Visual Dataflow Introduction

Event dataflow programming is significantly different from standard structured programs in that there are independent processing components (nodes) which pass event data (events) from one node to the next node in a sequence. Applications are built by assembling these node building blocks into processing flows. While not appropriate for any kind of application, event flows are often a convenient and quick way to assemble high-level components without traditional programming.

Dataflow programming is a programming paradigm that models a program as a directed graph of the data flowing between operations. Applications are built by assembling building blocks into processing flows.

Visual Dataflow programming allows the assembly of dataflow programs using a visual UI.

VDAB, which stands for **Visual Dataflow Application Builder**, provides a powerful framework for visually assembling dataflow components. These processing components or nodes can be distributed among any number of computers running VDAB.

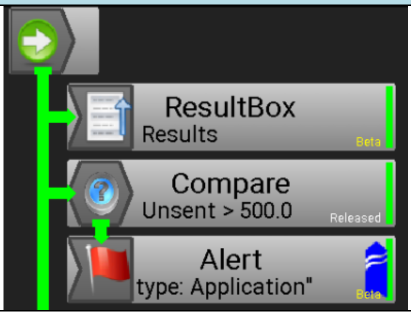
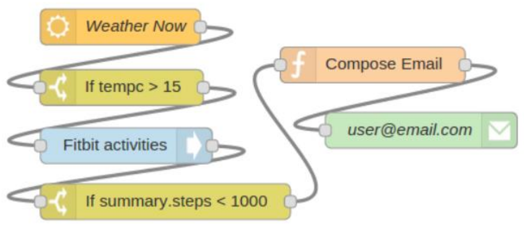
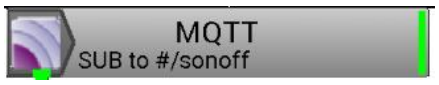
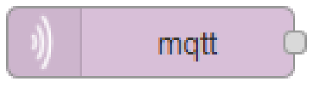
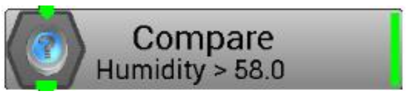
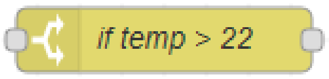
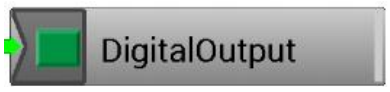
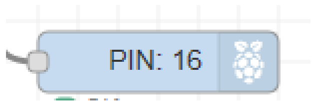
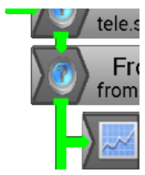
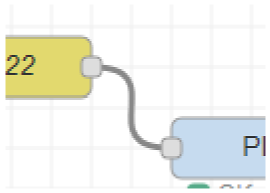
This document introduces some of the basic concepts of data flow programs and describes how they are created with VDAB.

Before This	This guide has no prerequisites.

Contents

Key Concepts

Node-RED and VDAB are both examples of visual dataflow languages. Both include **nodes** which are assembled to build **flows.** Data in the form of **events** passes between nodes in the flow on paths representing

Platform	VDAB	Node-RED
Flows	 ResultBox Results Beta Compare Unsent > 500.0 Released Alert type: Application" Beta	 Weather Now If tempc > 15 Fitbit activities If summary.steps < 1000 Compose Email user@email.com
Node (Source)		
Node (Conditional)		
Node (Target)		
Path		

The following key points should be considered:

- With the exception of source nodes, nodes sit and listen for events and only do work when an event (or special trigger event) arrive telling them to start working.

- The node processes the incoming event and depending on the type of node, may create a new event or events as a result of the incoming event.

- Data always passes from one processing component to another with events. For situations where data must be shared, VDAB introduces the concept of a data pool.

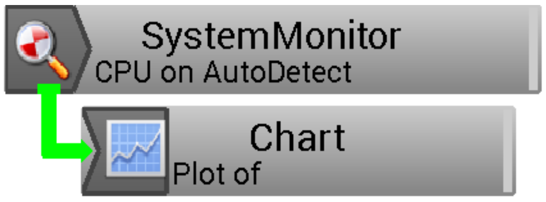
VDAB Terminology

While VDAB share some key terminology with Node-RED and other dataflow languages. There are some useful distinctions that may or may not apply more widely. The glossary below includes definitions that are common to the industry along with some specific to VDAB.

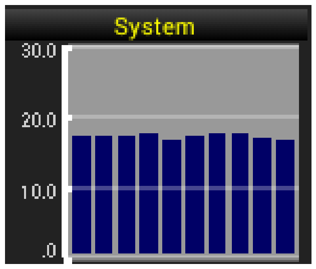
Term	VDAB Definition
Event	A data item passed between nodes that includes. • Source • Timestamp • Data
Trigger	A specialize type of event with no data. Used to pass control to the next node in the sequence
Node	A processing component that becomes active when it receives an event or trigger and completes processing by creating one or more events.
Source Node	A node that does not handle events but generates event periodically or through the actions of an external actor.
Target Node	A node that processes events but does not produce any events.
Function Node	A node that receives event data performs a local operation which is the result of that operation and publishes result data.
Service Node	A node that receives event data, invokes a local or remote service to process the data and return with result data from that service. Unlike a function node, the service is dependent on outside systems and may fail.
Visual Node	A node that has a visual representation such as a chart, switch or indicator.
Application	One or more flows that work together for a complementary purpose.
Pool Nodes	With VDAB data can be shared using named data pools that can be included in the flow.

Simple Monitoring Flow

The simple VDAB flow illustrates the concepts of a *flow*, *source node* and *target node*. It also demonstrates that something very useful can be built without programming using just a couple of



1. The *SystemMonitor source node* is set to poll for system utilization information and publishes that information as an *event*.
2. The *event*, which includes system and user CPU utilization, is pushed on the green *path* to all nodes that are listening.
3. In this case only the *Chart target node* is listening. When it receives the data it keeps it in a local data set and plots the data.
4. The *Chart target node* is a *visual node* and has a visual representation on the Web and Android Client.


System
30.0
20.0
10.0
.0