# **How to retrieve run-time parameters of a dell Boomi integration process?**

>> Using groovy script.

* 1. Below are the properties which are retrieved by the process:
     1. Process execution ID.
     2. Parent process execution ID.
     3. To identify if the given process is child process or not.
     4. Process name.
  2. Data process shape with custom scripting can be used to call the groovy script.
  3. Notify shape is used to display the values of the property.
  4. Use of DPP to store and display Execution properties.

/\*

This script retrieves the process execution ID, parent process execution ID, process name and indicates whether the process is a sub-process or not?. It then sets a user defined process property for each document passed into the process.

Documents are passed through the script unmodified.

\*/

import java.util.Properties;  
import java.io.InputStream;  
import com.boomi.execution.ExecutionManager;  
import com.boomi.execution.ExecutionUtil;

// Get process name of the boomi integration process.  
def processName = ExecutionManager.getCurrent().getProcessName();

// Get execution ID of the boomi integration process.  
def executionID = ExecutionManager.getCurrent().getId();  
  
// Get Parent process execution id from a subprocess.  
// Note if the process itself a Parent process then, parent execution id will be the same as execution id  
  
String parentExecutionId = ExecutionManager().getCurrent().getTopLevelExecutionId();  
  
// Check whether current process is child process or not.  
String isNested = ExecutionManager().getCurrent().isNested();  
  
for(int i=0; i<dataContext.getDataCount(); i++)  
{  
InputStream is = dataContext.getStream(i);  
Properties props = dataContext.getProperties(i);  
  
ExecutionUtil.setDynamicProcessProperty(“ExecutionID”, executionId, false);  
ExecutionUtil.setDynamicProcessProperty(“ProcessName”, processName ExecutionUtil.setDynamicProcessProperty(“ParentExecutionID”, parentExecutionId, false);  
ExecutionUtil.setDynamicProcessProperty(“IsChildProcess”, isNested, false);

dataContext.storeStream(is,props);  
}

# **How to gracefully handle HTTP responses from a RESTful web service server inside a Boomi process?**

>> Check “Return HTTP Responses”.

1. After enabling Return HTTP responses, the server response will be stored a “Document Property”. Else, the program will end abruptly on error.
2. We can log the error if required.
3. Steps to log error:
   * 1. Use “**Decision Shape**” right after HTTP client connector.
     2. Inside shape, configure – **Document property > Base > Application Status Code**.
     3. You may put “**Matches Regular Expression**” in comparison field.
     4. Second value can be: Static value of 20[0|1|2]

# **Common Myths about Dell Boomi Molecule.**

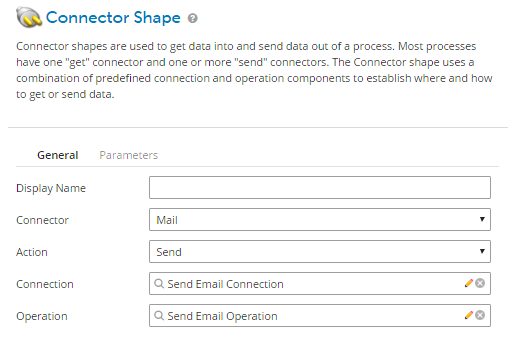
1. Myth #1: Molecule load balances real-time executions.
   * 1. Molecule automatically load balances scheduled executions, and not real-time web service requests.
2. Myth #2: Molecule handles mid execution failover.
   * 1. A process failure does not automatically resume/continue on another node.
3. Myth #3: Molecule is faster than atom and it has two or more atoms clustered together.
   * 1. Molecule are not inherently faster at an individual execution on single node level, though overall throughput can be greater as more than one node is working at the same time.
4. Myth #4: Molecule can automatically process large datasets in parallel.
   * 1. Yes it is possible, but it’s a conscious decision using a “Flow Control” shape.

# **Integrate GMAIL with Dell Boomi Mail Connector.**

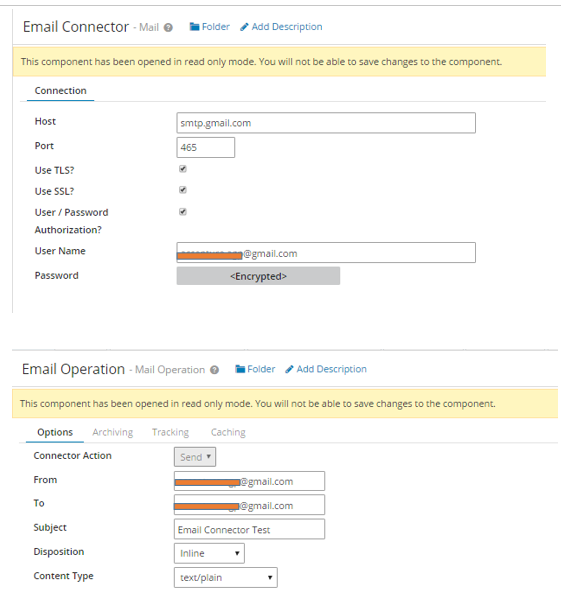
## **Send email using the Boomi Process.**

>> Please follow below steps:

1. Configure the “Mail Connector” as below:



1. Configure the connection(host: smtp.gmail.com | port: 465 | Use TSL: Checked | Use SSL: Checked), and operation as below:



1. “**From**”, “**To**”, and “**Subject**” can also be set using the “**Set** **Properties**” shape and **Document Property** before the email connector, so that the value can be passed dynamically during the execution of the process.

## **Get/Receive email in the Boomi process using GMAIL.**

* 1. Drag the email connector and set values as below:

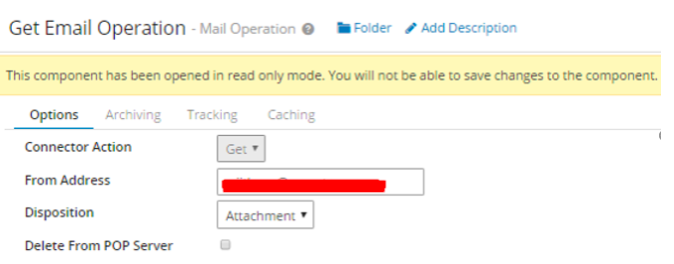
A screenshot of a cell phone

Description automatically generated

* 1. Configure the connections as:
     1. Host: pop.gmail.com;
     2. Port: 995.
     3. Use TSL: Un-checked.
     4. Use SSL: Checked.
  2. To receive email from GMAL, please make below changes in your GMAIL account.

A screenshot of a cell phone

Description automatically generated



# **All about MDM (Master Data Management) in Dell Boomi.**

## **What is MDM? What is the need for MDM?**

1. MDM is used to ensure uniformity, accuracy, stewardship, semantic consistency, and accountability of enterprise’s official shared master data assets.
2. Master data is consistent and uniform set of identifiers and extended attributes that describes the core entities of enterprise including customers, prospects, citizens, suppliers, sites, hierarchies, and chart of accounts.

## **Organizational challenges.**

1. Organisations face challenge while evolving from simple IT systems to a hybrid, heterogeneous ecosystem.
2. During this process they tend to loose track of data and thereby data accuracy decreases day by day.
3. There’s no “Single source of truth” for key assets like, Customer, Contacts, Products etc.

## **How Boomi MDM Solves the Problem?**

1. Main purpose of MDM is to ensure that the organization data is consistent across all systems.
2. This is done by maintaining high accuracy and quality at a centralized repository which is to be considered a single source of truth.
3. The integration between heterogeneous systems are fulfilled by “**Hub-Spoke Model**”.
4. Dell boomi master data hub allows the implementation of the MDM solution in 4 different architectural ways:
   * 1. **Consolidation** **style**: In this there are systems that contribute the data to MDM repository.
     2. **Registry** **style**: In this the master data resides in the source system.
     3. **Coexistence** **style**: In this data authoring is distributed, and golden records are in the master data hub.
     4. **Centralized** **style**: In this data authoring is done in the master data hub.