all tabs except for the **Welcome** and **BlueCoat-Proxy-V1.0-EVENTS** Feed tabs.

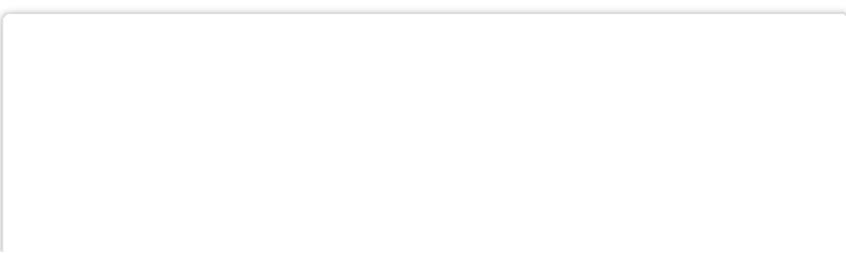
On the **BlueCoat-Proxy-V1.0-EVENTS** Feed tab, select the **Data** hyper-link to be presented with the **Data** pane of our tab.

[Stroom UI Create Feed - Translation - Data Pane](#)

Although we can post our test data set to this feed, we will manually upload it via the **Data** pane. To do this we press the Upload button



in the top **Data** pane to display the **Upload** configuration window





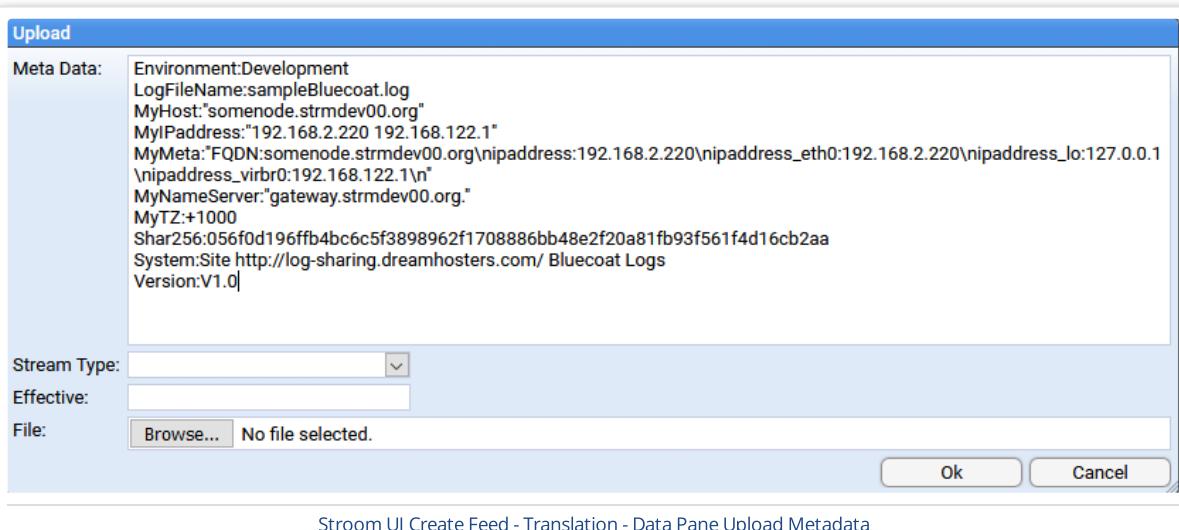
[Stroom UI Create Feed - Translation - Data Pane Upload](#)

In a *Production* situation, where we would post log files to Stroom, we would include certain HTTP Header variables that, as we shall see, will be used as part of the translation. These header variables typically provide situational awareness of the source system sending the events.

For our purposes we set the following HTTP Header variables

```
Environment:Development
LogFile:sampleBluecoat.log
MyHost:"somenode.strmdev00.org"
MyIPaddress:"192.168.2.220 192.168.122.1"
MyMeta:"FQDN:somenode.strmdev00.org\nipaddress:192.168.2.220\nipaddress_eth0:192.168.2.220\nipaddress_lo:127.0.0.1\nipaddress_virbr0:192.168.122.1"
MyNameServer:"gateway.strmdev00.org."
MyTZ:+1000
Shar256:056f0d196ffb4bc6c5f3898962f1708886bb48e2f20a81fb93f561f4d16cb2aa
System:Site http://log-sharing.dreamhosters.com/ Bluecoat Logs
Version:V1.0
```

These are set by entering them into the **Meta Data:** entry box.

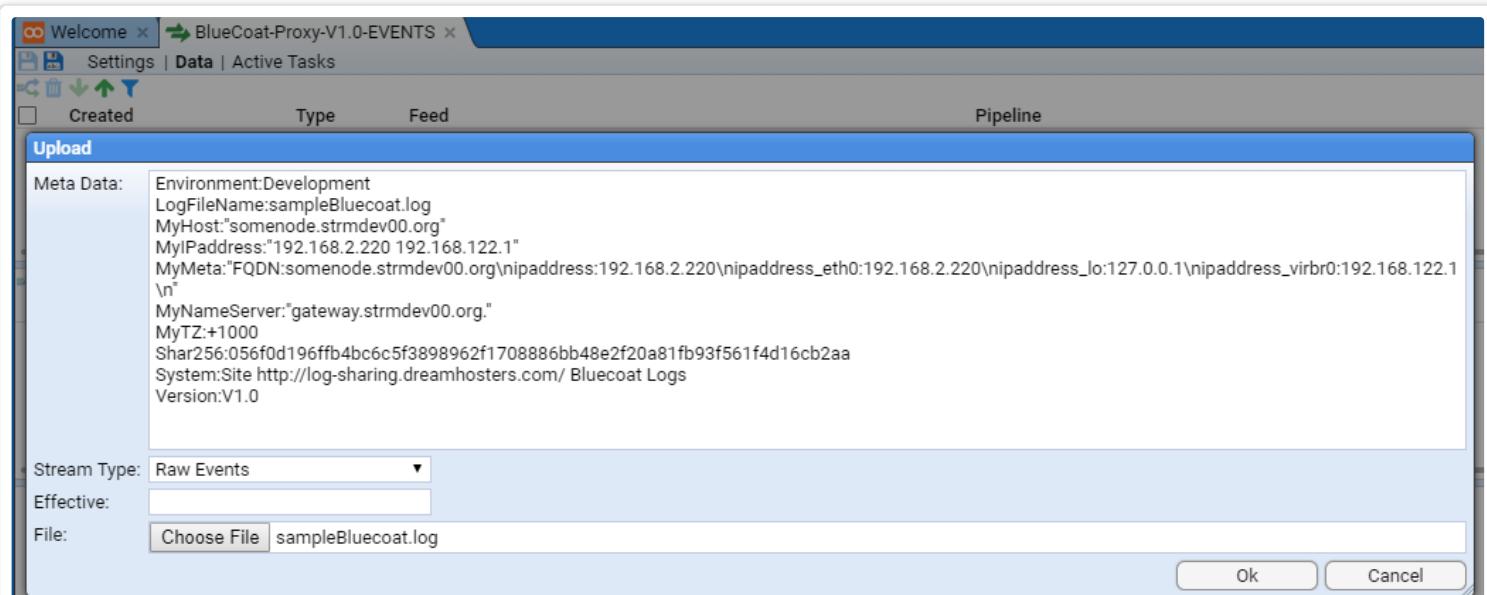


[Stroom UI Create Feed - Translation - Data Pane Upload Metadata](#)

Having done this we select a **Stream Type:** of Raw Events

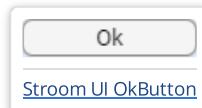
We leave the **Effective:** entry box empty as this stream of raw event logs does not have an Effective Date (only Reference Feeds set this).

And we choose our file sampleBluecoat.log , by clicking on the **Browse** button in the **File:** entry box, which brings up the brower's standard file upload selection window. Having selected our file, we see

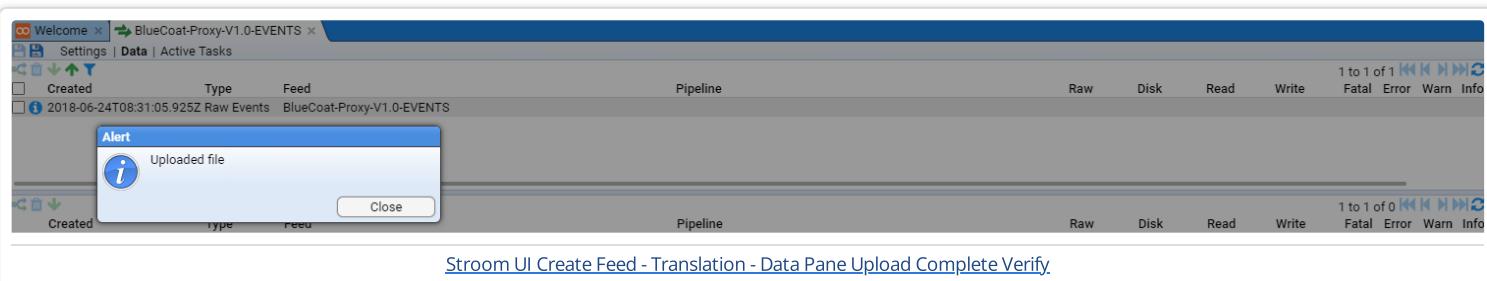


[Stroom UI Create Feed - Translation - Data Pane Upload Complete](#)

On pressing



and **Alert** pop-up window is presented indicating the file was uploaded

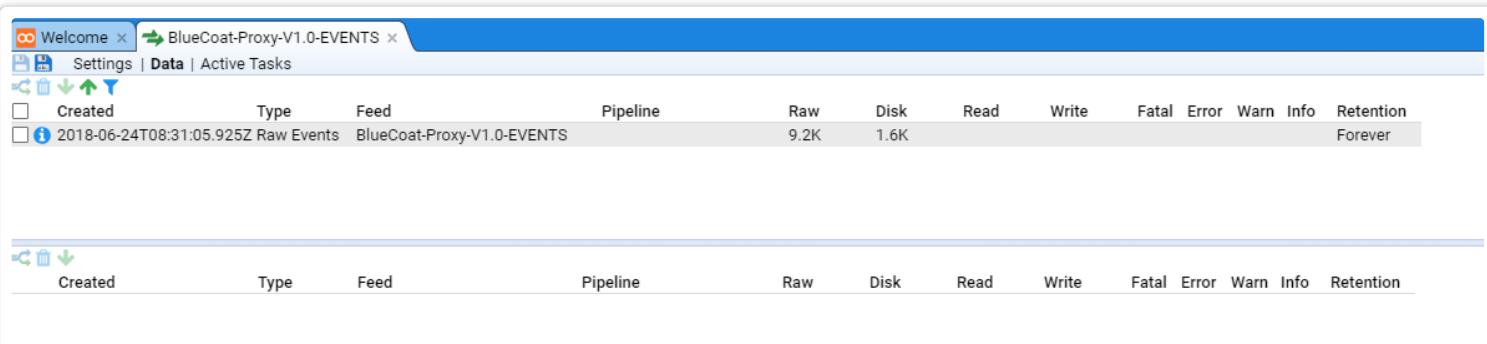


[Stroom UI Create Feed - Translation - Data Pane Upload Complete Verify](#)

Again press



to show that the data has been uploaded as a Stream into the **BlueCoat-Proxy-V1.0-EVENTS** Event Feed.



The top pane holds a table of the latest streams that pertain to the feed. We see the one item which is the stream we uploaded. If we select it, we see that a stream summary is also displayed in the centre pane (which shows details of the *specific* selected feed and associated streams. We also see that the bottom pane displays the data associated with the selected item. In this case, the first lines of content from the BlueCoat sample log file.

If we were to select the **Meta** hyper-link of the lower pane, one would see the metadata Stroom records for this Stream of data.

Welcome x BlueCoat-Proxy-V1.0-EVENTS x

Settings | Data | Active Tasks

Created Type Feed Pipeline Raw Disk Read Write Fatal Error Warn Info Retention

2018-06-24T08:31:05.925Z Raw Events BlueCoat-Proxy-V1.0-EVENTS 9.2K 1.6K Forever

Created Type Feed Pipeline Raw Disk Read Write Fatal Error Warn Info Retention

> 2018-06-24T08:31:05.925Z Raw Events BlueCoat-Proxy-V1.0-EVENTS 9.2K 1.6K Forever

Data | Meta

```
1 Environment:Development
2 Feed:BlueCoat-Proxy-V1.0-EVENTS
3 LogFileName:sampleBluecoat.log
4 MyHost:"somenode.strmdev00.org"
5 MyIPAddress:"192.168.2.220 192.168.122.1"
6 MyMeta:"FQDN:somenode.strmdev00.org\nipaddress:192.168.2.220\nipaddress_eth0:192.168.2.220\nipaddress_lo:127.0.0.1\nipaddress_virbr0:192.168.122.1\n"
7 MyNameServer:"gateway.strmdev00.org."
8 MyTZ:+1000
9 ReceivedTime:2018-06-24T08:31:05.850Z
10 RemoteFile:c:\fakepath\sampleBluecoat.log
11 Shar256:056fd196ffb4bc6c5f3898962f1708886bb48e2f20a81fb93f561f4d16cb2aa
12 StreamSize:9423
13 System:Site http://log-sharing.dreamhosters.com/ Bluecoat Logs
14 user-agent:STROOM-UI
15 Version:V1.0
16
```

You should see all the HTTP variables we set as part of the Upload step as well as some that Stroom has automatically set.

We now switch back to the **Data** hyper-link before we start to develop the actual translation.

4.6.2.5.1 Stepping the Pipeline

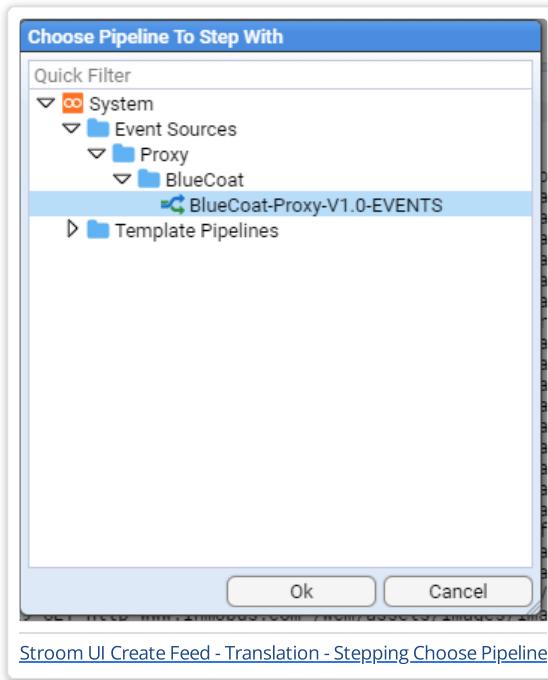
We will now author the two translation components of the pipeline, the data splitter that will transform our lines of BlueCoat data into a simple xml format and then the XSLT translation that will take this simple xml format and translate it into appropriate Stroom Event Logging XML form.

We start by ensuring our `Raw Events` Data stream is selected and we press the `Enter Stepping Mode`

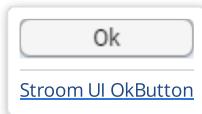


button on the lower right hand side of the bottom `Stream Data` pane.

You will be prompted to select a pipeline to step with. Choose the `BlueCoat-Proxy-V1.0-EVENTS` pipeline



then press



4.6.2.5.2 Stepping the Pipeline - Source

You will be presented with the `Source` element of the pipeline that shows our selected stream's raw data.

The bottom part displays the first page (up to 100 lines) of data along with a set of blue Data Selection Actions. The Data Selection Actions are used to step through the source data 100 lines at a time. When multiple files have been aggregated into a single stream, two Data Selection Actions control buttons will be offered. The right hand one will allow a user to step though the source data as before, but the left hand set of control buttons allows one to step between files from the aggregated event log files.

4.6.2.5.3 Stepping the Pipeline - dsParser

We now select the `dsParser` pipeline element that results in the window below.

The screenshot shows the Stroom UI interface for creating a feed. The top navigation bar has tabs for 'Welcome', 'BlueCoat-Proxy-V1.0-EVENTS', and 'BlueCoat-Proxy-V1.0-EVENTS'. Below the tabs is a toolbar with icons for back, forward, and search. A horizontal flowchart at the top represents the data processing pipeline:

```
graph LR; Source((Source)) --> dsParser[dsParser]; dsParser --> translationFilter[translationFilter]; translationFilter --> decorationFilter[decorationFilter]; decorationFilter --> schemaFilter[schemaFilter]; schemaFilter --> xmlWriter[xmlWriter]; xmlWriter --> streamAppender[streamAppender]
```

The main workspace is currently empty, indicated by the number '1' in the top-left corner. At the bottom of the screen, there is a status bar with the text 'UNKNOWN CLASSIFICATION'.

This window is made up of four panes.

The top pane remains the same - a display of the pipeline structure and the *step indicator* and green *Stepping Actions*.

The next pane down is the editing pane for the Text Converter. This pane is used to edit the text converter that converts our *line based* BlueCoat Proxy logs into a XML format. We make use of the Stroom Data Splitter facility to perform this transformation. See [here](#) for complete details on the data splitter.

The lower two panes are the *input* and *output* displays for the text converter.

The authoring of this data splitter translation is outside the scope of this HOWTO. It is recommended that one reads up on the [Data Splitter](#) and review the various samples found in the Stroom Context packs published, or the Pull Requests of github.com/gchq/stroom-content ([external link](#)).

For the purpose of this HOWTO, the Datasplitter appears below. The author believes the comments should support the understanding of the transformation.

```
<?xml version="1.0" encoding="UTF-8"?>
<dataSplitter bufferSize="5000000" xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocat
<!--
This datasplitter gains the Software and and Proxy version strings along with the log field names from the comments section of
That is from the lines ...
#Software: SGOS 3.2.4.28
#Version: 1.0
#Date: 2005-04-27 20:57:09
#Fields: date time time-taken c-ip sc-status s-action sc-bytes cs-bytes cs-method ... x-icap-error-code x-icap-error-details

We use the Field values as the header for the subsequent log fields
-->

<!-- Match the software comment line and save it in _bc_software -->
<regex id="software" pattern="^#Software: (.+) ?\n*>
  <data name="_bc_software" value="$1" />
</regex>
<!-- Match the version comment line and save it in _bc_version -->

<regex id="version" pattern="^#Version: (.+) ?\n*>
  <data name="_bc_version" value="$1" />
</regex>

<!-- Match against a Fields: header comment and save all the field names in a headings -->

<regex id="heading" pattern="^#Fields: (.+) ?\n*>
  <group value="$1">
    <regex pattern="^(\S+) ?\n*>
      <var id="headings" />
    </regex>
  </group>
</regex>

<!-- Skip all other comment lines -->
<regex pattern="^#.+\n*>
  <var id="ignorea" />
</regex>

<!-- We now match all other lines, applying the headings captured at the start of the file to each field value -->

<regex id="body" pattern="^[^#].+\n*>
  <group>
    <regex pattern="^\"([^\"]*)\" ?\n*>
      <data name="$headings$1" value="$1" />
    </regex>
    <regex pattern="^([^\n]+) *\n*>
      <data name="$headings$1" value="$1" />
    </regex>
  </group>
</regex>

<!-- -->
</dataSplitter>
```

It should be entered into the Text Converter's editing pane as per

Source → T dsParser → translationFilter → decorationFilter → schemaFilter → xmlWriter → streamAppender

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <dataSplitter bufferSize="5000000" xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3 file:///data-spli
3
4 <!--
5 This datasplitter gains the Software and and Proxy version strings along with the log field names from the comments section of the log file.
6 That is from the lines ...
7
8 #Software: SGOS 3.2.4.28
9 #Version: 1.0
10 #Date: 2005-04-27 20:57:09
11 #Fields: date time time-taken c-ip sc-status s-action sc-bytes cs-bytes cs-method ... x-icap-error-code x-icap-error-details
12
13 We use the Field values as the header for the subsequent log fields
-->
14
15 <!-- Match the software comment line and save it in _bc_software -->
16 <regex id="software" pattern="#Software: (.+) ?\n*>
17 | <data name="_bc_software" value="$1" />
18 </regex>
19 <!-- Match the version comment line and save it in _bc_version -->
20 <regex id="version" pattern="#Version: (.+) ?\n*>
21 | <data name="_bc_version" value="$1" />
22 </regex>
23
24 <!-- Match against a Fields: header comment and save all the field names in a headings -->
25
26 <regex id="heading" pattern="#Fields: (.+) ?\n*>
27 | <group value="$1">
28 |   <regex pattern="^(\S+) ?\n*>
29 |     <var id="headings" />
30 |   </regex>
31 | </group>
32 </regex>
33
34 <!-- Skip all other comment lines -->
35 <regex pattern="#.*\n*>
36 | <var id="ignorearea" />
37 </regex>
38
39 <!-- We now match all other lines, applying the headings captured at the start of the file to each field value -->
40
41 <regex id="body" pattern="^#[^\n]*>
42 | <group>
43 |   <regex pattern="^\"([^\"]*)\"; ?\n*>
44 |     <data name="$headings$1" value="$1" />
45 |   </regex>
46 |   <regex pattern="^([^\n]+) *\n*>
47 |     <data name="$headings$1" value="$1" />
48 |   </regex>
49 | </group>
50 </regex>
51
52

```

[Stroom UI Create Feed - Translation - Stepping dsParser textConverter code](#)

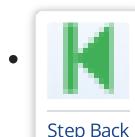
A copy of this DataSplitter can be found [here](#).

As mentioned earlier, to step the translation, one uses the green *Stepping Actions*.

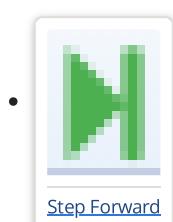
The actions are



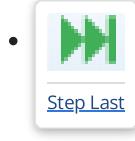
- progress the transformation to the first line of the translation input



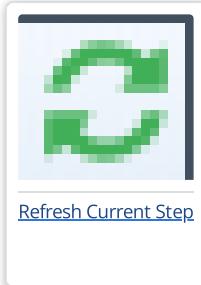
- progress the transformation one step backward



- progress the transformation one step forward



- progress the transformation to the end of the translation input



- refresh the transformation based on the current translation input

So, if one was to press the



stepping action we would be presented with

The screenshot shows the Stroom UI interface with a configuration for a dsParser step. The top navigation bar includes tabs for Source, dsParser, translationFilter, decorationFilter, schemaFilter, xmlWriter, and streamAppender. The dsParser tab is active.

The configuration code (line numbers 1-38) is as follows:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <dataSplitter bufferSize="500000" xmlns="data-splitter:3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="data-splitter:3 file:///data-splitter-v3.0.xsd" version="3.0" ignore="true">
3   <!--
4     This datasplitter gains the Software and and Proxy version strings along with the log field names from the comments section of the log file.
5     That is from the lines ...
6
7   #Software: SGOS 3.2.4.28
8   #Version: 1.0
9   #Date: 2005-04-27 20:57:09
10  #Fields: date time time-taken c-ip sc-status sc-bytes cs-bytes cs-method ... x-icap-error-code x-icap-error-details
11
12  We use the Field values as the header for the subsequent log fields
13  -->
14
15  <!-- Match the software comment line and save it in _bc_software -->
16  <regex id="software" pattern="#Software: (.+) ?\n*>
17    | <data name="_bc_software" value="$1" />
18  </regex>
19
20  <!-- Match the version comment line and save it in _bc_version -->
21
22  <regex id="version" pattern="#Version: (.+) ?\n*>
23    | <data name="_bc_version" value="$1" />
24  </regex>
25
26  <!-- Match against a Fields: header comment and save all the field names in a headings -->
27
28  <regex id="heading" pattern="#Fields: (.+) ?\n*>
29    <group value="$1">
30      <regex pattern="(\S+) ?\n*>
31        | <var id="headings" />
32      </regex>
33    </group>
34  </regex>
35
36  <!-- Skip all other comment lines -->
37  <regex pattern="#.+?\n*>
38
```

The bottom left pane shows the input log line:

```
#Software: SGOS 3.2.4.28
```

The bottom right pane shows the resulting XML output:

```
<?xml version="1.1" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="records:2 file:///data-splitter-v3.0.xsd" version="3.0" ignore="true">
  <record>
    <data name="_bc_software" value="SGOS 3.2.4.28" />
  </record>
</records>
```

A status bar at the bottom center indicates "UNKNOWN CLASSIFICATION".

We see that the *input* pane has the first line of input from our sample file and the *output* pane has an XML **record** structure where we have defined a **data** element with the *name* attribute of *bc_software* and its *value* attribute of *SGOS 3.2.4.28*. The definition of the **record** structure can be found in the **System/XML Schemas/records** folder.

This is the result of the code in our editor

```
<!-- Match the software comment line and save it in _bc_software -->
<regex id="software" pattern="^#Software: (.+) ?\n*>
  <data name="_bc_software" value="$1" />
</regex>
```

If one presses the



stepping action again, we see that we have moved to the second line of the input file with the resultant output of a **data** element with the *name* attribute of *bc_version* and its *value* attribute of *1.0*.

```

14    -->
15    <!-- Match the software comment line and save it in _bc_software -->
16    <regex id="software" pattern="^#Software: (.+) ?\n*>
17      <data name="_bc_software" value="$1" />
18    </regex>
19    <!-- Match the version comment line and save it in _bc_version -->
20
21    <!-- Match the version comment line and save it in _bc_version -->
22    <regex id="version" pattern="^#Version: (.+) ?\n*>
23      <data name="_bc_version" value="$1" />
24    </regex>
25
26    <!-- Match against a Fields: header comment and save all the field names in a headings -->
27
28    <regex id="heading" pattern="^#Fields: (.+) ?\n*>
29      <group value="$1">
30        <regex pattern="^(\S+) ?\n*>
31          <var id="headings" />
32        </regex>
33      </group>
34    </regex>
35
36    <!-- Skip all other comment lines -->
37    <regex pattern="^#.+\n*>
38
```

#Version: 1.0

<?xml version="1.1" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="r
 <record>
 | <data name="_bc_version" value="1.0" />
</record>
</records>

[Stroom UI Create Feed - Translation - Stepping dsParser textConverter 2](#)

Stepping forward once more causes the translation to ignore the Date comment line, define a Data Splitter \$headings variable from the Fields comment line and transform the first line of actual event data.

The screenshot shows a Java application's internal processing logic. The flow starts with 'Source' and passes through several filters and appenders:

- dsParser**: The first filter.
- translationFilter**: Converts text based on specific rules.
- decorationFilter**: Adds or removes decorations from the text.
- schemaFilter**: Validates the schema of the data.
- xmlWriter**: Writes the data to XML format.
- streamAppender**: Finally outputs the data to a stream.

The code itself is a series of regular expression patterns (regex) defined in a configuration file. It handles various comments and fields, applying specific logic to each. For example, it matches version comments, field names, and body content, and applies these matches to the data as it flows through the filters.

```
</regex>
  | <!-- Match the version comment line and save it in _bc_version -->
  |
<regex id="version" pattern="^#Version: (.+) ?\n*>
  | <data name="_bc_version" value="$1" />
</regex>
<!-- Match against a Fields: header comment and save all the field names in the headings -->
<regex id="heading" pattern="^#Fields: (.+) ?\n*>
  | <group value="$1">
    | <regex pattern="^(\S+)?\n*>
      | | <var id="headings" />
    | </regex>
  | </group>
</regex>
<!-- Skip all other comment lines -->
<regex pattern="#.+?\n*>
  | <var id="ignorea" />
</regex>
<!-- We now match all other lines, applying the headings captured at the start of the file to each field value -->
<regex id="body" pattern="^#[^\n*>
  | <group>
    | <regex pattern="^#[^#;]*;[^#;]*;? \n*>
      | | <data name="headings$1" value="$1" />
    | </regex>
    | <regex pattern="^([^\n]+) *\n*>
      | | <data name="$headings$1" value="$1" />
    | </regex>
  | </group>
</regex>
<!-- --->
</dataSplitter>
```

```
'[ ' streamId ':' subStreamId ':' recordNo ']
```

```
<data name="time-taken" value="10.10.1.1" />
<data name="c-ip" value="45.110.2.82" />
<data name="sc-status" value="200" />
<data name="s-ip" value="10.10.1.1" />
<data name="cs-bytes" value="941" />
<data name="cs-bytes" value="729" />
<data name="cs-method" value="GET" />
<data name="cs-uri-scheme" value="http" />
<data name="cs-host" value="www.inmobi.com" />
<data name="cs-uri-path" value="/wcm/assets/images/imagefileicon.gif" />
<data name="cs-uri-query" value="-" />
<data name="cs-username" value="george" />
<data name="s-hierarchy" value="DIRECT" />
```

- **recordNo** - is the record number within the sub stream.

One can double click on either the **subStreamId** or **recordNo** entry and enter a new value. This allows you to *jump* around a stream rather than just relying on first, previous, next and last movements.

Hovering the mouse over the *stepping indicator* will change the cursor to a hand pointer. Selecting (by a left click) the recordNo will allow you to edit it's value (and the other values for that matter). You will see the display change from



Stroom UI Create Feed - Translation - Stepping Indicator

to



[Stroom UI Create Feed](#) - [Translation](#) - [Stepping Indicator](#)

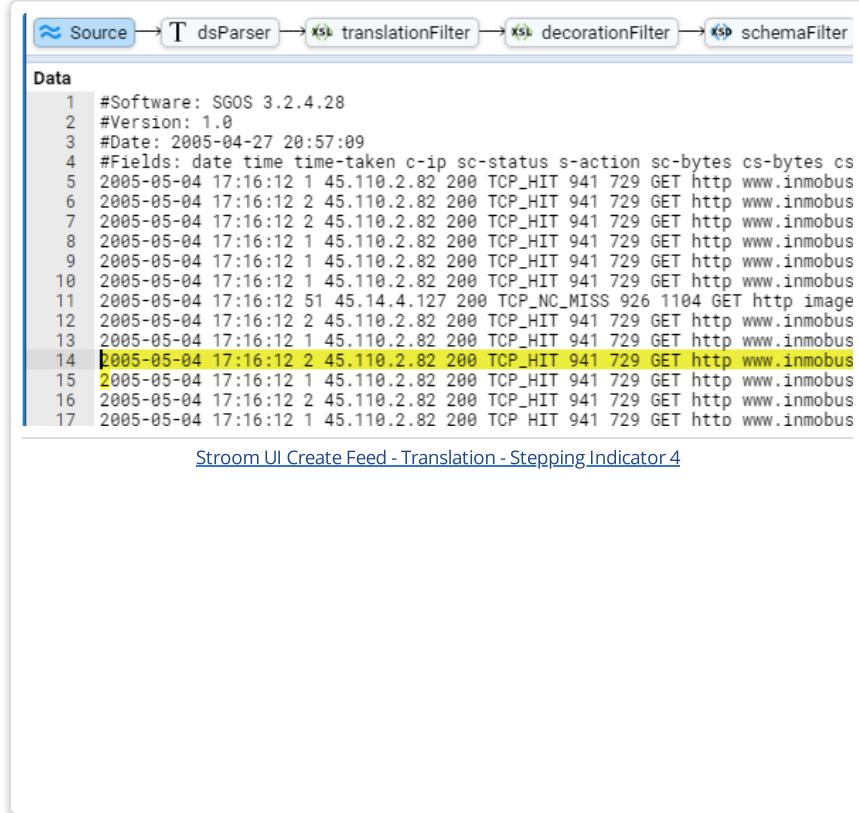
If we change the record number from **3** to **12** then either press Enter or press the



action we see



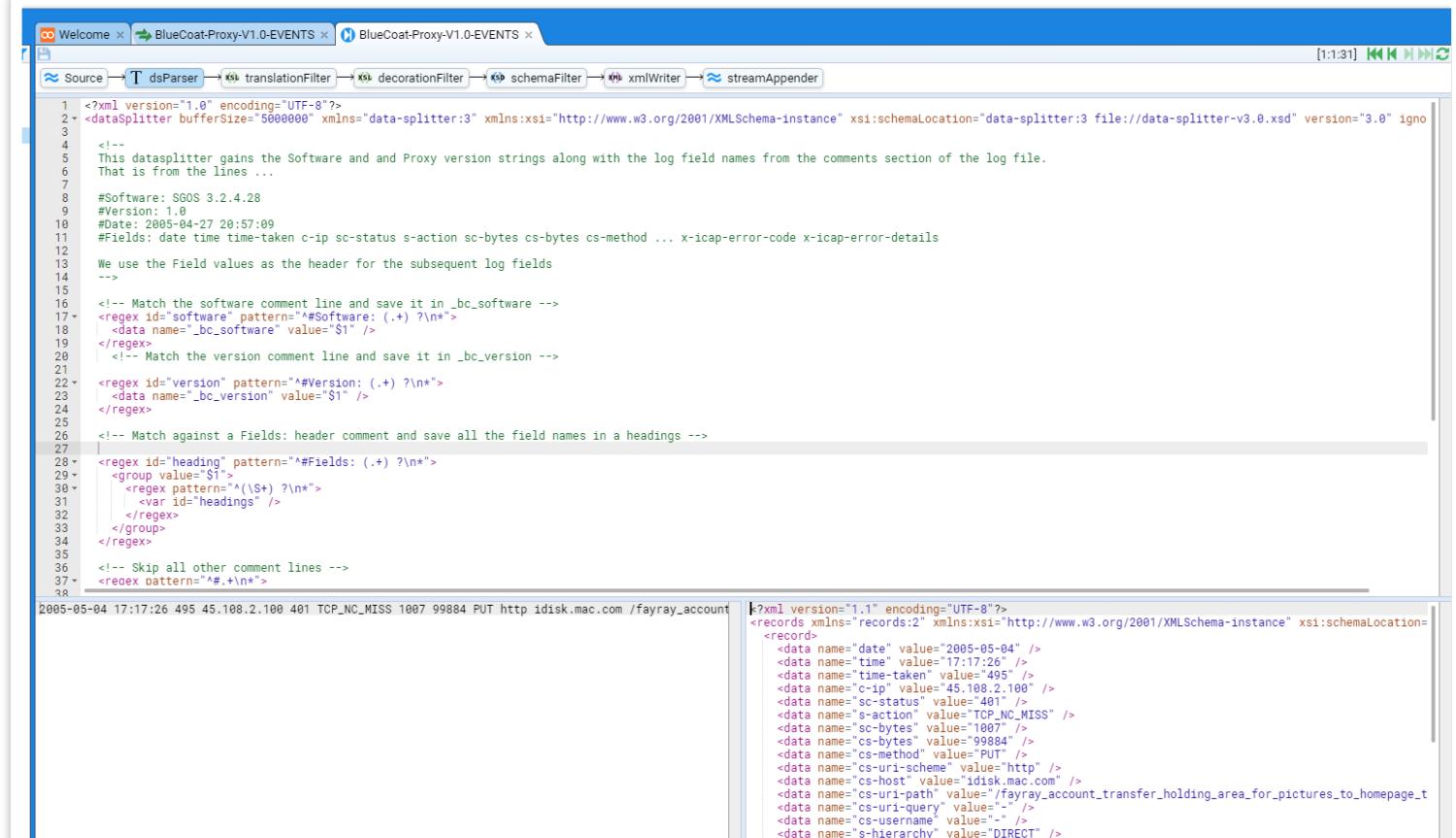
and note that a new record has been processed in the *input* and *output* panes. Further, if one steps back to the `Source` element of the pipeline to view the raw source file, we see that the highlighted **current** line is the 12th line of processed data. It is the 10th actual bluecoat event, but remember the `#Software`, `#Version` lines are considered as processed data ($2+10 = 12$). Also noted that the `#Date` and `#Fields` lines are not considered processed data, and hence do not contribute to the **recordNo** value.



If we select the `dsParser` pipeline element then press the



action we see the `recordNo` jump to 31 which is the last processed line of our sample log file.



Stroom || Create Feed - Translation - Stepping Indicator 5

4.6.2.5.4 Stepping the Pipeline - translationFilter

We now select the `translationFilter` pipeline element that results in

The screenshot shows the Stroom UI interface with the following components:

- Pipeline Structure:** At the top, a horizontal bar shows the pipeline flow: Source → dsParser → **translationFilter** → decorationFilter → schemaFilter → xmlWriter → streamAppender.
- Step Indicator:** A green progress bar at the top indicates the current step is the `translationFilter`.
- Input XML:** The left pane displays the input XML structure, which is a simple key-value pair (`<records>`) in XML Schema format.
- Output XML:** The right pane displays the output XML structure, which is identical to the input XML, indicating a direct copy.
- Central Area:** The main area contains the URL [Stroom UI Create Feed - Translation - Stepping translationFilter Element](#).

As for the `dsParser`, this window is made up of four panes.

The top pane remains the same - a display of the pipeline structure and the *step indicator* and green *Stepping Actions*.

The next pane down is the editing pane for the Translation Filter. This pane is used to edit an `xslt` translation that converts our simple key value pair `<records>` XML structure into another XML form.

The lower two panes are the *input* and *output* displays for the `xslt` translation. You will note that the *input* and *output* displays are identical for a null `xslt` translation is effectively a direct copy.

In this HOWTO we will transform the `<records>` XML structure into the *GCHQ Stroom Event Logging XML Schema* form which is documented [here \(external link\)](#).

The authoring of this `xslt` translation is outside the scope of this HOWTO, as is the use of the Stroom XML Schema. It is recommended that one reads up on [XSLT Conversion](#) and the [Stroom Event Logging XML Schema \(external link\)](#) and review the various samples found in the Stroom Context packs published, or the Pull Requests of [https://github.com/gchq/stroom-content \(external link\)](https://github.com/gchq/stroom-content (external link)).

We will build the translation in steps. We enter an initial portion of our `xslt` transformation that just consumes the `Software` and `Version` key values and converts the `date` and `time` values (which are in UTC) into the `EventTime/TimeCreated` element. This code segment is

```

<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xpath-default-namespace="records:2" xmlns="event-logging:3" xmlns:stroom="stroom"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs=
```

<!-- Bluecoat Proxy logs in W2C Extended Log File Format (ELF) -->

<!-- Ingest the record key value pair elements -->

```

<xsl:template match="records">
    <Events xsi:schemaLocation="event-logging:3 file://event-logging-v3.2.4.xsd" Version="3.2.4">
        <xsl:apply-templates />
    </Events>
</xsl:template>
```

<!-- Main record template for single event -->

```

<xsl:template match="record">
    <xsl:choose>

        <!-- Store the Software and Version information of the Bluecoat log file for use in the Event Source elements which are pr
        <xsl:when test="data[@name='_bc_software']">
            <xsl:value-of select="stroom:put('_bc_software', data[@name='_bc_software']/@value)" />
        </xsl:when>
        <xsl:when test="data[@name='_bc_version']">
            <xsl:value-of select="stroom:put('_bc_version', data[@name='_bc_version']/@value)" />
        </xsl:when>

        <!-- Process the event logs -->
        <xsl:otherwise>
            <Event>
                <xsl:call-template name="event_time" />
            </Event>
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>
```

<!-- Time -->

```

<xsl:template name="event_time">
    <EventTime>
        <TimeCreated>
            <xsl:value-of select="concat(data[@name = 'date']/@value, 'T', data[@name='time']/@value, '.000Z')" />
        </TimeCreated>
    </EventTime>
</xsl:template>
</xsl:stylesheet>
```

After entering this translation and pressing the



[Refresh Current Step](#)

action shows the display

BlueCoat-Proxy-V1.0-EVENTS * BlueCoat-Proxy-V1.0-EVENTS [1:1:31]

Source → dsParser → translationFilter → decorationFilter → schemaFilter → xmlWriter → streamAppender

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <xsl:stylesheet xpath-default-namespace="records:2" xmlns="event-logging:3" xmlns:stroom="stroom"
3   xmlns:xsi="http://www.w3.org/1999/XSL/Transform" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs="http://www.w3.org/2001/XMLSchema" version="3.0">
4
5   <!-- Bluecoat Proxy logs in N2C Extended Log File Format (ELLF) -->
6
7   <!-- Ingest the record key value pair elements -->
8   <xsl:template match="records">
9     <Events xsi:schemaLocation="event-logging:3 file://event-logging-v3.2.4.xsd Version="3.2.4">
10       <xsl:apply-templates />
11     </Events>
12   </xsl:template>
13
14   <!-- Main record template for single event -->
15   <xsl:template match="record">
16     <xsl:choose>
17
18       <!-- Store the Software and Version information of the Bluecoat log file for use in the Event Source elements which are processed later -->
19       <xsl:when test="data[@name='bc_software']">
20         <xsl:value-of select="stroom:put('bc_software', data[@name='bc_software']/@value)" />
21       </xsl:when>
22       <xsl:when test="data[@name='bc_version']">
23         <xsl:value-of select="stroom:put('bc_version', data[@name='bc_version']/@value)" />
24       </xsl:when>
25
26       <!-- Process the event logs -->
27       <xsl:otherwise>
28         <Event>
29           <xsl:call-template name="event_time" />
30         </Event>
31       </xsl:otherwise>
32     </xsl:choose>
33   </xsl:template>
34
35   <!-- Time -->
36   <xsl:template name="event_time">
37     <EventTime>
38       <TimeCreated>
39         <xsl:value-of select="concat(data[@name = 'date']/@value, 'T', data[@name='time']/@value, '.000Z)" />
40       </TimeCreated>
41     </EventTime>
42   </xsl:template>
43 </xsl:stylesheet>

```

UNKNOWN CLASSIFICATION

[Stroom UI Create Feed - Translation - Stepping XSLT Translation 1](#)

Note that this is the 31st record, so if we were to jump to the first record using the



action, we see that the *input* and *output* change appropriately.

[Stroom UI Create Feed - Translation - Stepping XSLT Translation 2](#)

```

<?xml version="1.1" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocations=">
<record>
  <data name="bc_software" value="SGOS 3.2.4.28" />
</record>
</records>

```

```

<?xml version="1.1" encoding="UTF-8"?>
<Events xmlns="event-logging:3" xmlns:stroom="stroom" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="event-logging:3 file://event-logging-v3.2.4.xsd Version="3.2.4">
<Event>
  <TimeCreated>2005-05-04T17:17:26.000Z</TimeCreated>
</Event>
</Events>

```

You will note that there is no `Event` element in the *output* pane as the *record* template in our xslt translation above is only storing the input's key value (`_bc_software`'s value).

Further note that the **BlueCoat_Proxy-V1.0-EVENTS** tab has a *star* in front of it and also the *Save* icon



is highlighted. This indicates that a component of the pipeline needs to be saved. In this case, the XSLT translation.

BlueCoat-Proxy-V1.0-EVENTS * BlueCoat-Proxy-V1.0-EVENTS [1:1:31]

Source → dsParser → translationFilter → decorationFilter → schemaFilter → xmlWriter → streamAppender

```

1 <?xml version="1.0" encoding="UTF-8" ?>

```

[Stroom UI Create Feed - Translation - Stepping XSLT Translation 3](#)

By pressing the *Save* icon, you will save the XSLT translation as it currently stands and both the *star* will be removed from the tab and the *Save* icon will no longer be highlighted.



We next extend our translation by authoring a **event_source** template to form an appropriate Stroom Event Logging `EventSource` element structure. Thus our translation now is

```

<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xpath-default-namespace="records:2" xmlns="event-logging:3" xmlns:stroom="stroom"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs=>

<!-- Bluecoat Proxy logs in W2C Extended Log File Format (ELF) -->

<!-- Ingest the record key value pair elements -->
<xsl:template match="records">
    <Events xsi:schemaLocation="event-logging:3 file://event-logging-v3.2.4.xsd" Version="3.2.4">
        <xsl:apply-templates />
    </Events>
</xsl:template>

<!-- Main record template for single event -->
<xsl:template match="record">
    <xsl:choose>

        <!-- Store the Software and Version information of the Bluecoat log file for use in the Event Source elements which are produced -->
        <xsl:when test="data[@name='_bc_software']">
            <xsl:value-of select="stroom:put('_bc_software', data[@name='_bc_software']/@value)" />
        </xsl:when>
        <xsl:when test="data[@name='_bc_version']">
            <xsl:value-of select="stroom:put('_bc_version', data[@name='_bc_version']/@value)" />
        </xsl:when>

        <!-- Process the event logs -->
        <xsl:otherwise>
            <Event>
                <xsl:call-template name="event_time" />
                <xsl:call-template name="event_source" />
            </Event>
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>

<!-- Time -->
<xsl:template name="event_time">
    <EventTime>
        <TimeCreated>
            <xsl:value-of select="concat(data[@name = 'date']/@value, 'T', data[@name='time']/@value, '.000Z')"/>
        </TimeCreated>
    </EventTime>
</xsl:template>

<!-- Template for event source-->
<xsl:template name="event_source">

    <!--
    We extract some situational awareness information that the posting script includes when posting the event data
    -->
    <xsl:variable name="_mymeta" select="translate(stroom:meta('MyMeta'), '"', '')" />

    <!-- Form the EventSource node -->
    <EventSource>
        <System>
            <Name>
                <xsl:value-of select="stroom:meta('System')"/>
            </Name>
            <Environment>
                <xsl:value-of select="stroom:meta('Environment')"/>
            </Environment>
        </System>
        <Generator>
            <xsl:variable name="gen">
                <xsl:if test="stroom:get('_bc_software')">

```

```

<xsl:value-of select="concat(' Software: ', stroom:get('_bc_software'))" />
</xsl:if>
<xsl:if test="stroom:get('_bc_version')">
    <xsl:value-of select="concat(' Version: ', stroom:get('_bc_version'))" />
</xsl:if>
</xsl:variable>
<xsl:value-of select="concat('Bluecoat', $gen)" />
</Generator>
<xsl:if test="data[@name='s-computername'] or data[@name='s-ip']">
    <Device>
        <xsl:if test="data[@name='s-computername']">
            <Name>
                <xsl:value-of select="data[@name='s-computername']/@value" />
            </Name>
        </xsl:if>
        <xsl:if test="data[@name='s-ip']">
            <IPAddress>
                <xsl:value-of select=" data[@name='s-ip']/@value" />
            </IPAddress>
        </xsl:if>
        <xsl:if test="data[@name='s-sitename']">
            <Data Name="ServiceType" Value="{data[@name='s-sitename']/@value}" />
        </xsl:if>
    </Device>
</xsl:if>

<!-- -->
<Client>
    <xsl:if test="data[@name='c-ip']/@value != '-'">
        <IPAddress>
            <xsl:value-of select="data[@name='c-ip']/@value" />
        </IPAddress>
    </xsl:if>

    <!-- Remote Port Number -->
    <xsl:if test="data[@name='c-port']/@value != '-'">
        <Port>
            <xsl:value-of select="data[@name='c-port']/@value" />
        </Port>
    </xsl:if>
</Client>

<!-- -->
<Server>
    <HostName>
        <xsl:value-of select="data[@name='cs-host']/@value" />
    </HostName>
</Server>

<!-- -->
<xsl:variable name="user">
    <xsl:value-of select="data[@name='cs-user']/@value" />
    <xsl:value-of select="data[@name='cs-username']/@value" />
    <xsl:value-of select="data[@name='cs-userdn']/@value" />
</xsl:variable>
<xsl:if test="$user != '-'">
    <User>
        <Id>
            <xsl:value-of select="$user" />
        </Id>
    </User>
</xsl:if>
<Data Name="MyMeta">
    <xsl:attribute name="Value" select="$mymeta" />
</Data>
</EventSource>

```

```
</xsl:template>
</xsl:stylesheet>
```

Stepping to the 3 record (the first real data record in our sample log) will reveal that our `output` pane has gained an `EventSource` element.

```
</xsl:template>
</xsl:stylesheet>

31      </Event>
32    </xsl:otherwise>
33  </xsl:choose>
34 </xsl:template>
35
36  <!-- Time -->
37 <xsl:template name="event_time">
38   <EventTime>
39     <TimeCreated>
40       <xsl:value-of select="concat(data[@name = 'date']/@value, 'T', data[@name='time']/@value, '.000Z')" />
41     </TimeCreated>
42   </EventTime>
43 </xsl:template>
44
45  <!-- Template for event source-->
46 <xsl:template name="event_source">
47
48  <!--
49  We extract some situational awareness information that the posting script includes when posting the event data
50  -->
51  <xsl:variable name="_mymeta" select="translate(stroom:meta('MyMeta'), '&quot;', '')" />
52
53  <!-- Form the EventSource node -->
54 <EventSource>
55   <System>
56     <Name>
57       <xsl:value-of select="stroom:meta('System')" />
58     </Name>

```

```
<?xml version="1.1" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation=<!--records-->
<record>
<data name="date" value="2005-05-04" />
<data name="time" value="17:16:12" />
<data name="time-taken" value="1" />
<data name="c-ip" value="45.110.2.82" />
<data name="sc-status" value="200" />
<data name="s-action" value="TCP_HIT" />
<data name="sc-bytes" value="941" />
<data name="cs-bytes" value="729" />
<data name="cs-method" value="GET" />
<data name="cs-uri-scheme" value="http" />
<data name="cs-host" value="www.inmabus.com" />
<data name="cs-uri-path" value="/wcm/assets/images/imagefileicon.gif" />
<data name="cs-uri-query" value="-" />
<data name="s-username" value="george" />
<data name="s-hierarchy" value="DIRECT" />
<data name="s-supplier-name" value="38.112.92.20" />
<data name="rs(Content-Type)" value="image/gif" />
<data name="cs(User-Agent)" value="Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET<br/><data name="sc-filter-result" value="PROXIED" />
<data name="sc-filter-category" value="none" />
<data name="x-virus-id" value="-" />
<data name="s-ip" value="192.16.170.42" />
<data name="s-sitename" value="SG-HTTP-Service" />
<data name="x-virus-details" value="-" />
<data name="x-icap-error-code" value="none" />
```

```
<Events xmlns="event-logging:3" xmlns:stroom="stroom" xmlns:xsi="http://www.w3.org/2001/XMLSchema-in
<Event>
<EventTime>
<TimeCreated>2005-05-04T17:16:12.000Z</TimeCreated>
</EventTime>
<EventSource>
<System>
<Name>Site http://log-sharing.dreamhosters.com/ Bluecoat Logs</Name>
<Environment>Development</Environment>
</System>
<Generator>Bluecoat Software: SGOS 3.2.4.28 Version: 1.0</Generator>
<Device>
<IPAddress>192.16.170.42</IPAddress>
<Data Name="ServiceType" Value="SG-HTTP-Service" />
</Device>
<Client>
<IPAddress>45.110.2.82</IPAddress>
</Client>
<Server>
<HostName>www.inmabus.com</HostName>
</Server>
<User>
<Id>george</Id>
</User>
<Data Name="MyMeta" Value="FQDN:somenode.strmdev00.org\nipaddress:192.168.2.220\nipaddress_eth0" />
</EventSource>
</Event>
```

Stroom UI Create Feed - Translation - Stepping XSLT Translation 5

Note also, that our **Save** icon



is also highlighted, so we should at some point save the extensions to our translation.

The complete translation now follows. A copy of the XSLT translation can be found [here](#).

```

<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xpath-default-namespace="records:2" xmlns="event-logging:3" xmlns:stroom="stroom"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs=>

<!-- Bluecoat Proxy logs in W2C Extended Log File Format (ELF) -->

<!-- Ingest the record key value pair elements -->
<xsl:template match="records">
    <Events xsi:schemaLocation="event-logging:3 file://event-logging-v3.2.4.xsd" Version="3.2.4">
        <xsl:apply-templates />
    </Events>
</xsl:template>

<!-- Main record template for single event -->
<xsl:template match="record">
    <xsl:choose>

        <!-- Store the Software and Version information of the Bluecoat log file for use in the Event Source elements which are pr
        <xsl:when test="data[@name='_bc_software']">
            <xsl:value-of select="stroom:put('_bc_software', data[@name='_bc_software']/@value)" />
        </xsl:when>
        <xsl:when test="data[@name='_bc_version']">
            <xsl:value-of select="stroom:put('_bc_version', data[@name='_bc_version']/@value)" />
        </xsl:when>

        <!-- Process the event logs -->
        <xsl:otherwise>
            <Event>
                <xsl:call-template name="event_time" />
                <xsl:call-template name="event_source" />
                <xsl:call-template name="event_detail" />
            </Event>
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>

<!-- Time -->
<xsl:template name="event_time">
    <EventTime>
        <TimeCreated>
            <xsl:value-of select="concat(data[@name = 'date']/@value, 'T', data[@name='time']/@value, '.000Z')" />
        </TimeCreated>
    </EventTime>
</xsl:template>

<!-- Template for event source-->
<xsl:template name="event_source">

    <!--
    We extract some situational awareness information that the posting script includes when posting the event data
    -->
    <xsl:variable name="_mymeta" select="translate(stroom:meta('MyMeta'), '"', '')" />

    <!-- Form the EventSource node -->
    <EventSource>
        <System>
            <Name>
                <xsl:value-of select="stroom:meta('System')" />
            </Name>
            <Environment>
                <xsl:value-of select="stroom:meta('Environment')" />
            </Environment>
        </System>
        <Generator>
            <xsl:variable name="gen">

```

```

<xsl:if test="stroom:get('_bc_software')">
  <xsl:value-of select="concat(' Software: ', stroom:get('_bc_software'))" />
</xsl:if>
<xsl:if test="stroom:get('_bc_version')">
  <xsl:value-of select="concat(' Version: ', stroom:get('_bc_version'))" />
</xsl:if>
</xsl:variable>
<xsl:value-of select="concat('Bluecoat', $gen)" />
</Generator>
<xsl:if test="data[@name='s-computername'] or data[@name='s-ip']">
  <Device>
    <xsl:if test="data[@name='s-computername']">
      <Name>
        <xsl:value-of select="data[@name='s-computername']/@value" />
      </Name>
    </xsl:if>
    <xsl:if test="data[@name='s-ip']">
      <IPAddress>
        <xsl:value-of select=" data[@name='s-ip']/@value" />
      </IPAddress>
    </xsl:if>
    <xsl:if test="data[@name='s-sitename']">
      <Data Name="ServiceType" Value="{data[@name='s-sitename']/@value}" />
    </xsl:if>
  </Device>
</xsl:if>

<!-- -->
<Client>
  <xsl:if test="data[@name='c-ip']/@value != '-'">
    <IPAddress>
      <xsl:value-of select="data[@name='c-ip']/@value" />
    </IPAddress>
  </xsl:if>

  <!-- Remote Port Number -->
  <xsl:if test="data[@name='c-port']/@value != '-'">
    <Port>
      <xsl:value-of select="data[@name='c-port']/@value" />
    </Port>
  </xsl:if>
</Client>

<!-- -->
<Server>
  <HostName>
    <xsl:value-of select="data[@name='cs-host']/@value" />
  </HostName>
</Server>

<!-- -->
<xsl:variable name="user">
  <xsl:value-of select="data[@name='cs-user']/@value" />
  <xsl:value-of select="data[@name='cs-username']/@value" />
  <xsl:value-of select="data[@name='cs-userdn']/@value" />
</xsl:variable>
<xsl:if test="$user != '-'">
  <User>
    <Id>
      <xsl:value-of select="$user" />
    </Id>
  </User>
</xsl:if>
<Data Name="MyMeta">
  <xsl:attribute name="Value" select="$_mymeta" />
</Data>

```

```

</EventSource>
</xsl:template>

<!-- Event detail -->
<xsl:template name="event_detail">
<EventDetail>

    <!--
        We model Proxy events as either Receive or Send events depending on the method.

        We make use of the Receive/Send sub-elements Source/Destination to map the Client/Destination Proxy values
        and the Payload sub-element to map the URL and other details of the activity. If we have a query, we model it
        as a Criteria
    -->

    <Type Id>
        <xsl:value-of select="concat('Bluecoat-', data[@name='cs-method']/@value, '-', data[@name='cs-uri-scheme']/@value)" />
        <xsl:if test="data[@name='cs-uri-query']/@value != '-'>-Query</xsl:if>
    </Type Id>
    <xsl:choose>
        <xsl:when test="matches(data[@name='cs-method']/@value, 'GET|OPTIONS|HEAD')">
            <Description>Receipt of information from a Resource via Proxy</Description>
            <Receive>
                <xsl:call-template name="setupParticipants" />
                <xsl:call-template name="setPayload" />
                <xsl:call-template name="setOutcome" />
            </Receive>
        </xsl:when>
        <xsl:otherwise>
            <Description>Transmission of information to a Resource via Proxy</Description>
            <Send>
                <xsl:call-template name="setupParticipants" />
                <xsl:call-template name="setPayload" />
                <xsl:call-template name="setOutcome" />
            </Send>
        </xsl:otherwise>
    </xsl:choose>
</EventDetail>
</xsl:template>

<!-- Establish the Source and Destination nodes -->
<xsl:template name="setupParticipants">
    <Source>
        <Device>
            <xsl:if test="data[@name='c-ip']/@value != '-'>
                <IPAddress>
                    <xsl:value-of select="data[@name='c-ip']/@value" />
                </IPAddress>
            </xsl:if>

            <!-- Remote Port Number -->
            <xsl:if test="data[@name='c-port']/@value != '-'>
                <Port>
                    <xsl:value-of select="data[@name='c-port']/@value" />
                </Port>
            </xsl:if>
        </Device>
    </Source>
    <Destination>
        <Device>
            <HostName>
                <xsl:value-of select="data[@name='cs-host']/@value" />
            </HostName>
        </Device>
    </Destination>
</xsl:template>
```

```

<!-- Define the Payload node -->
<xsl:template name="setPayload">

<Payload>
    <xsl:if test="data[@name='cs-uri-query']/@value != '-'">
        <Criteria>
            <DataSources>
                <DataSource>
                    <xsl:value-of select="concat(data[@name='cs-uri-scheme']/@value, '://', data[@name='cs-host']/@value)" />
                    <xsl:if test="data[@name='cs-uri-path']/@value != '/'">
                        <xsl:value-of select="data[@name='cs-uri-path']/@value" />
                    </xsl:if>
                </DataSource>
            </DataSources>
            <Query>
                <Raw>
                    <xsl:value-of select="data[@name='cs-uri-query']/@value" />
                </Raw>
            </Query>
        </Criteria>
    </xsl:if>
    <Resource>

        <!-- Check for auth groups the URL belongs to -->
        <xsl:variable name="authgroups">
            <xsl:value-of select="data[@name='cs-auth-group']/@value" />
            <xsl:if test="exists(data[@name='cs-auth-group']) and exists(data[@name='cs-auth-groups'])">, </xsl:if>
            <xsl:value-of select="data[@name='cs-auth-groups']/@value" />
        </xsl:variable>
        <xsl:choose>
            <xsl:when test="contains($authgroups, ',')">
                <Groups>
                    <xsl:for-each select="tokenize($authgroups, ',')">
                        <Group>
                            <Id>
                                <xsl:value-of select="." />
                            </Id>
                        </Group>
                    </xsl:for-each>
                </Groups>
            </xsl:when>
            <xsl:when test="$authgroups != '-' and $authgroups != ''">
                <Groups>
                    <Group>
                        <Id>
                            <xsl:value-of select="$authgroups" />
                        </Id>
                    </Group>
                </Groups>
            </xsl:when>
        </xsl:choose>

        <!-- Re-form the URL -->
        <URL>
            <xsl:value-of select="concat(data[@name='cs-uri-scheme']/@value, '://', data[@name='cs-host']/@value)" />
            <xsl:if test="data[@name='cs-uri-path']/@value != '/'">
                <xsl:value-of select="data[@name='cs-uri-path']/@value" />
            </xsl:if>
        </URL>
        <HTTPMethod>
            <xsl:value-of select="data[@name='cs-method']/@value" />
        </HTTPMethod>
        <xsl:if test="data[@name='cs(User-Agent)']/@value != '-'">
            <UserAgent>
                <xsl:value-of select="data[@name='cs(User-Agent)']/@value" />
            </UserAgent>
        </xsl:if>
    </Resource>

```

```

</xsl:if>

<!-- Inbound activity -->
<xsl:if test="data[@name='sc-bytes']/@value != '-'">
  <InboundSize>
    <xsl:value-of select="data[@name='sc-bytes']/@value" />
  </InboundSize>
</xsl:if>
<xsl:if test="data[@name='sc-bodylength']/@value != '-'">
  <InboundContentSize>
    <xsl:value-of select="data[@name='sc-bodylength']/@value" />
  </InboundContentSize>
</xsl:if>

<!-- Outbound activity -->
<xsl:if test="data[@name='cs-bytes']/@value != '-'">
  <OutboundSize>
    <xsl:value-of select="data[@name='cs-bytes']/@value" />
  </OutboundSize>
</xsl:if>
<xsl:if test="data[@name='cs-bodylength']/@value != '-'">
  <OutboundContentSize>
    <xsl:value-of select="data[@name='cs-bodylength']/@value" />
  </OutboundContentSize>
</xsl:if>

<!-- Miscellaneous -->
<RequestTime>
  <xsl:value-of select="data[@name='time-taken']/@value" />
</RequestTime>
<ResponseCode>
  <xsl:value-of select="data[@name='sc-status']/@value" />
</ResponseCode>
<xsl:if test="data[@name='rs(Content-Type)']/@value != '-'">
  <MimeType>
    <xsl:value-of select="data[@name='rs(Content-Type)']/@value" />
  </MimeType>
</xsl:if>
<xsl:if test="data[@name='cs-categories']/@value != 'none' or data[@name='sc-filter-category']/@value != 'none'">
  <Category>
    <xsl:value-of select="data[@name='cs-categories']/@value" />
    <xsl:value-of select="data[@name='sc-filter-category']/@value" />
  </Category>
</xsl:if>

<!-- Take up other items as data elements -->
<xsl:apply-templates select="data[@name='s-action']" />
<xsl:apply-templates select="data[@name='cs-uri-scheme']" />
<xsl:apply-templates select="data[@name='s-hierarchy']" />
<xsl:apply-templates select="data[@name='sc-filter-result']" />
<xsl:apply-templates select="data[@name='x-virus-id']" />
<xsl:apply-templates select="data[@name='x-virus-details']" />
<xsl:apply-templates select="data[@name='x-icap-error-code']" />
<xsl:apply-templates select="data[@name='x-icap-error-details']" />
</Resource>
</Payload>
</xsl:template>

<!-- Generic Data capture template so we capture all other Bluecoat objects not already consumed -->
<xsl:template match="data">
  <xsl:if test="@value != '-'">
    <Data Name="{@name}" Value="{@value}" />
  </xsl:if>
</xsl:template>

<!--

```

Set up the Outcome node.

We only set an Outcome for an error state. The absence of an Outcome infers success

```
-->
<xsl:template name="setOutcome">
  <xsl:choose>

    <!-- Favour squid specific errors first -->
    <xsl:when test="data[@name='sc-status']/@value > 500">
      <Outcome>
        <Success>false</Success>
        <Description>
          <xsl:call-template name="responseCodeDesc">
            <xsl:with-param name="code" select="data[@name='sc-status']/@value" />
          </xsl:call-template>
        </Description>
      </Outcome>
    </xsl:when>

    <!-- Now check for 'normal' errors -->
    <xsl:when test="data[@name='sc-status']/@value > 400">
      <Outcome>
        <Success>false</Success>
        <Description>
          <xsl:call-template name="responseCodeDesc">
            <xsl:with-param name="code" select="data[@name='sc-status']/@value" />
          </xsl:call-template>
        </Description>
      </Outcome>
    </xsl:when>
  </xsl:choose>
</xsl:template>

<!-- Response Code map to Descriptions -->
<xsl:template name="responseCodeDesc">
  <xsl:param name="code" />
  <xsl:choose>

    <!-- Informational -->
    <xsl:when test="$code = 100">Continue</xsl:when>
    <xsl:when test="$code = 101">Switching Protocols</xsl:when>
    <xsl:when test="$code = 102">Processing</xsl:when>

    <!-- Successful Transaction -->
    <xsl:when test="$code = 200">OK</xsl:when>
    <xsl:when test="$code = 201">Created</xsl:when>
    <xsl:when test="$code = 202">Accepted</xsl:when>
    <xsl:when test="$code = 203">Non-Authoritative Information</xsl:when>
    <xsl:when test="$code = 204">No Content</xsl:when>
    <xsl:when test="$code = 205">Reset Content</xsl:when>
    <xsl:when test="$code = 206">Partial Content</xsl:when>
    <xsl:when test="$code = 207">Multi Status</xsl:when>

    <!-- Redirection -->
    <xsl:when test="$code = 300">Multiple Choices</xsl:when>
    <xsl:when test="$code = 301">Moved Permanently</xsl:when>
    <xsl:when test="$code = 302">Moved Temporarily</xsl:when>
    <xsl:when test="$code = 303">See Other</xsl:when>
    <xsl:when test="$code = 304">Not Modified</xsl:when>
    <xsl:when test="$code = 305">Use Proxy</xsl:when>
    <xsl:when test="$code = 307">Temporary Redirect</xsl:when>

    <!-- Client Error -->
    <xsl:when test="$code = 400">Bad Request</xsl:when>
    <xsl:when test="$code = 401">Unauthorized</xsl:when>
    <xsl:when test="$code = 402">Payment Required</xsl:when>
```

```

<xsl:when test="$code = 403">Forbidden</xsl:when>
<xsl:when test="$code = 404">Not Found</xsl:when>
<xsl:when test="$code = 405">Method Not Allowed</xsl:when>
<xsl:when test="$code = 406">Not Acceptable</xsl:when>
<xsl:when test="$code = 407">Proxy Authentication Required</xsl:when>
<xsl:when test="$code = 408">Request Timeout</xsl:when>
<xsl:when test="$code = 409">Conflict</xsl:when>
<xsl:when test="$code = 410">Gone</xsl:when>
<xsl:when test="$code = 411">Length Required</xsl:when>
<xsl:when test="$code = 412">Precondition Failed</xsl:when>
<xsl:when test="$code = 413">Request Entity Too Large</xsl:when>
<xsl:when test="$code = 414">Request URI Too Large</xsl:when>
<xsl:when test="$code = 415">Unsupported Media Type</xsl:when>
<xsl:when test="$code = 416">Request Range Not Satisfiable</xsl:when>
<xsl:when test="$code = 417">Expectation Failed</xsl:when>
<xsl:when test="$code = 422">Unprocessable Entity</xsl:when>
<xsl:when test="$code = 424">Locked/Failed Dependency</xsl:when>
<xsl:when test="$code = 433">Unprocessable Entity</xsl:when>

<!-- Server Error -->
<xsl:when test="$code = 500">Internal Server Error</xsl:when>
<xsl:when test="$code = 501">Not Implemented</xsl:when>
<xsl:when test="$code = 502">Bad Gateway</xsl:when>
<xsl:when test="$code = 503">Service Unavailable</xsl:when>
<xsl:when test="$code = 504">Gateway Timeout</xsl:when>
<xsl:when test="$code = 505">HTTP Version Not Supported</xsl:when>
<xsl:when test="$code = 507">Insufficient Storage</xsl:when>
<xsl:when test="$code = 600">Squid: header parsing error</xsl:when>
<xsl:when test="$code = 601">Squid: header size overflow detected while parsing/roundcube: software configuration error</xsl:when>
<xsl:when test="$code = 603">roundcube: invalid authorization</xsl:when>
<xsl:otherwise>
    <xsl:value-of select="concat('Unknown Code:', $code)" />
</xsl:otherwise>
</xsl:choose>
</xsl:template>
</xsl:stylesheet>

```

Refreshing the current event will show the *output* pane contains

```

<?xml version="1.1" encoding="UTF-8"?>
<Events xmlns="event-logging:3" xmlns:stroom="stroom" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs="http://www.w3.org/2001/XMLSchema">
</Events>
<Event>
<EventTime>
<TimeCreated>2005-05-04T17:16:12.000Z</TimeCreated>
</EventTime>
<EventSource>
<System>
<Name>Site http://log-sharing.dreamhosters.com/ Bluecoat Logs</Name>
<Environment>Development</Environment>
</System>
<Generator>Bluecoat Software: SGOS 3.2.4.28 Version: 1.0</Generator>
<Device>
<IPAddress>192.16.170.42</IPAddress>
<Data Name="ServiceType" Value="SG-HTTP-Service" />
</Device>
<Client>
<IPAddress>45.110.2.82</IPAddress>
</Client>
<Server>
<HostName>www.inmobus.com</HostName>
</Server>
<User>
<Id>george</Id>
</User>
<Data Name="MyMeta" Value="FQDN:somenode.strmdev00.org\nipaddress:192.168.2.220\nipaddress_eth0:192.168.2.220\nipaddress_1
</EventSource>
<EventDetail>
<TypeId>Bluecoat-GET-http</TypeId>
<Description>Receipt of information from a Resource via Proxy</Description>
<Receive>
<Source>
<Device>
<IPAddress>45.110.2.82</IPAddress>
</Device>
</Source>
<Destination>
<Device>
<HostName>www.inmobus.com</HostName>
</Device>
</Destination>
<Payload>
<Resource>
<URL>http://www.inmobus.com/wcm/assets/images/imagefileicon.gif</URL>
<HTTPMethod>GET</HTTPMethod>
<UserAgent>Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)</UserAgent>
<InboundSize>941</InboundSize>
<OutboundSize>729</OutboundSize>
<RequestTime>1</RequestTime>
<ResponseCode>200</ResponseCode>
<MimeType>image/gif</MimeType>
<Data Name="s-action" Value="TCP_HIT" />
<Data Name="cs-uri-scheme" Value="http" />
<Data Name="s-hierarchy" Value="DIRECT" />
<Data Name="sc-filter-result" Value="PROXIED" />
<Data Name="x-icap-error-code" Value="none" />
</Resource>
</Payload>
</Receive>
</EventDetail>
</Event>
</Events>

```

for the given input

```

<?xml version="1.1" encoding="UTF-8"?>
<records xmlns="records:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="records:2 file:///records-v2
<record>
  <data name="date" value="2005-05-04" />
  <data name="time" value="17:16:12" />
  <data name="time-taken" value="1" />
  <data name="c-ip" value="45.110.2.82" />
  <data name="sc-status" value="200" />
  <data name="s-action" value="TCP_HIT" />
  <data name="sc-bytes" value="941" />
  <data name="cs-bytes" value="729" />
  <data name="cs-method" value="GET" />
  <data name="cs-uri-scheme" value="http" />
  <data name="cs-host" value="www.inmobus.com" />
  <data name="cs-uri-path" value="/wcm/assets/images/imagefileicon.gif" />
  <data name="cs-uri-query" value="-" />
  <data name="cs-username" value="george" />
  <data name="s-hierarchy" value="DIRECT" />
  <data name="s-supplier-name" value="38.112.92.20" />
  <data name="rs(Content-Type)" value="image/gif" />
  <data name="cs(User-Agent)" value="Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)" />
  <data name="sc-filter-result" value="PROXIED" />
  <data name="sc-filter-category" value="none" />
  <data name="x-virus-id" value="-" />
  <data name="s-ip" value="192.16.170.42" />
  <data name="s-sitename" value="SG-HTTP-Service" />
  <data name="x-virus-details" value="-" />
  <data name="x-icap-error-code" value="none" />
  <data name="x-icap-error-details" value="-" />
</record>
</records>
```

Do not forget to Save



[Save](#)

the translation as we are complete.

4.6.2.5.4.1 Schema Validation

One last point, validating the use of the Stroom Event Logging Schema is performed in the `schemaFilter` component of the pipeline. Had our translation resulted in a malformed Event, this pipeline component displays any errors. In the screen below, we have purposely changed the `EventTime/TimeCreated` element to be `EventTime/TimeCreatd`. If one selects the `schemaFilter` component and then Refresh



[Refresh Current Step](#)

the current step, we will see that

- there is an error as indicated by a square **Red** box



[Error Indicator](#)

in the top right hand corner

- there is a **Red** rectangle line indicator mark



[Error Line Indicator](#)

on the right hand side in the display slide bar

- there is a **Red** error marker



[Error Marker](#)

on the left hand side

```
1 <?xml version="1.1" encoding="UTF-8"?>
2 <Events xmlns="event-logging:3" xmlns:stroom="stroom" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:schemaLocation="event-logging:3 file:///eve
3 <Event>
4   <EventTime>
5     <TimeCreatd>2005-05-04T17:16:12.000Z</TimeCreatd>
6   </EventTime>
7   <EventSource>
8     <System>
9       <Name>Site http://log-sharing.dreamhosters.com/ Bluecoat Logs</Name>
10      <Environment>Development</Environment>
```

[Stroom UI Create Feed - Translation - Stepping XSLT Translation 6](#)

Hovering over the error marker

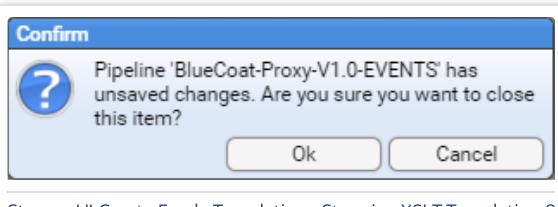


on the left hand side will bring a pop-up describing the error.

3 <Event>
4 <EventTime>
5 <TimeCreatd>2005-05-04T17:16:12.000Z</TimeCreatd>
6 </EventTime>
7 Invalid content was found starting with element 'TimeCreatd'. One of '{TimeCreated}' is expected.
8 <System>
9 <Name>Site http://log-sharing.dreamhosters.com/ Bluecoat Logs</Name>
10 <Environment>Development</Environment>

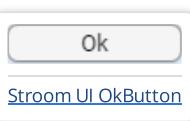
[Stroom UI Create Feed - Translation - Stepping XSLT Translation 7](#)

At this point, close the **BlueCoat-Proxy-V1.0-EVENTS** stepping tab, acknowledging you do not want to save your errant changes



[Stroom UI Create Feed - Translation - Stepping XSLT Translation 8](#)

by pressing the



[Stroom UI OkButton](#)

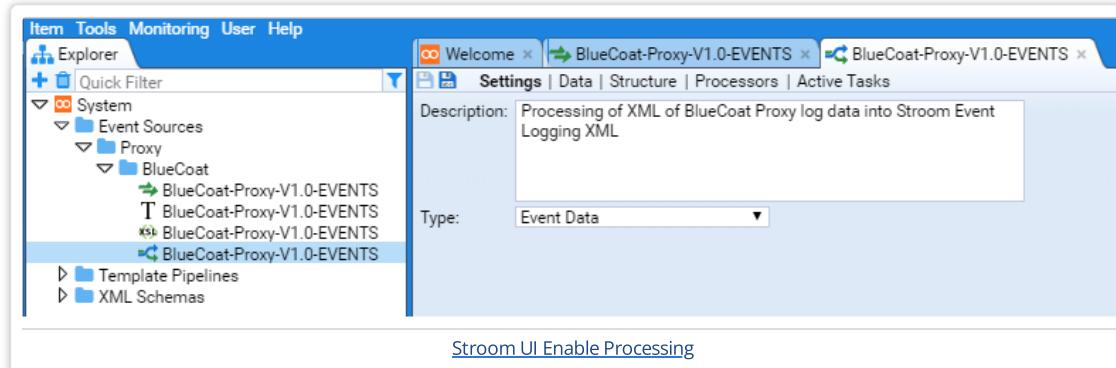
button.

4.6.2.6 Automated Processing

Now that we have authored our translation, we want to enable Stroom to automatically process streams of raw event log data as it arrives. We do this by configuring a Processor in the **BlueCoat-Proxy-V1.0-EVENTS** pipeline.

4.6.2.6.1 Adding a Pipeline Processor

Open the **BlueCoat-Proxy-V1.0-EVENTS** pipeline by selecting it (*double left click*) in the Explorer display to show



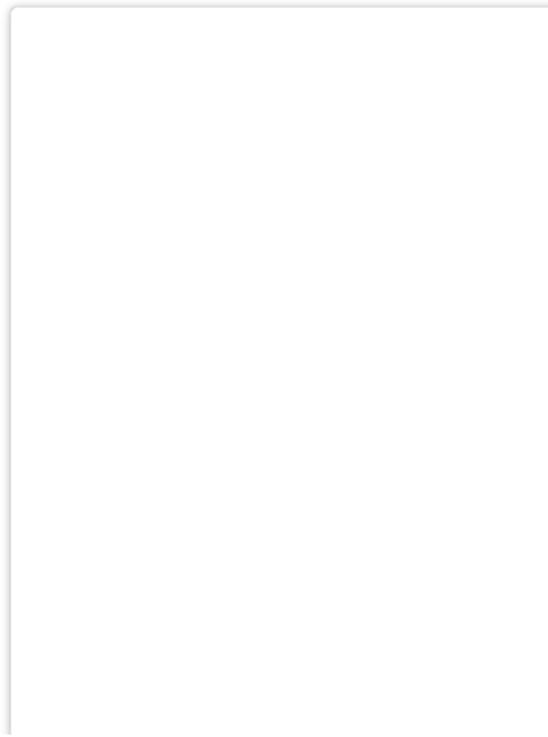
To configure a Processor we select the `Processors` hyper-link of the `*BlueCoat-Proxy-V1.0-EVENTS` Pipeline tab to reveal

Pipeline	Tracker Ms	Tracker %	Last Poll Age	Task Count	Priority	Streams	Events	Status	Enabled
Stroom UI Enable Processing - Processors table									

We add a Processor by pressing the add processor button



in the top left hand corner. At this you will be presented with an **Add Filter** configuration window.



Add Filter

Template:

Folders:	Feeds:
+ -	<input type="button" value="Edit"/>
Pipelines:	Stream Types:
+ -	+ -
Stream Id:	Parent Stream Id:
Created:	
Effective:	
Stream Attributes:	
+ -	

[Stroom UI Enable Processing - Add Filter 1](#)

As we wish to create a Processor that will automatically process all **BlueCoat-Proxy-V1.0-EVENTS** feed Raw Events we will select the BlueCoat-Proxy-V1.0-EVENTS Feed and Raw Event Stream Type .

To select the feed, we press the Edit button



[Stroom UI EditButton](#)

. At this, the **Choose Feeds To Include And Exclude** configuration window is displayed.



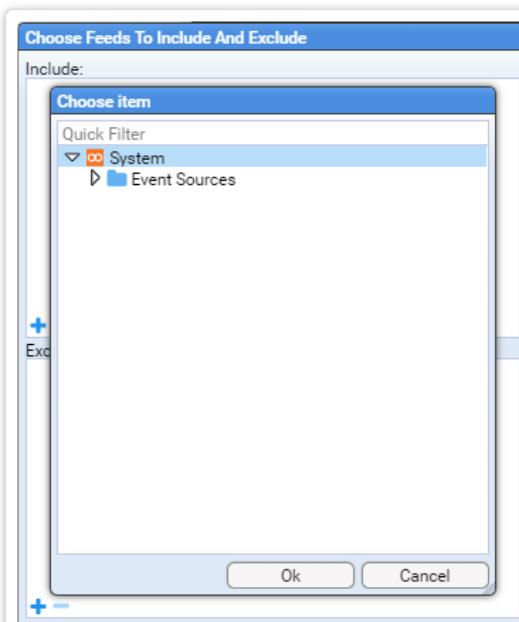


[Stroom UI Enable Processing - Add Filter 2](#)

As we need to **Include** the **BlueCoat-Proxy-V1.0-EVENTS** Feed in our selection, press the

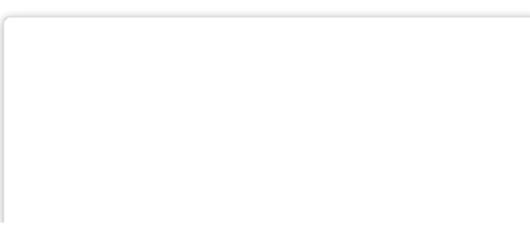


button in the **Include:** pane of the window to be presented with a **Choose Item** configuration window.



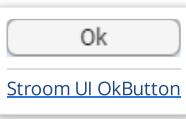
[Stroom UI Enable Processing - Add Filter 3](#)

Navigate to the **Event Sources/Proxy/BlueCoat** folder and select the **BlueCoat-Proxy-V1.0-EVENTS** Feed

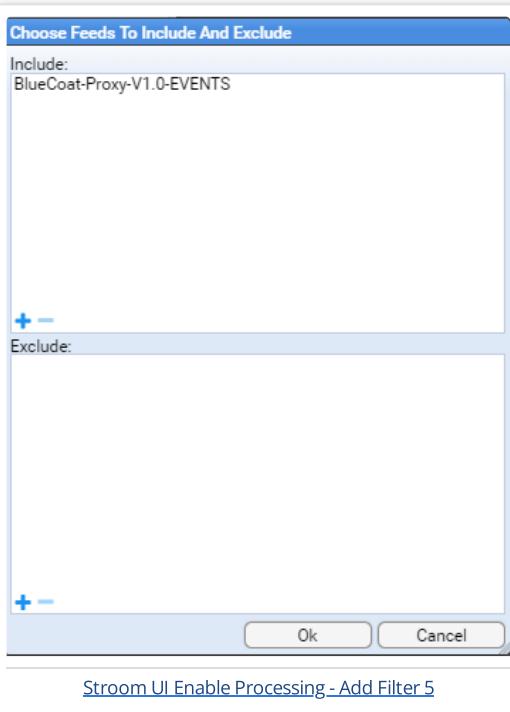




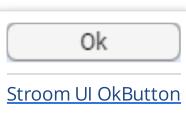
then press the



button to select and see that the feed is included.



Again press the



button to close the **Choose Feeds To Include And Exclude** window to show that we have selected our feed in the **Feeds:** selection pane of the **Add Filter** configuration window.

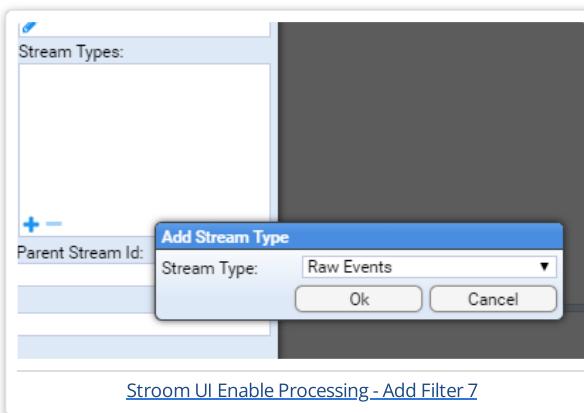


[Stroom UI Enable Processing - Add Filter 6](#)

We now need to select our Stream Type . Press the

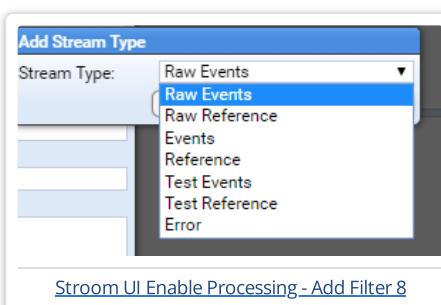


button in the Stream Types: pane of the window to be presented with a Add Stream Type window with a Stream Type: selection drop down.



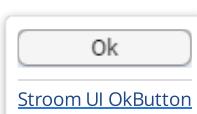
[Stroom UI Enable Processing - Add Filter 7](#)

We select (*left click*) the drop down selection to display the types of Stream we can choose



[Stroom UI Enable Processing - Add Filter 8](#)

and as we are selecting Raw Events we select that item then press the



[Stroom UI OkButton](#)

button at which we see that our Add Filter configuration window displays

Add Filter

Template:

Folders:	Feeds:
	+ BlueCoat-Proxy-V1.0-EVENTS

Pipelines:

Stream Types:
Raw Events

Stream Id: **Parent Stream Id:**

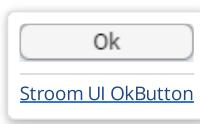
Created: **Effective:**

Stream Attributes:

Ok **Cancel**

[Stroom UI Enable Processing - Add Filter 9](#)

As we have selected our filter items, press the



button to display our configured Processors.

[Stroom UI Enable Processing - Configured Processors](#)

Pipeline	Tracker Ms	Last Poll Age	Task Count	Priority	Streams	Events	Status	Enabled
BlueCoat-Proxy-V1.0-EVENTS								<input type="checkbox"/>
BlueCoat-Proxy-V1.0-EVENTS								<input type="checkbox"/>

AND { Feed = BlueCoat-Proxy-V1.0-EVENTS
Stream Type = Raw Events }

We now see our display is divided into two panes. The Processors table pane at the top and the specific Processor pane below. In our case, our filter selection has left the **BlueCoat-Proxy-V1.0-EVENTS** Filter selected in the Processors table

[Stroom UI Enable Processing - Configured Processors - Selected Processor](#)

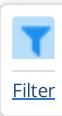
and the specific filter's details in the bottom pane.

AND { Feed = BlueCoat-Proxy-V1.0-EVENTS
Stream Type = Raw Events

[Stroom UI Enable Processing - Configured Processors - Selected Processor Detail](#)

The column entries in the Processors Table pane describe

- Pipeline - the name of the Processor pipeline (



Filter

)

- Tracker Ms - the last time the tracker updated
- Tracker % - the percentage of available streams completed
- Last Poll Age - the last time the processor found new streams to process
- Task Count - the number of processor tasks currently running
- Priority - the queue scheduling priority of task submission to available stream processors
- Streams - the number of streams that have been processed (includes currently running streams)
- Events - ??
- Status - the status of the processor. Normally empty if the number of stream is open-ended. If only a subset of streams were chosen (e.g. a time range in the filter) then the status will be *Complete*
- Enabled - check box to indicate the processor is enabled

We now need only `Enable` both the pipeline Processor and the pipeline Filter for automatic processing to occur. We do this by selecting both check boxes in the `Enabled` column.

Pipeline	Tracker Ms	Tracker %	Last Poll Age	Task Count	Priority	Streams	Events	Status	Enabled
▽ BlueCoat-Proxy-V1.0-EVENTS						10			<input checked="" type="checkbox"/>
> BlueCoat-Proxy-V1.0-EVENTS									<input checked="" type="checkbox"/>

[Stroom UI Enable Processing - Configured Processors - Enable Processor](#)

If we refresh our Processor table by pressing the



[Stroom UI RefreshButton](#)

button in the top right hand corner, we will see that more table entries have been filled in.

Pipeline	Tracker Ms	Tracker %	Last Poll Age	Task Count	Priority	Streams	Events	Status	Enabled
▽ BlueCoat-Proxy-V1.0-EVENTS						10			<input checked="" type="checkbox"/>
> BlueCoat-Proxy-V1.0-EVENTS	2018-07-14T04:00:35.289Z	100	2.3m	0	10	1			<input checked="" type="checkbox"/>

[Stroom UI Enable Processing - Configured Processors - Enable Processor Result](#)

We see that the tracker last updated at `2018-07-14T04:00:35.289Z`, the percentage complete is `100` (we only had one stream after all), the last time active streams were checked for was `2.3` minutes ago, there are no tasks running and that `1` stream has completed. Note that the `Status` column is blank as we have an *open ended* filter in that the processor will continue to select and process any new stream of `Raw Events` coming into the `BlueCoat-Proxy-V1.0-EVENTS` feed.

If we return to the `BlueCoat-Proxy-V1.0-EVENTS*` Feed tab, ensuring the **Data** hyper-link is selected and then refresh (



[Refresh](#)

) the top pane that holds the summary of the latest Feed streams

Stroom UI Enable Processing - Configured Processors - Feed Display

We see a new entry in the table. The columns display

- Created - The time the stream was created.
- Type - The type of stream. Our new entry has a type of 'Events' as we have processed our Raw Events data.
- Feed - The name of the stream's feed
- Pipeline - The name of the pipeline involved in the generation of the stream
- Raw - The size in bytes of the raw stream data
- Disk - The size in bytes of the raw stream data when stored in compressed form on the disk
- Read - The number of records read by a pipeline
- Write - The number of records (events) written by a pipeline. In this case the difference is that we did not generate events for the Software or Version records we read.
- Fatal - The number of fatal errors the pipeline encountered when processing this stream
- Error - The number of errors the pipeline encountered when processing this stream
- Warn - The number of warnings the pipeline encountered when processing this stream
- Info - The number of informational alerts the pipeline encountered when processing this stream
- Retention - The retention period for this stream of data

If we also refresh (



) the specific feed pane (middle) we again see a new entry of the **Events** Type

Stroom UI Enable Processing - Configured Processors - Specific Feed Display

If we select (*left click*) on the **Events** Type in either pane, we will see that the data pane displays the first event in the *GCHQ Stroom Event Logging XML Schema* form.

Stroom UI Enable Processing - Configured Processors - Event Display.

We can now send a file of BlueCoat Proxy logs to our Stroom instance from a Linux host using *curl* command and see how Stroom will automatically processes the file. Use the command

```
curl -k --data-binary @sampleBluecoat.log https://stroomp.strmdev00.org/stroom/datafeed -H "Feed:BlueCoat-Proxy-V1.0-EVENTS" -H "E
```

After Stroom's Proxy aggregation has occurred, we will see that the new file posted via *curl* has been loaded into Stroom as per

Stroom UI Enable Processing - Configured Processors - New Posted Stream

and this new *Raw Event* stream is automatically processed a few seconds later as per

We note that since we have used the same sample file again, the Stream sizes and record counts are the same.

If we switch to the *Processors* tab of the pipeline we see that the Tracker timestamp has changed and the number of Streams processed has increased.

Stroom UI Enable Processing - Configured Processors - New Posted Stream Processors

4.7 - General

General How Tos for using Stroom.

4.7.1 - Feed Management

This HOWTO demonstrates how to manage feeds.

This HOWTO demonstrates how to manage [Feeds](#)

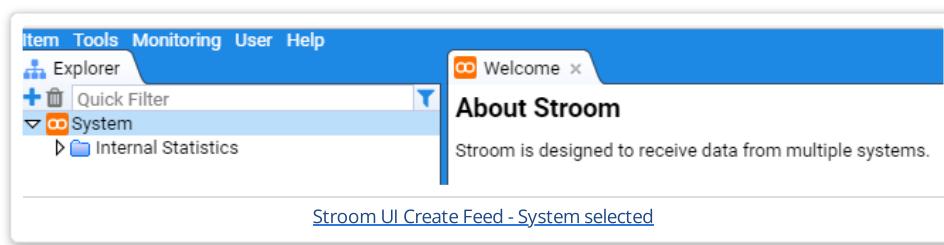
4.7.1.1 Assumptions

- All Sections
- an account with the [Administrator Application Permission](#) is currently logged in.

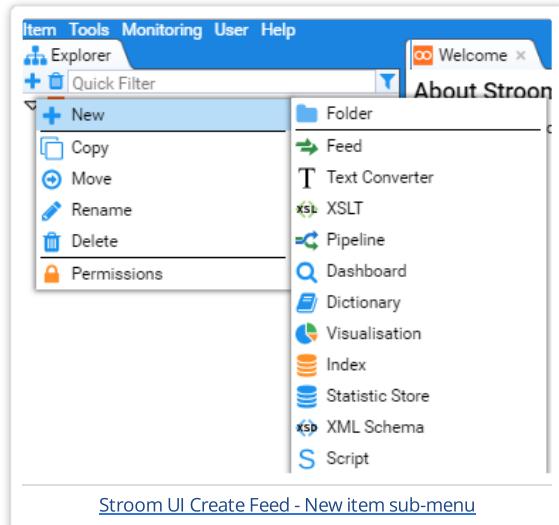
4.7.1.1.1 Creation of an Event Feed

We will be creating an Event Feed with the name `TEST-FEED-V1_0`.

Once you have logged in, move the cursor to the **System** folder within the `Explorer` tab and select it.



Once selected, *right click* to bring up the `New Item` selection sub-menu. By selecting the **System** folder we are requesting any *new item* created to be placed within it.



Now move the cursor to the *Feed* sub-item



and select it. You will be presented with a [New Feed](#) configuration window.





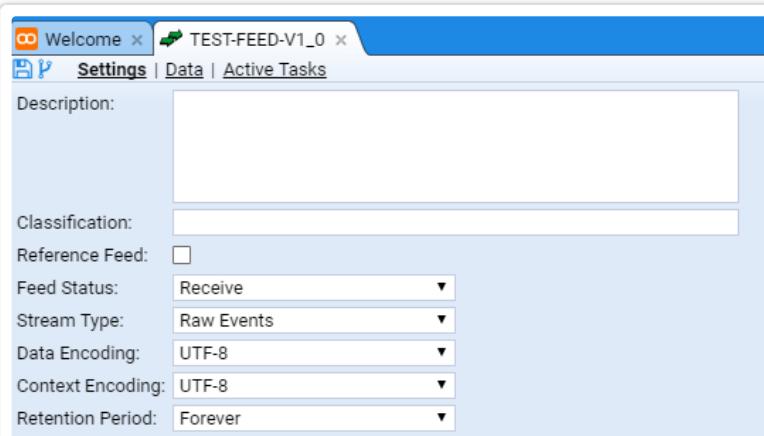
You will note that the **System** folder has already been selected as the *parent group* and all we need to do is enter our feed's name in the **Name:** entry box



On pressing



we are presented with the **Feed** tab for our new feed. The tab is labelled with the feed name **TEST-FEED-V1_0**.



We will leave the definitions of the Feed attributes for the present, but we *will* enter a **Description:** for our feed as we should *ALWAYS* do this fundamental tenet of data management - document the data. We will use the description of '*Feed for installation validation only. No data value.*'.

Stroom UI Create Feed - New feed tab with Description

One should note that the `Feed` tab has been marked as having unsaved changes. This is indicated by the asterisk character * between the `Feed` icon



and the name of the feed `TEST-FEED-V1_0`. We can save the changes to our feed by pressing the `Save` icon



in the top left of the `TEST-FEED-V1_0` tab. At this point one should notice two things, the first is that the asterisk has disappeared from the `Feed` tab and the `Save` icon



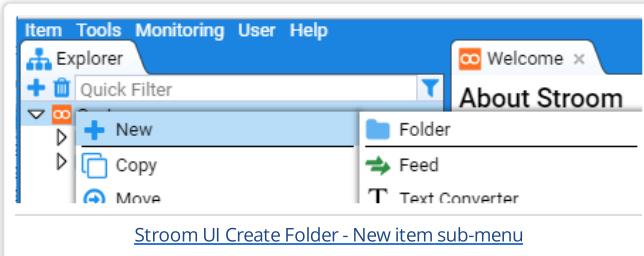
is *ghosted*.

Stroom UI Create Feed - New feed tab with description saved

4.7.1.1.2 Folder Structure for Event Sources

In order to simplify the management of multiple event sources being processed by Stroom, it is suggested that an Event Source folder is created at the root of the **System** folder in the **Explorer** tab.

This can be achieved by moving the cursor to the **System** folder within the **Explorer** tab and select it. Once selected, *right click* to bring up the **New Item** selection sub-menu.



Now move the cursor to the *Folder* sub-item



and select it. You will be presented with a **New Folder** configuration window.



You will note that the **System** folder has already been selected as the *parent group* and all we need to do is enter our folder's name in the **Name:** entry box



On pressing



we are presented with the **Folder** tab for our new folder. The tab is labelled with the folder name **Event Sources**.



You will also note that the **Explorer** tab has displayed the **Event Sources** folder in its display.

4.7.1.1.3 Create Folder for specific Event Source

In order to manage all artefacts of a given Event Source (aka **Feed**), one would create an appropriately named sub-folder within the **Event Sources** folder structure.

In this example, we will create one for a BlueCoat Proxy **Feed**.

As we may eventually have multiple proxy event sources, we will first create a **Proxy** folder in the **Event Sources** before creating the desired **BlueCoat** folder that will hold the processing components.

So, move the cursor to the **Event Sources** folder on the **Explorer** tab, select it and then *right click* to bring up the **New Item** selection sub-menu and move the cursor to the **Folder** sub-item



and select it. You will be presented with a **New Folder** configuration window.

Enter **Proxy** as the folder **Name**:



[Stroom UI Create Folder - New sub folder configuration window](#)

and press the



At this you will be presented with a new `Folder` tab for the new sub-folder and we note that it has been added below the **Event Sources** folder in the `Explorer` tab



[Stroom UI Create Folder - New sub folder tab](#)

Repeat this process to create the desired **BlueCoat** sub-folder with the result



[Stroom UI Create Folder - New BlueCoat sub folder tab](#)

4.7.2 - Raw Source Tracking

How to link every Event back to the Raw log

Stroom v6.1 introduced a new feature (`stroom:source()`) to allow a translation developer to obtain positional details of the source file that is currently being processed. Using the positional information it is possible to tag Events with sufficient details to link back to the Raw source.

4.7.2.1 Assumptions

1. You have a working pipeline that processes logs into Events.
2. Events are indexed
3. You have a Dashboard uses a Search Extraction pipeline.

4.7.2.2 Steps

1. Create a new XSLT called Source Decoration containing the following:

```
<xsl:stylesheet
    xpath-default-namespace="event-logging:3"
    xmlns:sm="stroom-meta" xmlns="event-logging:3"
    xmlns:rec="records:2"
    xmlns:stroom="stroom"
    version="3.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="@*|node()">
        <xsl:copy>
            <xsl:apply-templates select="@*|node()" />
        </xsl:copy>
    </xsl:template>
    <xsl:template match="Event/Meta[not(sm:source)]">
        <xsl:copy>
            <xsl:apply-templates />
            <xsl:copy-of select="stroom:source()" />
        </xsl:copy>
    </xsl:template>
    <xsl:template match="Event[not(Meta)]">
        <xsl:copy>
            <xsl:element name="Meta">
                <xsl:copy-of select="stroom:source()" />
            </xsl:element>
            <xsl:apply-templates />
        </xsl:copy>
    </xsl:template>
</xsl:stylesheet>
```

This XSLT will add or augment the Meta section of the Event with the source details.

2. Insert a new XSLT filter into your translation pipeline after your translation filter and set it to the XSLT created above.
3. Reprocess the Events through the modified pipeline, also ensure your Events are indexed.
4. Amend the translation performed by the Extraction pipeline to include the new data items that represent the source position data.
Add the following to the XSLT:

```

<xsl:element name="data">
  <xsl:attribute name="name">
    <xsl:text>src-id</xsl:text>
  </xsl:attribute>
  <xsl:attribute name="value" select="Meta/sm:source/sm:id" />
</xsl:element>
<xsl:element name="data">
  <xsl:attribute name="name">
    <xsl:text>src-partNo</xsl:text>
  </xsl:attribute>
  <xsl:attribute name="value" select="Meta/sm:source/sm:partNo" />
</xsl:element>
<xsl:element name="data">
  <xsl:attribute name="name">
    <xsl:text>src-recordNo</xsl:text>
  </xsl:attribute>
  <xsl:attribute name="value" select="Meta/sm:source/sm:recordNo" />
</xsl:element>
<xsl:element name="data">
  <xsl:attribute name="name">
    <xsl:text>src-lineFrom</xsl:text>
  </xsl:attribute>
  <xsl:attribute name="value" select="Meta/sm:source/sm:lineFrom" />
</xsl:element>
<xsl:element name="data">
  <xsl:attribute name="name">
    <xsl:text>src-colFrom</xsl:text>
  </xsl:attribute>
  <xsl:attribute name="value" select="Meta/sm:source/sm:colFrom" />
</xsl:element>
<xsl:element name="data">
  <xsl:attribute name="name">
    <xsl:text>src-lineTo</xsl:text>
  </xsl:attribute>
  <xsl:attribute name="value" select="Meta/sm:source/sm:lineTo" />
</xsl:element>
<xsl:element name="data">
  <xsl:attribute name="name">
    <xsl:text>src-colTo</xsl:text>
  </xsl:attribute>
  <xsl:attribute name="value" select="Meta/sm:source/sm:colTo" />
</xsl:element>

```

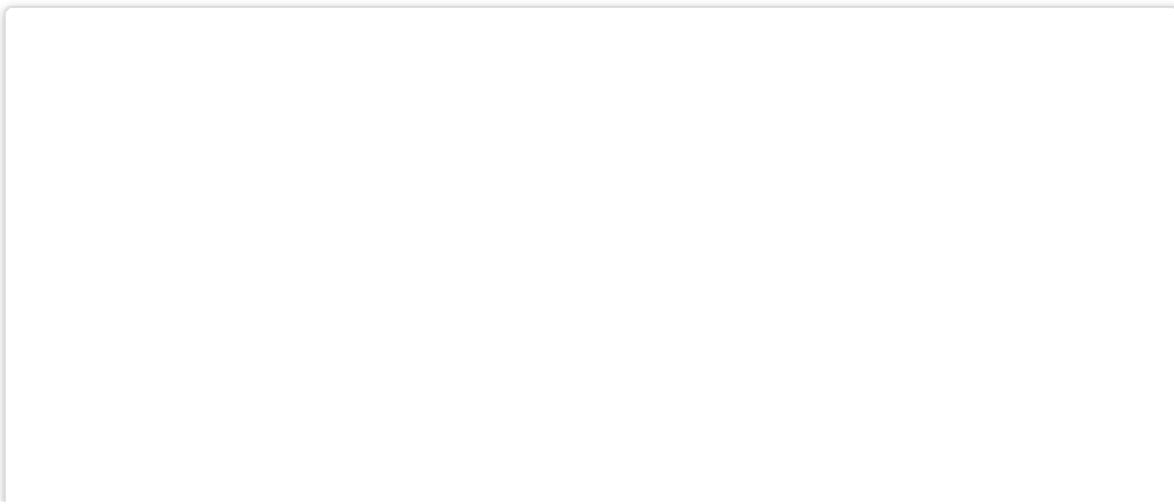
5. Open your dashboard, now add the following custom fields to your table:

```

${src-id}, ${src-partNo}, ${src-recordNo}, ${src-lineFrom}, ${src-lineTo}, ${src-colFrom}, ${src-colTo}

```

6. Now add a New Text Window to your Dashboard, and configure it as below:





7. You can also add a column to the table that will open a data window showing the source. Add a custom column with the following expression:

```
data('Raw Log', ${src-id}, ${src-partNo}, '', ${src-lineFrom}, ${src-colFrom}, ${src-lineTo}, ${src-colTo})
```

4.7.3 - Task Management

This HOWTO demonstrates how to manage background tasks.

Various [Tasks](#) run in the background within Stroom. This HOWTO demonstrates how to manage these tasks

4.7.3.1 Assumptions

- All Sections
- an account with the Administrator Application [Permission](#) is currently logged in.
- Proxy Aggregation Tasks
- we have a multi node Stroom cluster with two nodes, `stroomp00` and `stroomp01`.
- Stream Processor Tasks
- we have a multi node Stroom cluster with two nodes, `stroomp00` and `stroomp01`.
- when demonstrating adding a new node to an existing cluster, the new node is `stroomp02`.

4.7.3.2 Proxy Aggregation

4.7.3.3 Turn Off Proxy Aggregation

We first select the `Monitoring` item of the **Main Menu** to bring up the `Monitoring` sub-menu.



then move down and select the `Jobs` sub-item to be presented with the `Jobs` configuration tab as seen below.

Job	Node	Type	Max	Cur	Last Executed	Enabled
Disable Unused Accounts						
File System Clean						
Node Status						
Proxy Aggregation						
Stream Processor						
Volume Status						
Index Shard Retention						
Pipeline Destination Roll						
Property Cache Reload						
SQL Stats Database Aggregation						
SQL Stats In Memory Flush						
Stream Attributes Retention						
Stream Delete						
Stream Retention						

[Stroom UI Jobs Management - management tab](#)

At this we can select the `Proxy Aggregation` *Job* whose check-box is selected and the tab will show the individual Stroom Processor nodes in the deployment.

Job	Enabled	Description
Disable Unused Accounts	<input checked="" type="checkbox"/>	Job to disable unused accounts
File System Clean	<input checked="" type="checkbox"/>	Job to process a volume deleting files that are no longer indexed (maybe the retention period has past or they have been deleted)
Node Status	<input checked="" type="checkbox"/>	Job to record status of node (CPU and Memory usage)
Proxy Aggregation	<input checked="" type="checkbox"/>	Job to pick up the data written by the proxy and store it in Stroom
Stream Processor	<input checked="" type="checkbox"/>	Job to process streams matching stream processor filters with their associated pipelines
Volume Status	<input checked="" type="checkbox"/>	Update the usage status of volumes owned by the node
Index Shard Retention	<input checked="" type="checkbox"/>	Job to delete index shards that have past their retention period
Pipeline Destination Roll	<input checked="" type="checkbox"/>	Roll any destinations based on their roll settings
Property Cache Reload	<input checked="" type="checkbox"/>	Reload properties in the cluster
SQL Stats Database Aggregation	<input checked="" type="checkbox"/>	Run SQL stats database aggregation
SQL Stats In Memory Flush	<input checked="" type="checkbox"/>	SQL Stats In Memory Flush (Cache to DB)
Stream Attributes Retention	<input checked="" type="checkbox"/>	Delete attributes older than system property stroom.streamAttribute.deleteAge)
Stream Delete	<input checked="" type="checkbox"/>	Physically delete streams that have been logically deleted based on age of delete (stroom.stream.deletePurgeAge)
Stream Retention	<input checked="" type="checkbox"/>	Job to delete data that has past it's retention period

Job	Node	Type	Max	Cur	Last Executed	Enabled
Proxy Aggregation	stroomp00	Cron 0,10,20,30,40,50 * * ?	0	2017-01-14T06:30:05.708Z	<input checked="" type="checkbox"/>	
Proxy Aggregation	stroomp01	Cron 0,10,20,30,40,50 * * ?	0	2017-01-14T06:30:03.111Z	<input checked="" type="checkbox"/>	

[Stroom UI Jobs Management - Proxy Aggregation Job](#)

At this, uncheck the Enabled check-boxes for both nodes and also the main Proxy Aggregation check-box to see.

Job	Enabled	Description
Disable Unused Accounts	<input checked="" type="checkbox"/>	Job to disable unused accounts
File System Clean	<input checked="" type="checkbox"/>	Job to process a volume deleting files that are no longer indexed (maybe the retention period has past or they have been deleted)
Node Status	<input checked="" type="checkbox"/>	Job to record status of node (CPU and Memory usage)
Proxy Aggregation	<input type="checkbox"/>	Job to pick up the data written by the proxy and store it in Stroom
Stream Processor	<input checked="" type="checkbox"/>	Job to process streams matching stream processor filters with their associated pipelines
Volume Status	<input checked="" type="checkbox"/>	Update the usage status of volumes owned by the node
Index Shard Retention	<input checked="" type="checkbox"/>	Job to delete index shards that have past their retention period
Pipeline Destination Roll	<input checked="" type="checkbox"/>	Roll any destinations based on their roll settings
Property Cache Reload	<input checked="" type="checkbox"/>	Reload properties in the cluster
SQL Stats Database Aggregation	<input checked="" type="checkbox"/>	Run SQL stats database aggregation
SQL Stats In Memory Flush	<input checked="" type="checkbox"/>	SQL Stats In Memory Flush (Cache to DB)
Stream Attributes Retention	<input checked="" type="checkbox"/>	Delete attributes older than system property stroom.streamAttribute.deleteAge)
Stream Delete	<input checked="" type="checkbox"/>	Physically delete streams that have been logically deleted based on age of delete (stroom.stream.deletePurgeAge)
Stream Retention	<input checked="" type="checkbox"/>	Job to delete data that has past it's retention period

Job	Node	Type	Max	Cur	Last Executed	Enabled
Proxy Aggregation	stroomp00	Cron 0,10,20,30,40,50 * * ?	0	2017-01-14T06:30:05.708Z	<input type="checkbox"/>	
Proxy Aggregation	stroomp01	Cron 0,10,20,30,40,50 * * ?	0	2017-01-14T06:30:03.111Z	<input type="checkbox"/>	

[Stroom UI Jobs Management - Proxy Aggregation Job Off](#)

At this point, no new proxy aggregation will occur and any inbound files received by the Store Proxies will accumulate in the proxy storage area.

4.7.3.3.1 Turn On Proxy Aggregation

We first select the Monitoring item of the **Main Menu** to bring up the Monitoring sub-menu.



[Stroom UI Monitoring sub-menu](#)

then move down and select the `Jobs` sub-item then select the `Proxy Aggregation Job` to be presented with the `Jobs` configuration tab as seen below.

The screenshot shows the Stroom UI Jobs Management interface. At the top, there are tabs for 'Welcome' and 'Jobs'. The 'Jobs' tab is selected, indicated by a blue bar and a highlighted 'Jobs' icon.

System Jobs:

Job	Enabled	Description
Disable Unused Accounts	<input checked="" type="checkbox"/>	Job to disable unused accounts
File System Clean	<input checked="" type="checkbox"/>	Job to process a volume deleting files that are no longer indexed (maybe the retention period has past or they have been deleted)
Node Status	<input checked="" type="checkbox"/>	Job to record status of node (CPU and Memory usage)
Proxy Aggregation	<input type="checkbox"/>	Job to pick up the data written by the proxy and store it in Stroom
Stream Processor	<input checked="" type="checkbox"/>	Job to process streams matching stream processor filters with their associated pipelines
Volume Status	<input checked="" type="checkbox"/>	Update the usage status of volumes owned by the node
Index Shard Retention	<input checked="" type="checkbox"/>	Job to delete index shards that have past their retention period
Pipeline Destination Roll	<input checked="" type="checkbox"/>	Roll any destinations based on their roll settings
Property Cache Reload	<input checked="" type="checkbox"/>	Reload properties in the cluster
SQL Stats Database Aggregation	<input checked="" type="checkbox"/>	Run SQL stats database aggregation
SQL Stats In Memory Flush	<input checked="" type="checkbox"/>	SQL Stats In Memory Flush (Cache to DB)
Stream Attributes Retention	<input checked="" type="checkbox"/>	Delete attributes older than system property stroom.streamAttribute.deleteAge)
Stream Delete	<input checked="" type="checkbox"/>	Physically delete streams that have been logically deleted based on age of delete (stroom.stream.deletePurgeAge)
Stream Retention	<input type="checkbox"/>	Job to delete data that has past it's retention period

Scheduled Jobs:

Job	Node	Type	Max	Cur	Last Executed	Enabled
Proxy Aggregation	stroomp00	Cron 0,10,20,30,40,50 * * ?	0	0	2017-01-14T06:30:05.708Z	<input type="checkbox"/>
Proxy Aggregation	stroomp01	Cron 0,10,20,30,40,50 * * ?	0	0	2017-01-14T06:30:03.111Z	<input type="checkbox"/>

[Stroom UI Jobs Management - Proxy Aggregation Job Off](#)

Now, re-enable each node's `Proxy Aggregation` check-box and the main `Proxy Aggregation` check-box.

After checking the check-boxes, perform a refresh of the display by pressing the `Refresh` icon



[Stroom UI RefreshButton](#)

on the top right of the lower (node display) pane. You should note the `Last Executed` date/time change to see

Job	Enabled	Description
Disable Unused Accounts	<input checked="" type="checkbox"/>	Job to disable unused accounts
File System Clean	<input checked="" type="checkbox"/>	Job to process a volume deleting files that are no longer indexed (maybe the retention period has past or they have been deleted)
Node Status	<input checked="" type="checkbox"/>	Job to record status of node (CPU and Memory usage)
Proxy Aggregation	<input checked="" type="checkbox"/>	Job to pick up the data written by the proxy and store it in Stroom
Stream Processor	<input checked="" type="checkbox"/>	Job to process streams matching stream processor filters with their associated pipelines
Volume Status	<input checked="" type="checkbox"/>	Update the usage status of volumes owned by the node
Index Shard Retention	<input checked="" type="checkbox"/>	Job to delete index shards that have past their retention period
Pipeline Destination Roll	<input checked="" type="checkbox"/>	Roll any destinations based on their roll settings
Property Cache Reload	<input checked="" type="checkbox"/>	Reload properties in the cluster
SQL Stats Database Aggregation	<input checked="" type="checkbox"/>	Run SQL stats database aggregation
SQL Stats In Memory Flush	<input checked="" type="checkbox"/>	SQL Stats In Memory Flush (Cache to DB)
Stream Attributes Retention	<input checked="" type="checkbox"/>	Delete attributes older than system property stroom.streamAttribute.deleteAge)
Stream Delete	<input checked="" type="checkbox"/>	Physically delete streams that have been logically deleted based on age of delete (stroom.stream.deletePurgeAge)
Stream Retention	<input type="checkbox"/>	Job to delete data that has past it's retention period

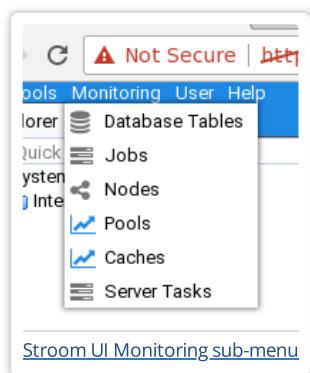
Job	Node	Type	Max	Cur	Last Executed	Enabled
Proxy Aggregation	stroomp00	Cron 0,10,20,30,40,50 * * *	0	0	2017-01-14T07:58:59.101Z	<input checked="" type="checkbox"/>
Proxy Aggregation	stroomp01	Cron 0,10,20,30,40,50 * * *	0	0	2017-01-14T06:30:03.111Z	<input checked="" type="checkbox"/>

[Stroom UI Test Feed - Re-enable Proxy Aggregation](#)

4.7.3.4 Stream Processors

4.7.3.4.1 Enable Stream Processors

To enable the Stream Processors task, move to the Monitoring item of the **Main Menu** and select it to bring up the Monitoring sub-menu.



[Stroom UI Monitoring sub-menu](#)

then move down and select the `Jobs` sub-item to be presented with the `Jobs` configuration tab as seen below.

Job	Enabled	Description
Disable Unused Accounts	<input checked="" type="checkbox"/>	Job to disable unused accounts
File System Clean	<input checked="" type="checkbox"/>	Job to process a volume deleting files that are no longer indexed (maybe the retention period has past or they have been deleted)
Node Status	<input checked="" type="checkbox"/>	Job to record status of node (CPU and Memory usage)
Proxy Aggregation	<input checked="" type="checkbox"/>	Job to pick up the data written by the proxy and store it in Stroom
Stream Processor	<input type="checkbox"/>	Job to process streams matching stream processor filters with their associated pipelines
Volume Status	<input checked="" type="checkbox"/>	Update the usage status of volumes owned by the node
Index Shard Retention	<input checked="" type="checkbox"/>	Job to delete index shards that have past their retention period

[Stroom UI Jobs Management - management tab](#)

At this, we select the `Stream Processor` `Job` whose check-box is not selected and the tab will show the individual Stroom Processor nodes in the Stroom deployment.

Job	Enabled	Description
Disable Unused Accounts	<input checked="" type="checkbox"/>	Job to disable unused accounts
File System Clean	<input checked="" type="checkbox"/>	Job to process a volume deleting files that are no longer indexed (maybe the retention period has past or they have been deleted)
Node Status	<input checked="" type="checkbox"/>	Job to record status of node (CPU and Memory usage)
Proxy Aggregation	<input checked="" type="checkbox"/>	Job to pick up the data written by the proxy and store it in Stroom
Stream Processor	<input type="checkbox"/>	Job to process streams matching stream processor filters with their associated pipelines
Volume Status	<input checked="" type="checkbox"/>	Update the usage status of volumes owned by the node
Index Shard Retention	<input checked="" type="checkbox"/>	Job to delete index shards that have past their retention period
Pipeline Destination Roll	<input checked="" type="checkbox"/>	Roll any destinations based on their roll settings
Property Cache Reload	<input checked="" type="checkbox"/>	Reload properties in the cluster
SQL Stats Database Aggregation	<input checked="" type="checkbox"/>	Run SQL stats database aggregation
SQL Stats In Memory Flush	<input checked="" type="checkbox"/>	SQL Stats In Memory Flush (Cache to DB)
Stream Attributes Retention	<input checked="" type="checkbox"/>	Delete attributes older than system property stroom.streamAttribute.deleteAge)
Stream Delete	<input checked="" type="checkbox"/>	Physically delete streams that have been logically deleted based on age of delete (stroom.stream.deletePurgeAge)
Stream Retention	<input checked="" type="checkbox"/>	Job to delete data that has past it's retention period

Job	Node	Type	Max	Cur	Last Executed	Enabled
Stream Processor	stroomp00	Distributed	20	0		<input type="checkbox"/>
Stream Processor	stroomp01	Distributed	20	0		<input type="checkbox"/>

We enable nodes nodes by selecting their check boxes as well as the main Stream Processors check box. Do so.

[Stroom UI Jobs Management - Stream Processor](#)

Job	Enabled	Description
Proxy Aggregation	<input checked="" type="checkbox"/>	Job to pick up the data written by the proxy and store it in Stroom
Stream Processor	<input checked="" type="checkbox"/>	Job to process streams matching stream processor filters with their associated pipelines
Volume Status	<input checked="" type="checkbox"/>	Update the usage status of volumes owned by the node
Index Shard Retention	<input checked="" type="checkbox"/>	Job to delete index shards that have past their retention period
Pipeline Destination Roll	<input checked="" type="checkbox"/>	Roll any destinations based on their roll settings
Property Cache Reload	<input checked="" type="checkbox"/>	Reload properties in the cluster
SQL Stats Database Aggregation	<input checked="" type="checkbox"/>	Run SQL stats database aggregation
SQL Stats In Memory Flush	<input checked="" type="checkbox"/>	SQL Stats In Memory Flush (Cache to DB)
Stream Attributes Retention	<input checked="" type="checkbox"/>	Delete attributes older than system property stroom.streamAttribute.deleteAge)
Stream Delete	<input checked="" type="checkbox"/>	Physically delete streams that have been logically deleted based on age of delete (stroom.stream.deletePurgeAge)
Stream Retention	<input checked="" type="checkbox"/>	Job to delete data that has past it's retention period

Job	Node	Type	Max	Cur	Last Executed	Enabled
Stream Processor	stroomp00	Distributed	20	0		<input checked="" type="checkbox"/>
Stream Processor	stroomp01	Distributed	20	0		<input checked="" type="checkbox"/>

[Stroom UI Jobs Management - Stream Processor enabled](#)

That is it. Stroom will automatically take note of these changes and internally start each node's Stream Processor task.

4.7.3.4.2 Enable Stream Processors On New Node

When one expands a Multi Node Stroom cluster deployment, after the installation of the Stroom Proxy and Application software and services on the new node, we need to enable it's Stream Processors task.

To enable the Stream Processors for this new node, move to the Monitoring item of the **Main Menu** and select it to bring up the Monitoring sub-menu.



then move down and select the Jobs sub-item to be presented with the Jobs configuration tab as seen below.

Job	Enabled	Description
Disable Unused Accounts	<input checked="" type="checkbox"/>	Job to disable unused accounts
File System Clean	<input checked="" type="checkbox"/>	Job to process a volume deleting files that are no longer indexed (maybe the retention period has past or they have been deleted)
Node Status	<input checked="" type="checkbox"/>	Job to record status of node (CPU and Memory usage)
Proxy Aggregation	<input checked="" type="checkbox"/>	Job to pick up the data written by the proxy and store it in Stroom
Stream Processor	<input type="checkbox"/>	Job to process streams matching stream processor filters with their associated pipelines
Volume Status	<input checked="" type="checkbox"/>	Update the usage status of volumes owned by the node
Index Shard Retention	<input checked="" type="checkbox"/>	Job to delete index shards that have past their retention period

[Stroom UI Jobs Management - management tab](#)

At this we select the Stream Processor Job whose check-box is selected

Job	Enabled	Description
Disable Unused Accounts	<input checked="" type="checkbox"/>	Job to disable unused accounts
File System Clean	<input checked="" type="checkbox"/>	Job to process a volume deleting files that are no longer indexed (maybe the retention period has past or they have been deleted)
Node Status	<input checked="" type="checkbox"/>	Job to record status of node (CPU and Memory usage)
Proxy Aggregation	<input checked="" type="checkbox"/>	Job to pick up the data written by the proxy and store it in Stroom
Stream Processor	<input checked="" type="checkbox"/>	Job to process streams matching stream processor filters with their associated pipelines
Volume Status	<input checked="" type="checkbox"/>	Update the usage status of volumes owned by the node
Index Shard Retention	<input checked="" type="checkbox"/>	Job to delete index shards that have past their retention period
Pipeline Destination Roll	<input checked="" type="checkbox"/>	Roll any destinations based on their roll settings
Property Cache Reload	<input checked="" type="checkbox"/>	Reload properties in the cluster
SQL Stats Database Aggregation	<input checked="" type="checkbox"/>	Run SQL stats database aggregation
SQL Stats In Memory Flush	<input checked="" type="checkbox"/>	SQL Stats In Memory Flush (Cache to DB)
Stream Attributes Retention	<input checked="" type="checkbox"/>	Delete attributes older than system property stroom.streamAttribute.deleteAge)
Stream Delete	<input checked="" type="checkbox"/>	Physically delete streams that have been logically deleted based on age of delete (stroom.stream.deletePurgeAge)
Stream Retention	<input type="checkbox"/>	Job to delete data that has past it's retention period

Job	Node	Type	Max	Cur	Last Executed	Enabled
Stream Processor	stroomp00	Distributed	20	0		<input checked="" type="checkbox"/>
Stream Processor	stroomp01	Distributed	20	0		<input checked="" type="checkbox"/>
Stream Processor	stroomp02	Distributed	20	0		<input type="checkbox"/>

[Stroom UI Jobs Management - Stream Processor new node](#)

We enable the new node by selecting it's check-box.

Job	Enabled	Description
Disable Unused Accounts	<input checked="" type="checkbox"/>	Job to disable unused accounts
File System Clean	<input checked="" type="checkbox"/>	Job to process a volume deleting files that are no longer indexed (maybe the retention period has past or they have been deleted)
Node Status	<input checked="" type="checkbox"/>	Job to record status of node (CPU and Memory usage)
Proxy Aggregation	<input checked="" type="checkbox"/>	Job to pick up the data written by the proxy and store it in Stroom
Stream Processor	<input checked="" type="checkbox"/>	Job to process streams matching stream processor filters with their associated pipelines
Volume Status	<input checked="" type="checkbox"/>	Update the usage status of volumes owned by the node
Index Shard Retention	<input checked="" type="checkbox"/>	Job to delete index shards that have past their retention period
Pipeline Destination Roll	<input checked="" type="checkbox"/>	Roll any destinations based on their roll settings
Property Cache Reload	<input checked="" type="checkbox"/>	Reload properties in the cluster
SQL Stats Database Aggregation	<input checked="" type="checkbox"/>	Run SQL stats database aggregation
SQL Stats In Memory Flush	<input checked="" type="checkbox"/>	SQL Stats In Memory Flush (Cache to DB)
Stream Attributes Retention	<input checked="" type="checkbox"/>	Delete attributes older than system property stroom.streamAttribute.deleteAge)
Stream Delete	<input checked="" type="checkbox"/>	Physically delete streams that have been logically deleted based on age of delete (stroom.stream.deletePurgeAge)
Stream Retention	<input checked="" type="checkbox"/>	Job to delete data that has past it's retention period

Job	Node	Type	Max	Cur	Last Executed	Enabled
Stream Processor	stroomp00	Distributed	20	0		<input checked="" type="checkbox"/>
Stream Processor	stroomp01	Distributed	20	0		<input checked="" type="checkbox"/>
Stream Processor	stroomp02	Distributed	20	0		<input checked="" type="checkbox"/>

[Stroom UI Jobs Management - Stream Processor enabled on new node](#)

5 - Sending Data to Stroom

How to send data (event logs) into Stroom or one of its proxies.

Stroom and Stroom Proxy have a simple HTTP POST interface that requires HTTP header arguments to be supplied as described [here](#).

Files are posted to Stroom and Stroom Proxy as described [here](#).

Stroom will return a response code indicating the success or failure status of the post as described [here](#)

Data can be sent from many operating systems or applications. Some examples to aid in sending data can be found [here](#)

5.1 - Example Clients

A collection of example client applications for sending data to Stroom or one of its proxies.

The following article provides examples to help data providers send data to Stroom via the HTTPS interface. The code for the clients is in the [stroom-clients](#) repository [stroom-clients \(external link\)](#).

5.1.1 - curl (Linux)

How to use the `curl` command to send data to Stroom.

Curl is a standard unix tool to send data to or from a server. In the following examples -H is used to specify the header arguments required by Stroom, see [Header Arguments](#).

Notes:

- The @ character must be used in front of the file being posted. If it is not then curl will post the file name instead of its contents.
- The -data-binary argument must always be used even for text formats, in order to prevent data corruption by curl stripping out newlines.

5.1.1.1 Example HTTPS post without authentication:

```
curl -k --data-binary @file.dat "https://<Stroom_HOST>/stroom/datafeed"  
-H "Feed:EXAMPLE_FEED"  
-H "System:EXAMPLE_SYSTEM"  
-H "Environment:EXAMPLE_ENVIRONMENT"
```

In the above example -k is required to stop curl from authenticating the server. The next example must be used to supply the necessary CA to authenticate the server if this is required.

5.1.1.2 Example HTTPS With 1 way SSL authentication:

```
curl --cacert root_ca.crt --data-binary @file.dat "https://<Stroom_HOST>/stroom/datafeed"  
-H "Feed:EXAMPLE_FEED"  
-H "System:EXAMPLE_SYSTEM"  
-H "Environment:EXAMPLE_ENVIRONMENT"
```

The above example verifies that the certificate presented by Stroom is signed by the CA. The CA is provided to curl using the '--cacert root_ca.crt' parameter.

For step by step instructions for creating, configuring and testing the PKI authentication, see the [SSL Guide](#)

5.1.1.3 Example HTTPS With 2 way SSL authentication:

```
curl --cert example.pem --cacert root_ca.crt --data-binary @file.dat "https://<Stroom_HOST>/stroom/datafeed"  
-H "Feed:EXAMPLE_FEED"  
-H "System:EXAMPLE_SYSTEM"  
-H "Environment:EXAMPLE_ENVIRONMENT"
```

The above example both verifies that the certificate presented by Stroom is signed by the CA and also provides a certificate to authenticate itself with Stroom. The data provider provides a certificate using the '--cert example.pem' parameter.

If your input file is not compressed you should compress it as follows:

```
gzip -c uncompressedfile.dat | curl --cert example.pem --cacert root_ca.crt --data-binary @- "https://<Stroom_HOST>/stroom/data"
-H "Feed:EXAMPLE_FEED"
-H "System:EXAMPLE_SYSTEM"
-H "Environment:EXAMPLE_ENVIRONMENT"
-H "Compression:Gzip"
```

When delivering data from a RHEL4 host, an additional header argument must be added to specify the FQDN of the host:

```
-H "Hostname:host.being.audited"
```

The hostname being sent as a header argument may be resolved upon execution using the command `hostname -f`.

5.1.1.4 SSL Notes

To create a .pem format key simply append the private key and certificate.

```
cat <NAME>.key >> <NAME>.pem
cat <NAME>.crt >> <NAME>.pem
```

To remove the pass phrase from a openssl private key use.

```
openssl rsa -in server.key -out server-clear.key
```

The `send-logs.sh` script assumes the period start and end times are embedded in the file name (e.g. `log_2010-01-01T12:00:00.000Z_2010-01-02T12:00:00.000Z.log`). The certificates will need to be added to the script as above.

5.1.2 - curl (Windows)

Using Curl on Windows to send data to Stroom.

There is a version of curl for Windows

Windows 10 is the latest desktop OS offering from Microsoft. From Windows 10 build 17063 and later, curl is now natively included - you can execute it directly from Cmd.exe or PowerShell.exe. Curl.exe is located at c:\windows\system32 (which is included in the standard PATH environment variable) - all you need to do is run Command Prompt with administrative rights and you can use Curl. You can execute it directly from Cmd.exe or PowerShell.exe. For older versions of Windows, the cURL project has Windows binaries.

```
curl -s -k --data-binary @file.dat "https://stroomp.strmdev00.org/stroom/datafeed" -H"Feed:TEST-FEED-V1_0" -H"System:EXAMPLE_SYS
```

```
C:\Temp>curl  
curl: try 'curl --help' for more information  
  
C:\Temp>curl -s -k --data-binary @file.dat "https://stroomp.strmdev00.org/stroom/datafeed"  
-H "Feed:TEST-FEED-V1_0" -H"System:EXAMPLE_SYSTEM" -H"Environment:EXAMPLE_ENVIRONMENT"  
C:\Temp>
```

[Windows curl CLI](#)

5.1.3 - event-logging (Java library)

A Java library for logging events in Java applications.

[event-logging \(external link\)](#) is a Java API for logging audit events conforming to the [Event Logging XML Schema \(external link\)](#). The API uses a generated Java JAXB model of the *Event Logging XML Schema*. *Event Logging* can be incorporated into your Java application to provide a means of recording and outputting audit events or user actions for compliance, security or monitoring.

This library only generates the events. By default XML events are written to a file using a logging appender. In order to send the events to Stroom either the logged files will need to be sent to stroom using one of the other [clients](#).

5.1.4 - send_to_stroom.sh (Linux)

A shell script for sending logs to Stroom or one of its proxies

[send_to_stroom.sh \(external link\)](#) is a small bash script to make it easier to send data to *stroom*. To use it download the following files using wget or similar, replacing `SEND_TO_STROOM_VER` with the latest released version from [here \(external link\)](#):

```
SEND_TO_STROOM_VER="send-to-stroom-v2.0" && \
wget "https://raw.githubusercontent.com/gchq/stroom-clients/${SEND_TO_STROOM_VER}/bash/send_to_stroom.sh" && \
wget "https://raw.githubusercontent.com/gchq/stroom-clients/${SEND_TO_STROOM_VER}/bash/send_to_stroom_args.sh" && \
chmod u+x send_to_stroom*.sh
```

To see the help for *send_to_stroom.sh*, enter `./send_to_stroom.sh --help`

The following is an example of using *send_to_stroom.sh* to send all logs in a directory:

```
./send_to_stroom.sh \
--delete-after-sending \
--file-regex ".*/access-[0-9]+.*\log(\.gz)?$" \
--key ./client..key \
--cert ./client.pem.crt \
--cacert ./ca.pem.crt \
/some_directory/logs \
MY_FEED \
MY_SYSTEM \
DEV \
https://stroom-host/stroom/datafeed
```

5.1.5 - Simple C# Client

A simple C# client for sending data files to Stroom.

The `StroomCSharpClient` is a C# port of the Java client and behaves in the same way. Note that this is just an example, not a fully functional client. See [StroomCSharpClient \(external link\)](#).

5.1.6 - Simple Java Client

A simple Java client for sending data files to Stroom.

The `stroom-java-client` provides an example Java client that can:

- Read a zip, gzip or uncompressed an input file.
- Perform a HTTP post of data with zip, gzip or uncompressed compression.
- Pass down arguments on the command line as HTTP request arguments.
- Supports HTTP and HTTPS with 1 or 2 way authentication.

(*N.B. arguments must be in lower case*).

To use the example client first compile the Java code:

```
javac DataFeedClient.java
```

5.1.6.1 Example HTTP Post:

```
java -classpath . DataFeedClient inputfile=datafeed url=http://<Stroom_HOST>/stroom/datafeed system=EXAMPLE-SYSTEM environment=DE
```

5.1.6.2 Example HTTPS With 1 way SSL authentication:

```
java -classpath . -Djavax.net.ssl.trustStore=ca.jks -Djavax.net.ssl.trustStorePassword=capass DataFeedClient inputfile=datafeed
```

5.1.6.3 Example HTTPS With 2 way SSL authentication:

```
java -classpath . -Djavax.net.ssl.trustStore=ca.jks -Djavax.net.ssl.trustStorePassword=capass -Djavax.net.ssl.keyStore=example.j
```

5.1.7 - stroom-log-sender (Docker)

A Docker image for periodically sending log files generated by an application to Stroom.

[stroom-log-sender \(external link\)](#) is a small Docker image for sending data to *Stroom*.

This is the simplest way to get data into stroom if the data provider is itself running in docker. It can also be used for sending data to *Stroom* from data providers that are not running in Docker. *stroom-log-sender* makes use of the *send_to_stroom.sh* bash script that is described below. For details on how to use *stroom-log-sender*, see the Dockerhub link above.

5.1.8 - VBScript (Windows)

Using VBScript to send data to Stroom.

`extract-data.vbs` uses `wEvtutil.exe` to extract **Security** event information from the windows event log. This script has been tested on Windows 2008.

This script is designed to run periodically (say every 10 minutes). The first time the script is run it stores the current time in UTC format in the registry. Subsequent calls then extract event information from the last run time to the new current time. The events are stored in a zip file with the period dates embedded.

The script requires a working directory used as a buffer for the zip files. This can be set at the start of the script otherwise it will default to the working directory.

The `send-data.vbs` script is designed to run periodically (say every 10 minutes). The script will scan for zip files and send them to Stroom.

The script details several parameters that require setting per environment. Among these are the working directory that the zip files are stored in, the feed name and the URL of Stroom.

5.1.8.1 SSL

To send data over SSL (https) you must import a client certificate in p12 format into windows. To convert a certificate (.crt) and private key (.key) into a p12 format use the following command:

```
openssl pkcs12 -export -in <NAME>.crt -inkey <NAME>.key -out <NAME>.p12 -name "<NAME>"
```

Once in p12 format use the windows certificate wizard to import the public private key.

The `send-data-tree.vbs` script works through a directory for different feed types.

5.1.9 - wget (Windows)

Using **wget** on Windows to send data to Stroom.

There is a version of `wget` for windows

- Use `--post-file` argument to supply the data
- Use `--certificate` and `--certificate-type` arguments to specify your client certificate
- Use `--header` argument to inform Stroom which feed and environment your data relates to

5.2 - Header Arguments

The various HTTP headers that can be sent with data.

The following data must be passed in as HTTP header arguments when sending files to Stroom via HTTP POST. These arguments are case insensitive.

- **System** - The name by which the system is known within the organisation, e.g. `PAYROLL_SYSTEM`. This could be the name of a project/service or capability.
- **Environment** - A means to identify the deployed instance of a system. This may indicate the deployment status, e.g. `DEV`, `REF`, `LIVE`, `OPS`, etc., and/or the location where the instance is deployed. An environment may be a combination of these attributes separated with an underscore.
- **Feed** - The name of the feed this data relates to. This is mandatory and must match a feed defined within Stroom in order for Stroom to accept the data and know what to do with it.
- **Compression** - This token is optionally used when the POST payload is compressed with either gzip or zip compression. Value of `ZIP` and `GZIP` are valid. **Note:** The `Compression` token MUST not be used in conjunction with the standard HTTP header token `Content-Encoding` otherwise stroom will be unable to un-compress the data. Use either `Compression:GZIP` or `Content-Encoding:gzip`, not both. Using `Compression` is preferred.
- **EffectiveTime** - This is only applicable to reference data. It is used to indicate the point in time that the reference data is applicable to, i.e. all event data that uses the reference data that is created after the effective time will use the reference data until a new reference data item arrives with a later effective time. **Note:** This argument must be in *ISO 8601* date time format, i.e: `yyyy-MM-ddTHH:mm:ss.sssZ`.

Example header arguments for a feed called `MY_SYSTEM-EVENTS` from system `MY_SYSTEM` and environment `OPS`

```
System:MY_SYSTEM
Environment:OPS
Feed:MY_SYSTEM-EVENTS
```

The post payload must contain the events file. If the compression format is ZIP the payload must contain ZIP entries with the events files and optional context files ending in .ctx. Further details of supported payload formats can be found [here](#).

5.3 - Java Keystores

How to create java key/trust stores for use with Java client applications.

There are many times when you may wish to create a Java keystore from certificates and keys and vice versa. This guide aims to explain how this can be done.

5.3.1 Import

If you need to create a Java keystore from a .crt and .key then this is how to do it.

5.3.1.1 Convert your keys to der format

```
openssl x509 -in <YOUR KEY>.crt -inform PEM -out <YOUR KEY>.crt.der -outform DER  
openssl pkcs8 -topk8 -nocrypt -in <YOUR KEY>.key -inform PEM -out <YOUR KEY>.key.der -outform DER
```

5.3.1.2 ImportKey

Use the `ImportKey` class in the `stroom-java-client` library to import keys.

For example:

```
java ImportKey keystore=<YOUR KEY>.jks keypass=<YOUR PASSWORD> alias=<YOUR KEY> keyfile=<YOUR KEY>.key.der certfile=<YOUR KEY>.crt  
keytool -import -noprompt -alias CA -file <CA CERT>.crt -keystore ca.jks -storepass ca
```

5.3.2 Export

5.3.2.1 ExportKey

Use the `ExportKey` class in the `stroom-java-client` library to export keys. If you would like to use curl or similar application but only have keys contained within a Java keystore then they can be exported.

For example:

```
java ExportKey keystore=<YOUR KEY>.jks keypass=<YOUR PASSWORD> alias=<YOUR KEY>
```

This will print both the key and certificate to standard out. This can then be copied into a PEM file for use with cURL or other similar application.

5.4 - Payloads

Description of the data formats for sending data into a Stroom instance.

Stroom can support multiple payload formats with different compression applied. However all data once uncompressed must be text and not binary.

Stroom can receive data in the following formats:

- Uncompressed - Text data is sent to Stroom and no compression flag is set in the header arguments.
- GZIP - Text data is GZIP compressed and the compression flag is set to GZIP.
- ZIP - A text file is compressed into a ZIP archive and sent to Stroom with the compression flag set to ZIP. The ZIP file must contain one data file and an optional context file, see below.

5.4.1 Context Files

ZIP files sent to Stroom are expected to contain the data file and an optional context file *.ctx. If provided a context file can be used to provide reference data that is specific to the data file that has been sent. Context data is supplementary information that is not contained within logged events, e.g. the machine name, ip address etc may be delivered in a context file if it is not written by an application in each logged event.

5.4.2 Character Encodings

Although Stroom only supports data in text format, text can be encoded using multiple character encodings. Supported encodings include:

- ISO-8859-1 (understood by default)
- Windows-1252 - ANSI (understood by default)
- ASCII (understood by default)
- UTF-8 (with or without BOM)
- UTF-16LE (little endian with or without BOM)
- UTF-16BE (big endian with or without BOM)
- UTF-32LE (little endian with or without BOM)
- UTF-32BE (big endian with or without BOM)

In order to tell Stroom what character encoding to use the feed that the data belongs to can be configured within the Stroom application to use a specific character encoding. Separate character encodings can be specified for logged event and context data.

5.5 - Response Codes

The HTTP response codes returned by stroom.

Stroom will return a HTTP response code to indicate success or failure. An additional response Header "Stroom-Status" will indicate a more precise error message code. A user readable message will appear in the response body.

HTTP Status	Stroom-Status	Message	Reason
200	0	OK	Post of data successful
406	100	Feed must be specified	You must provide Feed as a header argument in the request
406	110	Feed is not set to receive data	The feed you have provided is not setup to receive data (maybe does not exist or is set to reject)
406	200	Unknown compression	Compression argument must be one of ZIP, GZIP and NONE
401	300	Client Certificate Required	The feed you have provided requires a client HTTPS certificate to send data
403	310	Client Certificate not authorised	The feed you have provided does not allow your client certificate to send data
500	400	Compressed stream invalid	The stream of data sent does not form a valid compressed file. Maybe it terminated unexpectedly or is corrupt.
500	999	Unknown error	An unknown unexpected error occurred

In the event that data is not successfully received by Stroom, i.e. the response code is not 200, the client system should buffer data and keep trying to re-send it. Data should only be removed from the client system when it has been sent successfully.

5.6 - SSL Configuration

Configuring SSL with cURL.

This page provides a step by step guide to getting PKI authentication working correctly for Unix hosts so as to be able to sign deliveries from cURL.

First make sure you have a copy of your organisations CA certificate.

Check that the CA certificate works by running the following command:

```
echo "Test" | curl --cacert CA.crt --data-binary @- "https://<Stroom_HOST>/stroom/datafeed"
```

If the response starts with the line:

```
curl: (60) SSL certificate problem, verify that the CA cert is OK.
```

then you do not have the correct CA certificate.

If the response contains the line

```
HTTP Status 406 - Stroom Status 100 - Feed must be specified
```

then one-way SSL authentication using the CA certificate is successful.

The VBScript file to check windows certificates is `check-certs.vbs` (TODO link).

#Final Testing

Once one-way authentication has been tested, two-way authentication should be configured:

The server certificate and private key should be concatenated to create a PEM file:

```
cat hostname.cert hostname.key > hostname.pem
```

Finally, test for 2-way authentication:

```
echo "Test" | curl --cacert CA.crt --cert hostname.pem --data-binary @- "https://<Stroom_HOST>/stroom/datafeed"
```

If the response contains the line

```
HTTP Status 406 - Stroom Status 100 - Feed must be specified
```

then two-way SSL authentication is successful.

#Final Tidy Up

The files `ca.crt` and `hostname.pem` are the only files required for two-way authentication and should be stored permanently on the server; all other remaining files may be deleted or backed up if required.

#Certificate Expiry

PKI certificates expire after 2 years. To check the expiry date of a certificate, run the following command:

```
openssl x509 -in /path/to/certificate.pem -noout -enddate
```

This will give a response looking similar to:

```
notAfter=Aug 15 10:01:42 2013 GMT
```

6 - Stroom Proxy

Stroom Proxy acts as a proxy when sending data to a Stroom instance. Stroom Proxy has various modes such as storing, aggregating and forwarding the received data. Stroom Proxies can be used to forward to other Stroom Proxy instances.

6.1 - Apache Forwarding

Warning

This document refers to v5.

Stroom Proxy defaults to listening for HTTP on port 9080. It is recommended that Apache is used to listen on the standard HTTP port 80 and forward requests on via the Apache mod_jk module and the AJP protocol (on 9009). Apache can also perform HTTPS on port 443 and pass over requests to Tomcat using the same AJP protocol.

It is additionally recommended that Stroom Proxy is used to front data ingest and so Apache is configured to route traffic to `http(s)://server/stroom/datafeed` to Stroom Proxy.

6.1.1 Prerequisites

- `tomcat-connectors-1.2.31-src.tar.gz`

6.1.2 Setup Apache

- As root
- Patch mod_jk

```
cd ~/tmp
tar -xvzf tomcat-connectors-1.2.31-src.tar.gz
cd tomcat-connectors-1.2.31-src/native
./configure --with-apxs=/usr/sbin/apxs
make
sudo cp apache-2.0/mod_jk.so /etc/httpd/modules/
cd
```

- Put the web server cert, private key, and CA cert into the web servers conf directory `/etc/httpd/conf`. E.g.

```
[user@node1 stroom-doc]$ ls -al /etc/httpd/conf
...
-rw-r--r-- 1 root root 1729 Aug 27 2013 host.crt
-rw-r--r-- 1 root root 1675 Aug 27 2013 host.key
-rw-r--r-- 1 root root 1289 Aug 27 2013 CA.crt
...
```

- Make changes to `/etc/httpd/conf.d/ssl.conf` as per below

```
JkMount /stroom/datafeed* loadbalancer_proxy
JkMount /stroom* loadbalancer_proxy
```

```
JkOptions +ForwardKeySize +ForwardURICcompat +ForwardSSLCertChain -ForwardDirectories
```

```
SSLCertificateFile /etc/httpd/conf/[YOUR SERVER].crt  
SSLCertificateKeyFile /etc/httpd/conf/[YOUR SERVER].key  
SSLCertificateChainFile /etc/httpd/conf/[YOUR CA].crt  
SSLCACertificateFile /etc/httpd/conf/[YOUR CA APPENDED LIST].crt
```

- Remove /etc/httpd/conf.d/nss.conf to avoid a 8443 port clash

```
rm /etc/httpd/conf.d/nss.conf
```

- Create a /etc/httpd/conf.d/mod_jk.conf configuration

```
LoadModule jk_module modules/mod_jk.so  
JKWorkersFile conf/workers.properties  
JkLogFile logs/mod_jk.log  
JkLogLevel info  
JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"  
JkOptions +ForwardKeySize +ForwardURICcompat +ForwardSSLCertChain -ForwardDirectories  
JkRequestLogFormat "%w %V %T"
```

```
JkMount /stroom/datafeed* loadbalancer_proxy  
JkMount /stroom* loadbalancer_proxy
```

```
JkShmFile logs/jk.shm  
<Location /jkstatus/>  
    JkMount status  
    Order deny,allow  
    Deny from all  
    Allow from 127.0.0.1  
</Location>
```

- Setup stroom-setup/cluster.txt, generate the workers file and copy into Apache. (as root and replace stroomuser with your processing user)

```
/home/stroomuser/stroom-setup/workers.properties.sh --cluster=/home/stroomuser/cluster.txt > /etc/httpd/conf/workers.properties
```

- Inspect /etc/httpd/conf/workers.properties to make sure it looks as you expect for your cluster text

```
worker.list=loadbalancer_proxy,local_proxy  
worker.stroom_1_proxy.port=9009  
worker.stroom_1_proxy.host=localhost  
worker.stroom_1_proxy.type=ajp13  
worker.stroom_1_proxy.lbfactor=1  
worker.stroom_1_proxy.max_packet_size=65536  
....  
....  
worker.loadbalancer_proxy.type=lb  
worker.loadbalancer_proxy.balance_workers=stroom_1_proxy,stroom_2_proxy  
worker.loadbalancer_proxy.sticky_session=1  
worker.local_proxy.type=lb  
worker.local_proxy.balance_workers=stroom_1_proxy  
worker.local_proxy.sticky_session=1
```

- Create a simple redirect page to the stroom web app for the root URL (e.g. DocumentRoot "/var/www/html", index.html)

```
&lt;html&gt;&lt;head&gt;&lt;meta http-equiv="Refresh" content="0; URL=stroom"&gt;&lt;/head&gt;&lt;/html&gt;
```

- Restart Apache and then test default http / https access.

```
sudo /etc/init.d/httpd restart
```

6.2 - Running with docker

TODO

This document is out of date and needs updating to refer to the stroom-proxy docker stack.

6.2.1 Clone and build *stroom-proxy*

```
git clone https://github.com/gchq/stroom-proxy.git  
mvn clean install
```

6.2.2 Unpack the *stroom-proxy* distribution in preparation for building the docker iamge

```
cd stroom-proxy-distribution  
unzip target/stroom-proxy-distribution-<version>-bin.zip -d target
```

6.2.3 Building and running the docker image

Here you need to say where you want data to be sent by *stroom-proxy*. This is done using a [--build-arg](#) ([external link](#)) parameter of *STROOM_PROXY_TYPE*. These values can be `forward`, `store`, or `store_nodb`. See the *stroom-proxy* section in the [stroom documentation](#) ([external link](#)) documentation for more details about these options.

```
docker stop stroom-proxy  
docker rm stroom-proxy  
docker rmi stroom-proxy  
  
docker build --build-arg STROOM_PROXY_TYPE=store_nodb --tag=stroom-proxy:latest target/stroom-proxy  
docker run -p 8080:8080 --name=stroom-proxy stroom-proxy
```

6.3 - Stroom Proxy Installation

Warning

This document refers to v5.

6.3.1 Prerequisites

- Linux Server's with at least 4GB RAM
- Install files stroom-proxy-X-Y-Z-distribution.zip, stroom-deploy-X-Y-Z-distribution.zip
- Temporarily allow port 9080 if not relying on Apache Forwarding (see below)

6.3.2 Processing User Setup

See [Processing User Setup](#).

6.3.3 Installing Stroom Proxy

As the processing user unpack the stroom-proxy-X-Y-Z-distribution.zip installation files in the processing users home directory.

```
unzip stroom-proxy-X-Y-Z-distribution.zip
```

Stroom Proxy can be setup as follows:

- forward - as an aggregation point to store and forwarding onto another Stroom or Stroom / Proxy
- store - to front Stroom for data ingest

6.3.3.1 Stroom Proxy - forward

In forward mode you need to know the server address that data is being sent onto. Run the setup script to capture the basic settings required to run Stroom Proxy in forward mode.

- @@ NODE @@ - Each Stroom instance in the cluster needs a unique name, if this is a reinstall ensure you use the previous deployment. **This name needs match the name used in your worker.properties (e.g. 'node1' in the case 'node1.my.org')**
- @@ PORT PREFIX @@ - By default Stroom Proxy will run on port 9080

```
[stroomuser@node1 ~]$ ./stroom-proxy/bin/setup.sh forward
```

```
[stroomuser@dev1 ~]$ ./stroom-proxy/bin/setup.sh forward
```

...

Parameters

=====

```
@@NODE@@      : Unique node name for install [node1] : node1
@@PORT_PREFIX@@ : HTTP prefix to use      [90] : 90
@@REPO_DIR@@   : Stroom Proxy Repository Dir    [/stroomdata/stroom-proxy] : /home/
@@FORWARD_SERVER@@ : Server to forward data on to [hostname] : audit.my.org
@@JAVA_OPTS@@   : Optional tomcat JVM settings ["-Xms512m -Xmx1g"] : 
```

...

6.3.3.2 Stroom Proxy - store

In store mode you need to know the mysql details to validate incoming data with.

```
[stroomuser@node1 ~]$ ./stroom-proxy-app/bin/setup.sh store

...
@@NODE@@      : Unique node name for install          [node] :
@@PORT_PREFIX@@ : HTTP prefix to use                  [90] :
@@REPO_DIR@@   : Stroom Proxy Repository Dir          [/stroomdata/stroom-proxy] :
@@JDBC_CLASSNAME@@ : JDBC class name                 [com.mysql.jdbc.Driver] :
@@JDBC_URL@@   : JDBC URL (jdbc:mysql://[HOST]/[DBNAME]) [jdbc:mysql://localhost/stroom] :
@@DB_USERNAME@@ : Database username                   [ ] :
@@DB_PASSWORD@@ : Database password                  [ ] :
@@JAVA_OPTS@@   : Optional tomcat JVM settings        ["-Xms512m -Xmx1g"] :
```

6.3.3.3 Install Check

Start the installed instance:

```
./stroom-deploy/start.sh
```

Inspect the logs:

```
tail -f stroom-proxy-app/instance/logs/stroom.log
```

6.3.3.4 Stroom Proxy Properties

The following properties can be configured in the stroom.properties file.

TODO - Could do with a column indicating which proxy mode these properties apply to, e.g. store/forward

Property Name	Description
repoDir	Optional Repository DIR. If set any incoming request will be written to the file system.
logRequest	Optional log line with header attributes output as defined by this property
bufferSize	Override default (8192) JDK buffer size to use
forwardUrl	Optional The URL's to forward onto This is pass-through mode if repoDir is not set
forwardThreadCount	Number of threads to forward with
forwardTimeoutMs	Time out when forwarding
forwardChunkSize	Chunk size to use over http(s) if not set requires buffer to be fully loaded into memory
rollCron	Interval to roll any writing repositories.
readCron	Cron style interval (e.g. every hour '0 * *', every half hour '0,30 * *') to read any ready repositories (if not defined we read all the time).
maxAggregation	Aggregate size to break at when building an aggregate.
zipFilenameDelimiter	The delimiter used to separate the id in the zip filename from the templated part

Property Name	Description
zipFilenameTemplate	A template for naming the zip files in the repository where files will be named nnn!zipFilenameTemplate.zip where nnn is the id prefix, ! is the configurable delimiter and zipFilenameTemplate will be something like '\${feed}!\${headerMapKey1}!\${headerMapKey2}'. The name of each variable must exactly match a key in the meta data else it will resolve to ''.
requestDelayMs	Sleep time used to aid with testing
forwardDelayMs	Debug setting to add a delay
dbRequestValidatorContext	Database Feed Validator - Data base JDBC context
dbRequestValidatorJndiName	Database Feed Validator - Data base JDBC JNDI name
dbRequestValidatorFeedQuery	Database Feed Validator - SQL to check feed status
dbRequestValidatorAuthQuery	Database Feed Validator - SQL to check authorisation required
remotingUrl	Url to use for remoting services
remotingReadTimeoutMs	Change from the default JVM settings.
remotingConnectTimeoutMs	Change from the default JVM settings.
maxStreamSize	Stream size to break at when building an aggregate.
maxFileScan	Max number of files to scan over during forwarding. Once this limit is hit it will wait until next read interval
cacheTimeToIdleSeconds	Time to idle settings to used for validating feed information
cacheTimeToLiveSeconds	Time to live settings to used for validating feed information

6.3.4 Apache Forwarding

See [Apache Forwarding](#).

6.3.5 Java Key Store Setup

If you require that Stroom Proxy communicates over 2-way https you will need to set up Java Key Stores.

See [Java Key Store Setup](#).

6.3.6 Securing Stroom

See [Securing Stroom](#).