

# SIDDHI VISUAL EDITOR

Design and Manipulation

## SIDDHI VISUAL EDITOR

Elucidating the individual element manipulations, property settings and constraints existing in the current version

Manuela Nayantara Jeyaraj

CEP Intern

Sri Lanka Institute of Information Technology

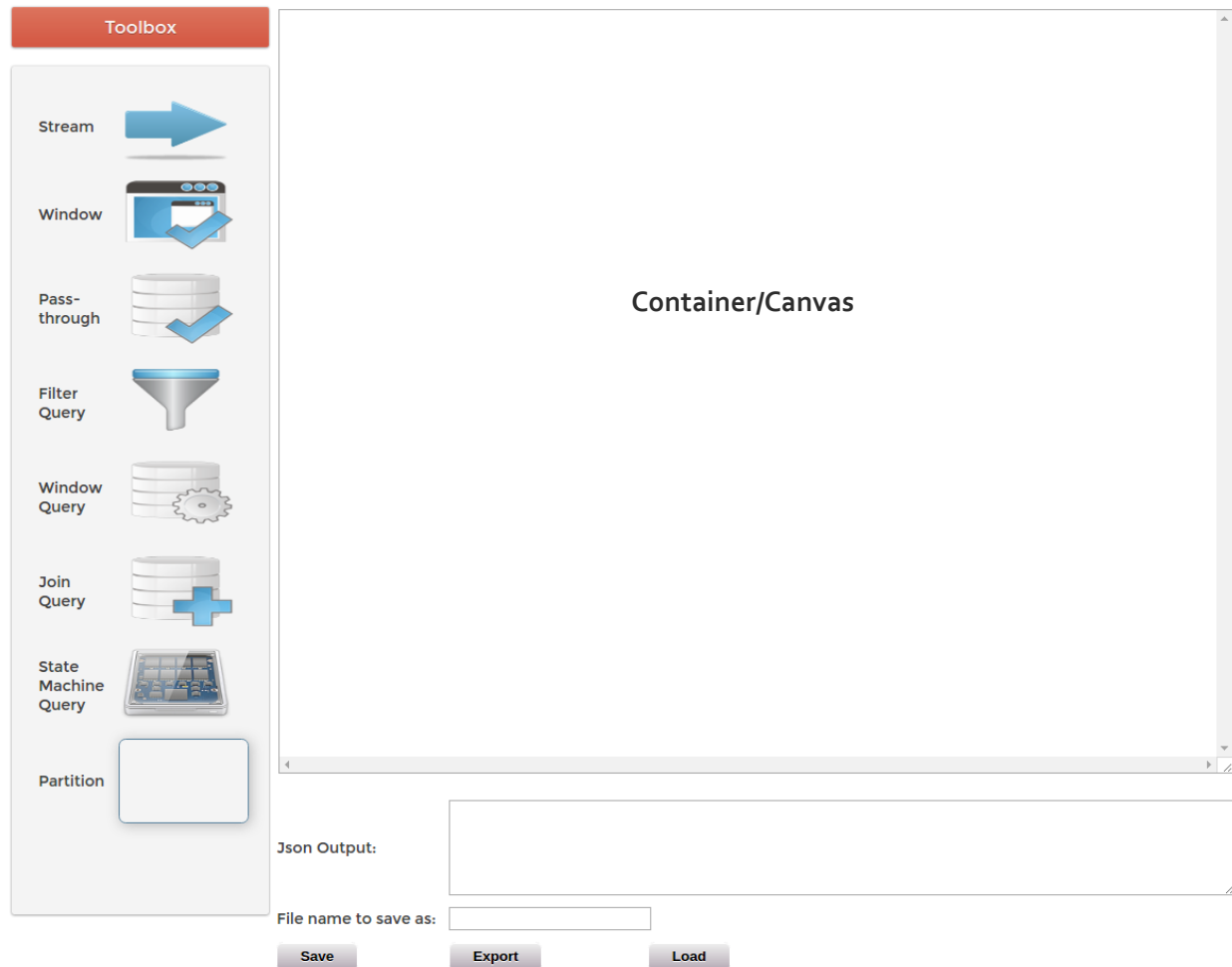
05/01/2016 – 10/31/2016

## CONTENTS

INITIAL VIEW .....	2
1. MANIPULATING STREAMS .....	2
1.1. IMPORT STREAMS .....	3
1.2. EXPORT STREAMS.....	5
1.3. DEFINED STREAMS.....	6
2. MANIPULATING WINDOWS.....	9
2.1. DERIVED WINDOW .....	9
2.2. DEFINED WINDOW .....	10
3. MANIPULATING PASS-THROUGH QUERIES.....	12
4. MANIPULATING FILTER QUERIES .....	16
5. MANIPULATING WINDOW QUERIES .....	17
6. MANIPULATING JOIN QUERIES .....	18
7. MANIPULATING STATE-MACHINE QUERIES .....	20
8. MANIPULATING PARTITIONS .....	22
9. MANIPULATING WINDOWS CONNECTED TO QUERIES .....	28
10. CONSTRAINTS.....	29

# SIDDHI VISUAL EDITOR

## INITIAL VIEW



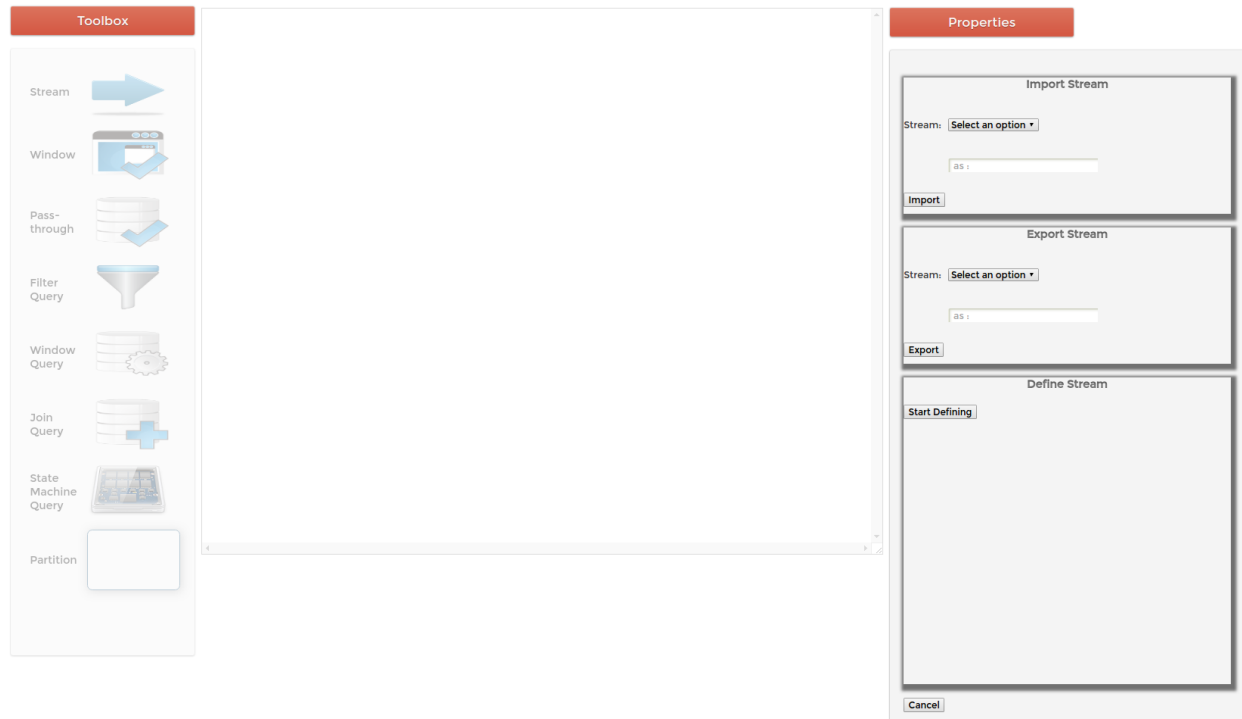
The container/ the canvas, as shown above becomes two-way scrollable depending on the placement of elements on the canvas. It is also expandable so as to provide a much larger drawing canvas.

## 1. MANIPULATING STREAMS

The Stream Element can be of 3 types. Namely:

1. Import Stream
2. Export Stream
3. Defined Stream

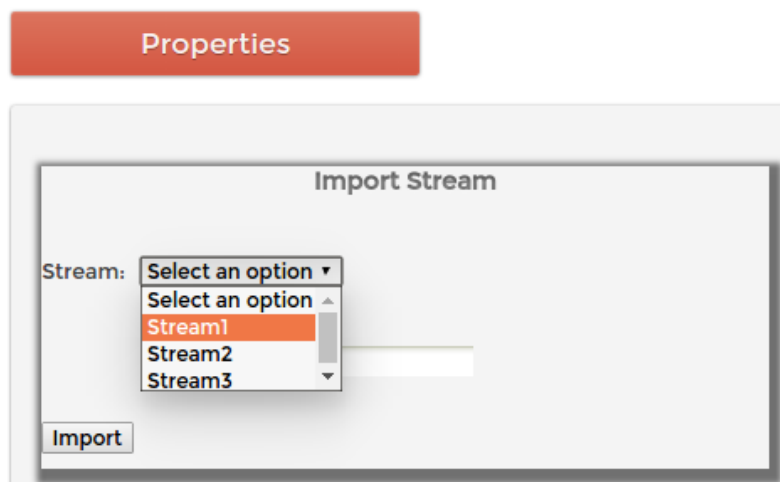
The type of Stream can be selected once the "Stream" element is dropped from the Toolbox on the left onto the container/canvas.



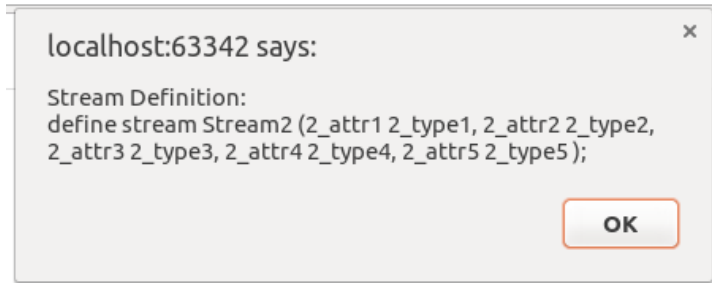
The toolbox and the container becomes disabled as soon as a stream element is dropped. This is done in order to prevent the user from making any additional modifications, before initializing the stream that has been dropped.

### 1.1. IMPORT STREAM

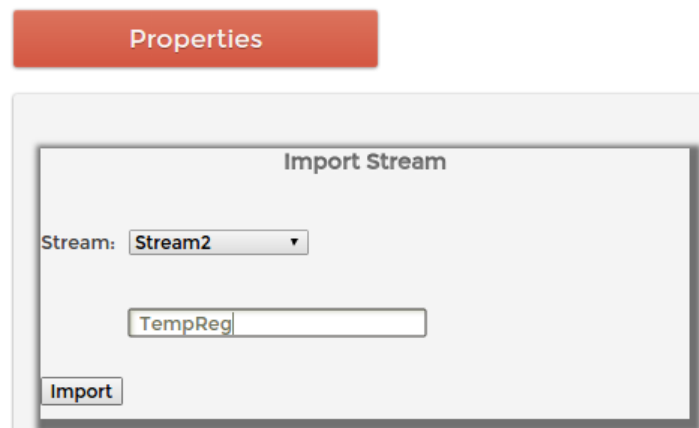
In order to configure an import stream, the Predefined stream from which the new stream's properties are derived, needs to be selected from the combo-box, listing all the available Predefined Streams.



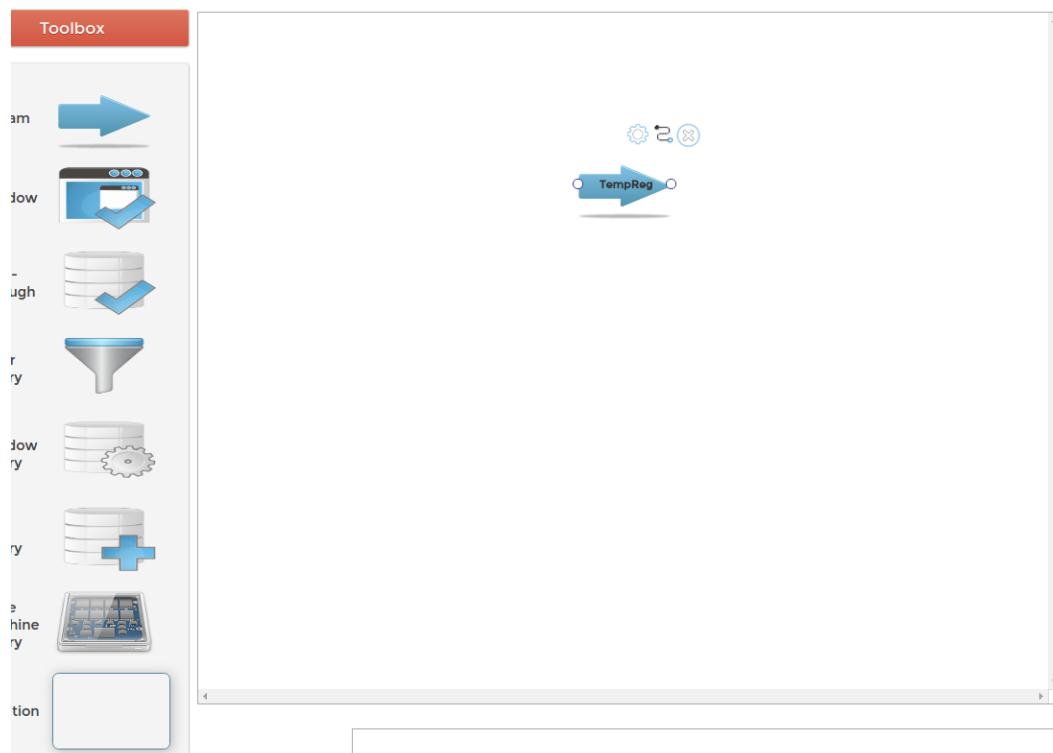
The definitions of the Predefined streams listed in the combo-box can be viewed via a pop-up alert that is displayed on-change of the selected Predefined Stream from the combo-box.



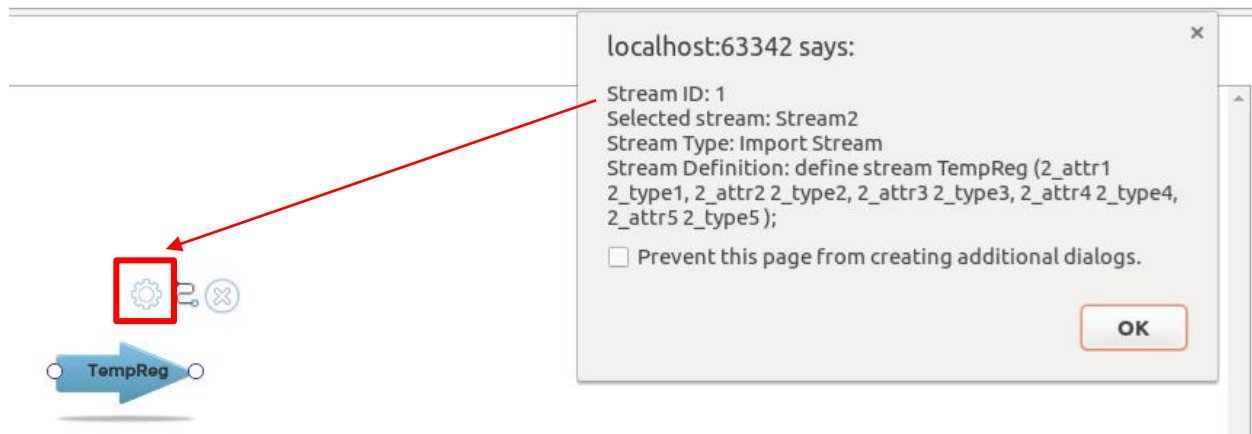
Then provide a name for the Import Stream and click on Import.



This will drop the newly created import stream on the canvas with the user-provided-name and the configured properties as chosen by the user.

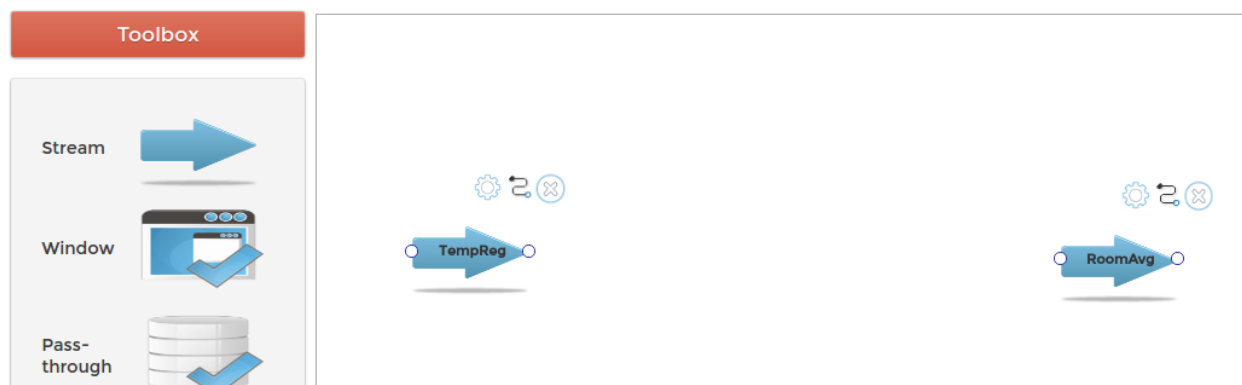
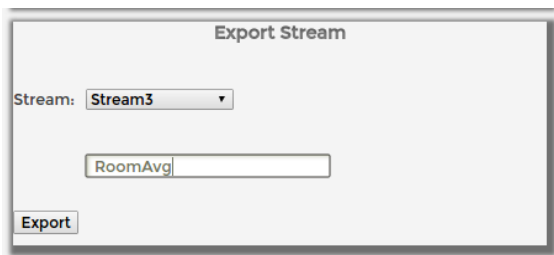


Further information with regard to the dropped stream can be viewed by clicking on the icon resembling the "Settings", which is the top leftmost icon out of the group of 3 icons.

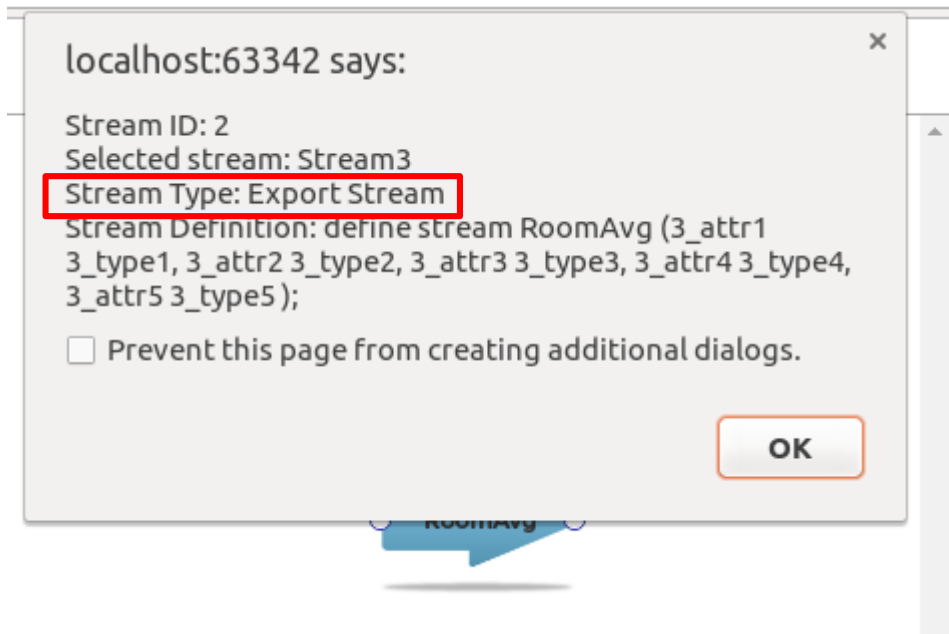


## 1.2. EXPORT STREAMS

To create an Export stream, the same procedure as followed in creating an Import stream can be followed.



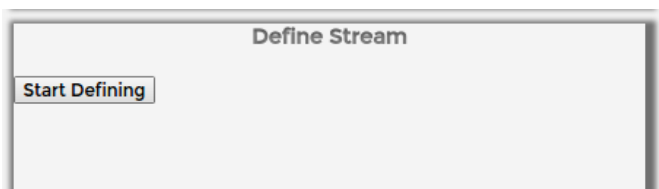
While viewing the set properties after the export stream is dropped, the type of the stream can be verified to be an "Export Stream". The same applies to Import and Defined streams with their relevant type.



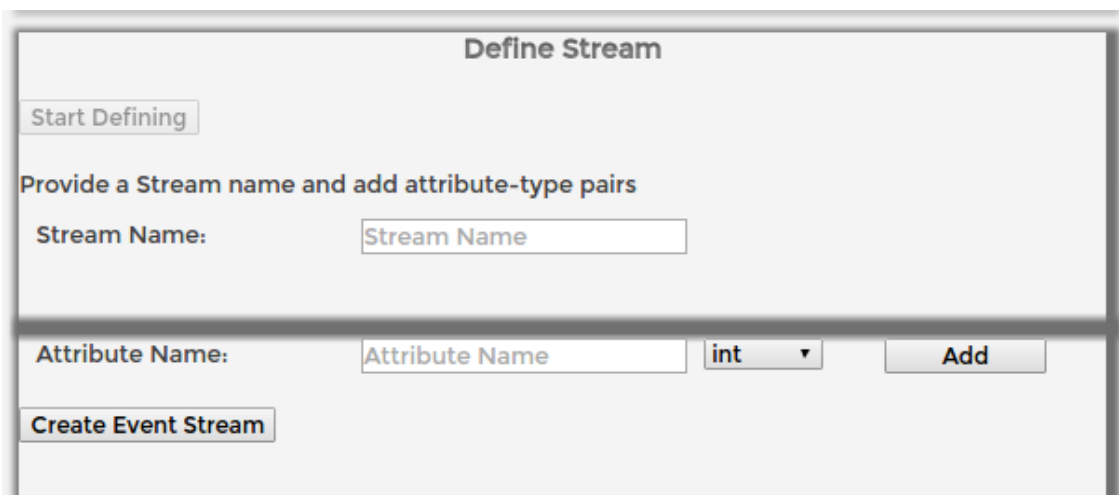
### 1.3. DEFINED STREAMS

Defined streams permit the user to create custom streams with completely user-defined attributes and attribute types.

Upon dropping a stream click on the “Start Defining” button under the Define Stream Division.



This will open a division where the user can provide a name for the Stream.



To add attribute-type pairs, enter the attribute name and select a type from the type-combo-box and click on the “Add” button.

The screenshot shows the 'Define Stream' dialog. At the top, there is a 'Start Defining' button. Below it, the instruction 'Provide a Stream name and add attribute-type pairs' is displayed. The 'Stream Name' field contains the text 'Factors'. Below this, there is a section for adding attributes. The 'Attribute Name' field contains 'roomNo'. To its right is a type-combo-box with a dropdown arrow, currently showing 'int'. To the right of the type-combo-box is an 'Add' button. Below the 'Attribute Name' field is a 'Create Event Stream' button. The type-combo-box is open, showing a list of data types: 'int', 'long', 'double', 'float', 'string', and 'bool'. The 'int' option is highlighted.

This will append the entered attribute onto a display table from which it can be deleted if not needed.

The screenshot shows the 'Define Stream' dialog. The 'Stream Name' field still contains 'Factors'. Below the instruction, there is a table with three columns: 'Attribute Name', 'Attribute Type', and 'Delete Row'. The table has one row with 'roomNo' in the 'Attribute Name' column, 'int' in the 'Attribute Type' column, and a red trash icon in the 'Delete Row' column. Below the table, the 'Attribute Name' field now contains 'roomNo', the type-combo-box shows 'int', and the 'Add' button is visible. The 'Create Event Stream' button is also present.

Attribute-type pairs can be associated with the defined stream to be created by adding them as mentioned above.

The screenshot shows the 'Define Stream' dialog. The 'Stream Name' field still contains 'Factors'. The table below the instruction now has two rows: the first row is 'roomNo' with type 'int' and a trash icon; the second row is 'reading' with type 'int' and a trash icon. Below the table, the 'Attribute Name' field now contains 'reading', the type-combo-box shows 'int', and the 'Add' button is visible. The 'Create Event Stream' button is also present. The type-combo-box is open, showing the list of data types: 'int', 'long', 'double', 'float', 'string', and 'bool'. The 'double' option is highlighted.



Define Stream

Start Defining

Provide a Stream name and add attribute-type pairs

Stream Name:

Attribute Name	Attribute Type	Delete Row
roomNo	int	
reading	double	

Attribute Name:  double ▾ Add

Create Event Stream

Then to finalize and drop the defined stream, click on the “Create Event Stream” button.

### Note:

*The attributes displayed/added on the attribute-type table displayed within the Define Stream Division, will only be considered as the attributes belonging to the created Stream.*



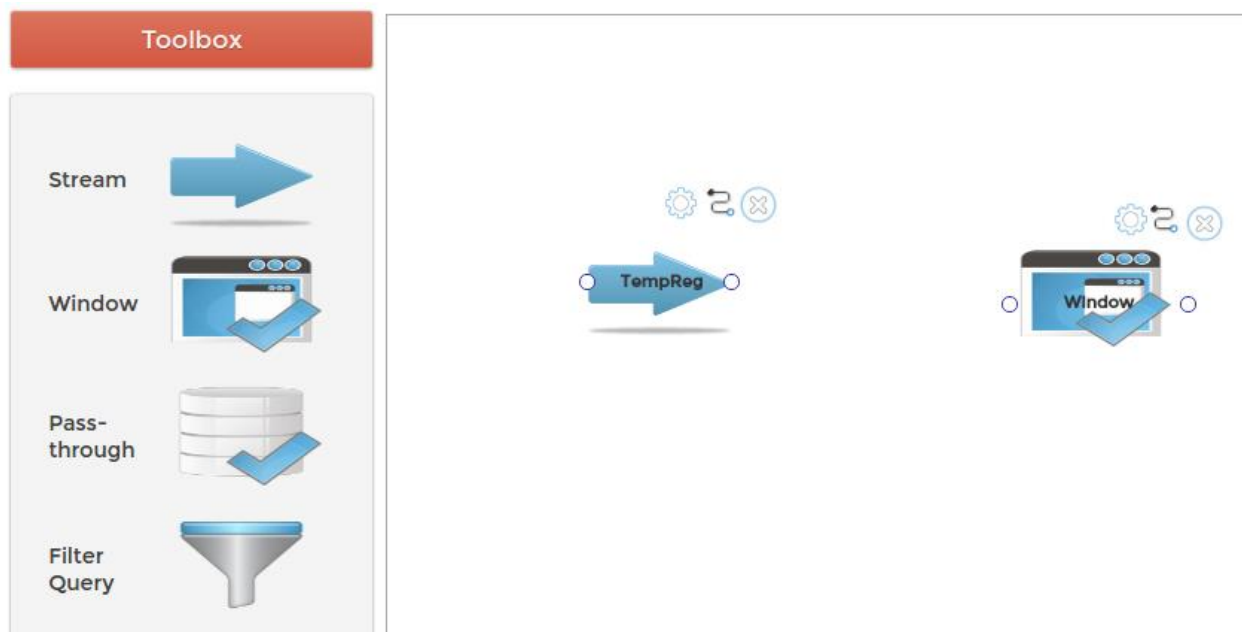
The defined stream's definition and related information can be viewed by clicking on the “Settings” icon as done for the other stream types.

## 2. MANIPULATING WINDOWS

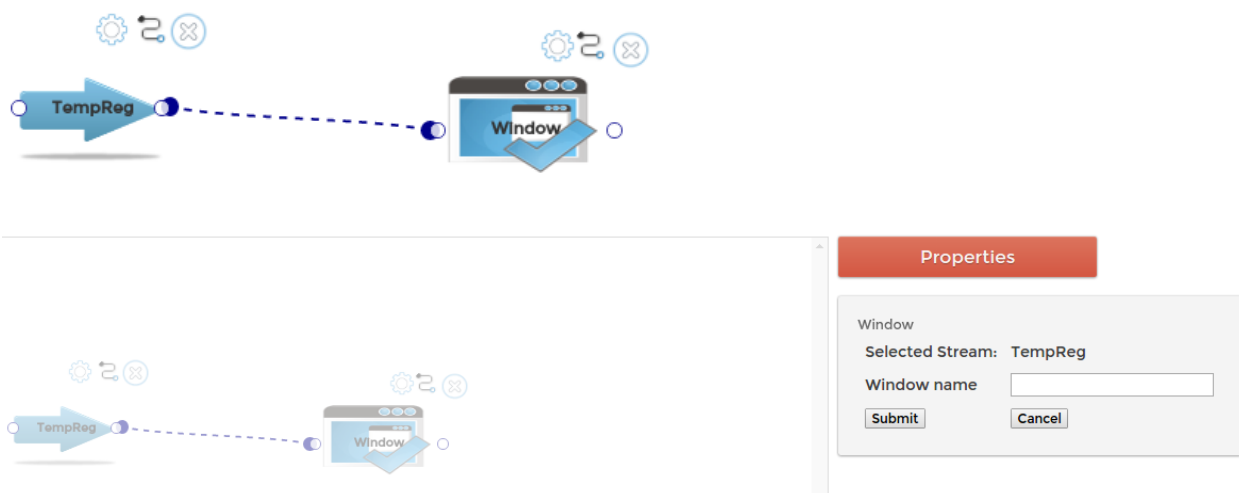
A window element can either be derived from a previously dropped Stream on the container or it can be defined by the user.

### 2.1. DERIVED WINDOW

To create a derived window, the stream from which the window is intended to derive, should be created before dropping the “Window” element. Following the stream's existence on the container, the window element can be dropped and the window and the respective stream need to be connected.



Once they are connected, the window can be initialized by clicking on the “Settings” icon on the “Window” element.



This will open the Properties pane where a name for the window element can be provided. This is the property that needs to be set as the rest of the properties such as attributes and types will be derived from the stream that it's connected to.

**Properties**

Window

Selected Stream: TempReg

Window name

This finalizes the window element displaying the element with the name provided by the user.



In cases of needing to modify the "Window" element, the settings icon can be clicked and the properties can be modified.

## 2.2. DEFINED WINDOW

A defined window resembles a Defined Stream. Hence setting up a defined window is similar to that of setting up a defined stream.

The diagram shows a 'TempReg' stream element connected to a 'unitTemp' window element. A dashed blue line connects the output of the stream to the input of the window. Both elements have a settings icon (gear) and a delete icon (X) above them.

**Properties**

Define Window

Window Name:

Attribute-Type pairs can be defined and associated with the Window.

**Properties**

**Define Window**  
Window Name:

**Properties**

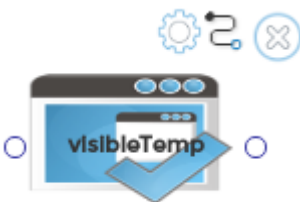
**Define Window**  
Window Name:   
  

Attribute Name	Attribute Type	Delete Row
scale	float	<input type="button" value="Delete"/>
unit	double	<input type="button" value="Delete"/>

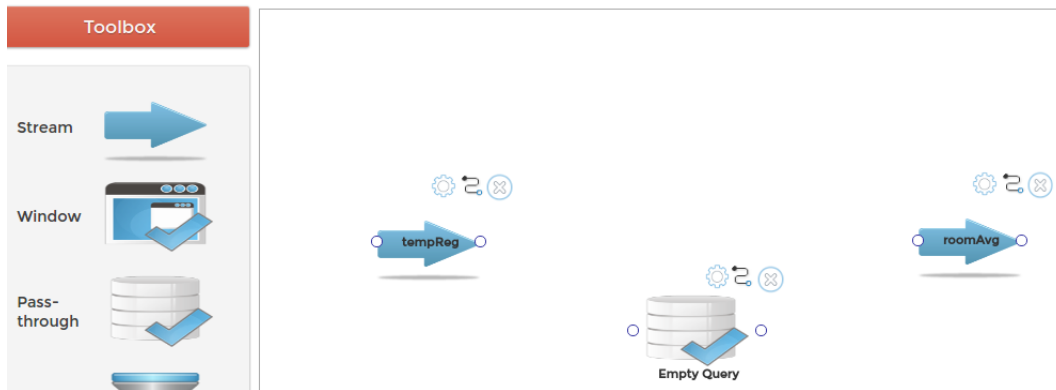
**Attribute Name:**   
**Attribute Type:**

The window element with the set-properties will be dropped onto the canvas on clicking the “Create Window” button.



### 3. MANIPULATING PASS-THROUGH QUERIES

A "Pass-through query" element can be dragged from the toolbox and dropped onto the canvas.



Before setting up the query's properties, it needs to be connected to the respective streams that will serve as the "From-stream" and the "Into-stream".



Following the connections, upon clicking the "Settings" icon, the following properties pane will be displayed with the "from-stream", into stream and attributes of the "into stream" predefined on the form as taken from the connections.

Properties

Pass-through Query

Query name

from: tempReg

Select :

<input type="text"/>	as	3_attr1
<input type="text"/>	as	3_attr2
<input type="text"/>	as	3_attr3
<input type="text"/>	as	3_attr4
<input type="text"/>	as	3_attr5

insert into: roomAvg

Rest of the data can be filled by the user and the *Query can be submitted*.

Properties

Pass-through Query

Query name

unitProcess

from:

tempReg

Select :

roomNo

as

3\_attr1

unit

as

3\_attr2

occupants

as

3\_attr3

temp

as

3\_attr4

area

as

3\_attr5

insert into:

roomAvg

Submit Query

Cancel



This simple execution plan can be stored and exported as Json formatted data. To view the elements and connections in the execution plan, click on “Save”.

Window

Pass-through

Filter Query

Window Query

Join Query

State Machine Query

Partition

```
graph LR; tempReg --> unitProcess; unitProcess --> roomAvg;
```

Json Output:

```
[{"node":{"id":"1","class":"streamdrop ui-draggable","position":{"top":210.20835182250977,"left":115.2083480078125,"bottom":282.20835182250977,"right":235.2083480078125},"predefinedStream":"Stream1","name":"tempReg","kind":"import"},"id":"2","class":"streamdrop ui-draggable","position":{"top":208.21183082641602,"left":678.2118575292968,"bottom":280.211830826416,"right":798.2118575292968},"predefinedStream":"Stream3","name":"roomAvg","kind":"export"},"id":"3","class":"sqquerydrop ui-draggable","position":{"top":290.19098732055664,"left":388.19446250976563,"bottom":361.5909873205567,"right":508.59446250976566},"name":
```

File name to save as:

Save

Export

Load

Json Output:


edStream": "Stream1", "name": "tempReg", "kind": "import"}, {"id": "2", "class": "stream", "top": 208.21183082641602, "left": 678.2118575292968, "bottom": 280.21183082641602, "tream": "Stream3", "name": "roomAvg", "kind": "export"}, {"id": "3", "class": "squerydr", "top": 290.19098732055664, "left": 388.19446250976563, "bottom": 361.59098732055664, "name": "roomAvg", "kind": "export"}]

File name to save as:

Save

Export

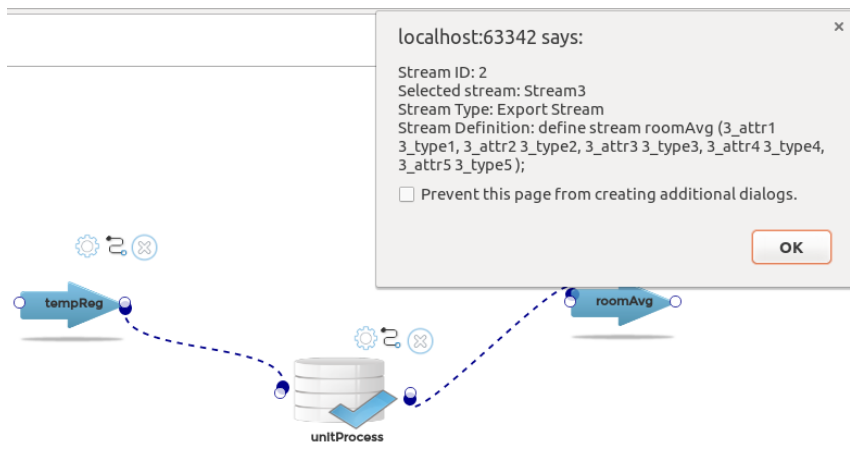
Load

 2f7d8970-e3c....txt ^

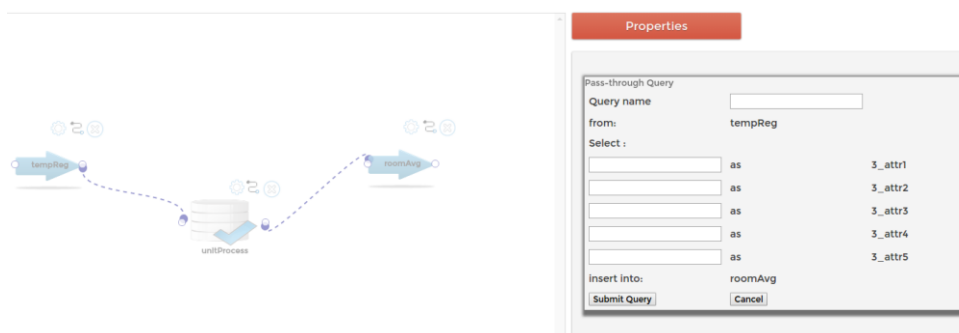
In regenerating a previously created execution plan, the data needs to be provided in a specified Json formatted data in the text area and click on the "Load" button.



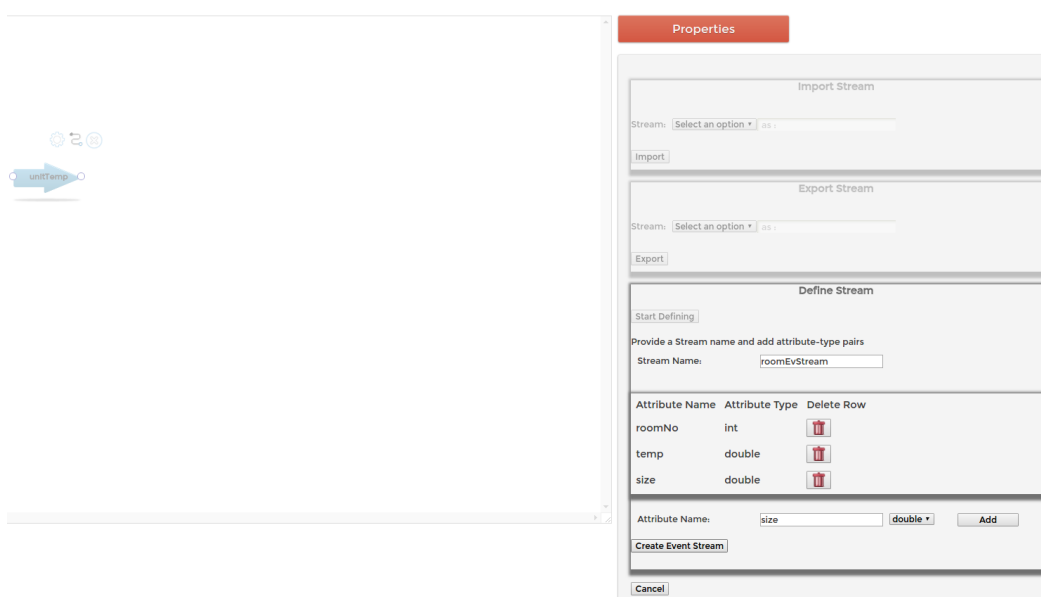
All the previously selected information will be reloaded into the generated model's elements. This can be checked by clicking on the "settings" icon.



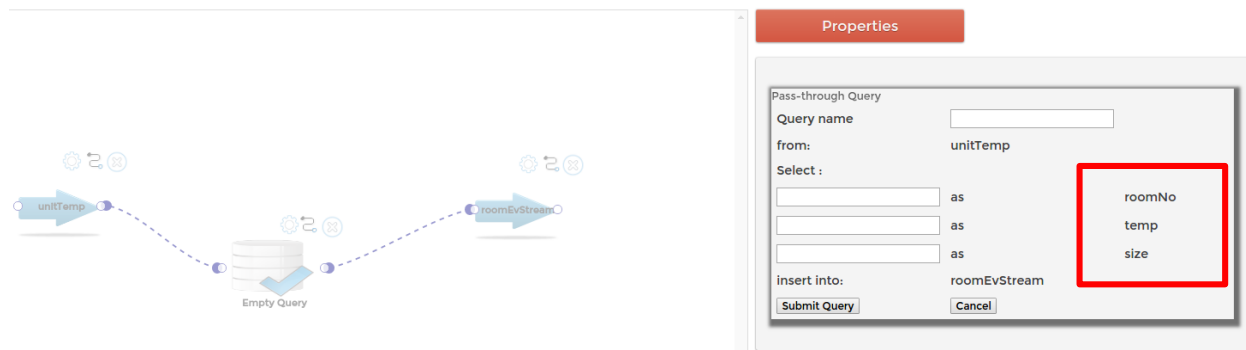
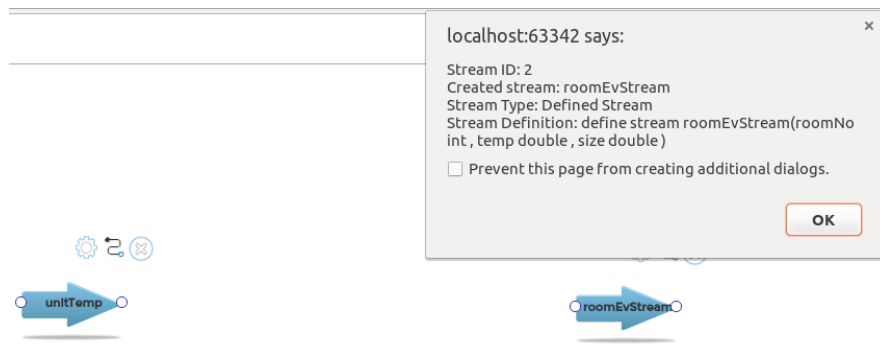
And the properties of the Pass-through query can also be reset.



In the case of a defined stream being connected as the into-stream of the Pass-through query, the defined attributes will be derived.

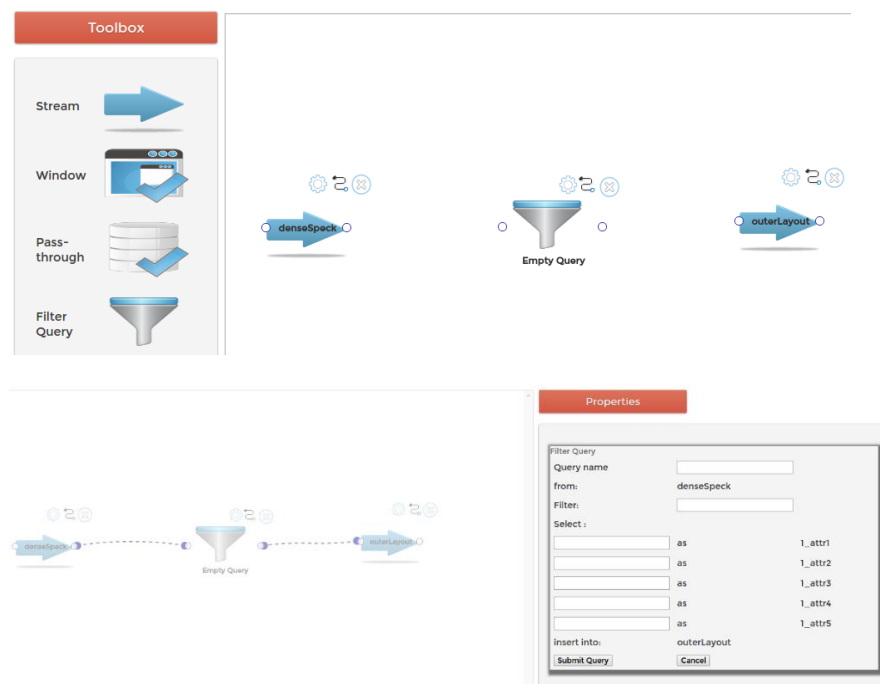






#### 4. MANIPULATING FILTER QUERIES

The filter query creation is very similar to that of the Pass-through query.



The only difference is the provision of an option to add a filter statement on the query via the form.

Properties

**Filter Query**

Query name:

from:

Filter:

Select :

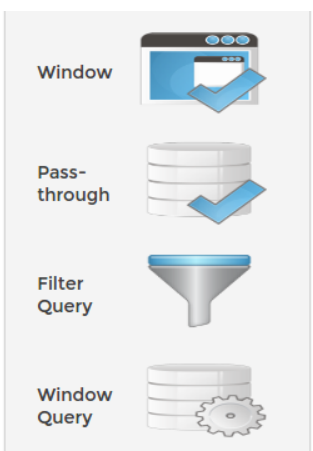
eventNo	as	1_attr1
items	as	1_attr2
price	as	1_attr3
location	as	1_attr4
wall	as	1_attr5

insert into:

```
{
  "node": [
    {
      "id": "1",
      "class": "streamdrop ui-draggable",
      "position": {
        "top": 253.09027931274414,
        "left": 47.10071250976563,
        "bottom": 325.09027931274414,
        "right": 167.10071250976563,
        "predefinedStream": "Stream2",
        "name": "denseSpeck",
        "kind": "import"
      },
      "type": "streamdrop"
    },
    {
      "id": "2",
      "class": "streamdrop ui-draggable",
      "position": {
        "top": 244.2187583166504,
        "left": 678.2118575292968,
        "bottom": 316.2187583166504,
        "right": 798.2118575292968,
        "predefinedStream": "Stream1",
        "name": "outerLayout",
        "kind": "export"
      },
      "type": "streamdrop"
    },
    {
      "id": "3",
      "class": "filterdrop ui-draggable",
      "position": {
        "top": 240.10416481079102,
        "left": 369.0972335058594,
        "bottom": 311.50416481079105,
        "right": 489.4972335058594,
        "name": "curatorProcess",
        "fromStream": {
          "index": 0,
          "name": "denseSpeck",
          "filter": "items >= 20 and price < 700",
          "attributes": [
            {
              "attrName": "eventNo",
              "attrType": "1_attr1"
            },
            {
              "attrName": "items",
              "attrType": "1_attr2"
            },
            {
              "attrName": "price",
              "attrType": "1_attr3"
            },
            {
              "attrName": "location",
              "attrType": "1_attr4"
            },
            {
              "attrName": "wall",
              "attrType": "1_attr5"
            }
          ],
          "intoStream": {
            "index": 1,
            "name": "outerLayout"
          }
        }
      },
      "type": "filterdrop"
    }
  ],
  "connections": [
    {
      "connectionId": "con_10",
      "pageSourceId": "1-Outimport",
      "pageTargetId": "3-in"
    },
    {
      "connectionId": "con_27",
      "pageSourceId": "3-out",
      "pageTargetId": "2-Inexport"
    }
  ]
}
```

## 5. MANIPULATING WINDOW QUERIES

The window query creation also is same as the previous two queries with the provision of adding windowing facilities via the form.



## Properties

Window Query

Query name:

from:

Filter 1:

Window:

Filter 2:

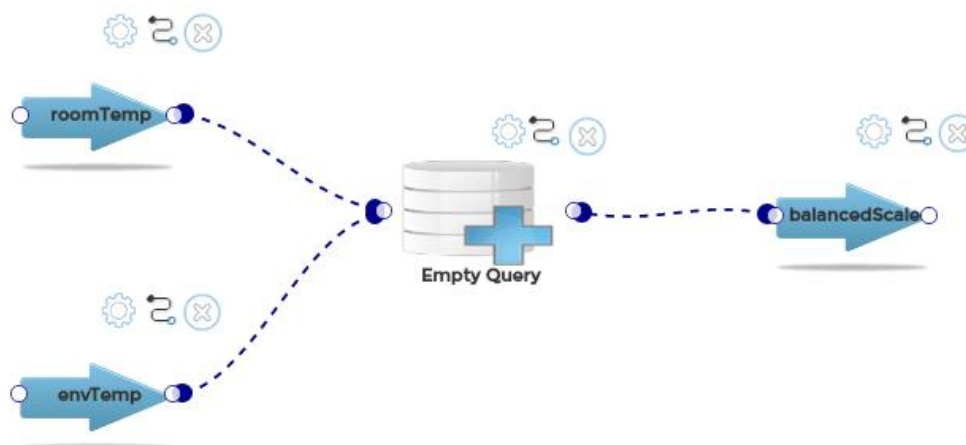
Select :

<input type="text" value="avgPrice"/>	as	<input type="text" value="avg"/>
<input type="text" value="price"/>	as	<input type="text" value="price"/>
<input type="text" value="maxPrice"/>	as	<input type="text" value="max"/>
<input type="text" value="minPrice"/>	as	<input type="text" value="min"/>

Insert into:

```
{
  "node": [
    {
      "id": "1",
      "class": "streamdrop ui-draggable",
      "position": {
        "top": 223.10764381469727,
        "left": 81.09375450195313,
        "bottom": 295.10764381469727,
        "right": 201.09375450195313
      },
      "predefinedStream": "Stream1",
      "name": "stockExch",
      "kind": "import"
    },
    {
      "id": "2",
      "class": "streamdrop ui-draggable",
      "position": {
        "top": 227.2223732055664,
        "left": 587.2048995214843,
        "bottom": 298.2223732055664,
        "right": 707.2048995214843
      },
      "name": "IBMStockQ",
      "numberOfAttributes": 5,
      "kind": "defined",
      "attributes": [
        {
          "attributeName": "avg",
          "attributeType": "double"
        },
        {
          "attributeName": "price",
          "attributeType": "double"
        },
        {
          "attributeName": "max",
          "attributeType": "double"
        },
        {
          "attributeName": "min",
          "attributeType": "double"
        }
      ]
    },
    {
      "id": "3",
      "class": "wquerydrop ui-draggable",
      "position": {
        "top": 216.19791481079102,
        "left": 331.1979415136719,
        "bottom": 287.59791481079102,
        "right": 450.5979415136719
      },
      "name": "terminal",
      "fromStream": {
        "index": 0,
        "name": "stockExch",
        "filter1": "price >= 20",
        "window": "length(50)",
        "filter2": "price <=100",
        "attributes": [
          {
            "attrName": "avgPrice",
            "attrType": "avg"
          },
          {
            "attrName": "price",
            "attrType": "price"
          },
          {
            "attrName": "maxPrice",
            "attrType": "max"
          },
          {
            "attrName": "minPrice",
            "attrType": "min"
          }
        ],
        "intoStream": {
          "index": 2,
          "name": "IBMStockQ"
        }
      }
    }
  ],
  "connections": [
    {
      "connectionId": "con_10",
      "pageSourceId": "1-Outimport",
      "pageTargetId": "3-In"
    },
    {
      "connectionId": "con_20",
      "pageSourceId": "3-out",
      "pageTargetId": "2-Indefined"
    }
  ]
}
```

## 6. MANIPULATING JOIN QUERIES



Before initializing a join query, at-most and always, 2 streams need to be connected to the join query as it's "From-streams", and a single into stream needs to be connected.

Once these connections are confirmed, the properties can be set.

**Properties**

**Join Query**  
Query Name:

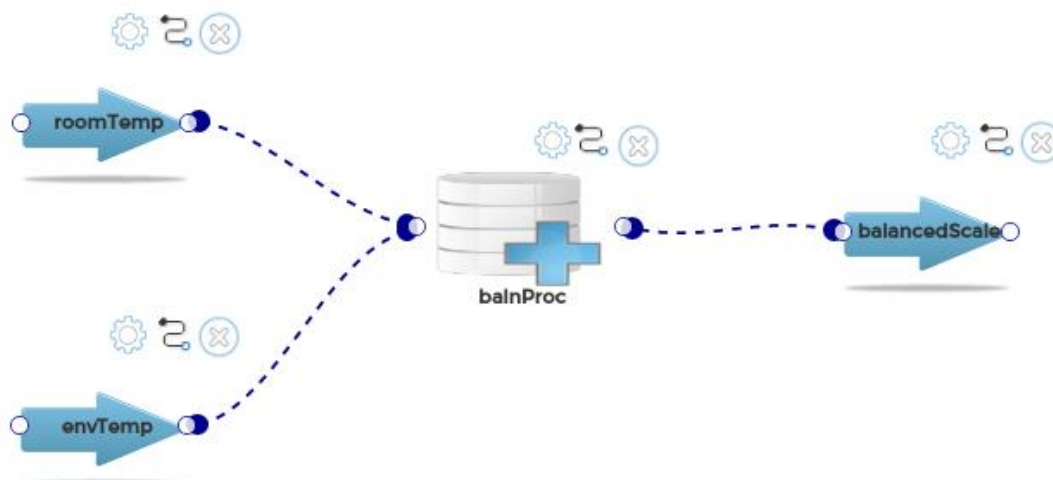
**Left Stream Details**  
Left Stream:   
Filter 1:   
Window:   
Filter 2:

**Right Stream Details**  
Right Stream:   
Filter 1:   
Window:   
Filter 2:

<input type="text" value="unit"/>	as	<input type="text" value="1_attr1"/>
<input type="text" value="value"/>	as	<input type="text" value="1_attr2"/>
<input type="text" value="balncscale"/>	as	<input type="text" value="1_attr3"/>
<input type="text" value="area"/>	as	<input type="text" value="1_attr4"/>
<input type="text" value="dimension"/>	as	<input type="text" value="1_attr5"/>

  
insert into:

In the properties form, one of the two “from-streams” needs to be selected from the combo-box as the left/right join stream and its respective properties needs to be set.



```

{"node":{

{"id":"1","class":"streamdrop ui-draggable","position":
{"top":157.0833365637207,"left":69.09723350585938,"bottom":229.0833365637207,"right":189.09723350585938},"predefinedStream":"Stream1","name":"roomTemp","kind":"import"},

{"id":"2","class":"streamdrop ui-draggable","position":
{"top":352.1007163244629,"left":68.10764,"bottom":424.1007163244629,"right":188.10764},"name":"envTemp","numberOfAttributes":5,"kind":"defined","attributes":
[{"attributeName":"location","attributeType":"string"},{"attributeName":"liveStream","attributeType":"string"},{"attributeName":"unit","attributeType":"string"},
{"attributeName":"value","attributeType":"double"}]},

{"id":"3","class":"streamdrop ui-draggable","position":
{"top":227.20487281860352,"left":597.2222335058593,"bottom":299.2048728186035,"right":717.2222335058593},"predefinedStream":"Stream1","name":"balancedScale","kind":"export"},

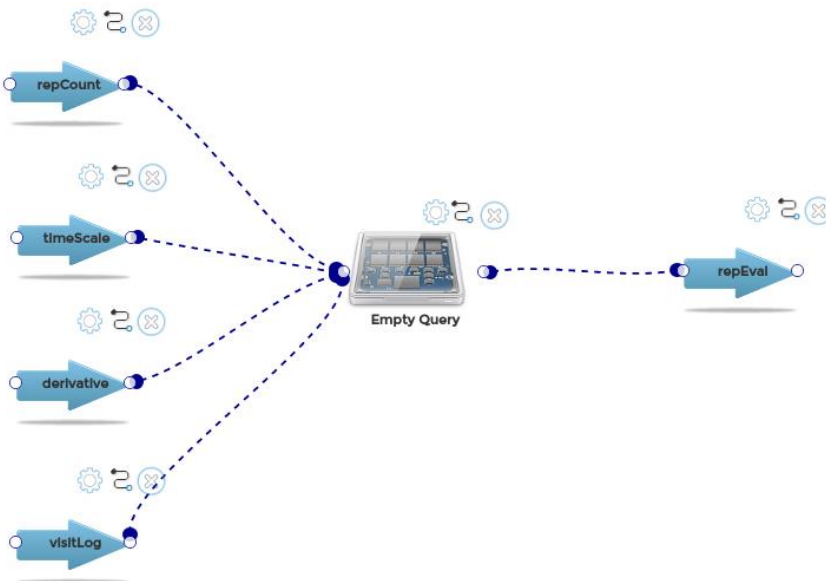
{"id":"4","class":"jquerydrop ui-draggable","position":
{"top":213.10764381469727,"left":329.0972335058594,"bottom":284.5076438146973,"right":449.4972335058594},"name":"balnProc","leftStream":
{"name":"roomTemp","filter1":{"unit='F'","window":"length(2000)","filter2":""},"rightStream":{"name":"envTemp","filter1":{"unit='F'","window":"time(500)","filter2":"time < 30"},"attributes":[{"attrName":"unit","attrType":"1_attr1"}],"intoStreamName":"balancedScale"}},

"connections":[{"connectionId":"con_11","pageSourceId":"1-Outimport","pageTargetId":"4-In"},{"connectionId":"con_21","pageSourceId":"2-Outdefined","pageTargetId":"4-In"},
{"connectionId":"con_31","pageSourceId":"4-out","pageTargetId":"3-Inexport"}]}

```

## 7. MANIPULATING STATE-MACHINE QUERIES

A state machine Query permits the connection of multiple “from-streams”.



Properties

Query Name:

Create Multiple States

State ID:

Stream:

Filter :

Enter the Process logic

Process logic:

	as	1_attr1
	as	1_attr2
	as	1_attr3
	as	1_attr4
	as	1_attr5

insert into:                      repEval

Multiple states can be added to the query using the “Add State” button on the form. After specifying the process logic and the attributes the state machine query can be dropped on the canvas.

Create Multiple States

State ID:

Stream:

Filter :

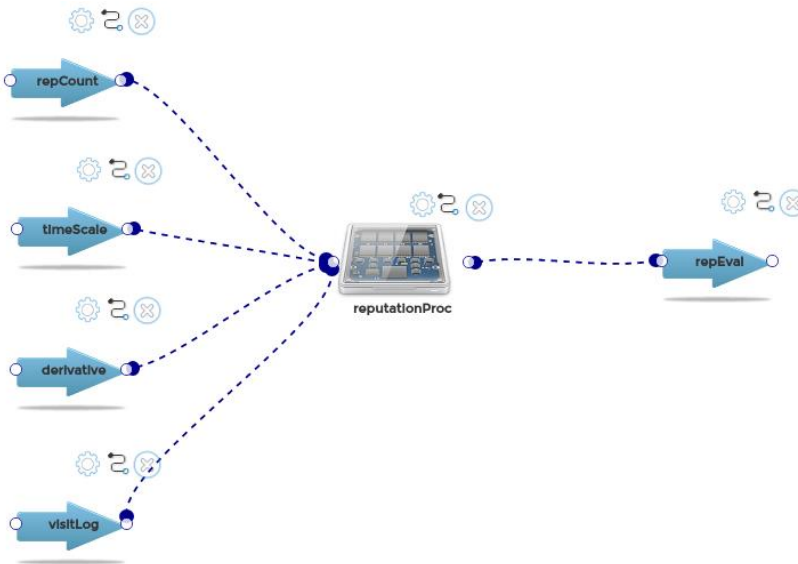
State ID:

Stream:

Filter :

Enter the Process logic

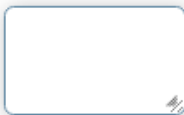
Process logic:



```
{
  "node": {
    {
      "id": "1",
      "class": "streamdrop ui-draggable",
      "position": {
        "top": 123.10764381469727,
        "left": 64.09723350585938,
        "bottom": 195.10764381469727,
        "right": 184.09723350585938,
        "predefinedStream": "Stream2",
        "name": "repCount",
        "kind": "import"
      },
    },
    {
      "id": "2",
      "class": "streamdrop ui-draggable",
      "position": {
        "top": 268.10764381469727,
        "left": 71.09375450195313,
        "bottom": 340.10764381469727,
        "right": 191.09375450195313,
        "predefinedStream": "Stream3",
        "name": "timeScale",
        "kind": "import"
      },
    },
    {
      "id": "3",
      "class": "streamdrop ui-draggable",
      "position": {
        "top": 404.20139381469727,
        "left": 70.2083480078125,
        "bottom": 476.20139381469727,
        "right": 190.2083480078125,
        "predefinedStream": "Stream3",
        "name": "derivative",
        "kind": "import"
      },
    },
    {
      "id": "4",
      "class": "streamdrop ui-draggable",
      "position": {
        "top": 554.2013938146972,
        "left": 70.19098350585938,
        "bottom": 626.2013938146972,
        "right": 190.19098350585938,
        "predefinedStream": "Stream2",
        "name": "visitLog",
        "kind": "import"
      },
    },
    {
      "id": "5",
      "class": "streamdrop ui-draggable",
      "position": {
        "top": 299.09723732055664,
        "left": 699.0972945410156,
        "bottom": 371.09723732055664,
        "right": 819.0972945410156,
        "predefinedStream": "Stream1",
        "name": "repEval",
        "kind": "export"
      },
    },
    {
      "id": "6",
      "class": "stquerydrop ui-draggable",
      "position": {
        "top": 288.10764381469727,
        "left": 384.0972335058594,
        "bottom": 359.5076438146973,
        "right": 503.4972335058594,
        "name": "reputationProc",
        "processLogic": " every a1 = OrderStock1[action a1 = repStack[action == 'earn'] and a2 = repStack[action == 'earn']",
        "intoStreamName": "repEval",
        "state": {
          [{"stateId": "st1", "stateSelectedStream": "timeScale", "stateFilter": "time < 300"}, {"stateId": "st2", "stateSelectedStream": "repCount", "stateFilter": "repCount >= 2000"}],
          [{"attrName": "user", "attrType": "1_attr1"}, {"attrName": "day", "attrType": "1_attr2"}, {"attrName": "timestamp", "attrType": "1_attr3"}, {"attrName": "derivativeInit", "attrType": "1_attr4"}, {"attrName": "reputations", "attrType": "1_attr5"}]
        }
      },
    },
  },
  "connections": [
    {
      "connectionId": "con_13",
      "pageSourceId": "1-Outimport",
      "pageTargetId": "6-In",
      "connectionId": "con_23",
      "pageSourceId": "2-Outimport",
      "pageTargetId": "6-In",
      "connectionId": "con_33",
      "pageSourceId": "3-Outimport",
      "pageTargetId": "6-In",
      "connectionId": "con_43",
      "pageSourceId": "4-Outimport",
      "pageTargetId": "6-In",
      "connectionId": "con_53",
      "pageSourceId": "6-out",
      "pageTargetId": "5-Inexport"
    }
  ]
}
```

## 8. MANIPULATING PARTITIONS

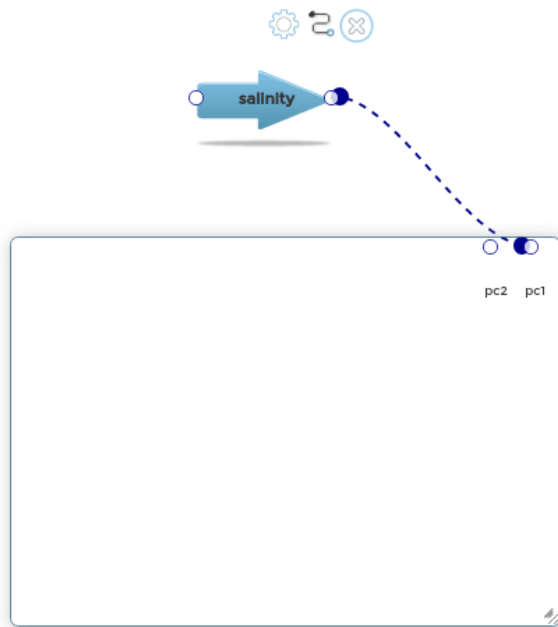
In simple terms, a partition can be used to group a set of elements. After dropping a partition element, it can be resized according to need.



To set a partition, we can create multiple partitions through which external streams can be connected through. Each "Partition Condition" condition can be created by *double clicking* on the partition.



Proceeding this, to associate a particular stream/window with a partition condition anchor, drag a connection from that element to the partition condition anchor and then click on the pc anchor point.



This will open the form as shown below. In the form, a partition can be declared with Variable partitioning or Range Partitioning, but not both simultaneously.

The screenshot shows the 'Properties' form on the right side of the diagram. The form has a red header with the text 'Properties'. Below the header, there is a section titled 'Define Partition Condition'. This section contains a 'Partition ID:' label followed by a text input field. Below this, there are two sections: 'Variable Partitioning' and 'Range Partitioning'. The 'Variable Partitioning' section has a 'Variable Partition Type:' label followed by a dropdown menu showing '1\_attr1' and a button labeled 'Add variable partition type'. The 'Range Partitioning' section has a 'Range Partition Type:' label followed by a text input field and a button labeled 'Add range partition type'. At the bottom of the form, there are two buttons: 'Apply Partition Condition' and 'Cancel'.

So, for example declaring the pc anchor with variable partitioning will display a combo with the connected stream/window's attributes loaded and the user can select the attributes that he/she wishes to be associated with the pc anchor being created and add it to the table as shown below.






## Properties

Define Partition Condition

Partition ID:

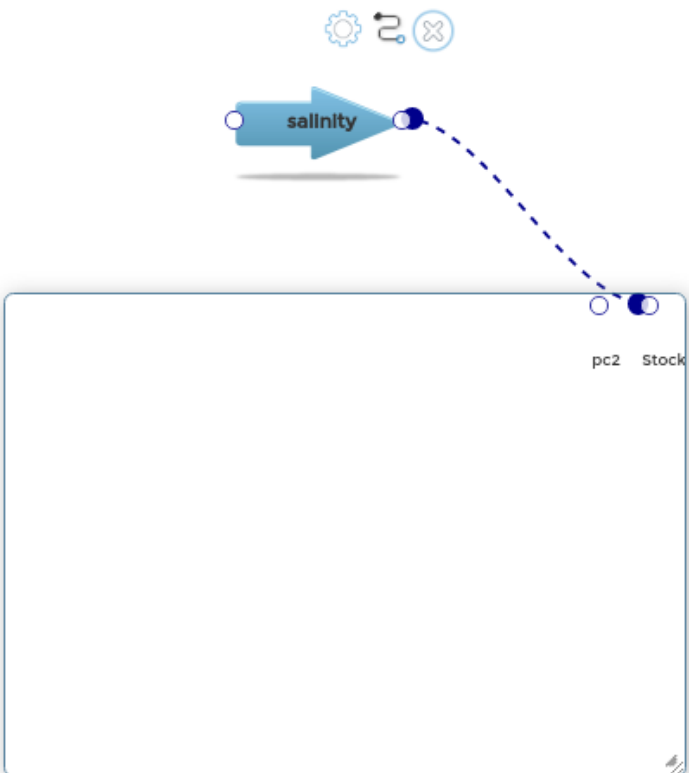
Variable Partitioning

Partition Type	Delete
1_attr1	
1_attr2	
1_attr5	

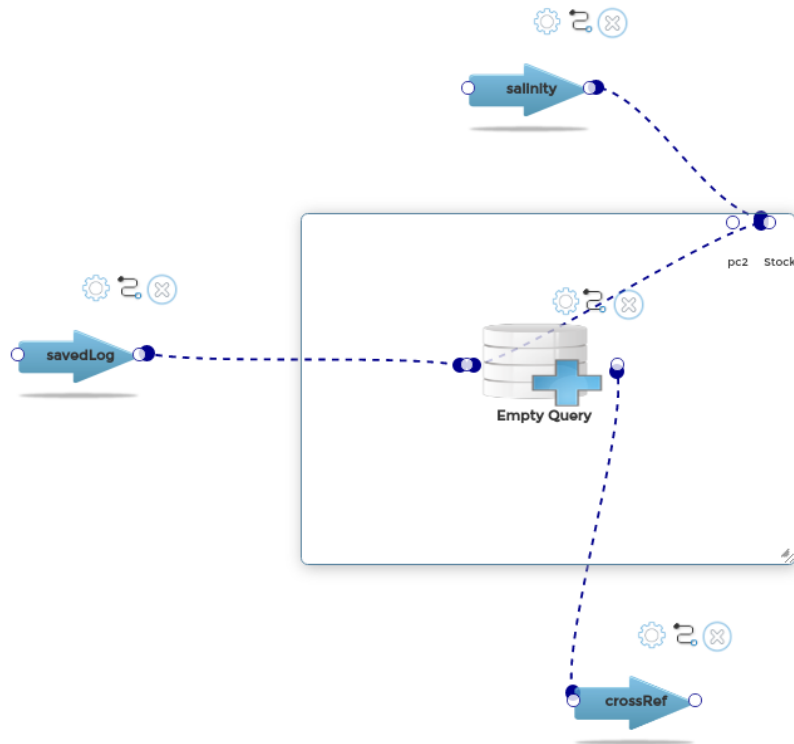
Variable Partition Type:

Range Partitioning

Range Partition Type:



Following this, the rest of the connections can be made with relation to the partition as usual.



In the above scenario of creating a join query with one from stream being a regular import stream and the other being a Partition condition, when the properties for the join query are set, the from streams from which the join query gets its left/right query will display both the connected streams.

**Properties**

Join Query

Query Name:

Left Stream Details

Left Stream: 

savedLog

savedLog

salinity

Filter 1:

Window:

All the respective properties for the query can be set.

Properties

**Join Query**

Query Name:

**Left Stream Details**

Left Stream:

Filter 1:

Window:

Filter 2:

**Right Stream Details**

Right Stream:

Filter 1:

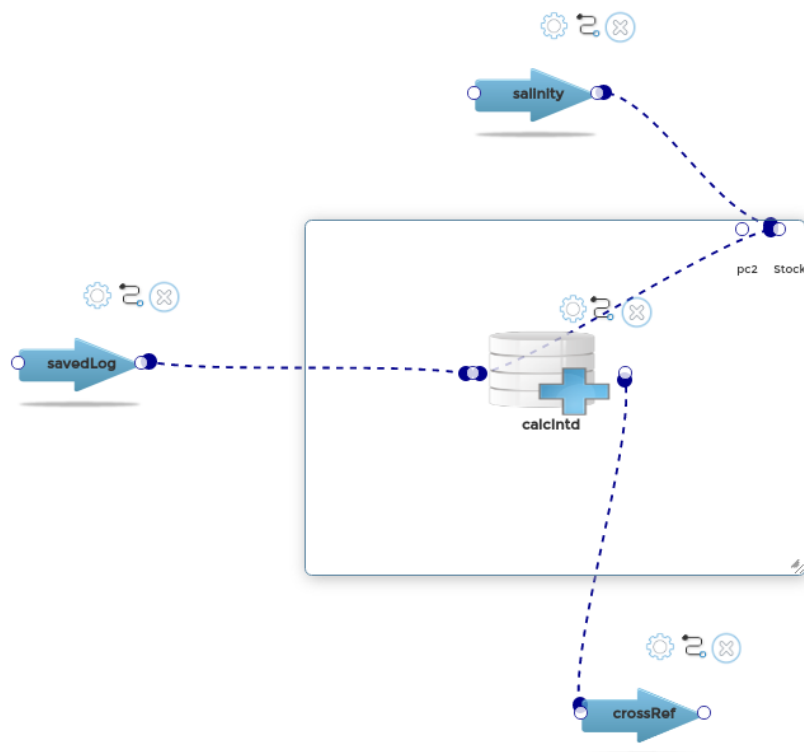
Window:

Filter 2:

<input type="text" value="value"/>	as	<input type="text" value="3_attr1"/>
<input type="text" value="rec"/>	as	<input type="text" value="3_attr2"/>
<input type="text" value="time"/>	as	<input type="text" value="3_attr3"/>
<input type="text" value="scale"/>	as	<input type="text" value="3_attr4"/>
<input type="text" value="unit"/>	as	<input type="text" value="3_attr5"/>

Insert into:

Once the final output is confirmed the model will look as follows



```

{"node":
[
{"id": "1", "class": "partitiondrop ui-draggable ui-resizable", "position":
{"top": 209.5312583106504, "left": 422.5173995214844, "bottom": 505.7090383166504, "right": 846.0951795214843}, "partitionName": "Stock", "type": "Partition
Condition", "numberOfAttributes": 3, "subPartitionConditionId": "1", "attributes": [{"attrName": "1_attr1", "attrType": "1_type1"}, {"attrName": "1_attr2", "attrType": "1_type2"},
{"attrName": "1_attr5", "attrType": "1_type5"}]},
[
{"id": "2", "class": "streamdrop ui-draggable", "position":
{"top": 68.50695106567383, "left": 564.5313155371093, "bottom": 139.50695106567383, "right": 684.5313155371093}, "predefinedStream": "Stream1", "name": "salinity", "kind": "import"},

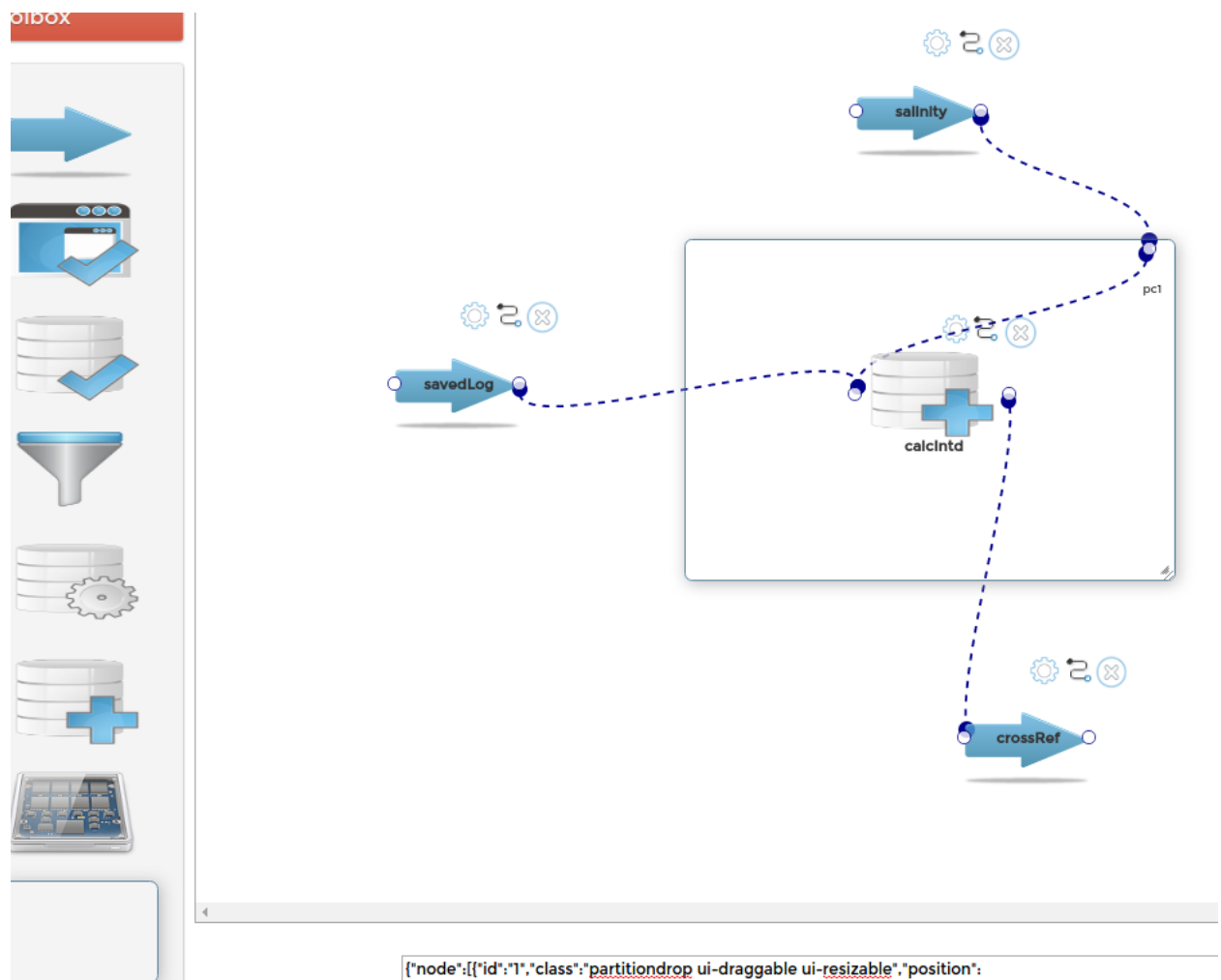
{"id": "4", "class": "joquerydrop ui-draggable", "position":
{"top": 301.3194663244629, "left": 569.2882125097656, "bottom": 372.7194663244629, "right": 689.6882125097655}, "name": "calcIntd", "leftStream": {"name": "savedLog", "filter1": "rec >
10", "window": "length(50)", "filter2": "rec < 200"}, "rightStream": {"name": "salinity", "filter1": "value >=2.3", "window": "length(50)", "filter2": "value <= 7.0"}, "attributes":
[], "intoStreamName": "crossRef"},

{"id": "5", "class": "streamdrop ui-draggable", "position":
{"top": 304.09723732055664, "left": 165.10419151367188, "bottom": 375.09723732055664, "right": 285.1041915136719}, "predefinedStream": "Stream2", "name": "savedLog", "kind": "import"},

{"id": "6", "class": "streamdrop ui-draggable", "position":
{"top": 610.1042258459472, "left": 657.1007125097656, "bottom": 682.1042258459472, "right": 777.1007125097656}, "predefinedStream": "Stream3", "name": "crossRef", "kind": "export"}],
"connections": [{"connectionId": "con_17", "pageSourceId": "2-Outimport", "pageTargetId": "1-pc1"}, {"connectionId": "con_36", "pageSourceId": "5-Outimport", "pageTargetId": "4-in"},
{"connectionId": "con_46", "pageSourceId": "1-pc1", "pageTargetId": "4-in"}, {"connectionId": "con_56", "pageSourceId": "4-out", "pageTargetId": "6-Inexport"}]}

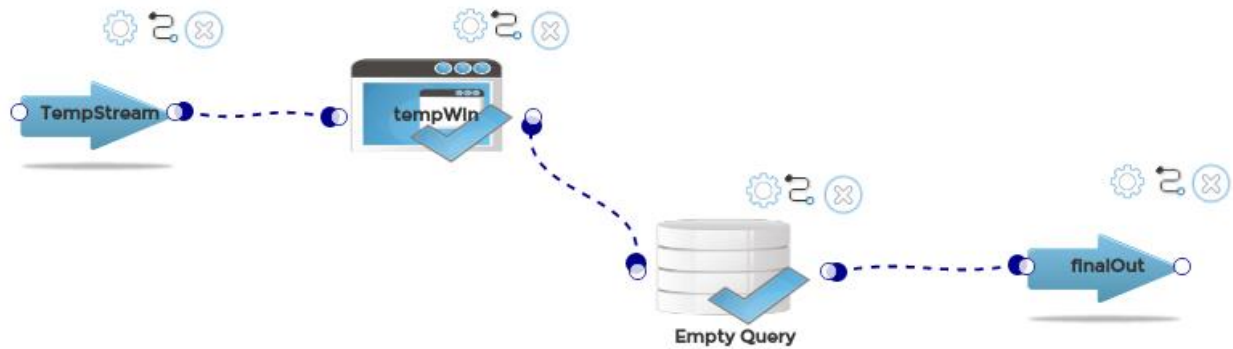
```

Regenerating the above same will redraw the entire flow as shown below.



## 9. MANIPULATING WINDOWS CONNECTED TO QUERIES

In instances where queries derive from windows, the same procedures applied for stream-query.



The form generated for the query settings derives its 'from- stream' from the window.

### Properties

Pass-through Query

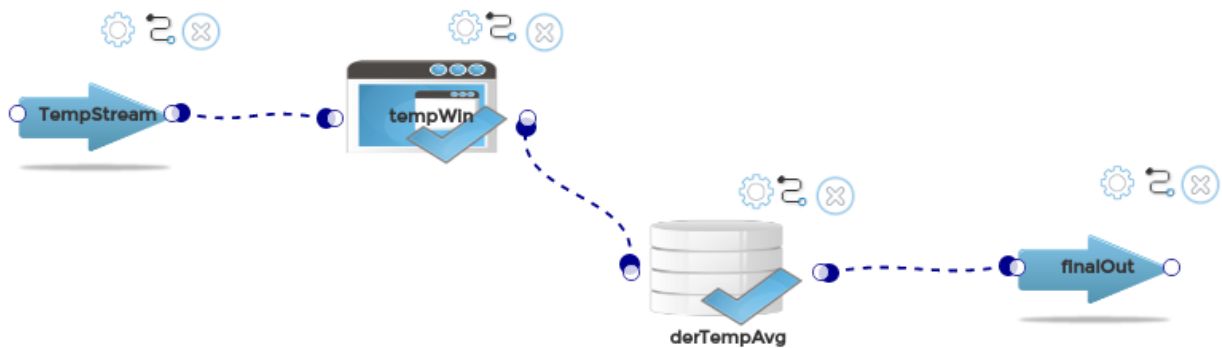
Query name:

from:

Select :

<input type="text" value="value"/>	as	<input type="text" value="3_attr1"/>
<input type="text" value="unit"/>	as	<input type="text" value="3_attr2"/>
<input type="text" value="scale"/>	as	<input type="text" value="3_attr3"/>
<input type="text" value="roomNo"/>	as	<input type="text" value="3_attr4"/>
<input type="text" value="occupants"/>	as	<input type="text" value="3_attr5"/>

insert into:



```
{
  "node": [
    {
      "id": "1",
      "class": "streamdrop ui-draggable",
      "position": {
        "top": 171.1112281860352,
        "left": 63.10764,
        "bottom": 243.1112281860352,
        "right": 183.10764,
        "predefinedStream": "Stream1",
        "name": "TempStream",
        "kind": "import"
      },
      "index": 2,
      "name": "tempWin",
      "filter": "No filter Defined",
      "attributes": [
        {
          "attrName": "value",
          "attrType": "3_attr1",
          "attrName": "unit",
          "attrType": "3_attr2",
          "attrName": "scale",
          "attrType": "3_attr3",
          "attrName": "roomNo",
          "attrType": "3_attr4",
          "attrName": "occupants",
          "attrType": "3_attr5"
        }
      ],
      "intoStream": {
        "index": 3,
        "name": "finalOut"
      }
    },
    {
      "id": "2",
      "class": "wstreamdrop ui-draggable",
      "position": {
        "top": 157.20487281860352,
        "left": 291.1806075292969,
        "bottom": 230.60487281860352,
        "right": 410.5806075292969,
        "name": "tempWin",
        "fromStreamIndex": 0,
        "fromStreamName": "TempStream",
        "kind": "window"
      }
    },
    {
      "id": "3",
      "class": "squerydrop ui-draggable",
      "position": {
        "top": 269.0972373205564,
        "left": 496.0938155371094,
        "bottom": 340.4972373205567,
        "right": 616.4938155371094,
        "name": "derTempAvg",
        "fromStream": {
          "index": 2,
          "name": "tempWin"
        },
        "filter": "No filter Defined",
        "attributes": [
          {
            "attrName": "value",
            "attrType": "3_attr1",
            "attrName": "unit",
            "attrType": "3_attr2",
            "attrName": "scale",
            "attrType": "3_attr3",
            "attrName": "roomNo",
            "attrType": "3_attr4",
            "attrName": "occupants",
            "attrType": "3_attr5"
          }
        ],
        "intoStream": {
          "index": 3,
          "name": "finalOut"
        }
      }
    },
    {
      "id": "4",
      "class": "streamdrop ui-draggable",
      "position": {
        "top": 277.1007163244629,
        "left": 760.1041915136718,
        "bottom": 349.1007163244629,
        "right": 880.1041915136718,
        "predefinedStream": "Stream3",
        "name": "finalOut",
        "kind": "export"
      }
    }
  ],
  "connections": [
    {
      "connectionId": "con_9",
      "pageSourceId": "1-Outinport",
      "pageTargetId": "2-in"
    },
    {
      "connectionId": "con_20",
      "pageSourceId": "2-out",
      "pageTargetId": "3-in"
    },
    {
      "connectionId": "con_38",
      "pageSourceId": "3-out",
      "pageTargetId": "4-Inexport"
    }
  ]
}
```

## 10. CONSTRAINTS

The constraints in the current version of this system are listed as follows.

- In creating Windows, the stream from which a window derives always needs to be created before the window itself.  
*Reason: In regenerating the model, the connections and properties are redrawn in order of their creation/ID. So if the stream from which the window was created doesn't exist on the canvas at the moment, the window cannot be practically considered to be valid.*
- Dropping new elements on the canvas appends them in different places on the canvas instead of the place where the mouse drops it on.  
*Reason: The mouse offsets passed to the method that drops the element, gets the offset values based on the entire page/browser and not the container alone.*
- Forms are created dynamically based on pure html. For UI enhancements, Bootstrap/ Angular schema forms could be used.
- All the data about elements dropped on the container are stored in *multidimensional arrays* and not objects.  
*Reason: Since this data is intermediate and only the final outcome is utilized, up to the point where the Json output is generated for the complete execution plan, all data are stored in arrays and only taken to JS objects during the time of saving it to generate the Json output.*
- In instances where the execution plan could be expected to span out to be large, a zoom feature would be preferable whereas it is not present in the current version.
- The predefined stream data are currently being retrieved from an array within the JS itself. This needs to be taken from the server.
- When dropping a stream element, the form to set its properties is displayed while making the toolbox and the container disabled. But, during this time, the existence of a dummy stream element on the container was preferred to show what kind of element is being set at the moment. Once the initial properties are set and the container is enabled again, the stream element can be appended with its set properties.