appviewX

Visual Workflow User Guide

Version: 2020.3.0

Copyright AppViewX, Inc.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

This document may not be copied, disclosed, transferred, or modified without the prior written consent of AppViewX, Inc. While all content is believed to be correct at the time of publication, it is provided as general-purpose information. The content is subject to change without notice and is provided “as is” and with no expressed or implied warranties whatsoever, including, but not limited to, a warranty for accuracy made by AppViewX. The software described in this document is provided under written license only, contains valuable trade secrets and proprietary information, and is protected by the copyright laws of the United States and other countries. Unauthorized use of software or its documentation can result in civil damages and criminal prosecution.

Trademarks

The trademarks, logos, and service marks displayed in this manual are the property of AppViewX or other third parties. Users are not permitted to use these marks without the prior written consent of AppViewX or such third party which may own the mark.

External Reference Links

This product includes software developed by the CentOS Project (www.centos.org).

This product includes software developed by Red Hat, Inc. (www.redhat.com).

This product includes software developed by VMware, Inc. (www.vmware.com).

All other trademarks mentioned in this document are the property of their respective owners.

Contact Information

AppViewX, Inc.

222 Broadway, FL 19

New York, NY 10038

Email: info@appviewx.com

Web: www.appviewx.com

Contents

Preface 10

Revision History 10

About this Guide 10

Audience 10

Text Conventions 10

Chapter 1. Module Overview 12

Solutions 12

Salient Features 12

Chapter 2. Prerequisites 14

Web Browser Requirement 14

Chapter 3. Getting Started with Visual Workflow 15

Visual Workflow Concepts 15

Chapter 4. Role Based Access Control 16

Overview 16

Resource 16

Role 19

User Group 27

User 29

Chapter 5. Visual Workflow Modules 32

Chapter 6. Visual Workflow Studio 33

Chapter 7. Getting Started with Prebuilt Workflow Tasks 36

Overview 36

FAQs - Connecting Workflow Tasks 36

How to Connect Form to Form 38

How to Connect Form to Script 46

How to Connect Script to Form 51

How to Connect Script to Script 55

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Auto-generate Forms..... 60
Sample Workflows using Prebuilt Tasks..... 64
Form with Email Notification..... 64
Executing a Ping Check using Command Task..... 70
Creating A Record on Infoblox device using prebuilt tasks..... 76
Creating a VIP with Incident Ticket..... 81
Application Delivery Automation..... 96
F5 BIG-IP Solutions..... 96
AVI Solutions..... 121
Automate DNS Services..... 130
Infoblox Solutions..... 130
Bluecat Solutions..... 133
Change Automation..... 140
Incident Management..... 141
Change Management..... 142
ServiceNow CMDB..... 142
DevOps and Continuous Configuration Automation..... 143
Ansible..... 143
Ansible Tower..... 147
Jenkins..... 148
GitOps Integration..... 149
GitLab..... 149
GitHub..... 149
Utilities..... 150
Collection..... 150
Pagerduty and ChatOps..... 151
Approval and Notification..... 152
Scheduled Follow-Ups..... 153
SCP and SFTP..... 154

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

SDK Automation....155
Chapter 8. Workflow Inventory....158
Overview....158
Workflow Inventory Actions....162
Cloning a Workflow....162
Enabling a Workflow....164
Disabling a Workflow....166
Deleting a Workflow....168
Exporting a Workflow....170
Workflow Inventory Menu....171
Design....174
Store....177
Import....184
Hooks....186
Magic Variables....191
Helper Script....195
Regex Library....205
Source Control....207
Integration Hub....223
Python Library....252
Chapter 9. Workflow Tasks....257
Overview....257
Folders....258
Task Category - General....261
Ansible Executor....263
Break Loop....264
Delay....265
Dependency....266
If....268

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Implementation....270
Join....271
Loop....272
Postvaildation....276
Prevalidation....277
REST....278
REST(I)....287
Retry....288
Rollback....289
Schedule....290
Script....291
Split....326
Switch....326
WorkOrder....327
Task Category - User Interface....328
Chart....329
Command....339
Diff Checker....348
Form....357
Grid....441
Review....451
YAML....461
Task Category - ChatOps and Notifications....466
Email....467
Pagerduty....481
Slack....486
Skype....488
Task Category - Change Management....491
Integrations....494

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Postman - Visual Workflow Northbound Integration....495
Jenkins - Visual Workflow Northbound Integration....502
Terraform - Visual Workflow Northbound Integration....514
Ansible - Visual Workflow Northbound Integration....517
Openshift PaaS Orchestration....521
Ansible Southbound Integration....527
Automation Collision....547
Workflow Task Actions....549
Updating a Task....550
Cloning a Task....551
Deleting a Task....552
Saving and Reusing a Task....553
Input/Output Variable Mapping....561
Variable Mapping....562
Global Variables....565
Magic Variable....573
Rollback Workflow....573
Manual Rollback....574
Auto Rollback....577
Task Scheduler....581
Workflow Cart....584
Nested Workflows....584
Workflow Options....590
Version Control....590
Preview....594
Customizable Connectors....598
Workflow Alignment....602
History....604
Workflow Settings....605

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Validating a Workflow..... 609
Switching between Edit mode/Citizen mode..... 611
Connecting tasks in Workflow Studio..... 612
How to connect Form to Grid..... 612
How to connect Form to Grid to CSV..... 618
How to connect Form to Decision to Form..... 624
How to connect Script to Grid..... 629
How to connect Script to Review..... 634
How to connect Form to Create Ticket to Close Change Ticket..... 638
How to connect Form to Delay to Email..... 645
How to connect Form to Retry to Form..... 651
How to connect Script to Bar or Pie Chart..... 656
How to connect Script to Stacked Chart..... 661
How to connect Form with Loop and Printing..... 664
How to connect Form to Schedule to Script..... 669
Connecting Workflow Tasks with Failover - RGF Flow..... 672
Chapter 10. Visual Workflow Request..... 674
Overview..... 674
Accessing the Workflow Request Page..... 675
Workflow Dashboard..... 676
Request :: Overview..... 676
Custom Reports..... 678
My Workflows..... 679
Request :: View/Run..... 680
Scheduled Jobs..... 688
My Requests..... 689
Request :: All..... 690
Viewing the Workflow Log Summary..... 691
Viewing workflow details..... 692

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Pausing/Resuming a Workflow....693
Aborting a Workflow....694
Rollback a Workflow....695
Assigned Requests....696
Configuring RBAC for Assigned Requests....698
Audit....701
Settings....701
Selecting the Landing Page....702
Viewing the View/Run page in Classic mode....703
Managing the Catalogs....704
Archive Workflow Requests....706
Restore Workflow Archives....709
Configuring RBAC for Archive and Restore Requests....713
Chapter 11. Visual Workflow and Pages....715
....715

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Preface

Revision History

Revision	Description	Date
1.0	Initial release of document for Release 2020.3.0	September 2020

About this Guide

Welcome to the complete guide to getting started with Visual Workflow. Learn how to use Visual Workflow as a Network Service Orchestration and Automation platform with AppViewX modules, including working with inventory, connecting tasks and apps you know and designing workflows to automate repetitive tasks and solutions.

Audience

This guide aims to introduce basic visual workflow concepts and guides you through your automation journey. This guide is useful for experienced users with reasonable programming experience (Python), trying to enhance their skills with automating network tasks. You can use AppViewX pre-built workflow tasks, commands, API, SDK to automate across the L2-L7 layer, such as automate load balancers, DNS, WAF, PKI, switches, routers, and other network elements. This guide is intended for the following audience:

- DevOps
- Network Engineers
- Architect
- NetDevOps
- SceOps
- PKI

Text Conventions

The following text conventions are used in this document:

Convention	Description
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in the text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Convention	Description
`codeblock`	Indicates commands with a paragraph, URLs, codes in examples, text that appears on the screen, or text that you enter.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Chapter 1: Module Overview

AppViewX's Visual Workflow is a hybrid cloud automation and orchestration platform that allows for simple and complex network automation within your infrastructure.

Visual workflow supports a wide range of device types, vendors, actions and integrations, in order to help you manage and automate your entire network with a unified automation and orchestration platform.

Visual workflow's primary interface is through both the Web GUI interface and a HTTP Restful API. A developer would use this API for programmatic access.

AppViewX's Visual Workflow offers:

- Hybrid/Multi-cloud network orchestration and automation
- Single service platform with self-servicing Automation and API orchestration.
- Workflow Orchestration of system of records (SOR) across diverse application and infrastructure domains.
- DevOps, GITOps automation

Solutions

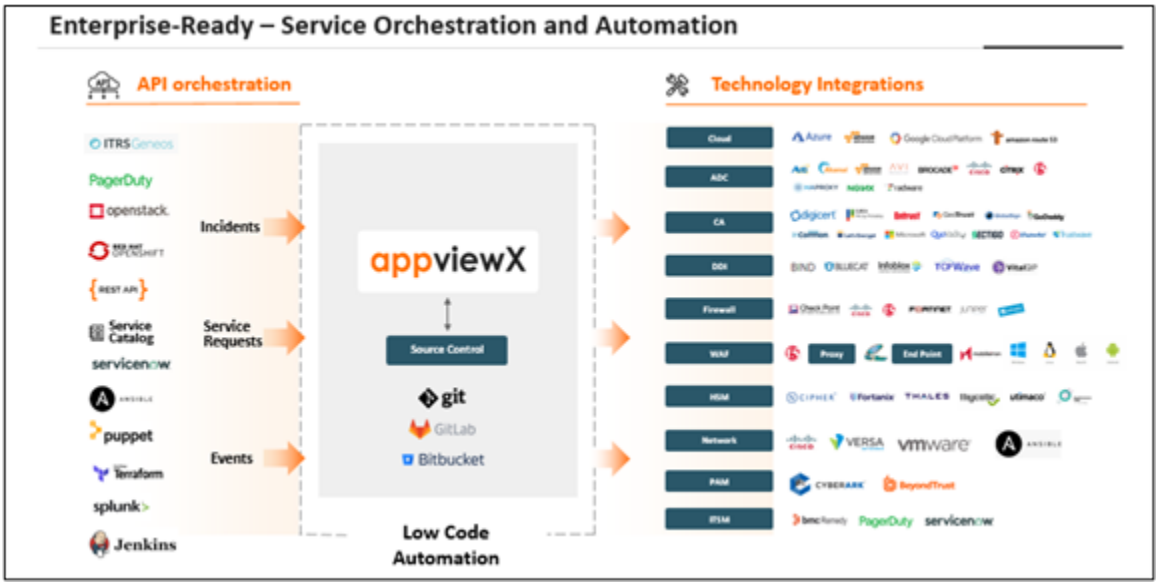
- Automate Application delivery and security (LB, DNS, WAF) services
- Automate Certificate Lifecycle Management (CLM)
- Automate Network Firewall changes
- Automate Cloud services
- Routers/switch automation
- Continuous compliance Automation
- Automate Vulnerability Detection (CVE) and Reporting
- Event driven automation, Closed loop remediation

Salient Features

- **Automate** network configurations and change automation
- **Automate** repetitive tasks for faster network changes and free up your time for more strategic initiatives
- **Automate** repeatable network patterns across operations and development using the same powerful and agentless automation platform
- Separate the workflow task execution from the execution layer (via AppViewX vendor modules)
- **Prebuilt tasks and solutions**: Automate application and security services quickly using out of the box automation tasks and solutions
- **Design** simple and complex workflows
- **Improve** agility, reduce operational costs, and automate repetitive tasks

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- **Programmable GUI**: Bring your own automation within a unified GUI-based framework
- **BPM (Business Process Automation)**: Streamline and map the business process, teams, and technology
- **Workflow tasks**: Objectivity of business logic is maintained through separate tasks
- **Reusability**: Offers reusability of workflow(s) and workflow task(s) for quick turnaround
- **Intuitive**: Offers control directly to the user to define a customized automation workflow
- **Rule Engine**: Custom Rule builder to aid event driven automation
- **Visual Representation**: Offers a visual representation of the entire automation and orchestration flow
- **Integration**: Integrate and automate with Multi-vendors (ITSM, ADC, PKI, DDI, DevOps, SDK, Open source tools)



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Chapter 2: Prerequisites

Web Browser Requirement

Browsers	Version
Internet Explorer	v11.0.9600.18817 or later
Firefox	v74.0.1 (64-bit) or later
Google Chrome	v85.0.4183.83 (64-bit) or later

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Chapter 3: Getting Started with Visual Workflow

Visual Workflow Concepts

- **Workflow Studio**: An inventory to design and manage automation workflows, design self-service reports (Build your own Reports), design self-service catalogs (Build your own Pages) for NetOps, SecOps, App teams and PKI admin.

  For more information, refer to the section on Visual Workflow Studio.

- **Tasks**: Tasks are the units of action (aka automation lego blocks) in Visual workflow. Tasks can be of various types depending on the nature of automation such as User Interface tasks, Scripts, YAML, REST API, or pre-built network automation tasks.

  For more information, refer to the section on Workflow Tasks.

- **Variables**: Workflow variables are properties that help define or describe the workflow activity required to perform per task. Variables provide the ability to store data (within a task) and can be referenced across any stage of the workflow automation process, thus enabling the flow of data seamlessly.

  For more information, refer to the section on Variable Mapping.

- **Nested Workflows**: An ordered list of tasks and workflows bundled together to enable their execution in an order repeatedly. Nested flows can include variables, tasks, and even a complete workflow.

  For more information, refer to the section on Nested Workflows.

- **Pre-built Task Library**: A folder consisting of pre-built automation tasks and solutions with device, vendor information, tags, and standard Input/Output variables defined.

  For more information, refer to the section on Getting Started with Prebuilt Workflow Tasks.

- **Role Based Access Control (RBAC)**: RBAC ensures access control permissions across tasks, workflows, permissions to one or more users, and user groups.

  For more information, refer to the section on RBAC.

- **Visual Workflow Request**: The Request console allows you to execute workflow(s). Workflows can be executed manually or can be scheduled.

  For more information, refer to the section on Visual Workflow Request.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Chapter 4: Role Based Access Control

- Overview
- Resource
- Role
- User Group
- User

Overview

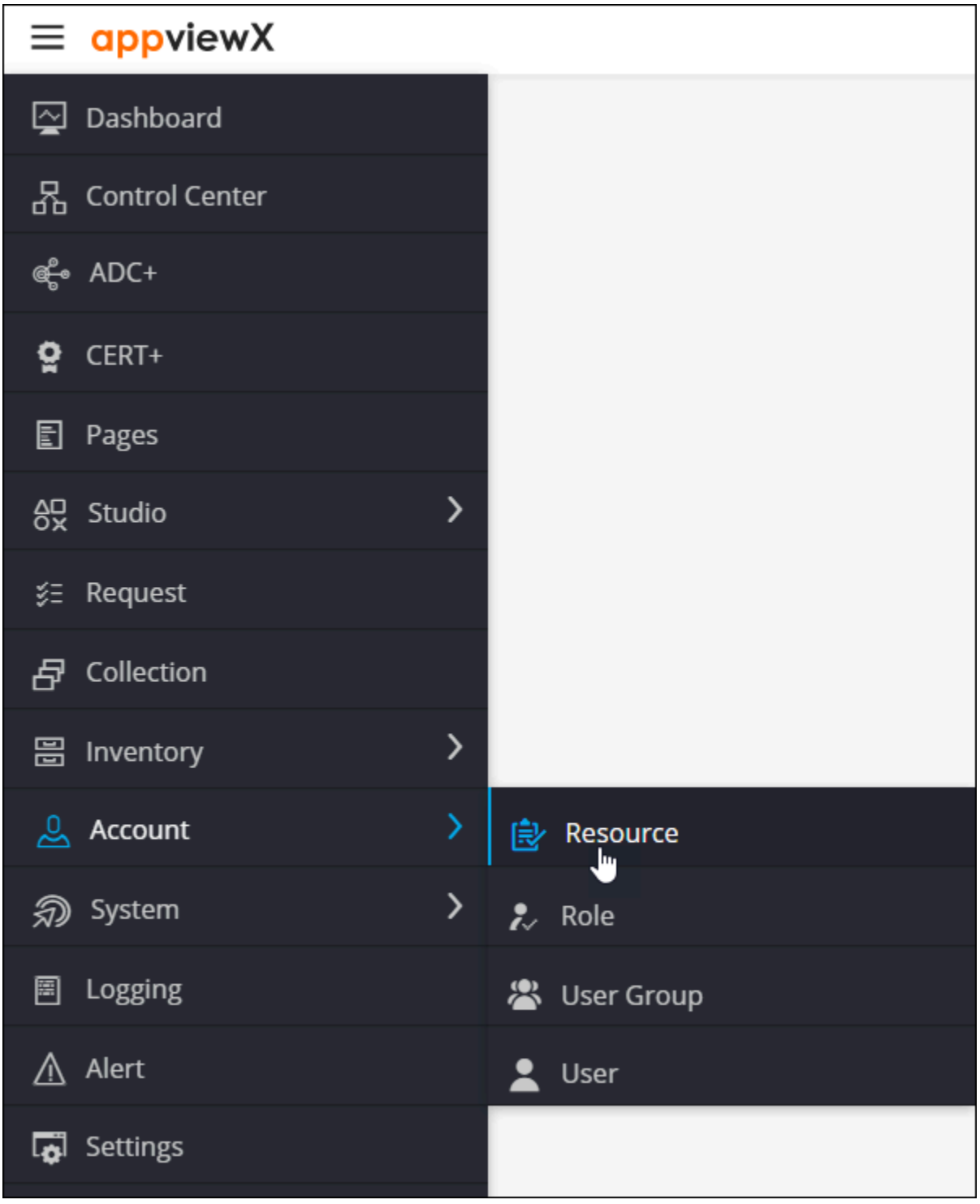
With role based access control (RBAC) you can assign permissions to users for accessing the module and allow/restrict them to perform certain actions.

Resource

To get an overview of the actions authorized for a particular resource:

1. From the upper left corner of the screen, click ☰.
2. From the menu displayed, select **Account** > **Resource**.

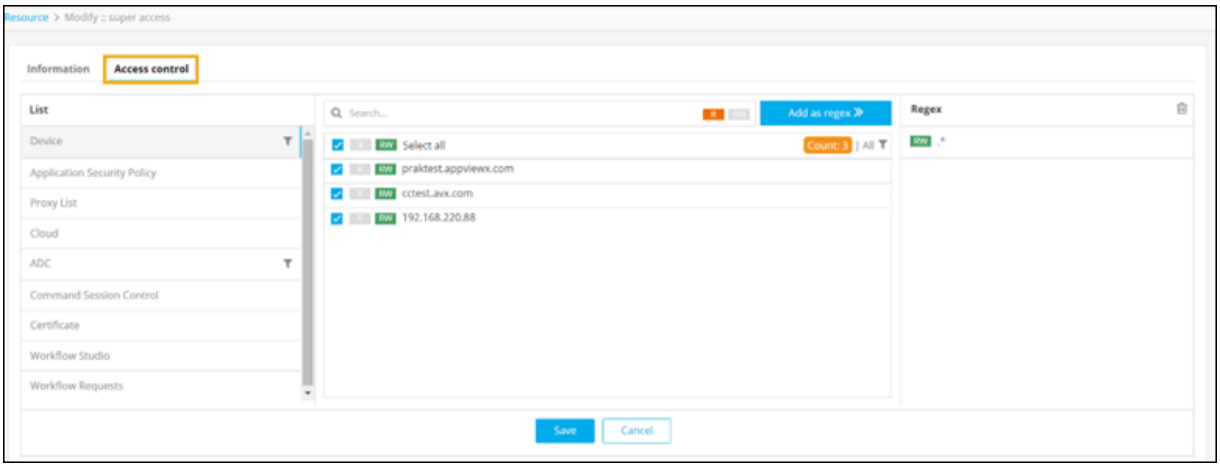
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



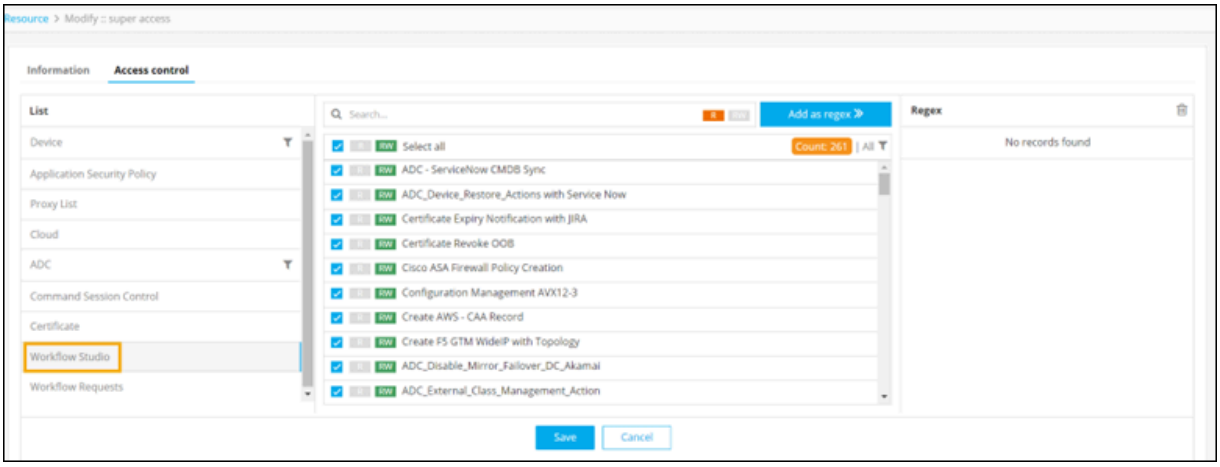
3. On the **Resource** page, click on the resource name.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

4. To see the actions authorized for this resource, click **Access Control**.



5. To see the permissions assigned to the resource in the Workflow Studio, select **Workflow Studio**.

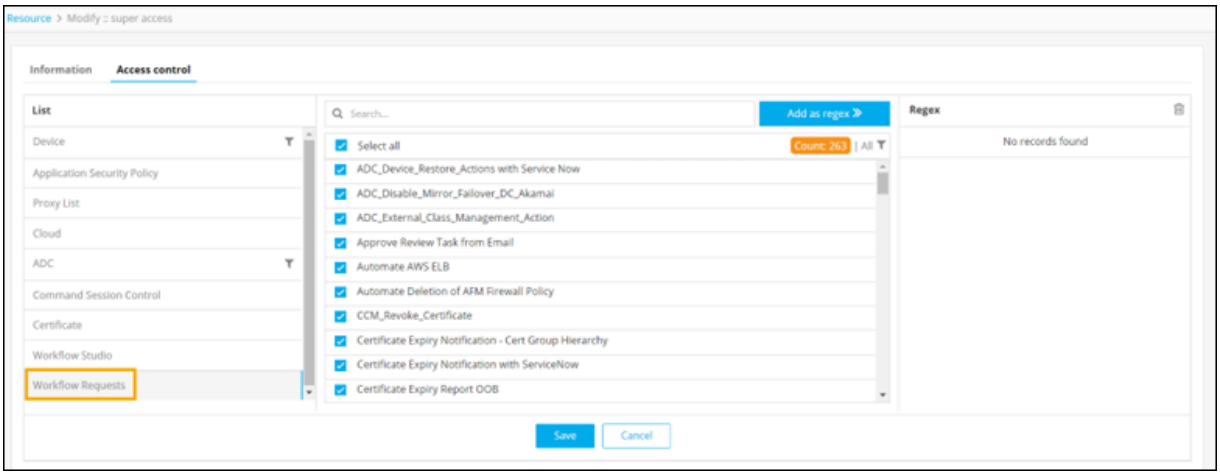


The following permissions can be defined:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Read (**R**): To allow only Read access to the resource, click **R** against the Workflow name. The resource will not be able to modify the workflow.
- Read/Write (**RW**): To allow a resource to view and modify a workflow, click **RW** against the workflow name.
- Select/Unselect: If you unselect a particular workflow, that workflow will not be available in the Workflow Studio Inventory.

6. To see the permissions assigned to the resource in the Requests console, select **Workflow Requests**.



Note: If you unselect a particular workflow, that workflow will not be available on the Workflow Request page.

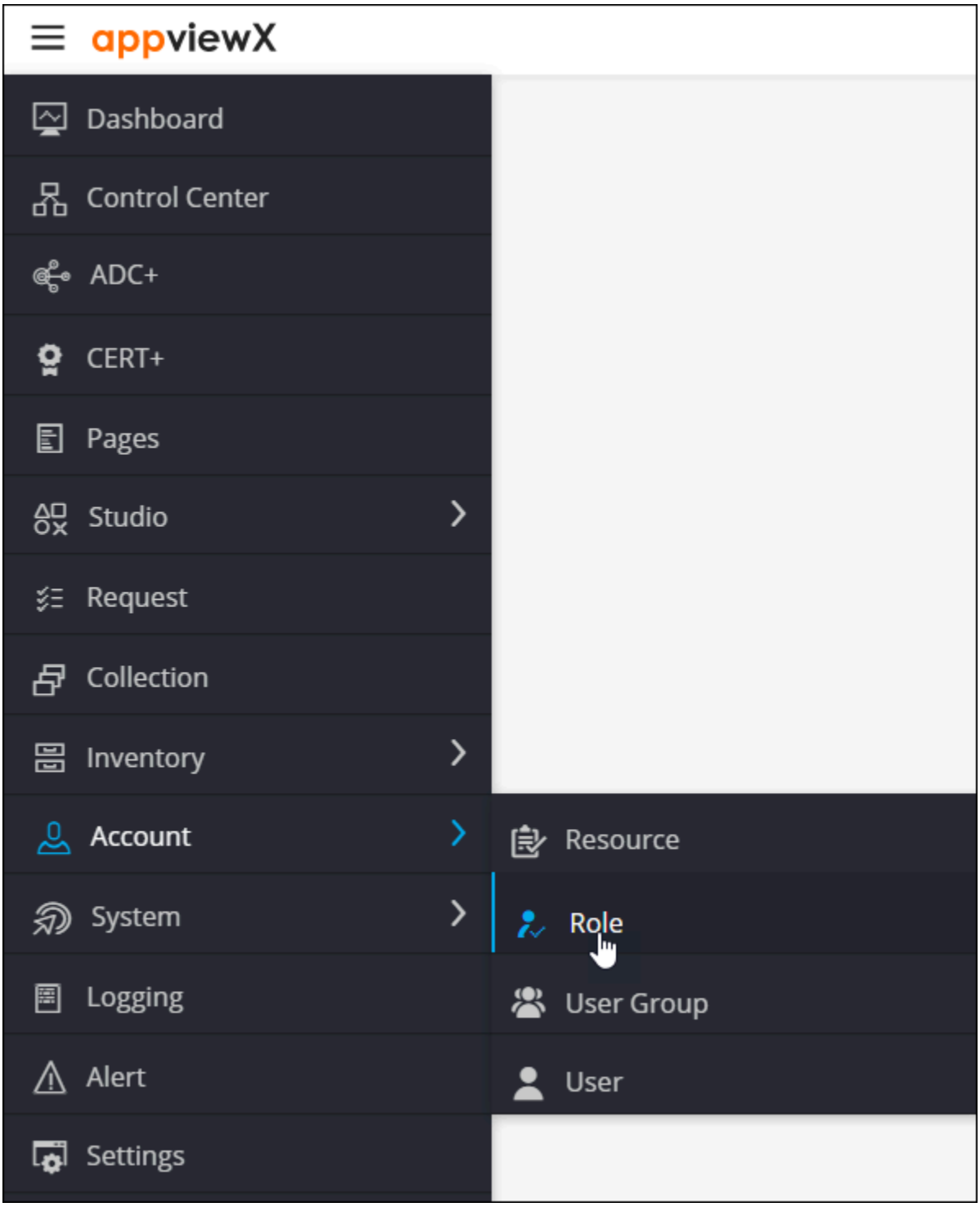
7. Click **Save**.

Role

To get an overview of the actions authorized to a particular resource:

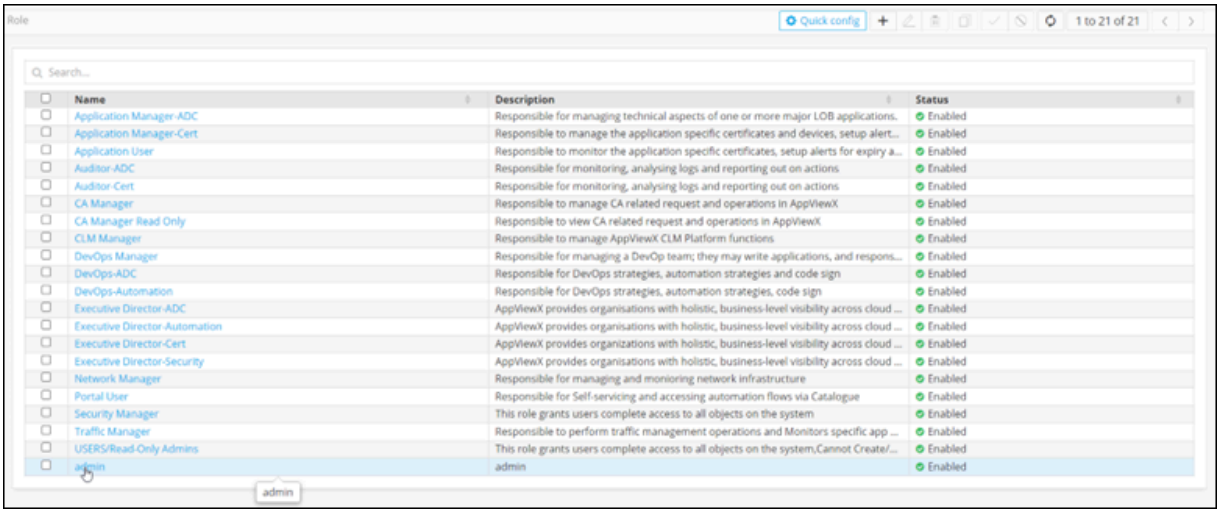
1. From the upper left corner of the screen, click ☰.
2. From the menu displayed, select **Account** > **Role**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

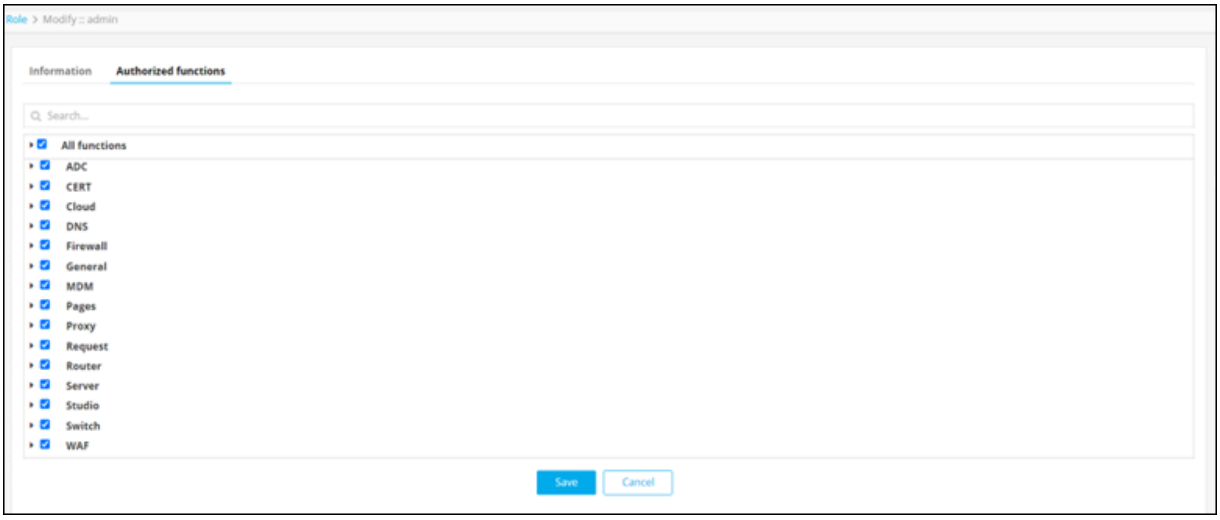


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

3. On the **Role** page, select any role from the list of roles, for example admin.

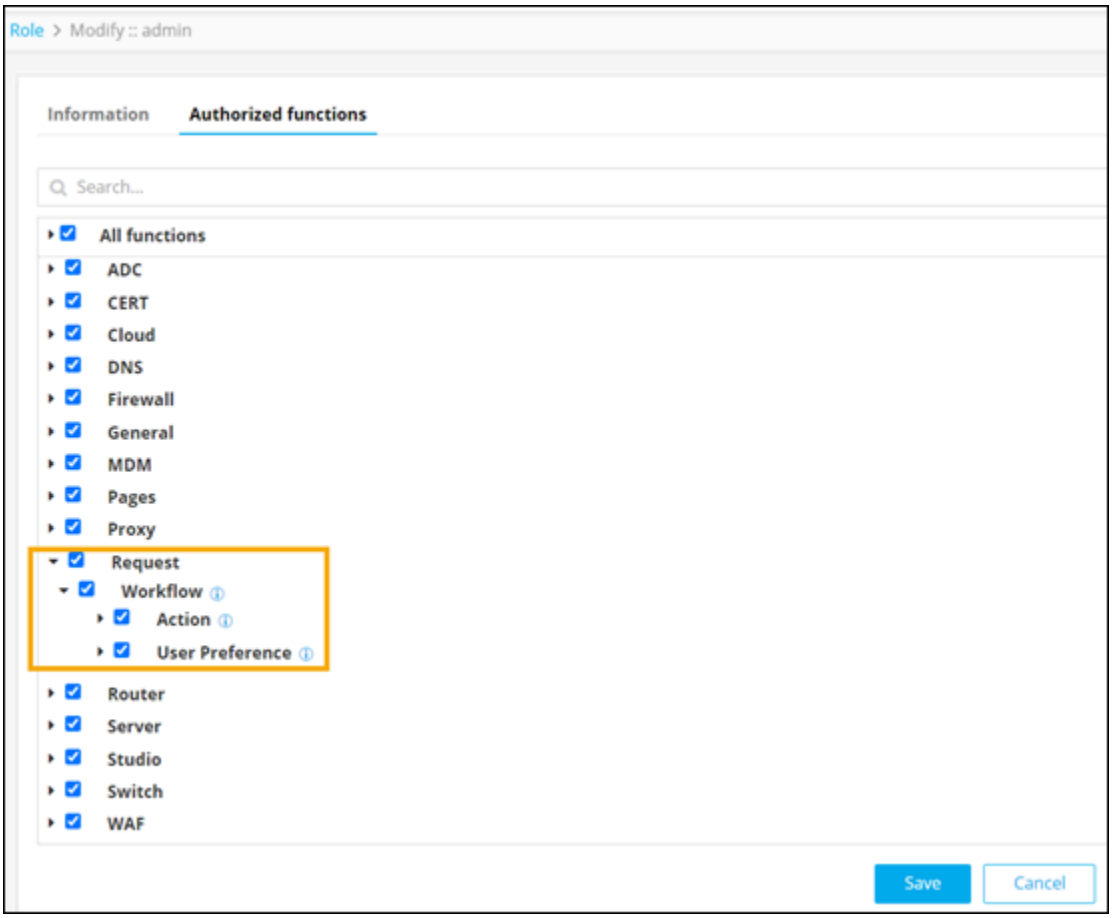


4. To see a list of functions authorized for this role, click **Authorized Functions**.



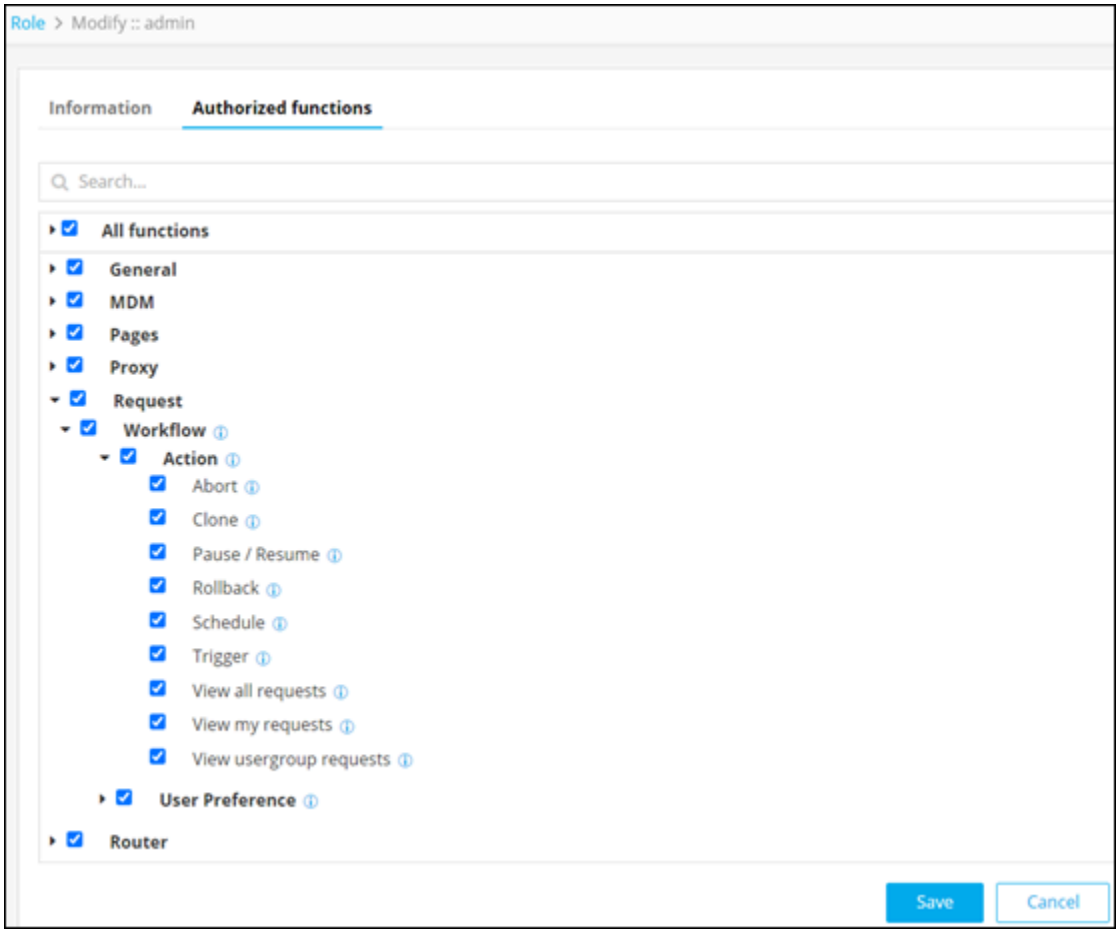
5. To see the actions authorized in the Request module, under **Request**, click **Workflow**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

• Permissions defined under **Action**.



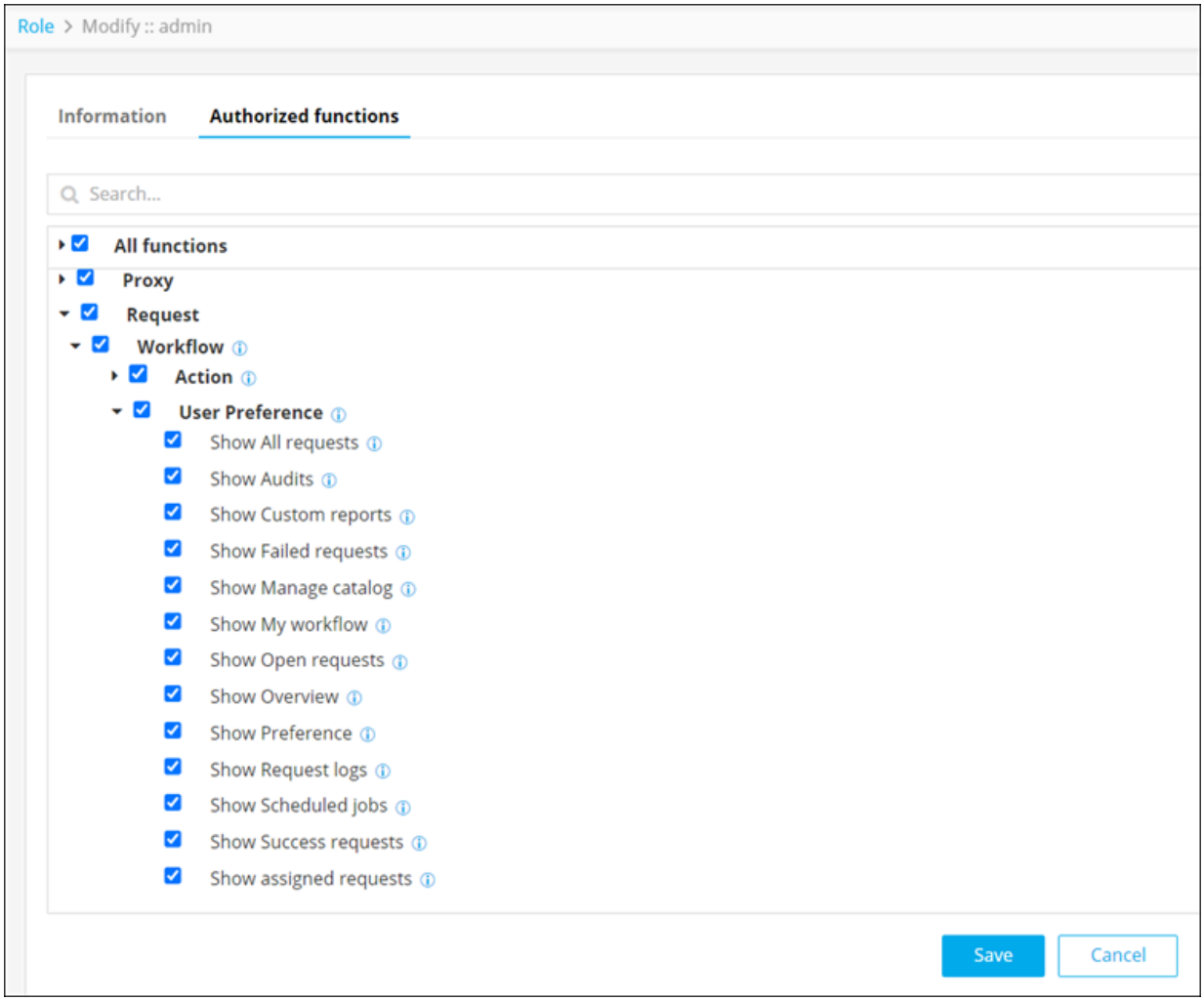
The following table describes the various actions that can be authorized or restricted for a user within workflows:

Action	Description
Abort	Provision to allow or restrict a user to abort a workflow request instantaneously.
Clone	Provision to allow or restrict a user to clone a workflow request with a different name.
Pause / Resume	Provision to allow or restrict a user to pause or resume a workflow request.
Rollback	Provision to allow or restrict a user to trigger an alternative rollback workflow that you have added or imported for a workflow.
Schedule	Provision to allow or restrict a user to schedule a specific time to initiate a workflow request.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Action	Description
Trigger	Provision to allow or restrict a user to trigger or submit a workflow request
View all requests	Provision to allow or restrict a user to view submitted workflow requests of all users.
View my requests	Provision to allow or restrict a user to view workflow requests initiated by the user.
View usergroup requests	Provision to allow or restrict a user to view workflow requests initiated by the usergroup.

- Permissions defined under **User Preference**.



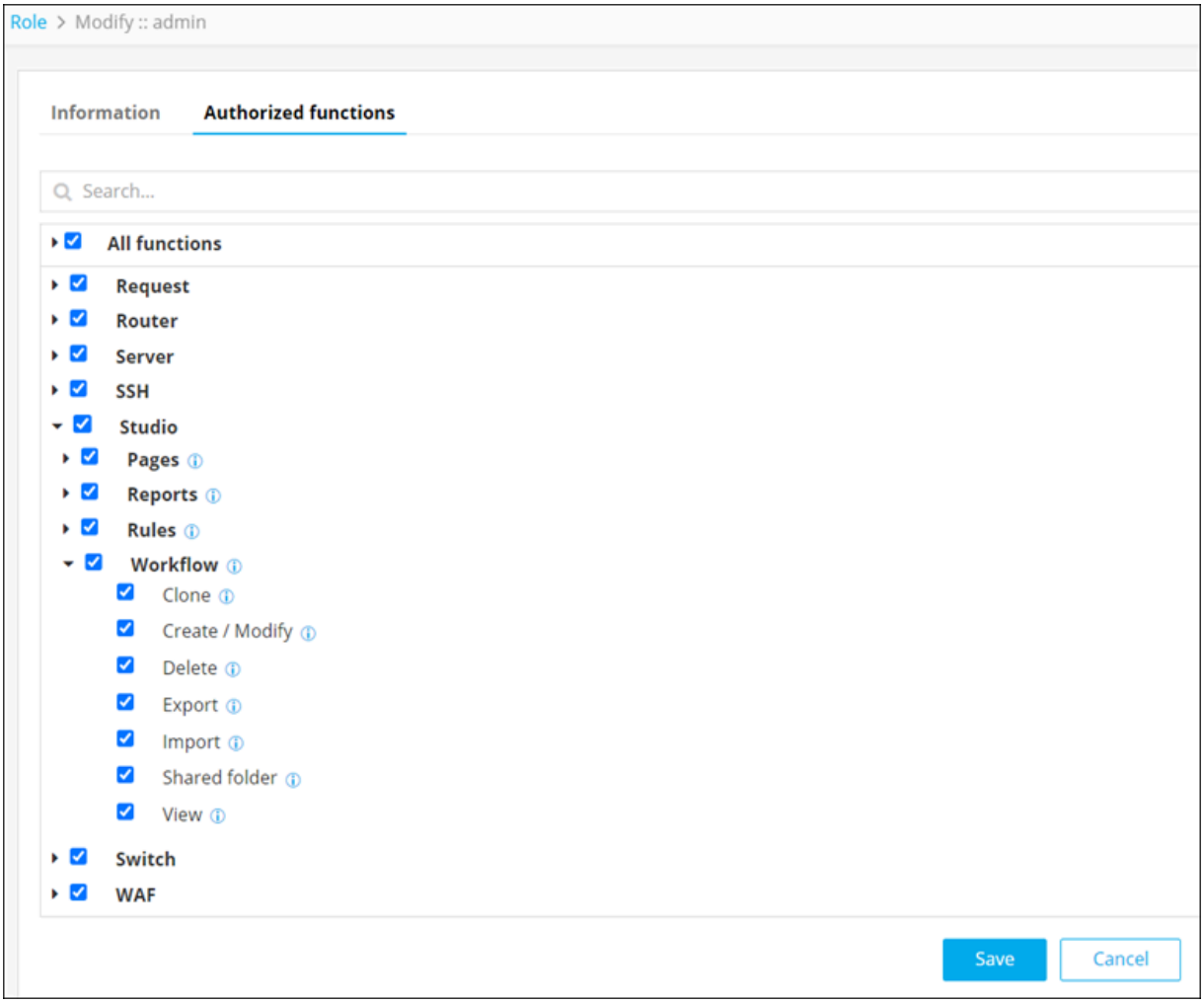
The following table describes the various actions that can be authorized or restricted for a user under **User Preference**:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Action	Description
Show All requests	Provision to show or hide All requests menu for a user.
Show Audits	Provision to show or hide Audit menu for a user.
Show Custom reports	Provision to show or hide Custom reports menu for a user.
Show Failed requests	Provision to show or hide Failed requests menu for a user.
Show Manage catalog	Provision to show or hide Manage Catalog menu for a user.
Show My workflow	Provision to show or hide My workflow menu for a user.
Show Open requests	Provision to show or hide Open requests menu for a user.
Show Overview	Provision to show or hide workflow dashboard Overview menu for a user.
Show Preference	Provision to show or hide the Request Landing preference menu for a user.
Show Request Logs	Provision to show or hide Request logs for a user.
Show Scheduled jobs	Provision to show or hide Scheduled jobs menu for a user.
Show Success requests	Provision to show or hide Success requests menu for a user.
Show Assigned requests	Provision to show or hide Assigned requests menu for a user.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

6. To see the actions authorized for the user in the Workflow Studio, under **Studio**, click **Workflow**.



The following table describes the various actions that can be authorised or restricted for a user in the **Workflow Studio** module:

Action	Description
Clone	Provision to allow or restrict a user to clone an existing workflow.
Create / Modify	Provision to allow or restrict a user to create or modify a workflow.
Delete	Provision to allow or restrict a user to delete an existing workflow.
Export	Provision to allow or restrict a user to export disabled workflows.
Import	Provision to allow or restrict a user to import workflow details as a .zip file.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Action	Description
Shared Folder	Provision to allow or restrict a user to share a folder that contains logically grouped workflows.
View	Provision to allow or restrict a user to view detailed information of workflows.

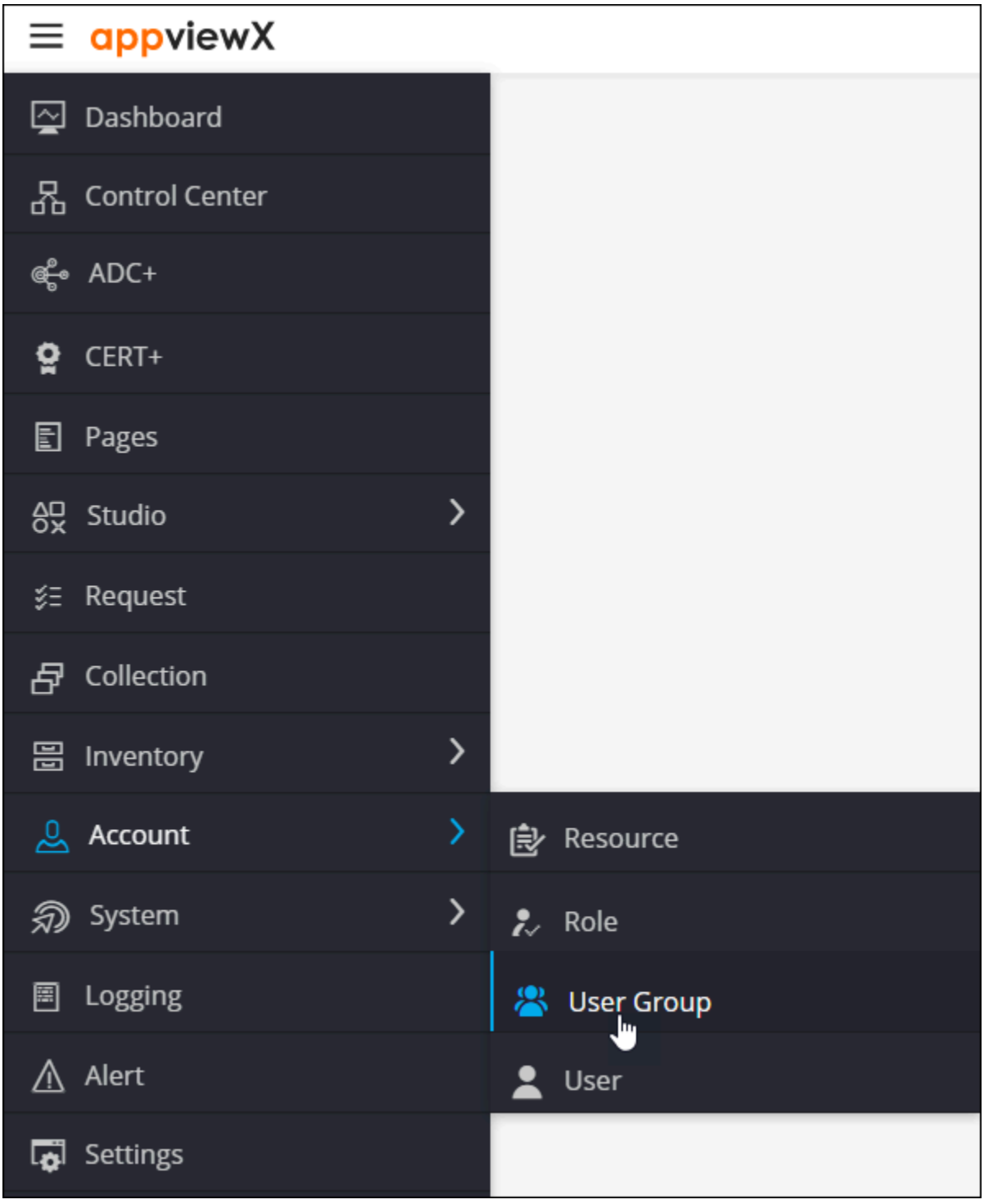
7. Click **Save**.

User Group

To see the details of a particular user:

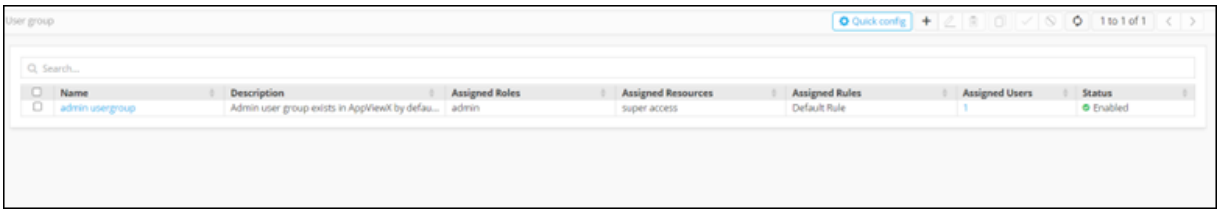
1. From the top left corner of the screen, click .
2. From the menu displayed, select **Account** > **User Group**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



User Group details are displayed.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

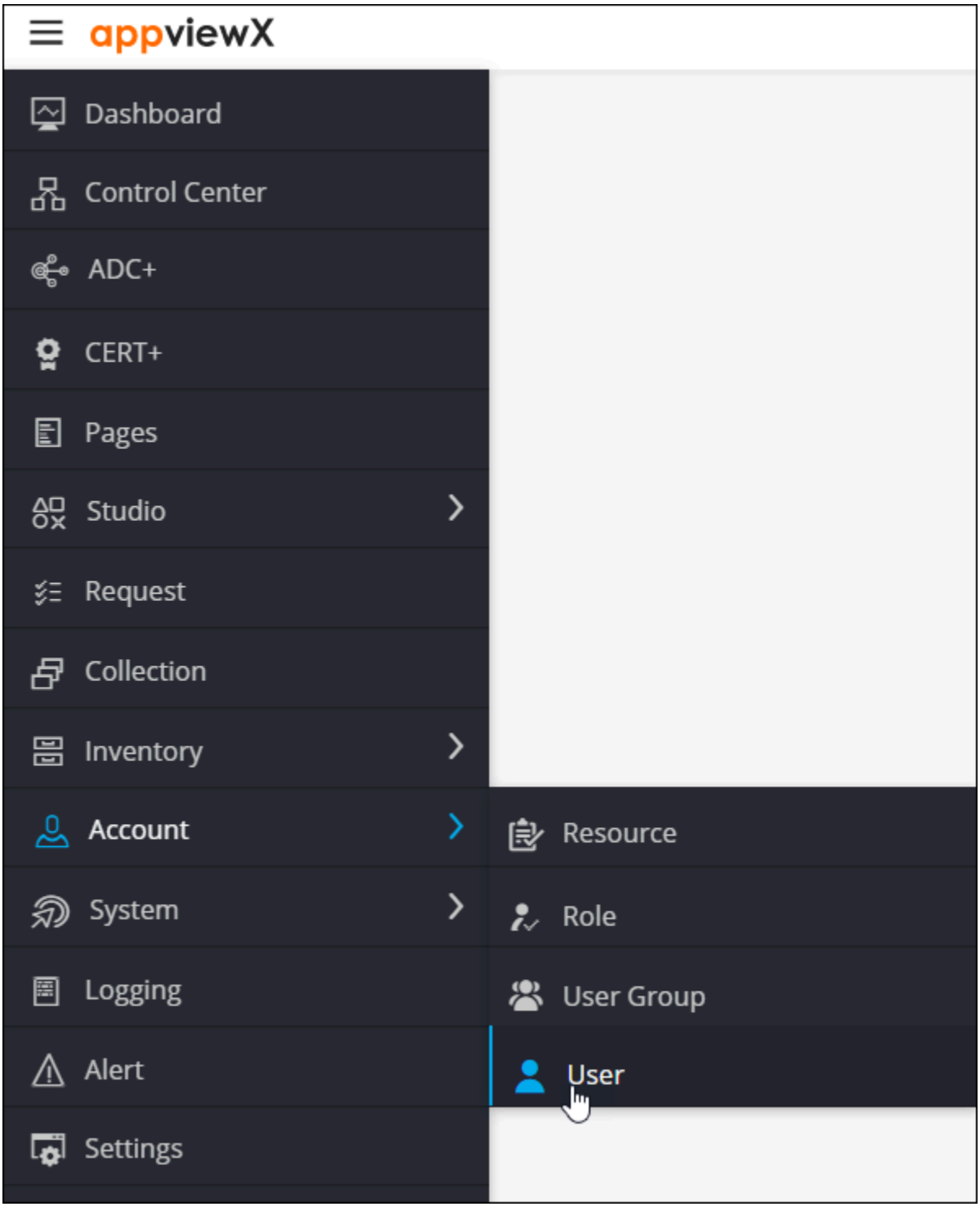


User

To see the details of a particular user:

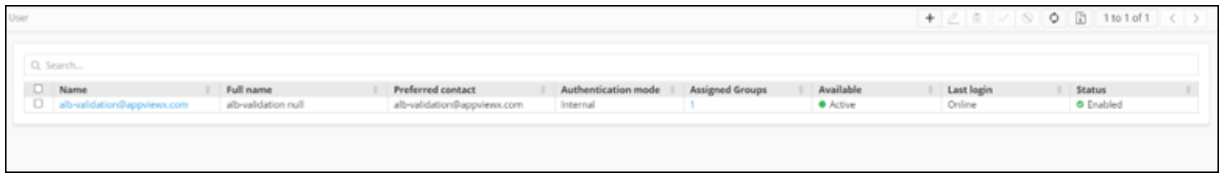
1. From the top left corner of the screen, click ☰.
2. From the menu displayed, select **Account** > **User**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



User details are displayed.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Chapter 5: Visual Workflow Modules

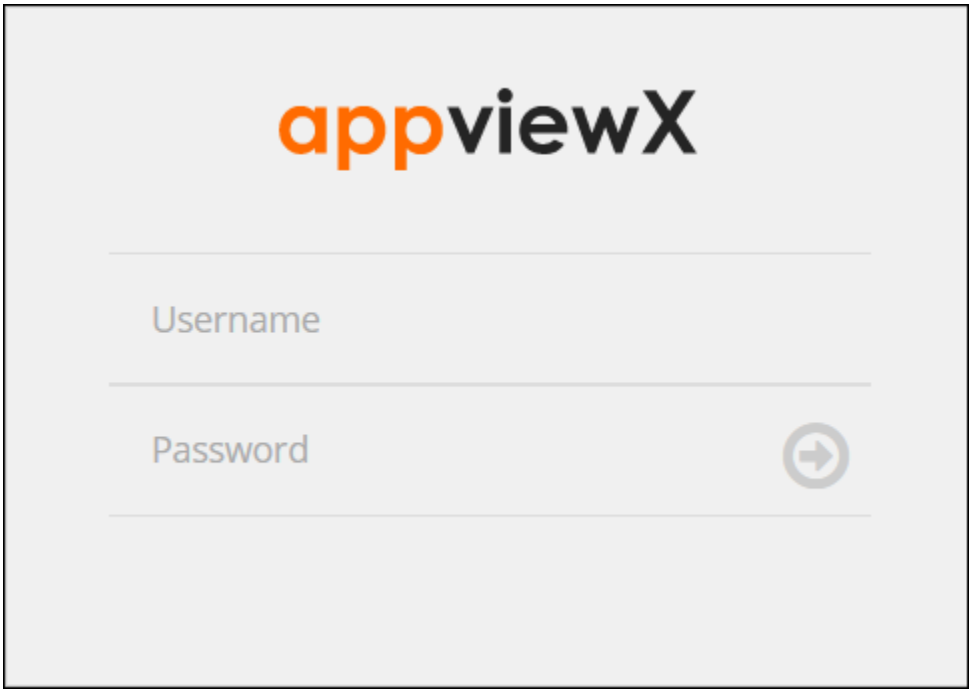
- **Studio**: A placeholder for you to design simple and complex workflow(s).
- **Request**: Request console allows you to execute workflow(s). Workflows can be executed manually or can be scheduled.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Chapter 6: Visual Workflow Studio

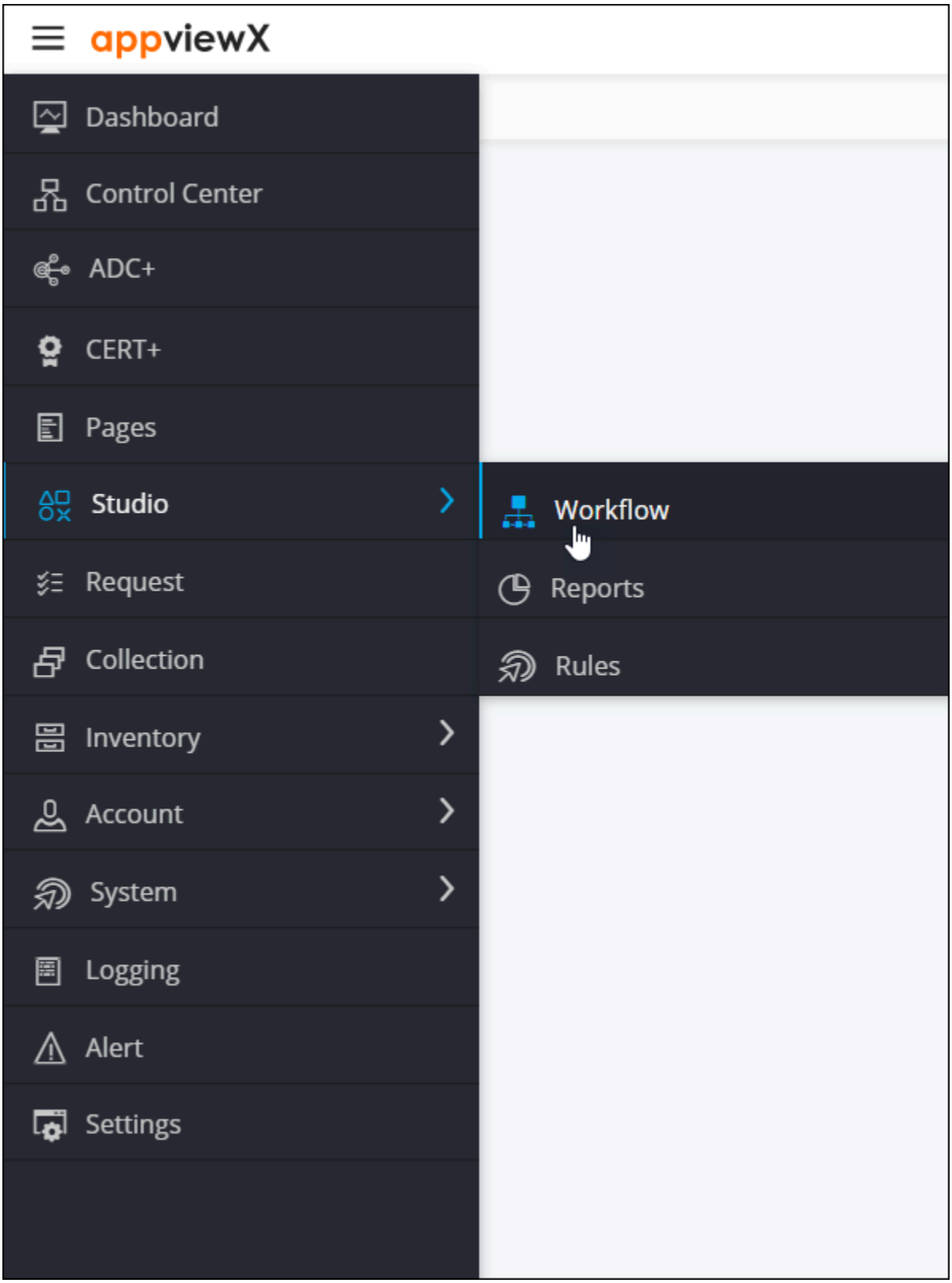
To access the Visual Workflow Studio module:

1. Log into AppViewX with valid credentials.



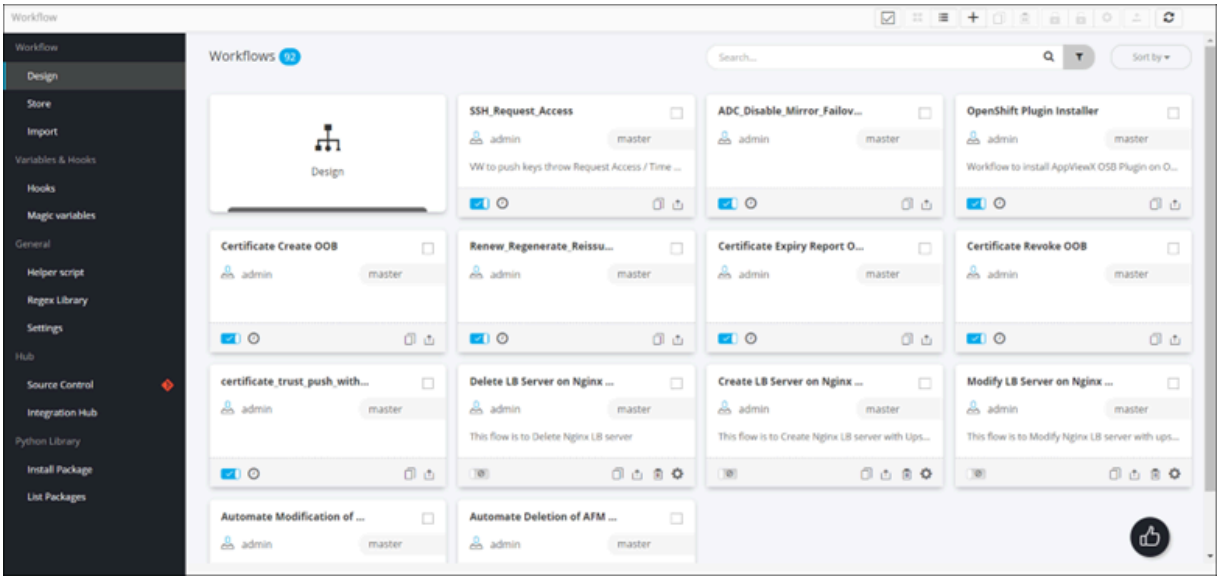
2. To access the navigation pane, hover the mouse over the ☰ icon.
3. From the menu displayed, select **Studio** > **Workflow**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

The **Workflow** inventory page is displayed.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Chapter 7: Getting Started with Prebuilt Workflow Tasks

- Overview
- FAQs - Connecting Workflow Tasks
- Auto-generate Forms
- Sample Workflows using Prebuilt Tasks
- Application Delivery Automation
- Automate DNS Services
- Change Automation
- DevOps and Continuous Configuration Automation
- GitOps Integration
- Utilities
- SDK Automation

Overview

In order to get you started with automation, the FAQs folder within the studio has some quick references to aid you in using simple tasks, variables and workflows, and basic workflow features.

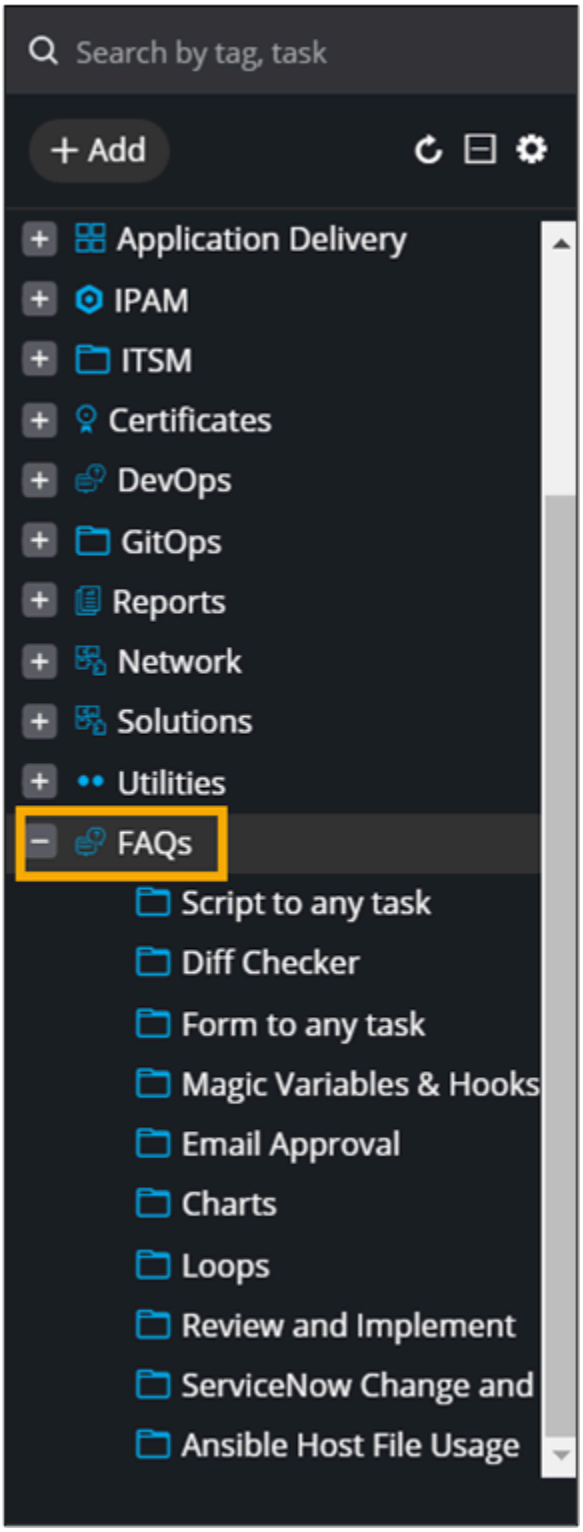
FAQs - Connecting Workflow Tasks

The FAQs folder contains quick reference workflows that will help you in designing and understanding workflow features. The FAQ workflows are available as part of the AppViewX Installation.

Following are a few sample flows that will help you learn how to connect tasks:

- Script to any task
- Form to any task
- Using Diff Checker
- Using Charts
- Using Loops
- Using Auto Email Approvals
- Using Hooks

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



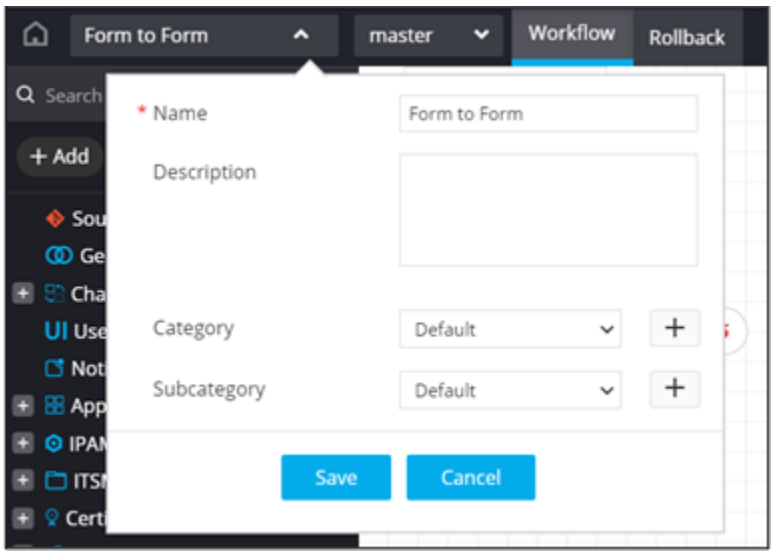
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- How to Connect Form to Form
- How to Connect Form to Script
- How to Connect Script to Form
- How to Connect Script to Script

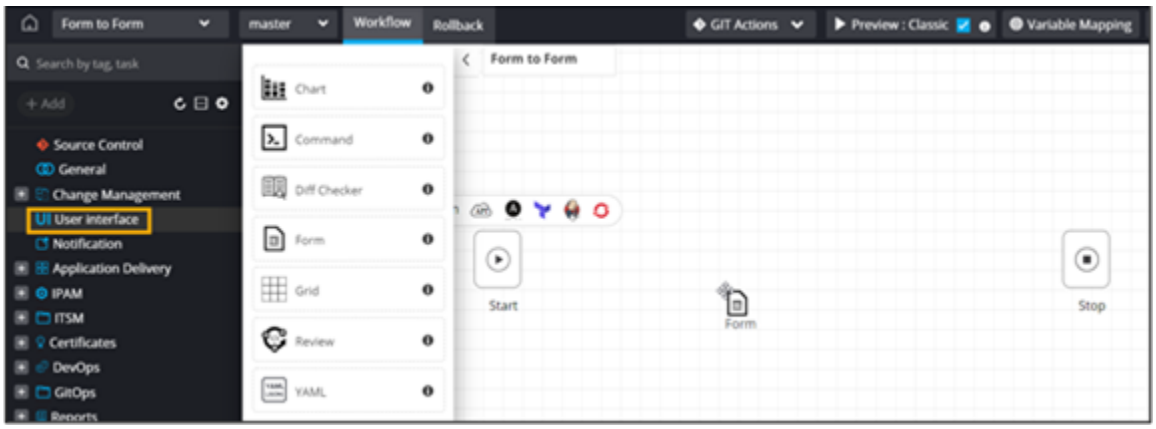
How to Connect Form to Form

You can design workflows to pass data between multiple form tasks using global variables.

1. Design a workflow.

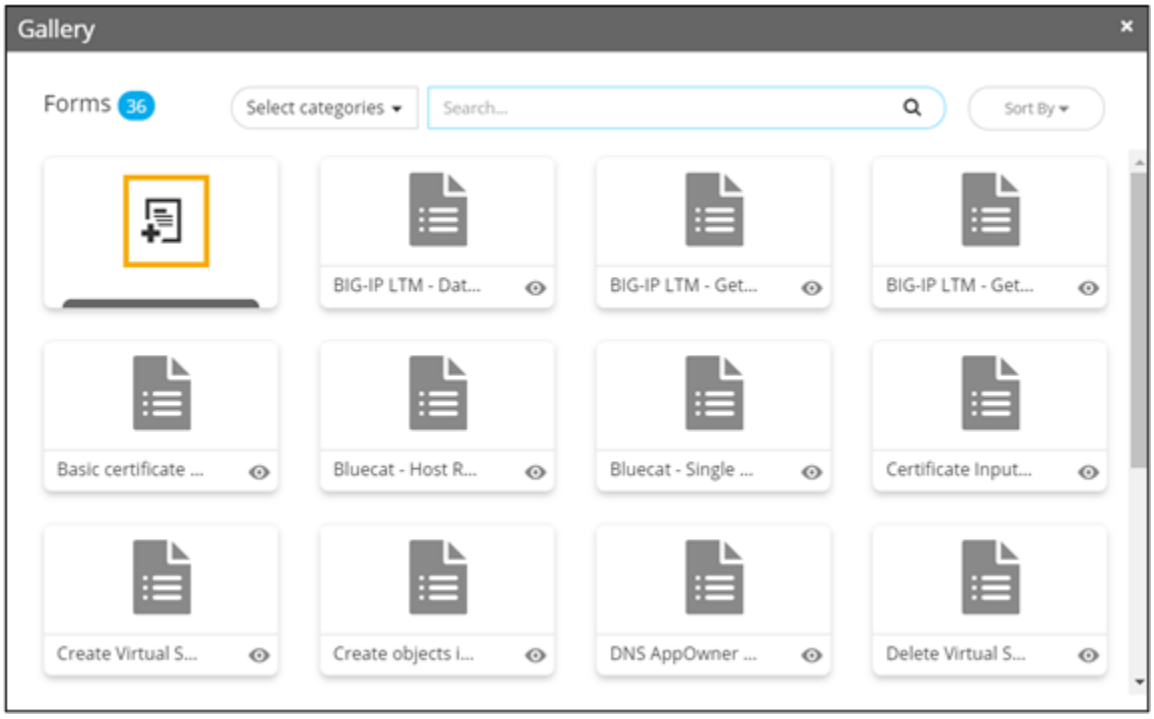


2. From the **User Interface** section, drag and drop the **Form** task.

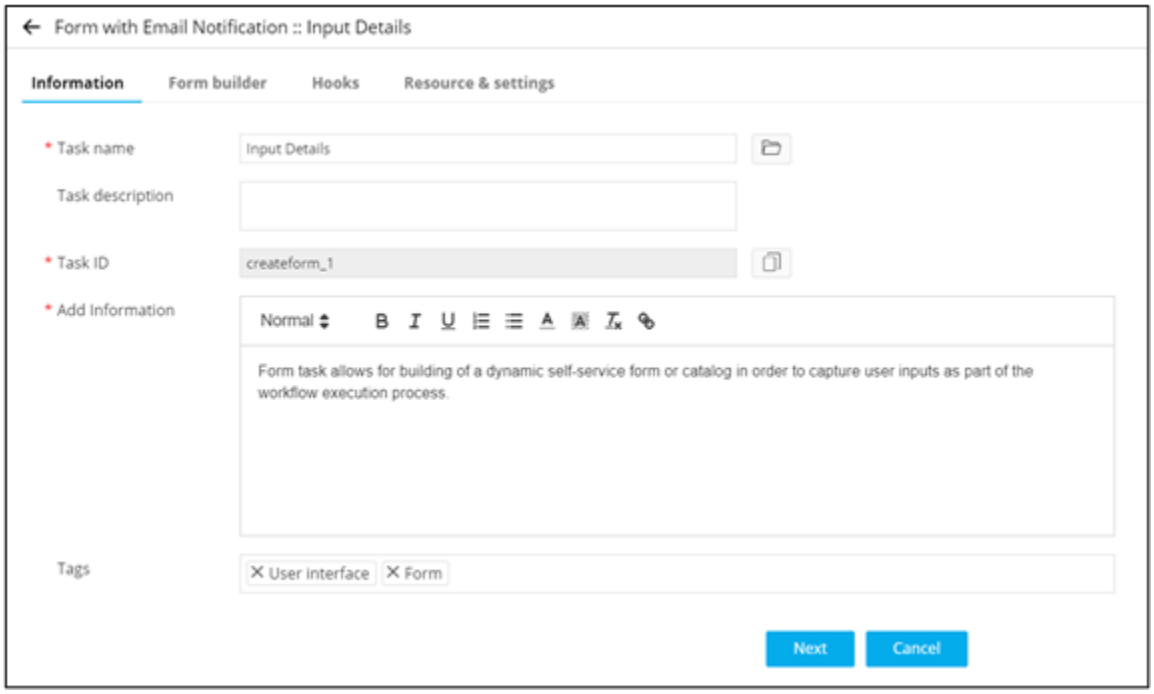


3. In the **Gallery** window, to design a new form, click .

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

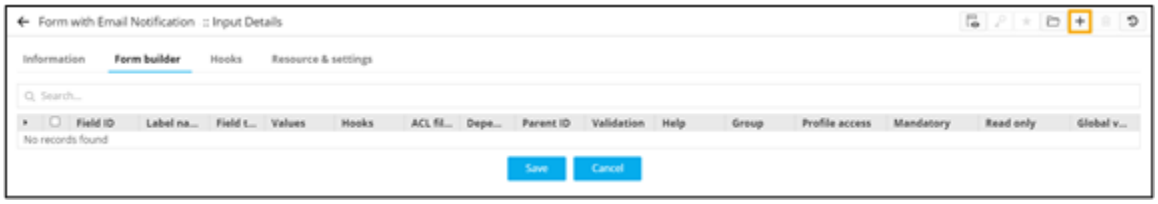


4. Under the **Information** tab, enter the **Task name**.

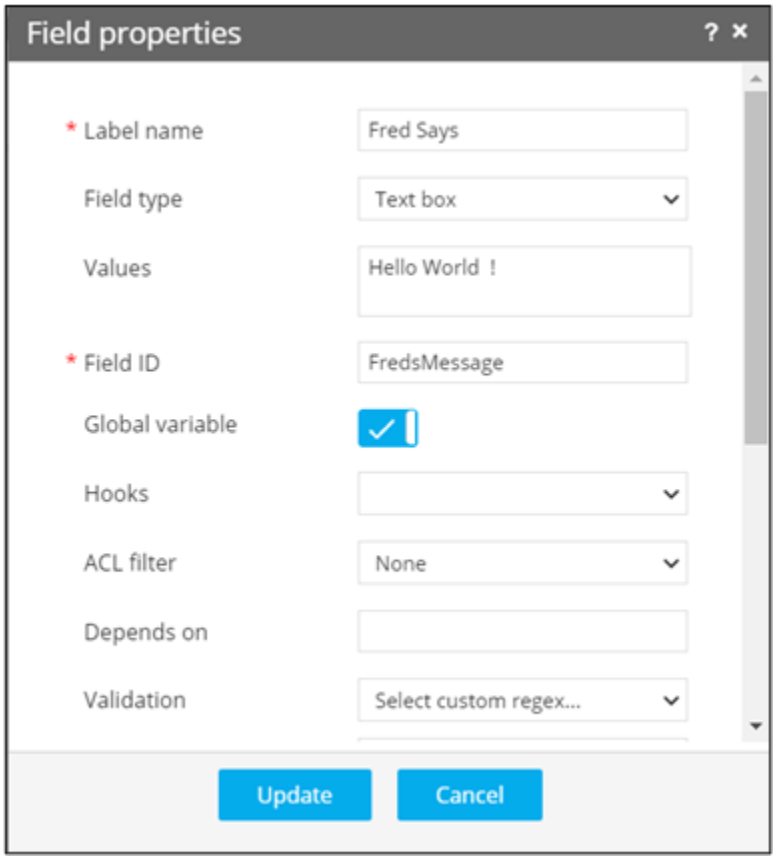


5. Click **Next**.
6. To add the form fields, under the **Form builder** tab, click +.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



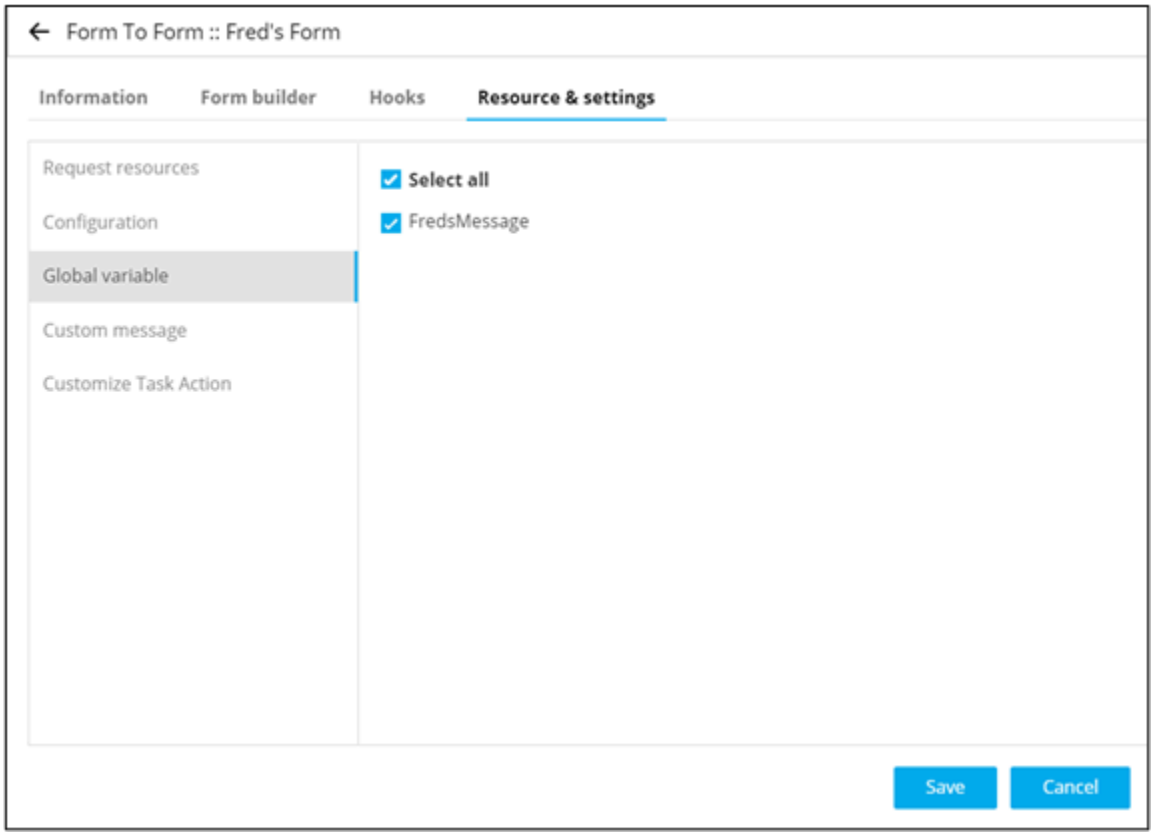
7. In the **Field properties** window, define the form field to get user inputs.



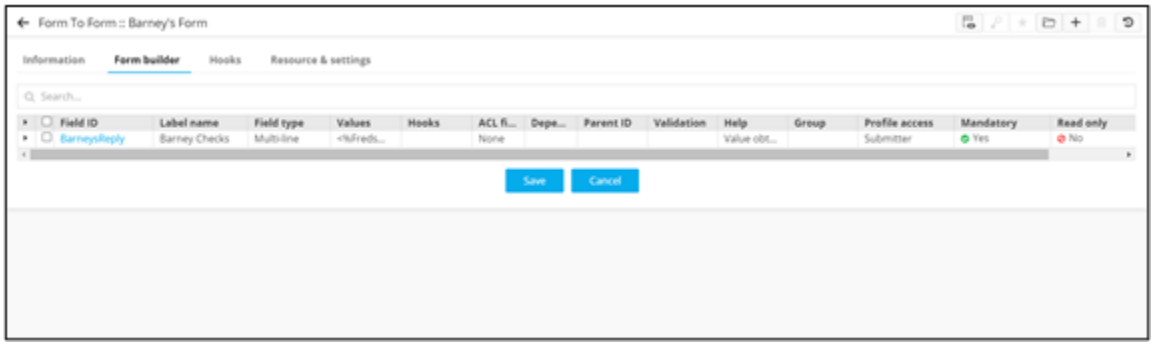
Note: For more information on adding form fields, click here.

8. Under the **Resource & Settings** tab, declare the **Global Variable**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

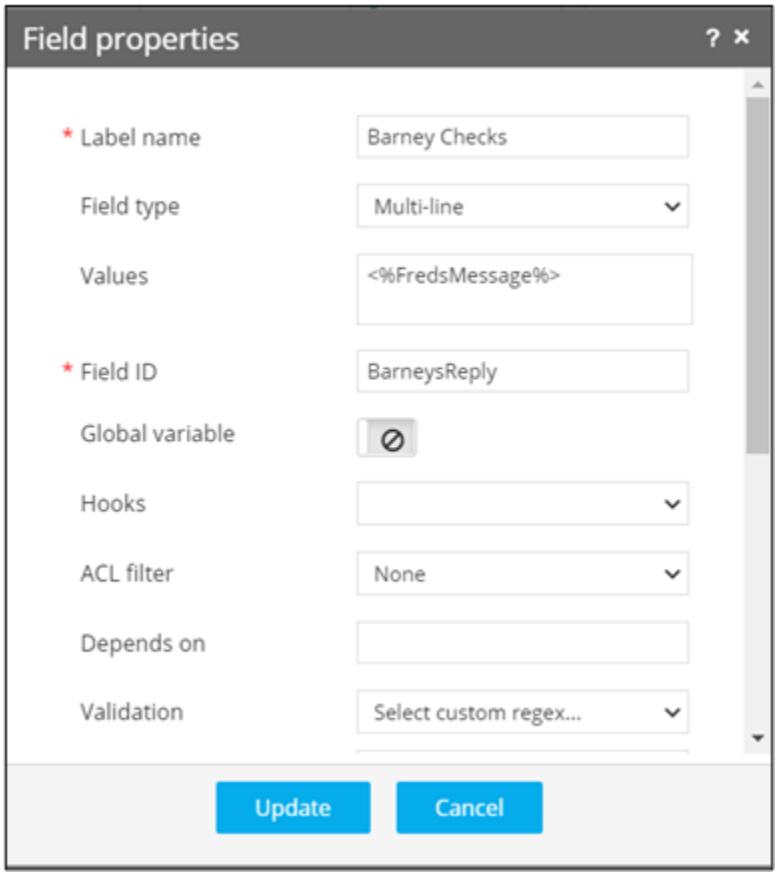


9. Click **Save**.
10. Drag and drop another **Form** task to receive the inputs from the first form.
11. Define the form fields to receive inputs from the first form.



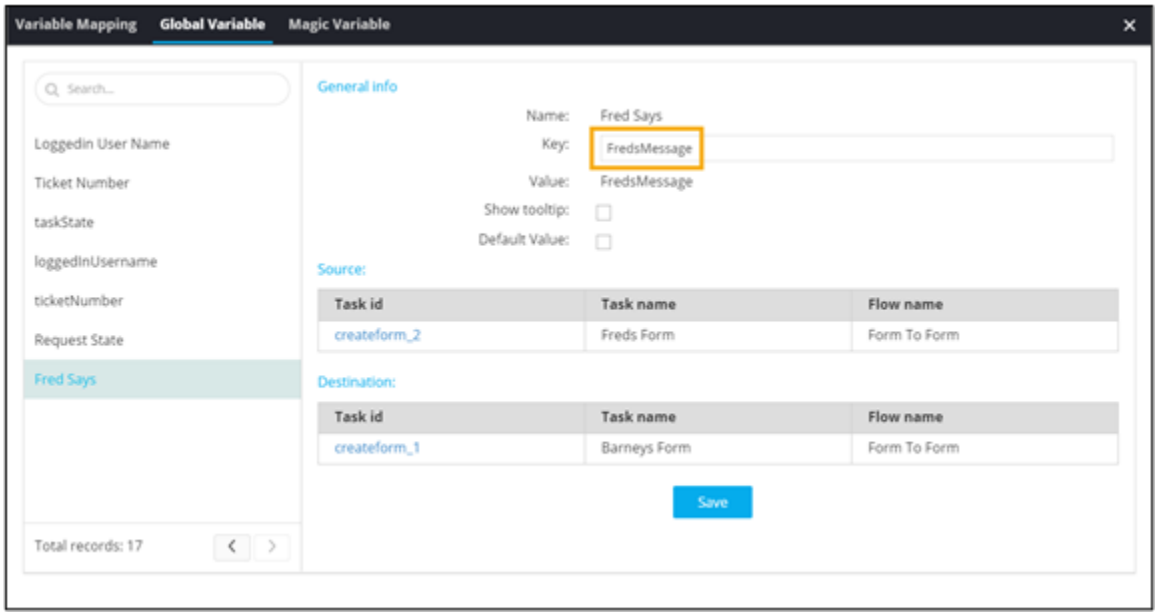
12. In the **Field properties** window, refer the global variable declared in the first form (`<%Fredsmessage%>`).

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



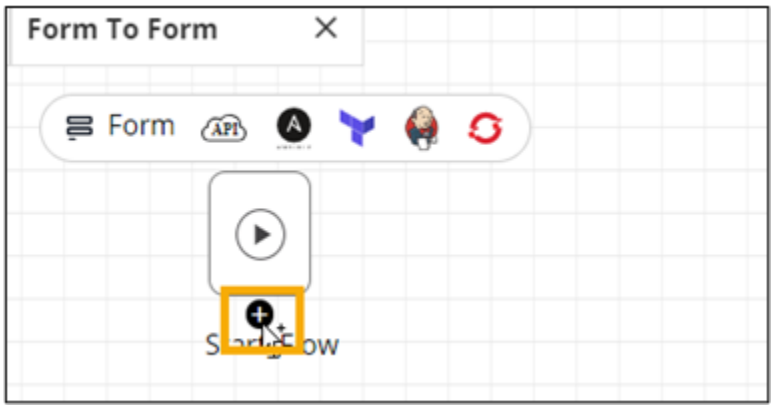
Note: You can also see the variable mapping in the Global Variables Inventory.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



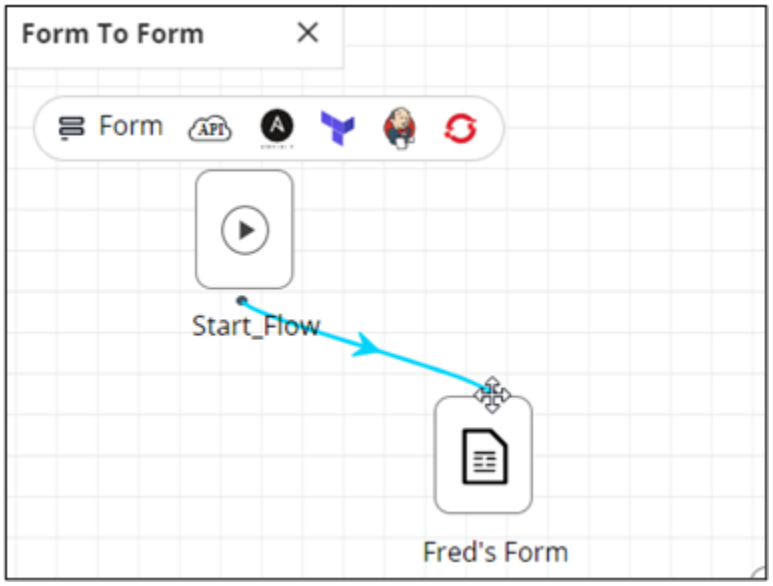
Note: For more information, refer to the section on Global Variable Inventory.

13. To connect the tasks, hover your mouse over a task and click .

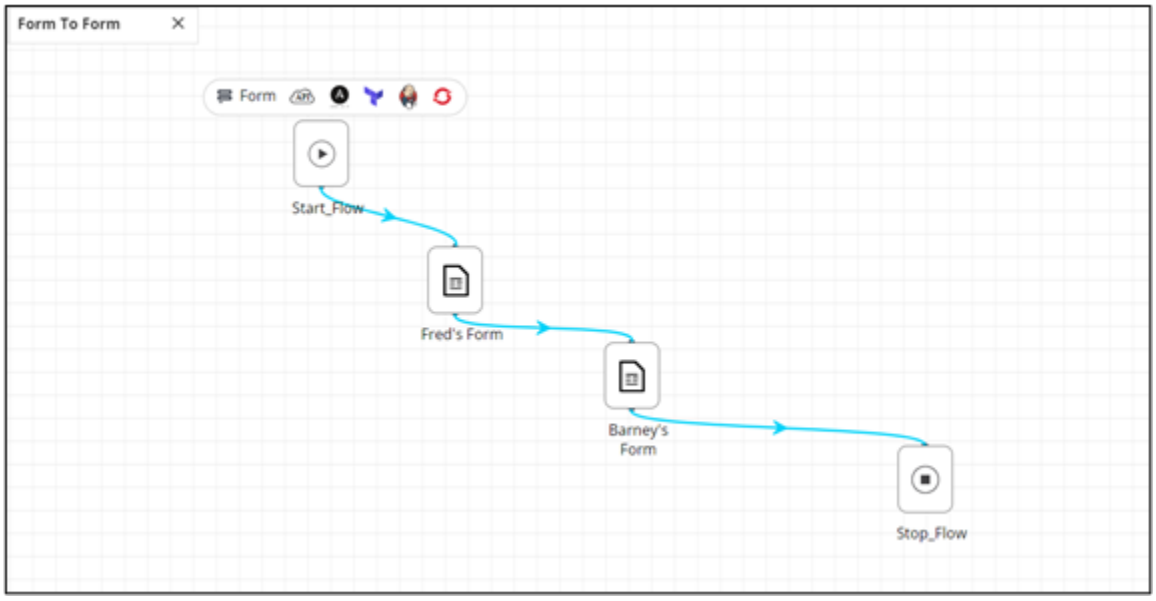


14. Hold and drag your mouse to connect to the next task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



15. Connect all the tasks in the workflow.



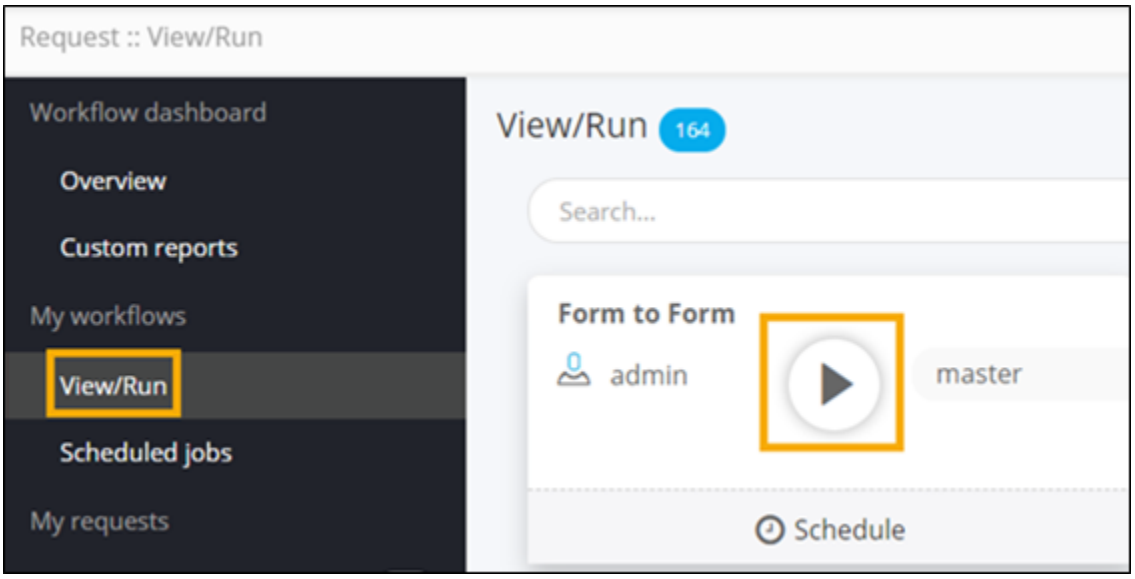
16. Enable the workflow.

> **Note:** For more information on enabling a workflow, click here.

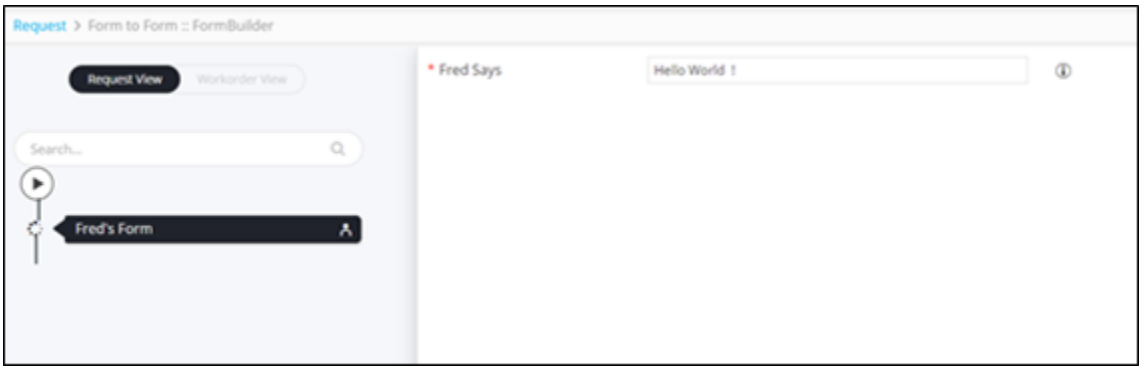
17. From the top left corner of the screen, click ☰.
18. From the menu displayed, click **Request**.
19. On the **Request :: Overview** page, from the navigation pane on the left, click **View/Run**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

20. To trigger the workflow, on the **Request :: View/Run** page, search for the **Form to Form** workflow and click .



21. Get inputs in the first form (Fred's form) and click **Next**.



Data is passed into the second form (Barney's form).

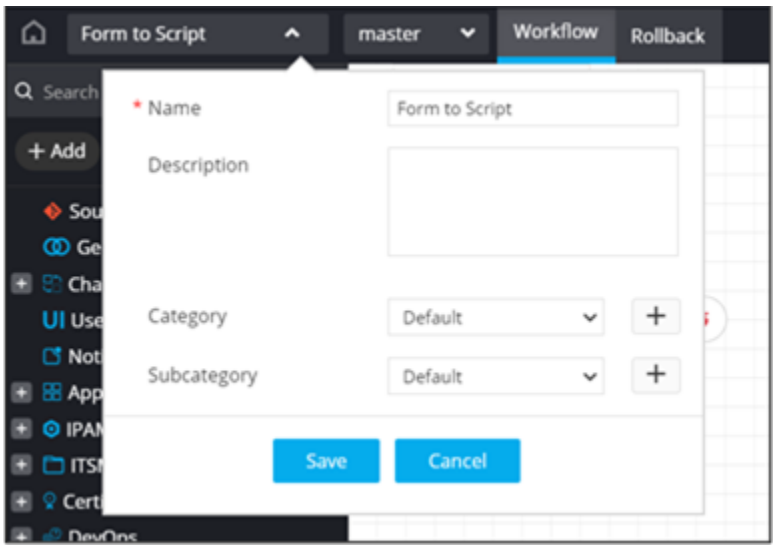
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Note: For more information on how to connect other tasks in the Visual Workflow studio, refer to the section Connecting tasks in Workflow Studio.

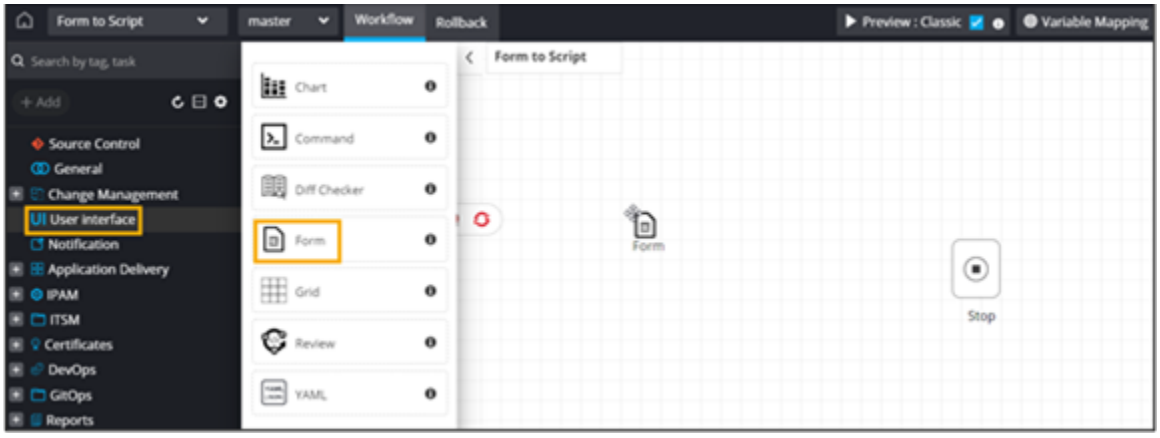
How to Connect Form to Script

You can pass data between a form and a script task using global variables.

1. Design a new workflow.

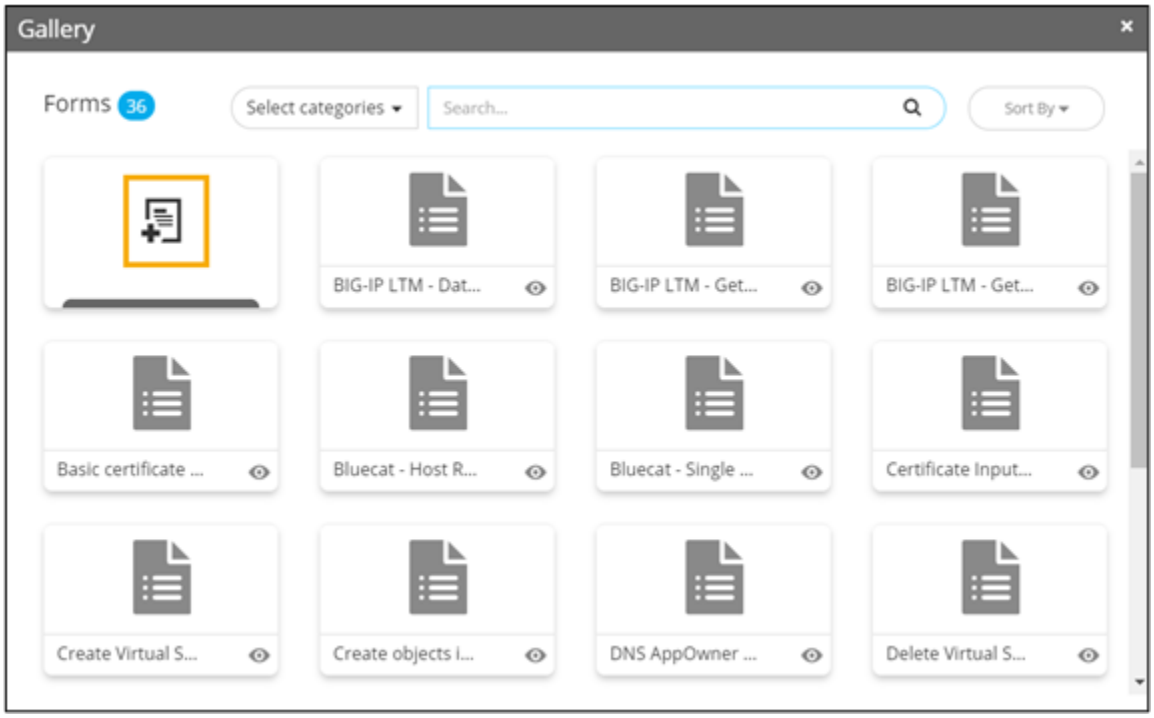


2. From the **User Interface** section, drag and drop the **Form** task.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

3. In the **Gallery** window, to design a new form, click .

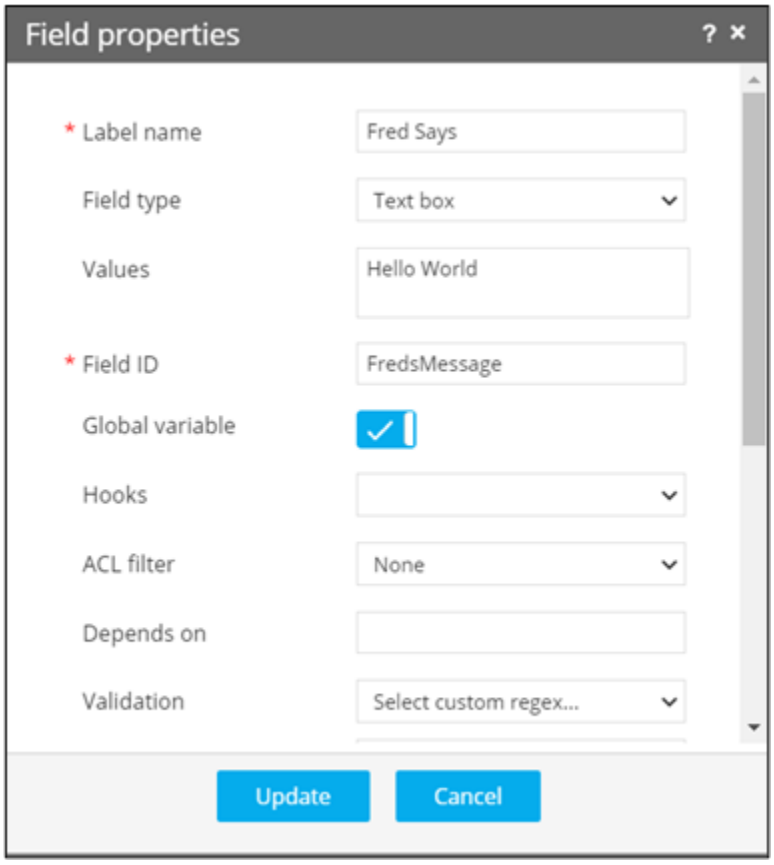


4. Under the **Form builder** tab, define the form field(s).

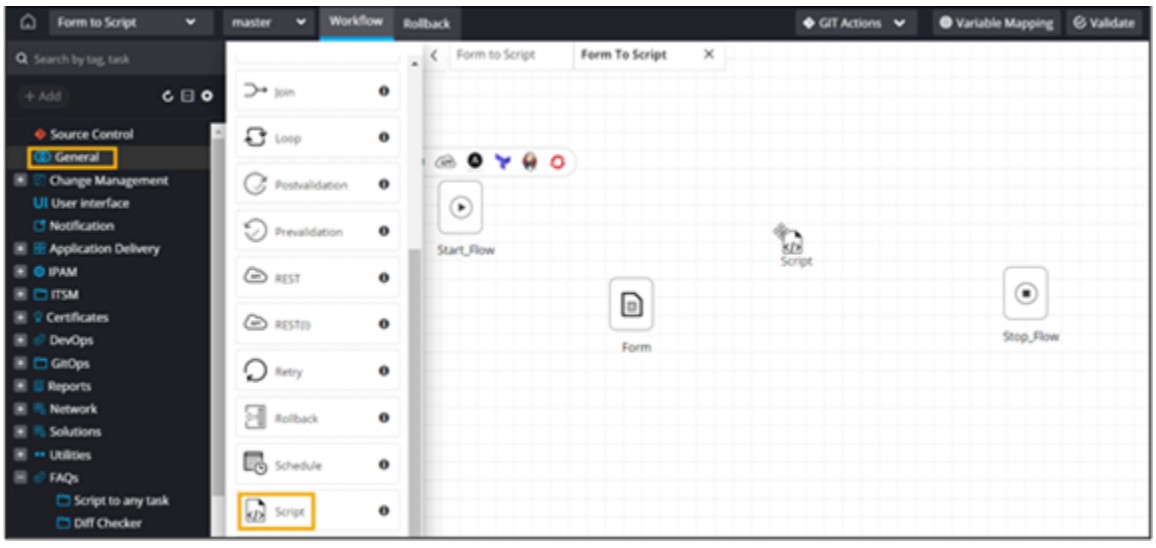
Note: For more information on adding form fields, click here.

5. In the **Field properties** window, declare the **Field ID** as the **Global Variable**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

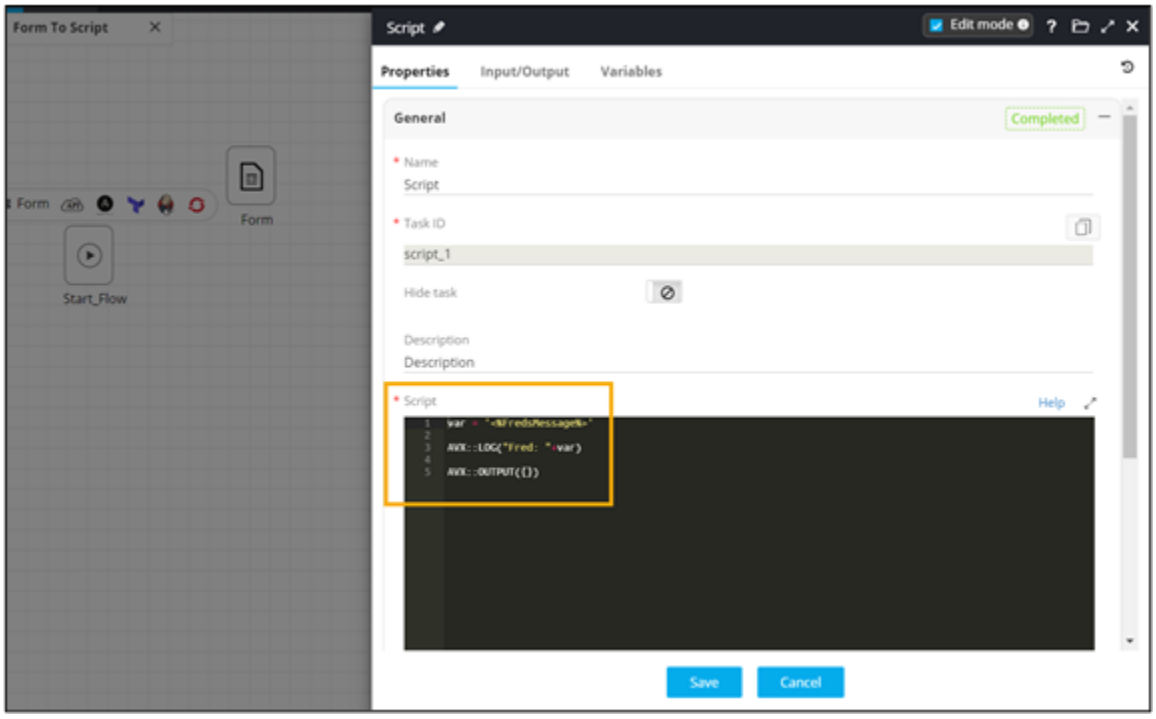


6. From the **General** section, drag and drop the **Script** task.

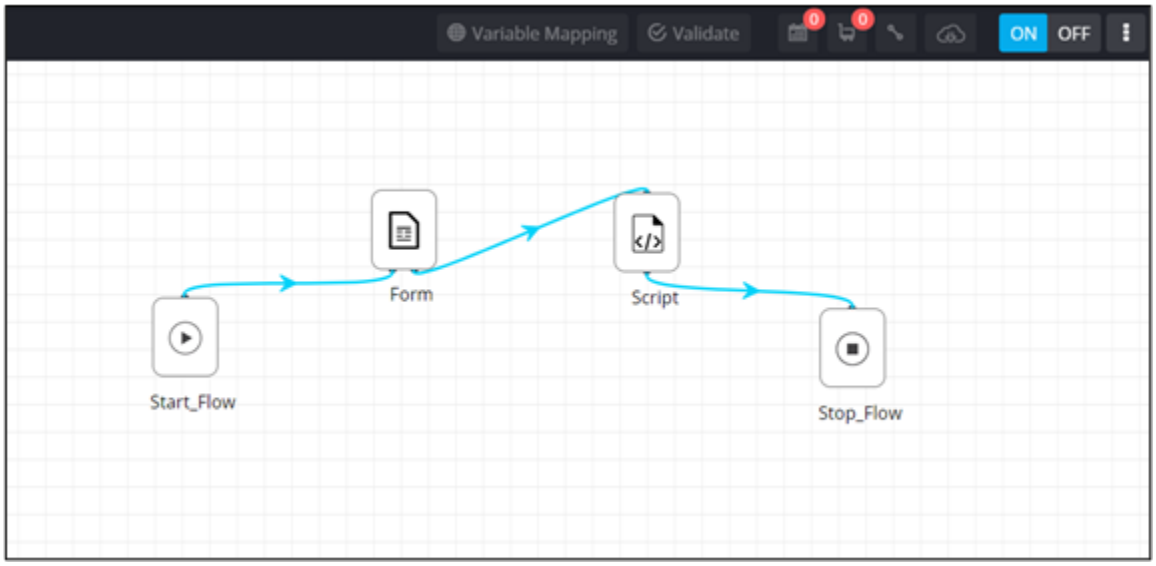


7. In the **Script** task window, define the script to get inputs from the form task and print data in the logs.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



8. Connect and enable the workflow.

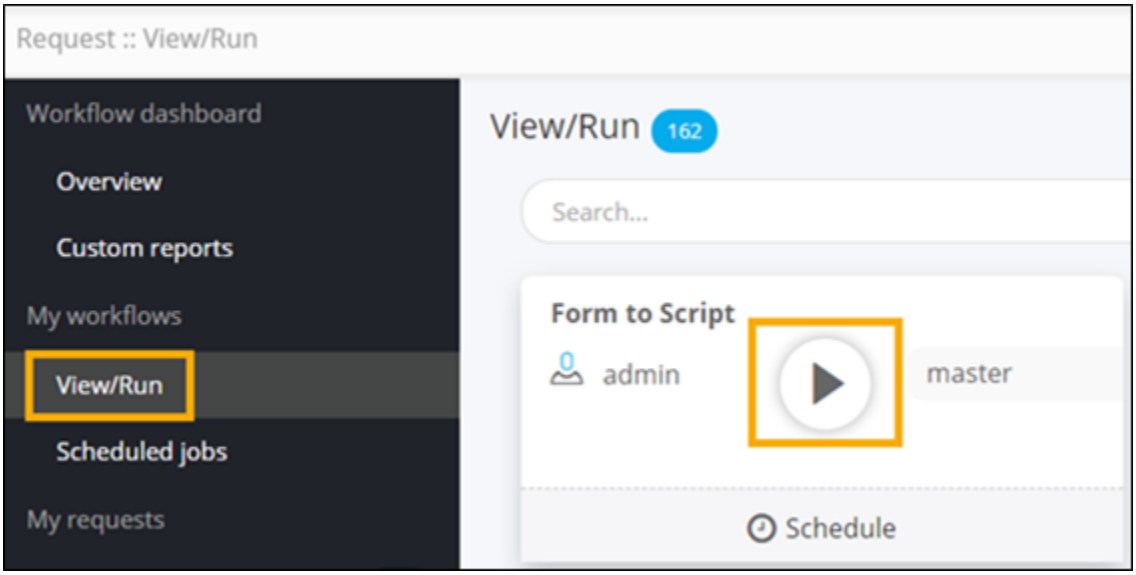


Note: For more information on enabling a workflow, click here.

9. From the top left corner of the screen, click ☰.
10. From the menu displayed, click **Request**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

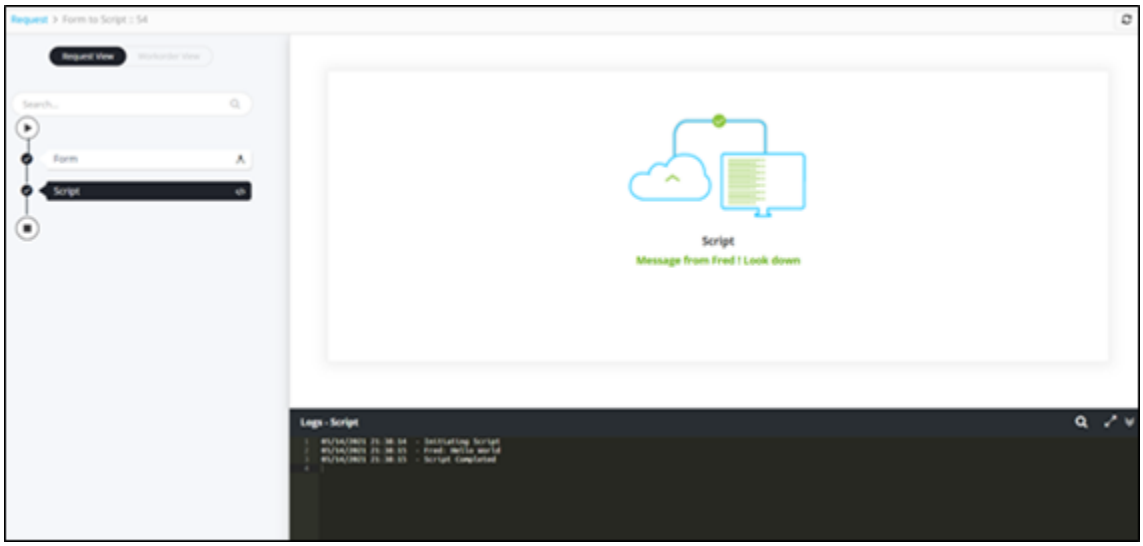
11. On the **Request :: Overview** page, from the navigation pane on the left, click **View/Run**.
12. To trigger the workflow, on the **Request :: View/Run** page, search for the **Form to Script** workflow and click .



- Inputs received in the form.

- Script is executed.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

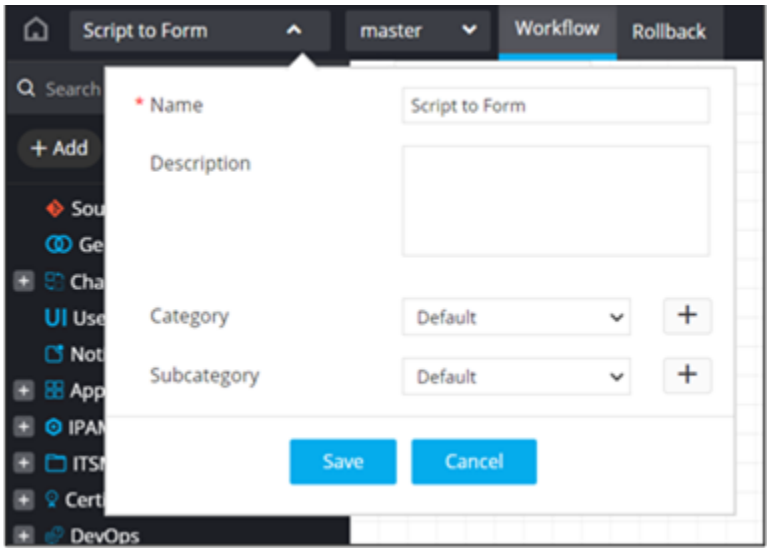


Note: For more information on how to connect other tasks in the Visual Workflow studio, refer to the section Connecting tasks in Workflow Studio.

How to Connect Script to Form

You can define a variable called “Hello World” through the Script and pass the data to a Form task.

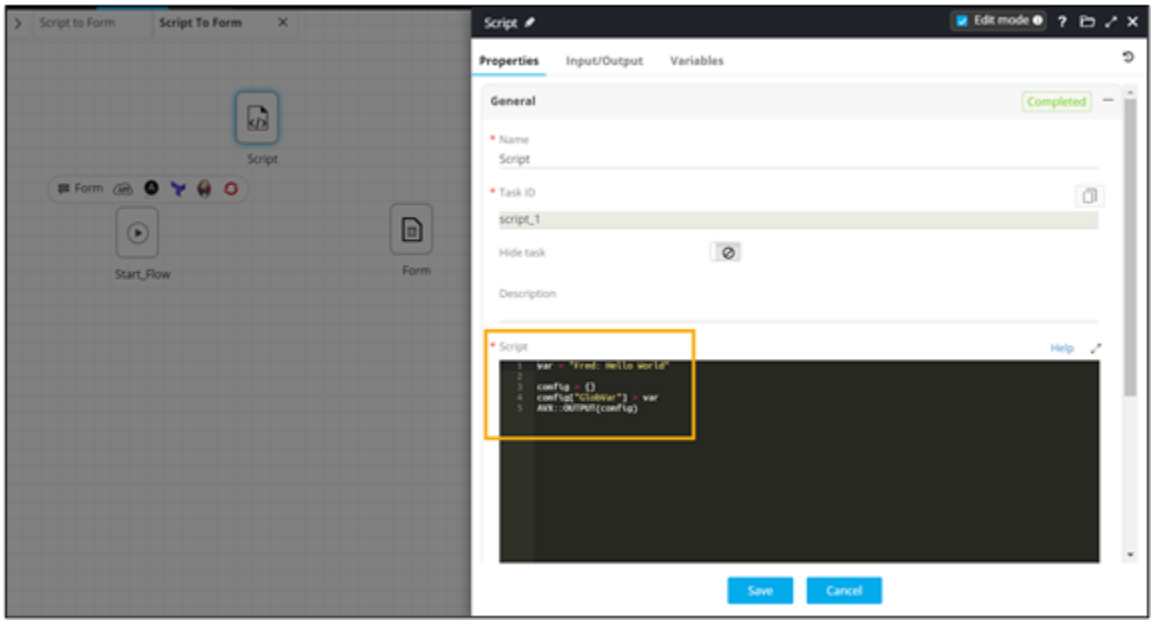
1. Design a new workflow.



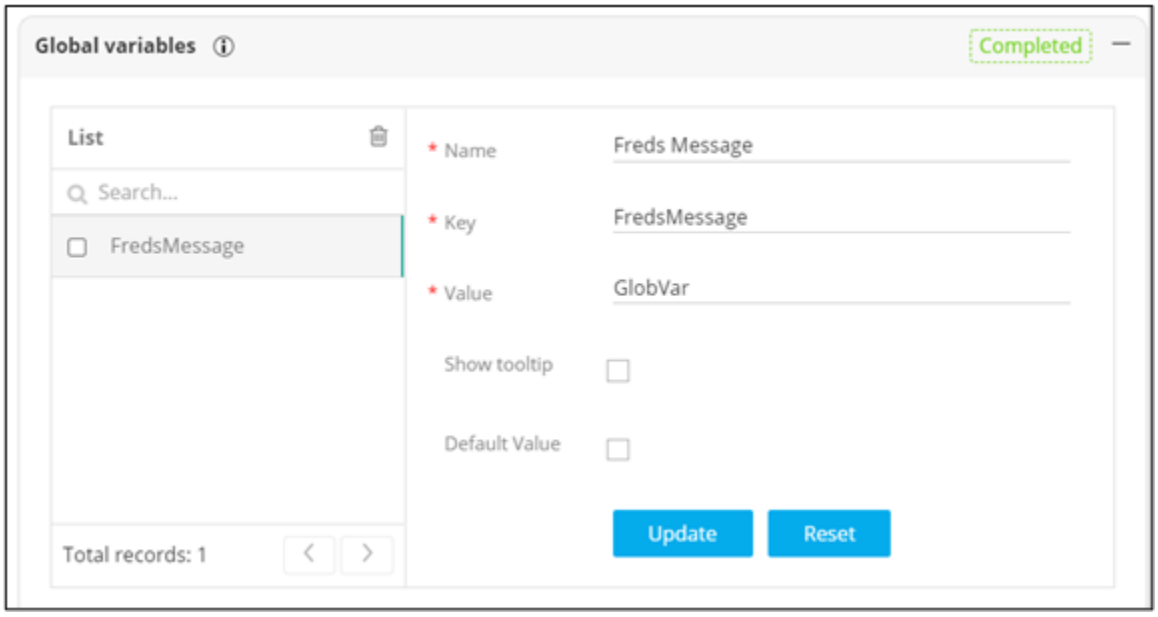
2. From the **General** section, drag and drop the **Script** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

3. In the **Script** task window, under the **General** section, define the Script.



4. In the **Script** task window, under the **Global variables** section, declare the global variables in order to refer the value in the Form task.



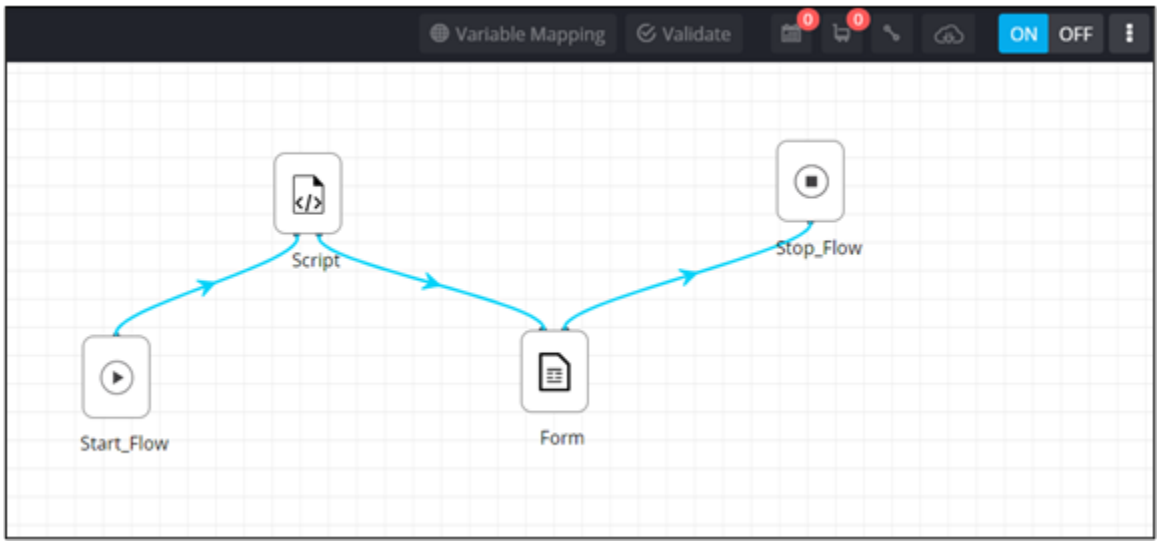
5. In the **Script** task window, under the **Custom message** section, define custom messages to be displayed during transitions.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

6. From the **User Interface** section, drag and drop the **Form** task.
7. Define form field(s) and declare the Global Variable <%Fredsmessage%> to be referenced.

Note: For more information on adding form fields, click here.

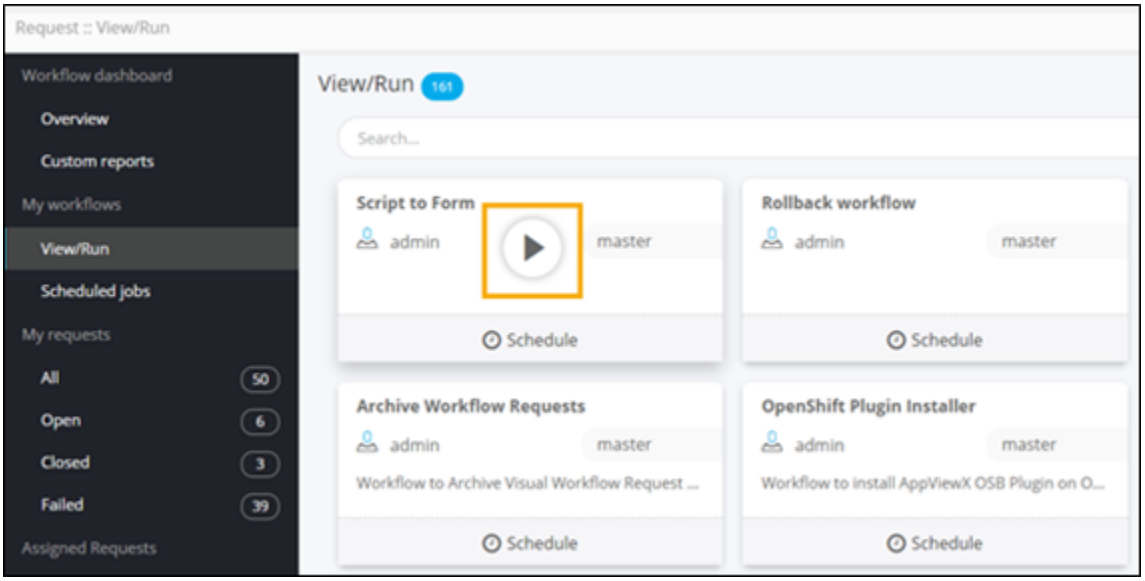
8. Connect and enable the workflow.



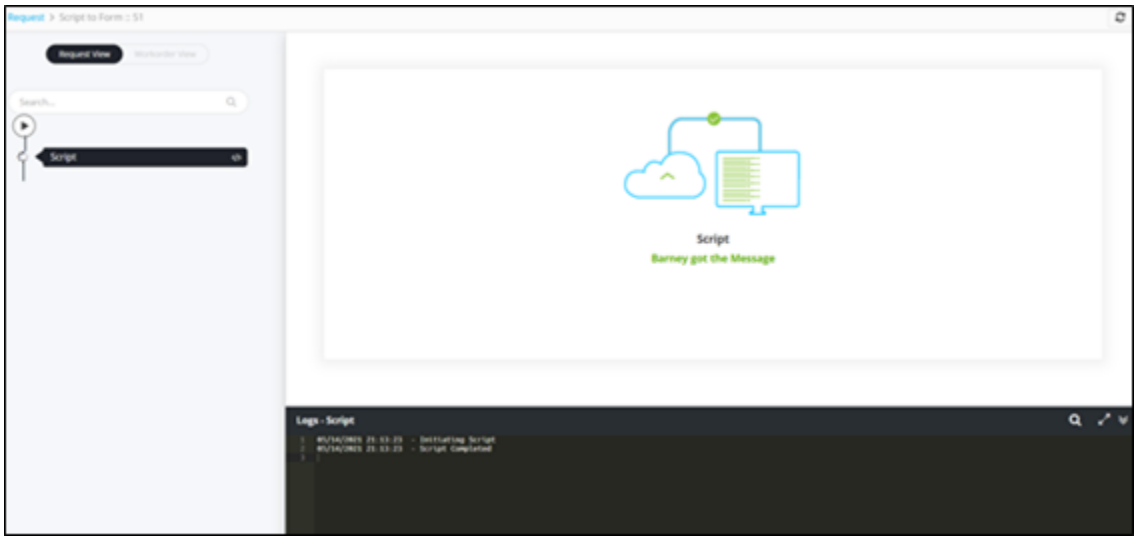
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Note: For more information on enabling a follow, click here.

9. From the top left corner of the screen, click ☰.
10. From the menu displayed, click **Request**.
11. On the **Request :: Overview** page, from the navigation pane on the left, click **View/Run**.
12. To trigger the workflow, on the **Request :: View/Run** page, search for the **Script to Form** workflow and click ▶.

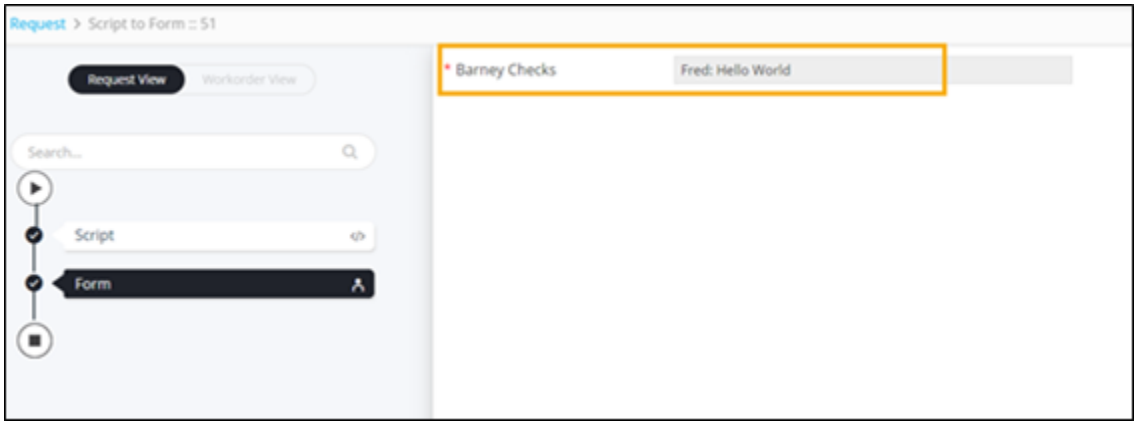


- The Script is executed.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- The data is passed from Script to Form.

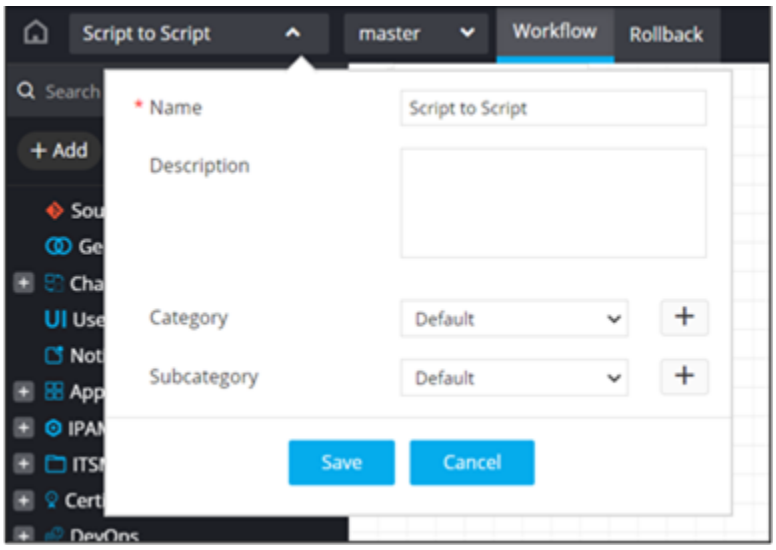


Note: For more information on how to connect other tasks in the Visual Workflow studio, refer to the section Connecting tasks in Workflow Studio.

How to Connect Script to Script

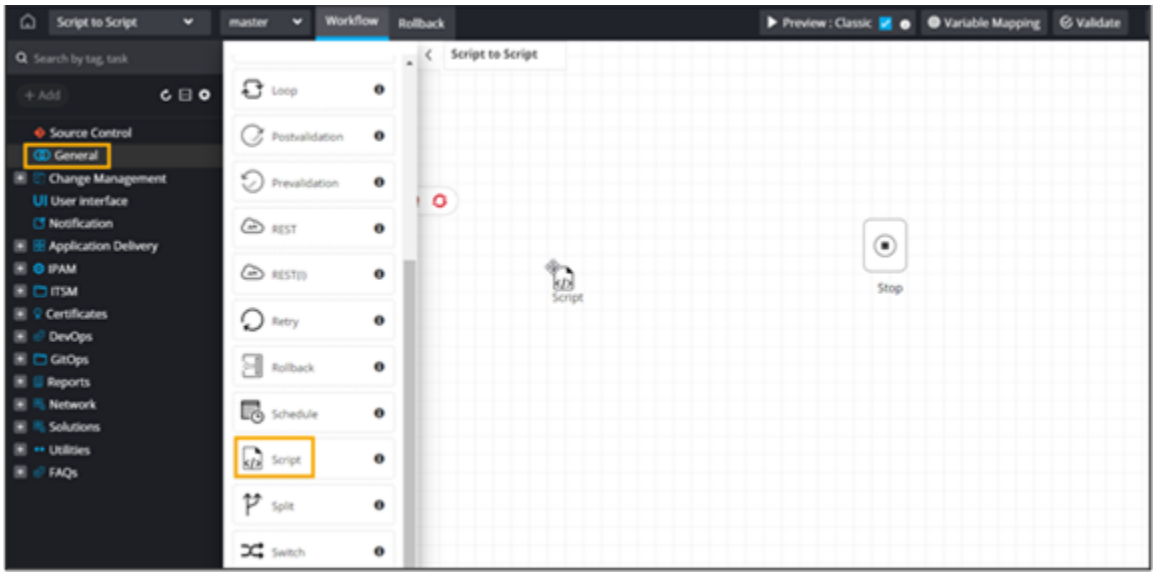
You can define and declare values from one script and pass them into another Script task.

1. Design a new workflow.

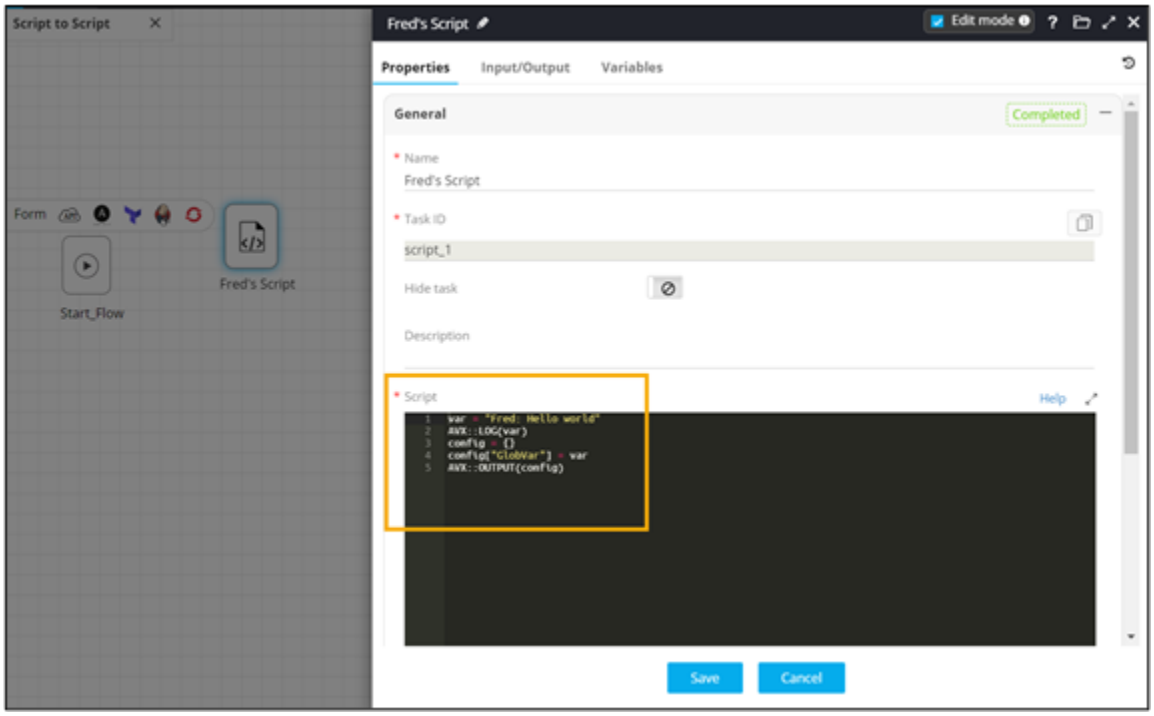


2. From the **General** section, drag and drop the **Script** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

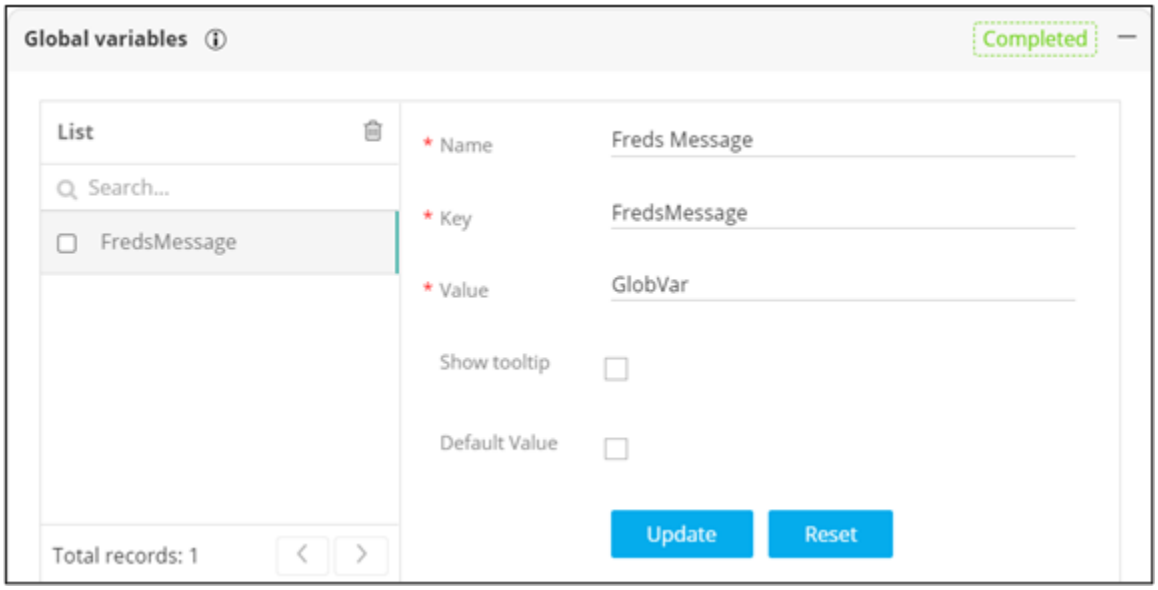


3. In the **Script** task window, under the **General** section, define the Script.



4. In the **Script** task window, under the **Global variables** section, define and declare the value as a global variable.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



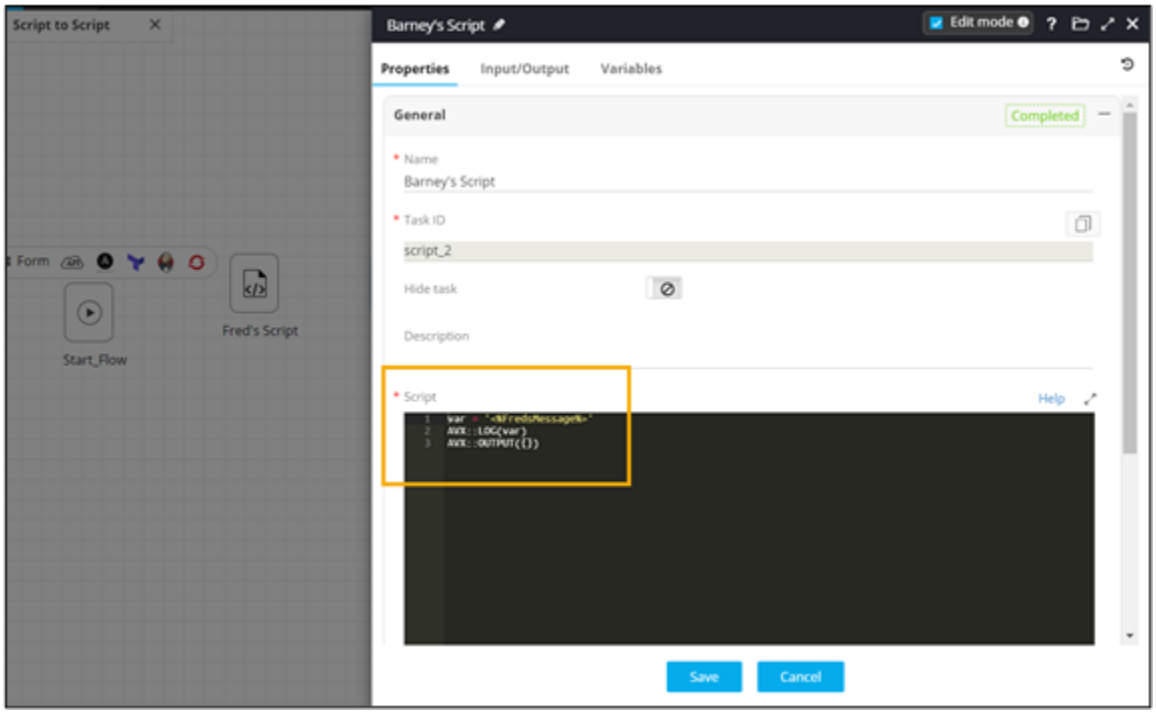
5. In the **Script** task window, under the **Custom message** section, define the custom messages to be displayed between workflow stages.

6. Drag and drop another **Script** task into the workspace.

> **Tip:** You can also right-click and clone the first script task.

7. Refer the global variable value from the first Script.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



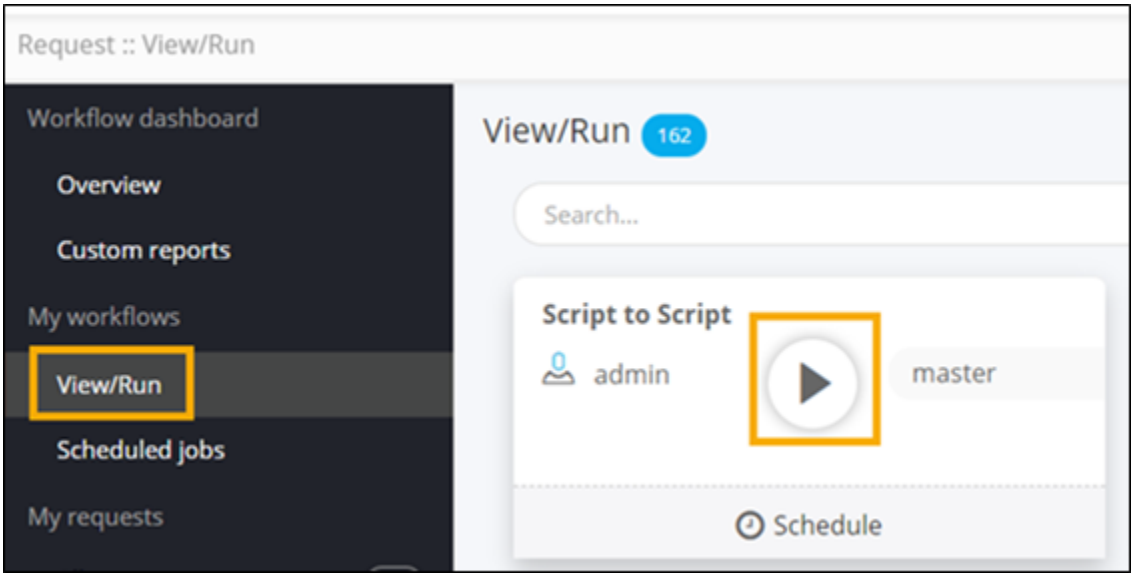
8. Connect and enable the workflow.

> **Note:** For more information on enabling a workflow, click here.

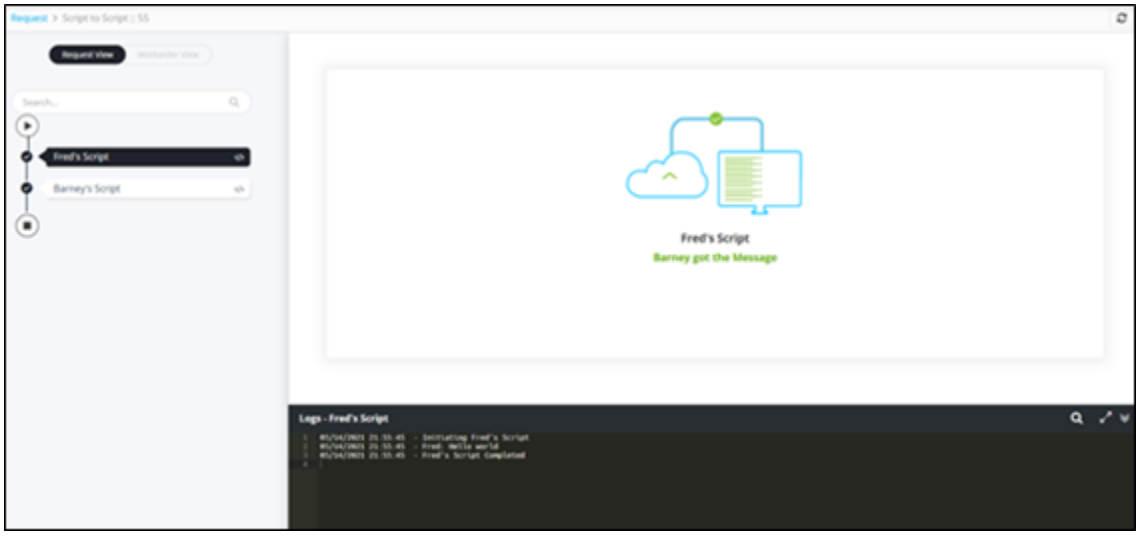
9. From the top left corner of the screen, click ☰.
10. From the menu displayed, click **Request**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

11. On the **Request :: Overview** page, from the navigation pane on the left, click **View/Run**.
12. To trigger the workflow, on the **Request :: View/Run** page, search for the **Script to Script** workflow and click .

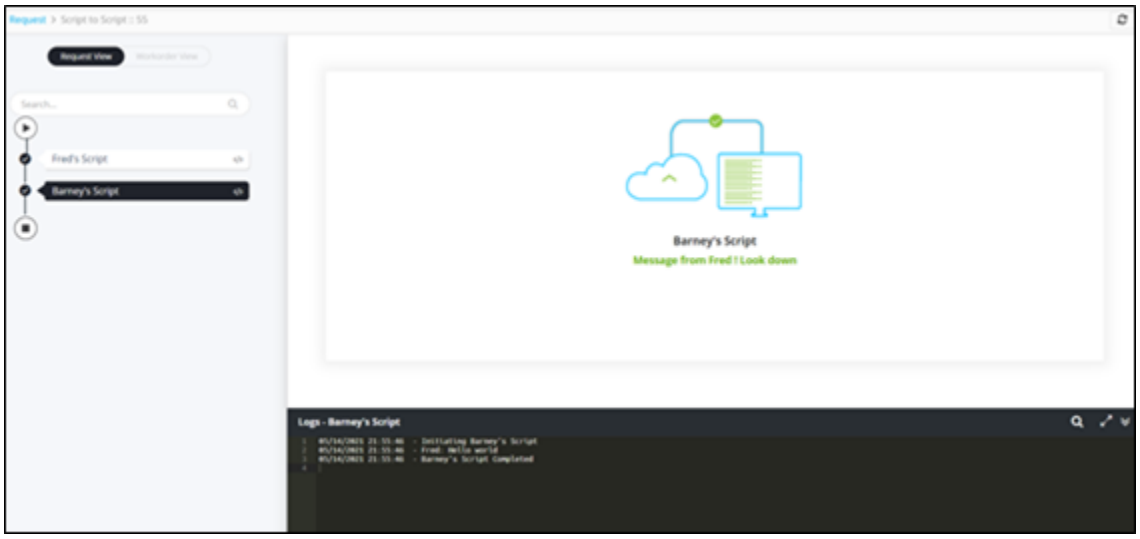


- The script is executed.



- Values are passed from one script to another.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

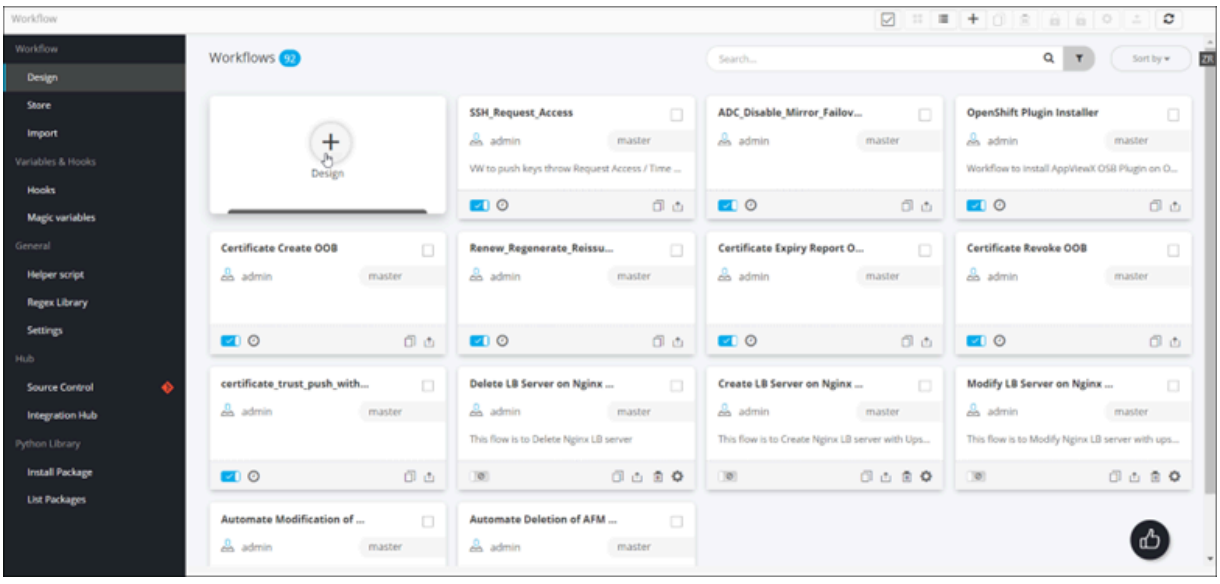


Note: For more information on how to connect various tasks in the Visual Workflow studio, refer to the section Connecting tasks in Workflow Studio.

Auto-generate Forms

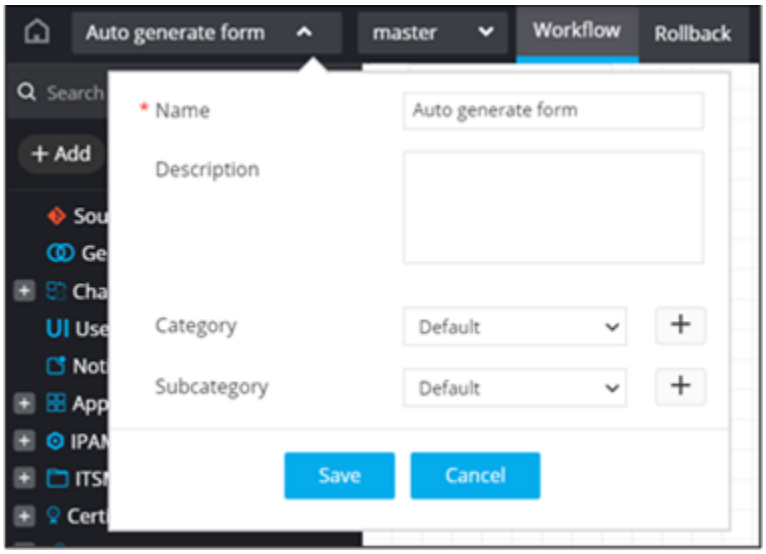
You can auto-generate a self-service form from any workflow task(s). The variables used within a task are rendered into the form automatically.

1. To create a new workflow, on the **Workflow** inventory page, click **Design**.

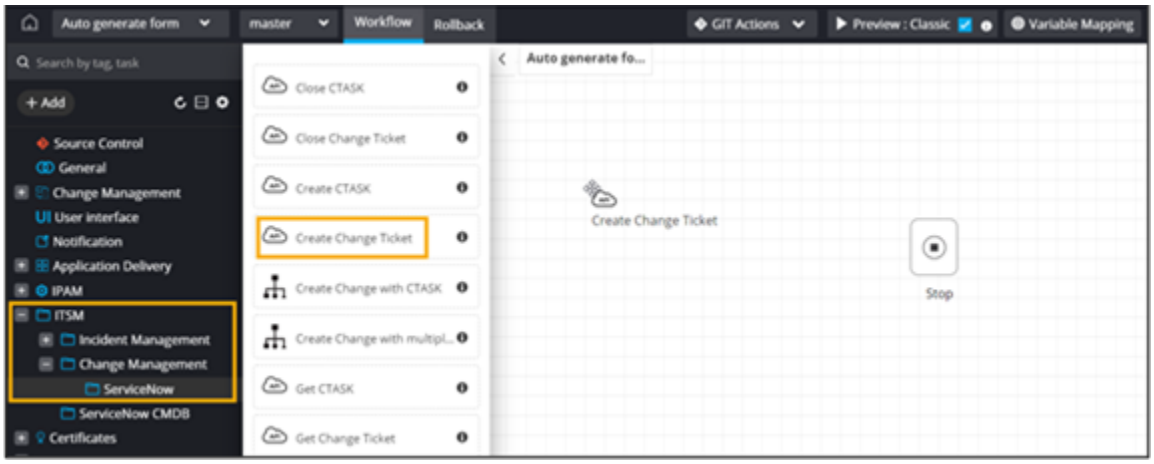


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

2. Enter the workflow **Name**.

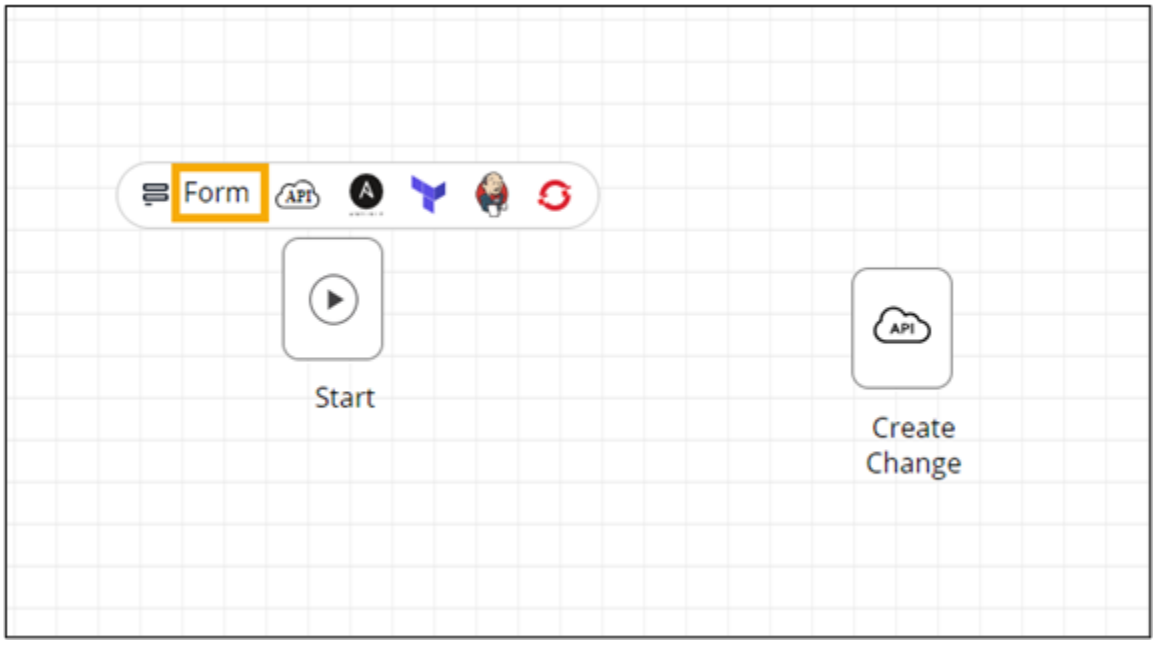


3. Click **Save**.
4. Drag and drop a prebuilt workflow task.

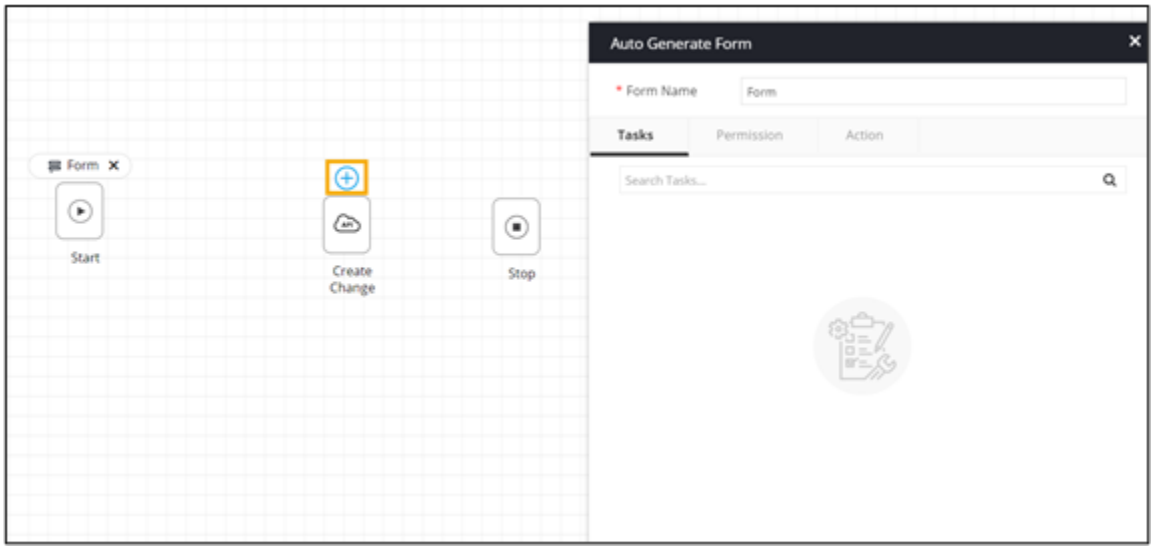


5. To auto-generate a self-service form, click **Form** above the **Start** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

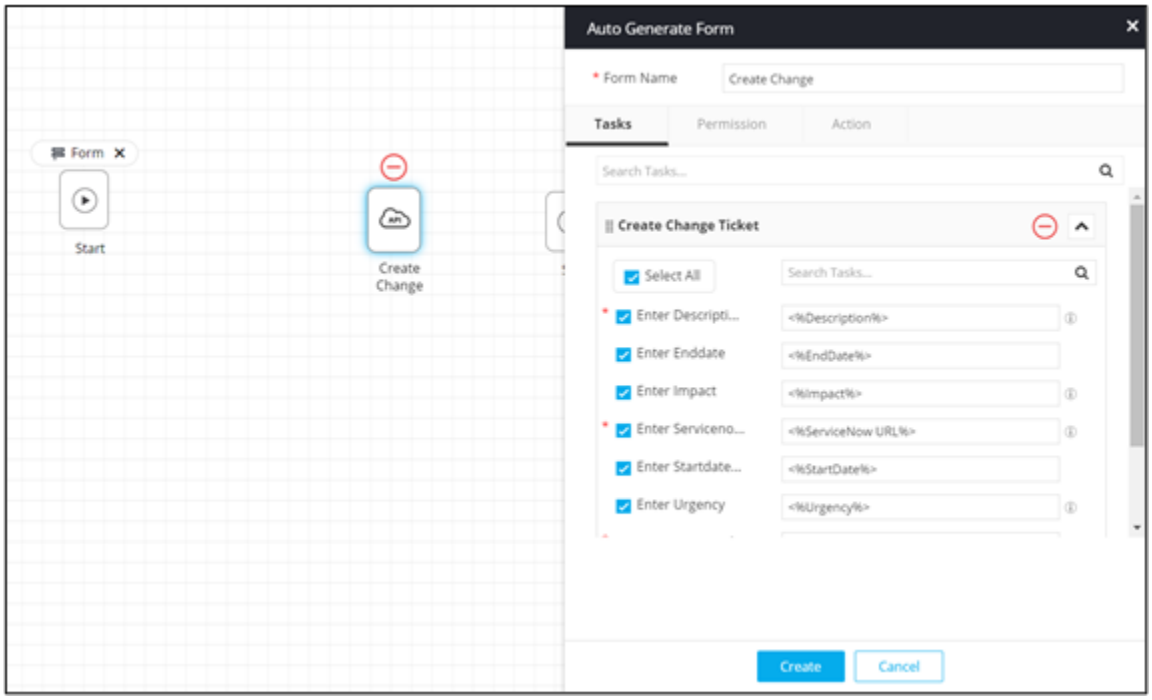


6. To auto-fill the variables in the form, click ⊕ above the **Create Change** task.

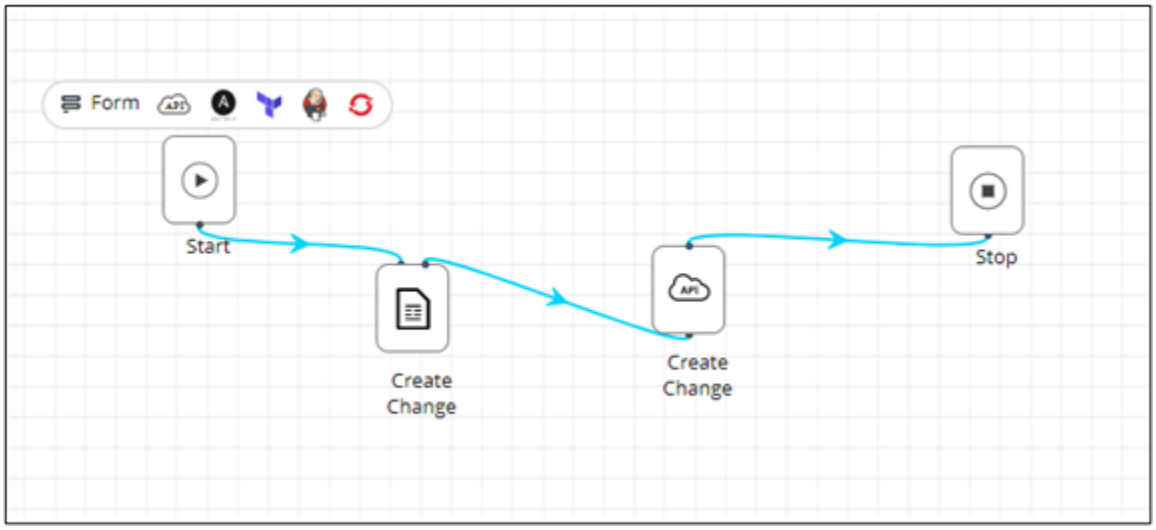


7. Enter the **Form Name**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



8. Select/deselect the fields to be displayed in the self-service form.
9. To generate the form with the selected field IDs, click **Create**.
10. Connect the workflow tasks and trigger the workflow from the Request :: View/Run page.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Sample Workflows using Prebuilt Tasks

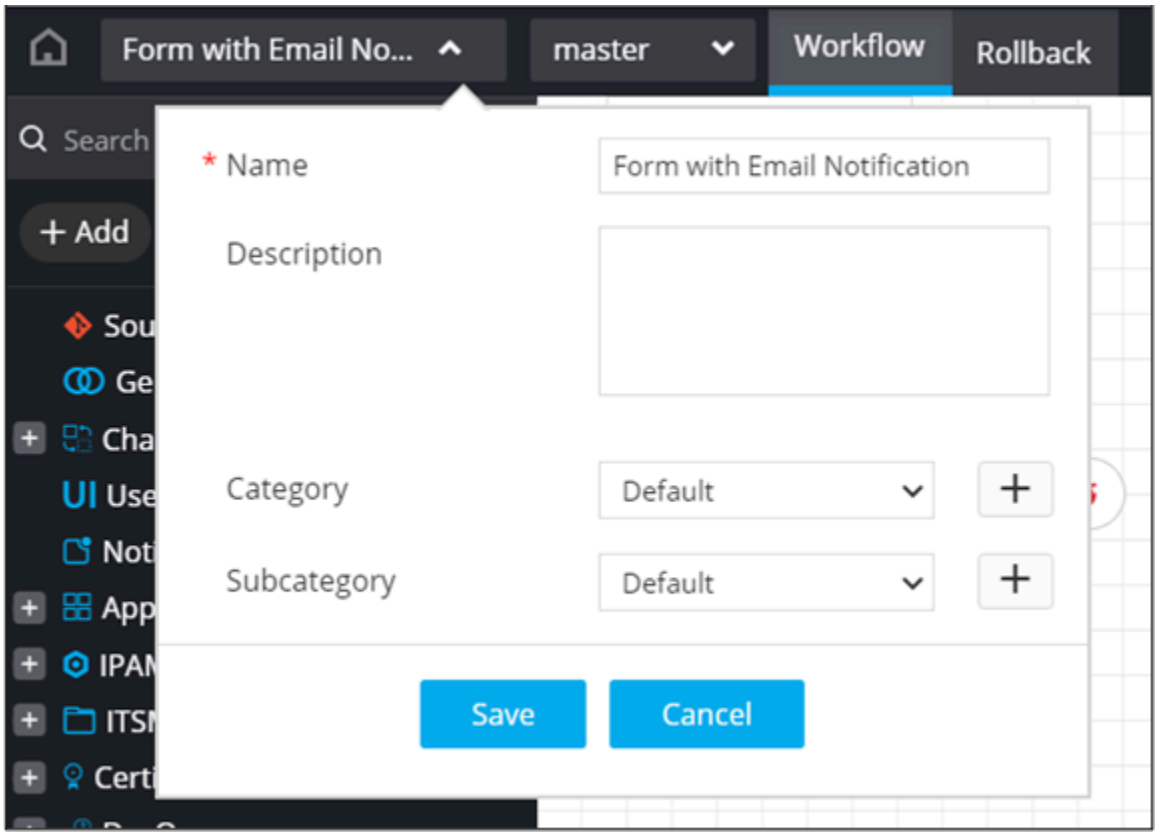
This section will walk you through a few sample workflows that describe the step-by-step process of creating workflows using prebuilt tasks.

- Form with Email Notification
- Executing a Ping Check using Command Task
- Creating A Record on Infoblox device using prebuilt tasks
- Creating a VIP with Incident Ticket

Form with Email Notification

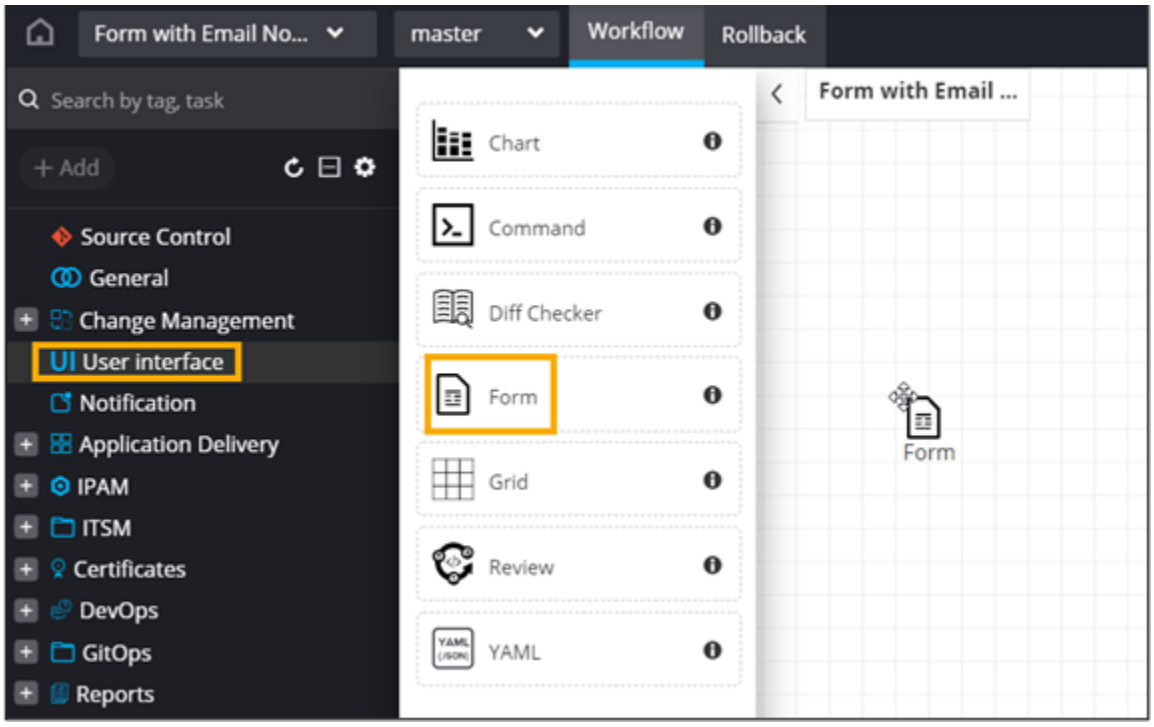
You can design a workflow to send an email notification based on the details provided in a form.

1. Design a new workflow.

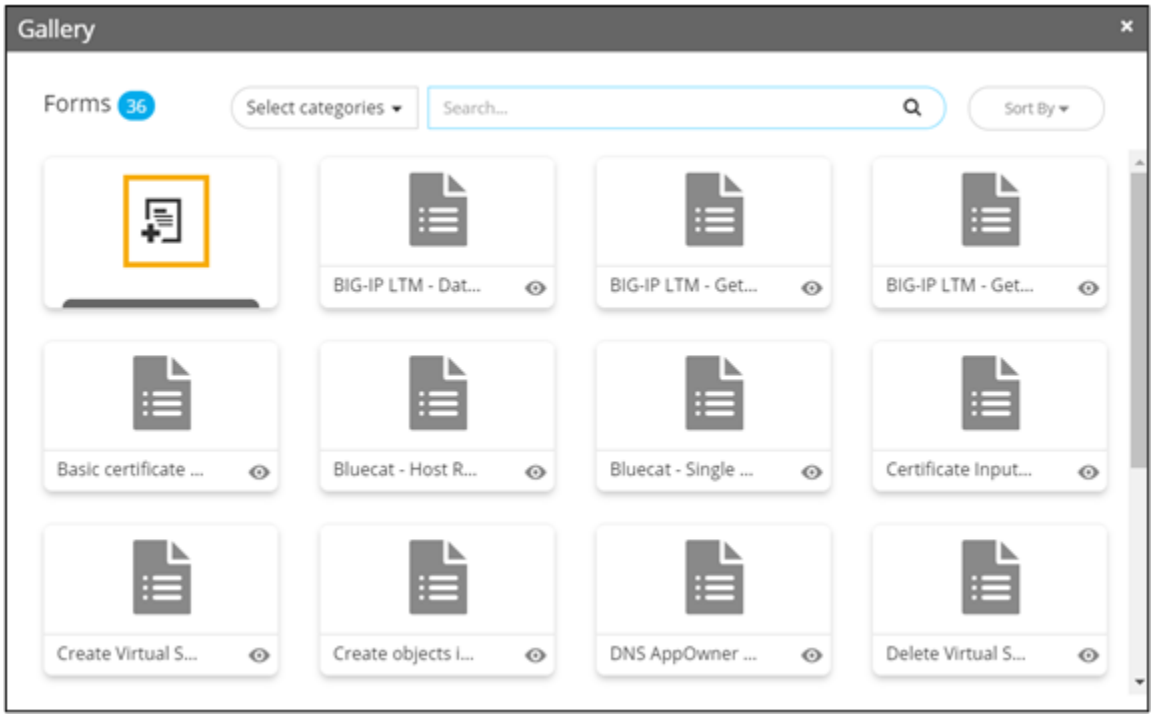


2. From the **User Interface** section, drag and drop a **Form** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

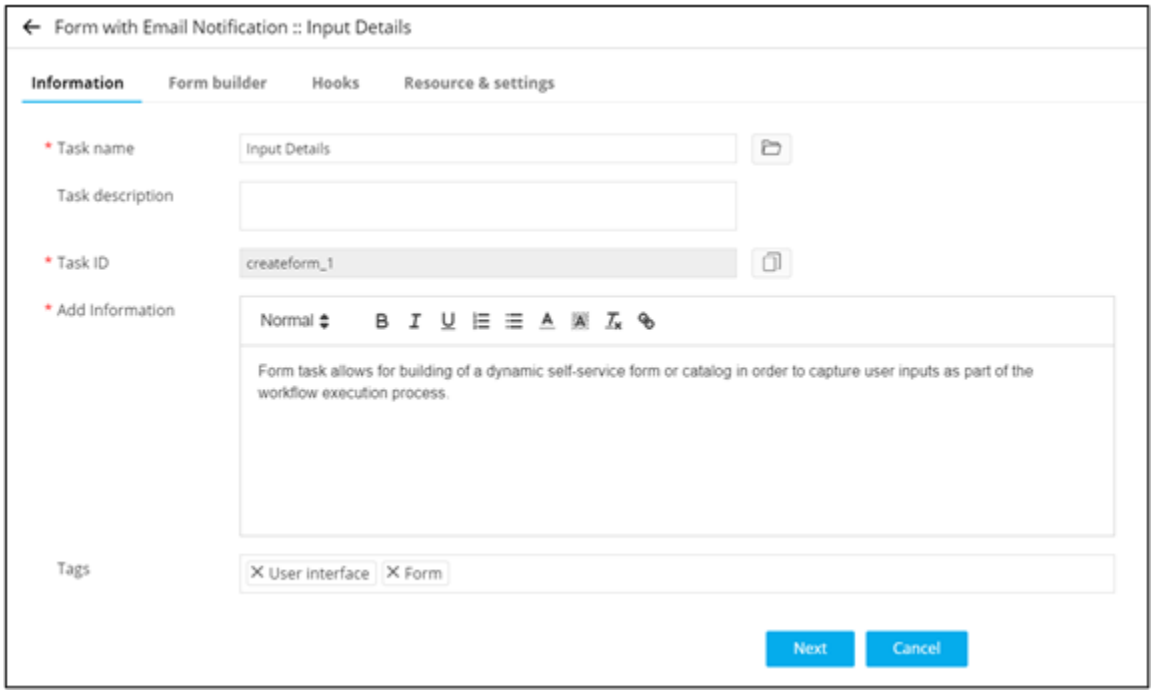


3. To design a new form, in the **Gallery** window, click .

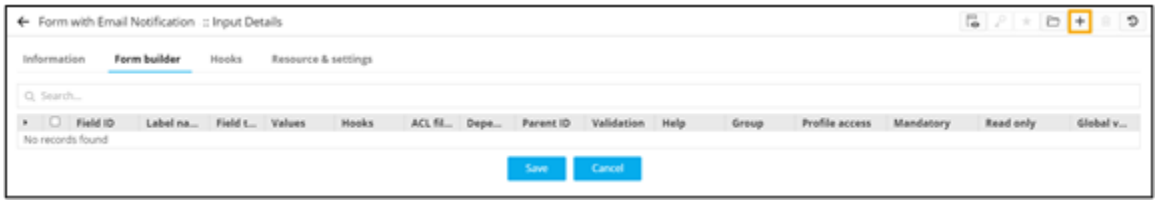


4. Under the **Information** tab, enter the **Task name**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



5. Click **Next**.
6. To add the form fields, under **Form builder**, from the top right corner of the screen, click +.

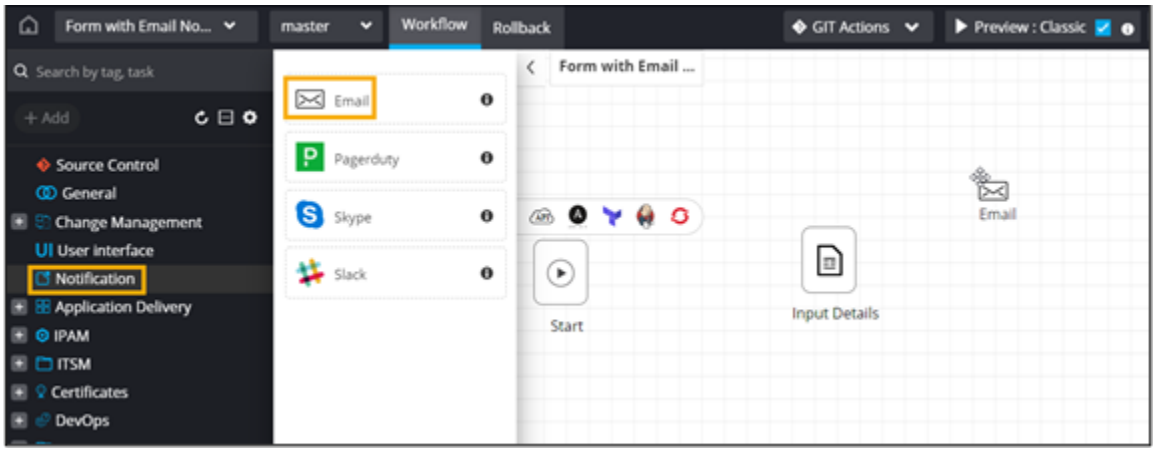


7. Add the following form fields and turn on the Global varibale toggle to enable the field IDs as global variables:
   - Enter your name: Field type - Text box, Field ID - name
   - Enter your email address: Field type - Text box, Field ID - email
   - Enter your message: Field type - Multi-line, Field ID - message

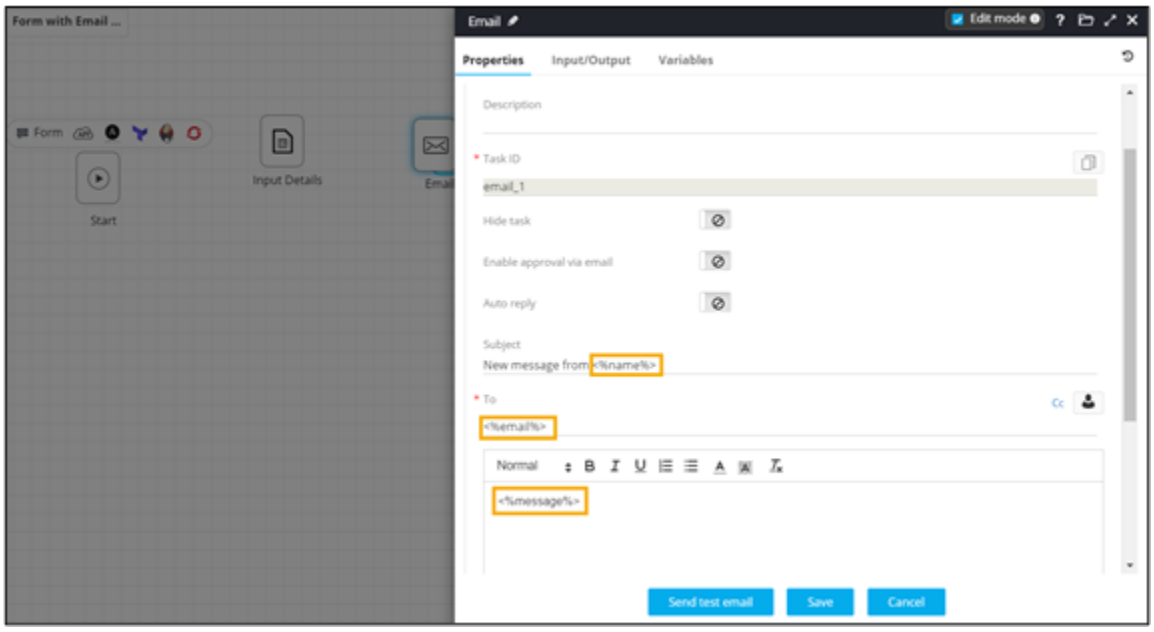
> **Note:** For more information on adding form fields, click here.

8. Click **Save**.
9. From the **Notification** section, drag and drop and **Email** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

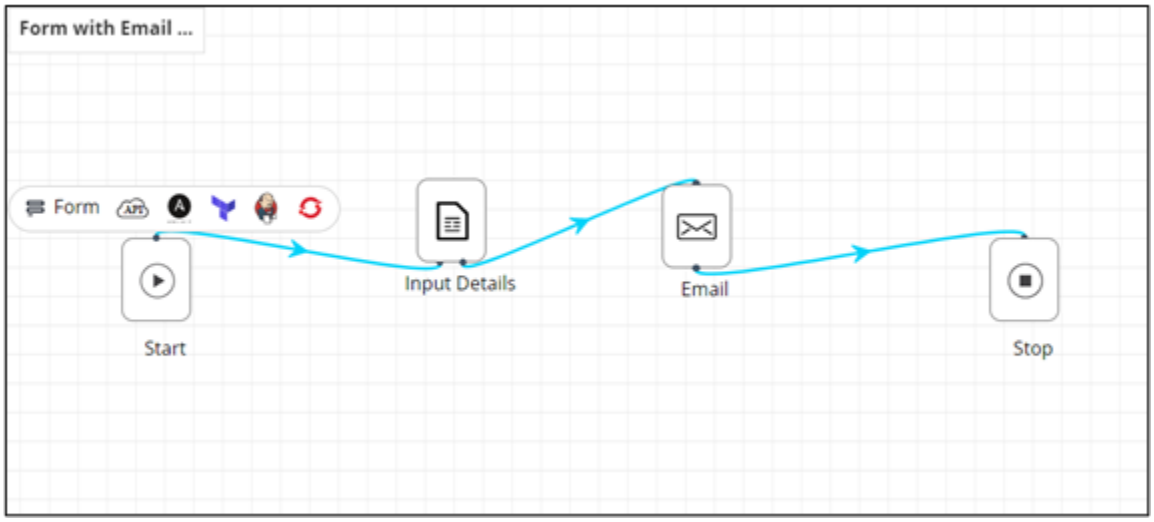


10. In the **Email** task window, under **Properties**, in the **General** section, refer the global variables defined in the Form task.



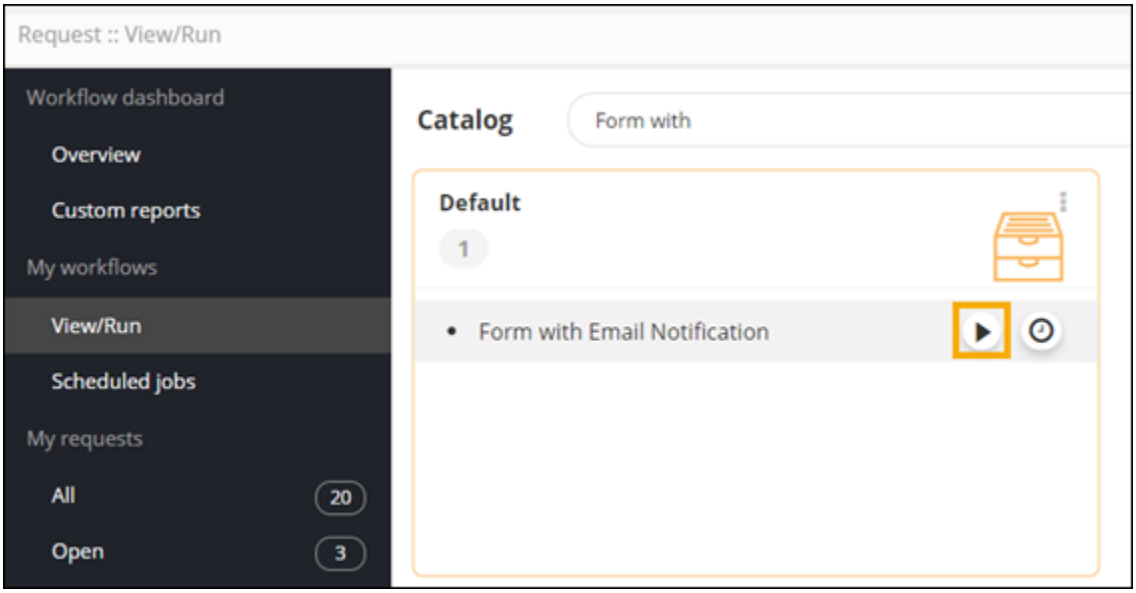
11. Click **Save**.
12. Connect and enable the workflow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



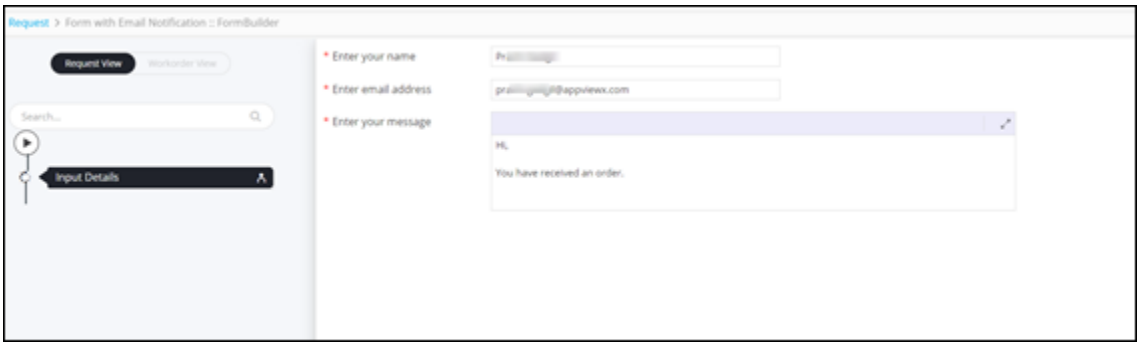
Note: For more information on enabling a workflow, click here.

13. From the top left corner of the screen, click ☰.
14. From the menu displayed, click **Request**.
15. On the **Request :: Overview** page, from the navigation pane on the left, click **View/Run**.
16. To trigger the workflow, on the **Request :: View/Run** page, search for the **Form with Email Notification** workflow and click ▶.

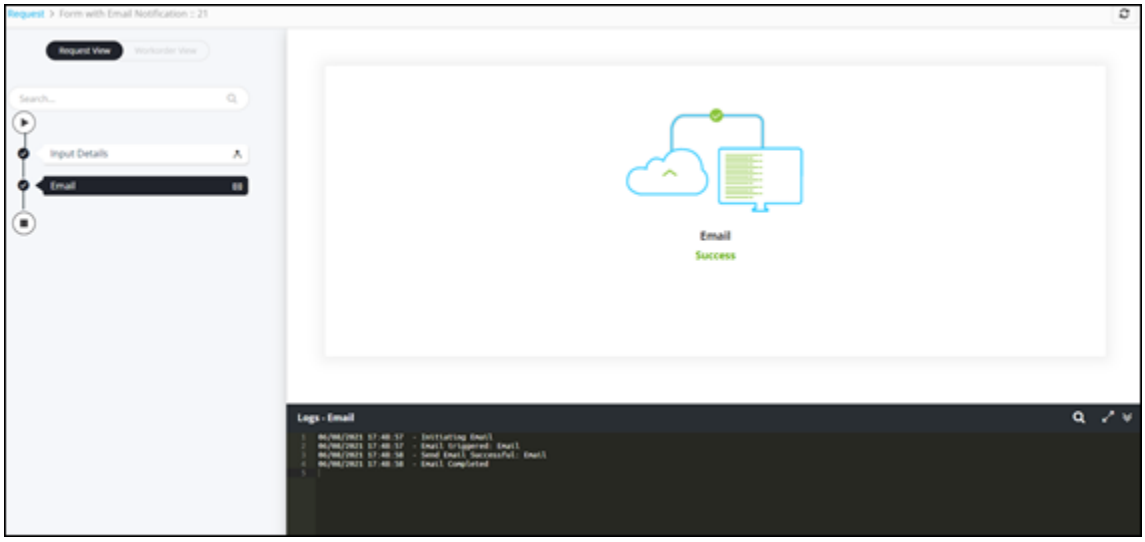


17. Enter the details in the input form.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

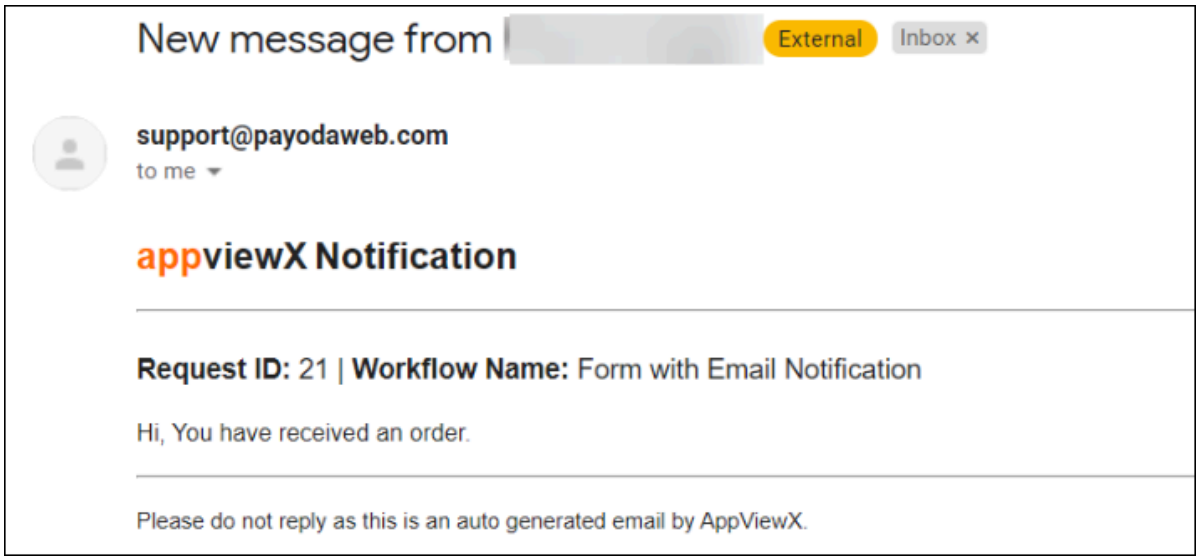


18. Click **Submit**.
    - Email notification successfully sent.



    - Email notification received.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

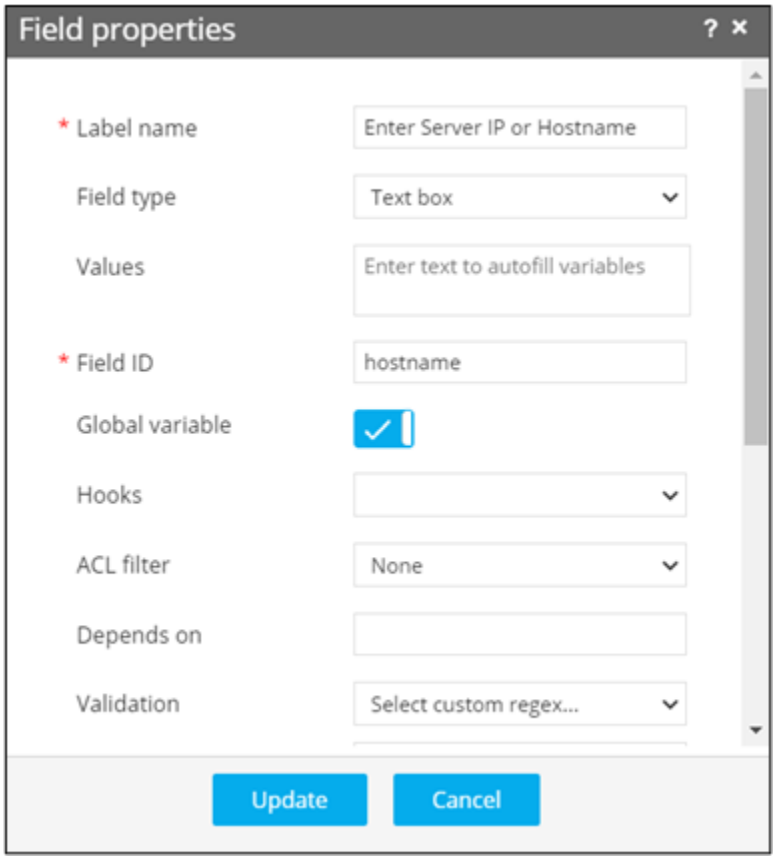


Executing a Ping Check using Command Task

You can design a workflow to use the Command task to execute a ping check on a hostname or server.

1. Design a new workflow.
2. From the **User Interface** section, drag and drop a **Form** task.
3. In the **Field properties** window, define the form field to get the Server IP or Hostname as shown.

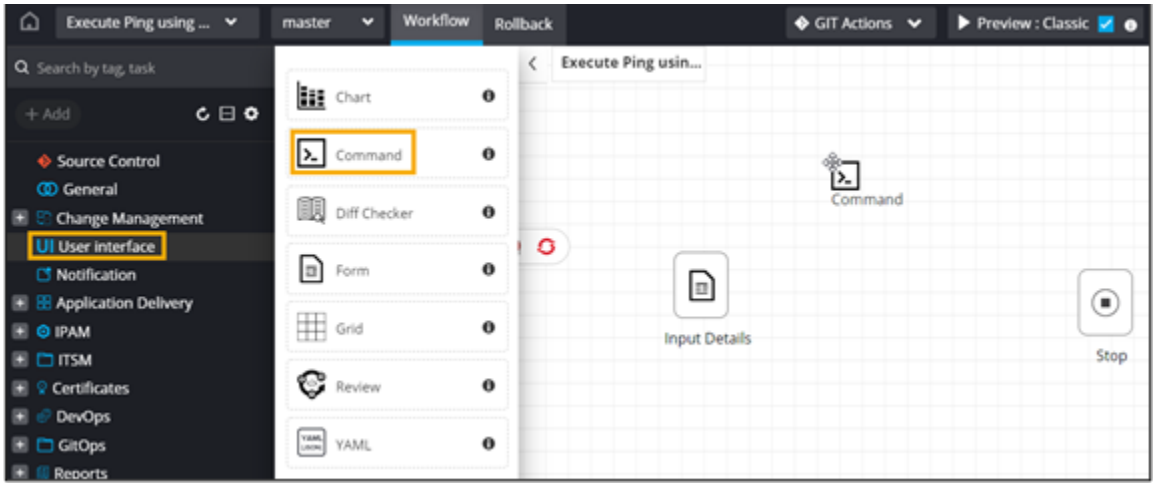
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



4. Declare the Field ID as a global variable, `<%hostname%>`, to be referred in the next task.

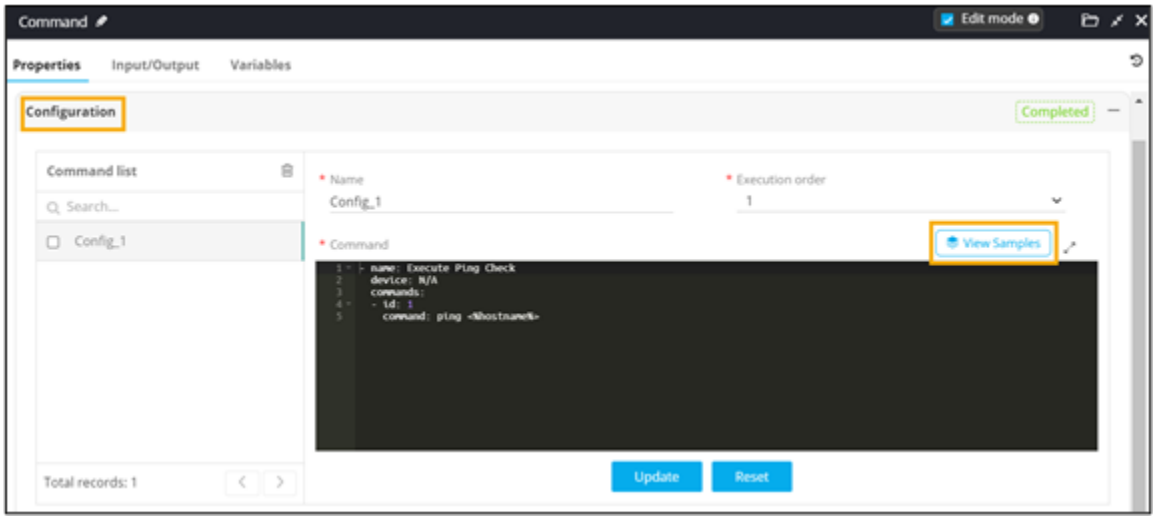
> **Note:** For more information on adding form fields, click here.

5. Click **Save**.
6. From the **User Interface** section, drag and drop the **Command** task.

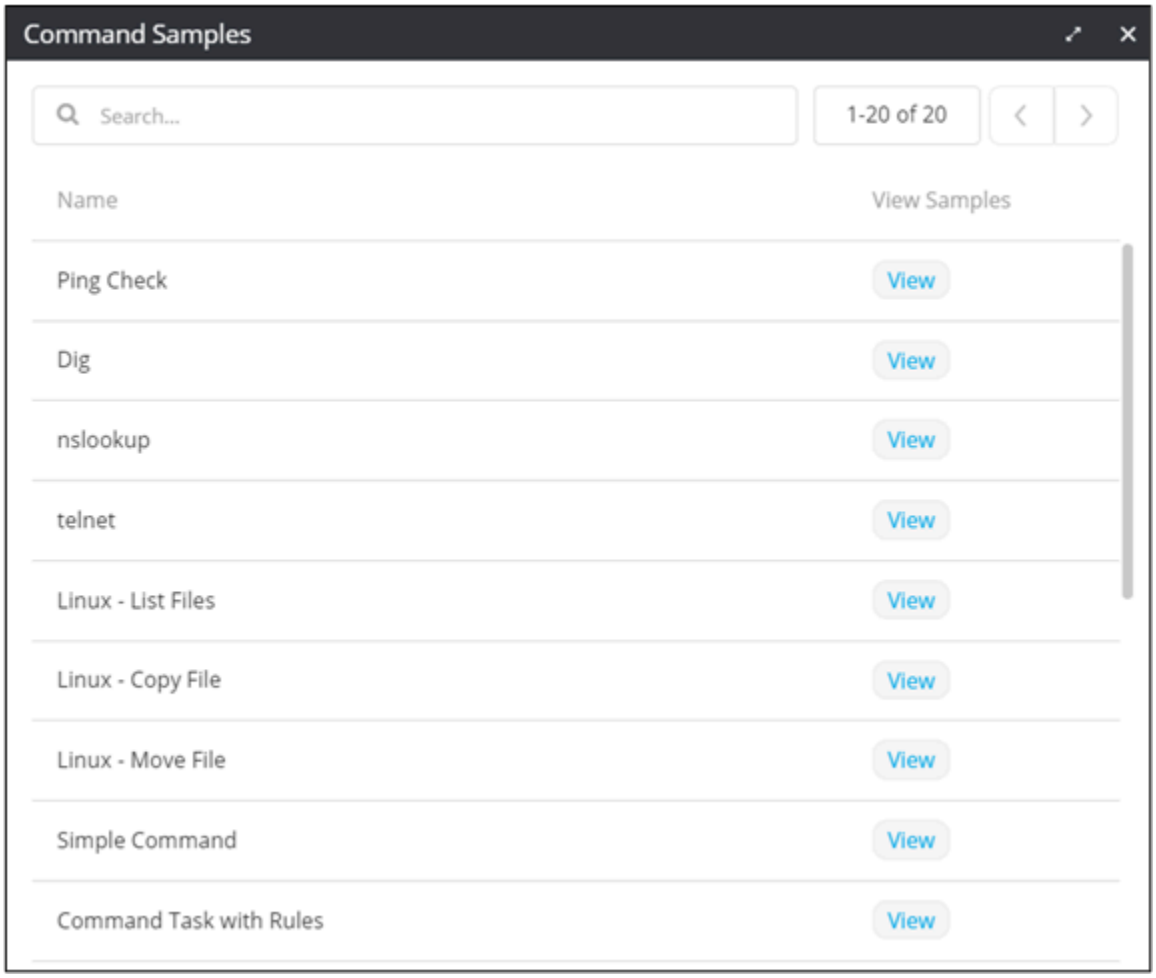


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

7. In the **Command** task window, under **Properties**, in the **Configuration** section, click **View Samples**.

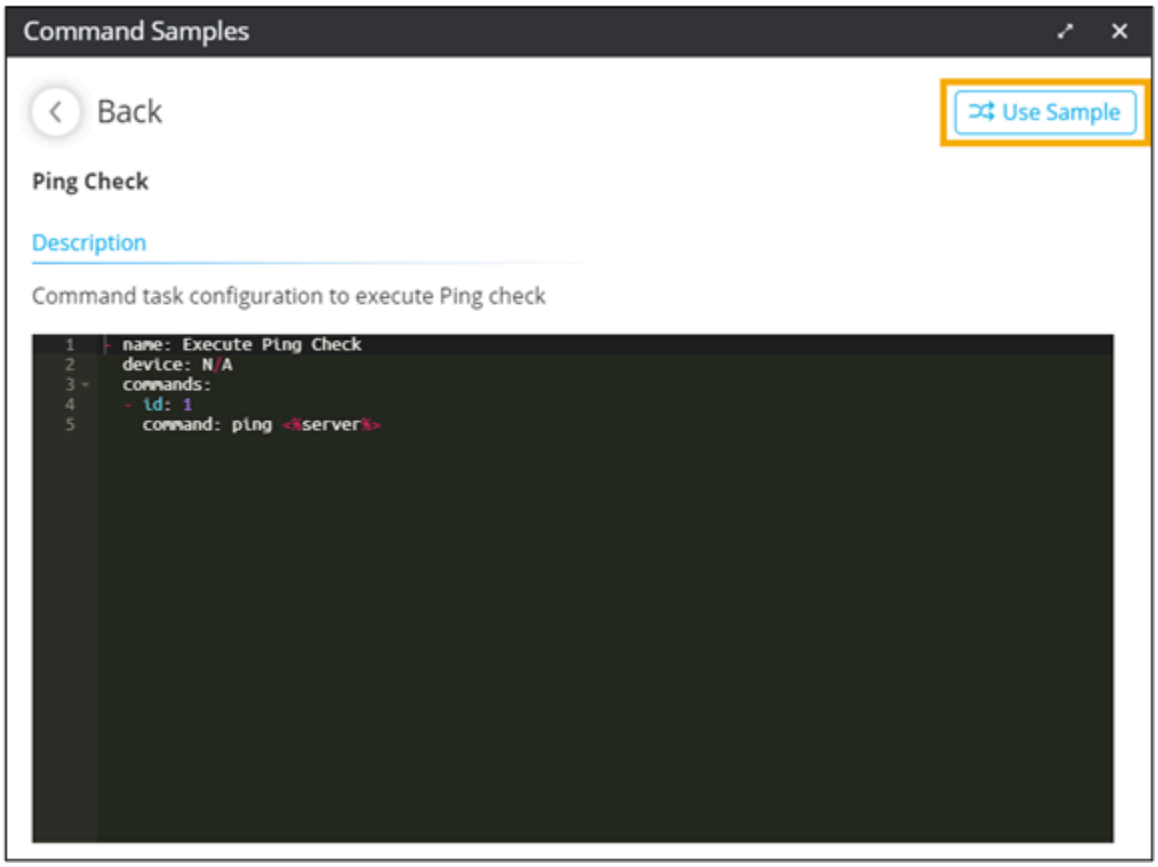


The **Command Samples** window is displayed.

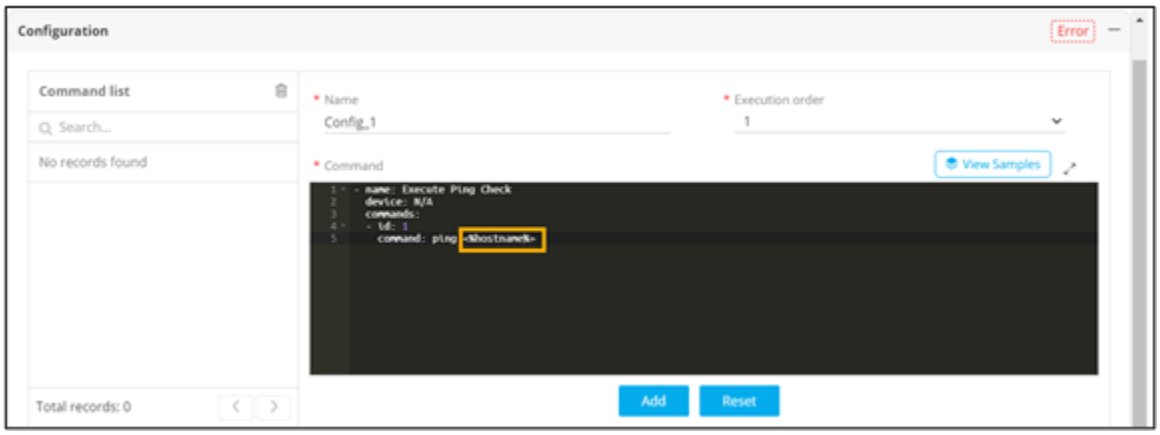


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

8. Click **View** next to the **Ping Check** command.
9. To use this command, click **Use Sample**.



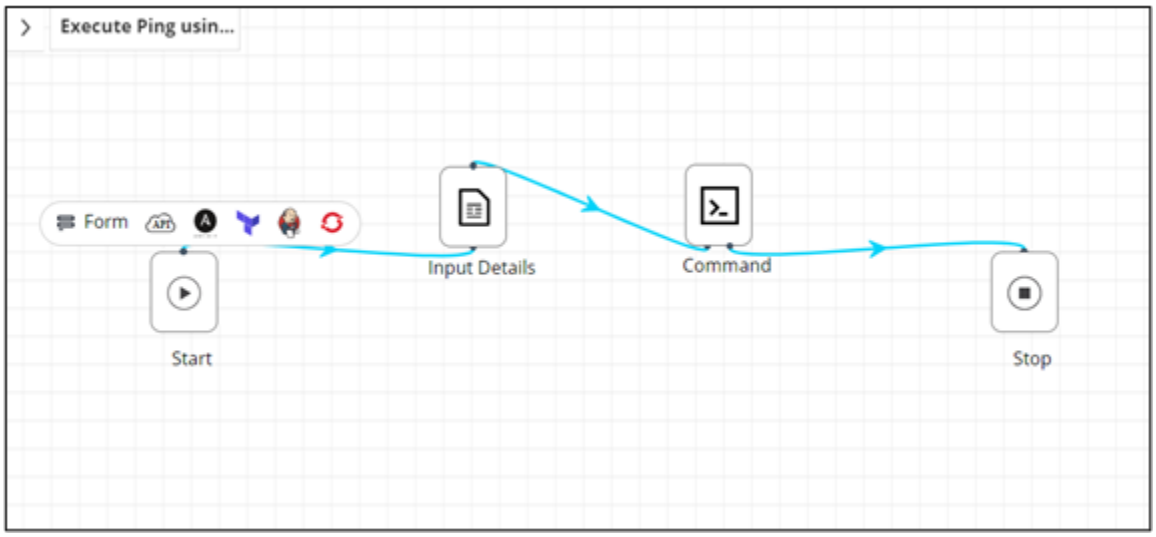
10. Refer the global variable <%hostname%> defined in the form task.



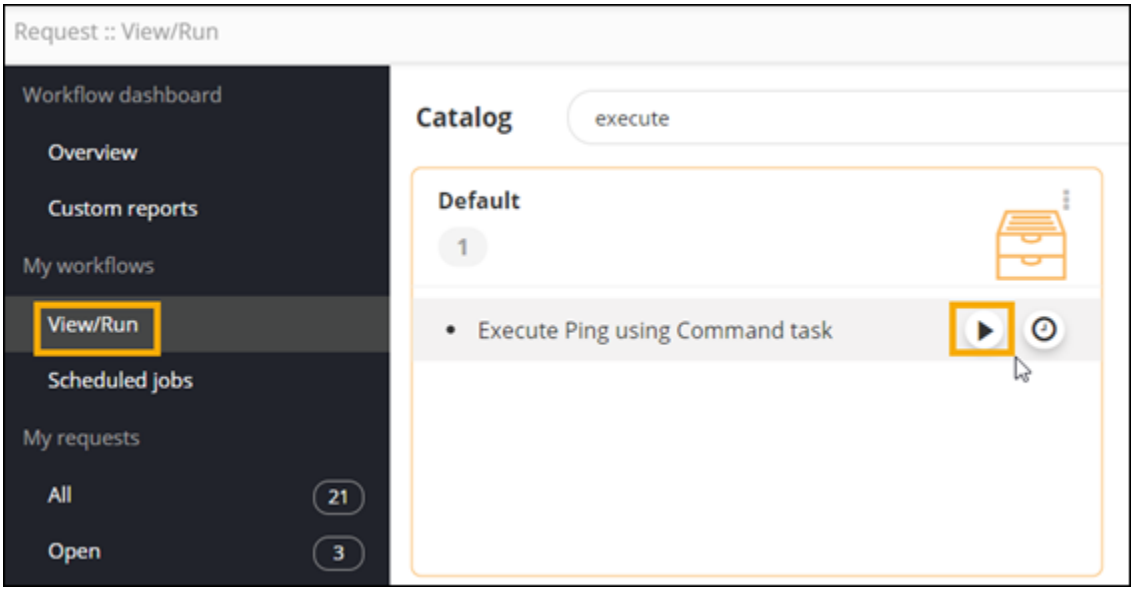
Note: device: N/A implies that the command will be executed on AppViewX server.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

11. To add this configuration, click **Add**.
12. Click **Save**.
13. Connect the workflow tasks.



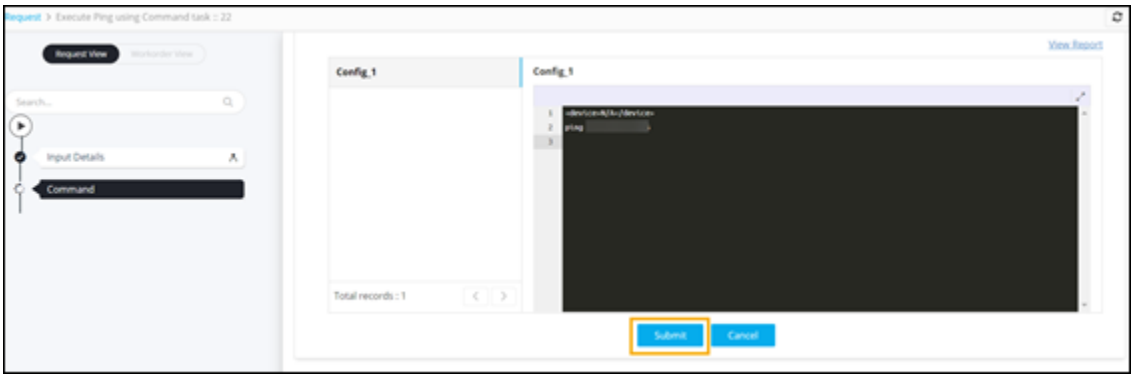
14. Enable the workflow.
15. From the upper left corner of the screen, click ☰.
16. From the menu displayed, select **Request**.
17. On the **Request :: Overview** page, from the navigation pane on the left, click **View/Run**.
18. To trigger the workflow, on the **Request :: View/Run** page, search for the **Execute Ping using Command task** workflow and click ▶.



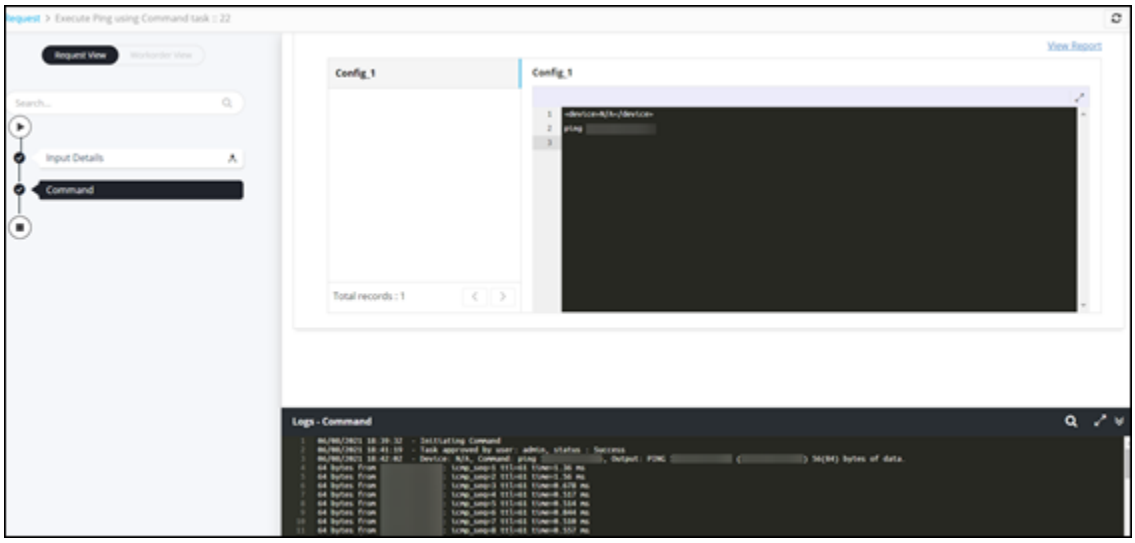
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

19. Enter the Server IP in the form.

20. Click **Submit**.
21. To execute the command for performing ping checks on the server, click **Submit**.

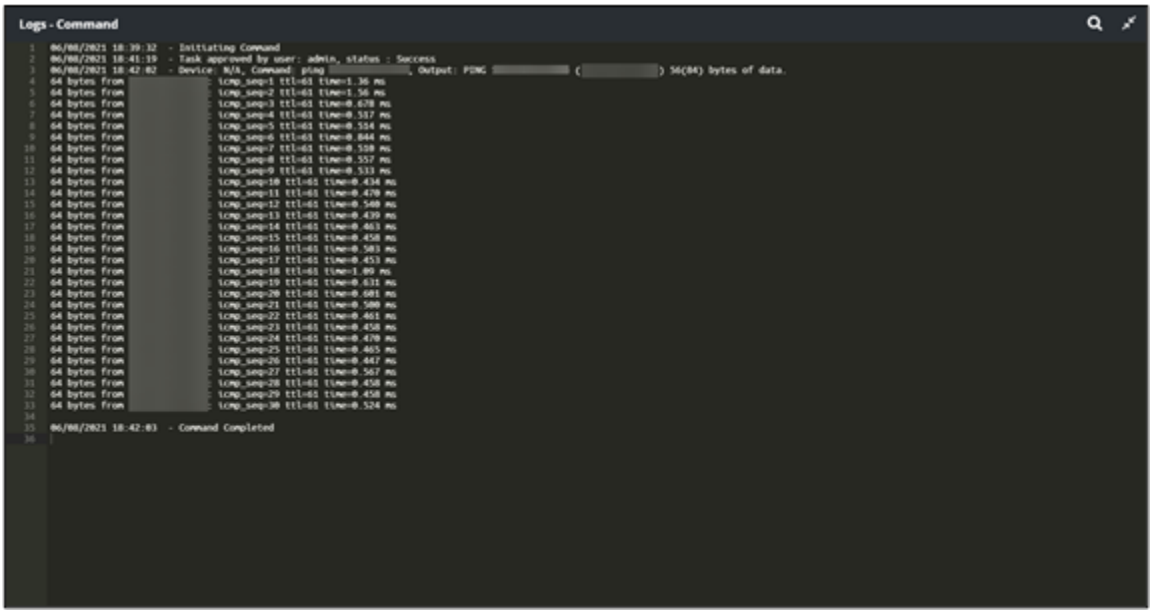


- Command is executed.



- Logs showing the server response.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



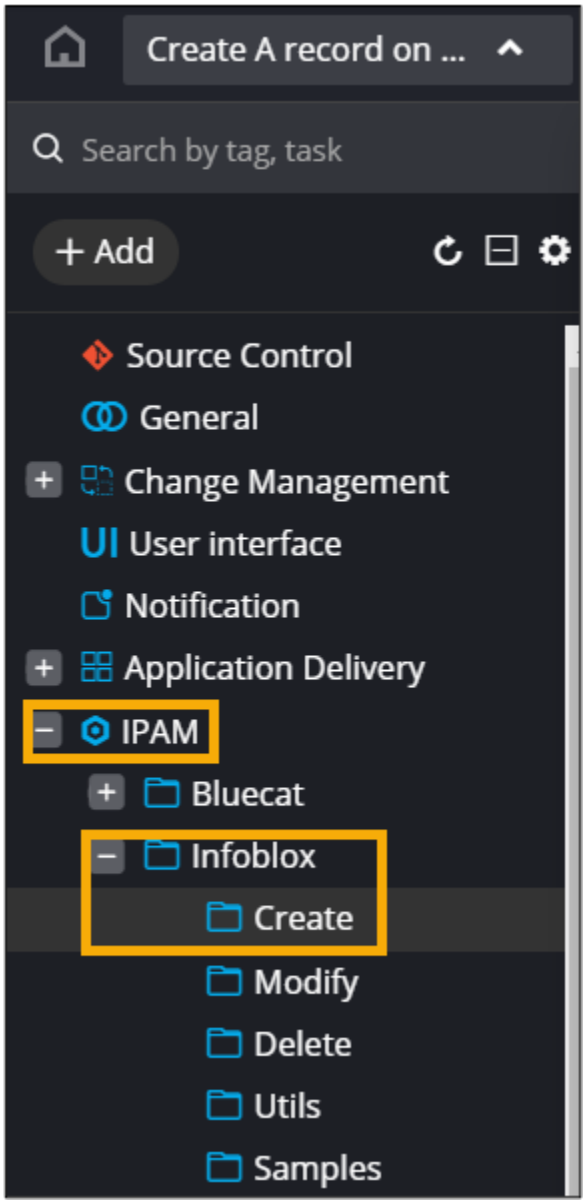
Creating A Record on Infoblox device using prebuilt tasks

You can design a workflow with an auto-generated form to create A record on an Infoblox device using prebuilt tasks.

Note: Before creating the workflow, ensure that the device(s) are available in the Device Inventory.

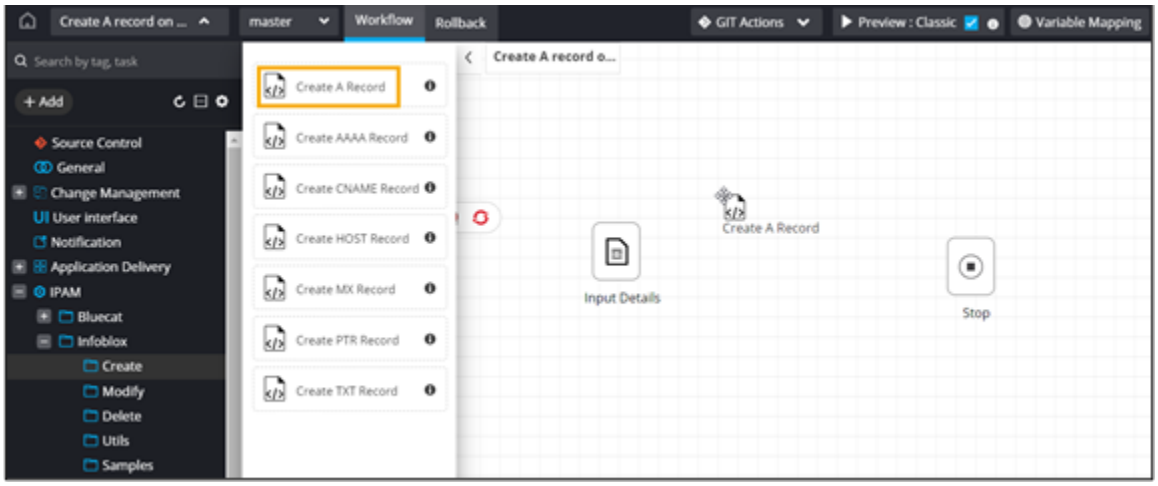
1. Design a new workflow.
2. From the navigation pane on the left, click next to the **IPAM** folder.
3. Under **IPAM**, click **Infoblox**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

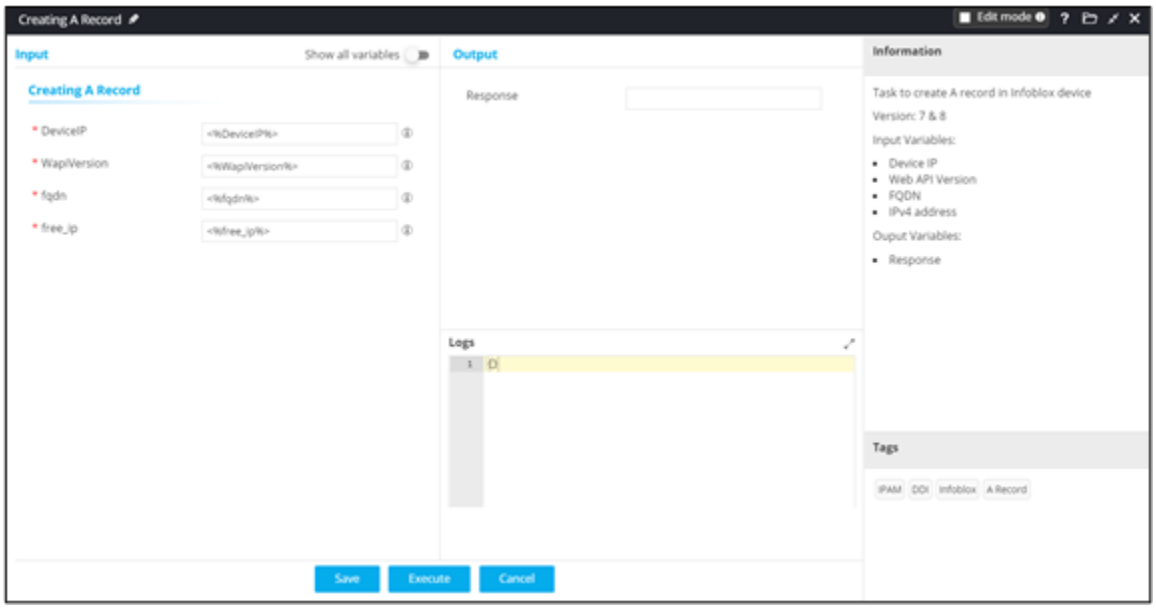


4. From the **Create** folder, drag and drop the prebuilt task **Create A Record**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



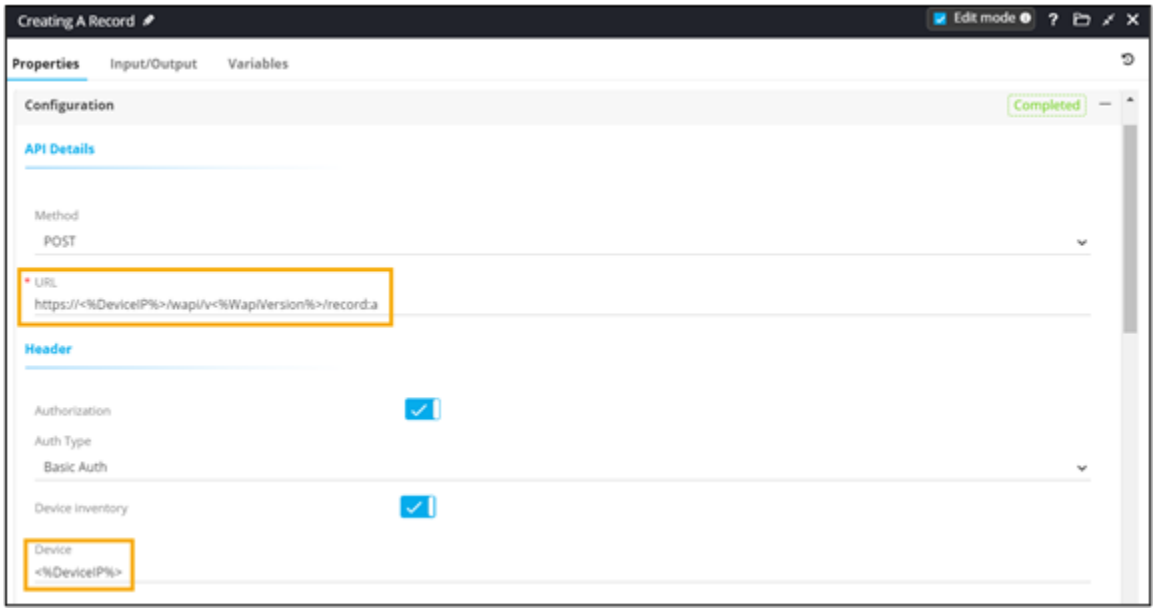
The **Creating A Record** window shows the **Input** variables required for this task in the Citizen mode.

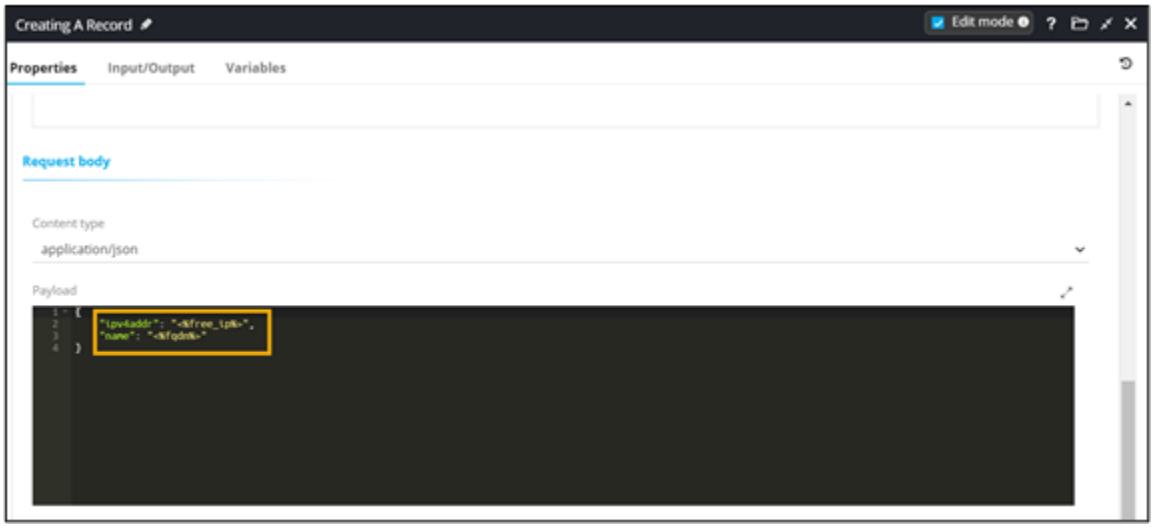


Note: For more information on switching between Edit mode and Citizen mode, refer to the section on Edit mode/Citizen mode.

5. To view the API configuration details, in the **Creating A Record** window, select the **Edit mode** checkbox.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.





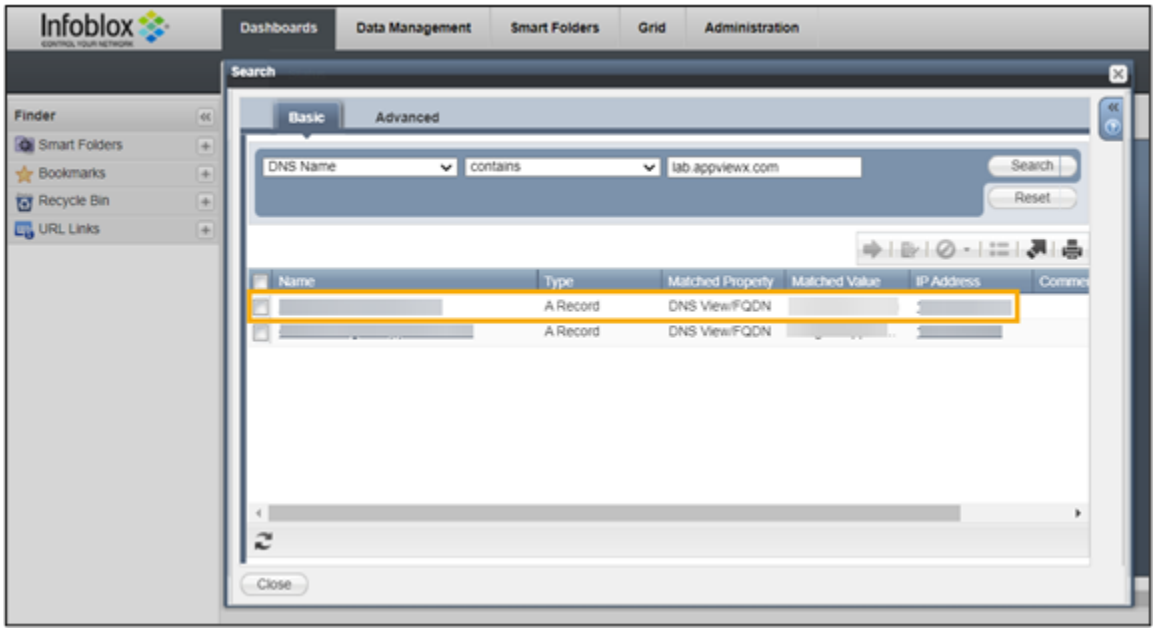
6. Click **Save**.
7. To design an auto-generated form, click **Form** above the **Start** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

8. To add form fields in the Auto Generate Form, click ⊕ above the **Creating A Record** task.
9. Enter the form details and click **Create**.
10. Connect and enable the workflow.
11. From the upper left corner of the screen, click ☰.
12. From the menu displayed, select **Request**.
13. On the **Request :: Overview** page, from the navigation pane on the left, click **View/Run**.
14. To trigger the workflow, on the **Request :: View/Run** page, search for the workflow and click ▶.
15. Enter the input details in the form.
16. Click **Next**.

    Creating A record task is completed. A record created on Infoblox.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Creating a VIP with Incident Ticket

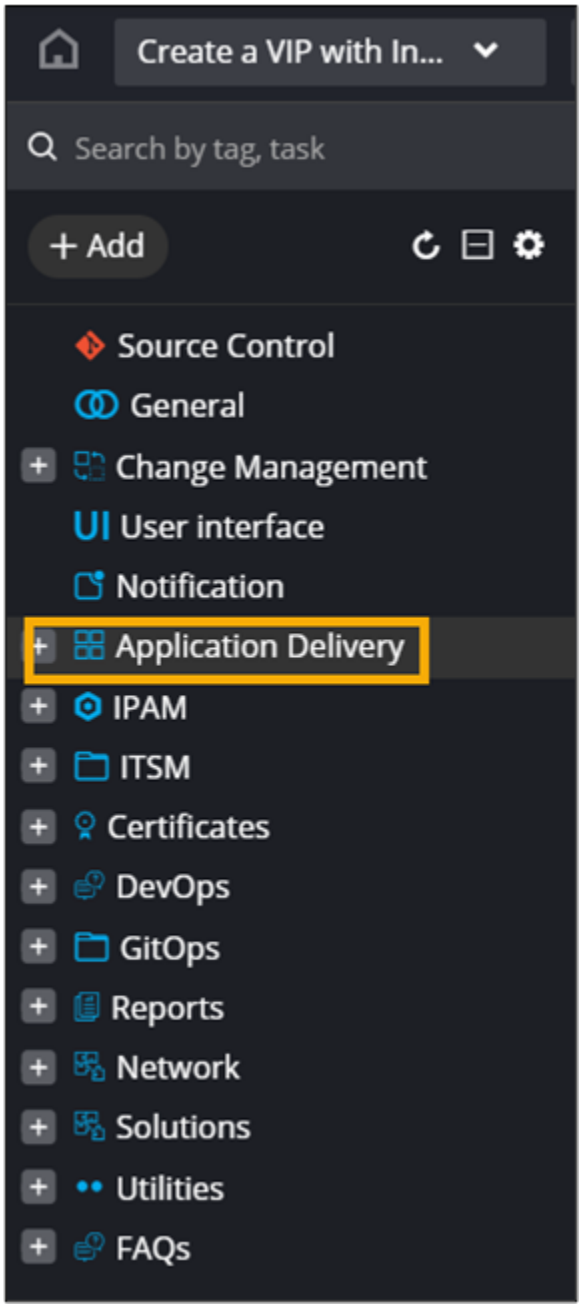
The ITSM folder allows you to leverage the prebuilt tasks for designing ITSM automation workflows.

Note: Before designing the workflow, ensure that you have configured an instance to integrate with the ITSM vendor. For more information, refer to the section on ITSM Vendor Configuration

To create a VIP with Incident ticket:

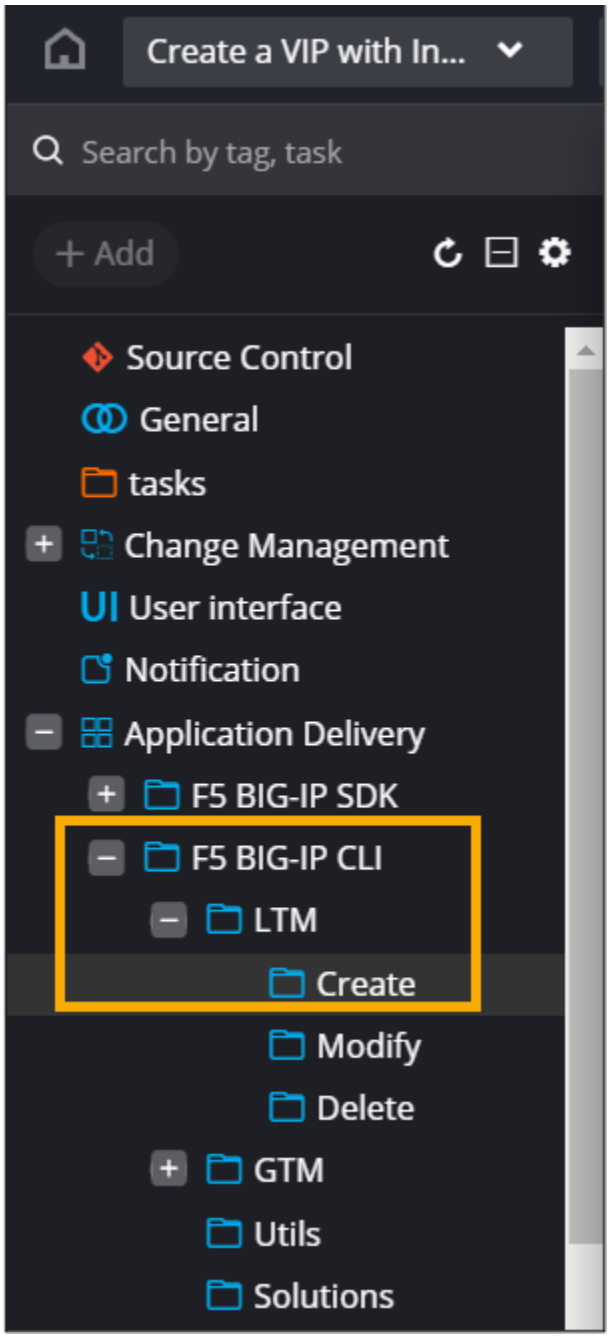
1. Design a new workflow.
2. From the navigation pane on the left, click + next to **Application Delivery**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



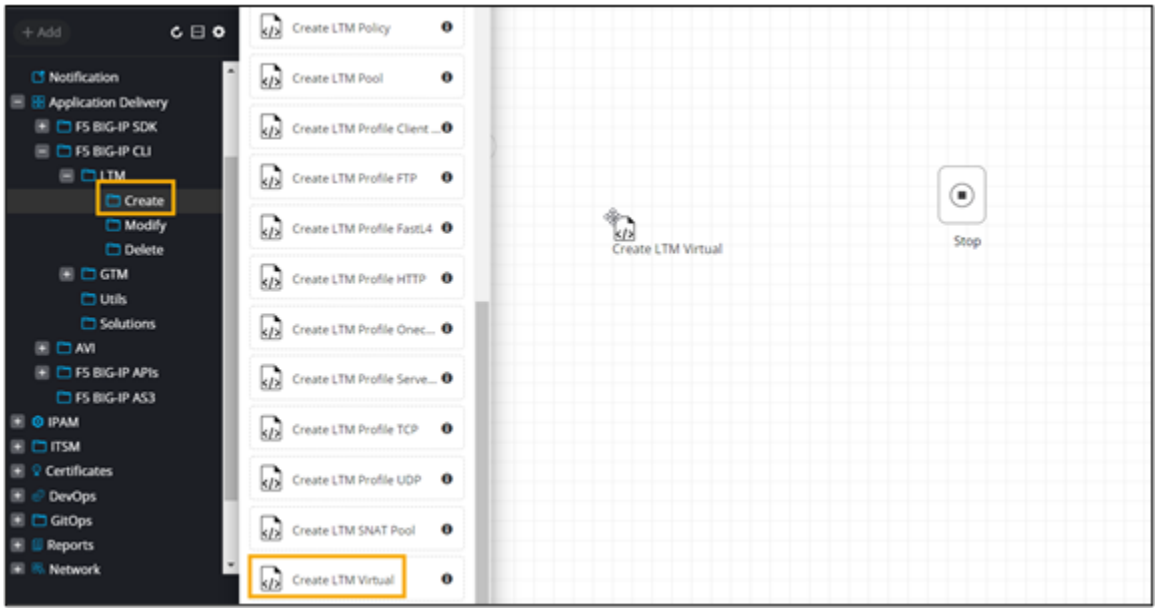
3. Under **Application Delivery**, select **F5 BIG-IP CLI Commands** > **LTM** > **Virtual**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

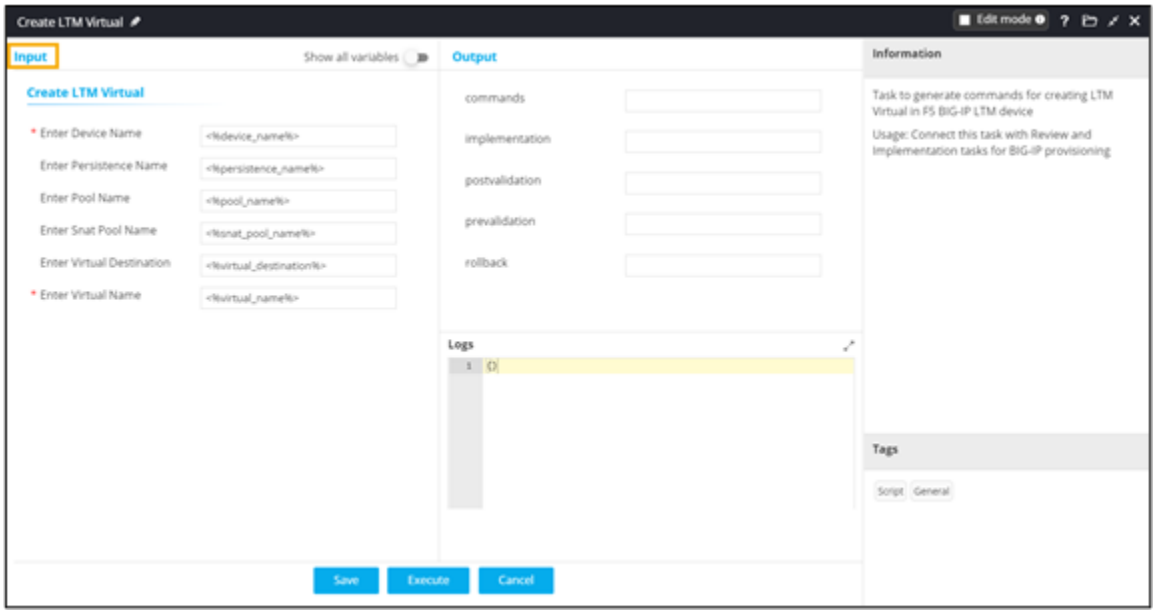


4. To create a virtual LTM server, drag and drop the prebuilt **Create LTM Virtual task**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

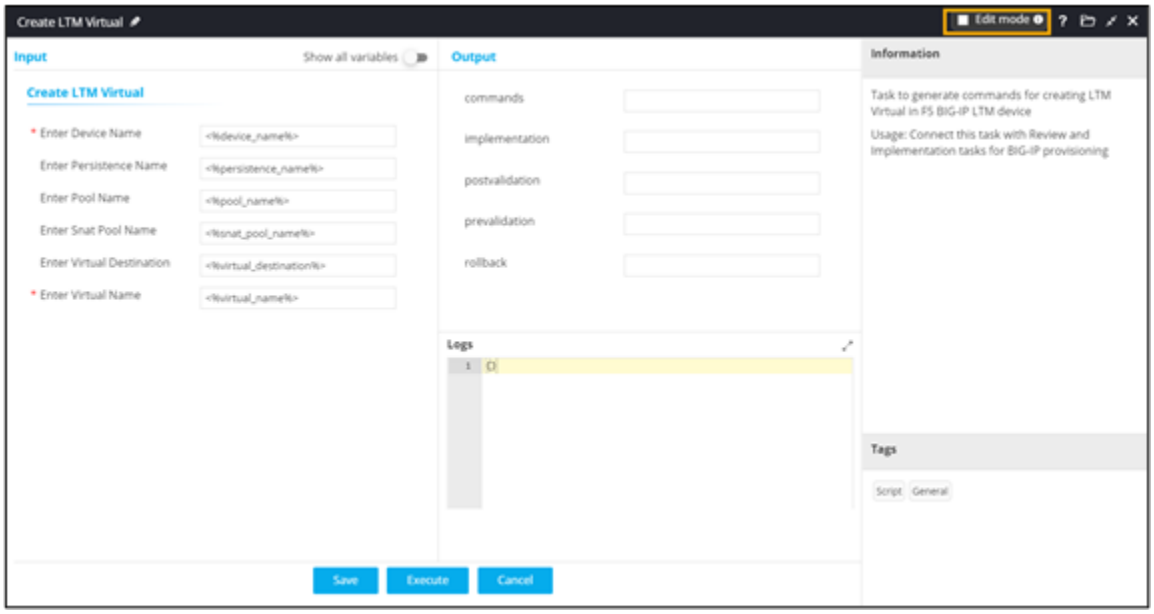


The **Create LTM Virtual** window shows the **Input** variables required for this task in the Citizen mode.

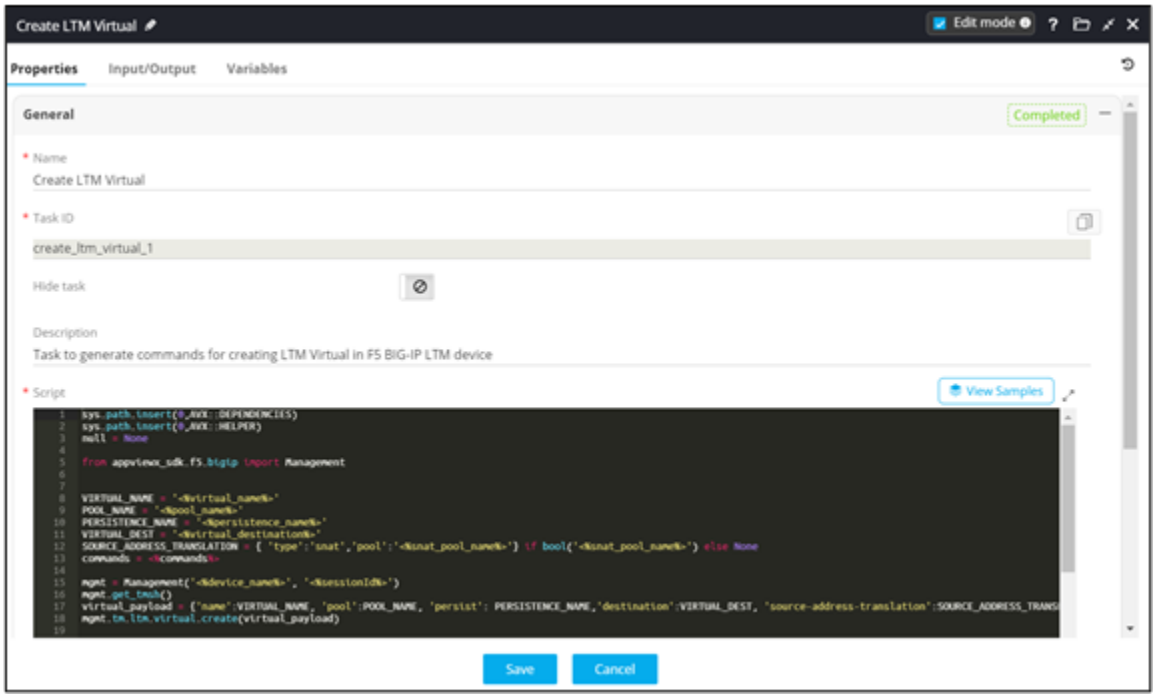


5. To view or edit the task configurations, in the **Create LTM Virtual** window, select the **Edit mode** checkbox.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

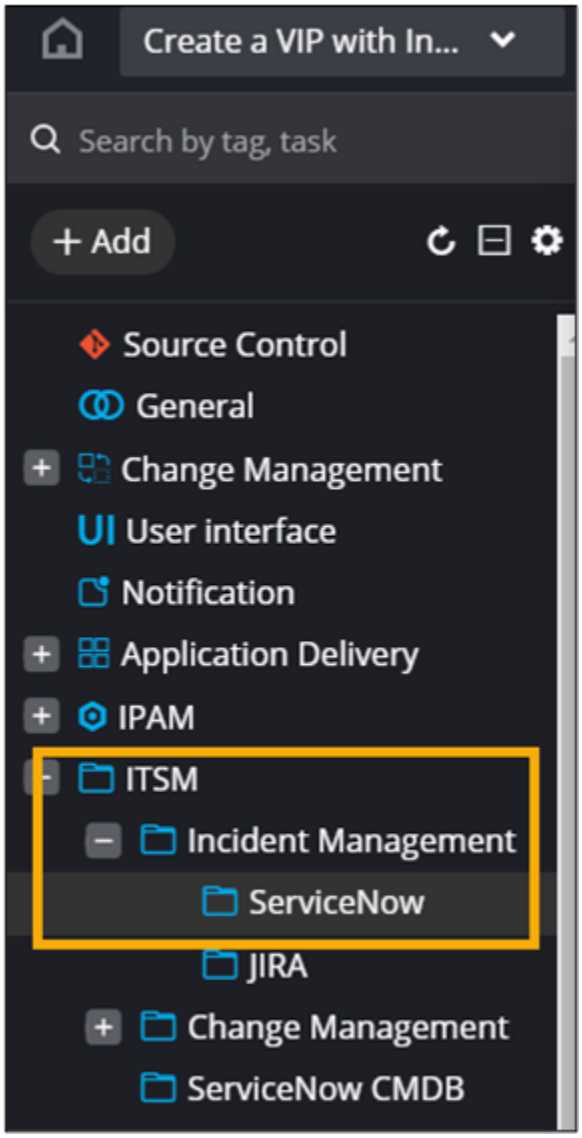


Create LTM Virtual task in Edit mode.



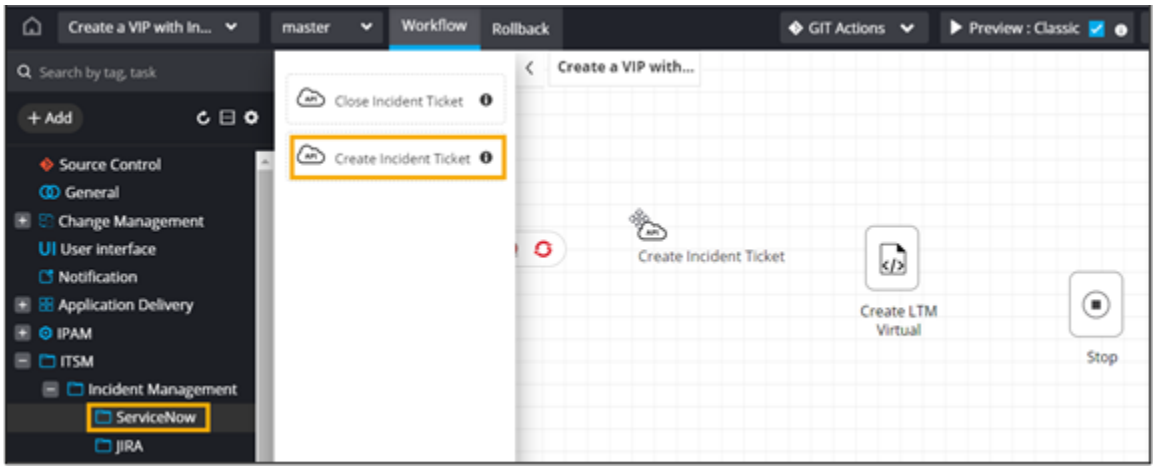
6. Click **Save**.
7. From the navigation pane on the left, click next to **ITSM**.
8. Under **ITSM**, select **Incident Management** > **ServiceNow**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

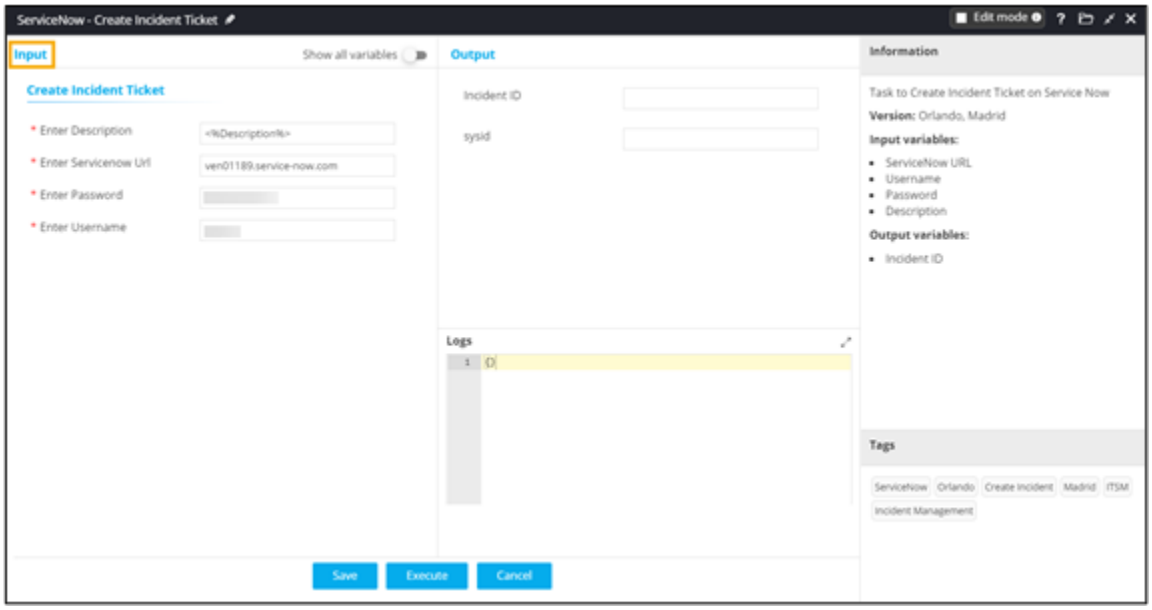


9. Drag and drop the **Create Incident Ticket** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

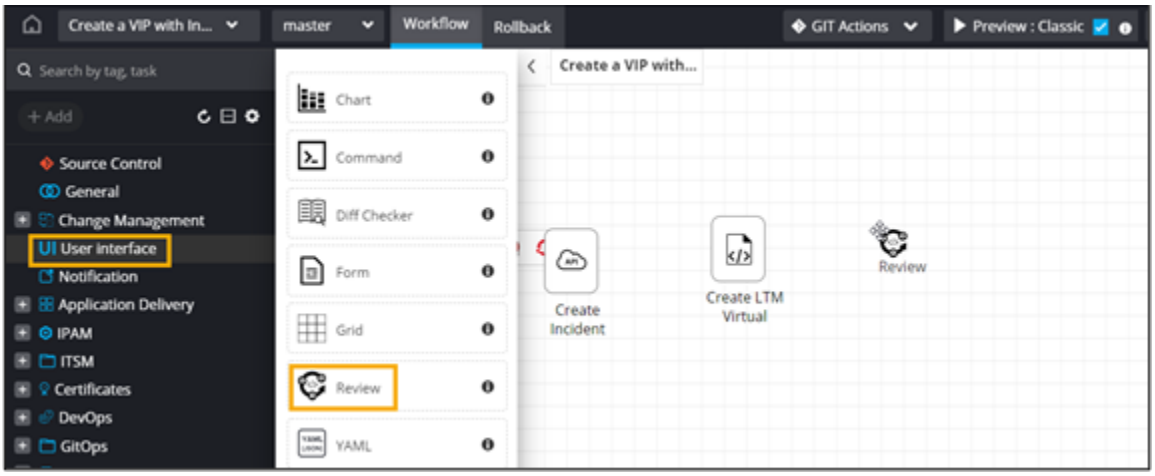


10. In the **ServiceNow - Create Incident Ticket** window, enter the **Input** variables required to create the Incident ticket.

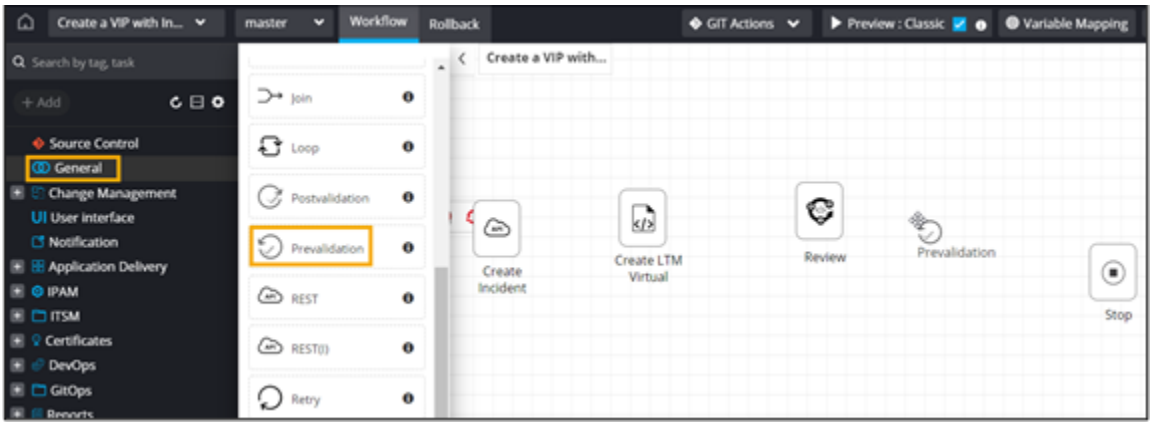


11. Click **Save**.
12. From the **User Interface** section, drag and drop a **Review** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

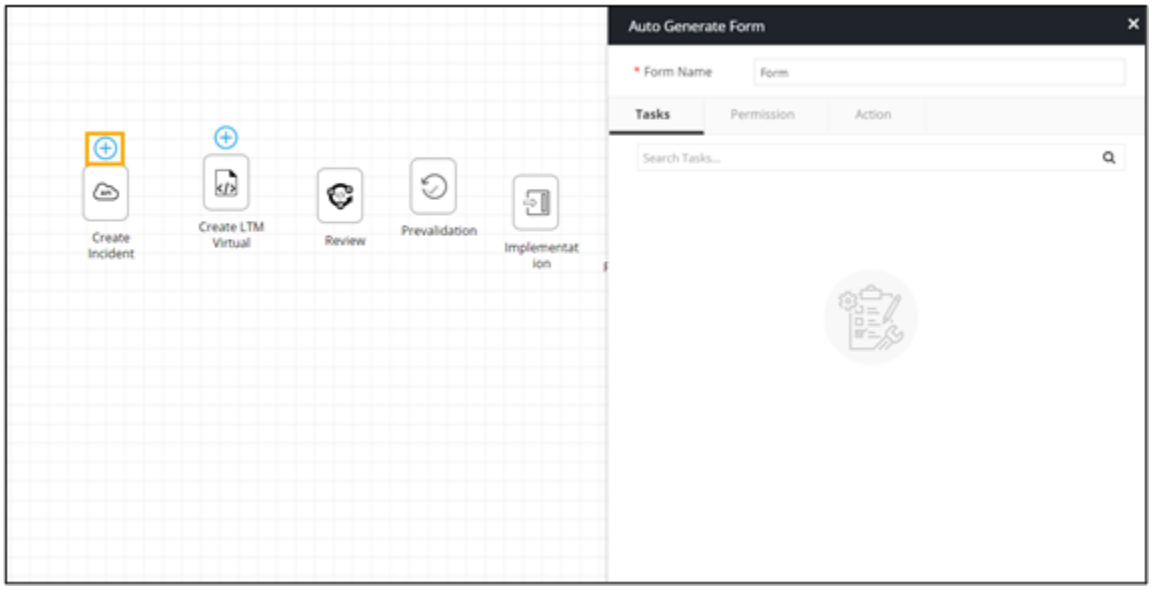


13. Click **Save**.
14. From the **General** section, drag and drop the **Prevalidation** task.

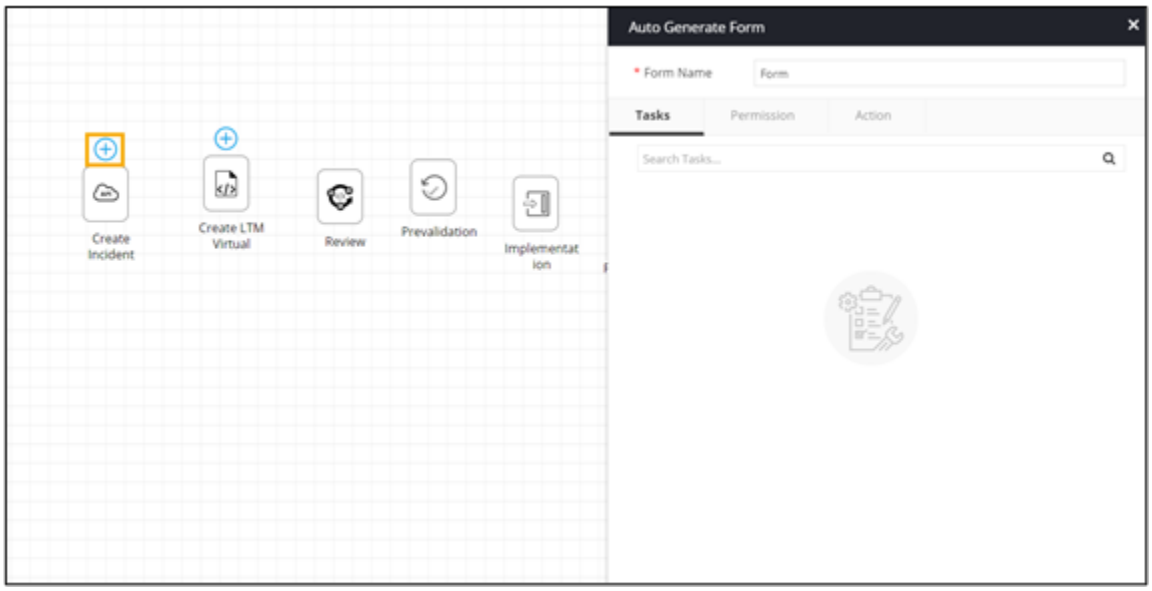


15. Click **Save**.
16. From the **General** section, drag and drop the **Implementation** task.
17. Click **Save**.
18. From the **General** section, drag and drop the **Postvalidation** task.
19. Click **Save**.
20. To create an auto-generated form, click **Form** above the **Start** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

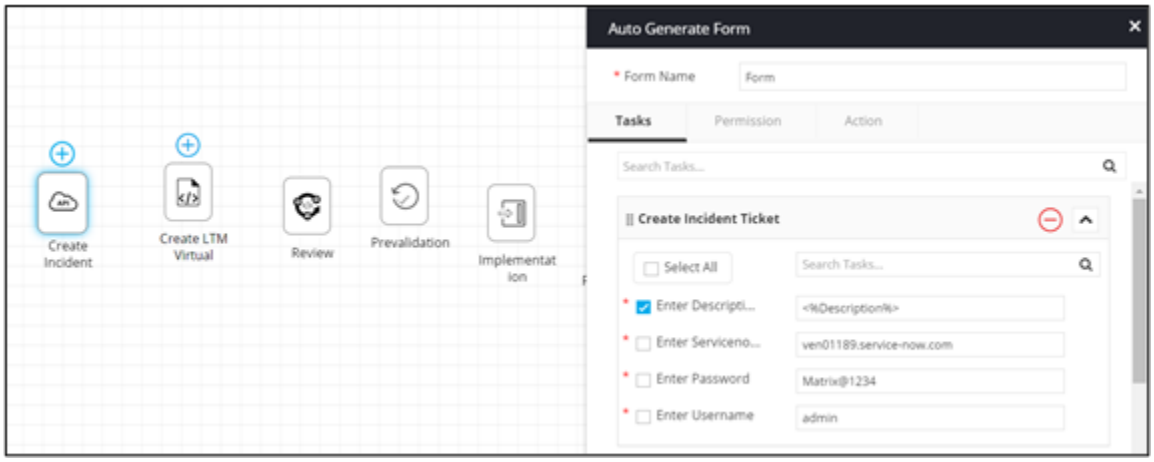


21. To auto-fill the variables, click ⊕ above the **Create Incident** task.



22. Select the relevant form fields.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

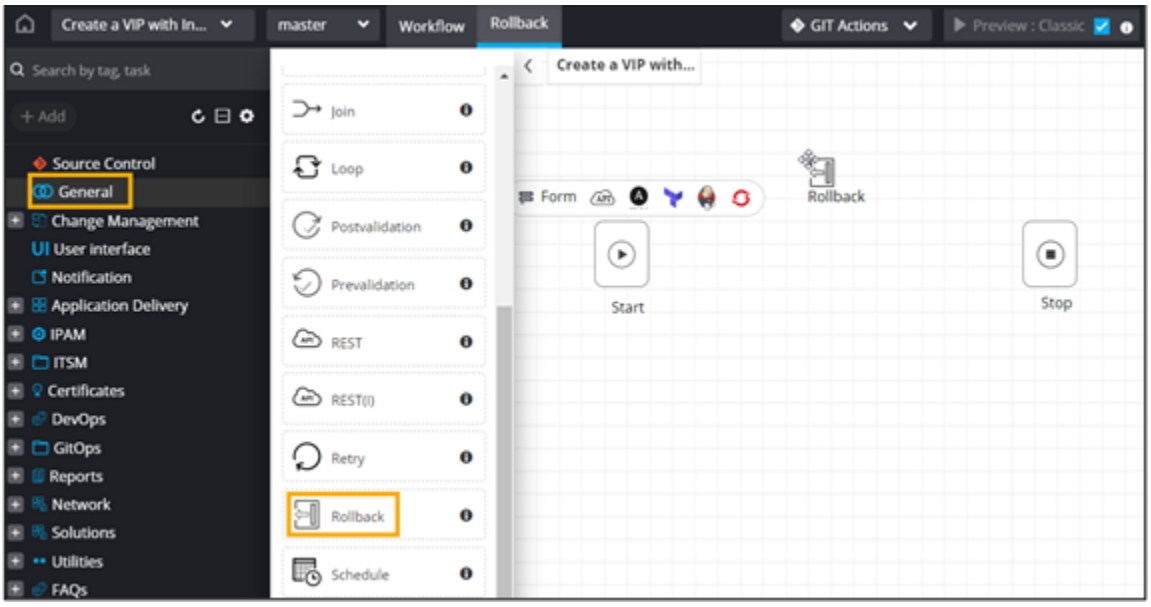


23. Click ⊕ above the **Create LTM Virtual** task.
24. Select the relevant form fields. Creating a VIP with Incident Ticket
25. Click **Create**.
26. Connect the workflow tasks.

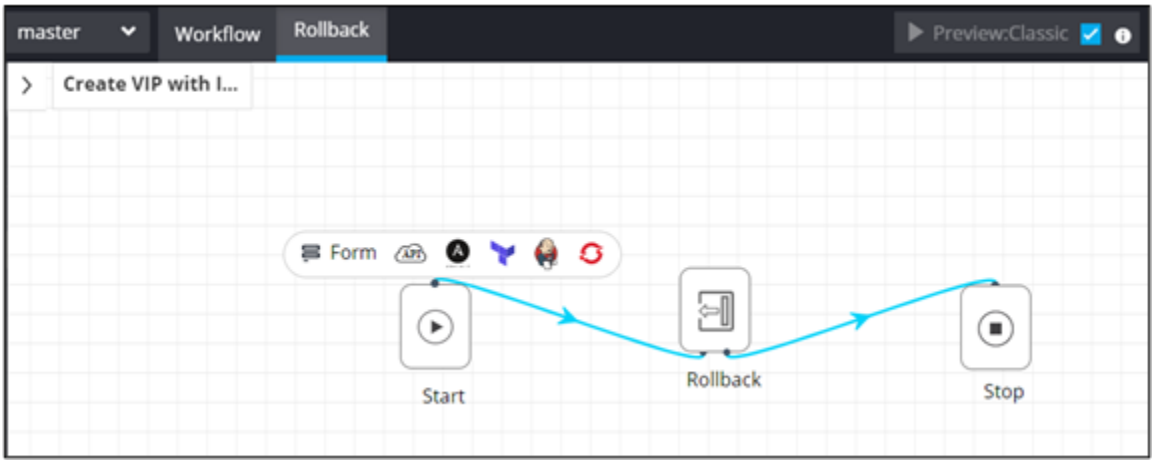
27. To add a rollback workflow, click **Rollback**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

28. From the **General** section, drag and drop the **Rollback** task.



29. Click **Save**.
30. Connect the rollback workflow tasks.

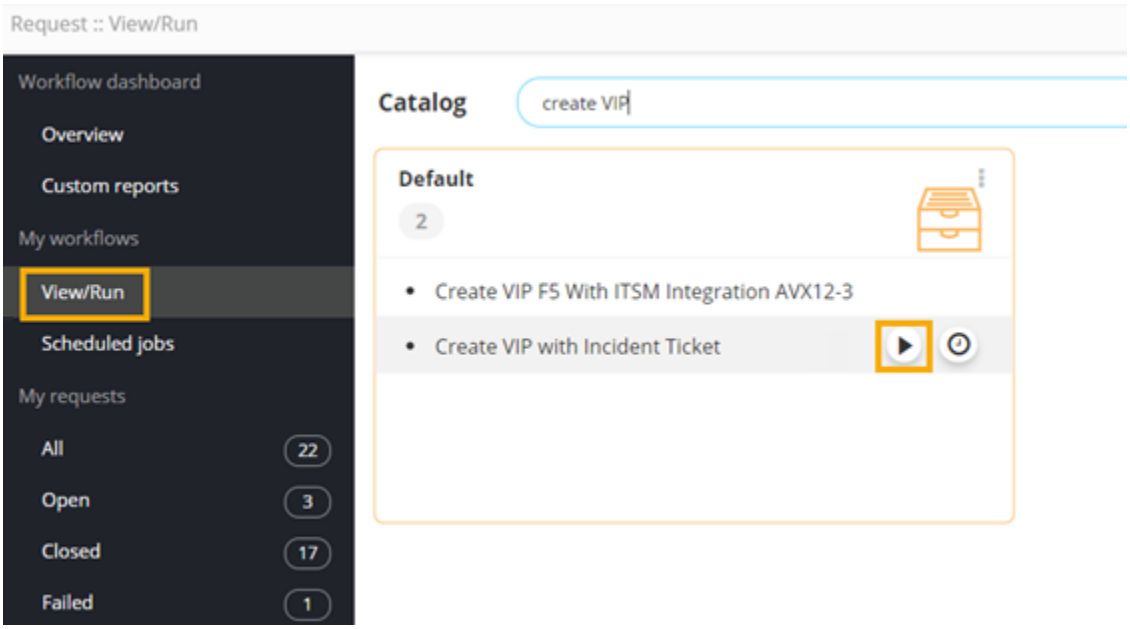


> **Note:** No values are passed in the Review, Prevalidaton, Implementation, Postvalidation, and Rollback tasks, since these tasks are already mapped to the Create LTM Virtual task as output .

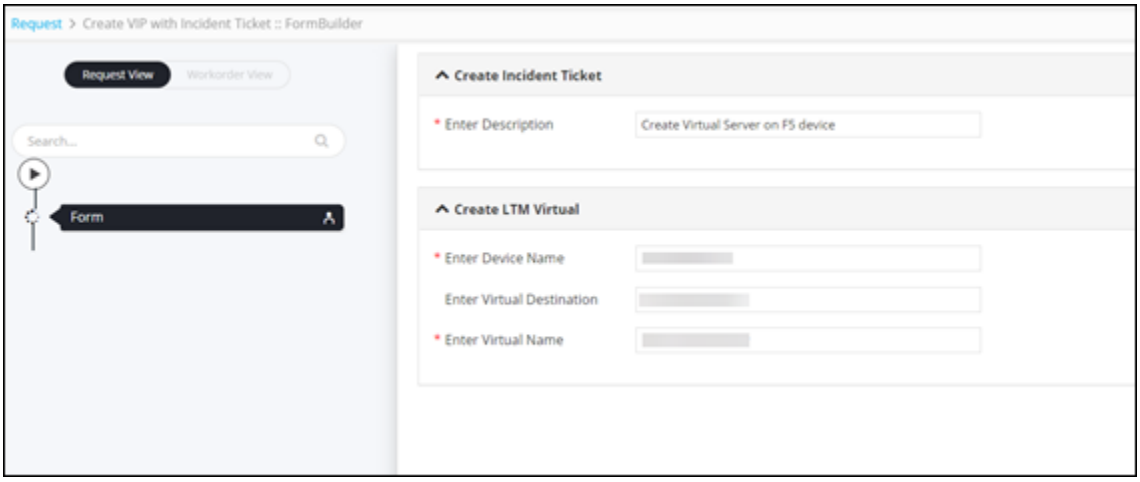
31. Enable the workflow.
32. From the upper left corner of the screen, click ☰.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

33. From the menu displayed, select **Request**.
34. On the **Request:: Overview** page, from the navigation pane on the left, click **View/Run**.
35. To trigger the workflow, on the **Request :: View/Run** page, search for the workflow and click .



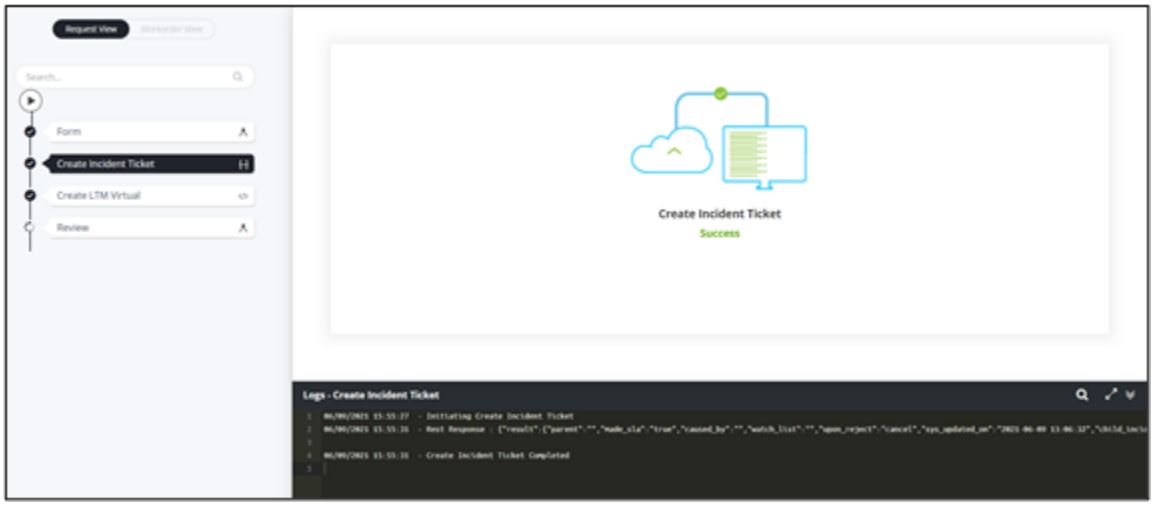
36. Enter the input details in the form.



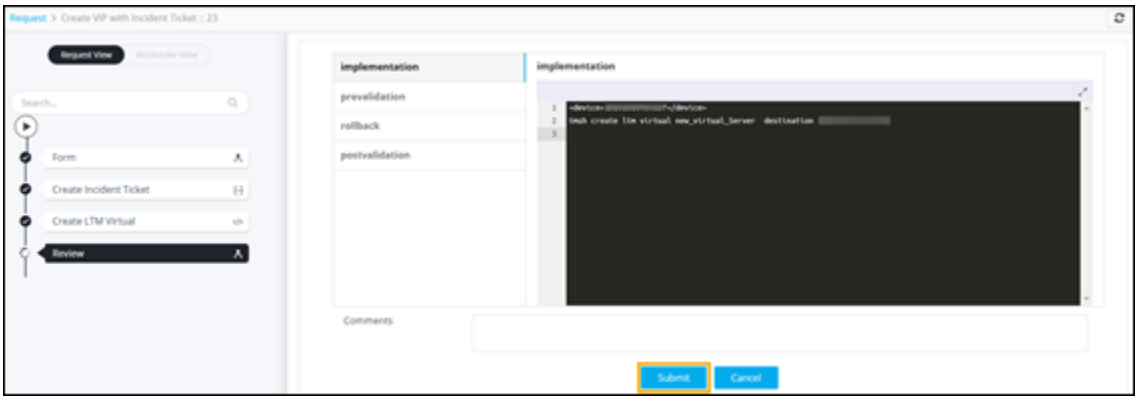
37. Click **Next**.

    Incident Ticket is created on ServiceNow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

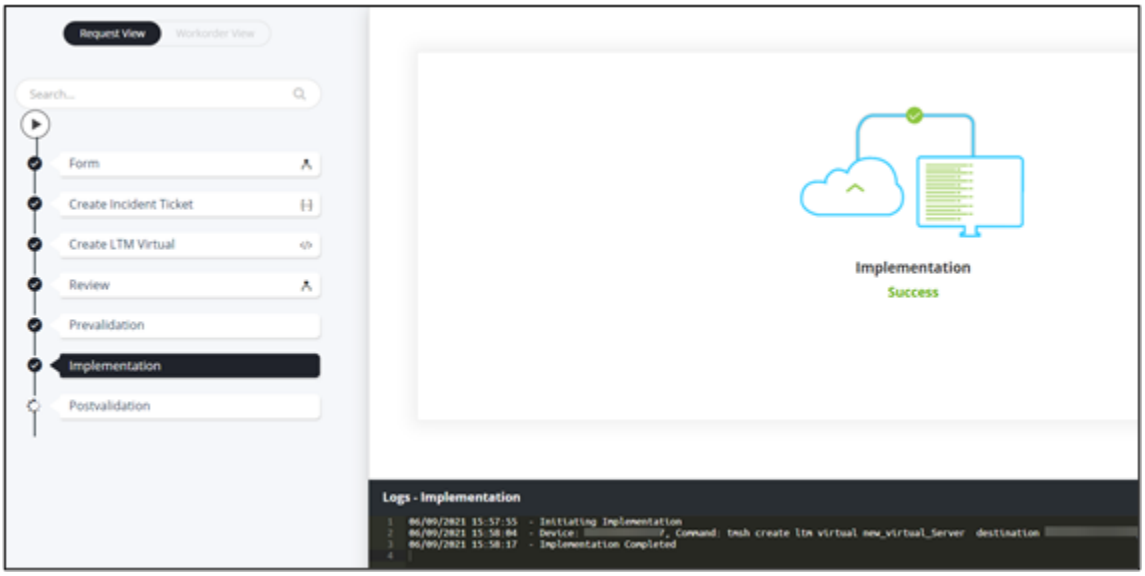


38. At the **Review** stage, to execute the implementation, prevalidation, rollback, and postvalidation commands, click **Submit**.

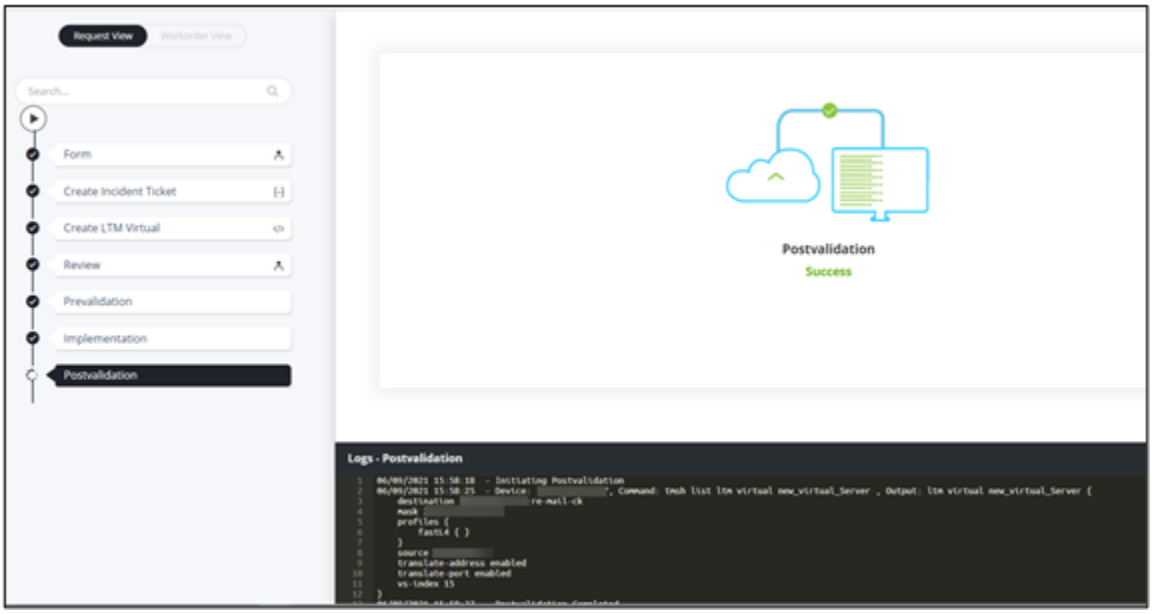


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Command to create a virtual server executed at the **Implementation** stage.

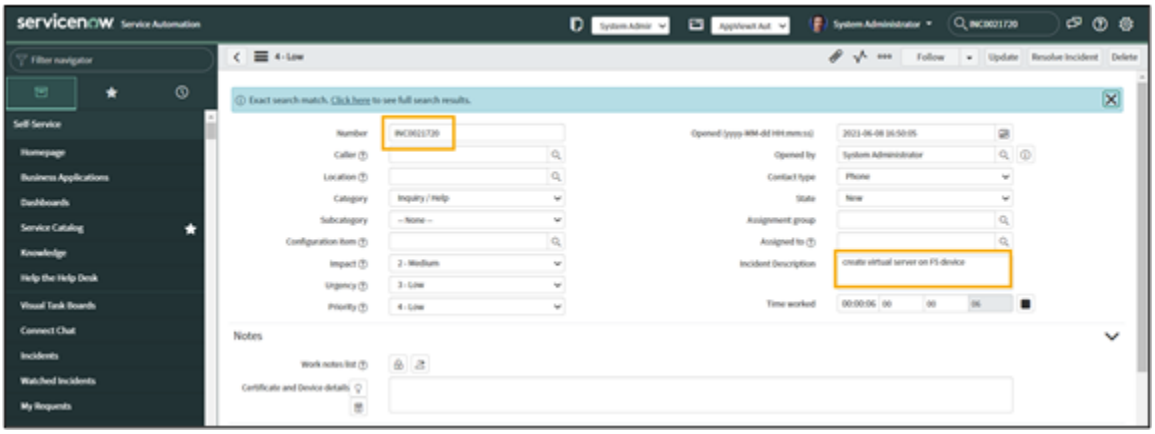


- **Postvalidation** command displaying the configuration of the new virtual server that was created.

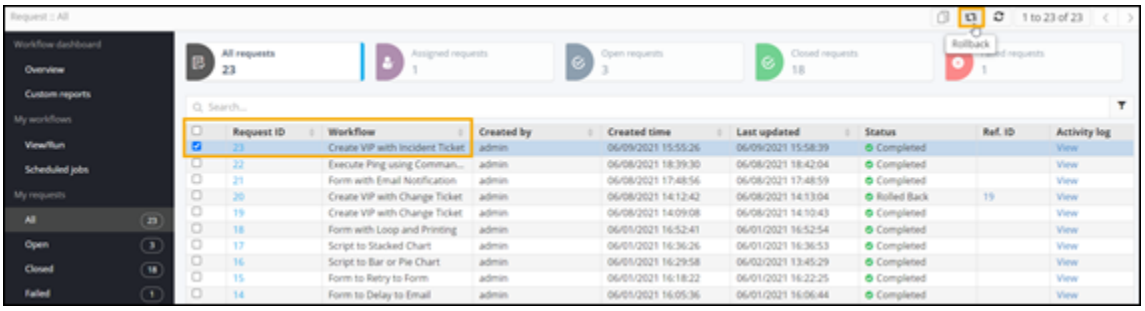


- Incident Ticket created on ServiceNow.

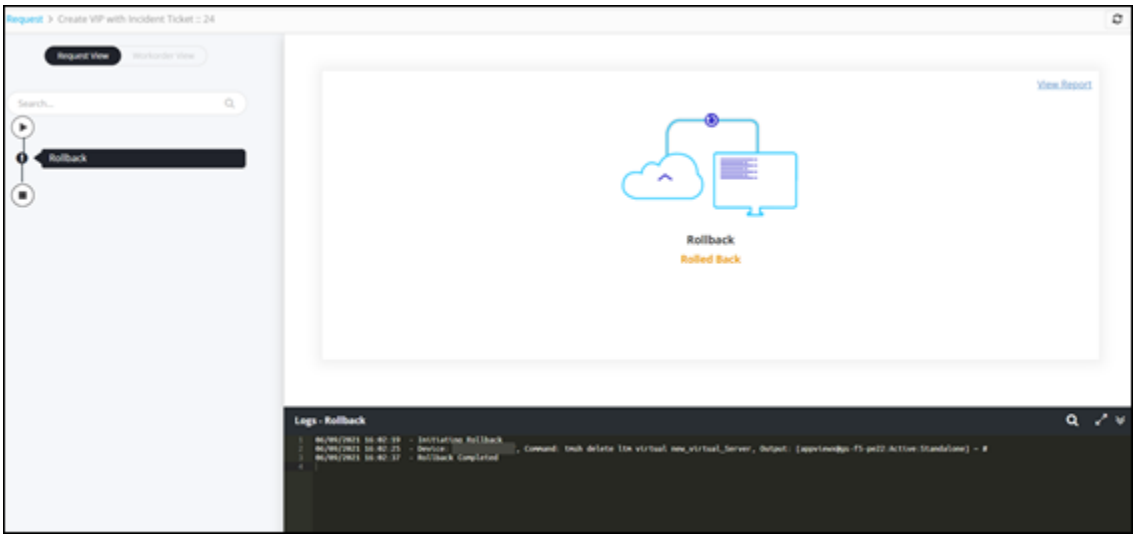
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



39. To execute the rollback workflow, on the **Request :: Overview** page, from the navigation pane on the left, click **All**.
40. On the **Request :: All** page, select the checkbox next to the workflow **Request ID**.
41. From the upper right corner of the screen, click .

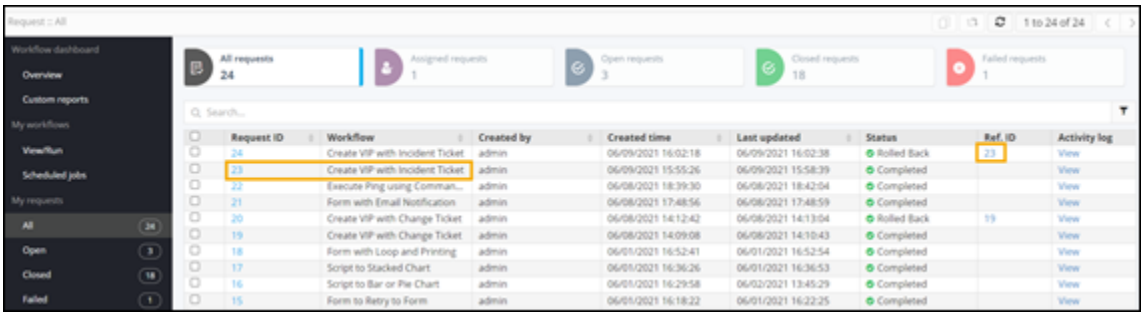


Rollback request to delete the virtual server that was created, executed successfully.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

42. To view the request ID of the rollback workflow, on the **Request :: Overall** page, click **All**.
    Rollback workflow request displays the **Ref. ID** of the workflow for which rollback was initiated.



Application Delivery Automation

With AppViewX's Application Delivery Automation, you can seamlessly deploy applications in heterogeneous environments (on-premise, multi-cloud, and hybrid) and simplify and accelerate application delivery.

- F5 BIG-IP Solutions
- AVI Solutions

F5 BIG-IP Solutions

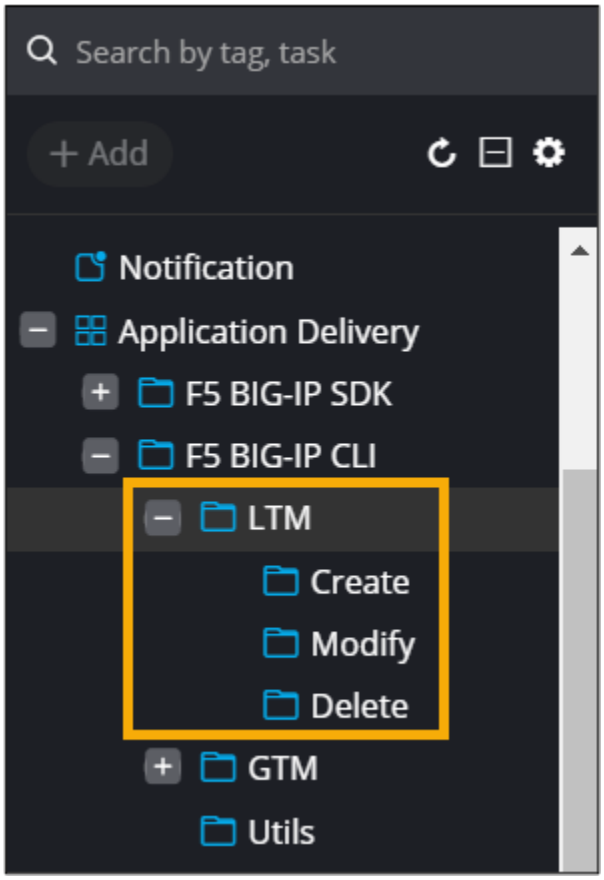
This section describes some of the workflows/tasks for F5 BIG-IP provisioning available in the Workflow Studio.

- LTM
- GTM
- Utils
- Creating a LTM Virtual Server on F5 BIG-IP

LTM

The workflows and tasks to create, modify, and delete LTM objects and devices on F5 BIG-IP are available under **Application Delivery** > **F5 BIG-IP CLI** > **LTM**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

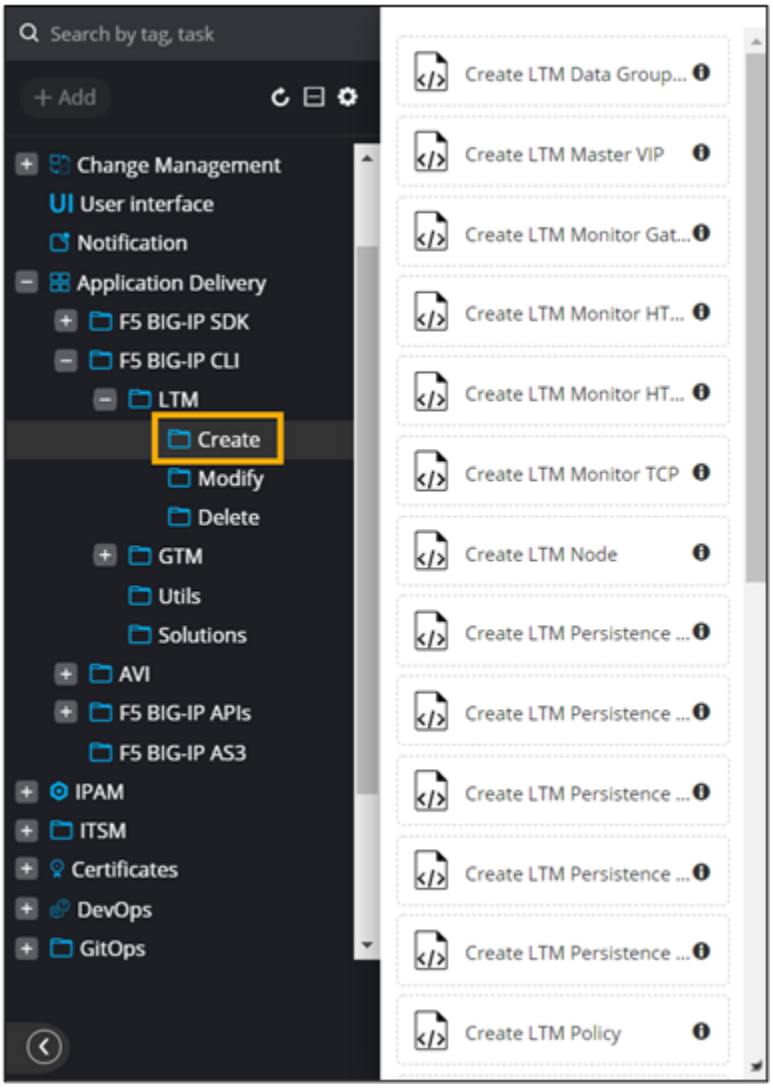


- LTM - Create
- LTM - Modify
- LTM - Delete

LTM - Create

The **Create** folder under **LTM** contains prebuilt tasks that can be leveraged to create LTM objects on F5 BIG-IP.

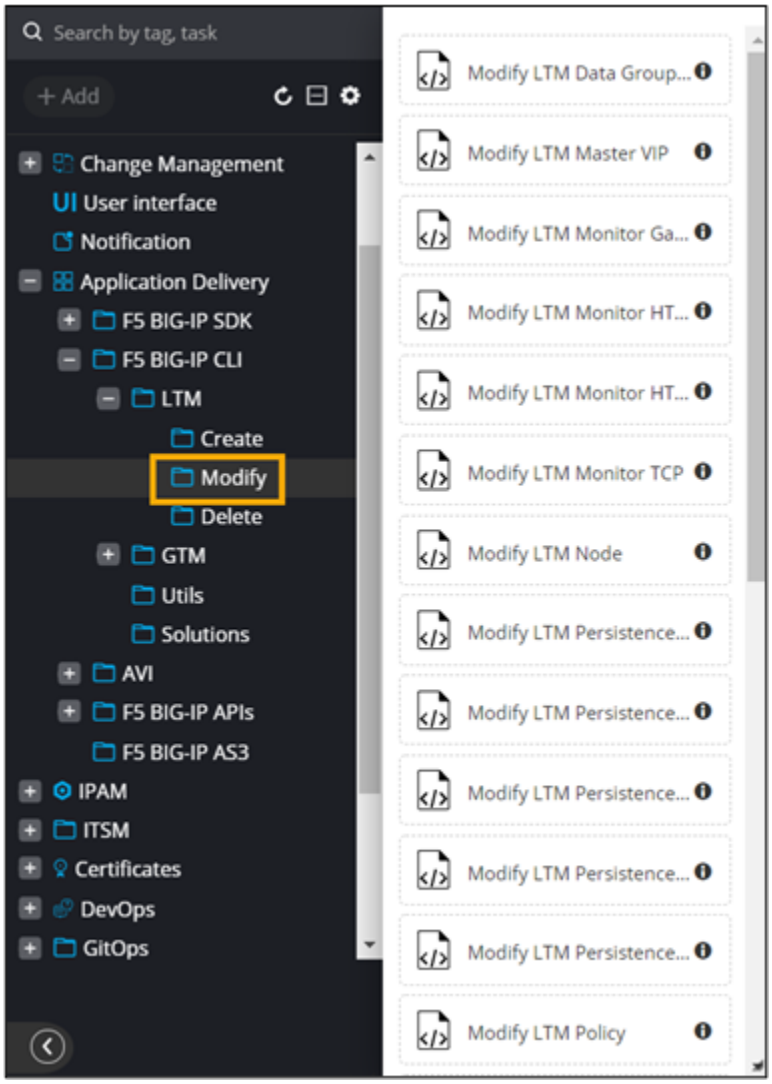
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



LTM - Modify

The **Modify** folder under **LTM** contains prebuilt tasks that can be leveraged to modify LTM objects on F5 BIG-IP.

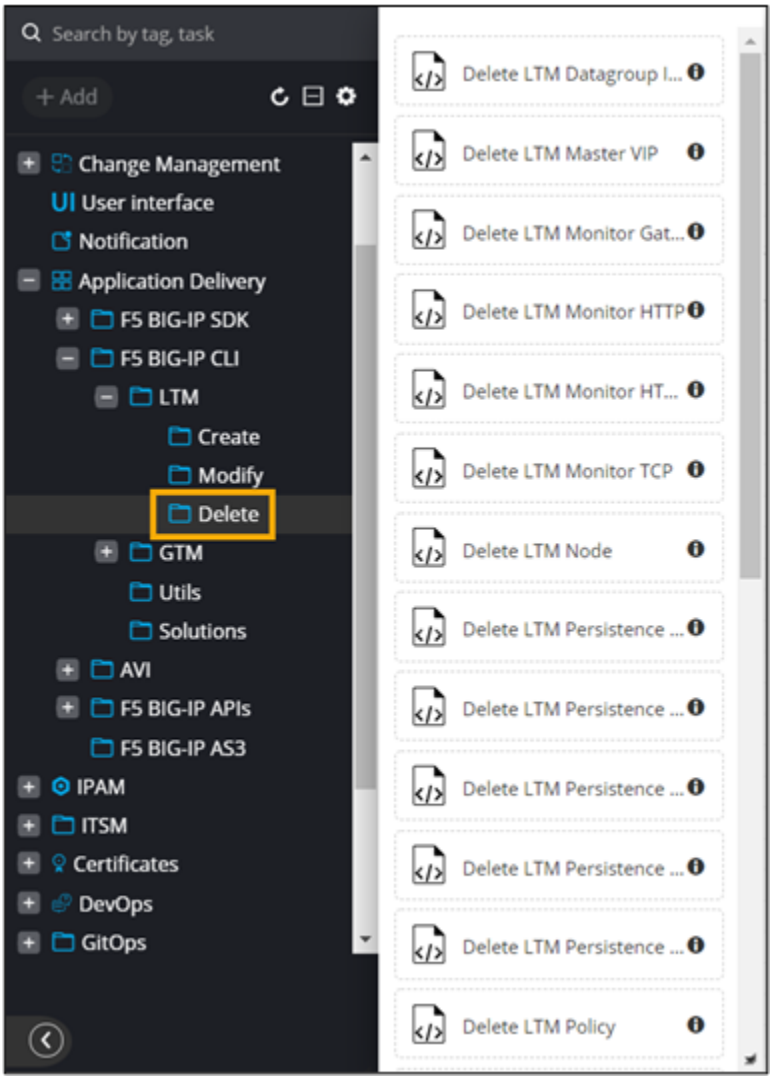
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



LTM - Delete

The **Delete** folder under **LTM** contains prebuilt tasks that can be leveraged to delete LTM objects on F5 BIG-IP.

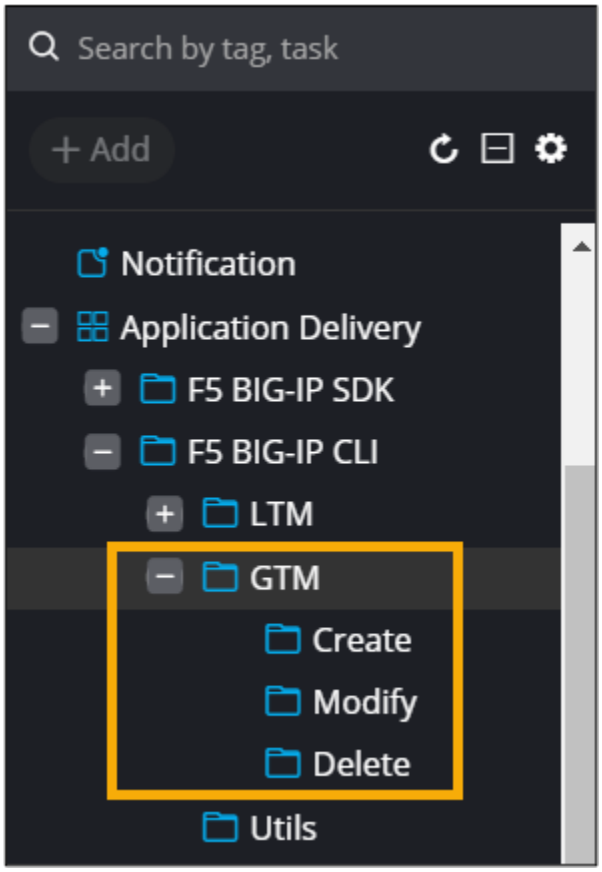
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



GTM

The workflows and tasks to create, modify, and delete GTM objects and devices on F5 BIG-IP are available under **Application Delivery** > **F5 BIG-IP CLI** > **GTM**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

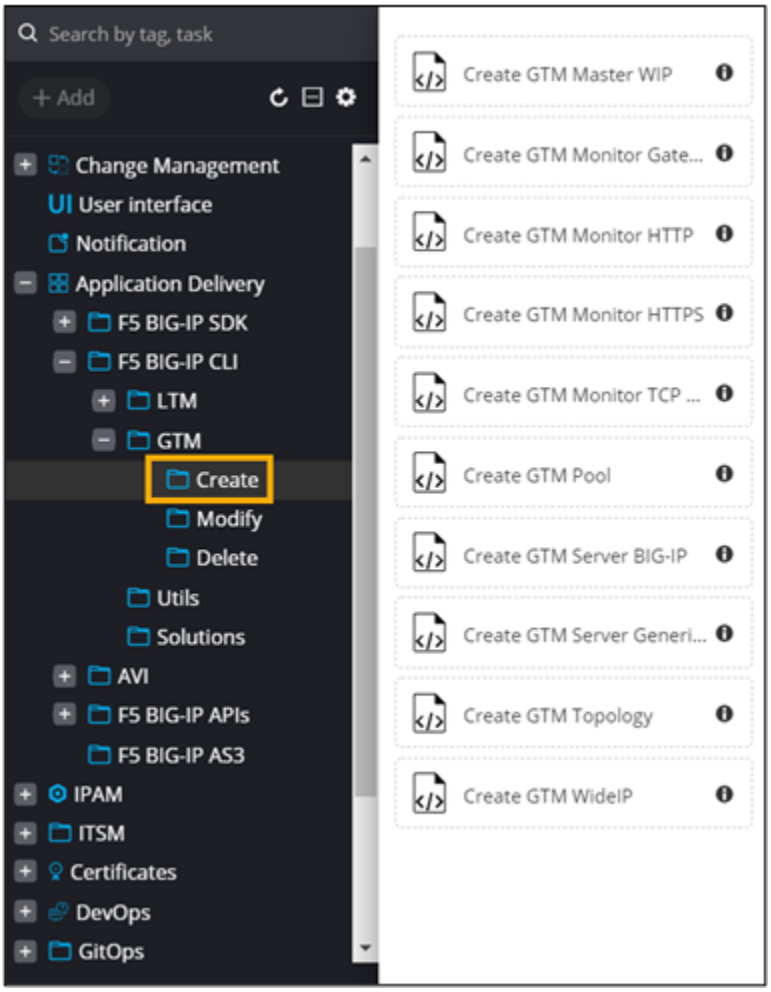


- GTM - Create
- GTM - Modify
- GTM - Delete

GTM - Create

The **Create** folder under **GTM** contains prebuilt tasks that can be leveraged to create GTM objects on F5 BIG-IP.

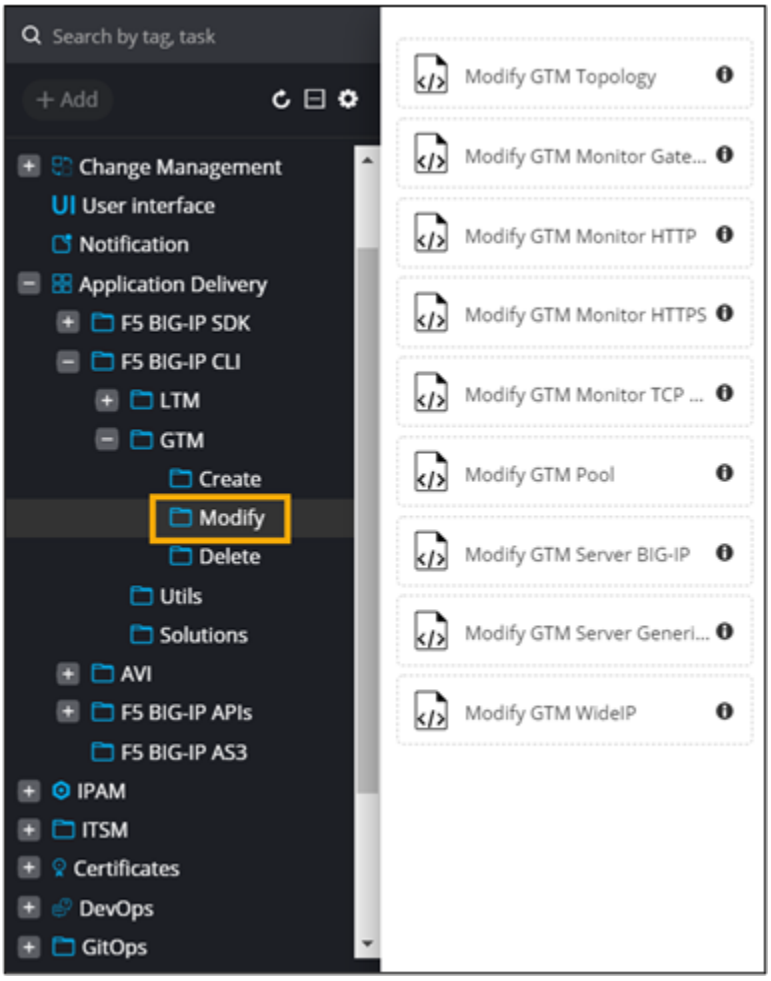
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



GTM - Modify

The **Modify** folder under **GTM** contains prebuilt tasks that can be leveraged to modify GTM objects on F5 BIG-IP.

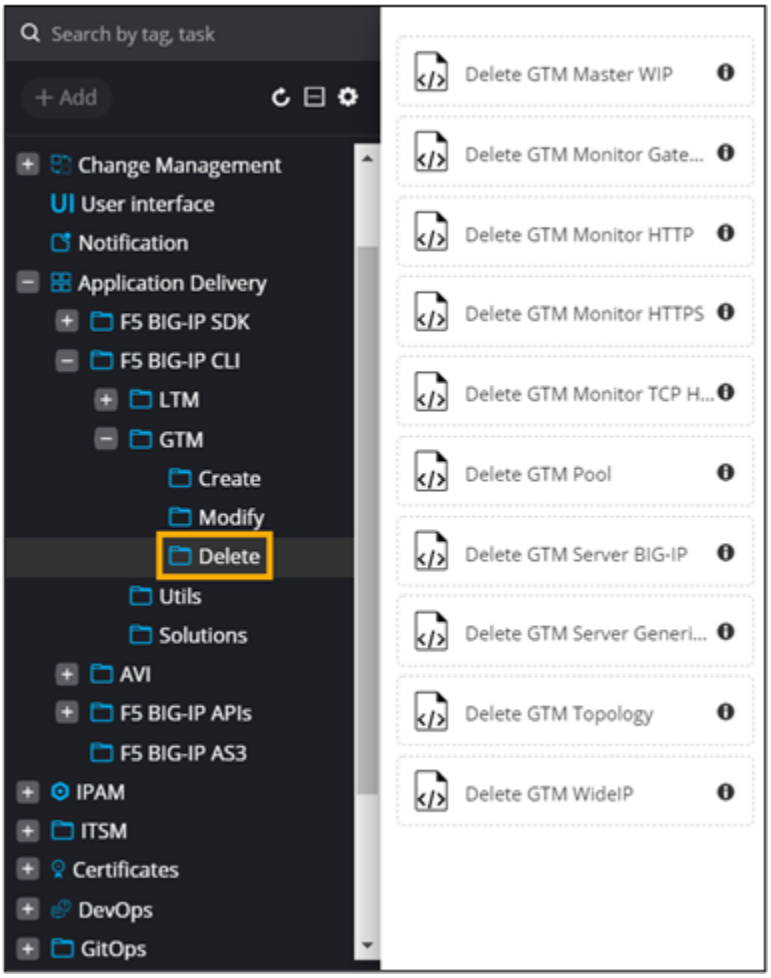
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



GTM - Delete

The **Delete** folder under **GTM** contains prebuilt tasks that can be leveraged to delete GTM objects on F5 BIG-IP.

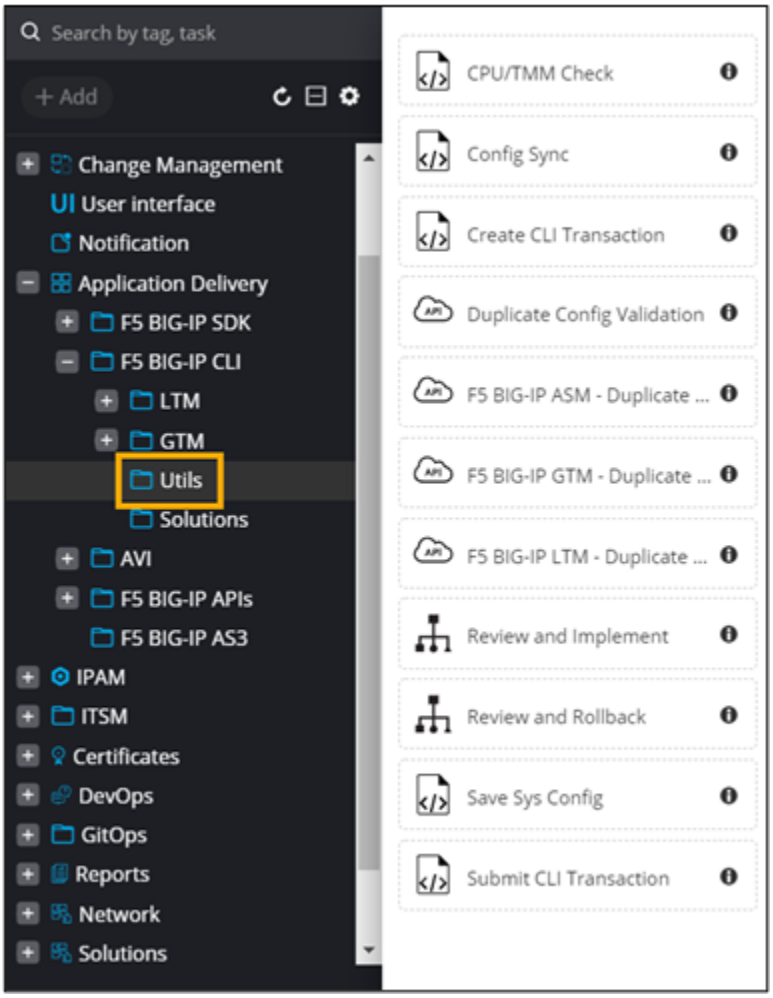
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Utils

This folder contains common tasks and workflows that are used across workflows for **F5 BIG-IP** provisioning.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

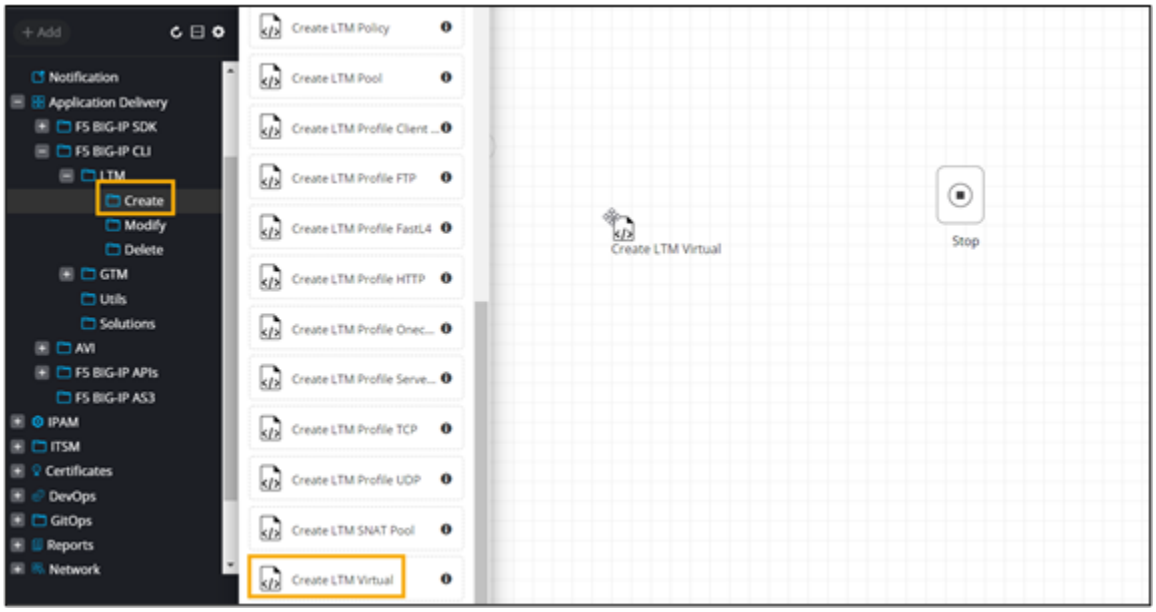


Creating a LTM Virtual Server on F5 BIG-IP

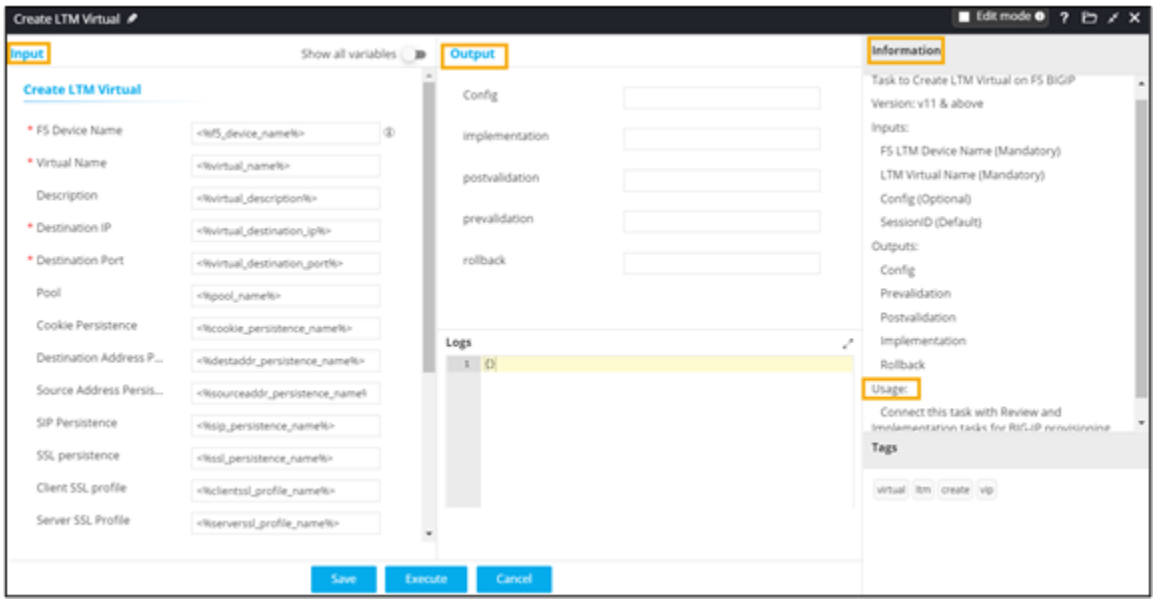
You can leverage the prebuilt LTM tasks to design a workflow for creating a LTM Virtual Server on F5 BIG-IP, add a CPU/TMM check, and also design a rollback workflow.

1. Design a new workflow.
2. From the menu on the left, select **Application Delivery** > **F5 BIG-IP CLI** > **LTM** > **Create**.
3. From the **Create** folder drag and drop the **Create LTM Virtual** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

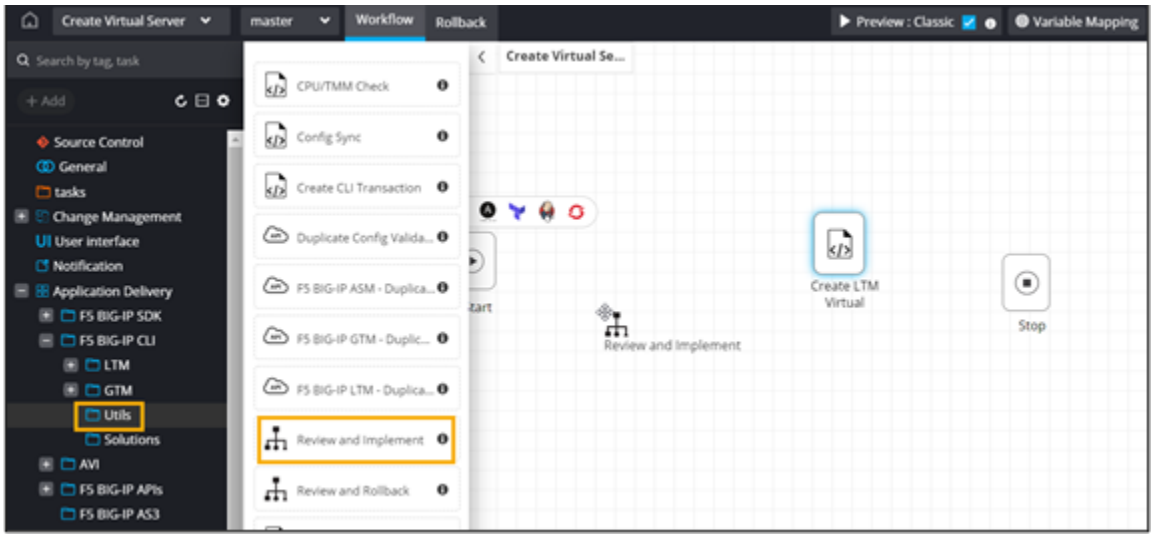


The **Create LTM Virtual** task window shows the defined **Input** and **Output** variables. The **Information** section shows what the task is for and also informs the user to connect this task with Review and Implementation tasks for BIG-IP provisioning.

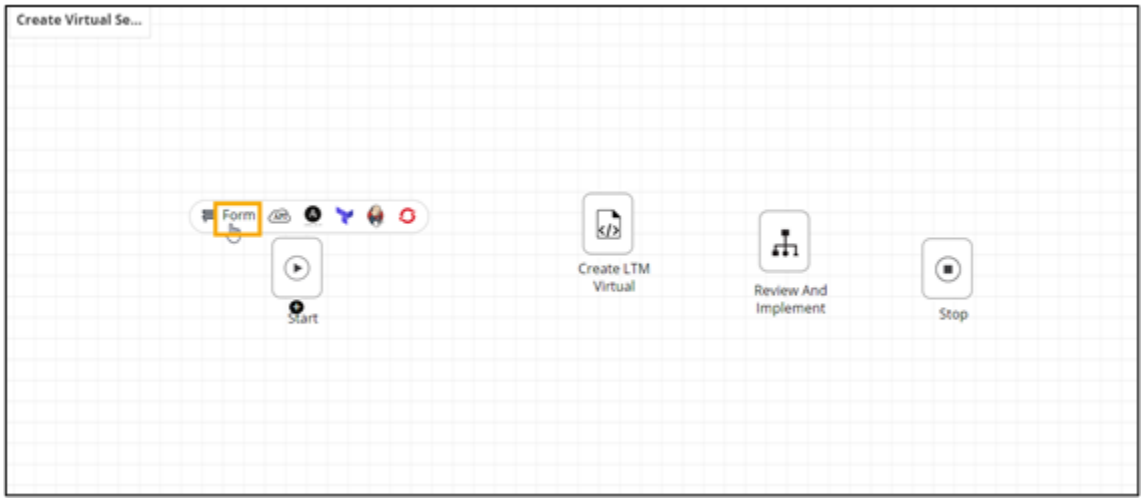


4. Click **Save**.
5. From the **Utils** folder, under **F5 BIG-IP CLI**, drag and drop the **Review and Implement** workflow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



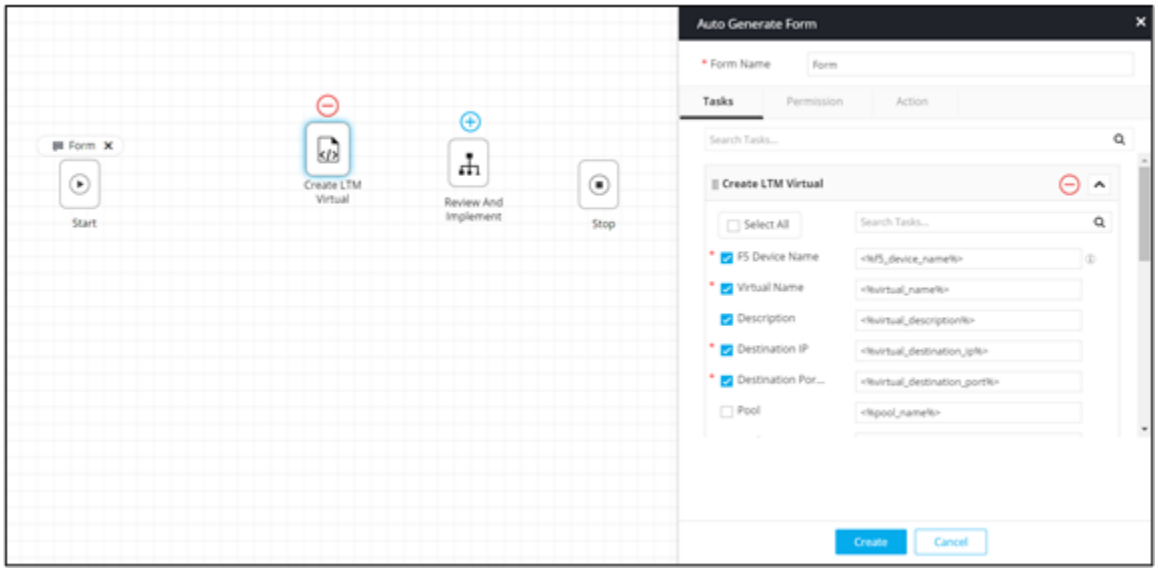
6. To auto-generate a form for this workflow, click **Form** above the **Start** task.



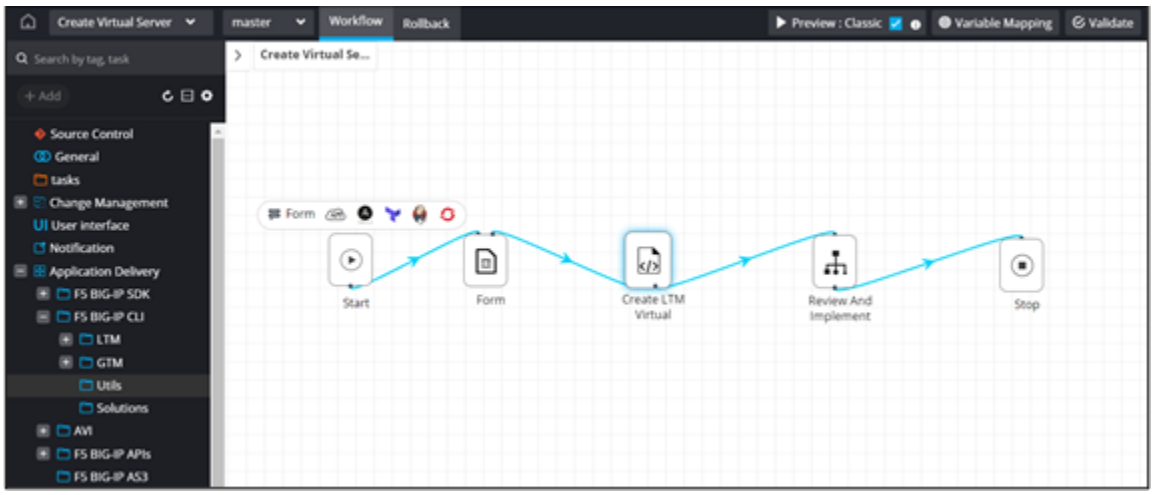
7. Click ⊕ above the **Create LTM Virtual** task and select the following form fields:

- F5 Device Name
- Virtual Name
- Description
- Destination IP
- Destination Port

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

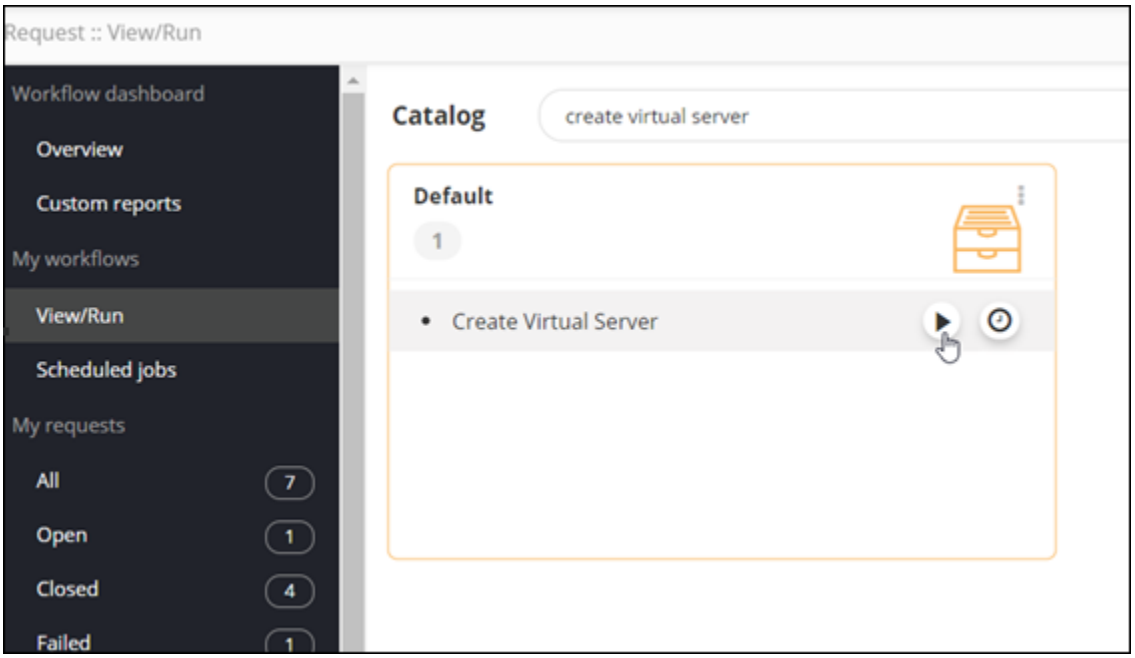


8. Click **Create**.
9. Connect and enable the workflow.



10. From the upper left corner of the screen, click ☰.
11. From the menu displayed, select **Request**.
12. On the **Request :: Overview** page, from the navigation pane on the left, click **View/Run**.
13. On the **Request :: View/Run** page, search for the workflow by typing the workflow name in the search bar.
14. To trigger the workflow, click ▶.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



15. Enter the details in the input form.

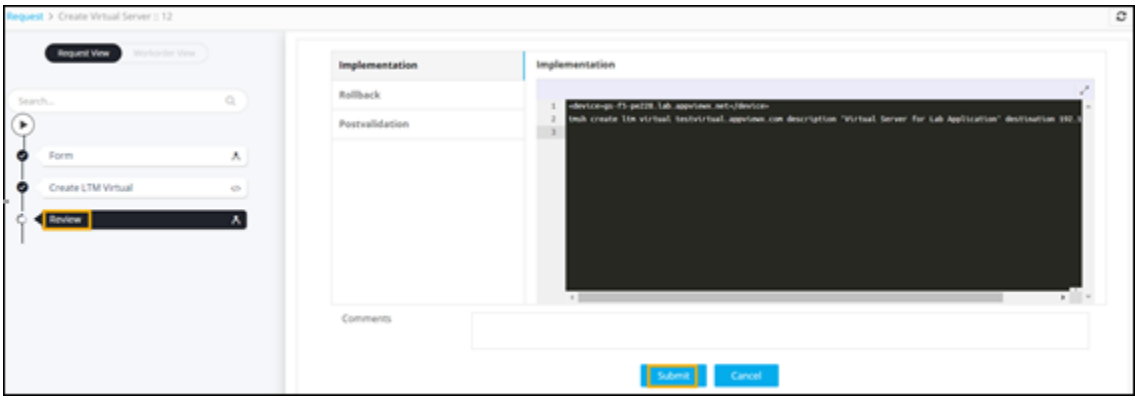
Note: Ensure that the device is available in the Device Inventory.

16. Click **Next**.
17. In the **Confirmation** pop-up window, click **Ok**.

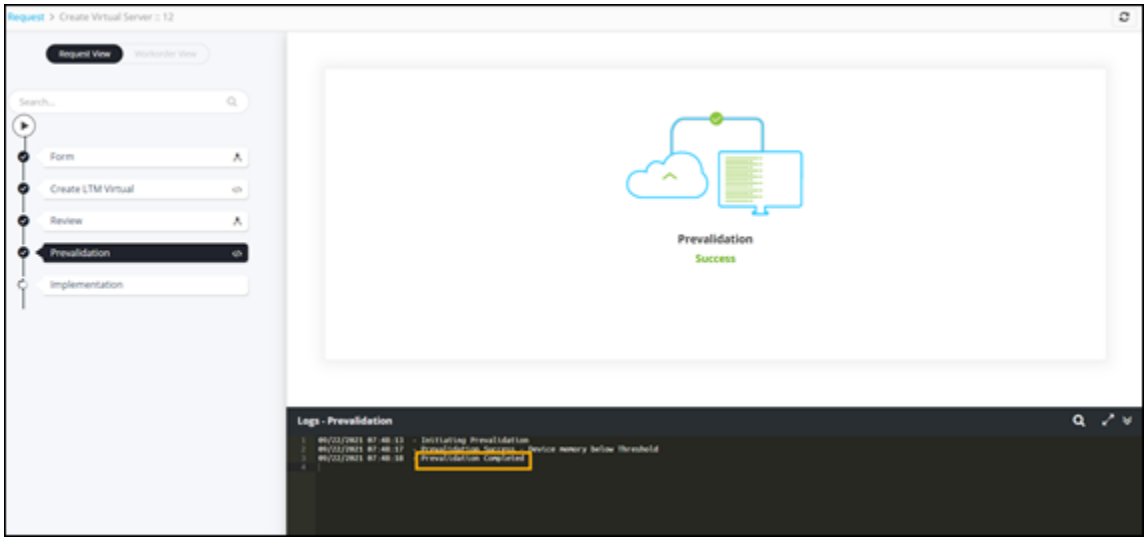
    Create LTM Virtual task successfully completed.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

18. At the **Review** stage, click **Submit**.

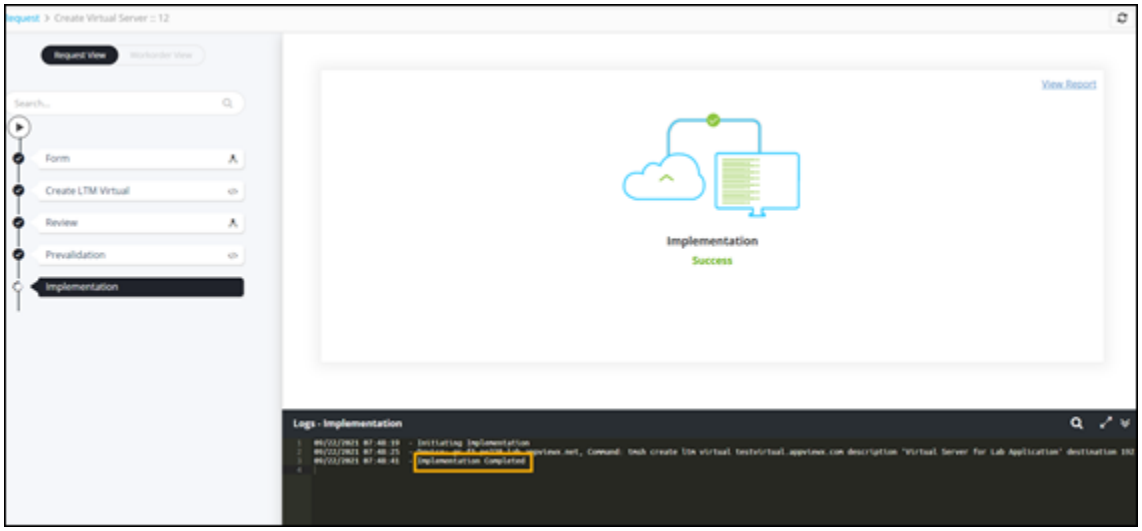


19. In the **Confirmation** pop-up window, click **Ok**.
    - Prevalidation task is completed.

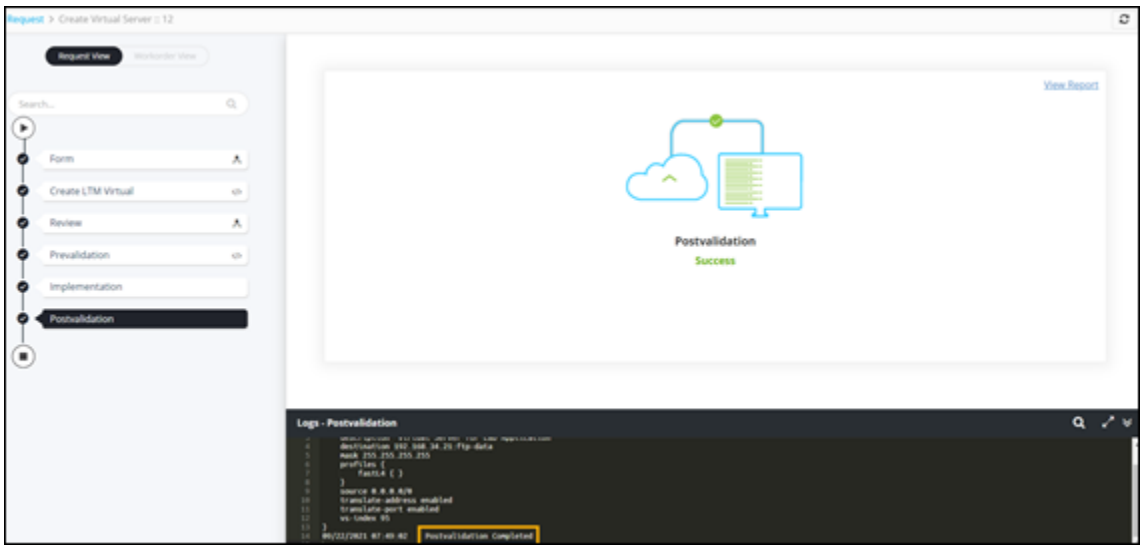


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Implementation task is completed.



- Postvalidation task is completed.

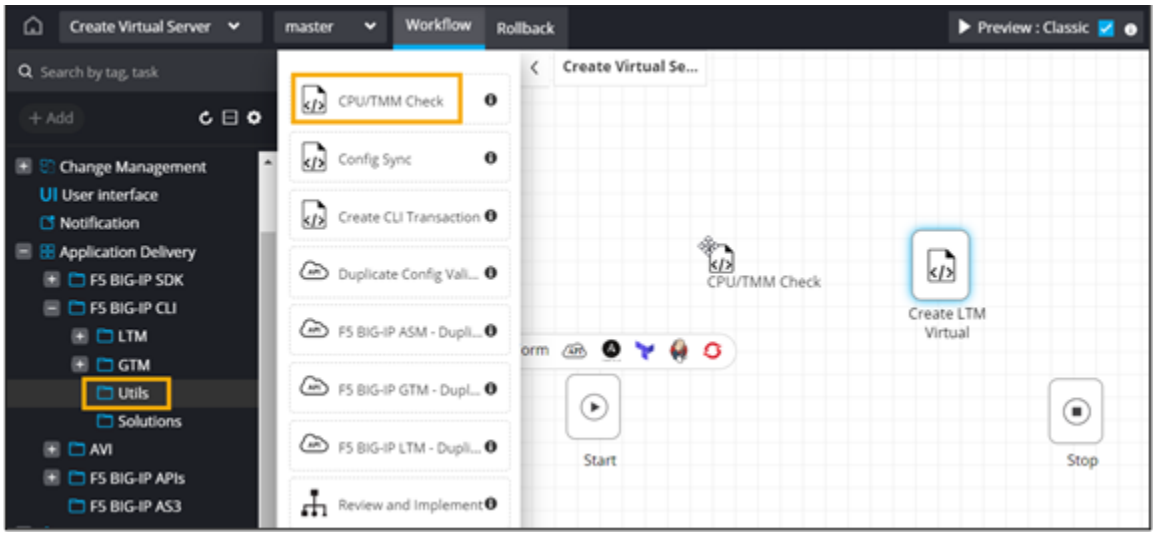


- Performing a CPU/TMM Check
- Adding a Rollback Workflow

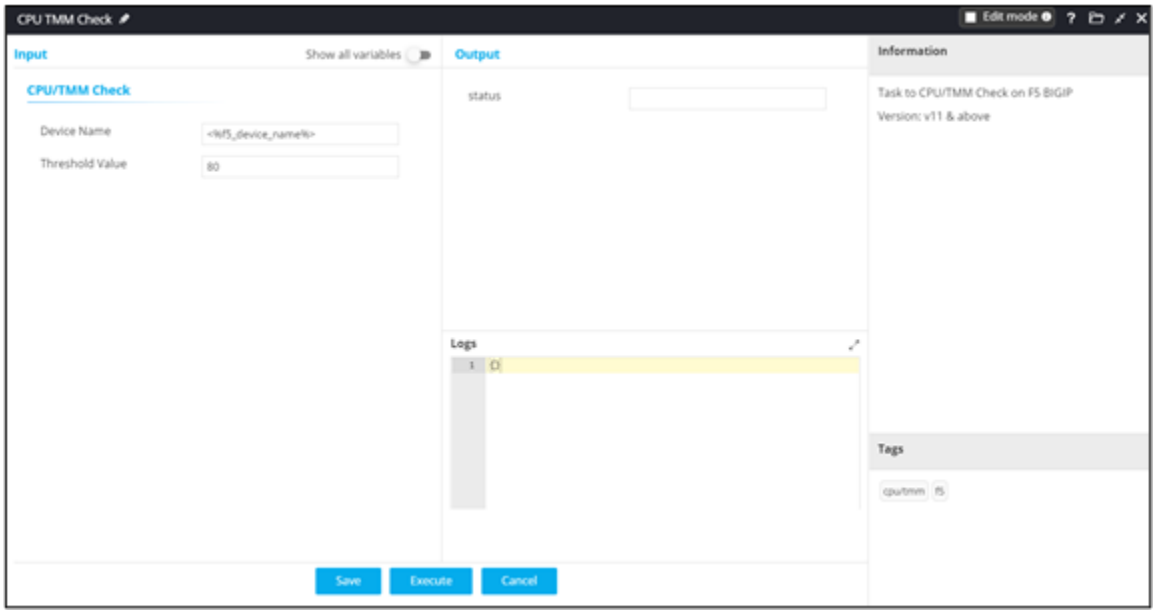
Performing a CPU/TMM Check

1. Design a workflow to create a LTM virtual server on F5 BIG-IP.
2. From the **Utils** folder, under **F5 BIG-IP CLI**, drag and drop the **CPU/TMM Check** task. This task performs a CPU/TMM check on F5 BIG-IP.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



The **CPU TMM Check** task opens in the Edit mode.

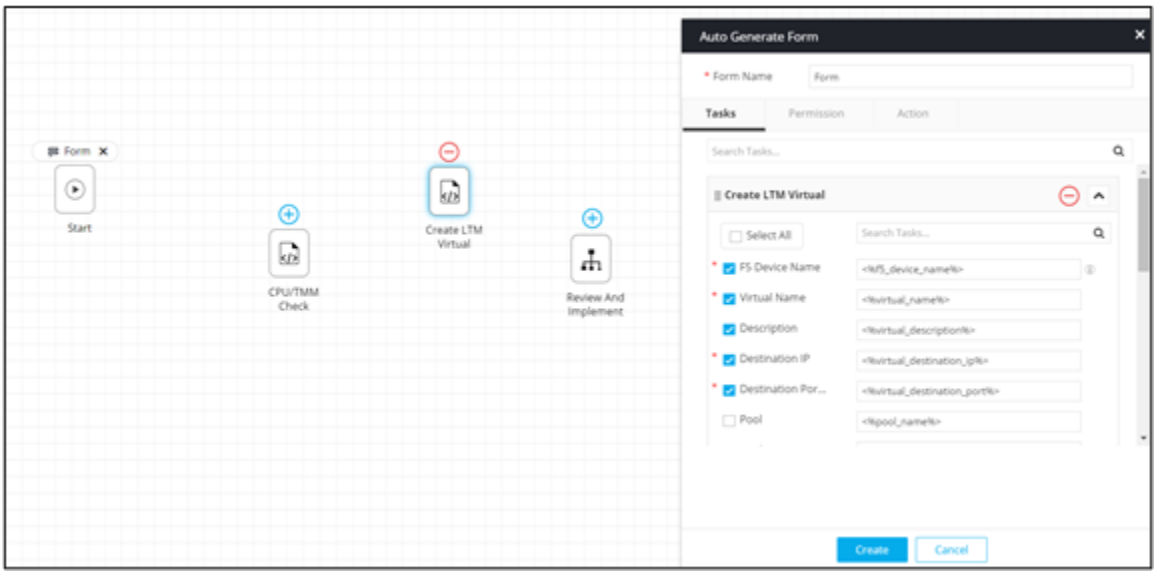


3. Click **Save**.
4. To auto-generate a form for this workflow, click **Form** above the **Start** task.
5. Click ⊕ above the **Create LTM Virtual** task and select the following tasks:

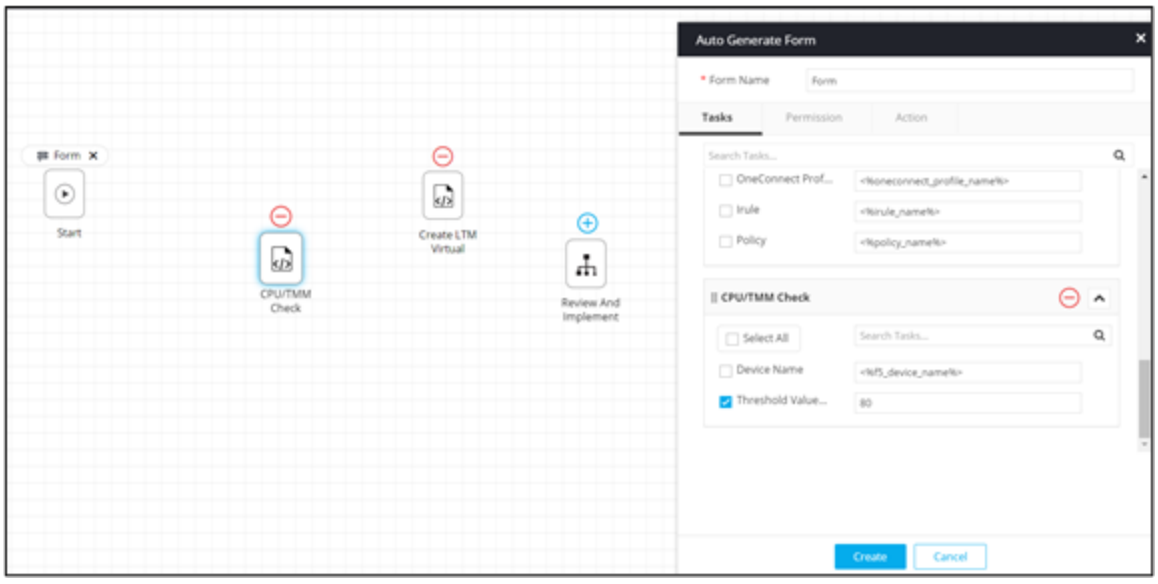
- F5 Device Name
- Virtual Name
- Description

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Destination IP
- Destination Port

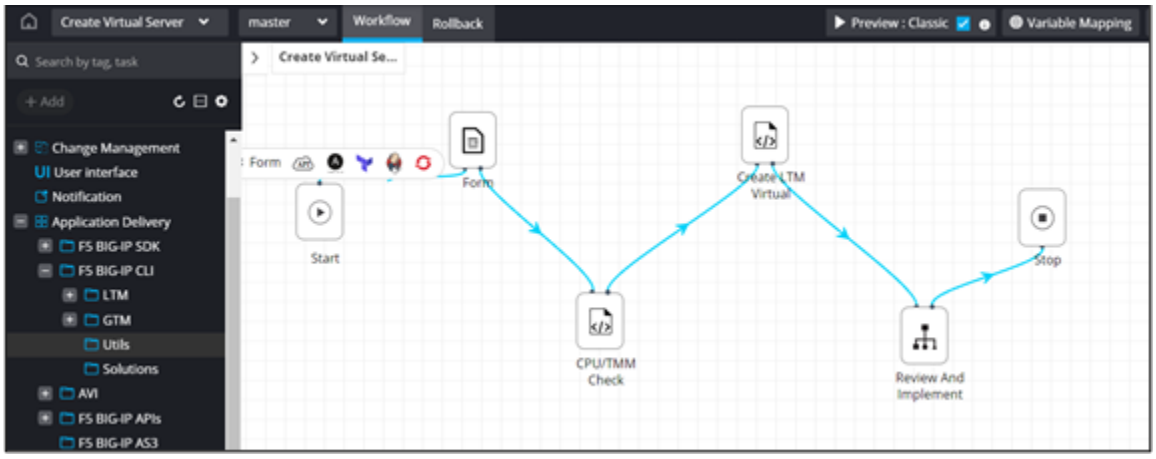


6. Click ⊕ above the **CPU/TMM Check** task and select the **Threshold Value** field for the form.

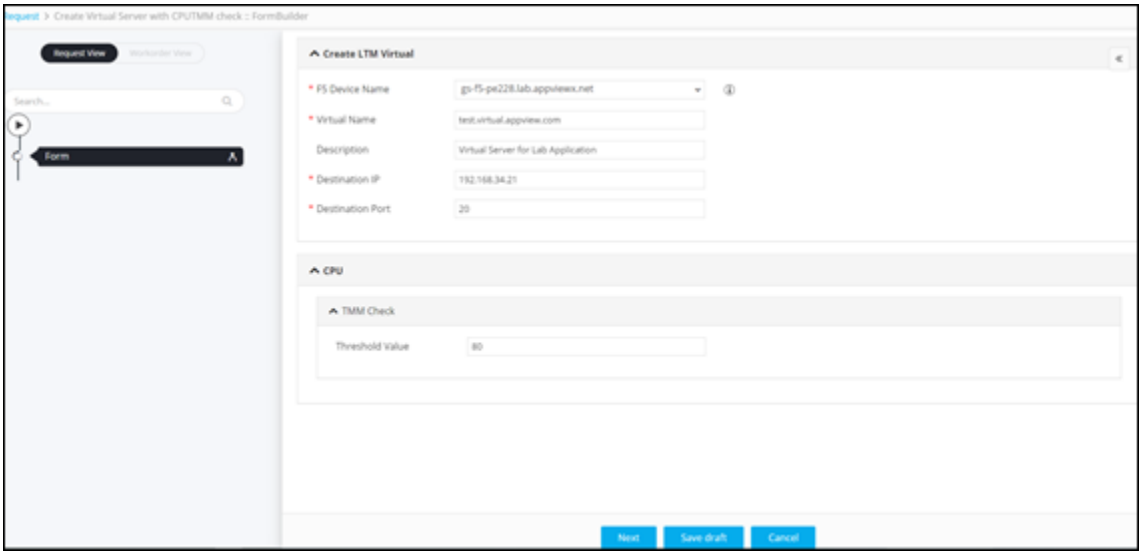


7. Click **Create**.
8. Connect and enable the workflow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



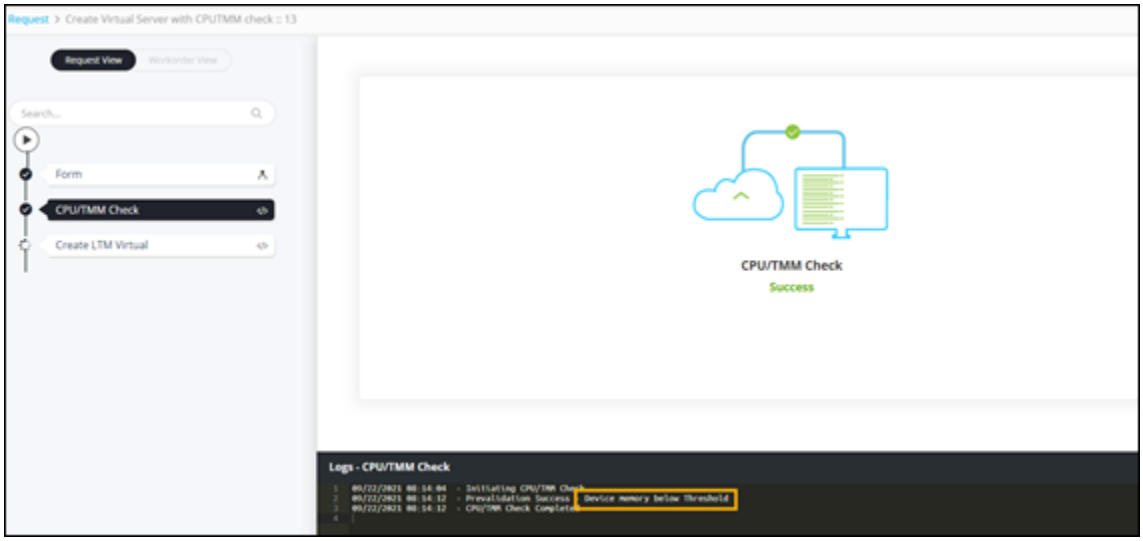
9. From the top left corner of the screen, click ☰.
10. From the menu displayed, click **Request**.
11. On the **Request :: Overview** page, from the navigation pane on the left, click **View/Run**.
12. To trigger the workflow, on the **Request :: View/Run** page, search for the **Form to Form** workflow and click ▶.
13. Enter the details in the input form.



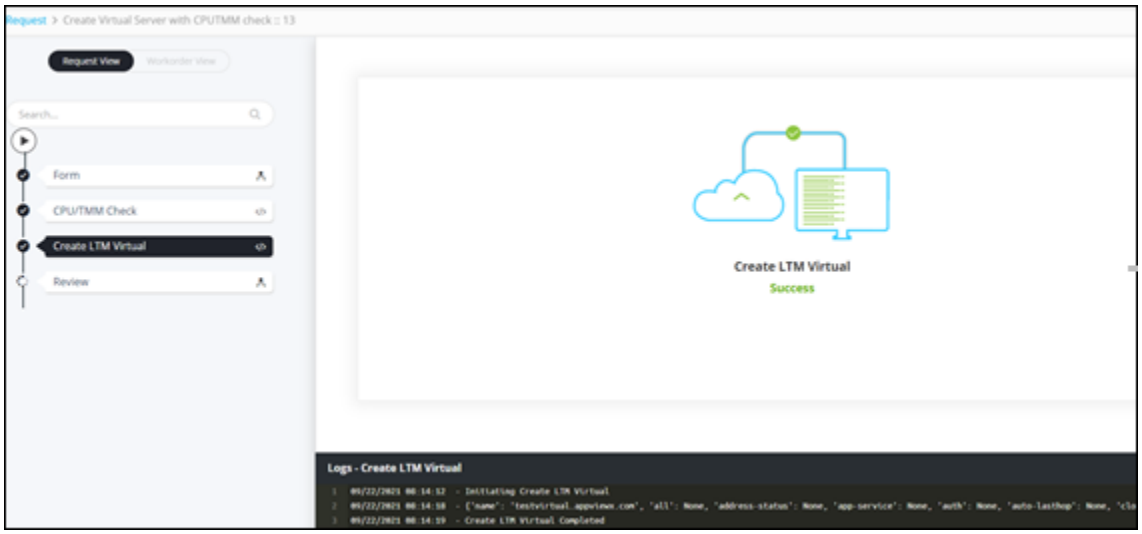
14. Click **Next**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- **CPU/TMM Check** is successful - Device memory is below threshold.



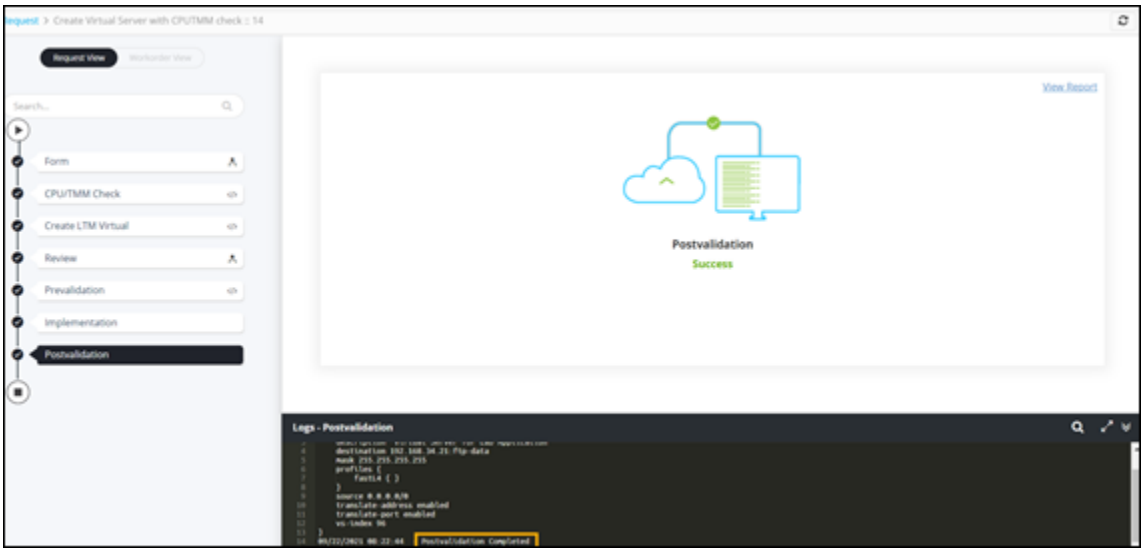
- **Create LTM Virtual** task successfully completed.



15. At the **Review** stage, click **Submit**.

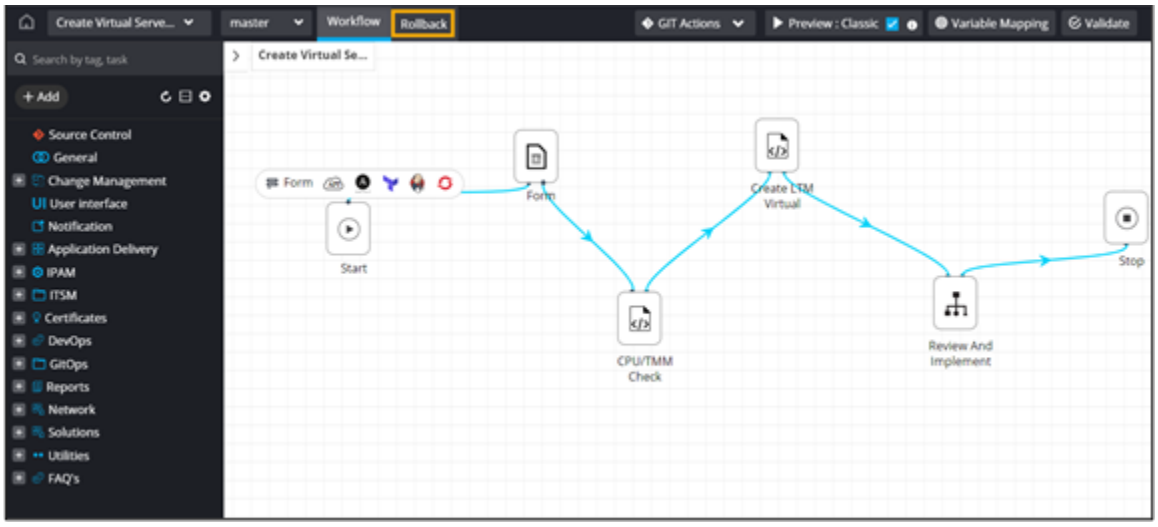
    Prevalidation, Implementation, Postvalidation tasks completed successfully.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



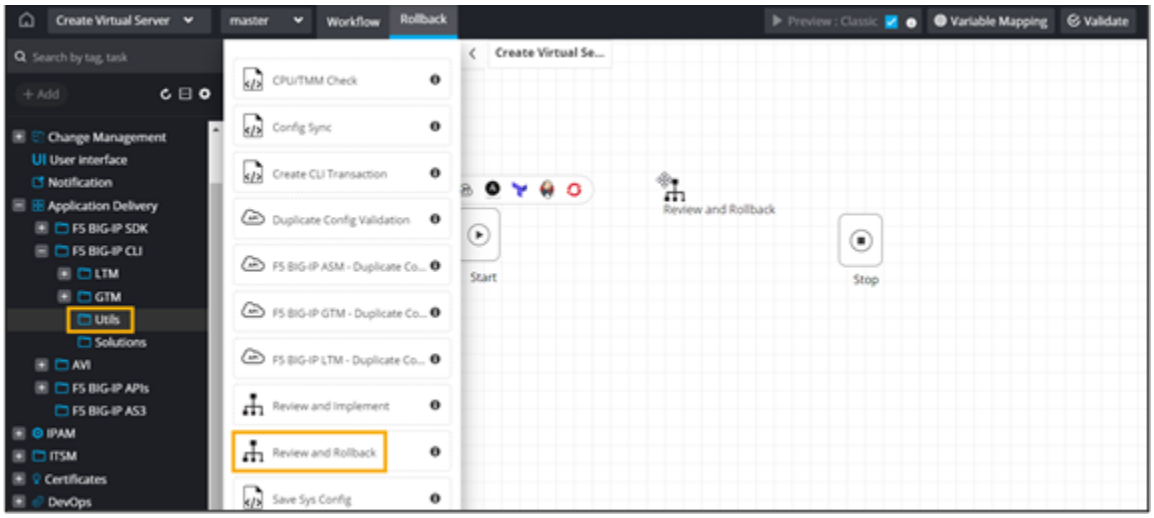
Adding a Rollback Workflow

1. Design a workflow to create a LTM virtual server on F5 BIG-IP with CPU/TMM check.
2. Click **Rollback**.

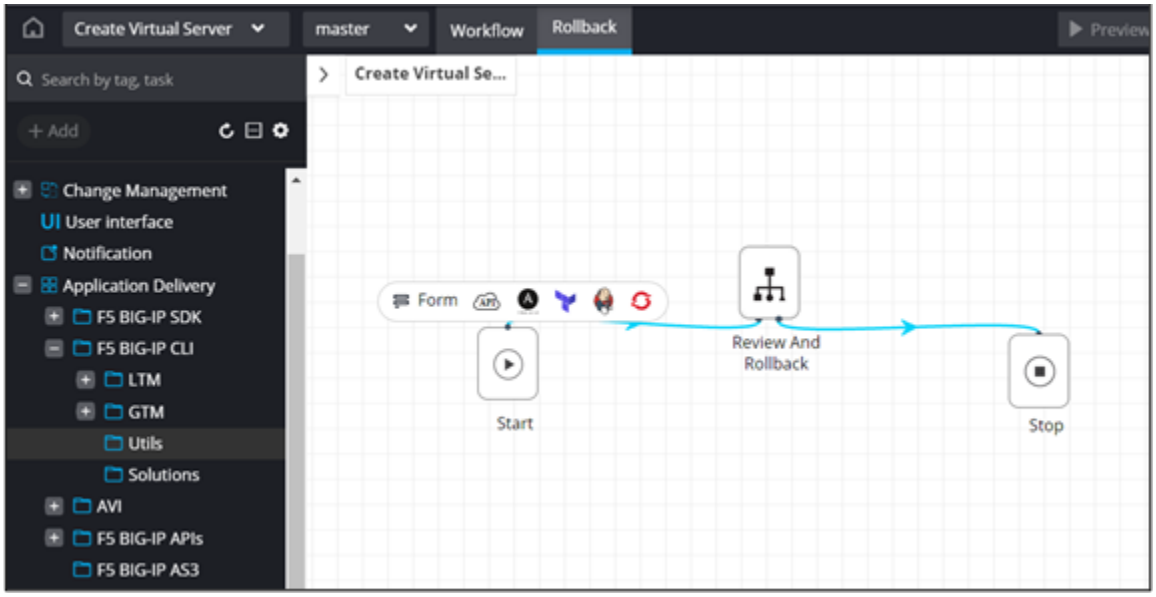


3. From the **Utils** folder, drag and drop the **Review and Rollback** workflow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

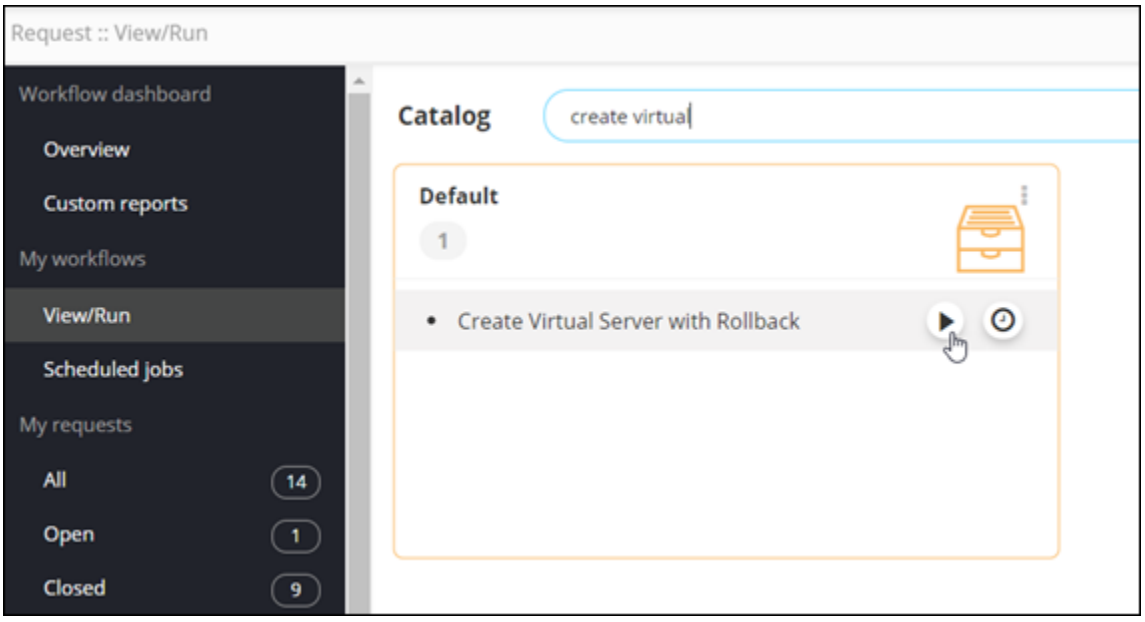


4. Connect and enable the workflow.

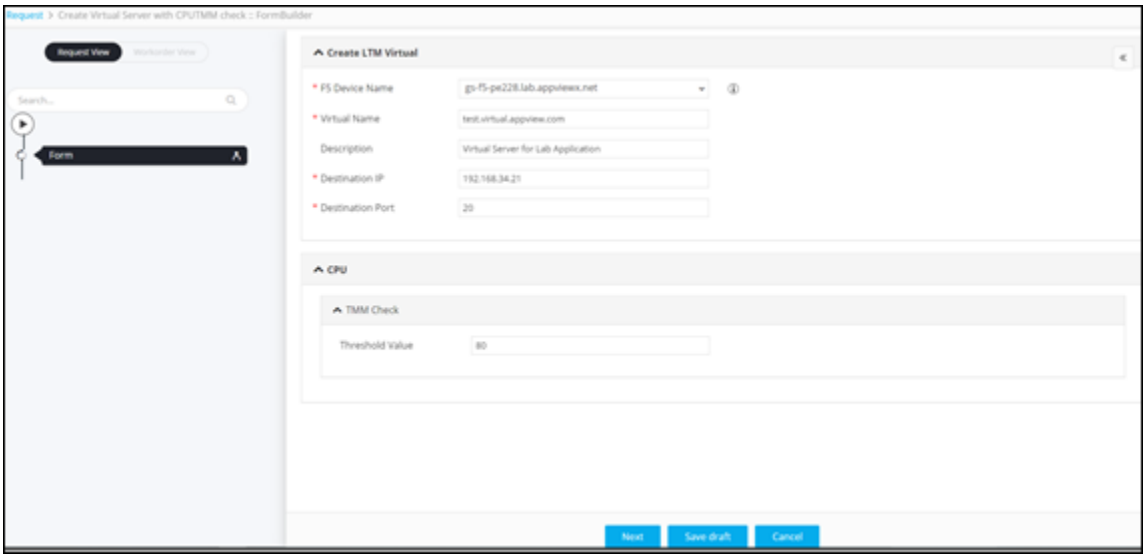


5. Trigger the workflow from the Request :: View/Run page.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

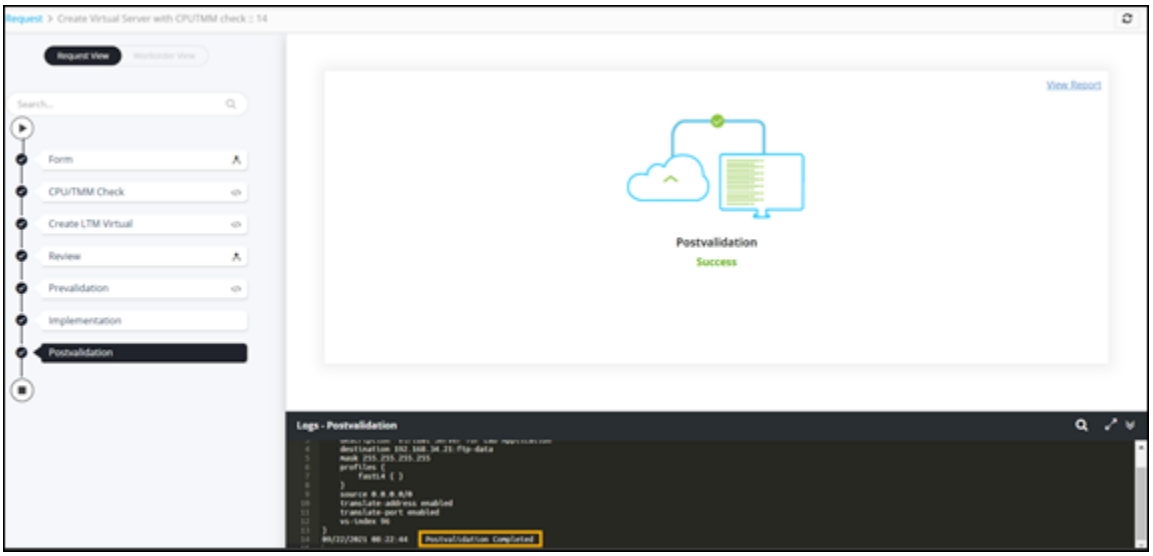


6. Enter the details in the input form.

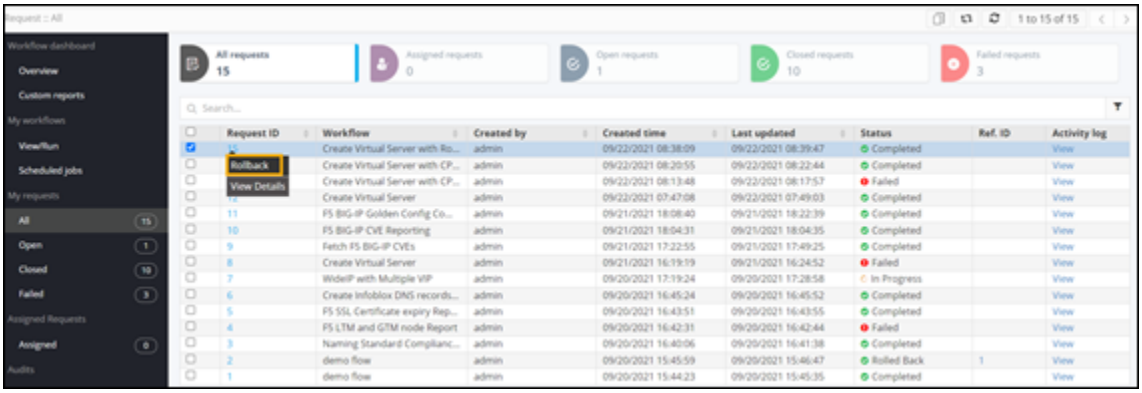


7. At the **Review** stage, click **Submit**. Prevalidation, Implementation, and Postvalidation tasks completed successfully.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

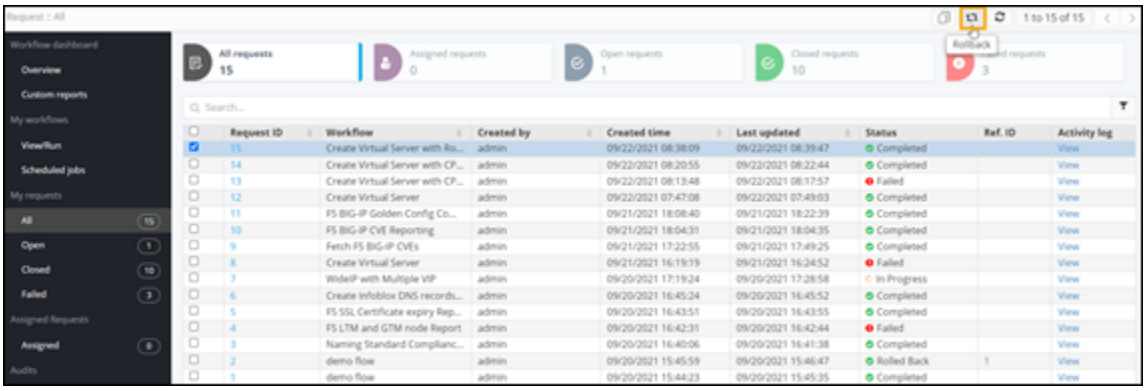


8. To trigger the rollback workflow, on the Request :: Overview page, under **My Requests**, click **All**.
9. On the **Request :: All** page, right-click on the **Request ID** of the workflow.
10. From the options displayed, select **Rollback**.

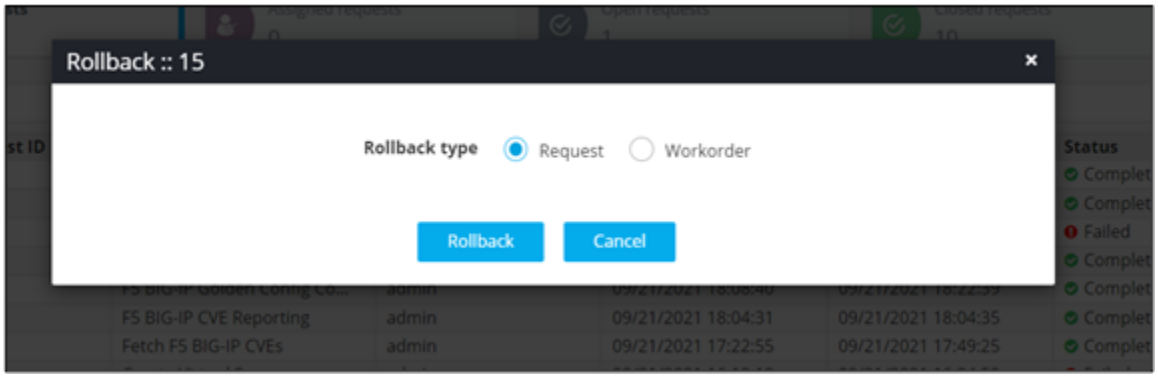


Tip: You can also select the workflow and click from the command bar in the top right corner of the screen.

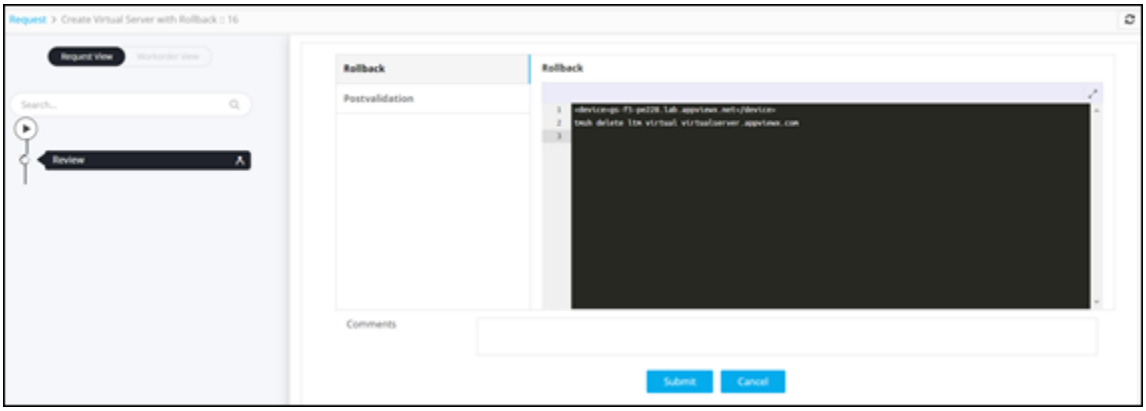
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



11. In the **Rollback** pop-up window, click **Yes**.
12. Select **Rollback type** as **Request**.

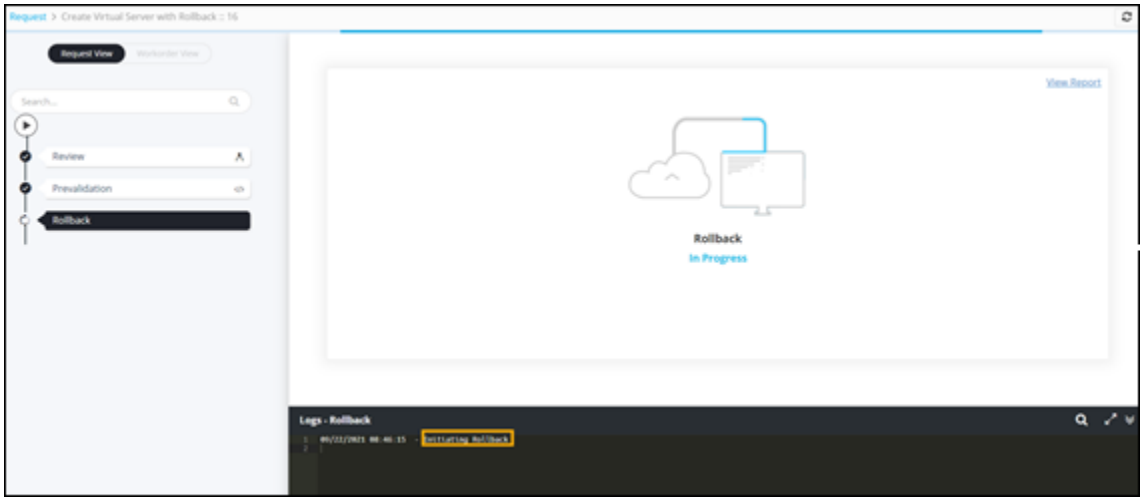


13. Click **Rollback**.
14. At the **Review** stage, click **Submit**.

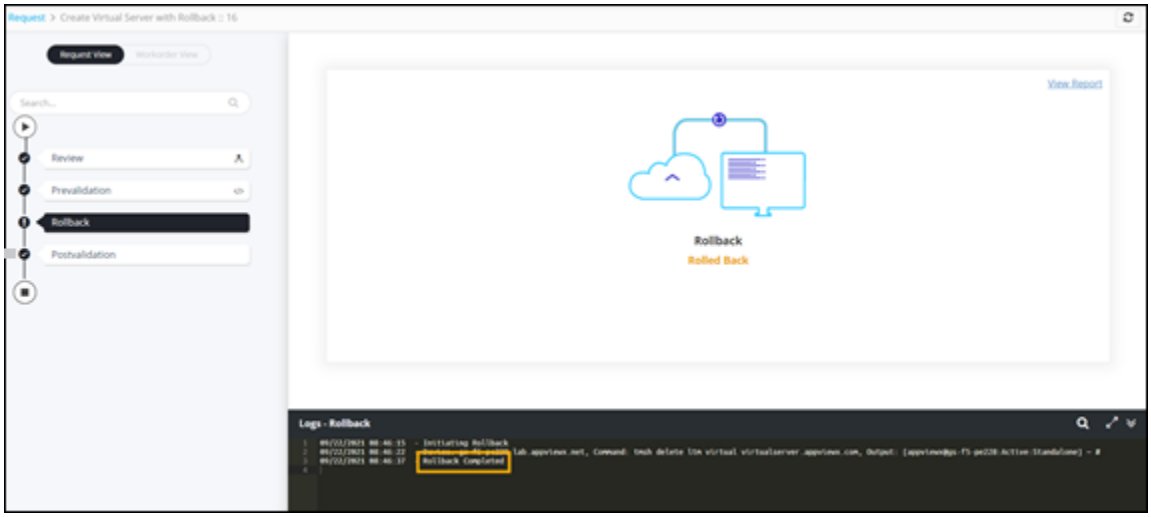


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Rollback initiated.



- Rollback completed.



Related information
Creating a LTM Virtual Server on F5 BIG-IP

AVI Solutions

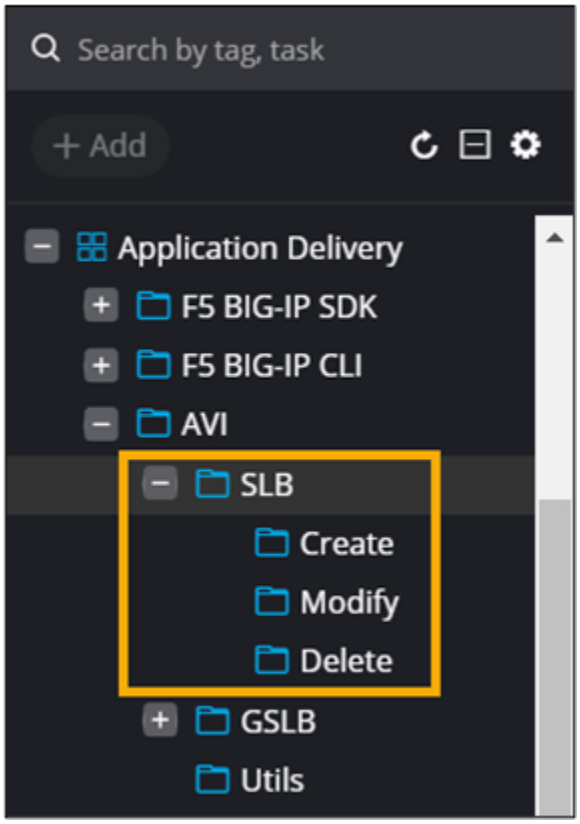
This section describes some of the workflows/tasks for AVI provisioning available in the Workflow Studio.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- SLB
- GSLB
- Utils

SLB

The workflows and tasks to create, modify, and delete SLB objects and devices on AVI are available under **Application Delivery** > **AVI** > **SLB**.

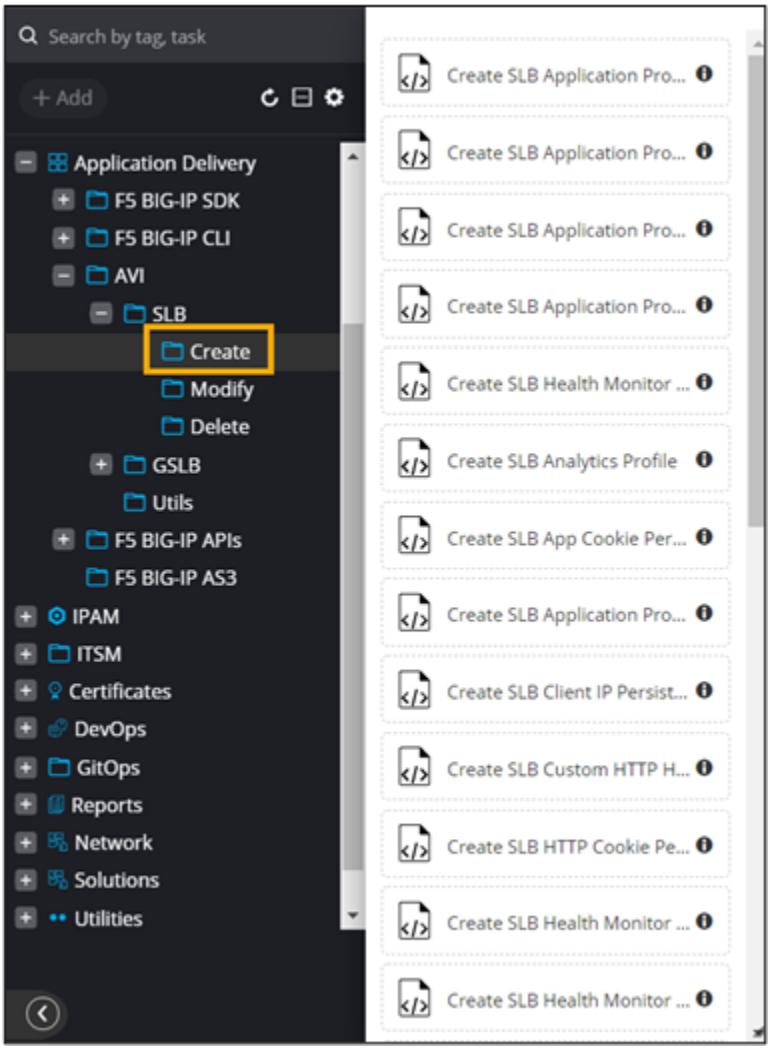


- SLB - Create
- SLB - Modify
- SLB - Delete

SLB - Create

The **Create** folder under **SLB** contains prebuilt tasks that can be leveraged to create SLB objects on AVI.

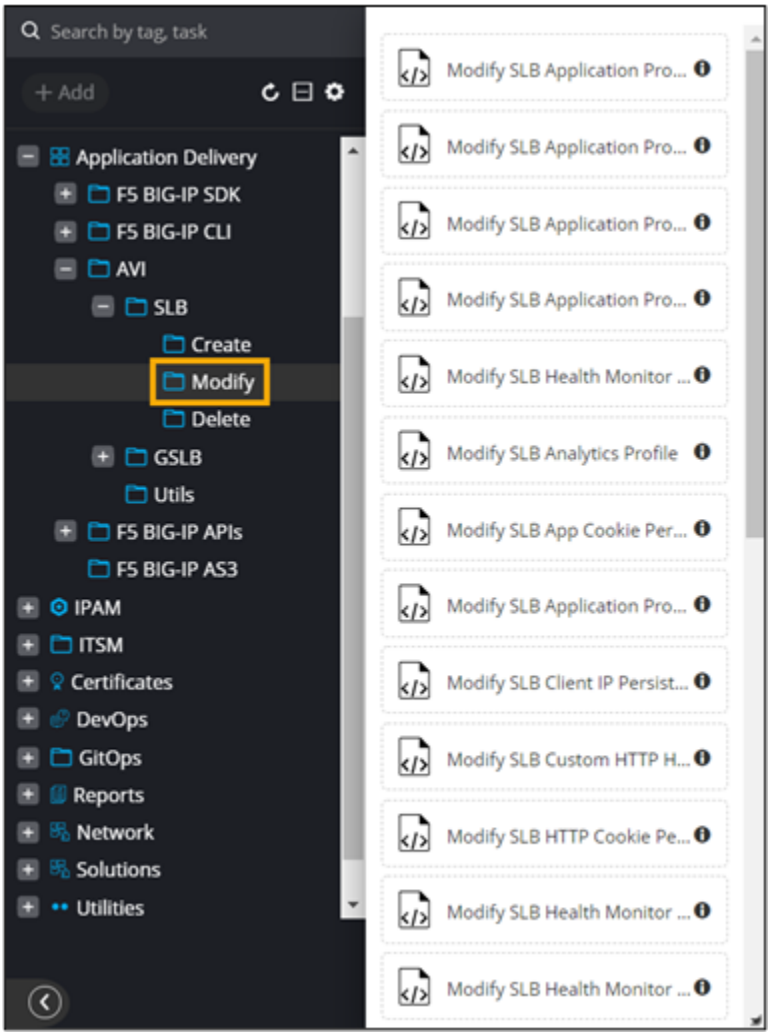
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



SLB - Modify

The **Modify** folder under **SLB** contains prebuilt tasks that can be leveraged to modify SLB objects on AVI.

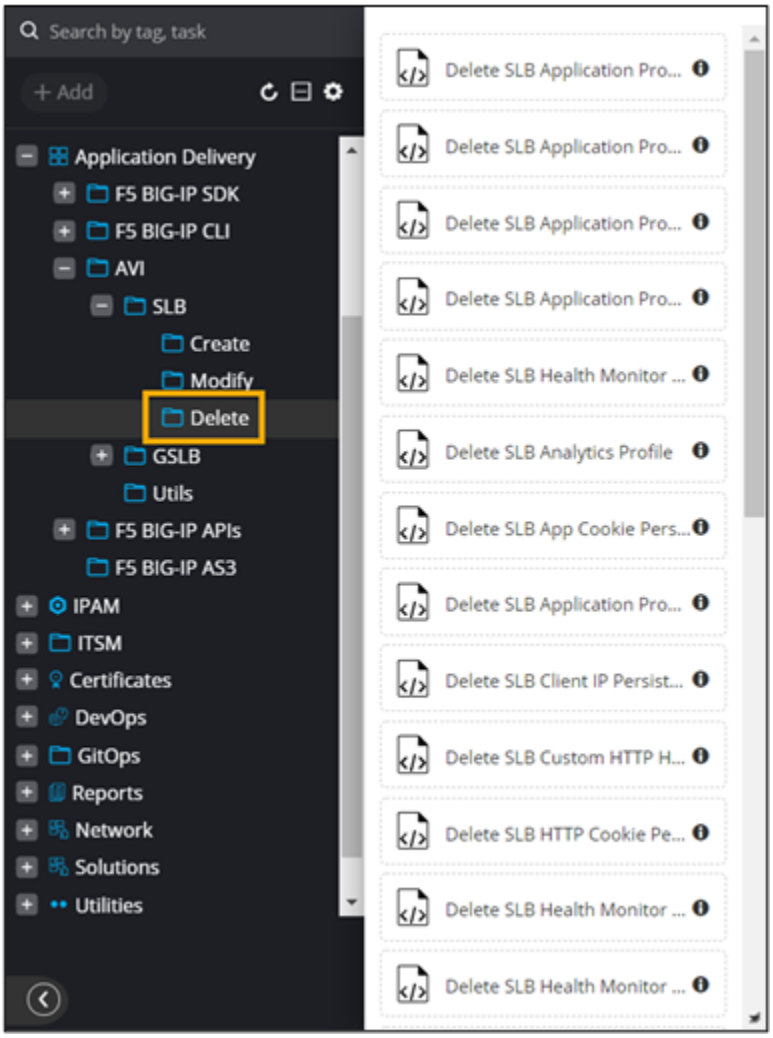
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



SLB - Delete

The **Delete** folder under **SLB** contains prebuilt tasks that can be leveraged to delete SLB objects on AVI.

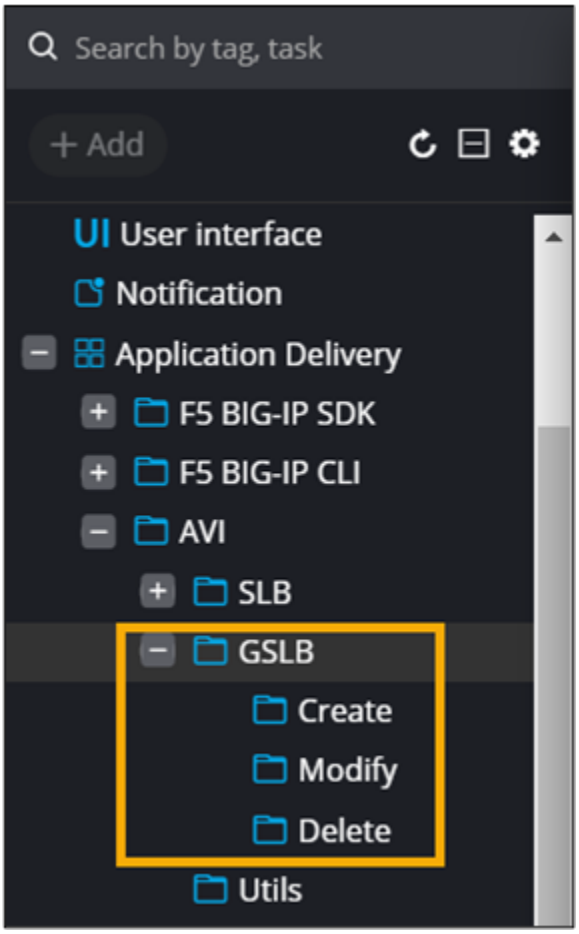
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



GSLB

The workflows and tasks to create, modify, and delete GSLB objects and devices on AVI are available under **Application Delivery** > **AVI** >**GSLB** GSLB.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

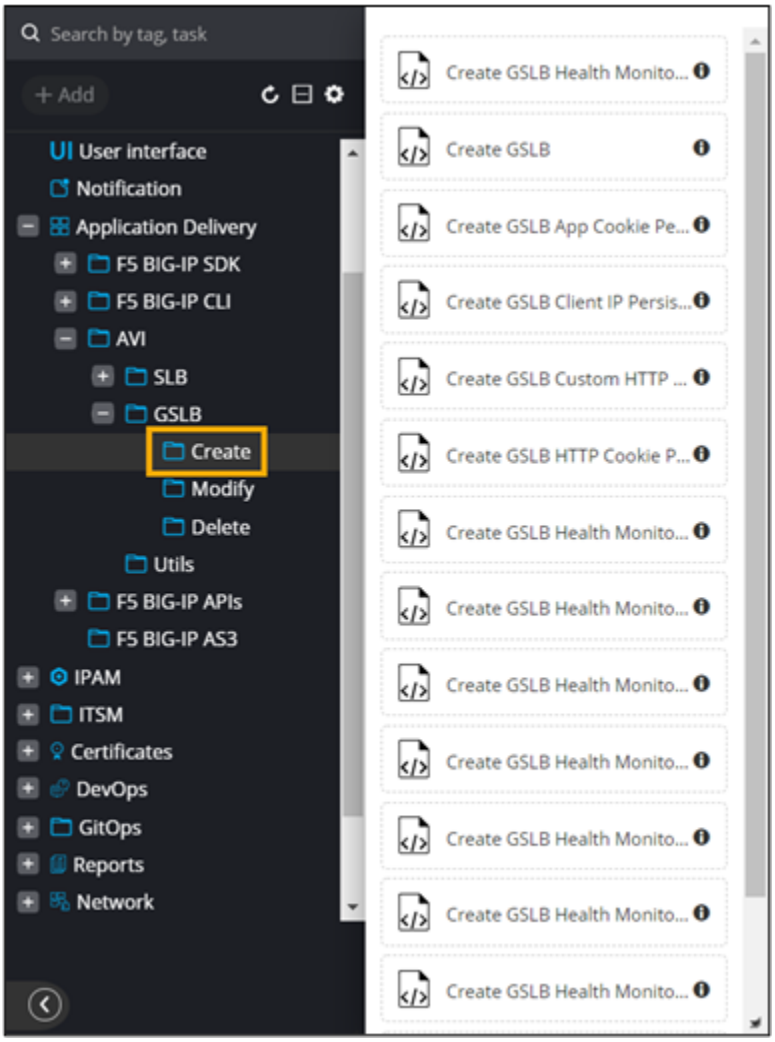


- GSLB - Create
- GSLB - Modify
- GSLB - Delete

GSLB - Create

The **Create** folder under **GSLB** contains prebuilt tasks that can be leveraged to create GSLB objects on AVI.

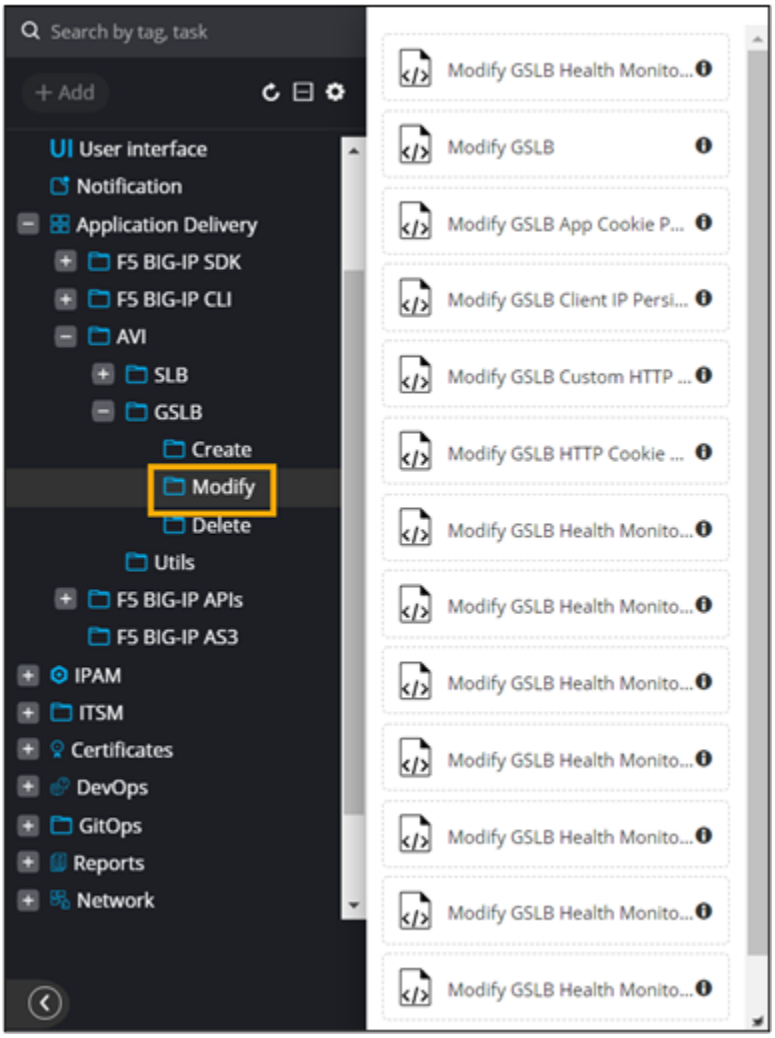
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



GSLB - Modify

The **Modify** folder under **GSLB** contains prebuilt tasks that can be leveraged to modify GSLB objects on AVI.

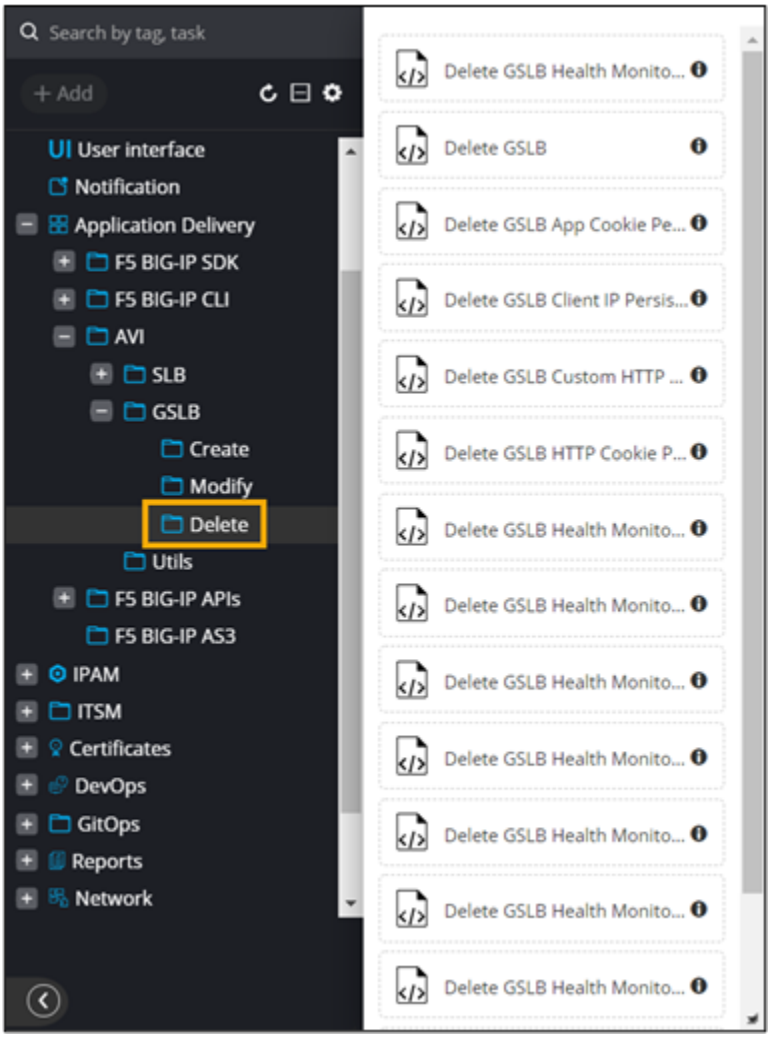
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



GSLB - Delete

The **Delete** folder under **GSLB** contains prebuilt tasks that can be leveraged to delete GSLB objects on AVI.

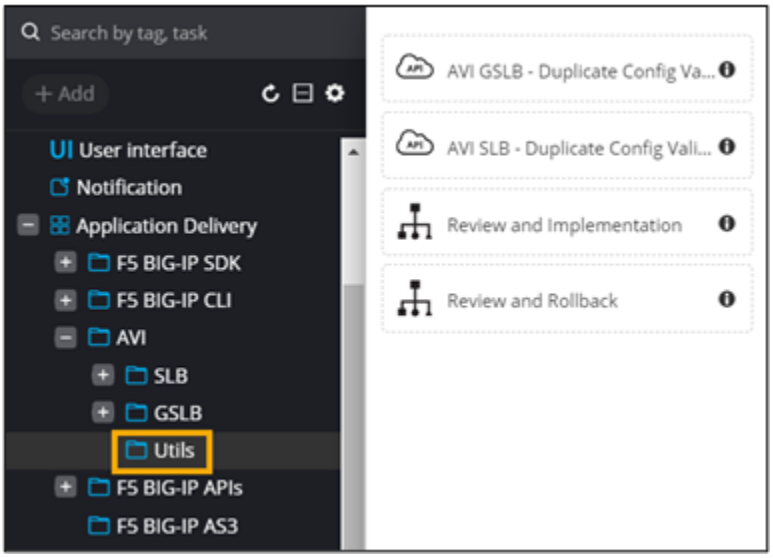
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Utils

This folder contains common tasks and workflows that are used across workflows for **AVI** provisioning.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Automate DNS Services

You can accelerate application delivery with prebuilt DNS automation tasks and solutions by integrating with IPAM tools such as Infoblox and Bluecat to fetch and reserve free IP addresses automatically when creating virtual servers.

- Infoblox Solutions
- Bluecat Solutions

Infoblox Solutions

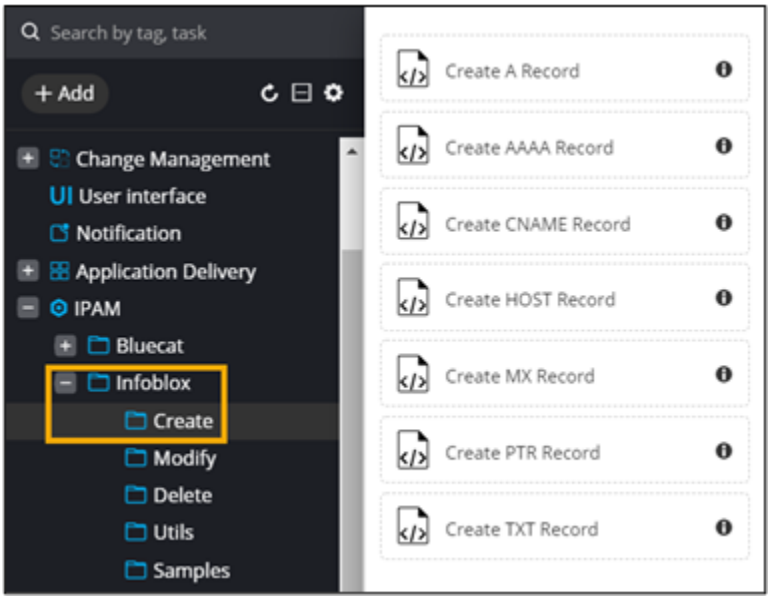
This section describes the tasks/workflows that can be leveraged to create, modify, and delete DNS records on Infoblox.

- Create
- Modify
- Delete
- Utils

Create

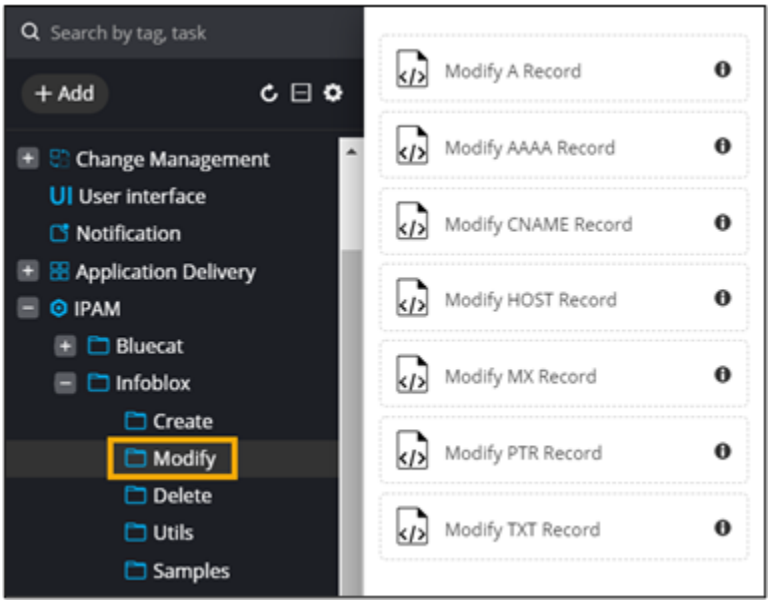
The **Create** folder under **Infoblox** contains prebuilt tasks that can be leveraged to create DNS records such as A Record, CNAME record, and PTR Record on Infoblox.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Modify

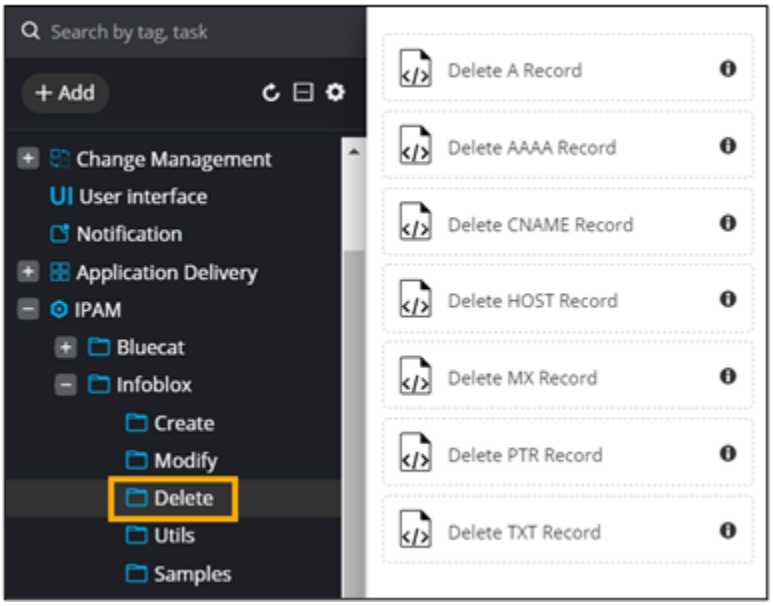
The **Modify** folder under **Infoblox** contains prebuilt tasks that can be leveraged to modify DNS records such as A Record, CNAME record, and PTR Record on Infoblox.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

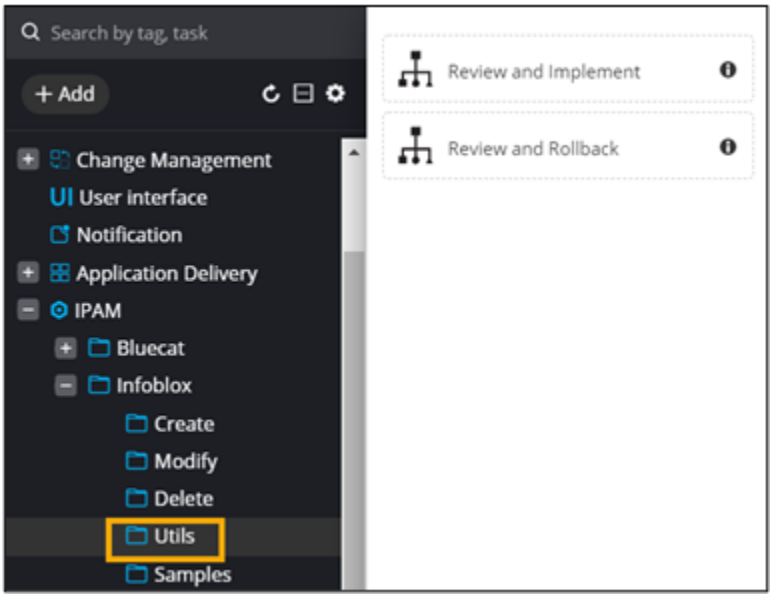
Delete

The **Delete** folder under **Infoblox** contains prebuilt tasks that can be leveraged to delete DNS records such as A Record, CNAME record, and PTR Record on Infoblox.



Utils

This folder contains common workflows that are used across workflows for **Infoblox** provisioning.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Note: For more information on using these prebuilt tasks to create DNS records on Infoblox, click here.

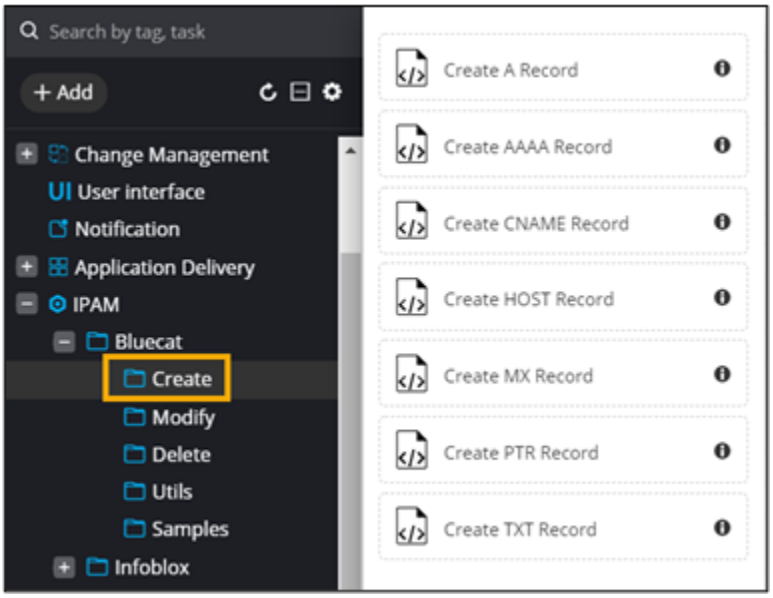
Bluecat Solutions

This section describes the tasks/workflows that can be leveraged to create, modify, and delete DNS records in Bluecat.

- Create
- Modify
- Delete
- Utils
- Creating a HOST Record in Bluecat

Create

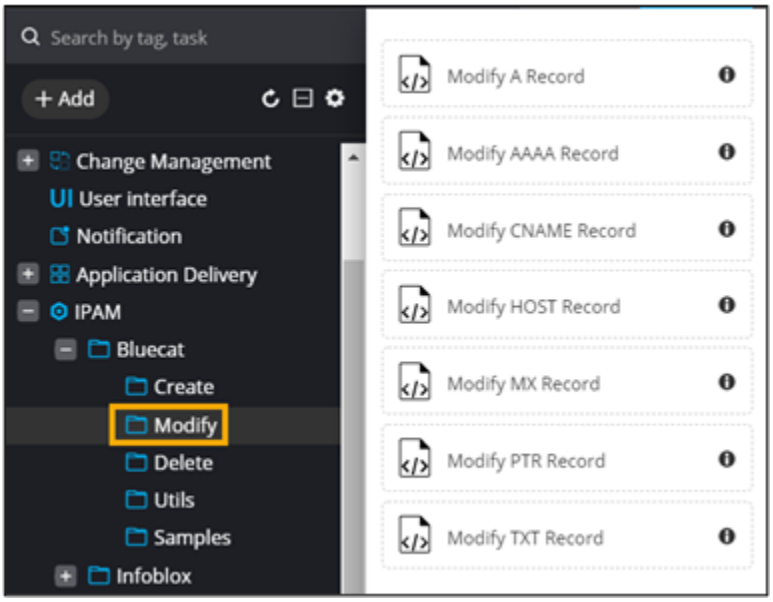
The **Create** folder under **Bluecat** contains prebuilt tasks that can be leveraged to create DNS records such as A Record, CNAME record, and PTR Record on Bluecat.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

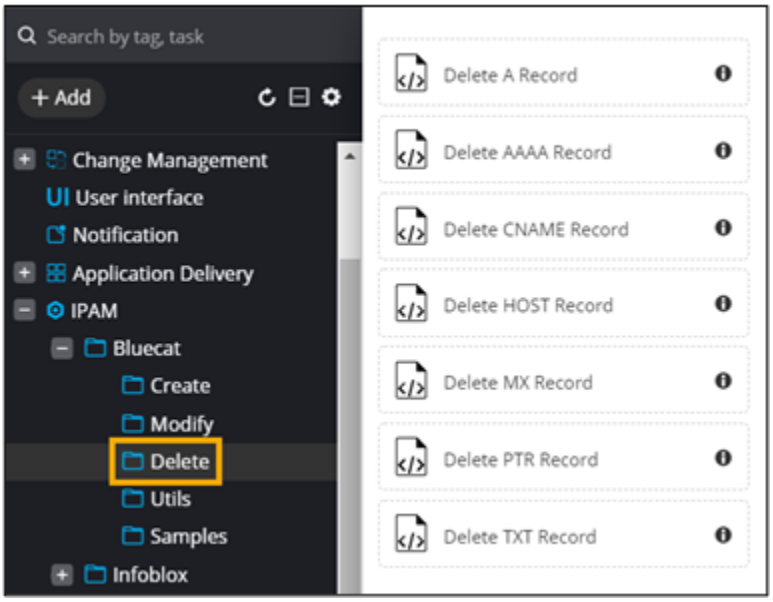
Modify

The **Modify** folder under **Bluecat** contains prebuilt tasks that can be leveraged to modify DNS records such as A Record, CNAME record, and PTR Record on Bluecat.



Delete

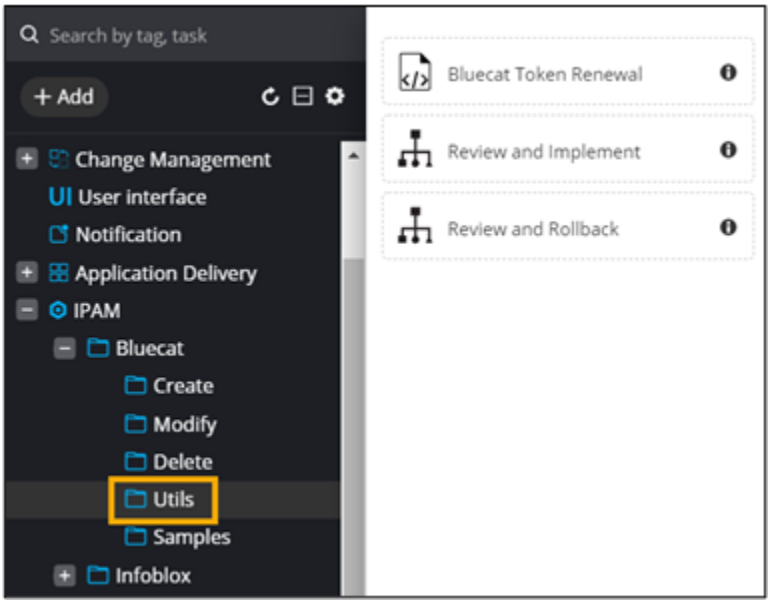
The **Delete** folder under **Bluecat** contains prebuilt tasks that can be leveraged to delete DNS records such as A Record, CNAME record, and PTR Record on Bluecat.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

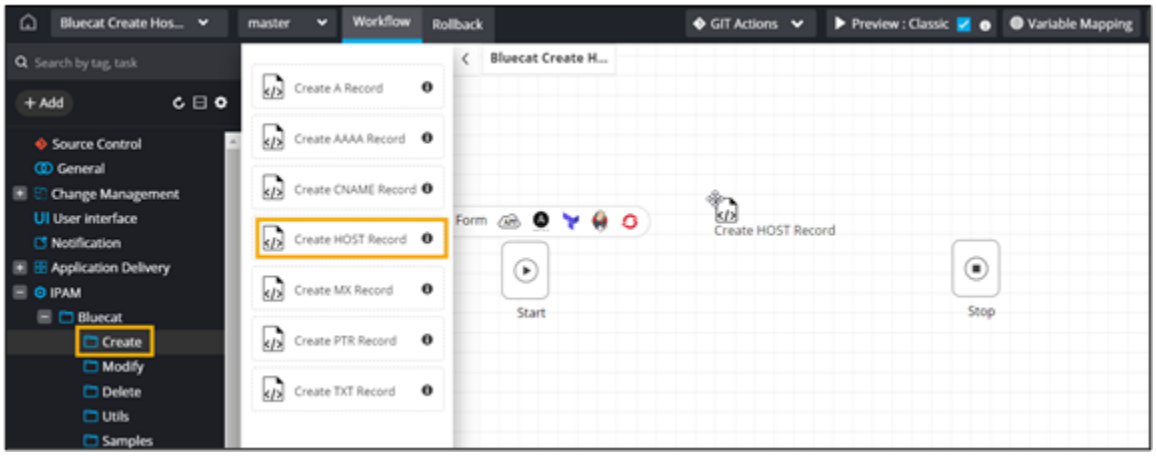
Utils

This folder contains common tasks and workflows that are used across workflows for **Bluecat** provisioning.



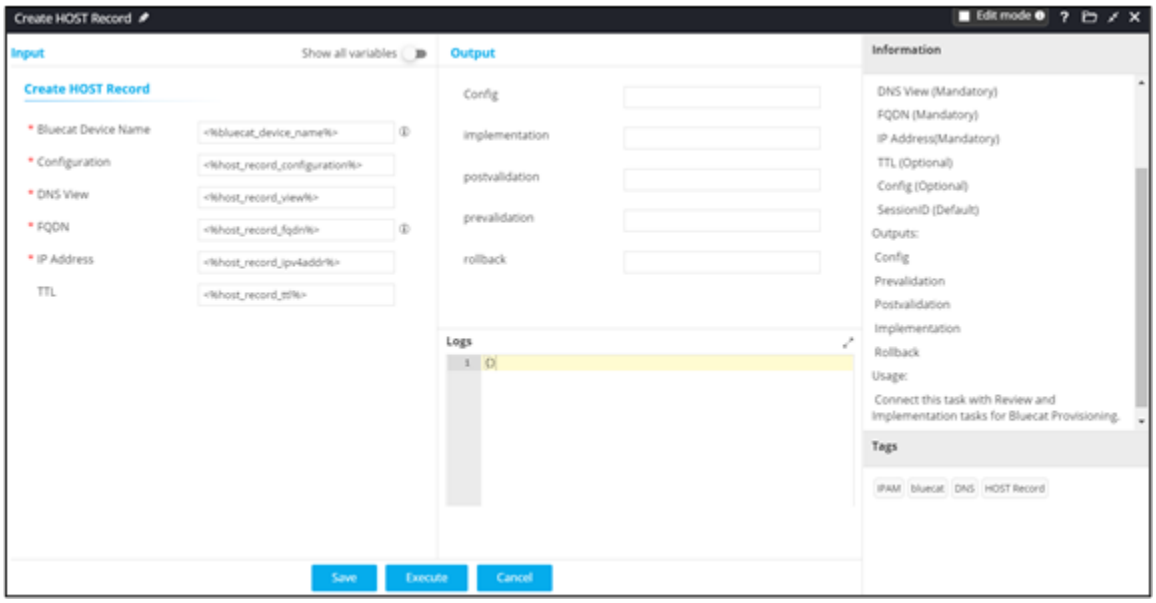
Creating a HOST Record in Bluecat

1. Design a new workflow.
2. From the menu on the left, select **IPAM** > **Bluecat** > **Create**.
3. From the **Create** folder, drag and drop the **Create HOST Record** task.

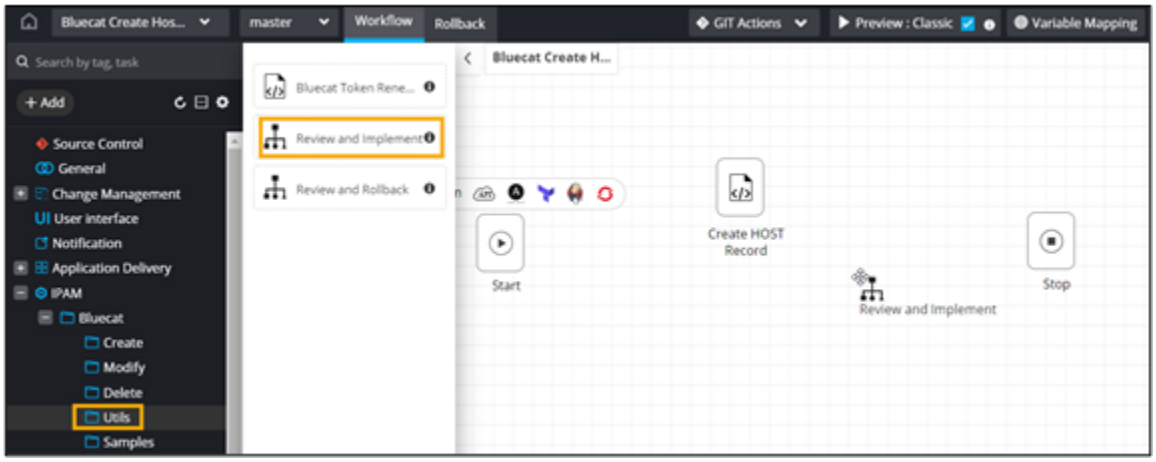


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

The **Create HOST Record** task window shows the defined **Input** and **Output** variables. The **Information** section shows what the task is for and also informs the user to connect this task with Review and Implementation tasks for Bluecat provisioning.



4. Click **Save**.
5. From the **Utils** folder, under **Bluecat**, drag and drop the **Review and Implement** workflow.

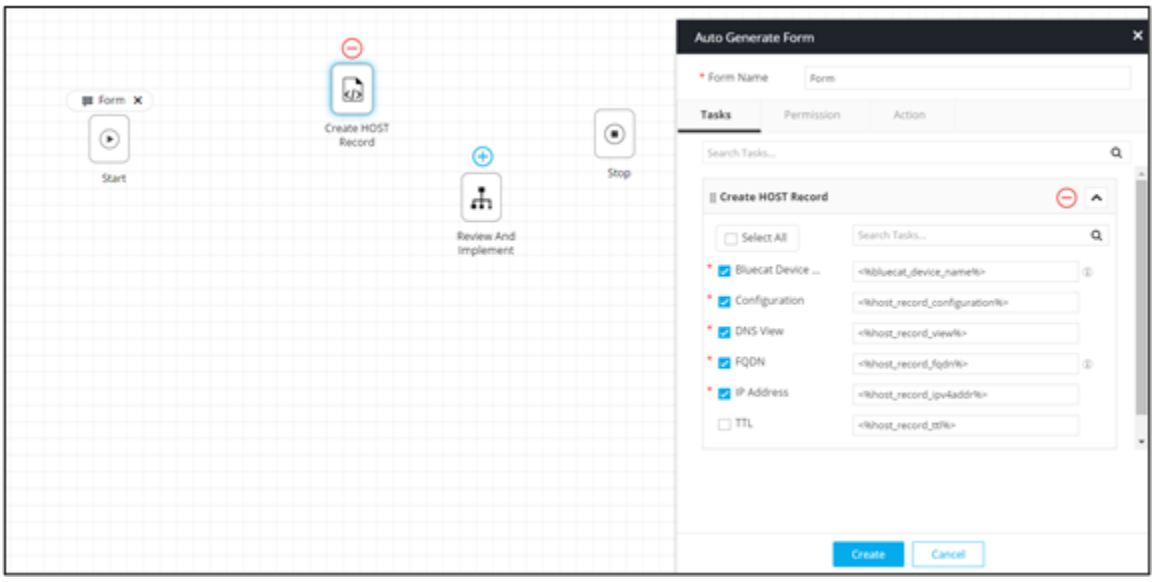


6. To auto-generate a form for this workflow, click **Form** above the **Start** task.
7. Click ⊕ above the **Create HOST Record** task and select the following tasks:

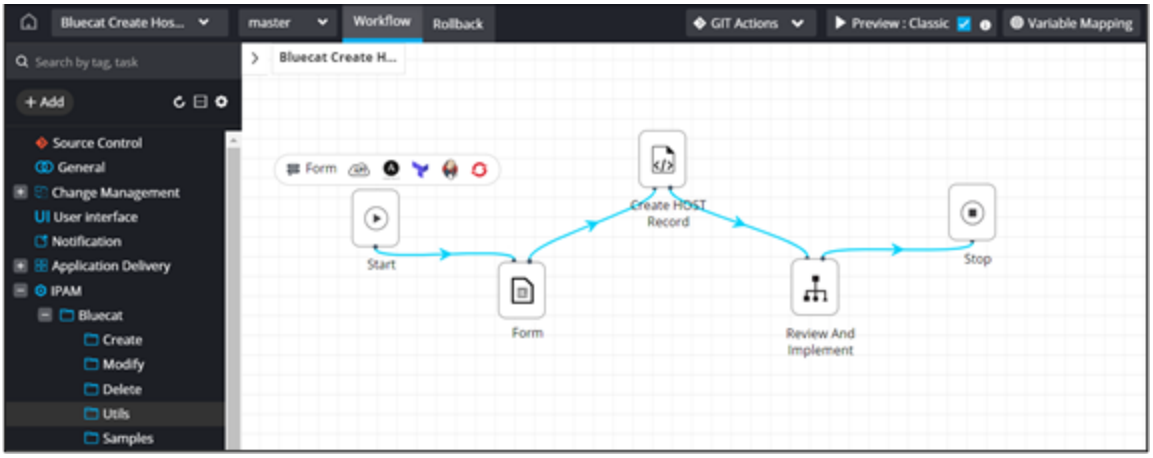
- Bluecat Device Name
- Configuration
- DNS View

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- FQDN
- IP Address

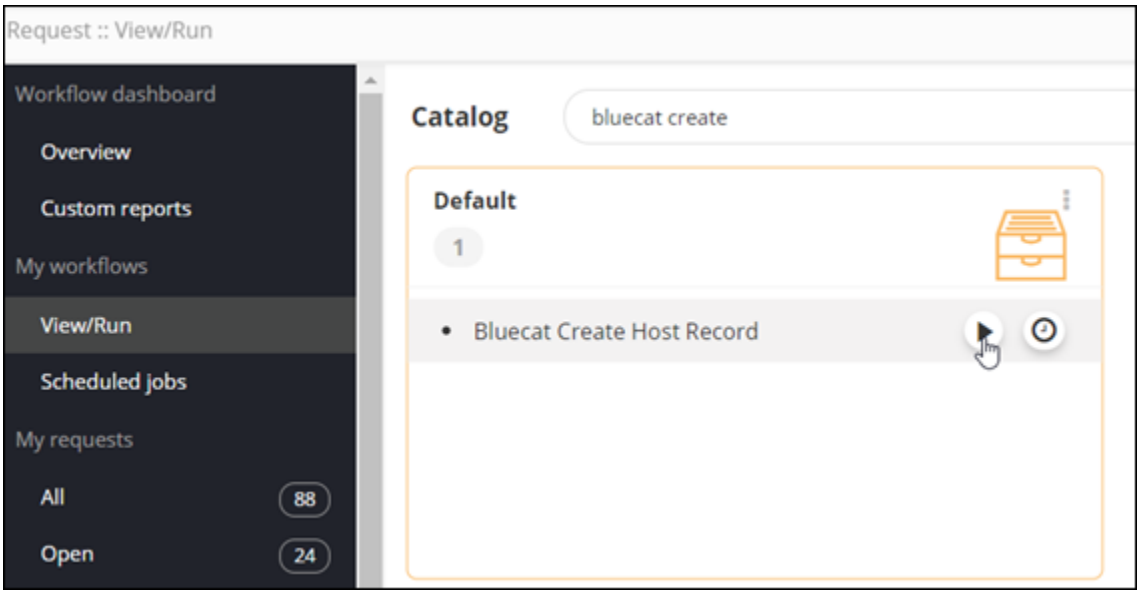


8. Click **Create**.
9. Connect and enable the workflow.

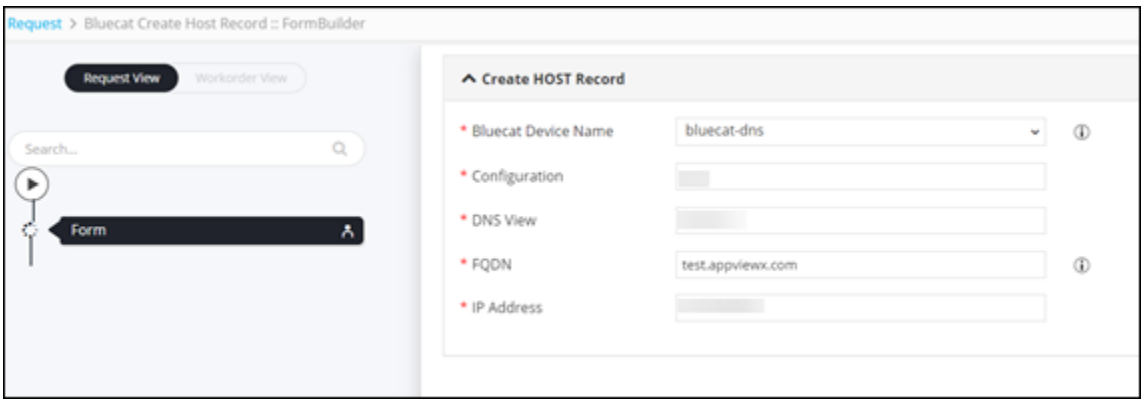


10. From the top left corner of the screen, click ☰.
11. From the menu displayed, click **Request**.
12. On the **Request :: Overview** page, from the navigation pane on the left, click **View/Run**.
13. To trigger the workflow, on the **Request :: View/Run** page, search for the workflow and click ▶.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



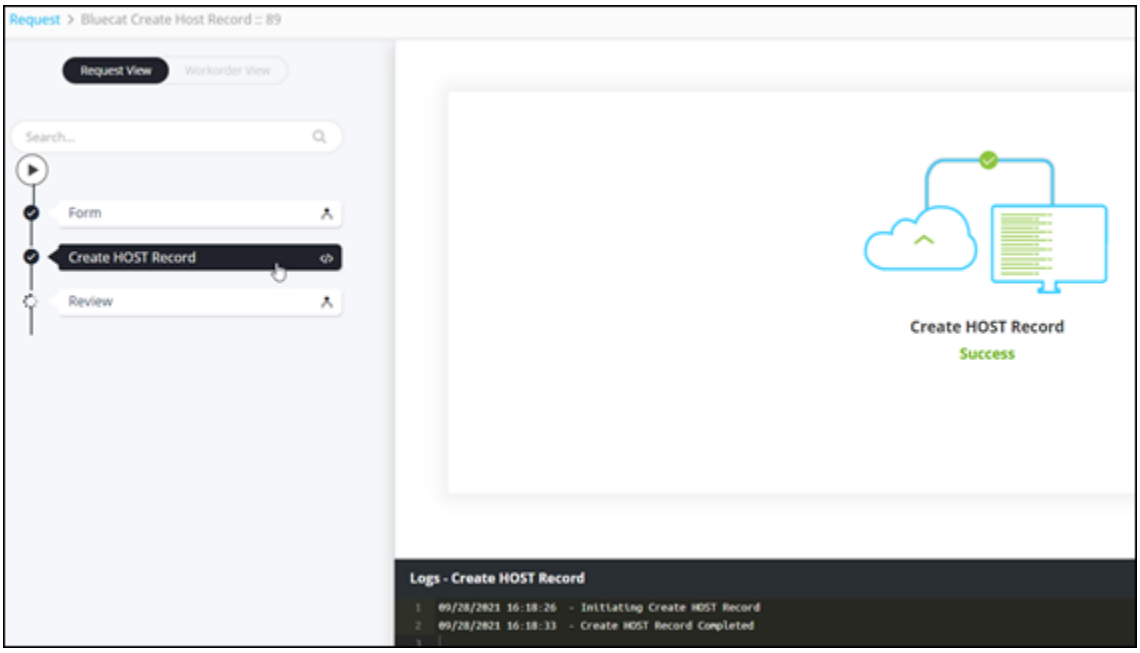
14. Enter the details in the input form.



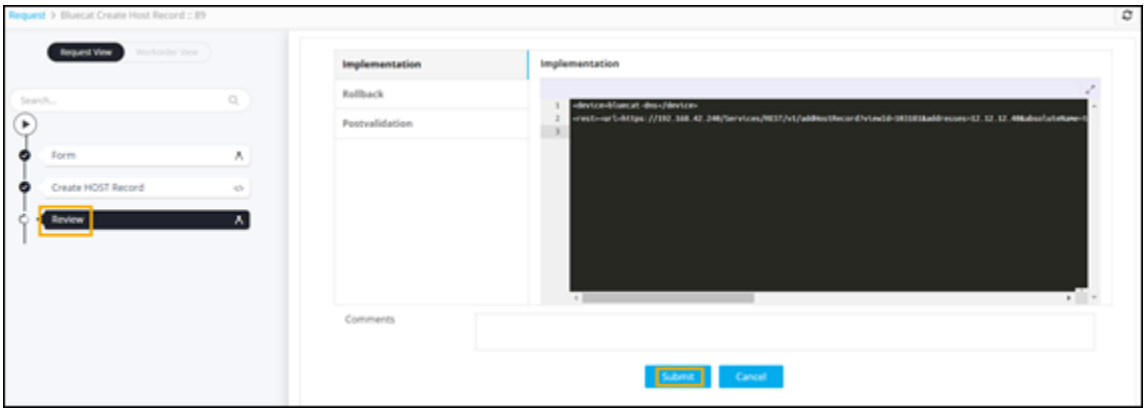
15. Click **Next**.

    Create HOST Record task completed.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



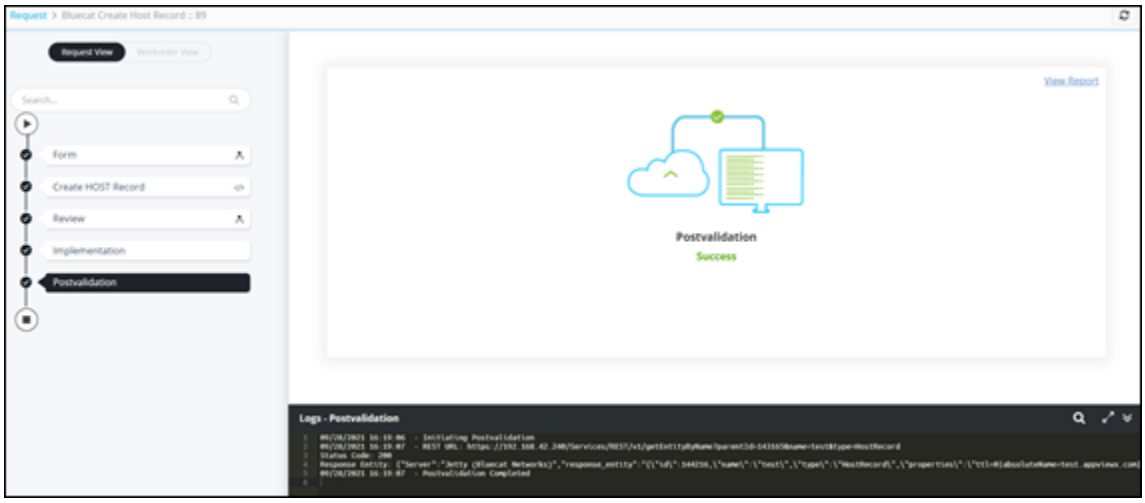
16. At the **Review** stage, click **Submit**.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Implementation task completed successfully.

- Postvalidation task completed successfully.



Change Automation

This folder consists of prebuilt tasks and workflows that allow you to integrate with ITSM tools. AppViewX supports the following vendors:

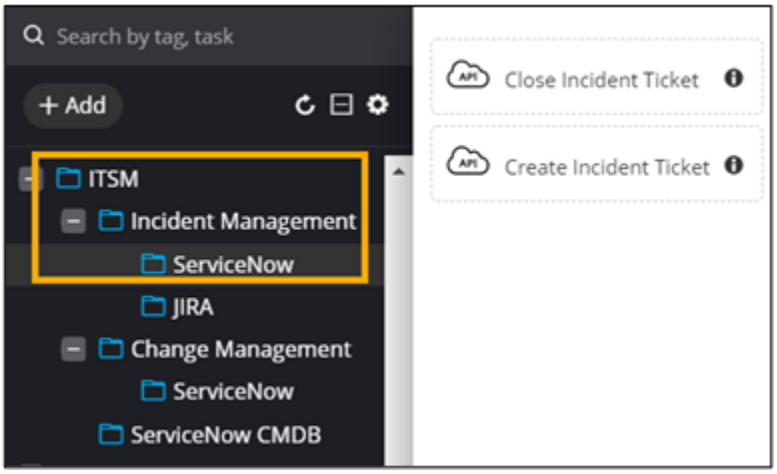
- ServiceNow
- Jira

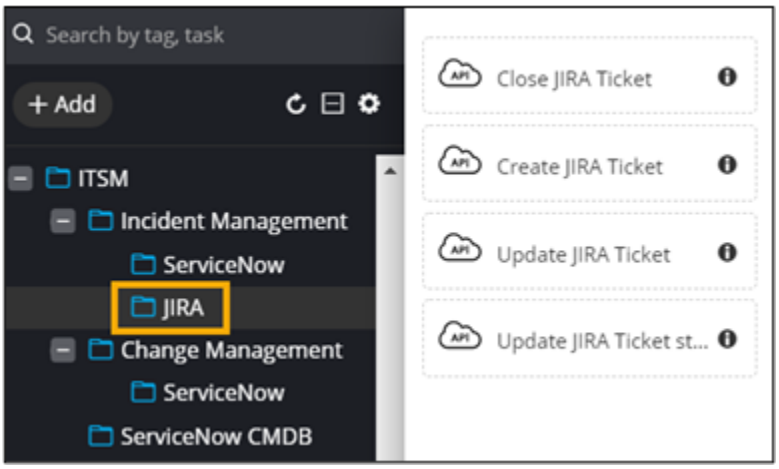
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Incident Management
- Change Management
- ServiceNow CMDB

Incident Management

The tasks available in the Incident Management folder allow you to design workflows for creating, updating and closing incident tickets on ServiceNow and Jira.



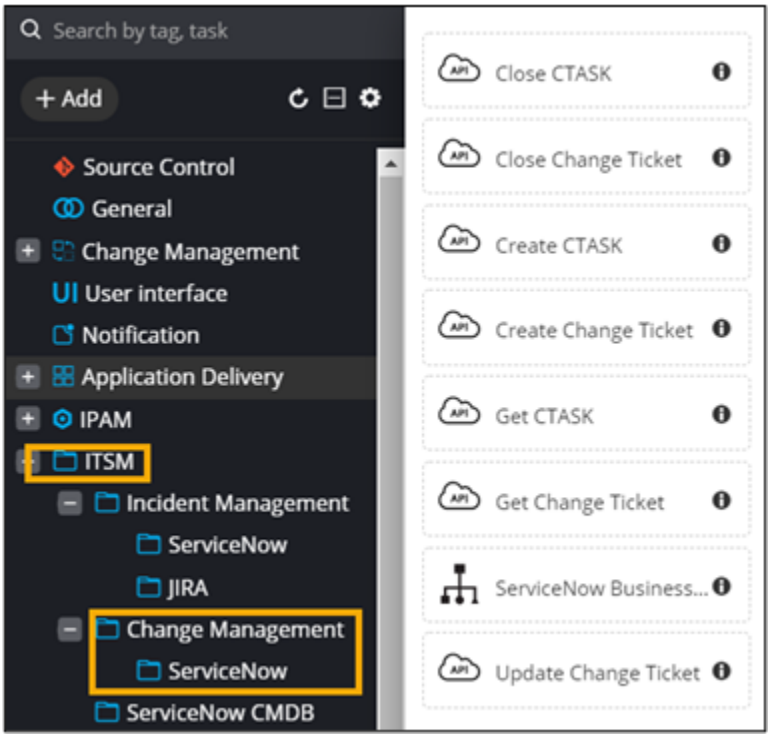


Note: For more information on using these tasks in a workflow, click here.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Change Management

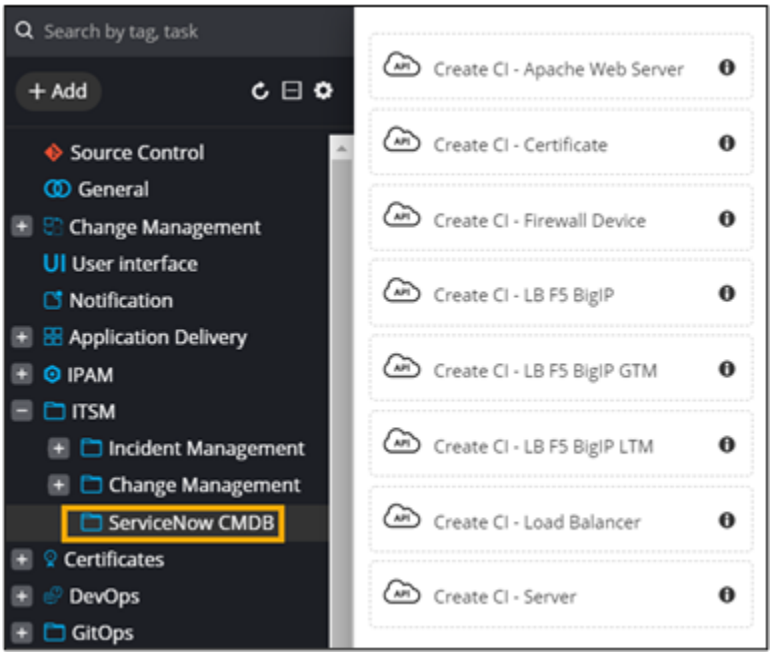
The tasks available in the **Change Management** folder allow you to design workflows for creating, updating and closing change tickets on ServiceNow.



ServiceNow CMDB

This folder consists of tasks and workflows that can be leveraged to push different CIs (configuration items) such as certificates, devices, and firewalls to ServiceNow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



DevOps and Continuous Configuration Automation

This folder consists of tasks and workflows that allow you to integrate with multiple vendors for network automation. AppViewX supports DevOps integration with Ansible and Jenkins.

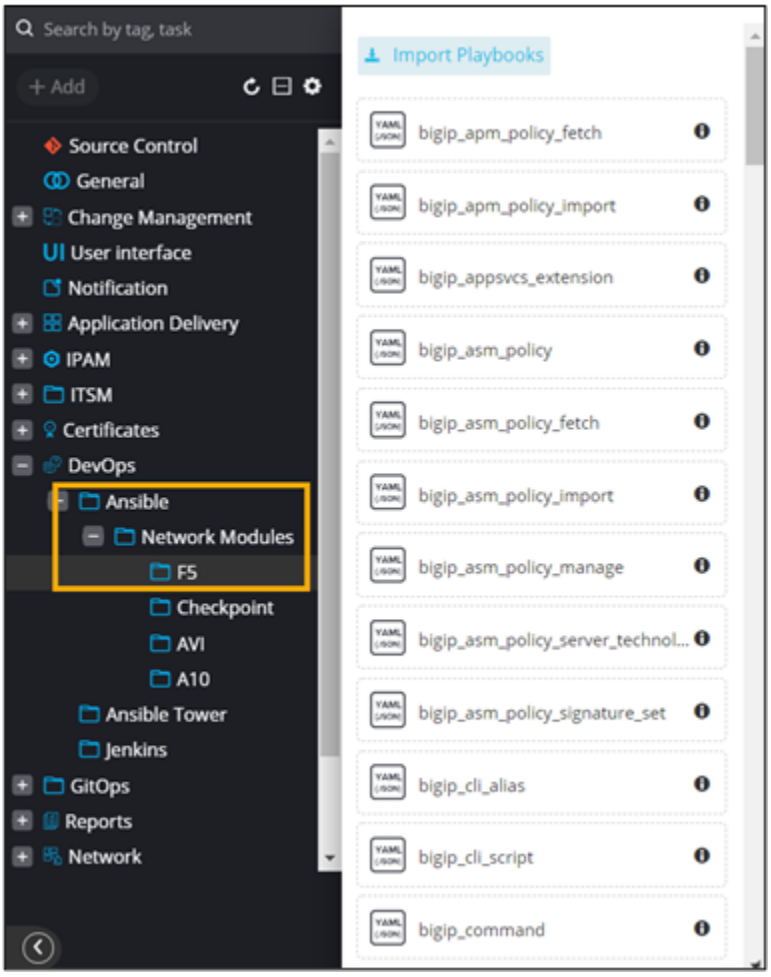
- Ansible
- Ansible Tower
- Jenkins

Ansible

This folder consists of the following Ansible Network Modules, which can be executed from playbooks:

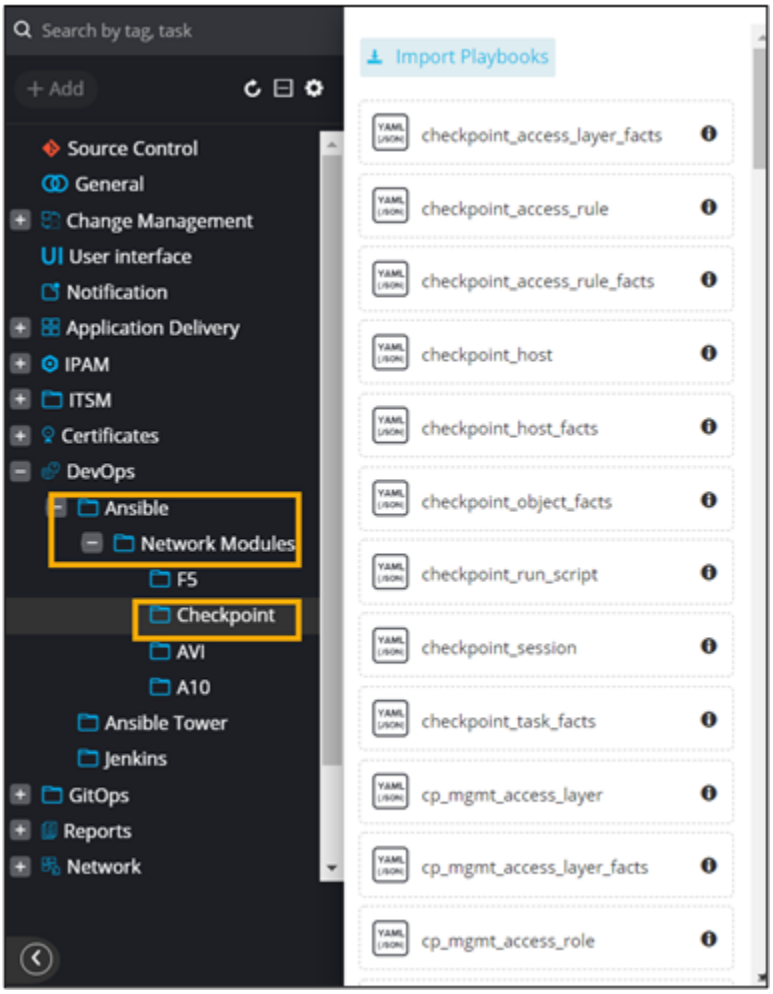
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- **F5**



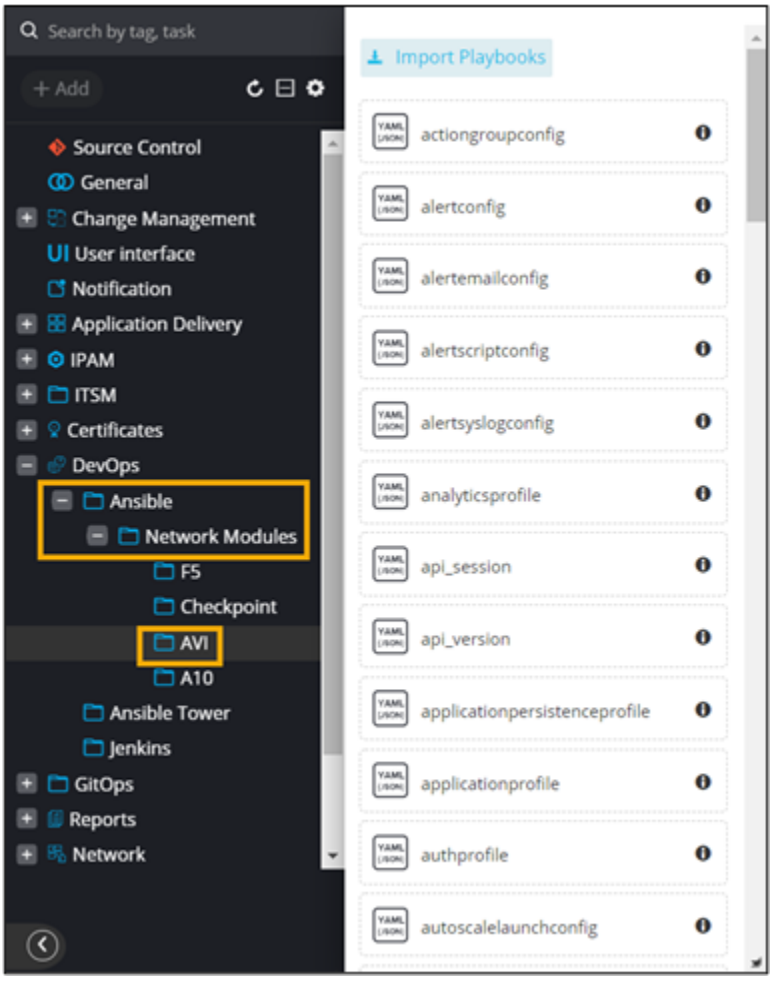
- **Checkpoint**

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



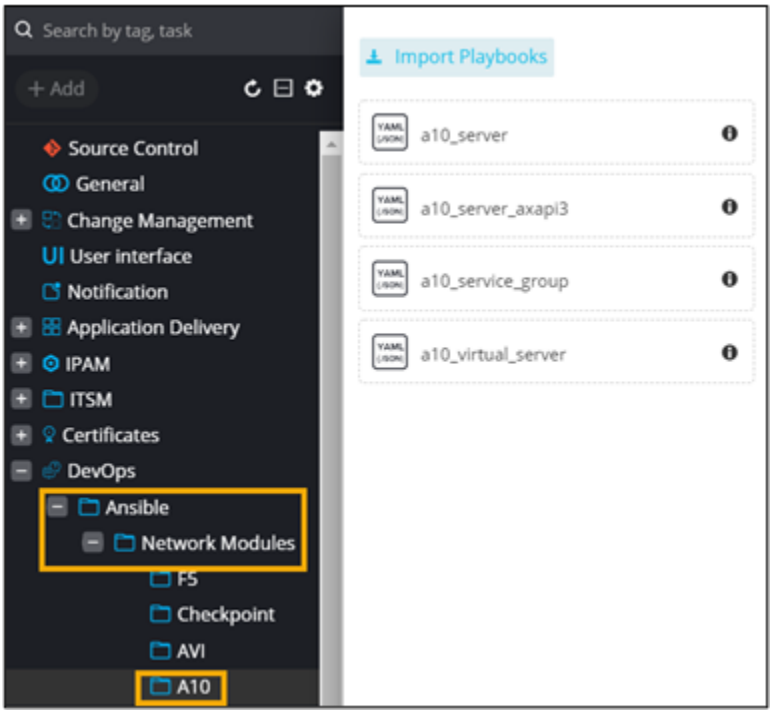
- **AVI**

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



- **A10**

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

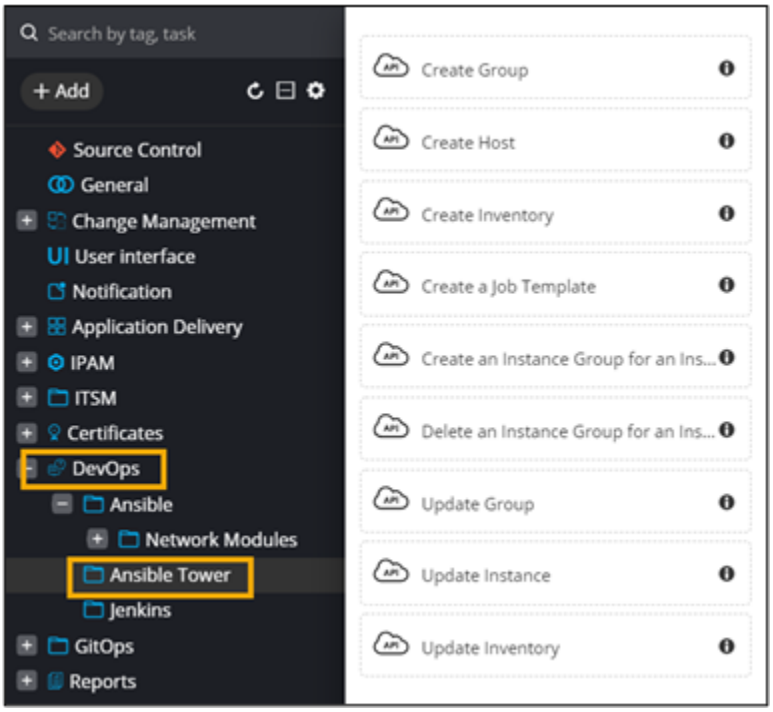


There is also a provision to import playbooks from Ansible.

Ansible Tower

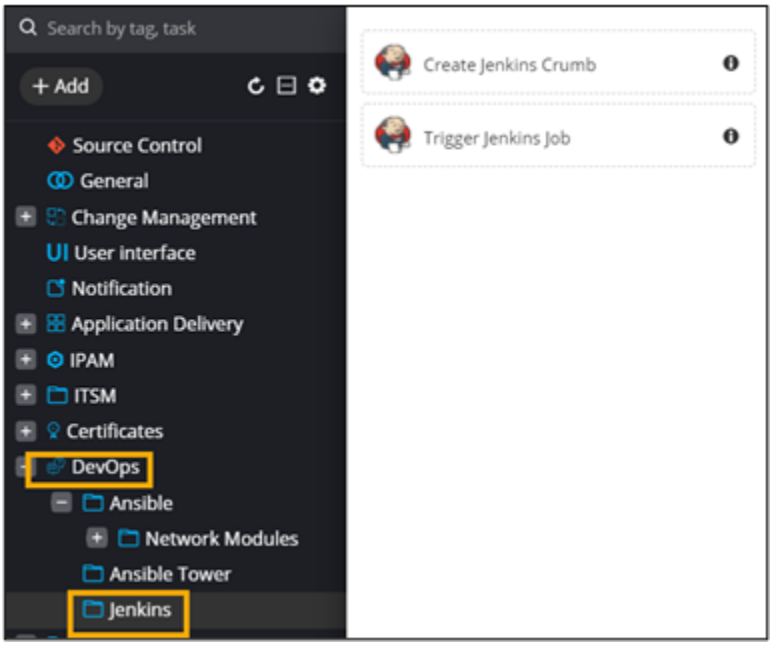
This folder consists of API tasks that can be used to create configurations on Ansible Tower and automate your tasks.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Jenkins

This folder contains tasks to allow you to create Jenkins jobs.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

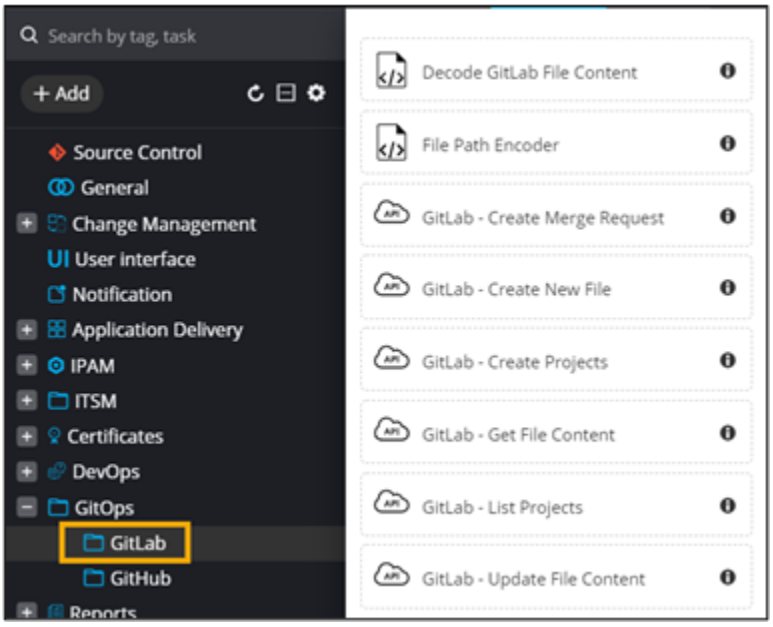
GitOps Integration

This folder consists of tasks that allow you to perform operations on GitHub and GitLab instances.

- GitLab
- GitHub

GitLab

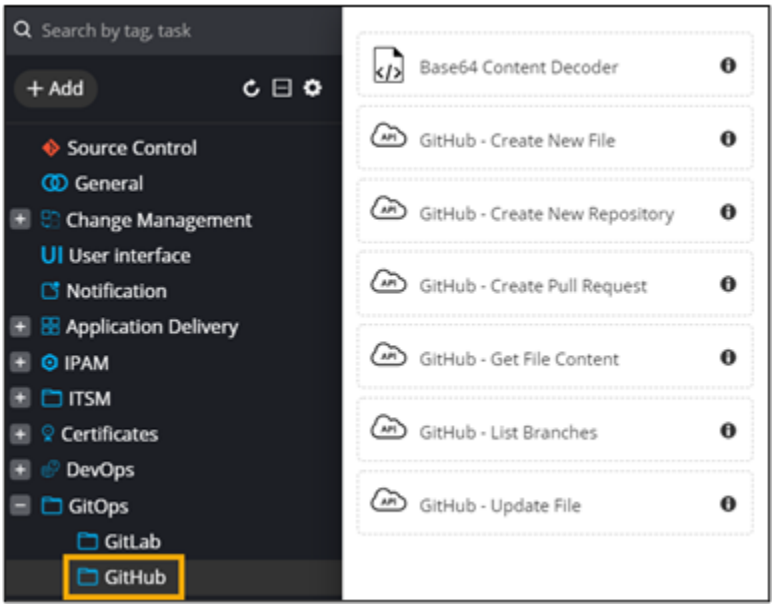
The **GitLab** folder contains tasks that can be used to perform actions on a GitLab instance.



GitHub

The **GitHub** folder contains tasks that can be used to perform actions on a GitHub instance.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Utilities

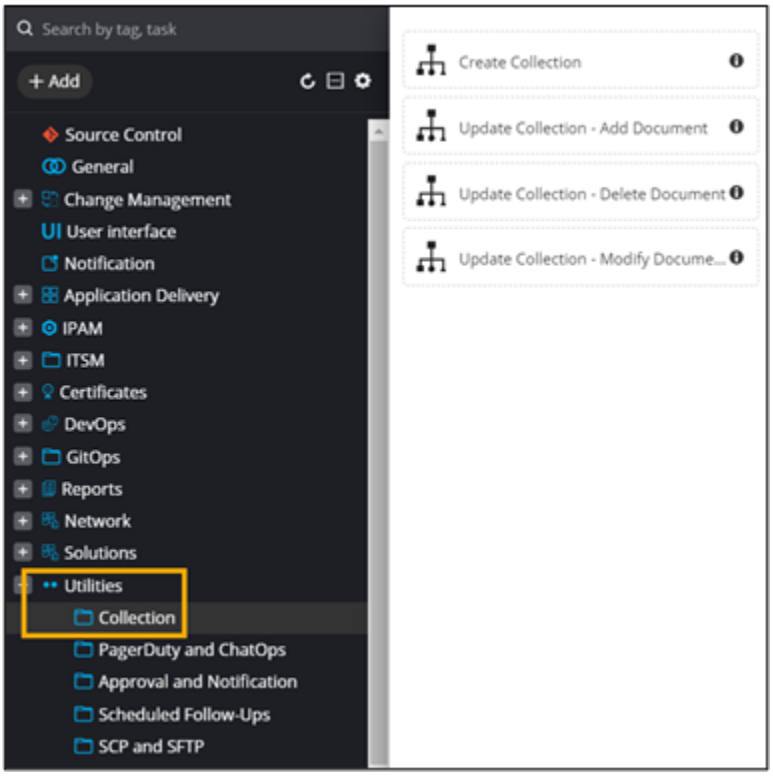
This folder consists of prebuilt flows that are commonly used across workflows, such as for creating collections, approvals and notifications, follow-ups, and file transfers.

- Collection
- Pagerduty and ChatOps
- Approval and Notification
- Scheduled Follow-Ups
- SCP and SFTP

Collection

This folder contains prebuilt workflows for creating and updating collections with metadata. These workflows can be used to add, delete or modify documents within an existing collection.

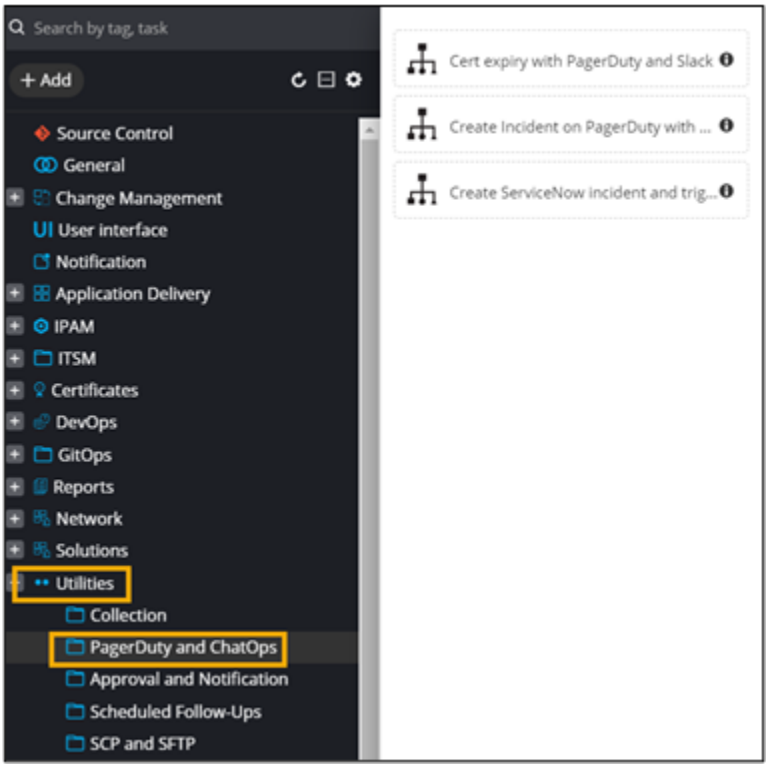
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Pagerduty and ChatOps

This folder contains prebuilt workflows for triggering alerts and notifications on PagerDuty, Slack, and other ChatOps after task execution.

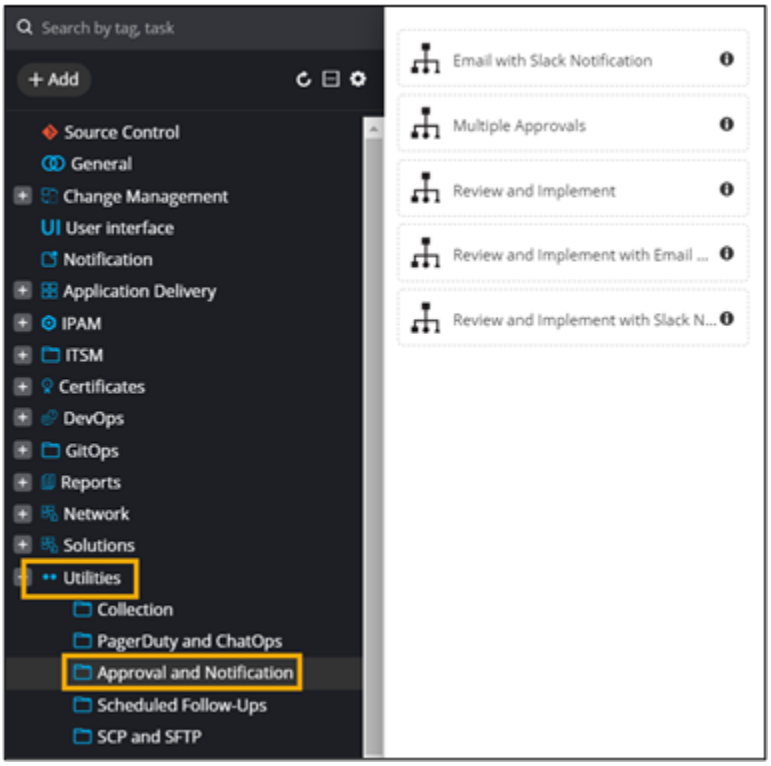
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Approval and Notification

This folder contains prebuilt workflows for sending approvals and notifications via email. These workflows can be used for seeking multiple approvals for task execution (review and implementation) and sending email notifications once a task is complete.

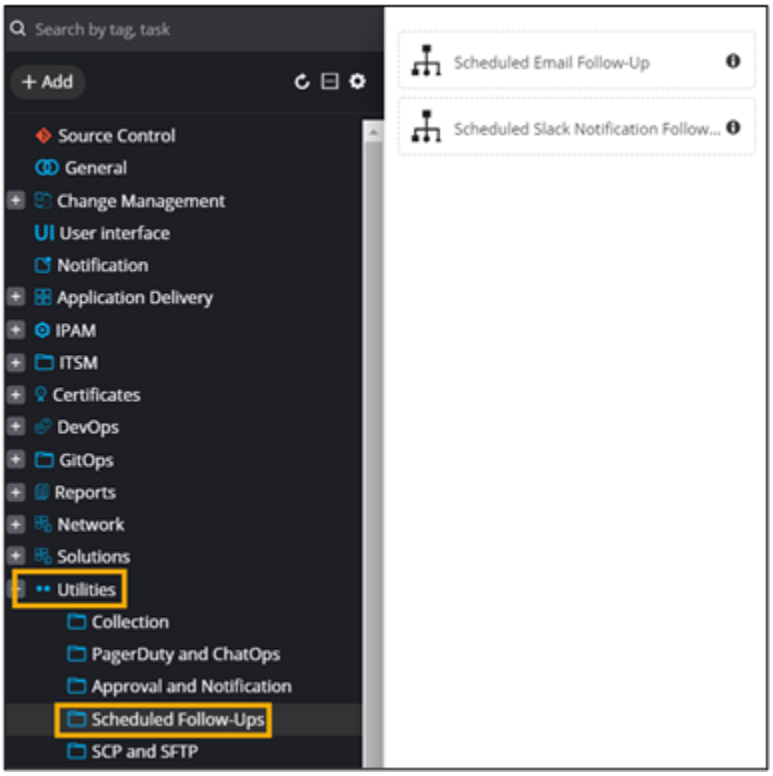
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Scheduled Follow-Ups

This folder contains prebuilt workflows that check for pending approvals with scheduled email or Slack notifications.

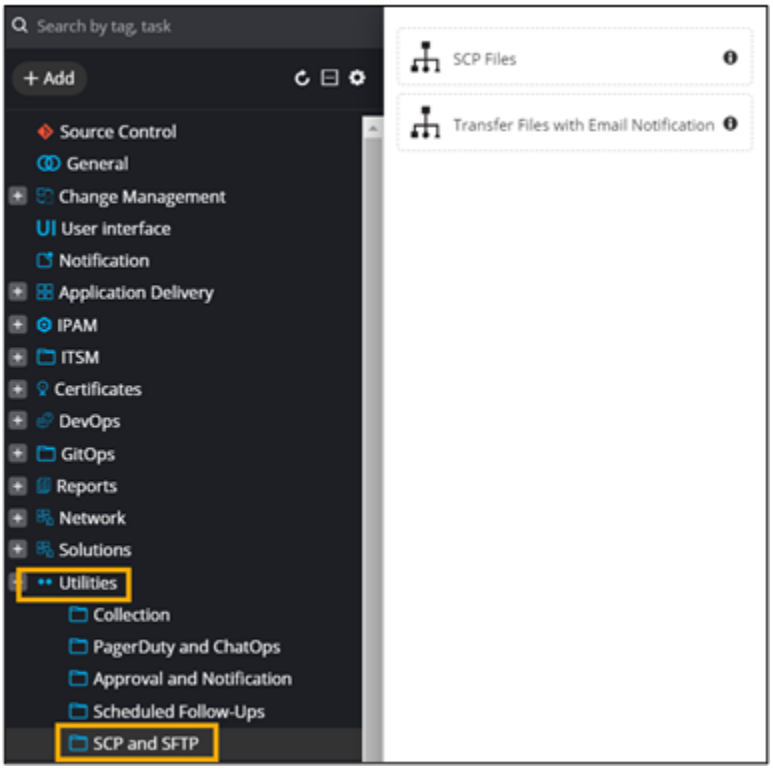
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



SCP and SFTP

This folder contains prebuilt workflows for transferring files (SCP and SFTP) with or without email notifications.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



SDK Automation

AppViewX's SDK Hub allows the user to import any Python SDK libraries, check for versions, and automate quickly across multiple vendors.
AppViewX SDKs are available for the following vendors:

- AVI
- Bluecat
- F5
- Infoblox

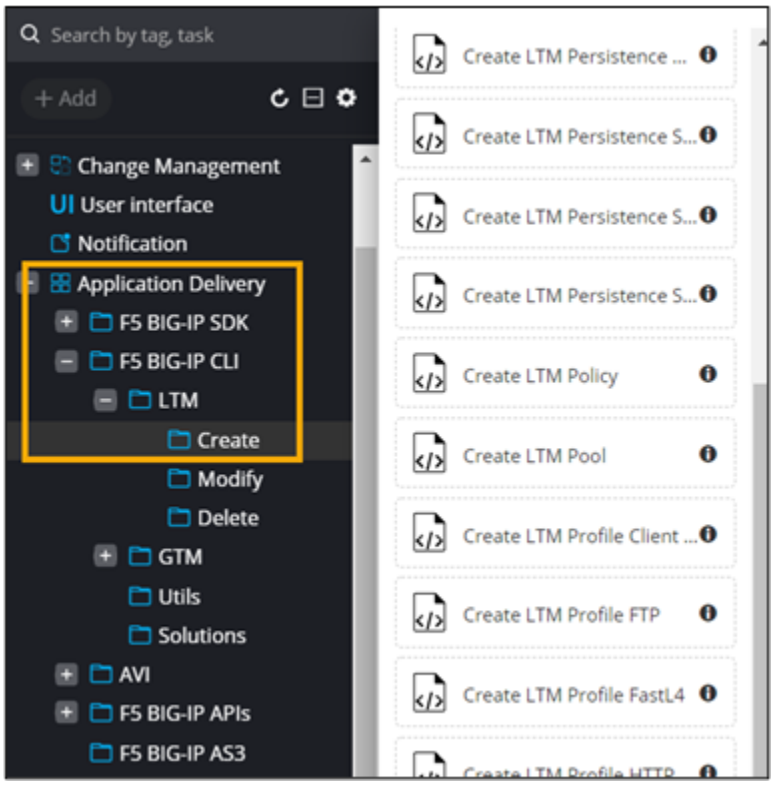
The pathway for the AppViewX folder is:

"<Installation Directory>/appviewx-dependencies/vw/dependencies/appviewx_sdk"

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

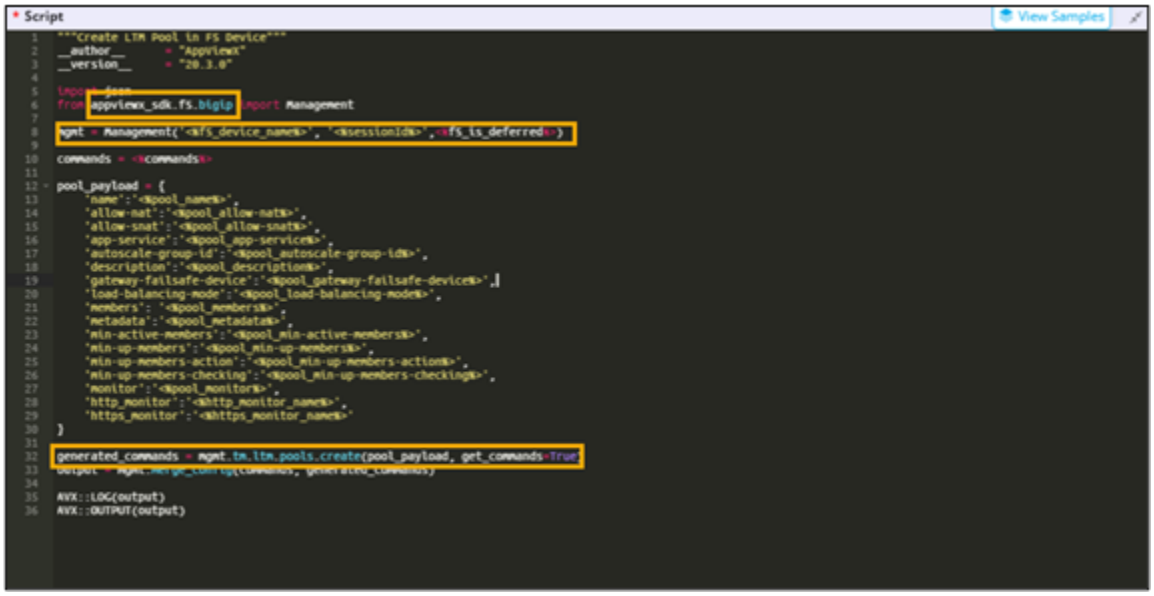
```
-bash-4.2$ pwd
/home/appviewx/appviewx/appviewx_dependencies/vw/dependencies
-bash-4.2$ ls -ltr | grep appviewx_sdk
drwxrwxr-x  8 appviewx appviewx      125 Jun 29 02:47 appviewx_sdk
-bash-4.2$ ls -ltr appviewx_sdk/
total 4
-rw-rw-r-- 1 appviewx appviewx 249 Jun  9 07:12 README.md
-rw-rw-r-- 1 appviewx appviewx   0 Jun  9 07:12 __init__.py
drwxrwxr-x 4 appviewx appviewx  51 Jun  9 07:12 f5
drwxrwxr-x 4 appviewx appviewx  88 Jun  9 07:12 bluecat
drwxrwxr-x 3 appviewx appviewx  81 Jun  9 07:12 avi
drwxr-xr-x 2 appviewx appviewx  37 Jun 29 02:47 __pycache__
drwxrwxr-x 3 appviewx appviewx 316 Jun 29 02:47 util
drwxrwxr-x 5 appviewx appviewx 107 Jul  7 05:25 infoblox
-bash-4.2$
```

1. Design a workflow.
2. From the **General** section, select **Application Delivery** > **F5 BIG-IP CLI** > **LTM** > **Create**.



3. Under **Create**, drag and drop the **Create LTM Pool** task. AppViewX SDK has been used in the script along with a command for creating an LTM pool.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



For more information on out of the box (OOB) workflows, refer to the ADC+ Automation Workflow User Guide.

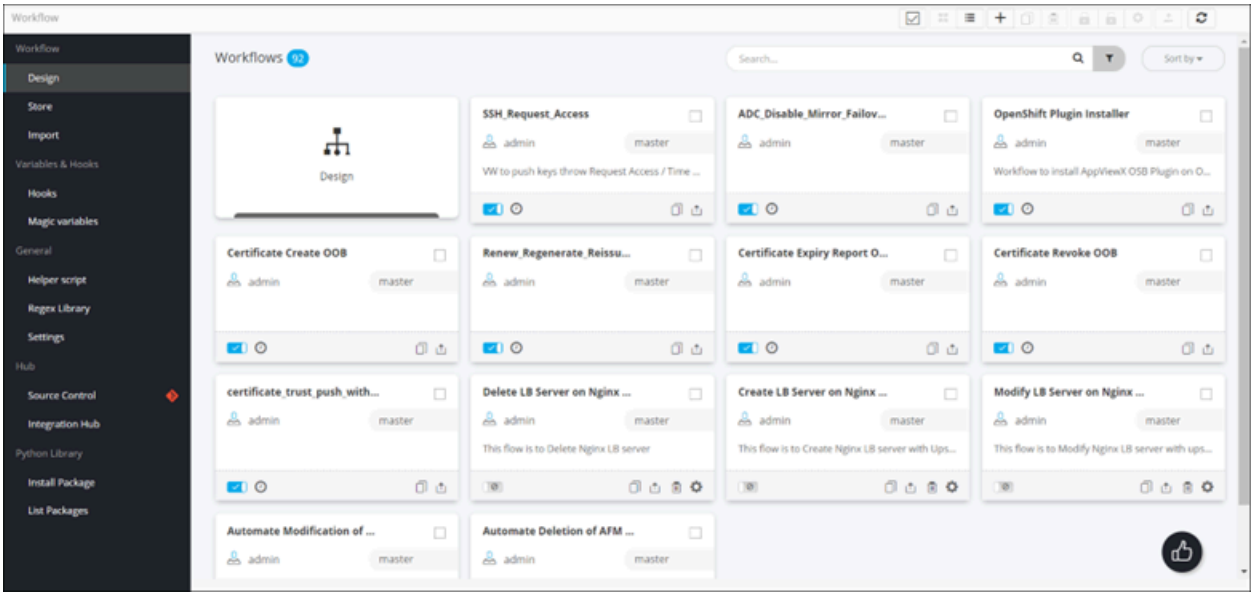
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Chapter 8: Workflow Inventory

- Overview
- Workflow Inventory Actions
- Workflow Inventory Menu

Overview

The Workflow Inventory page gives a consolidated view of the important aspects of the Visual Workflow platform and the actions that can be performed from the Workflow studio console.



The following table describes the various options available on the **Workflow** inventory page when the workflows are in **Card view** (default):

Option	Description
Workflow Menu	Allows you to navigate to different sections from the Workflow inventory page. **Note:** For more information, click here.
Design	Allows you to create a new workflow from scratch.

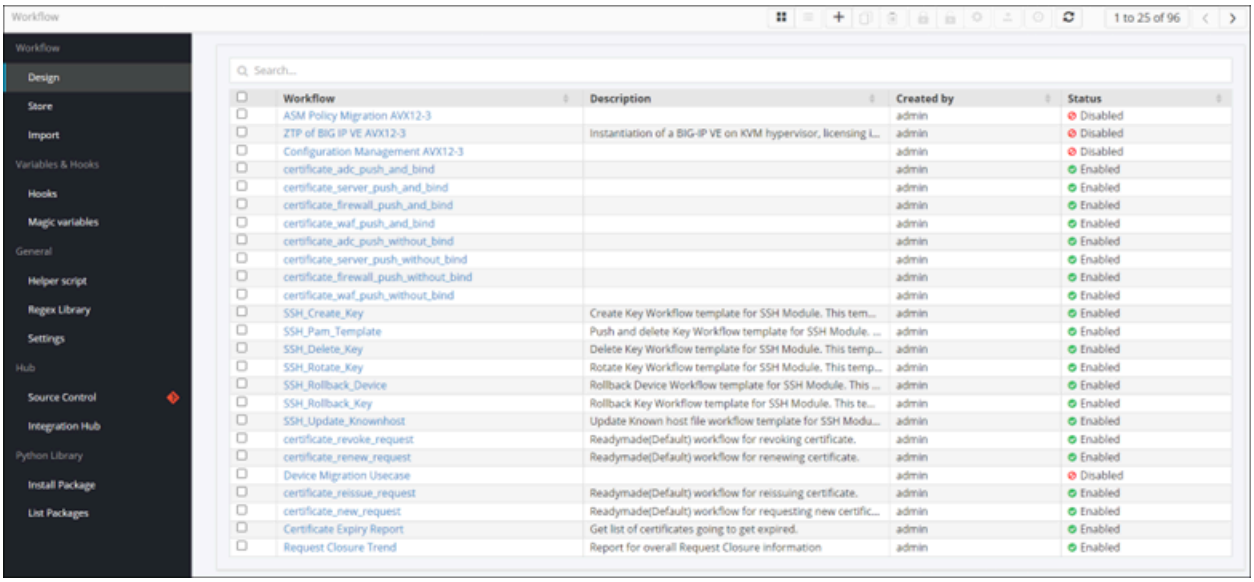
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Option	Description
	Note: For more information, click here.
Search bar	Allows you to search for a particular workflow by typing in the keyword.
	Allows you to filter workflows based on the category selected in the dropdown.
Source Control	Allows you to push, pull, and commit changes to remote repositories. **Note:** For more information, click here.
	Allows you to select all workflows displayed on the page with a single click.
	Displays the card view of the inventory.
	Displays the list view of the inventory.
	Allows you create a new workflow from scratch. **Note:** For more information, click here.
	Allows you to clone a workflow. **Note:** For more information, click here.
	Allows you to enable selected workflow(s). **Note:** For more information, click here.
	Allows you to disable selected workflow(s). **Note:** For more information, click here.
	Allows you to define RBAC and other settings for a particular workflow. **Note:** For more information, click here.
	Allows you to export selected workflow(s).

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Option	Description
	Note: For more information, click here.
	Refreshes the page.

Note: The **Workflow** inventory page is displayed in **Card view** by default. You can switch to the **List view** by clicking .



The following table describes the various options available on the **Workflow** inventory page when the workflows are in **List view**:

Option	Description
Workflow Menu	Allows you to navigate to different sections from the Workflow inventory page. **Note:** For more information, click here.
Search bar	Allows you to search for a particular workflow by typing in the keyword.
	Displays the card view of the inventory.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Option	Description
	Displays the list view of the inventory.
	Allows you create a new workflow from scratch. **Note:** For more information, click here.
	Allows you to clone a workflow. **Note:** For more information, click here.
	Allows you to delete selected workflow(s). **Note:** For more information, click here.
	Allows you to enable selected workflow(s). **Note:** For more information, click here.
	Allows you to disable selected workflow(s). **Note:** For more information, click here.
	Allows you to define RBAC and other settings for a particular workflow. **Note:** For more information, click here.
	Allows you to export selected workflow(s). **Note:** For more information, click here.
	Allows you to schedule the selected workflow. **Note:** For more information, click here.
	Refreshes the page.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

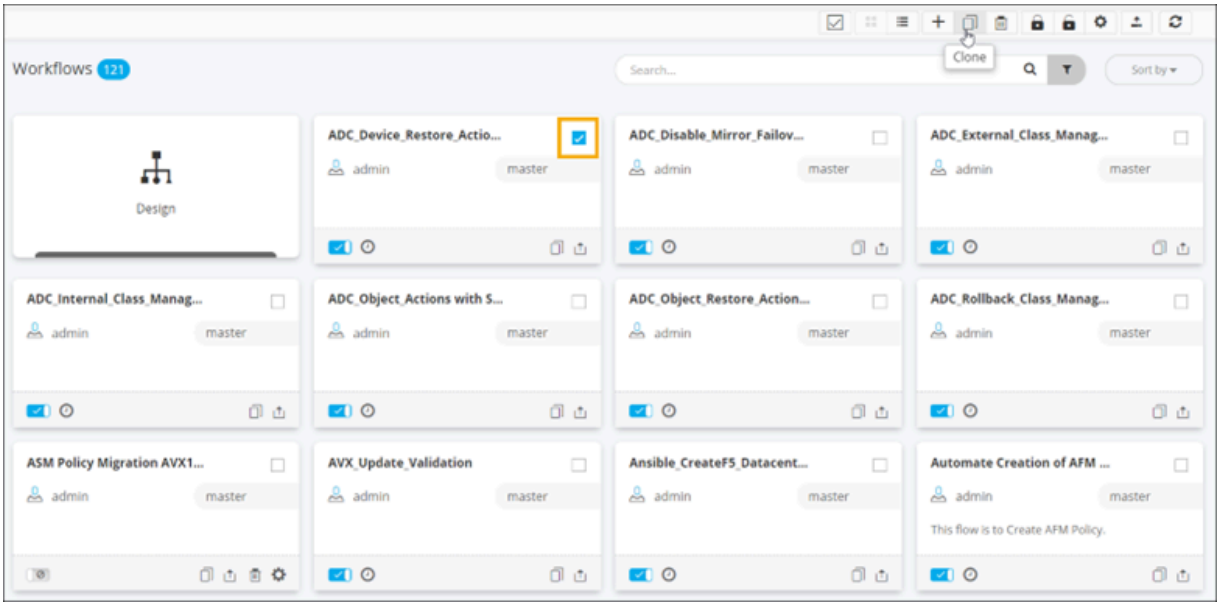
Workflow Inventory Actions

You can perform various actions on the workflows such clone, enable, disable, export, and delete.

- Cloning a Workflow
- Enabling a Workflow
- Disabling a Workflow
- Deleting a Workflow
- Exporting a Workflow

Cloning a Workflow

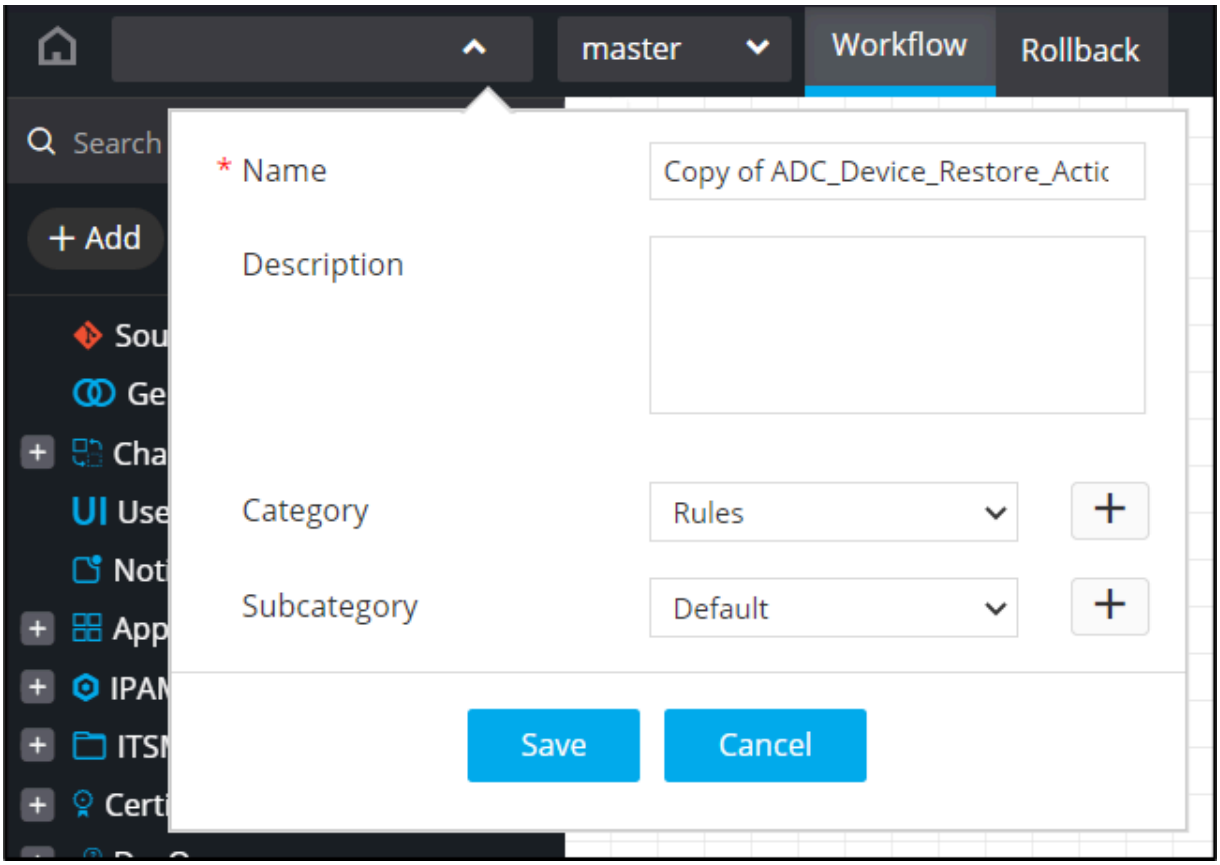
1. On the **Workflow** inventory page (**Card view**), select the workflow(s) to be cloned.
2. To clone the workflow, from the command bar, click .



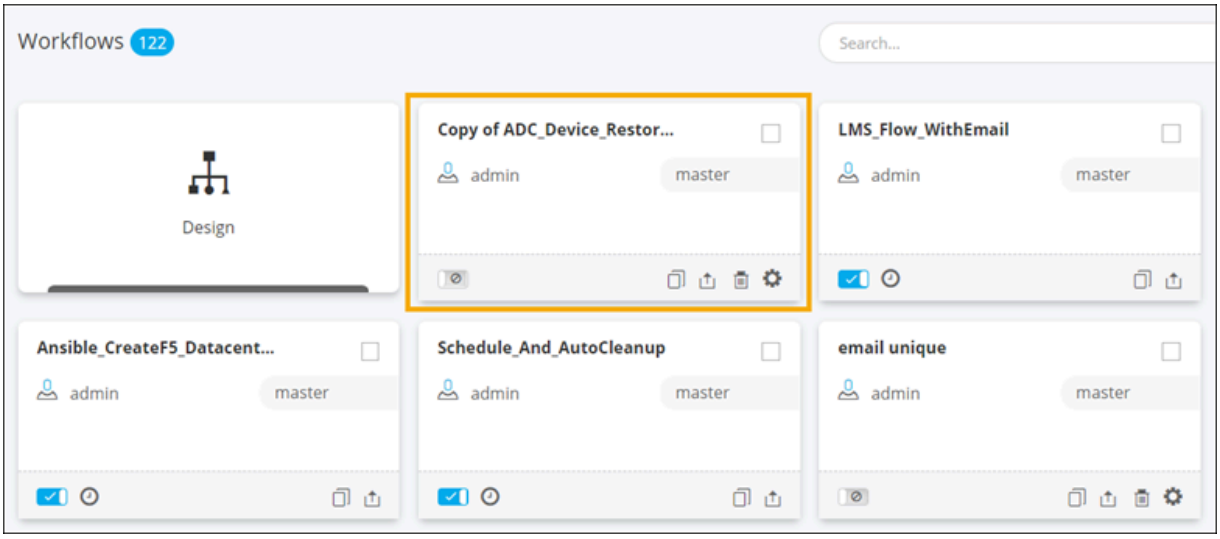
A copy of the cloned workflow opens in the design workspace.

3. Enter a **Name** for the cloned workflow and click **Save**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



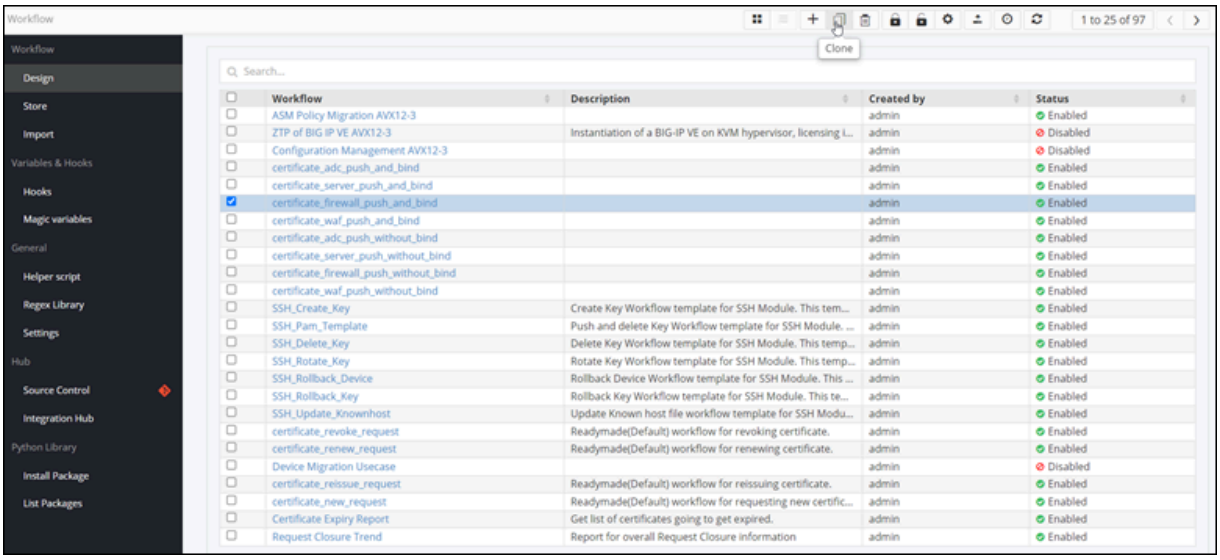
A copy of the cloned workflow is added to the Workflow Inventory page.



Tip: To clone a workflow, you can also click on the workflow to be cloned.

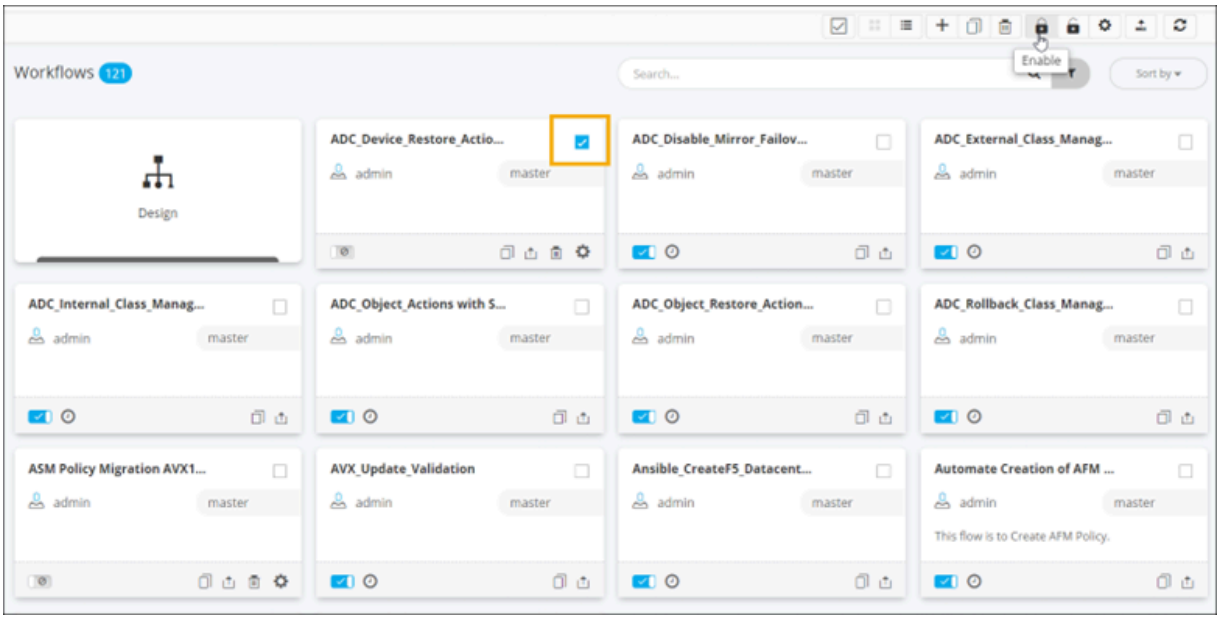
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

4. To clone a workflow when the Workflow inventory is in **List view**, select the workflow to be cloned and click .



Enabling a Workflow

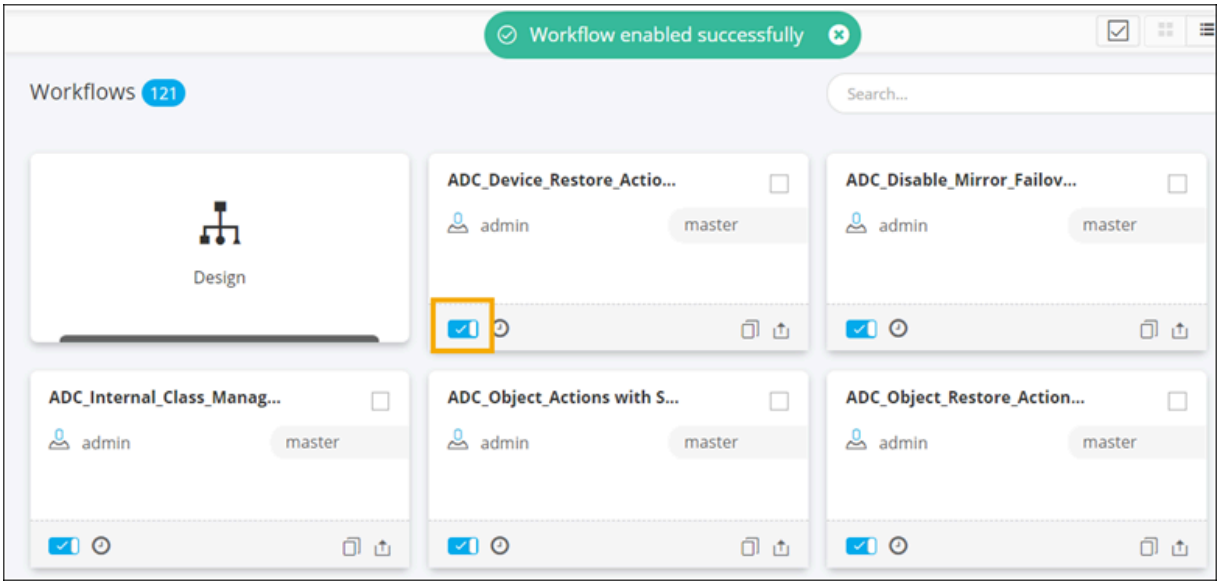
1. On the **Workflow** inventory page (**Card view**), select the workflow to be enabled.
2. To enable the selected workflow, from the command bar click .



3. Click **Yes** in the **Confirmation** pop-up window.

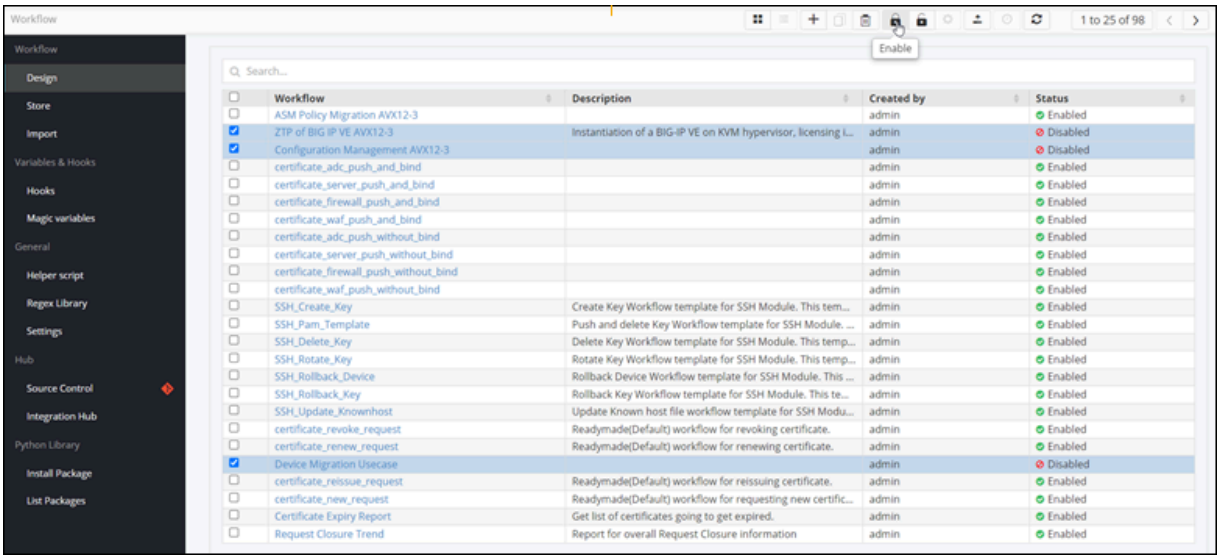
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

The workflow is enabled.



Tip: You can also enable the workflow by turning on the toggle on the selected workflow card.

4. To enable multiple workflows, you can either individually select the workflows to be enabled or click to select all workflows displayed on the page and click in the command bar.
5. To enable a workflow, when the **Workflow** inventory is in **List view**, select the workflow(s) to be enabled and click .

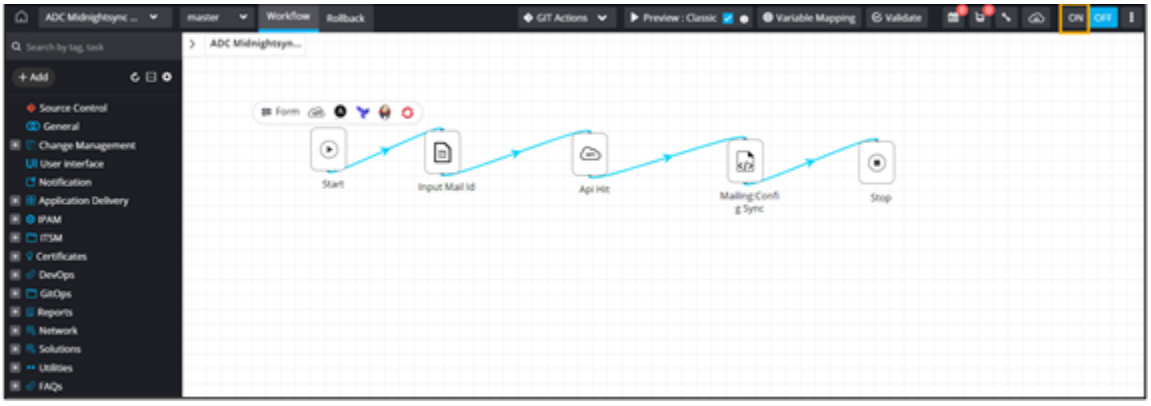


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

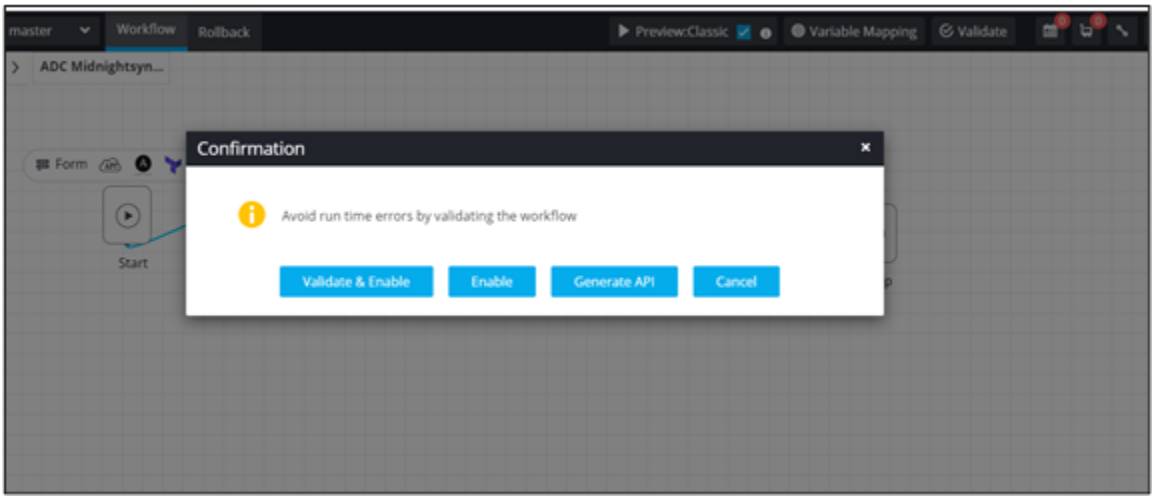
• Enabling a Workflow in Design Mode

Enabling a Workflow in Design Mode

1. Open the workflow in the Workflow Studio.
2. From the right upper corner of the screen, click **ON**.



3. In the **Confirmation** window, click **Validate & Enable**.

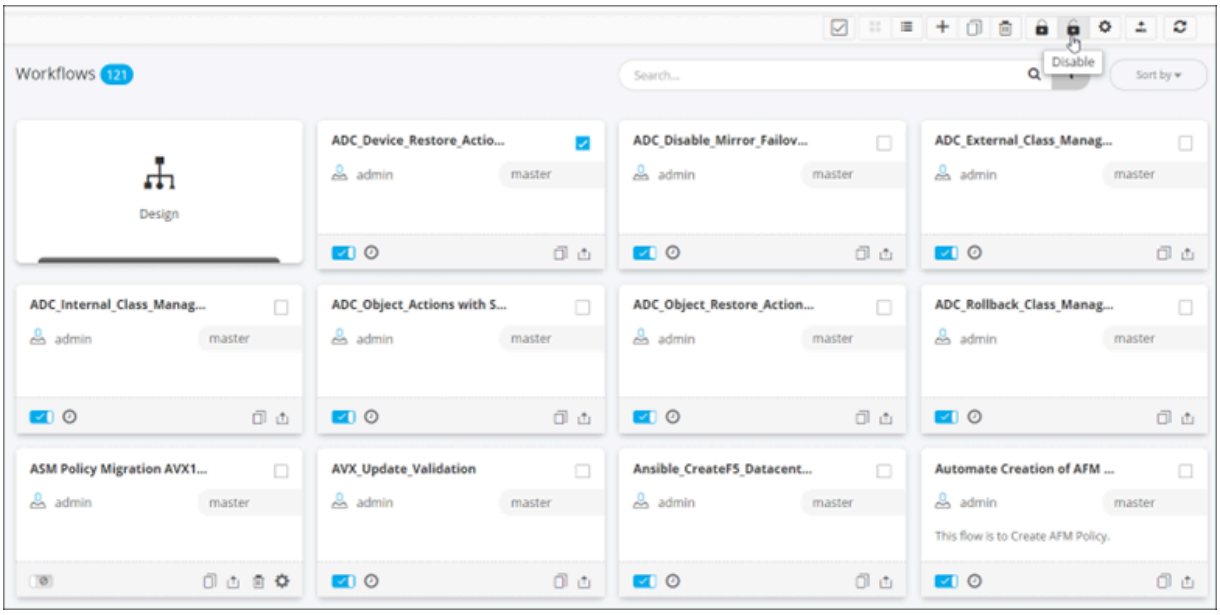


Note: For more information on validating a workflow, click here.

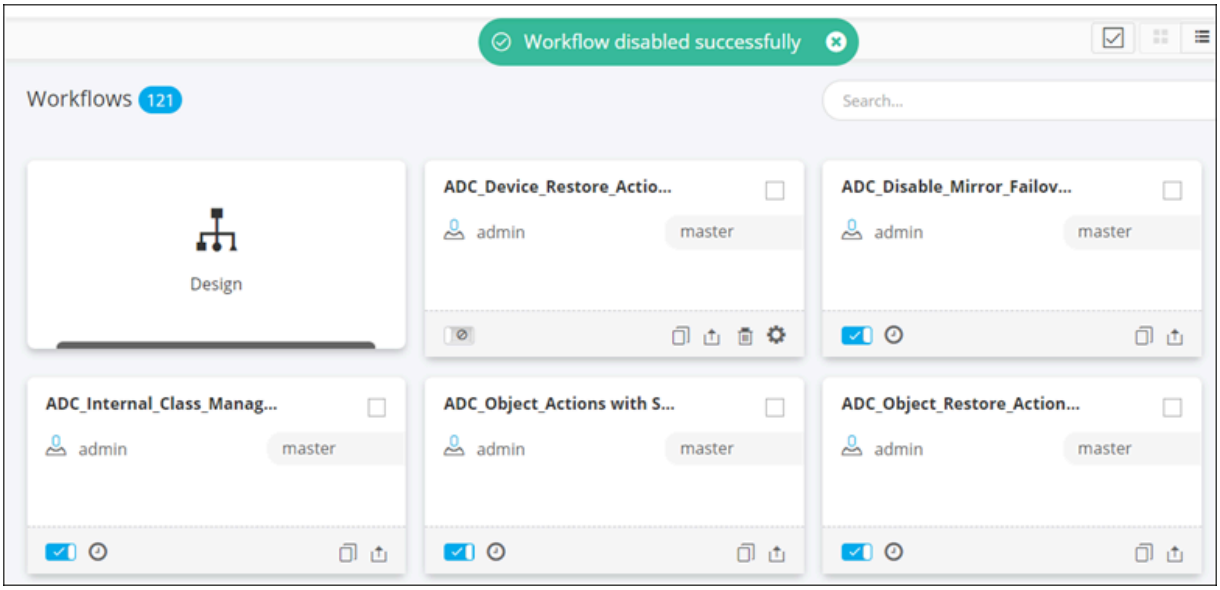
Disabling a Workflow

1. On the **Workflow** inventory page (**Card view**), select the workflow to be disabled.
2. To disable the selected workflow, from the command bar, click .

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



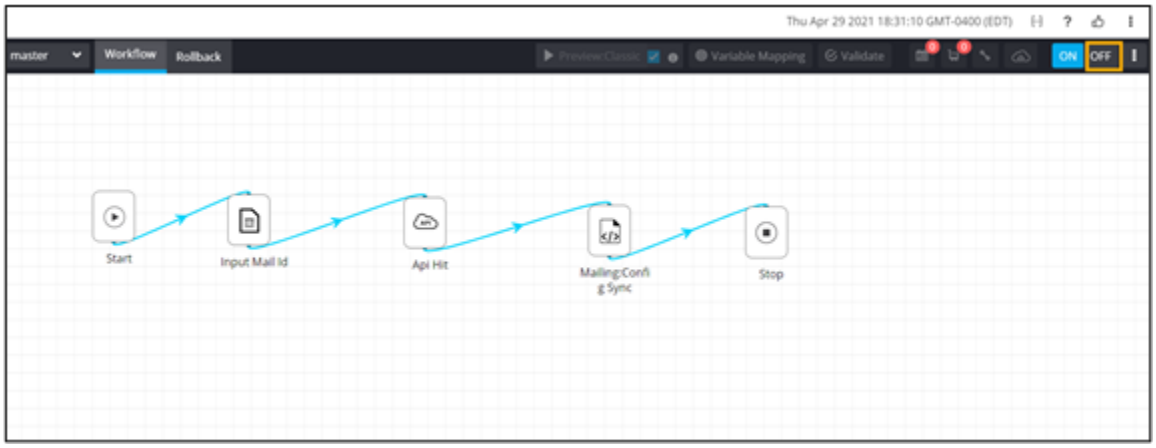
3. Click **Yes** in the **Confirmation** pop-up window.
   The workflow is disabled.



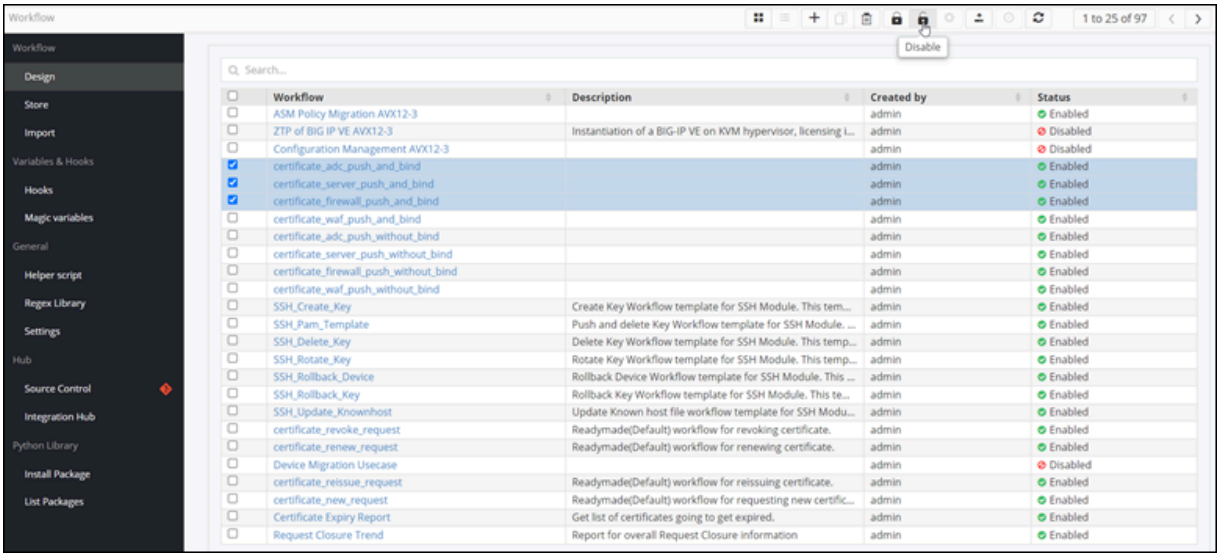
> **Tip:** You can also turn off the toggle on the workflow card to disable the workflow.

4. To disable multiple workflows, you can either select the workflows to be disabled individually or click ☑ to select all workflows displayed on the page and click **Disable** in the **Actions** dropdown menu.
5. To disable a workflow when it is in design mode, from the upper right corner of the screen, click **OFF**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



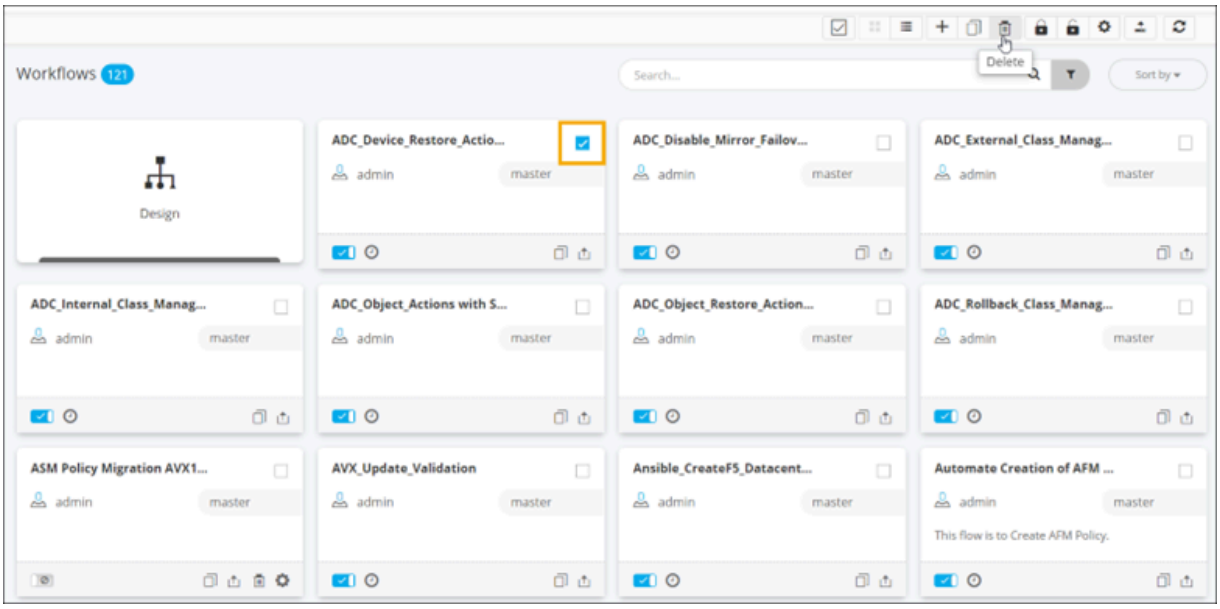
6. To disable a workflow when the **Workflow** inventory page is in **List view**, select the workflow(s) and click .



Deleting a Workflow

1. On the **Workflow** inventory page (**Card view**), disable the workflow to be deleted.
2. Select the workflow.
3. To delete the workflow, from the command bar, click .

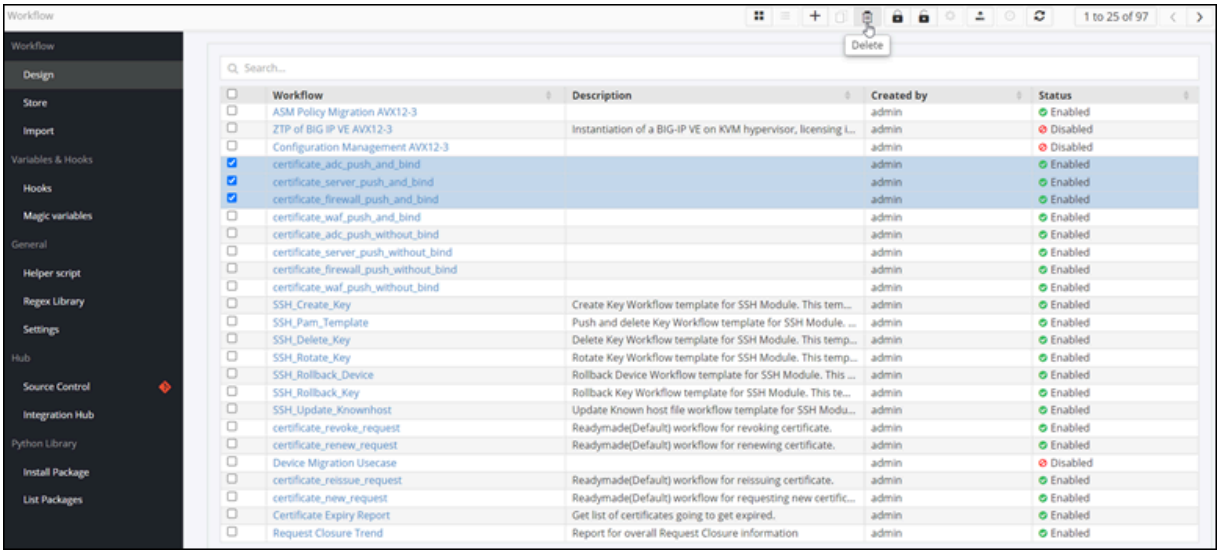
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



4. To delete multiple workflows, either select them individually or click to select all workflows displayed on the page and click from the command bar.

> **Note:** Ensure that the workflows to be deleted are disabled.

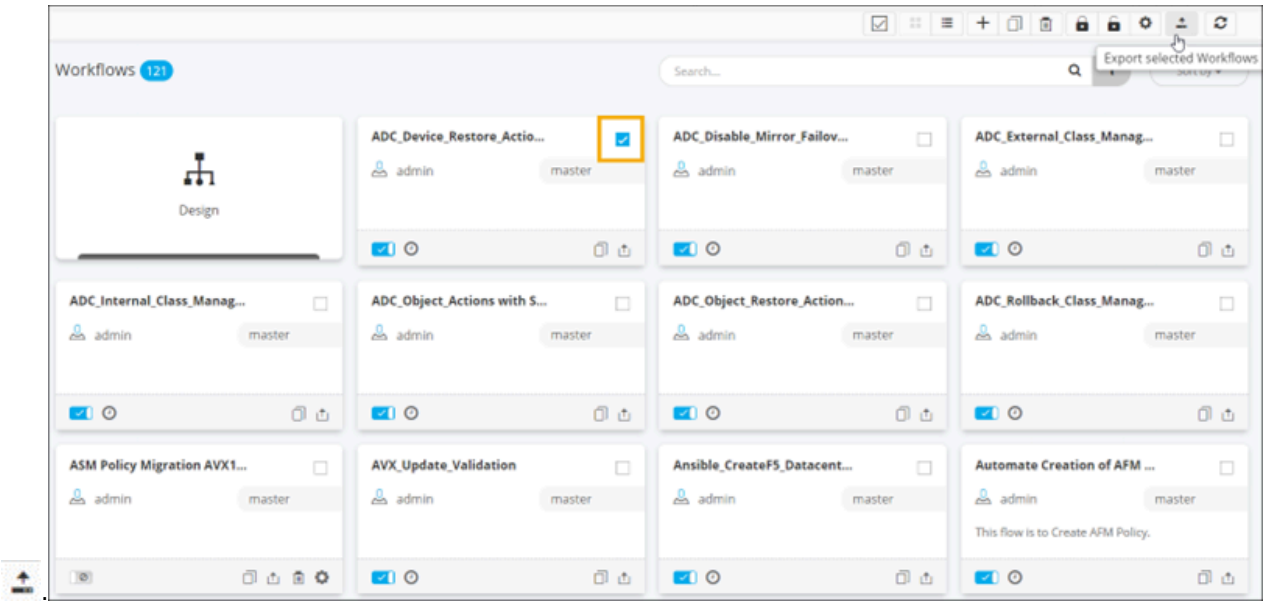
5. To delete a workflow when the **Workflow** inventory is in **List view**, select the workflow(s) to be deleted and click .



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

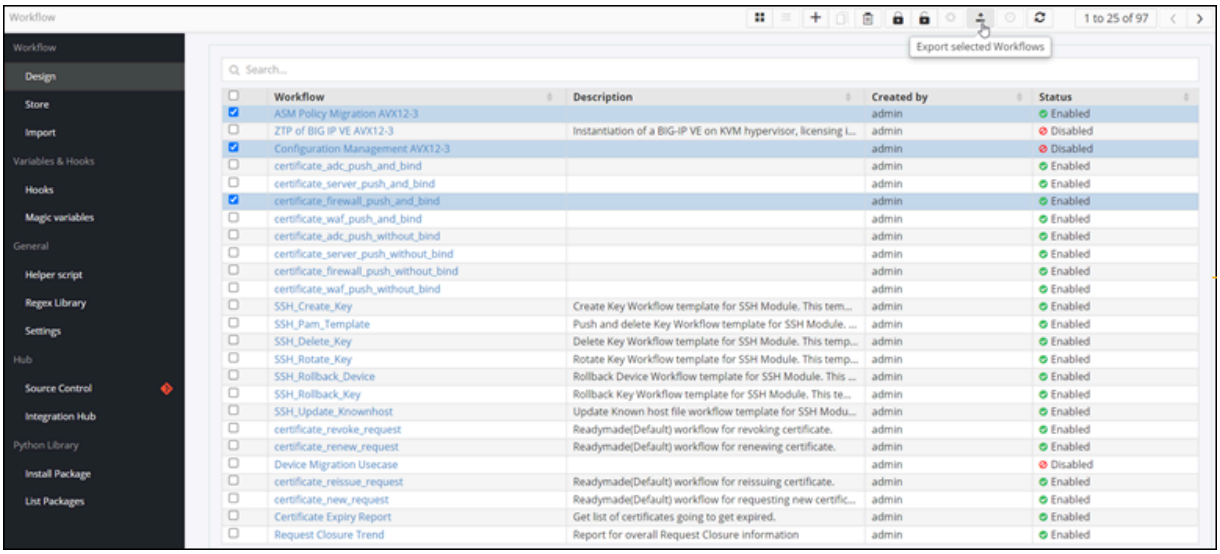
> **Tip:** Ensure that the workflow(s) to be deleted are disabled.

Exporting a Workflow

1. On the **Workflow** inventory page (**Card view**), select the workflow(s) to be exported.
2. To export the workflow, from the command bar, click 

3. Click **Ok** in the **Export** window.
   The selected workflow is saved to your machine as a .zip file.
4. To export a flow, you can also click on the workflow card.
5. To export multiple workflows, you can either select them individually or click to select all workflows displayed on the page and click from the command bar.
6. To export a workflow, when the **Workflow** inventory is in **List view**, select the workflow(s) to be enabled and click .

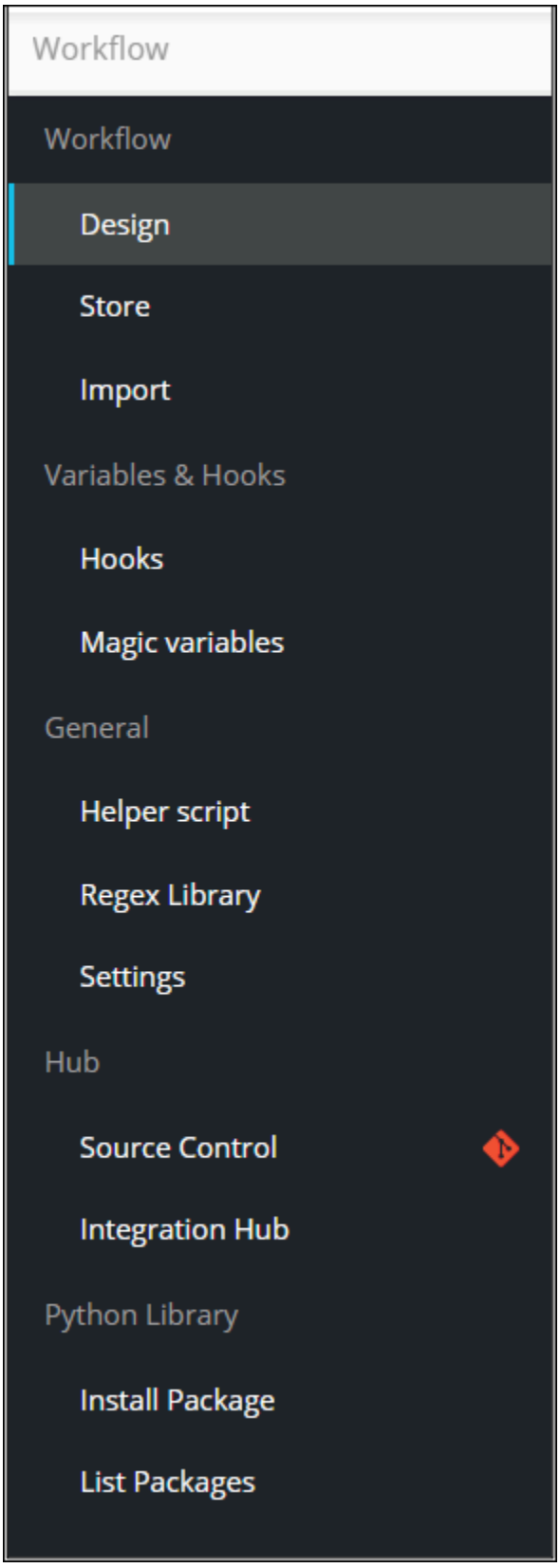
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Workflow Inventory Menu

The grouping in the left menu allows you to perform a variety of actions.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

The following actions can be performed from the Workflow Inventory menu:

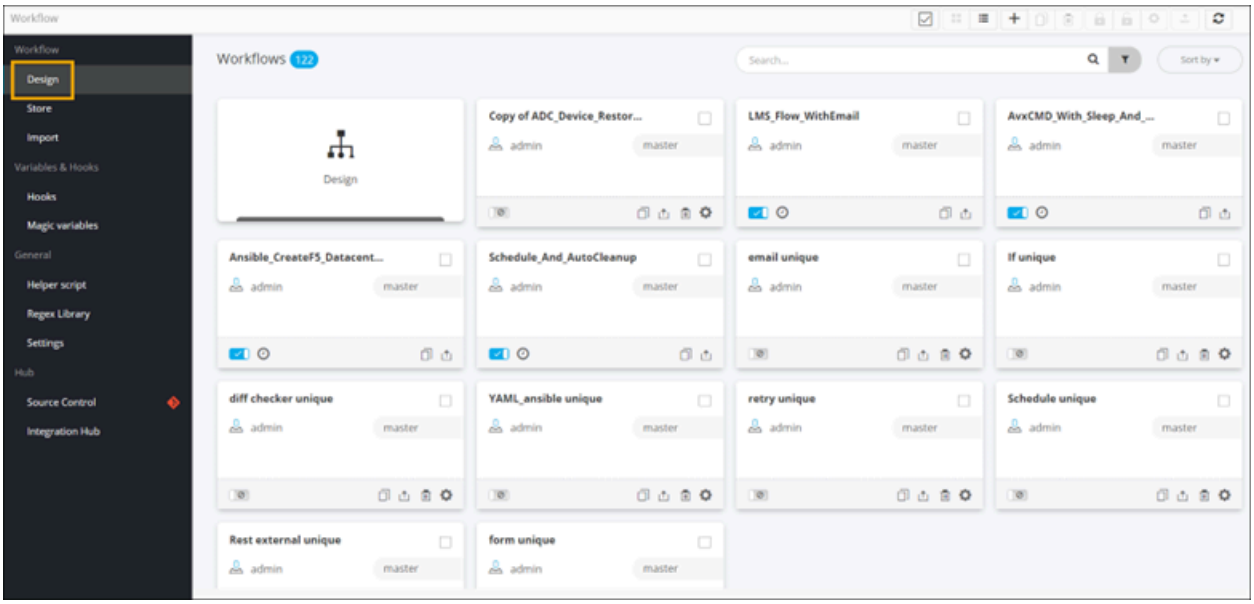
Group	Action	Description
Workflow		
	Design	Allows you to design an automation workflow.
	Store	Displays an inventory of all out of the box (OOB) workflows.
	Import	Allows you to import a workflow.
Variables & Hooks		
	Hooks	Displays the Hooks Inventory and allows you to create, delete, import and export hooks.
	Magic Variables	Allows you to define hooks as magic variables and use them within a workflow.
General		
	Helper Script	Displays a list of all available Helper Scripts and allows you to define and refer helper scripts within workflows.
	Regex Library	Displays a list of regular expressions and allows you to define custom regex validation that can be referenced against a specific form field.
	Settings	Allows you to set Role Based Access Control (RBAC) for accessing and executing workflows.
Hub		
	Source Control	Allows you to connect to AppViewX Git repository and download tasks, workflows.
	Integration Hub	Allows you to integrate workflows with an ITSM system.
Python Library		
	Install Package	Allows you to install a Python package from a third party repository.
	List Packages	Allows you to view the list of Python packages available within AppViewX.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Design
- Store
- Import
- Hooks
- Magic Variables
- Helper Script
- Regex Library
- Source Control
- Integration Hub
- Python Library

Design

This feature allows you to create a new workflow or modify an existing flow. The canvas is a placeholder to drag and drop tasks, to define custom logic, and stitch together an automation workflow.



The following options are available when designing a new workflow:

- Provision to design a new workflow and assign a category and a sub category.
- Provision to search and view the inventory of tasks by category
- Provision to drag and drop in-built, custom tasks to design a workflow
- Provision to clone existing tasks and use them in a workflow
- Provision to delete tasks

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

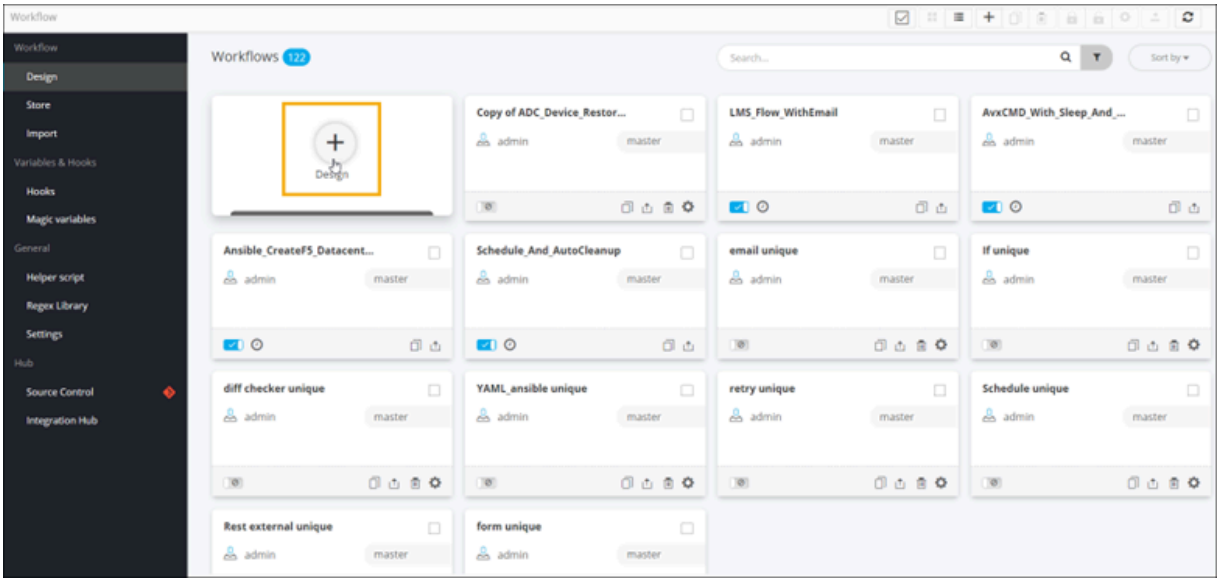
- Provision to link and update existing links and connect tasks within a workflow
- Provision to validate a workflow for errors
- Provision to assign a workflow to user role(s)
- Provision to enable or disable a workflow
- Provision to bookmark workflow task(s)
- Provision to preview a workflow task
- Provision to preview a workflow/nested workflow
- Provision to reuse and/or import workflow and workflow task(s)

- Designing/Modifying a Workflow

Designing/Modifying a Workflow

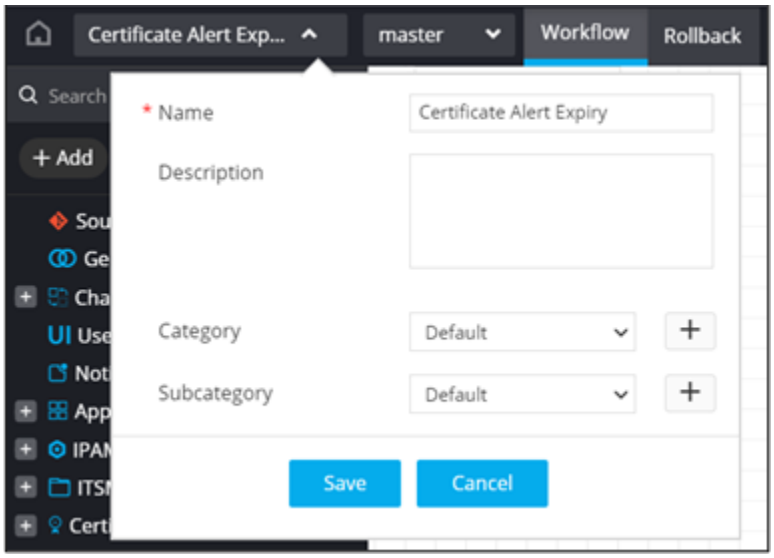
To design a new workflow:

1. On the **Workflow** inventory page, click **Design**.



2. Enter the workflow **Name** and **Description**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



3. Select the **Category** and **Subcategory**.

> **Note:** These fields are not mandatory.

4. Click **Save**.
5. Drag and drop tasks from the folders in the menu on the left to design workflows from scratch.

> **Note:** For more information on the Workflow Tasks, click here. For more information on getting started quickly with building workflows, click here.

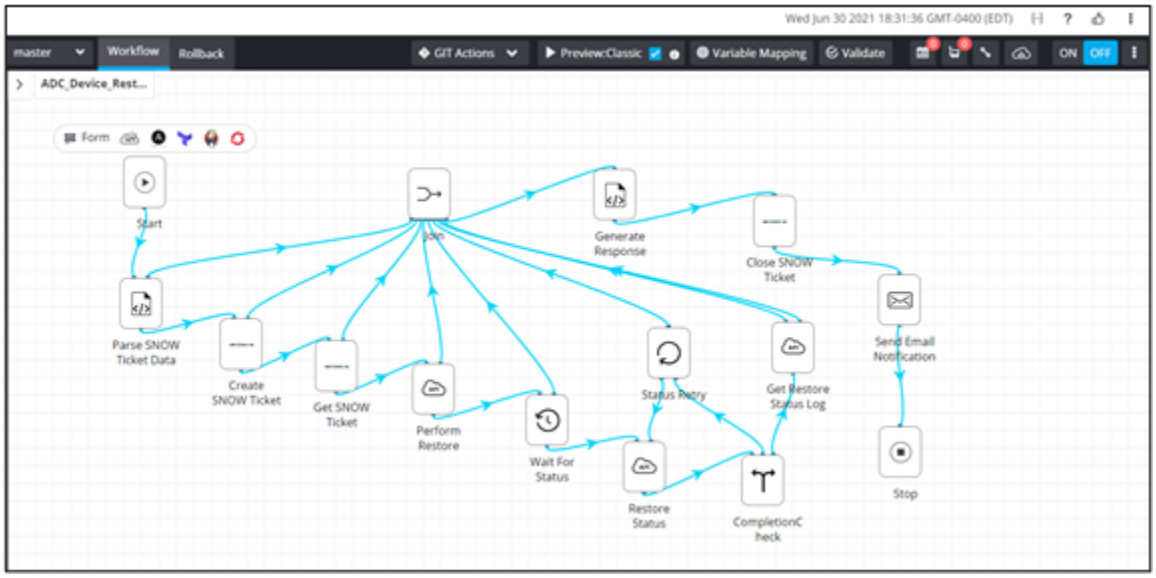
6. To modify a workflow, first ensure that the workflow is disabled.

> **Note:** For more information on disabling a workflow, click here.

7. Click on the workflow to modify it.

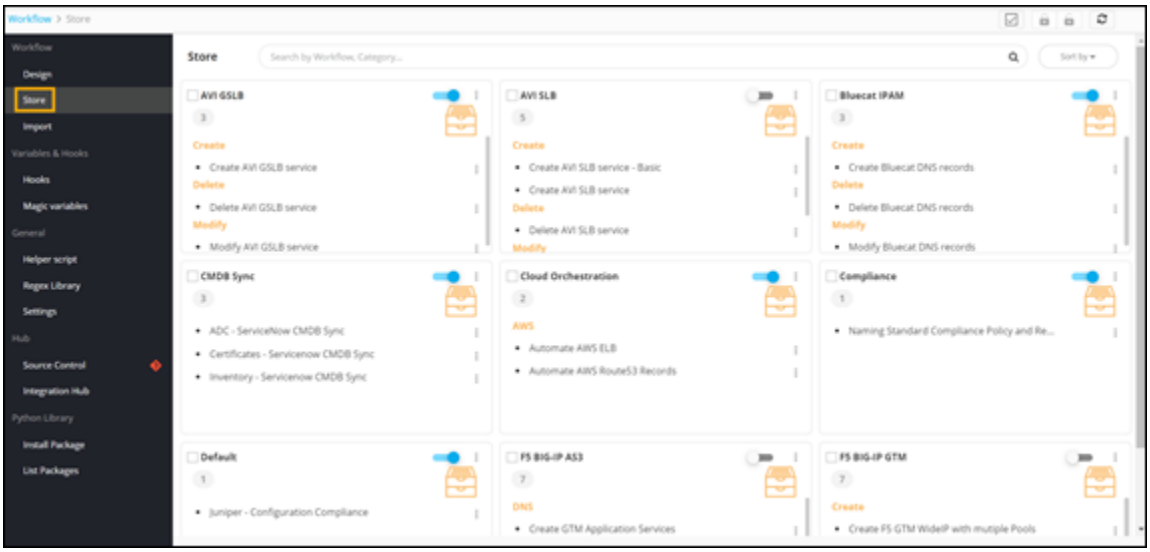
   The workflow opens in the design workspace. You can now modify workflow tasks as per your requirement.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Store

The store is an inventory of all out of the box (OOB) workflows made available to users. All OOB workflows are presented in a catalog view with related workflows grouped together.



The following options are available in the **Store**:

- Provision to enable or disable an entire workflow catalog.
- Provision to enable or disable individual workflow(s).
- Provision to sort the workflow catalogs by ascending or descending order.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

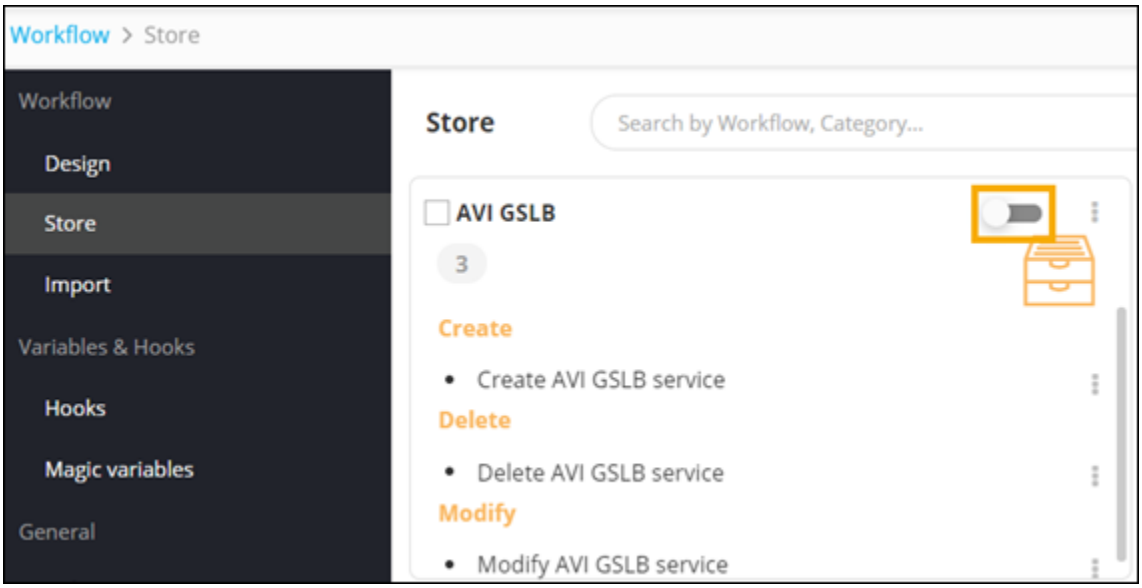
- Provision to clone an OOB workflow.
- Provision to schedule a workflow from the Store.
- Provision to define RBAC settings for different tasks within a workflow.

- Enabling/Disabling OOB Workflow Catalogs
- Enabling/Disabling individual OOB Workflows
- Cloning an OOB Workflow
- Scheduling an OOB Workflow
- Configuring RBAC for OOB Workflows

Enabling/Disabling OOB Workflow Catalogs

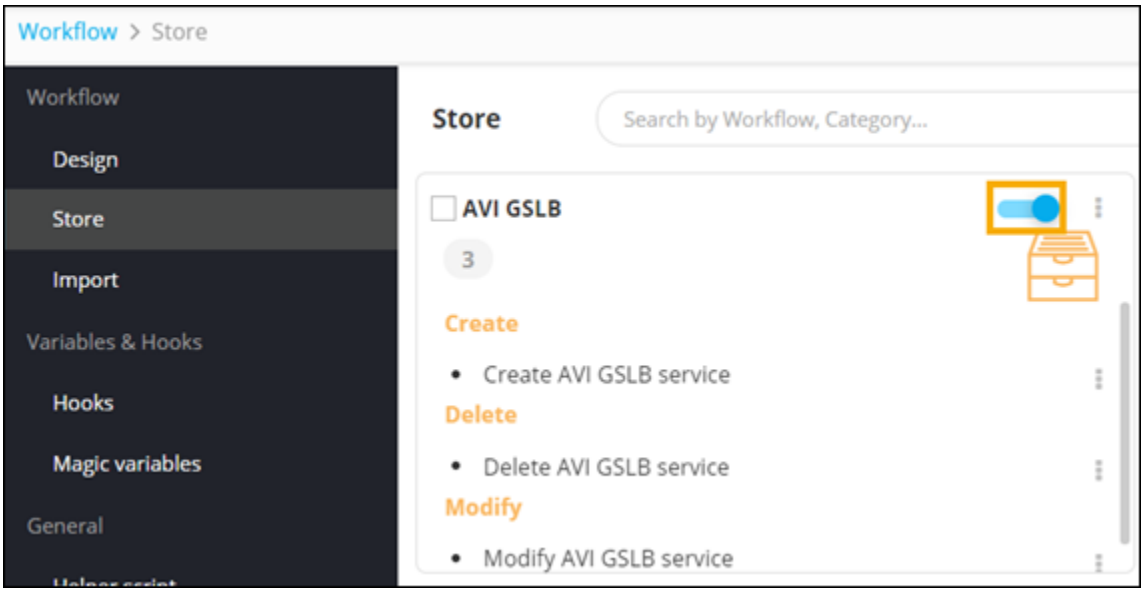
This feature allows you to enable/disable all the workflows in a given catalog. When you disable a workflow catalog, the workflows present in the catalog will not be available on the Workflow Request page and therefore cannot be scheduled or triggered.

1. To disable a workflow catalog, on the **Workflow > Store** page. turn off the toggle for the catalog to be disabled.



2. To enable a workflow catalog, on the **Workflow > Store** page, turn on the toggle for the catalog to be enabled.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

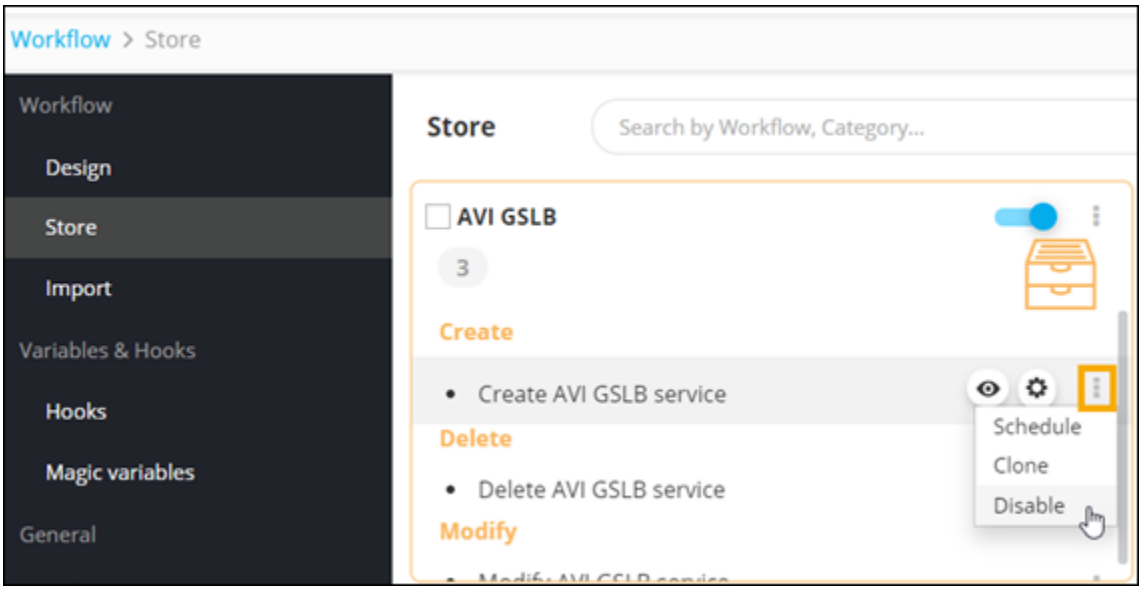


Enabling/Disabling individual OOB Workflows

You can also enable or disable individual workflow from the Store. When you disable a workflow, it will not be available on the Workflow Request page and therefore cannot be scheduled or triggered.

To disable an OOB workflow:

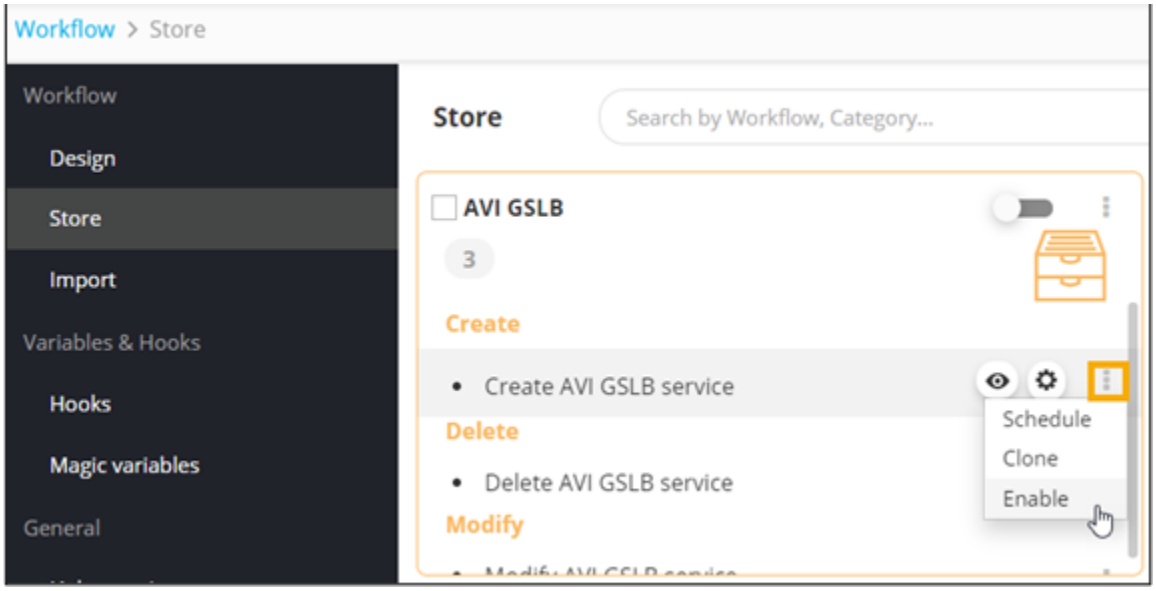
1. On the **Workflow > Store** page, hover your mouse over a workflow name and click ⋮.
2. From the available options, click **Disable**.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

The workflow is disabled.

3. To enable a disabled workflow, on the **Workflow > Store** page, hover your mouse over the workflow to be enabled, and click ⋮.
4. From the options available, click **Enable**.



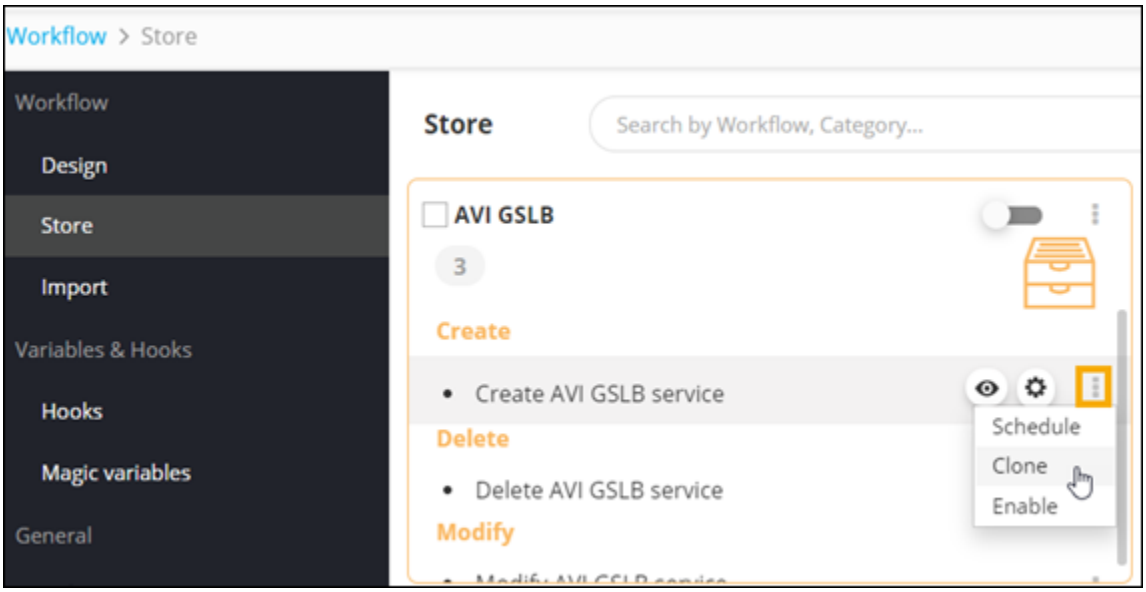
The workflow is enabled.

Cloning an OOB Workflow

To clone an individual OOB workflow inside a workflow catalog:

1. On the **Workflow > Store** page, hover your mouse over the workflow to be cloned, and click ⋮.
2. From the options available, click **Clone**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

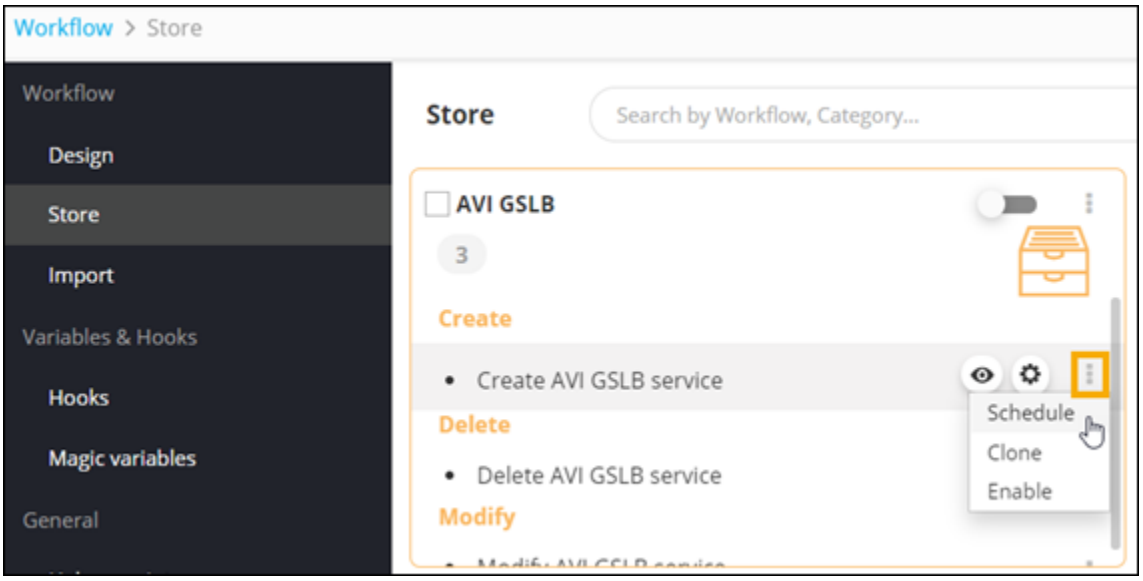


A copy of the cloned workflow is added to the Workflow inventory page in the Design section.

Scheduling an OOB Workflow

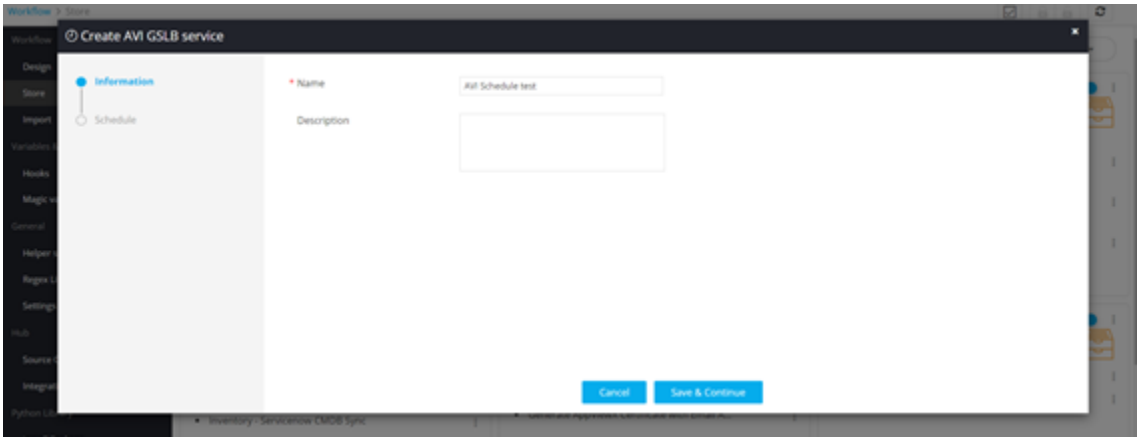
To schedule an OOB workflow from the Store:

1. On the **Workflow** > **Store** page, hover your mouse over the workflow you want to schedule, and click ⋮.
2. From the options available, click **Schedule**.

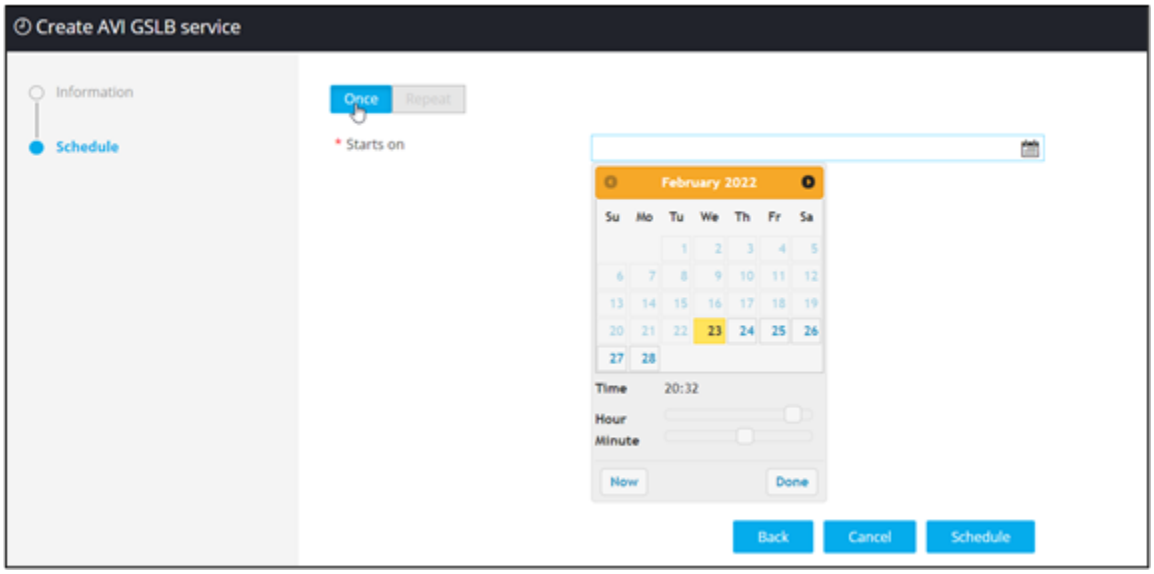


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

3. Enter the information in the scheduler window.

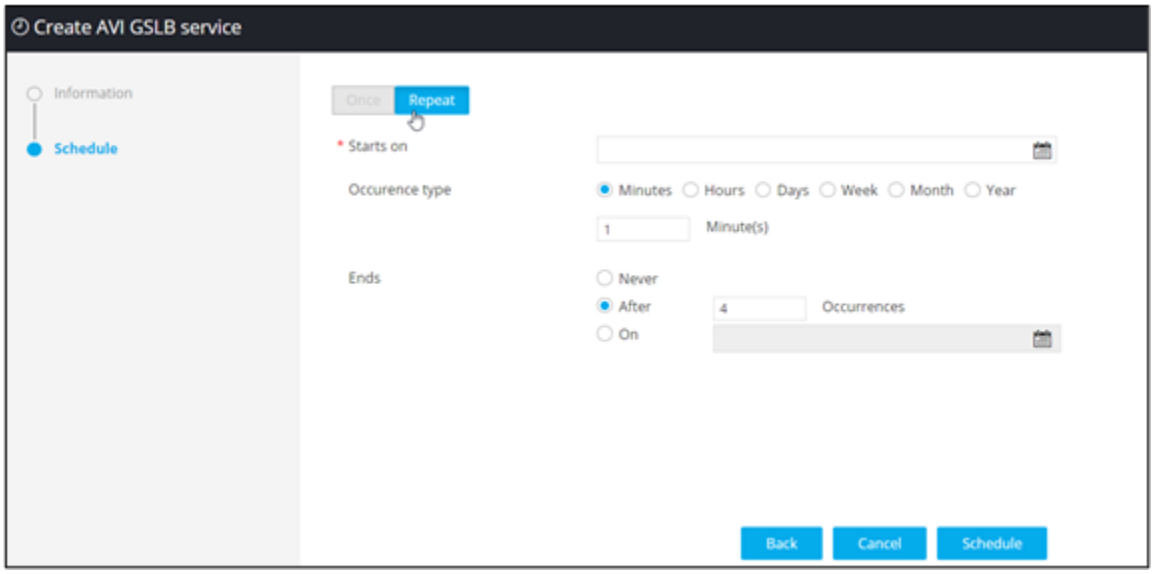


4. Click **Save & Continue**.
5. Assign the time for scheduling the workflow:
   - **Once** - Select this option if you want to schedule the workflow to be triggered only once.



   - **Repeat** - Select this option to schedule the workflow to trigger daily, weekly, monthly, or yearly.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



6. Click **Schedule**.

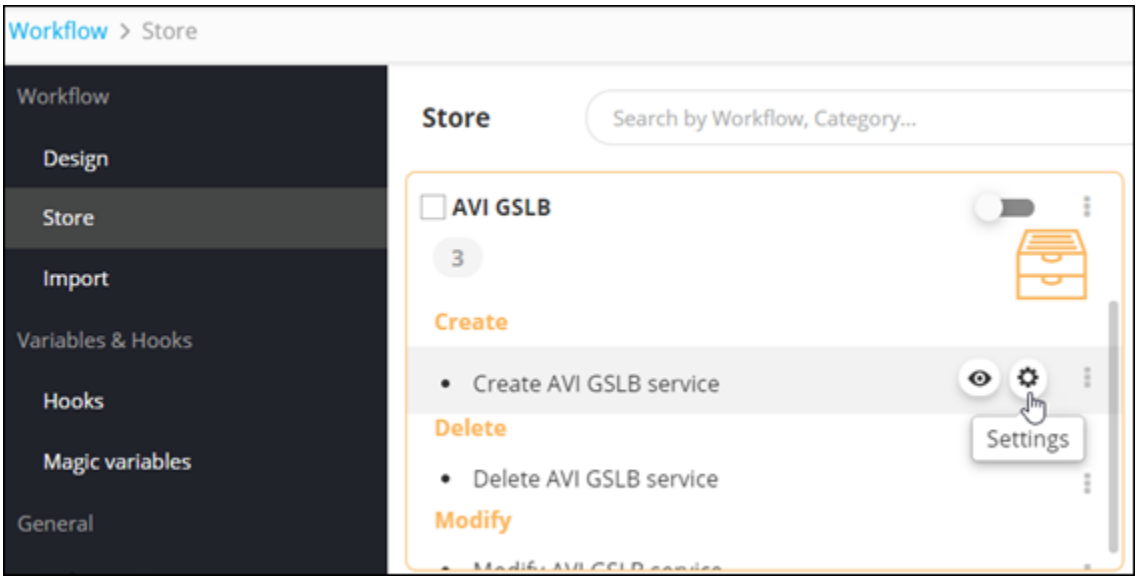
   The scheduled workflow is added to the **Scheduled jobs** on the Request :: Schedule jobs page.

Configuring RBAC for OOB Workflows

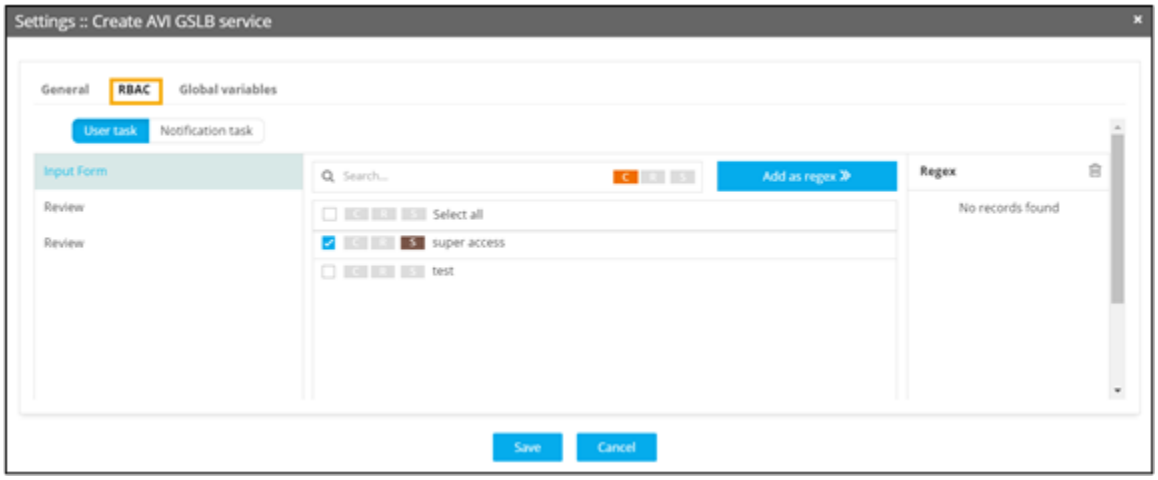
To configure RBAC settings for a particular OOB workflow:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. On the **Workflow > Store** page, hover your mouse over the workflow and click ⚙.



2. In the **Settings** window, under **RBAC**, you can map the access control settings for user(s) for all tasks within the workflow.



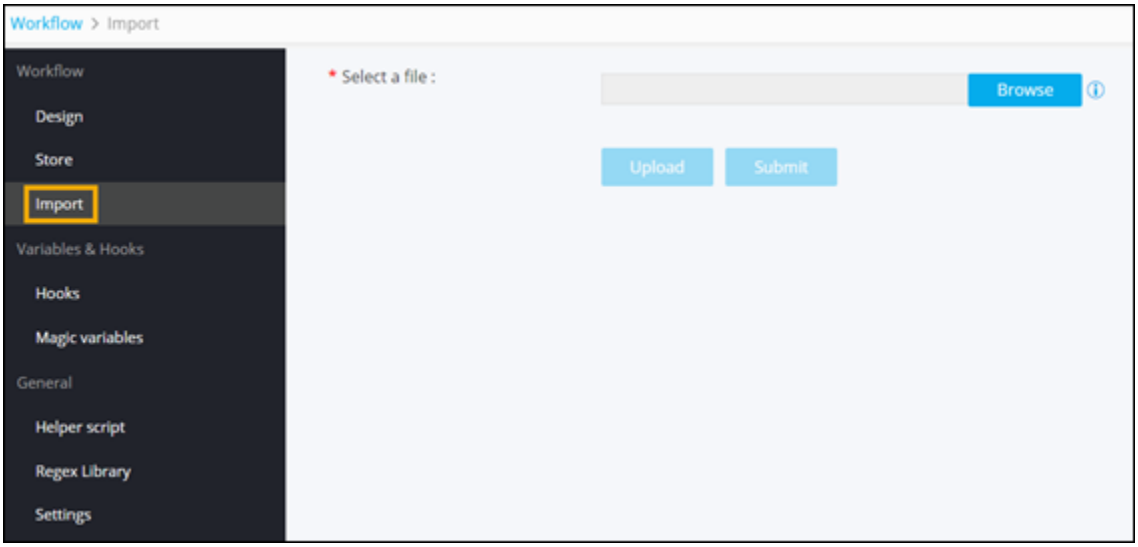
Import

You can import workflow(s) into AppViewX from any environment.

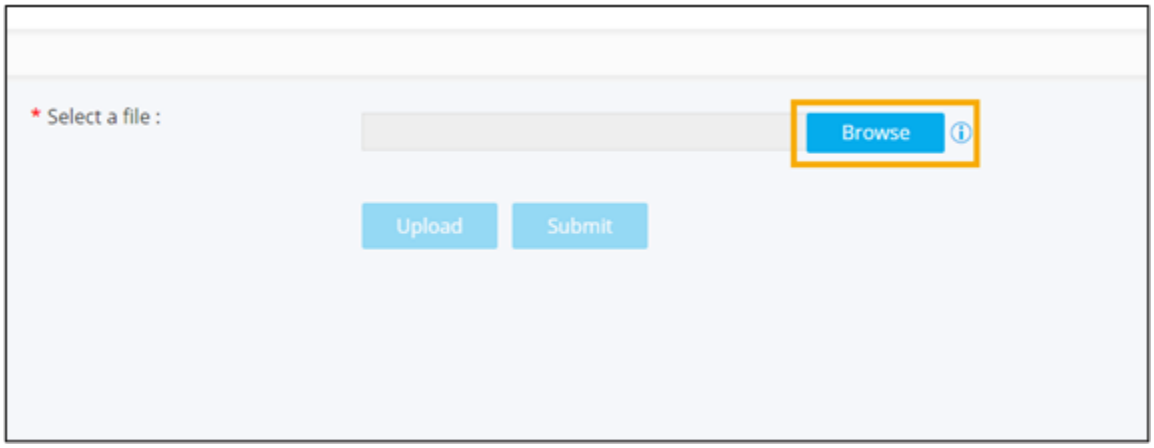
To import a workflow:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

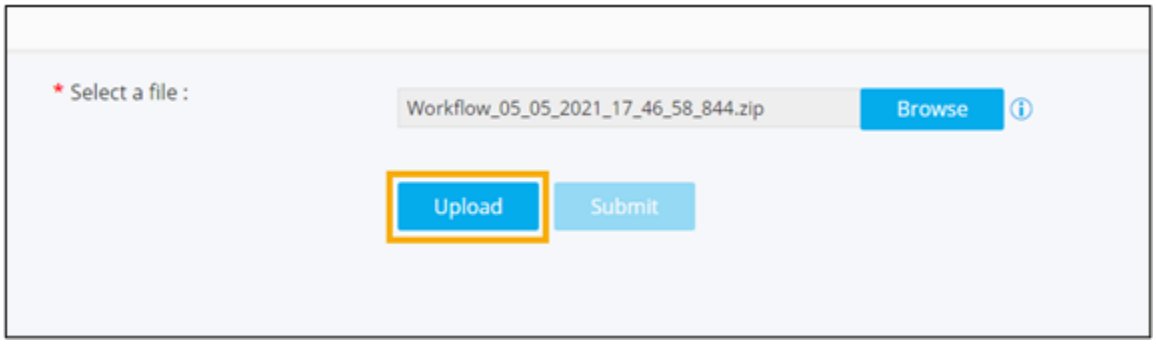
1. On the Workflow Inventory page, from the navigation menu on the left, click **Import**.



2. To select the file to be imported, click **Browse**.

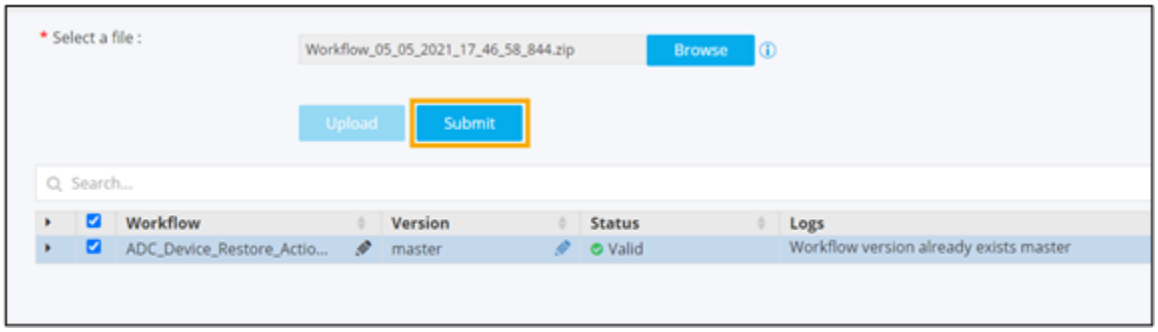


3. To upload the selected .zip file, click **Upload**.

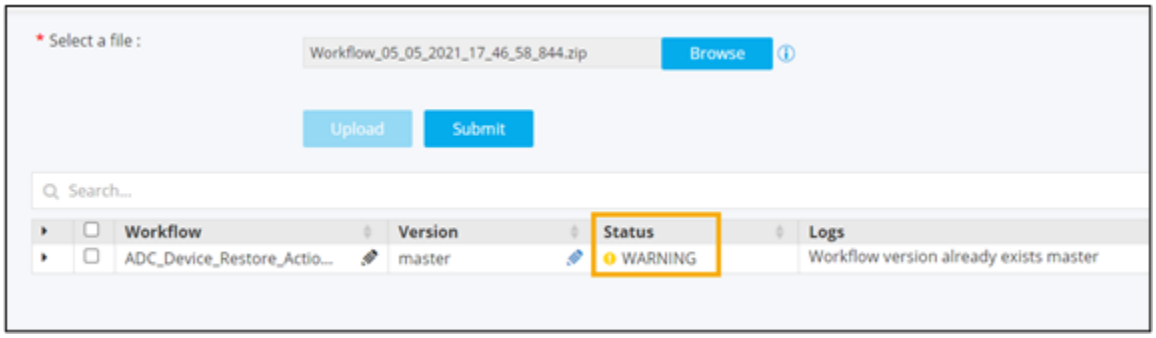


4. Select the workflow and click **Submit**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Warning: If you are importing a workflow that already exists, the workflow status will show a Warning. You need to either edit the workflow name or the version to be able to import this workflow.



Hooks

“Design and Define a Query once, reuse them forever…”

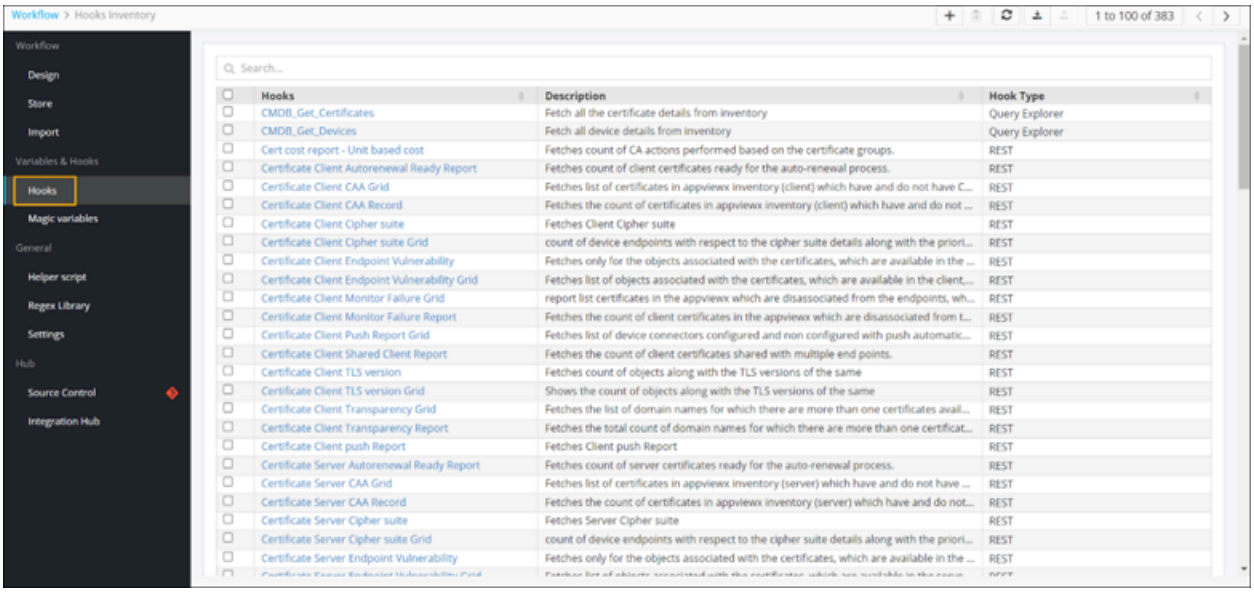
A ‘Hook’ is a mechanism used to query and retrieve data from different sources such as database, device, or external vendors; and leverage them as part of the report building process. The mode of query could be either through – Script, REST API and Query Explorer.

There are 3 ways of querying data through hooks:

- REST API
- Script (Python)
- Query Explorer (GUI based)

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

For example, a query to retrieve a list of managed BIG-IP F5 devices from the database can be defined once and reused across one or more workflow(s).



This table describes the options available on the Hooks Inventory page:

Option	Description
Search bar	Allows you to search for a particular hook by typing in keyword(s).
+	Allows you to create a new hook.
🗑	Allows you to delete hook(s).
↻	Refreshes the pages.
⬇	Allows you to import a hook.
⬆	Allows you to export hook(s).

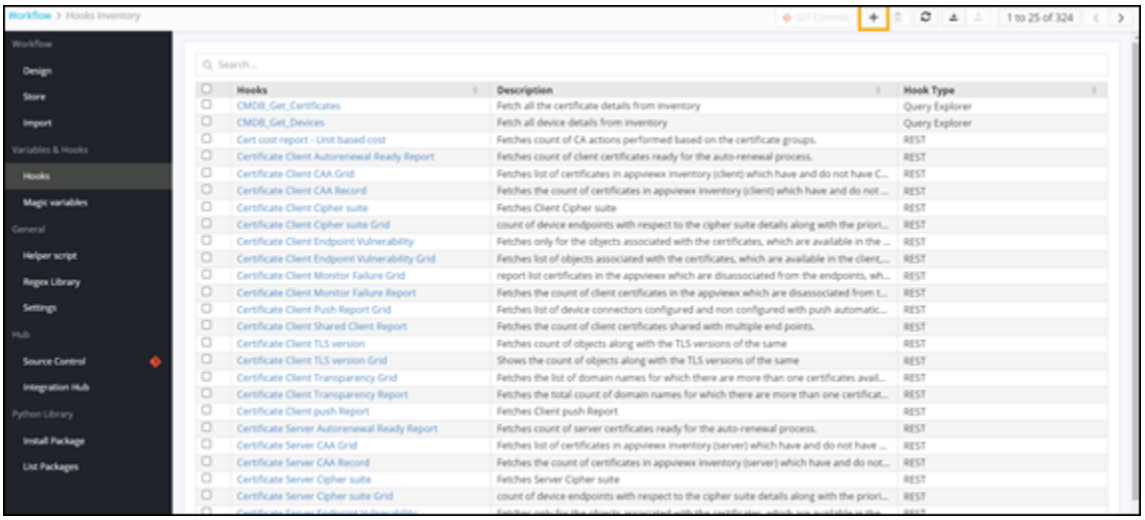
- Defining a Hook and using it within a Workflow

Defining a Hook and using it within a Workflow

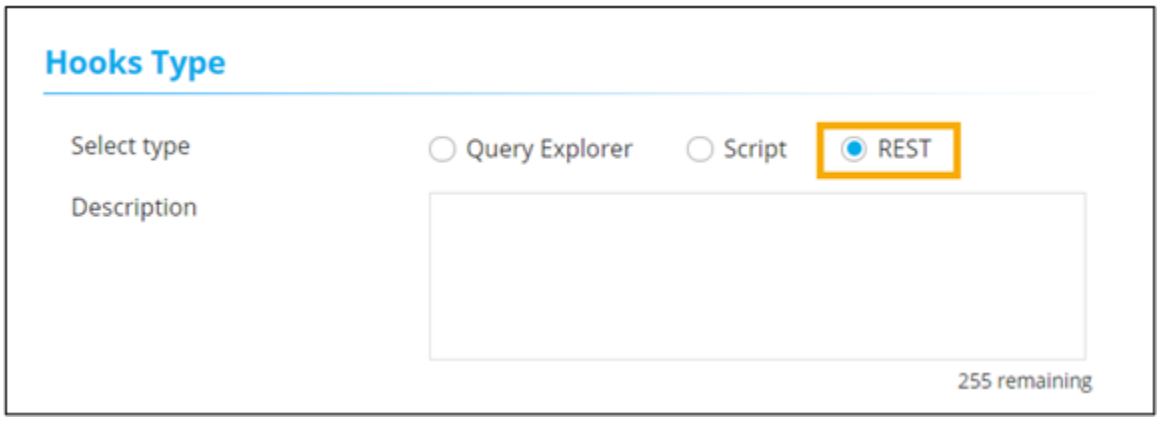
To define a common hook to create an incident ticket on ServiceNow and use it within a workflow:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

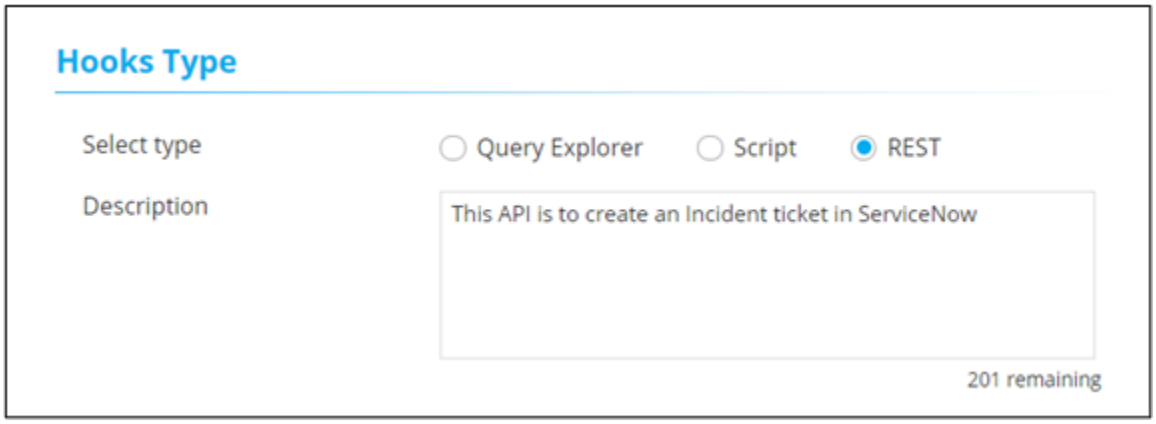
1. On the Workflow Inventory page, from the navigation pane on the left, select **Hooks**.
2. To create a hook, click + in the command bar.



3. Select **Hooks Type** as **REST**.

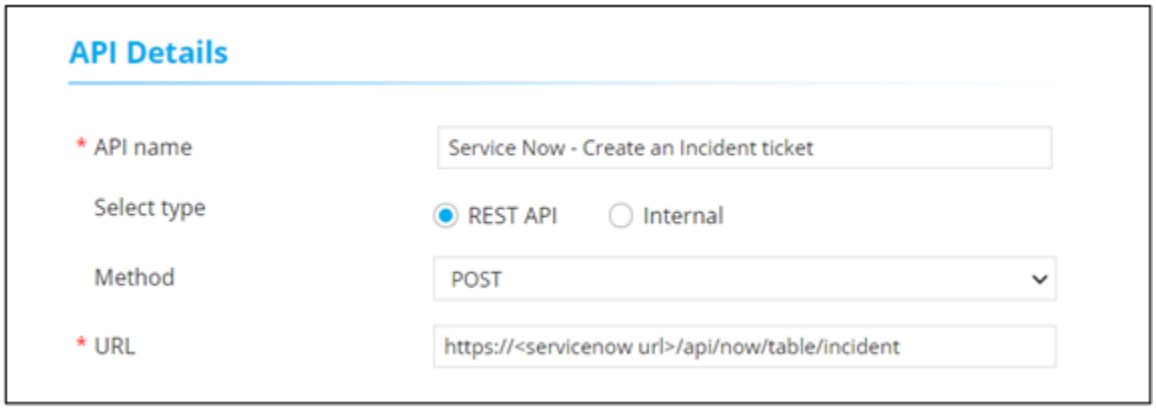


4. Enter a valid description for the hook.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

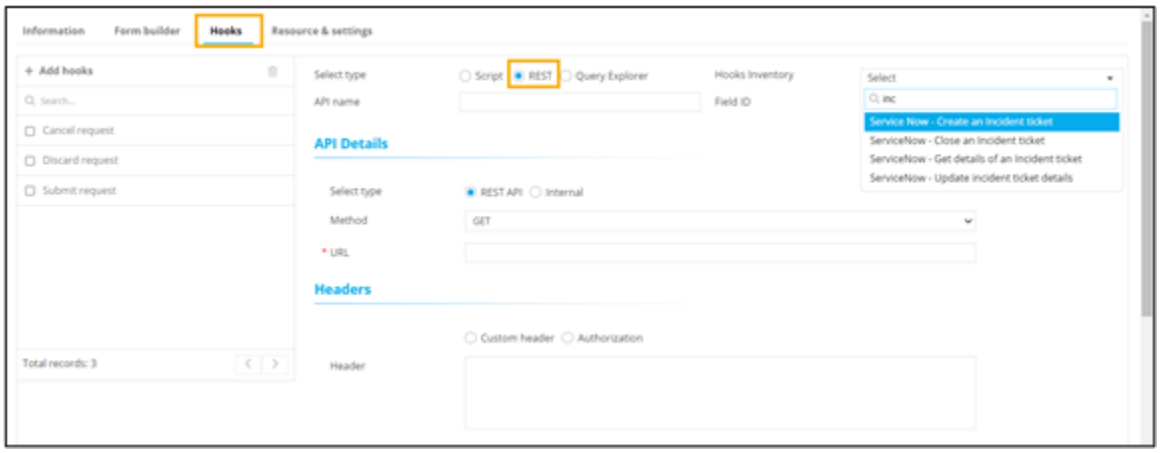
5. Enter or select the field information in the **API Details** section.



6. Click **Save**.
7. Design a workflow.
8. From the **User Interface** section, drag and drop a **Form** task.
9. Define the necessary form fields.

> **Note:** For more information on adding form fields, click here.

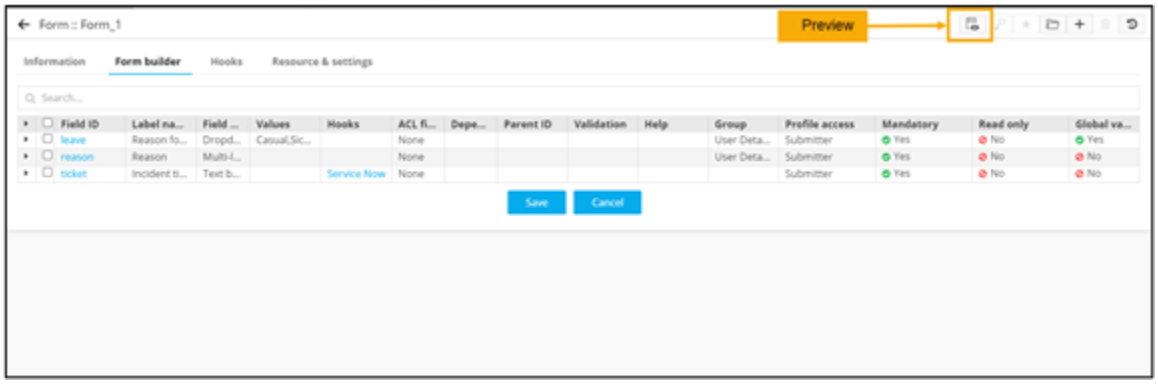
10. To display the Hooks inventory, under the **Hooks** tab, select the **Hook type** as **REST**.



11. Select the **Field ID** against which the hook should be mapped.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

12. Click **Add**.
13. Under the **Form Builder** tab, click .



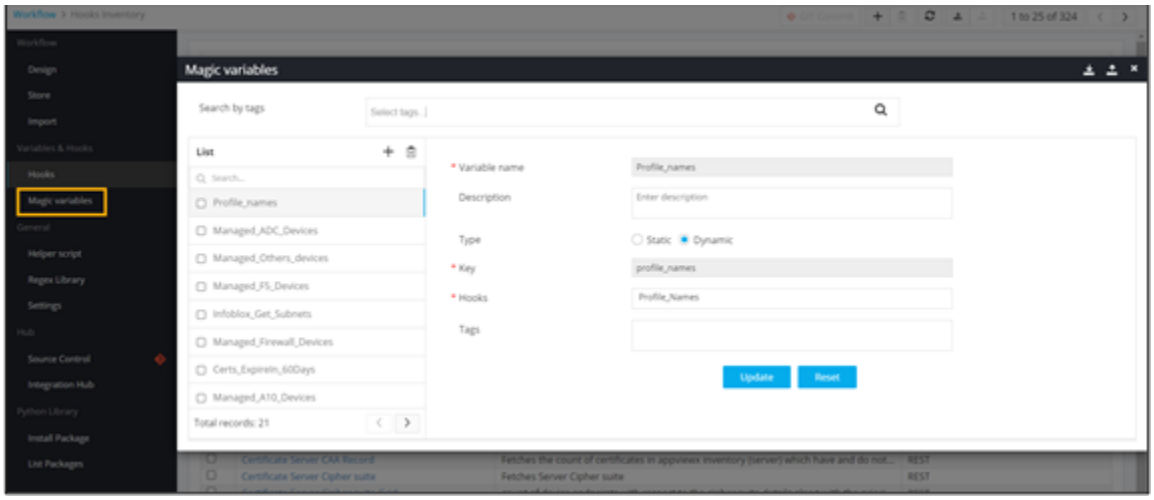
A preview of the form shows the hook added to the form field.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Magic Variables

“Define a hook once, refer to it as a magic variable; and reuse it forever…” Magic variable is a mechanism that allows for a custom or pre-defined hook to be used as a ‘variable’ anywhere within the workflow automation process. Once a hook is defined and is declared as a magic variable, it allows for reusability within forms across the workflow.

- Provision to define a new magic variable.
- Provision to modify and delete a magic variable.
- Provision to use Static and Dynamic Hook types as magic variables:
  - **Static**: Hooks which are static in nature can be referenced as magic variables. For example, Slack webhook, ITSM change ticket URL etc.
  - **Dynamic**: Hooks that execute dynamically can be referenced as magic variables. For example, Query to get list of devices, Query to get list of unused VIPs, Query to get list of managed F5 devices etc.
- Provision to tag magic variables and search by tag names.
- Provision to import and export hooks and variables from one environment to another.



Syntax to use magic variable: `(~<magic_variable>~)`

For example: `(~get_f5_devices~)`.

Tip: You can type keywords to auto fill the variables in the Form task.

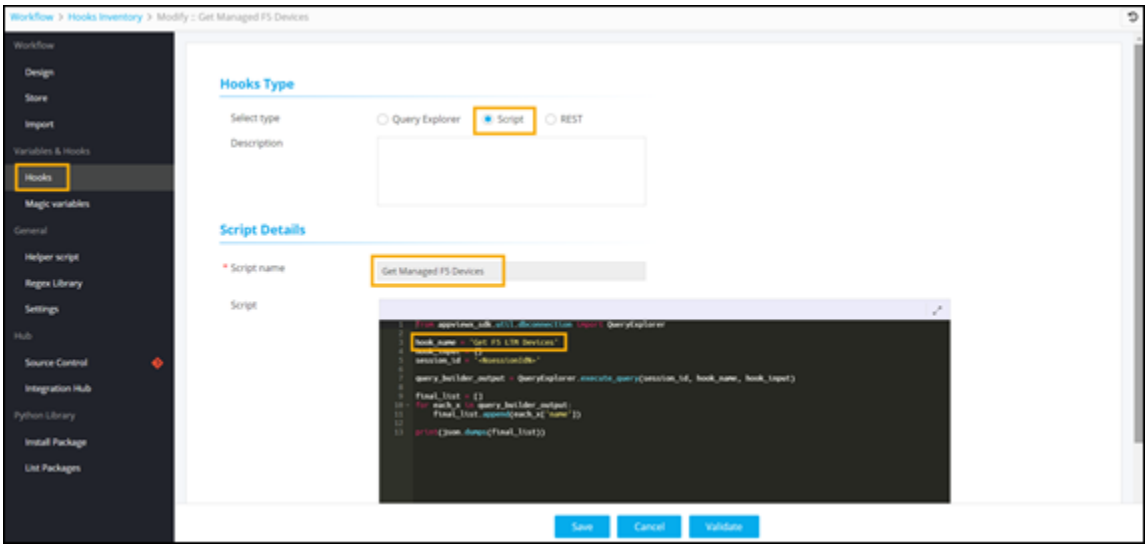
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Defining a Hook and declaring it as a Magic Variable
- Using a Magic Variable within a Workflow

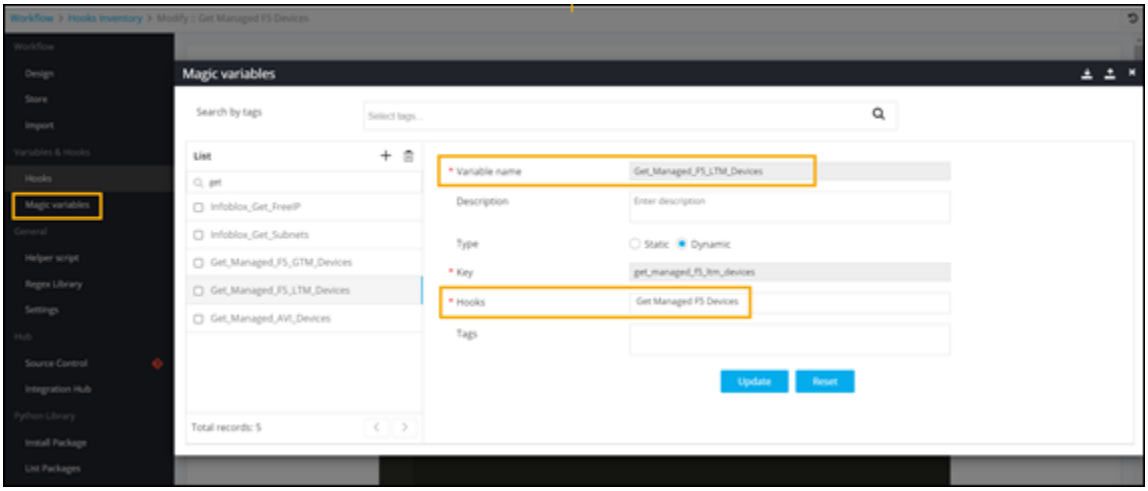
Defining a Hook and declaring it as a Magic Variable

You can define a common hook and refer it within a workflow form.

1. On the Workflow Inventory page, from the left navigation pane, select **Hooks**.
2. Create a Hook using REST API or Script.



3. Click **Save**.
4. From the navigation pane on the left, click **Magic Variables**.
5. Enter or select the field information.



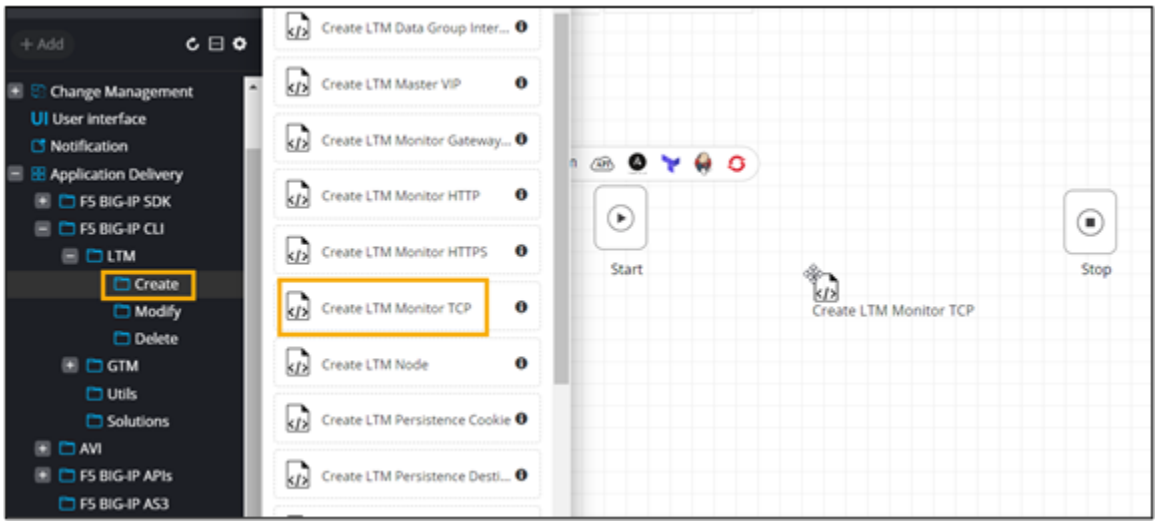
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

6. Click **Add**.

Using a Magic Variable within a Workflow

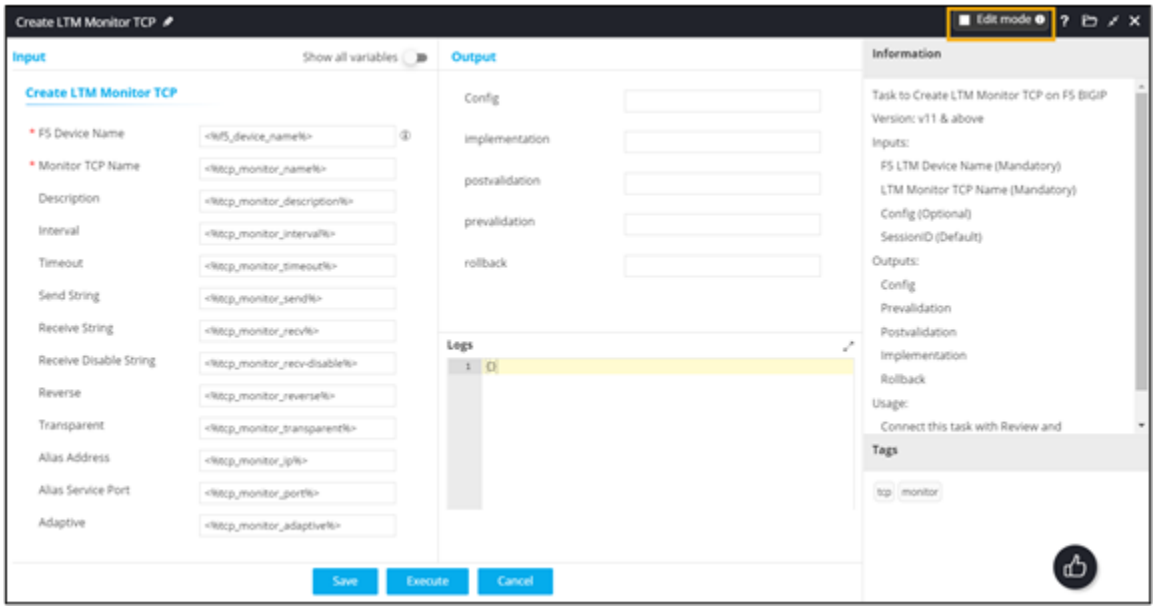
To define a hook as a magic variable and use it in the Form task:

1. Design a new workflow.
2. From the menu on the left, navigate to **Application Delivery** > **F5 BIG-IP CLI** > **LTM** > **Create**.
3. From the **Create** folder, drag and drop the **Create LTM Monitor TCP** task.



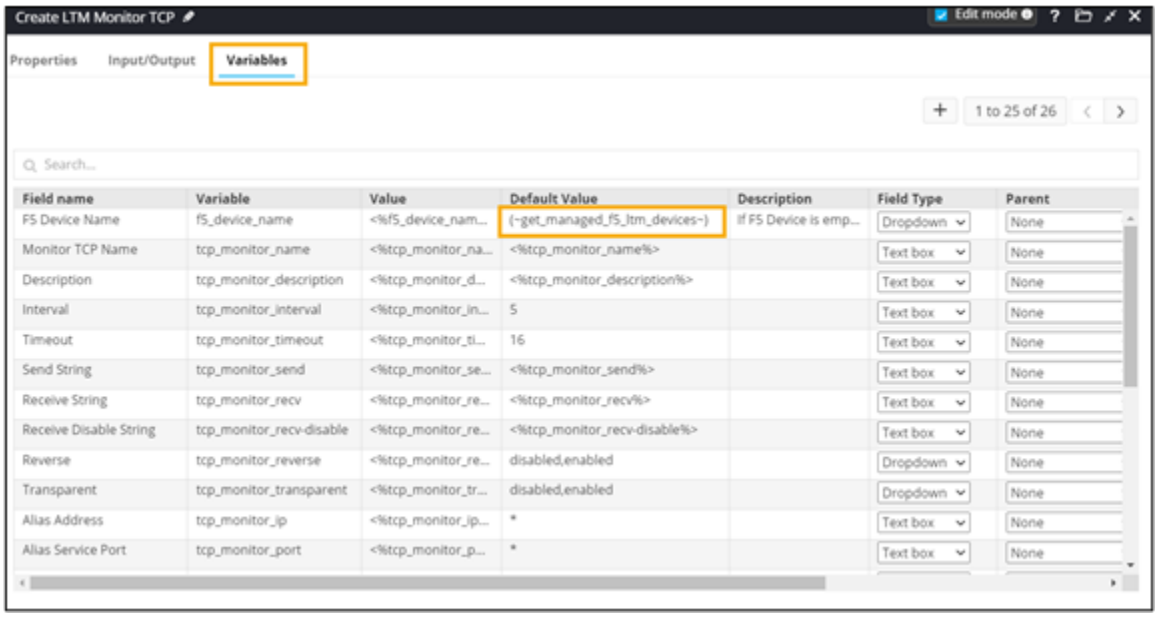
4. In the **Create LTM Monitor TCP** task window, select the **Edit mode** checkbox to see information on the variables used in this task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



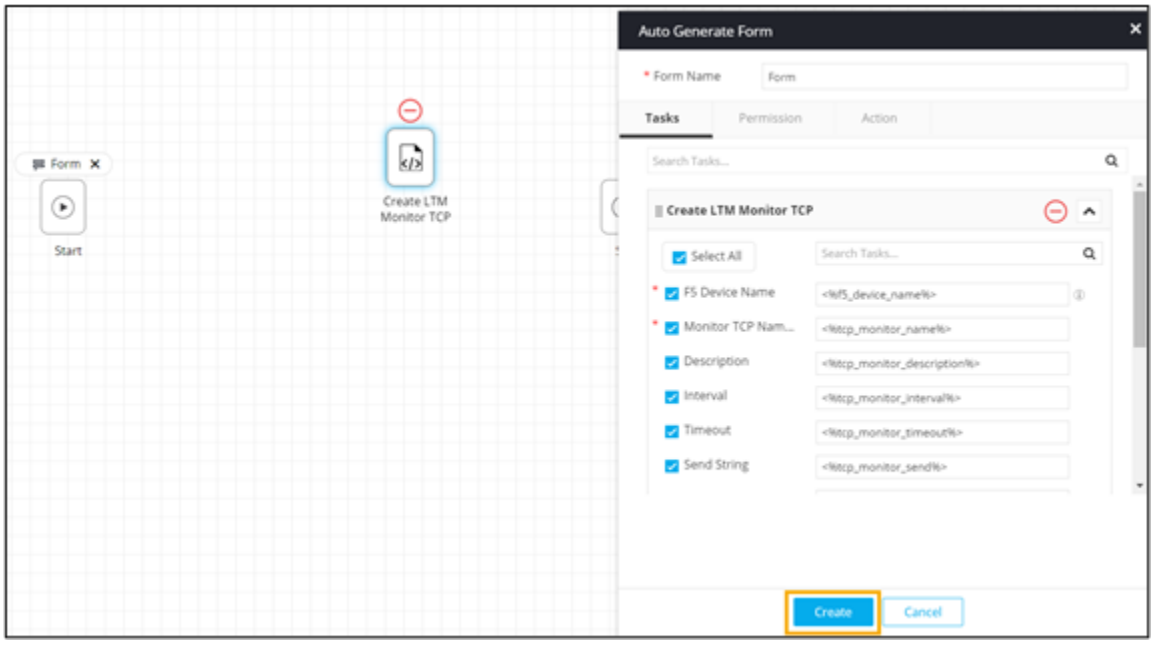
5. To see the magic variable used in this task, in the **Create LTM Monitor TCP** task window, click **Variables**.

   The magic variable (~get_managed_f5_ltm_devices~) retrieves the list of all managed F5 LTM devices.



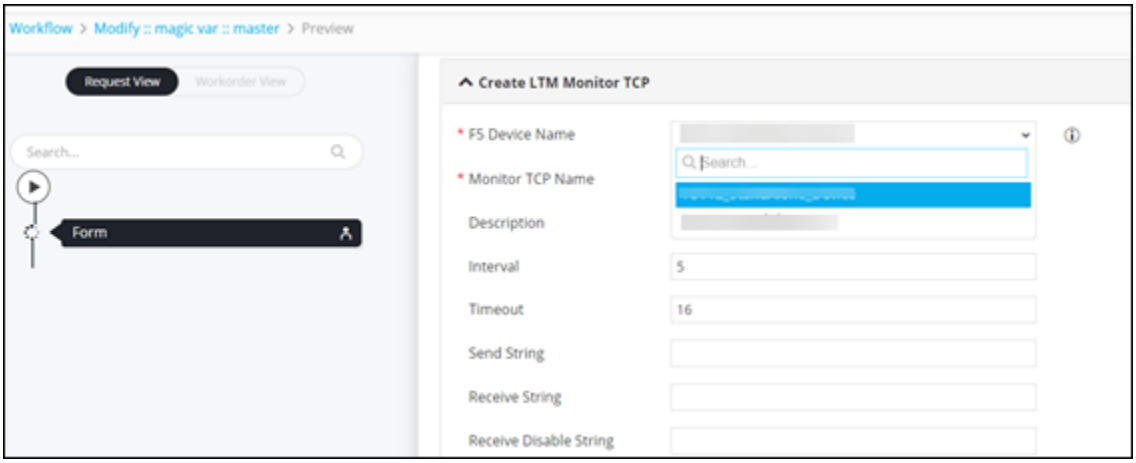
6. To auto generate a form for this task, click **Form** above the **Start** task.
7. To auto-populate the form fields, click ⊕ above the **Create LTM Monitor TCP** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



8. Select the form fields and click **Create**.
9. Connect the workflow and click **Preview**.

   The magic variable auto-populates the **F5 Device Name** form field.



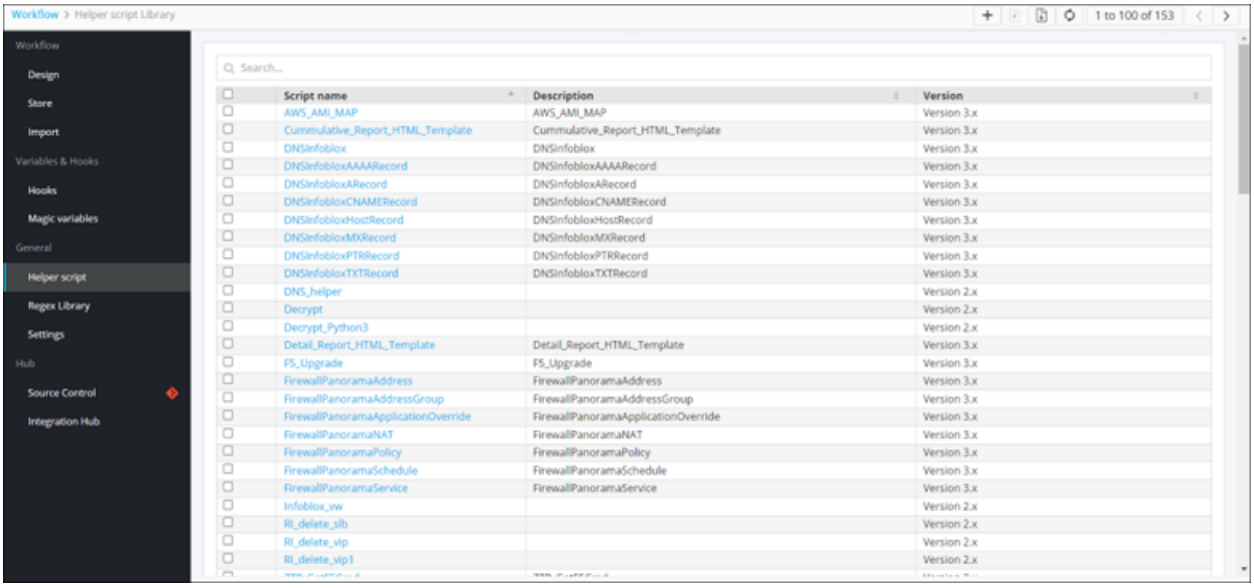
Helper Script

“Write once, reuse them forever…”

Helper scripts can be defined once with the relevant logic and can be reused across any workflow scripts with the relevant function name. One python script is essentially reused within another.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Frequently used methods and functions can be written as helper scripts once and reused across other custom scripts within the workflow. Either default helper script(s) can be leveraged and/or new helpers can be defined. For example, logic to establish database connection can be defined once and reused for any other database related queries via script.



The following table describes the options available on the Helper Script Library page:

Option	Description
Search bar	Allows you to search for a particular helper script by typing in keyword(s).
+	Allows you to create a new helper script.
	Allows you to export a helper script.
	Allows you to import a helper script.
	Refreshes the pages.

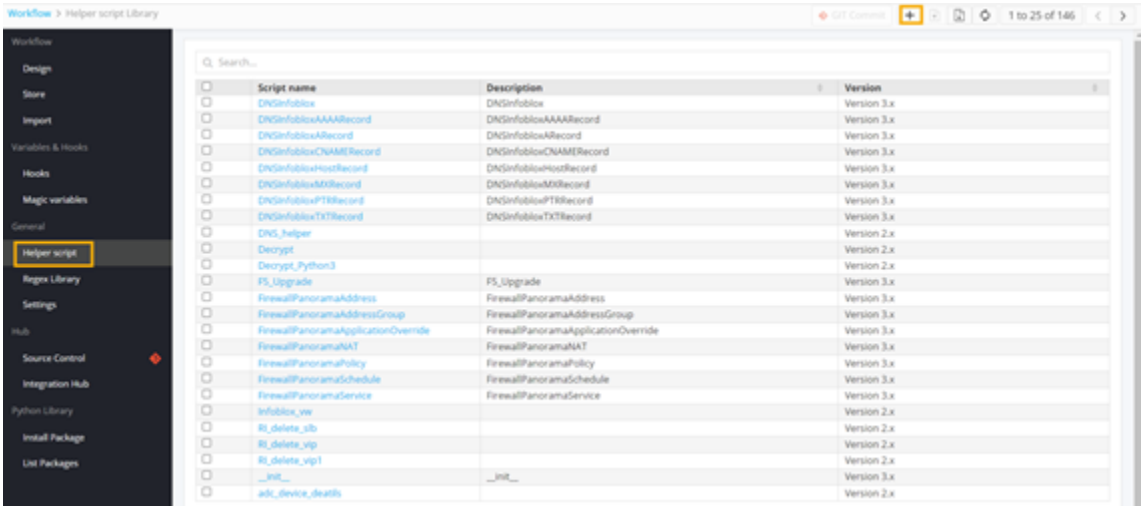
- Creating a Helper Script
- Exporting Helper Scripts
- Importing Helper Scripts

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Configuring RBAC for Helper Scripts
- Using a Default Helper Script

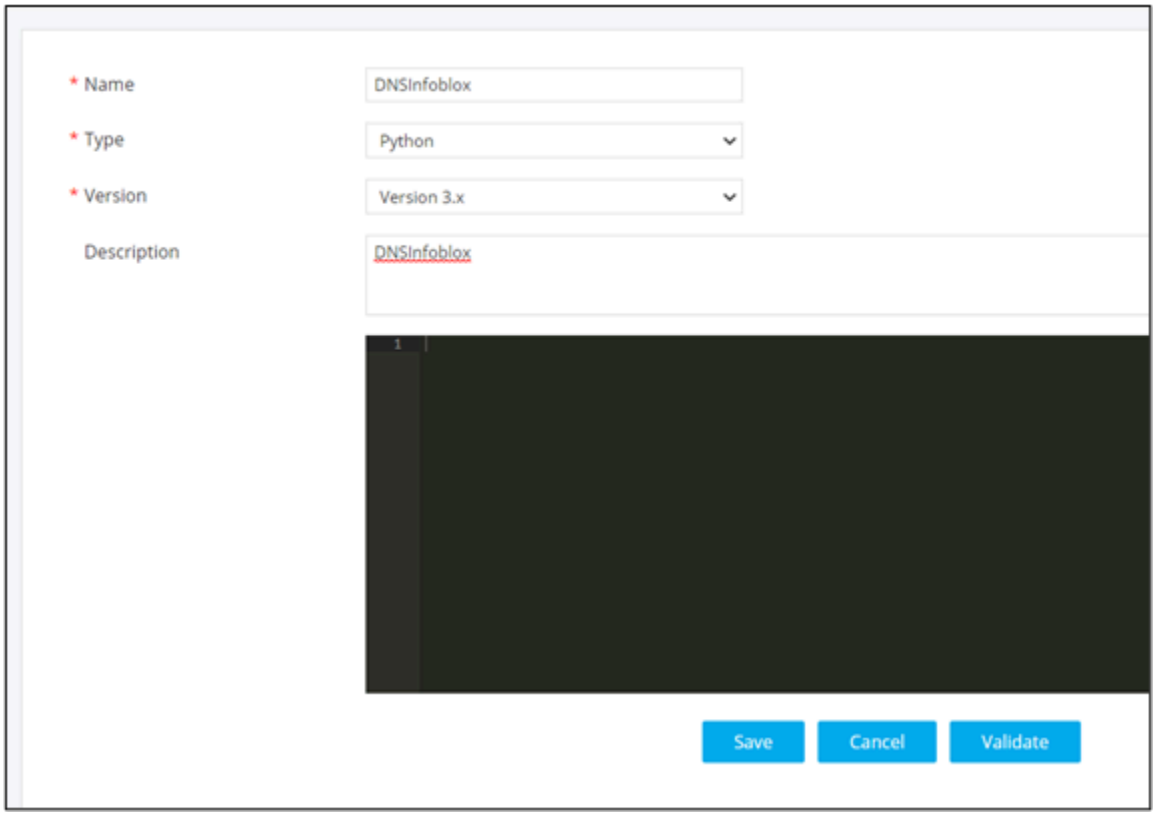
Creating a Helper Script

1. On the Workflow Inventory page, from the navigation pane on the left, click **Helper Script**.
   The **Helper Script Library** page is displayed.
2. To add a new Helper Script, on the **Helper Script Library** page, click + in the command bar.



3. Enter or select the field information.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

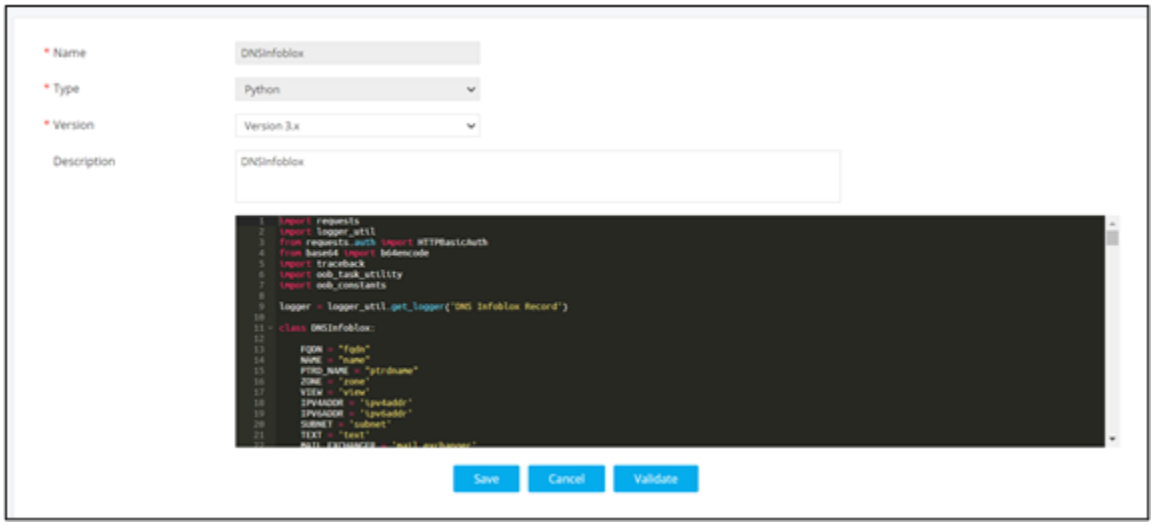


This table describes the field information in this section:

Field	Description
***Name**	Provide an appropriate name for the Helper Script.
***Type**	Select script as Python.
***Version**	Select version as 3.x
Description	Provide a valid description of the script.
All asterisk (*) marked fields are mandatory.	

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

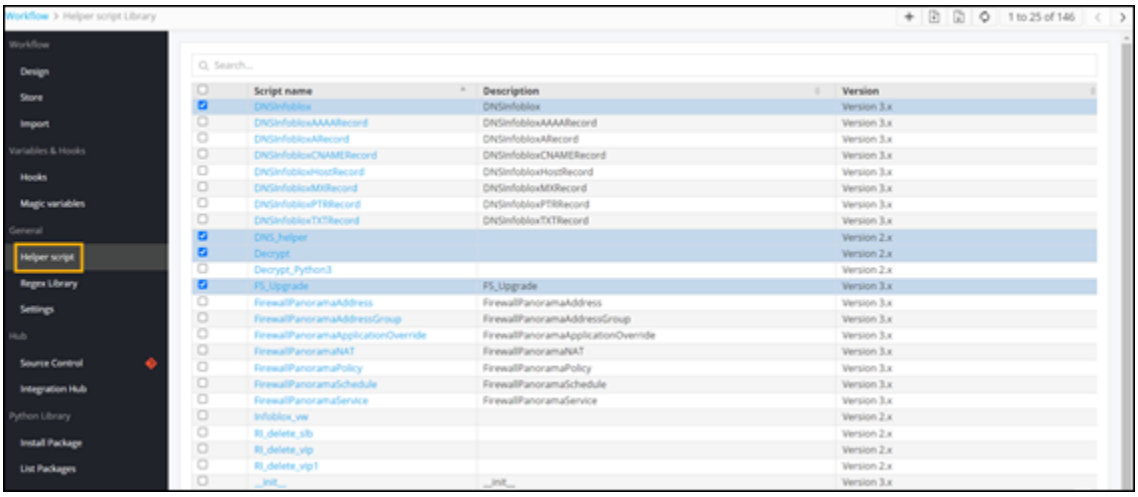
4. Define the script logic.



5. Click **Save**.

Exporting Helper Scripts

1. On the Workflow Inventory page, from the navigation pane on the left, click **Helper Script**.
   The **Helper Script Library** page is displayed.
2. On the **Helper Script Library** page, select the helper script(s) to be exported.

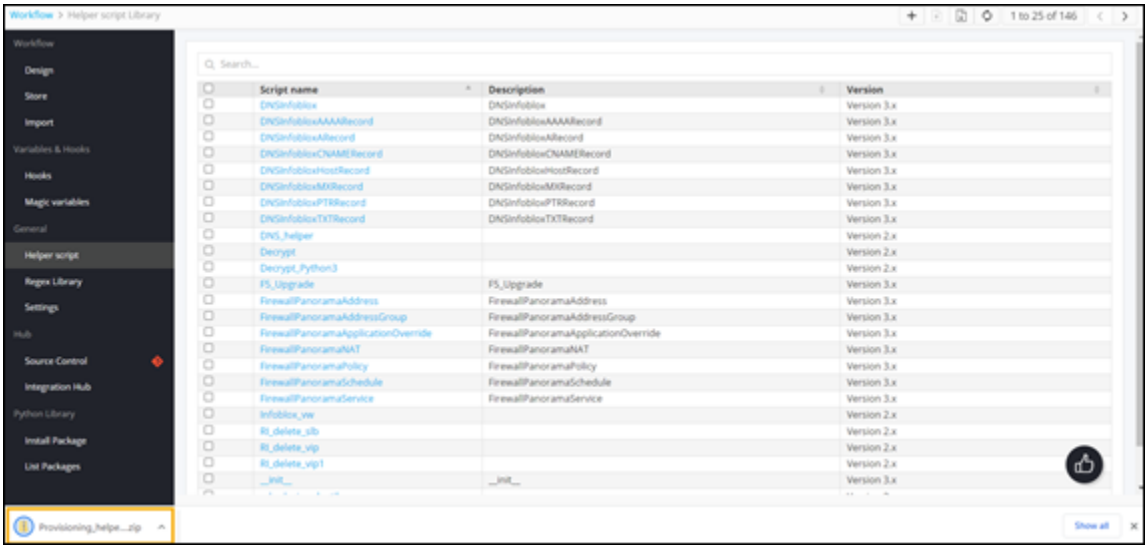


3. To export the selected helper script(s), click in the command bar.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

4. Click **Ok** in the **Confirmation** pop-up window.

   The helper script(s) is/are saved as a .zip file.



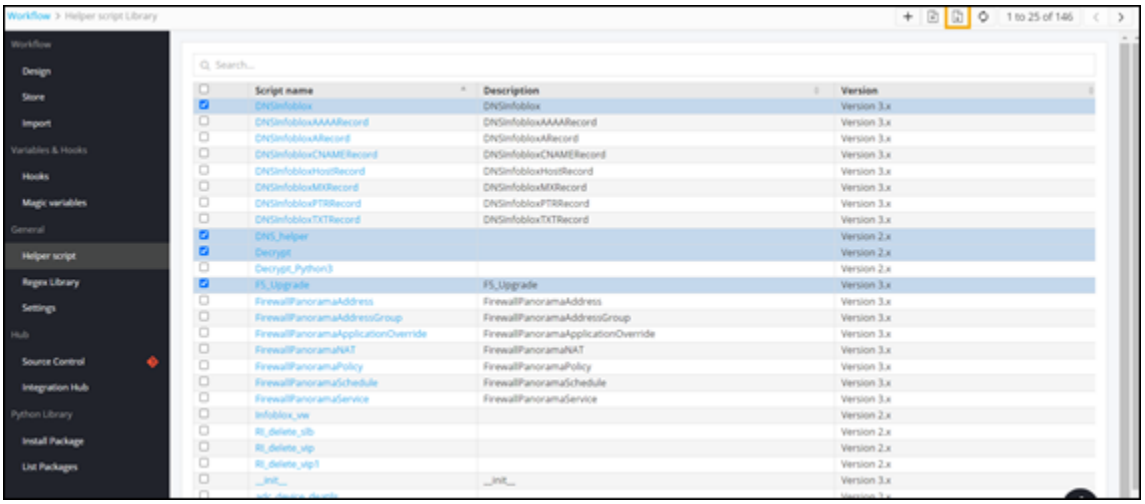
Importing Helper Scripts

1. On the Workflow Inventory page, from the navigation pane on the left, click **Helper Script**.

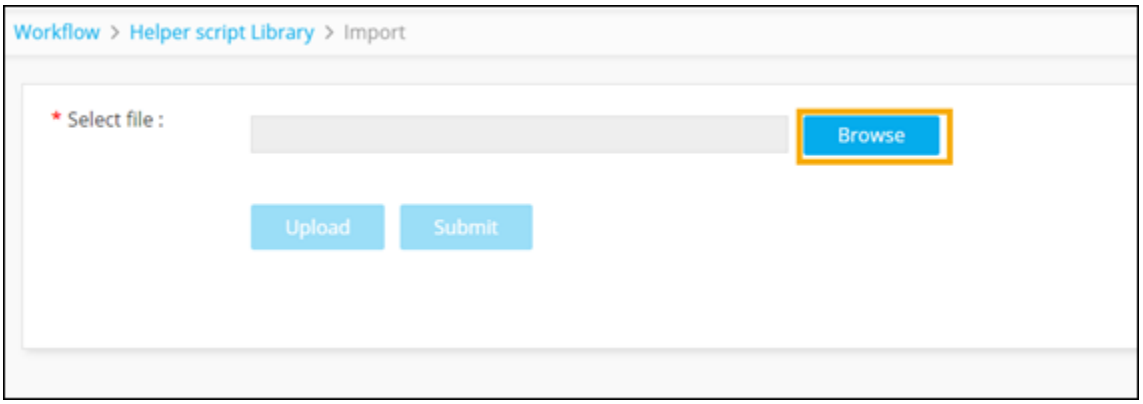
   The **Helper Script Library** page is displayed.

2. On the **Helper Scripts Library** page, click in the command bar.

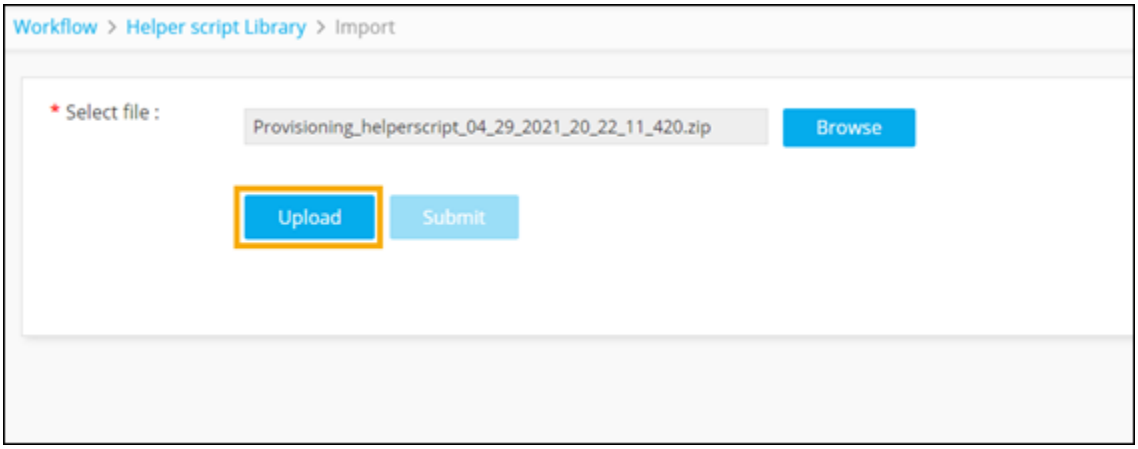
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



3. To select the helper script file to be uploaded, click **Browse**.

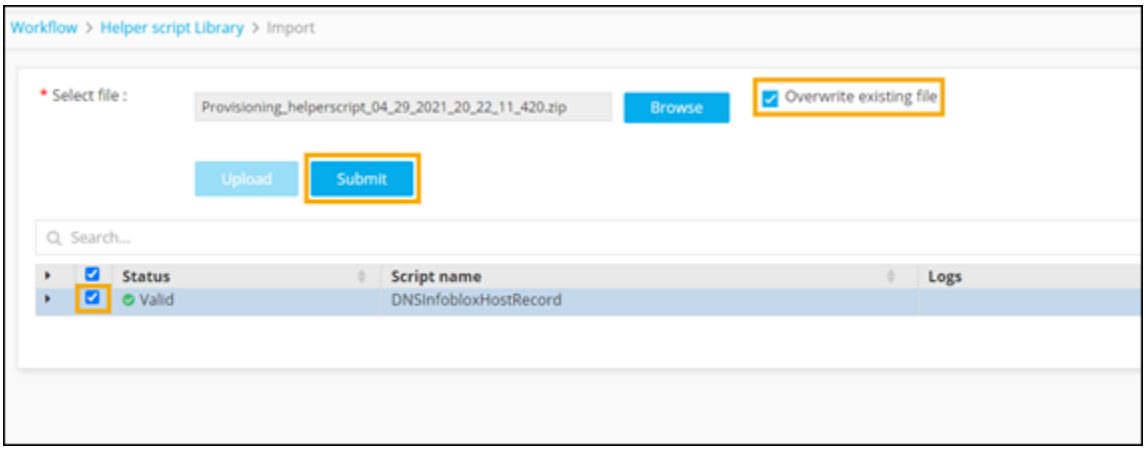


4. To upload the selected file, click **Upload**.



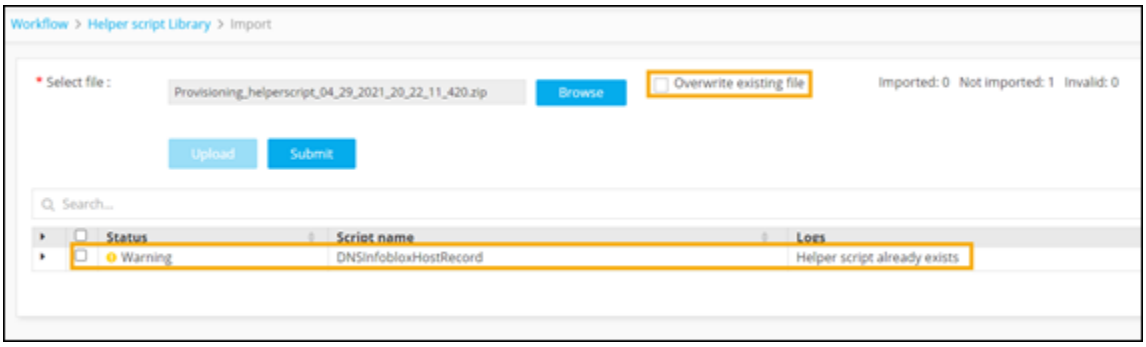
5. Select the script and click **Submit**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Note:

- By default, the 'Overwrite existing file' option will be checked. Any existing helper scripts will be overwritten when new files are imported.
- Only ".zip" files will be supported for import.
- If the 'overwrite existing file' option is not selected and the script already exists in the destination environment, the following notification will be displayed to the user on clicking Submit.



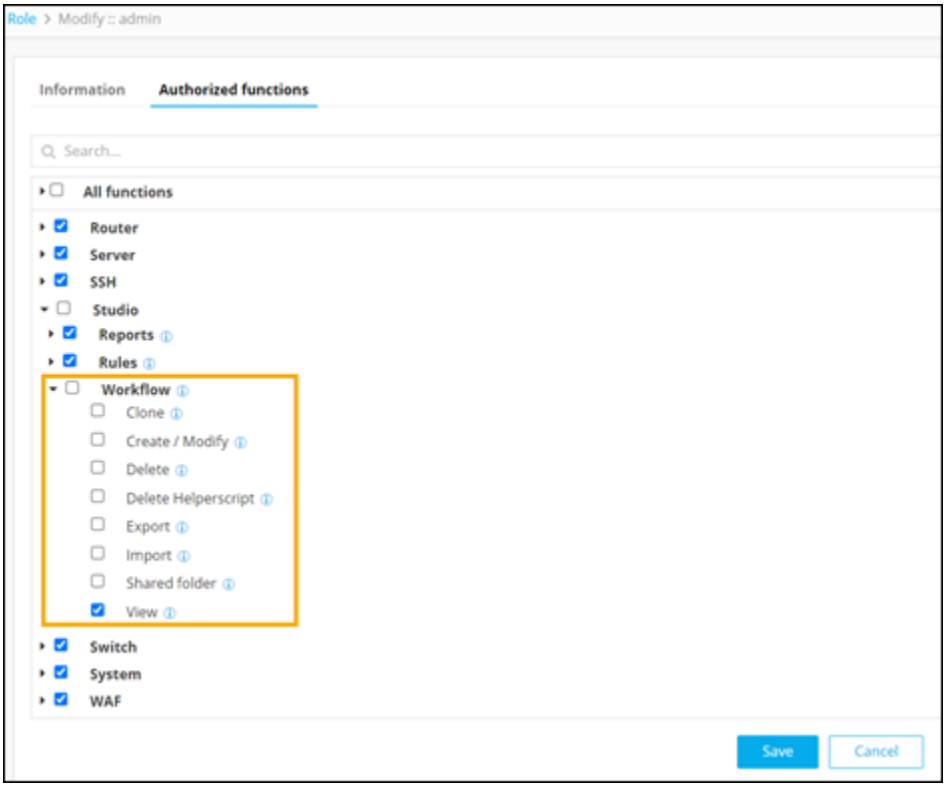
Configuring RBAC for Helper Scripts

You can allow or restrict a user to modify Helper Scripts. Users can modify helper scripts only when they have the Create/Modify permission for workflows. Users with 'View' permission cannot modify helper scripts.

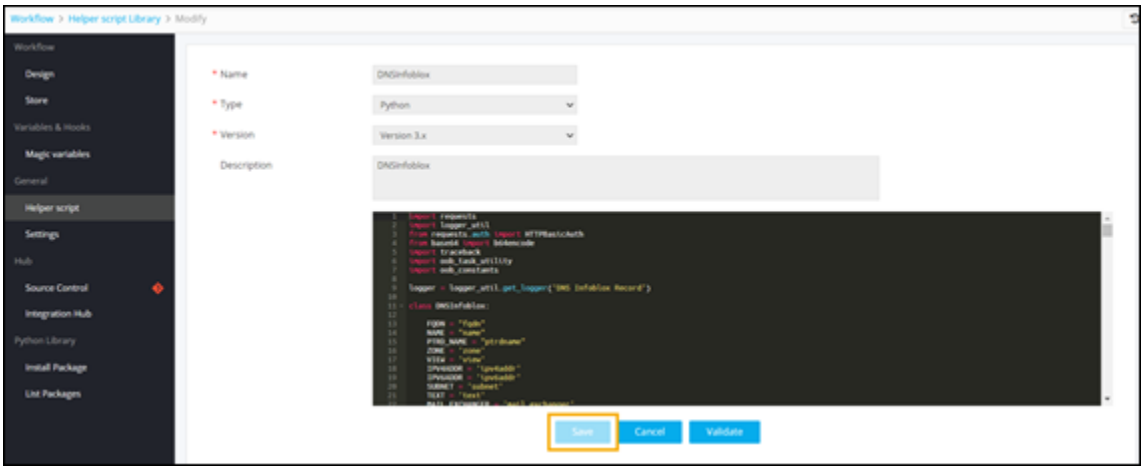
To restrict a user from modifying Helper Scripts in the Workflow Studio:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. From the upper left corner of the screen, click ☰.
2. From the menu displayed, select **Account** > **Role**.
3. Select the role for which you want to enable access, for example, admin.
4. Under **Authorized functions**, select **Request** > **Workflow**.
5. Under **Workflow**, only select the **View** checkbox.



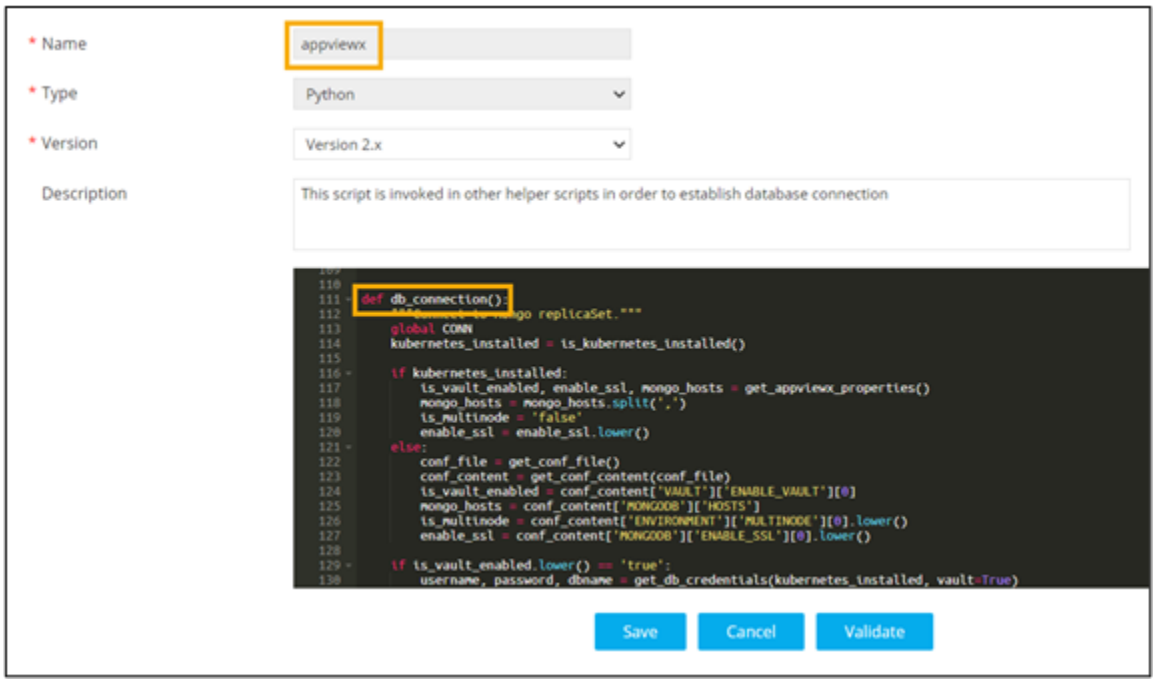
On the **Helper Script Library Modify** page, the script editor and **Save** button are disabled.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Using a Default Helper Script

1. Leverage the helper script called ‘appviewx’, which establishes database connection. Function name: db_connection



2. Reuse the predefined helper script across other workflow scripts such as ‘Get device list’.

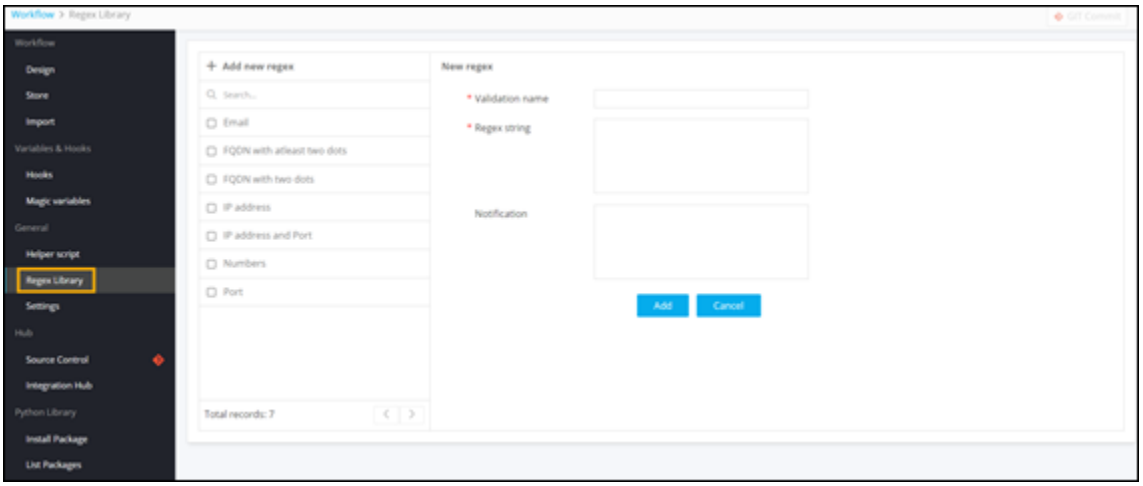
Get device associate script

```
import sys

import json

sys.path.insert(0, AVX::DEPENDENCIES)

sys.path.insert(0, AVX::HELPER)

import appviewx


def device_list():

    connection = appviewx.db_connection()  _--------------------------------> helper.helperfunction()

    collection = connection.appviewx.device

    dev_list = [value['name'] for value in collection.find({"vendor":"F5", "category" : "ADC"})]

    dev_list = sorted(filter(None,dev_list), key = lambda s: s.lower())

    print (json.dumps([{'Device': name} for name in dev_list])) if dev_list else (json.dumps({'error':"No Device to list"}))


device_list()
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Regex Library

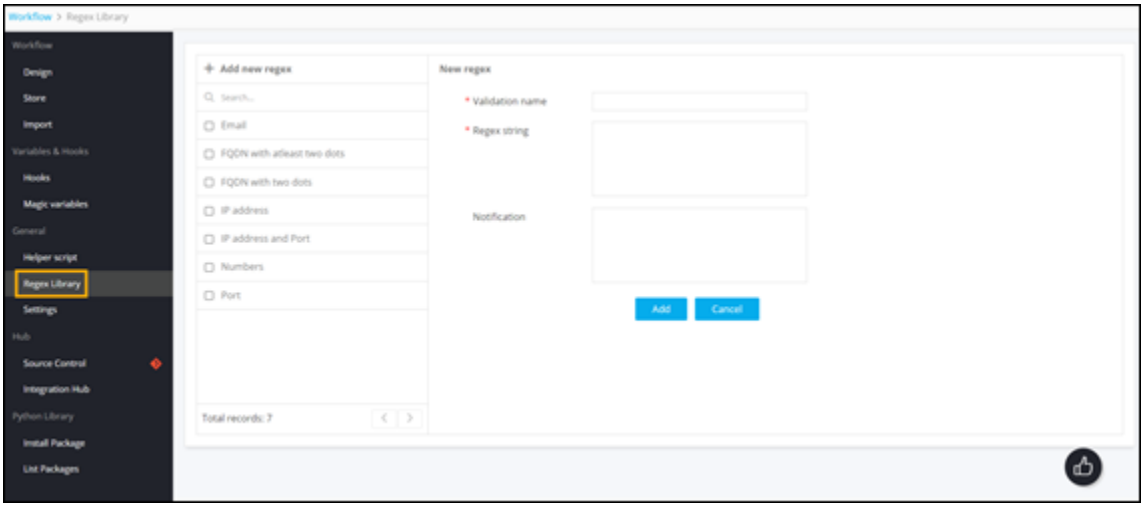
The Regex Library allows you to define custom regex validation or use the prebuilt regex patterns. Regex validations can be referenced against a form field with specific notifications to the user. You can add, edit any regex in the inventory.



- Adding a New Regex
- Modifying an Existing Regex

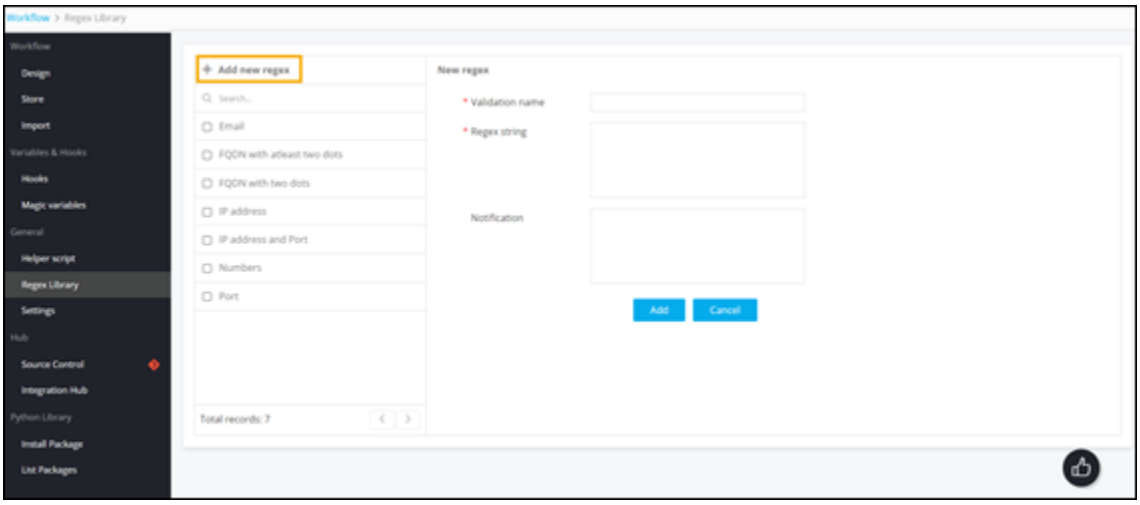
Adding a New Regex

1. On the Workflow Inventory page, from the navigation pane on the left, select **Regex Library**.
   The **Regex Library** page is displayed.

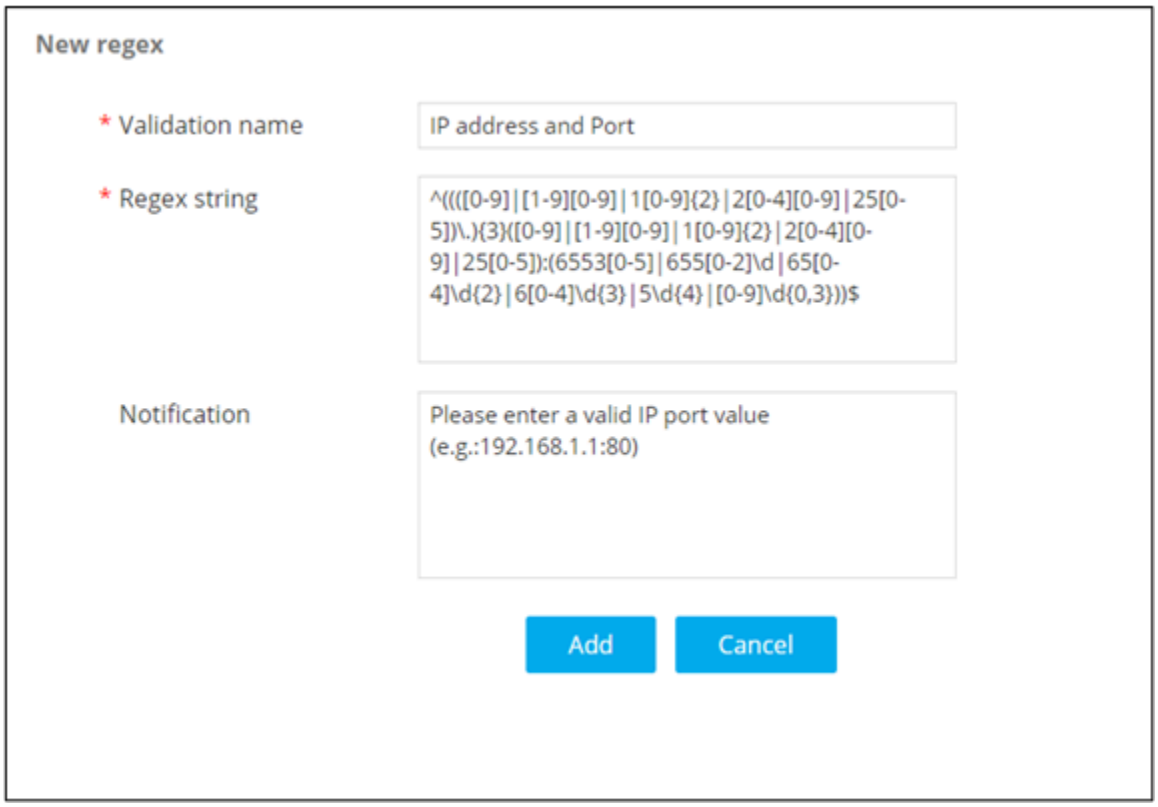


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

2. On the **Regex Library** page, click **Add new regex**.



3. To add a new regex string, enter the field information as shown.

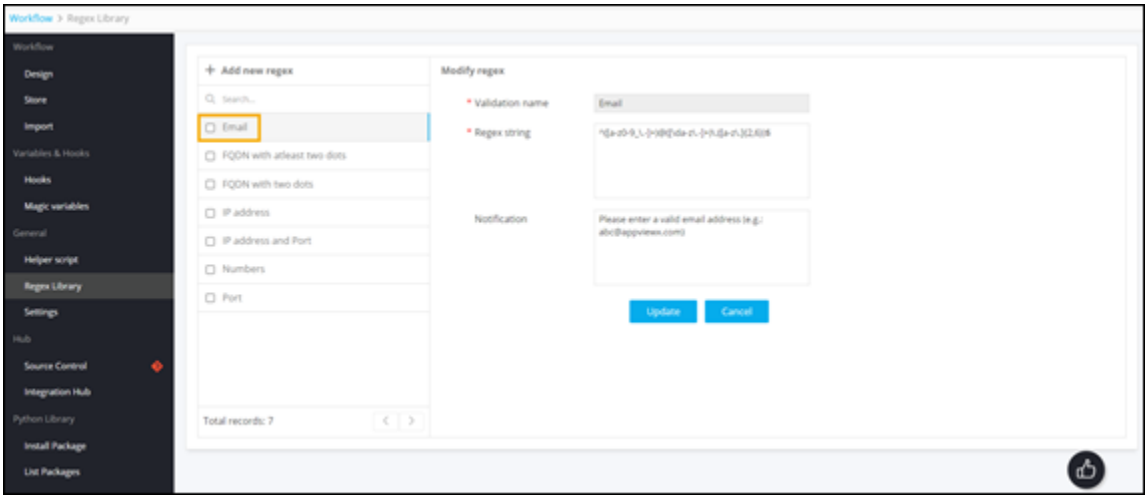


4. To save this new regex script to the library, click **Add**.

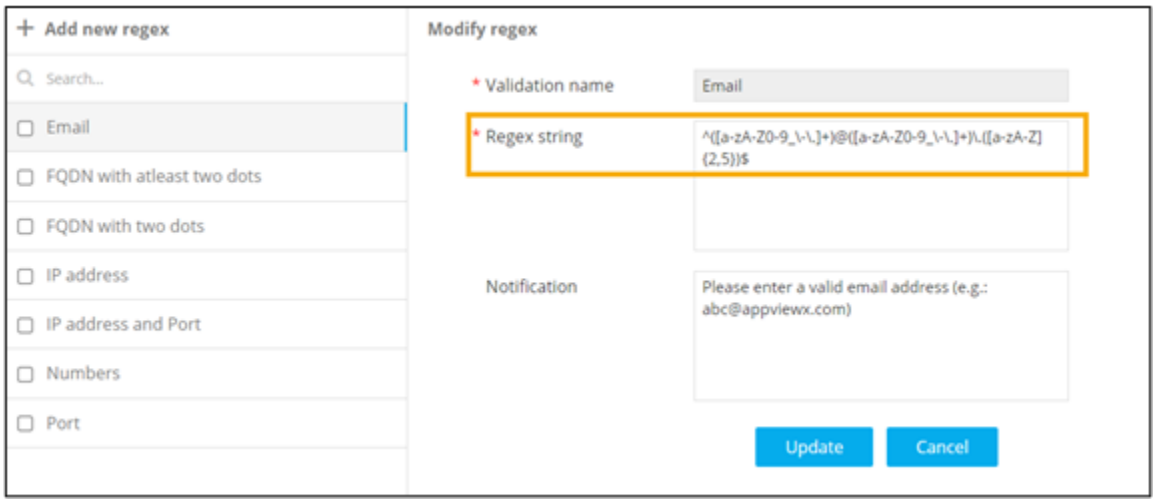
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Modifying an Existing Regex

1. On the Workflow Inventory page, from the navigation pane on the left, select **Regex Library**.
   The **Regex Library** page is displayed.
2. On the **Regex Library** page, select an existing Regex from the list.



3. To modify the **Regex string**, enter a different regex pattern for email validation.

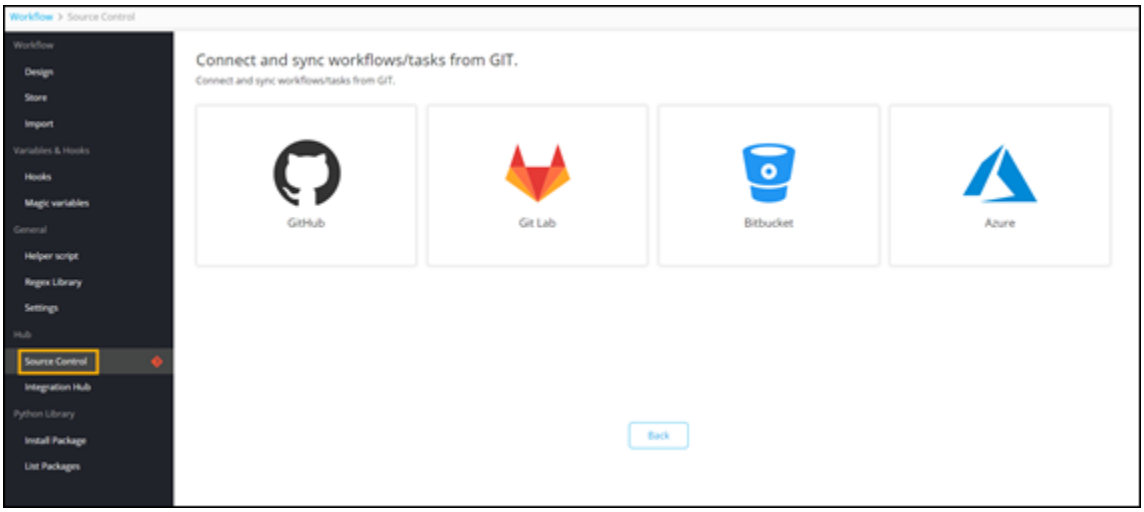


4. To save this modified regex pattern, click **Update**.

Source Control

This feature allows you to connect and sync workflows/tasks from GIT. You can integrate with GitHub, GitLab, Bitbucket, and Azure through AppViewX.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



- Prerequisites
- Linking a Repository
- Committing Workflows in Inventory
- Committing Workflows in the Design Section
- Committing Tasks and Subflows
- Pushing to a Remote Repository from Workflow Inventory Page
- Pushing to a Remote Repository from the Design Studio
- Pulling from the Repository from Workflow Inventory Page
- Pulling from the repository from the Design Studio
- Unlinking a Repository
- Logs

Prerequisites

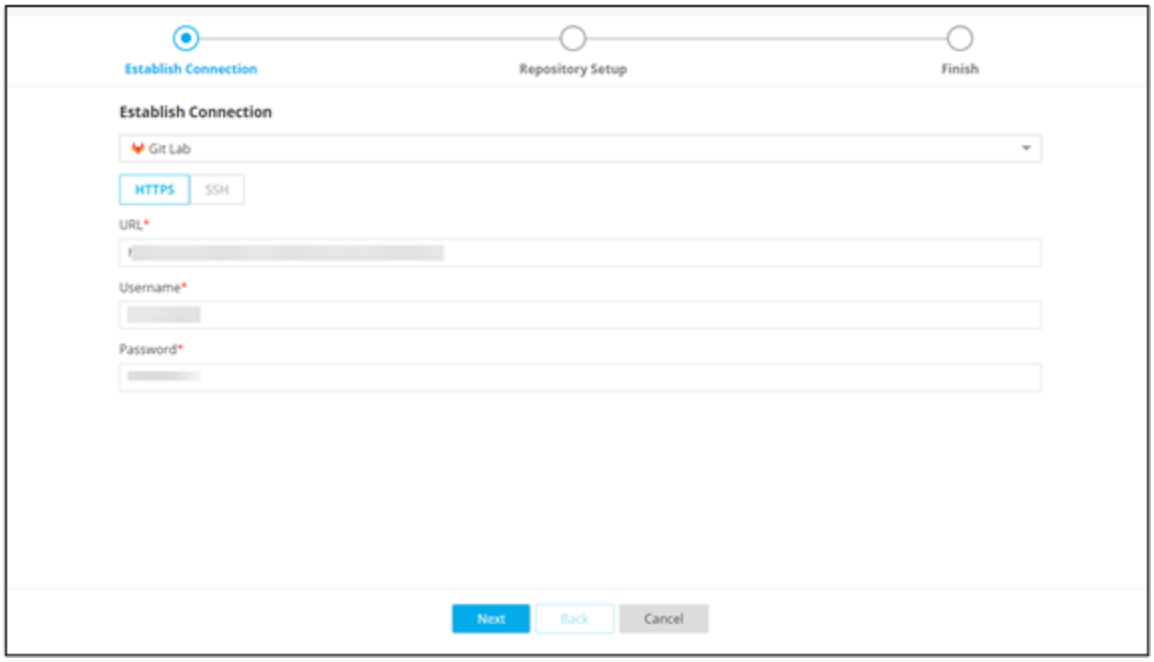
- Users should have role based access permission to the workflow studio
- The user should have a configured remote GIT repository

Linking a Repository

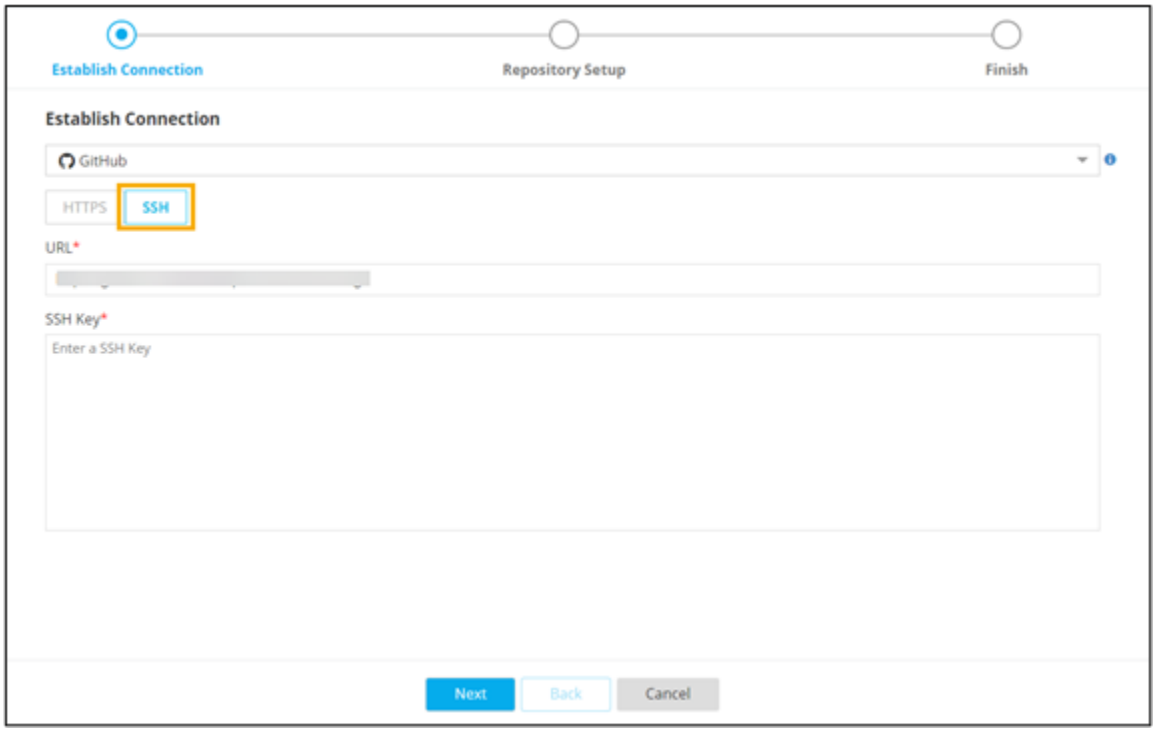
1. On the Workflow Inventory page, from the navigation pane on the left, click **Source Control**.
2. On the Source Control landing page, select the integration type as **GitLab**.
3. Select the connection type as HTTPS or SSH.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

4. For **HTTPS**, enter the valid credentials along with the HTTPS clone **URL**.



5. For **SSH**, enter the valid private key corresponding to the public key and SSH clone URL.



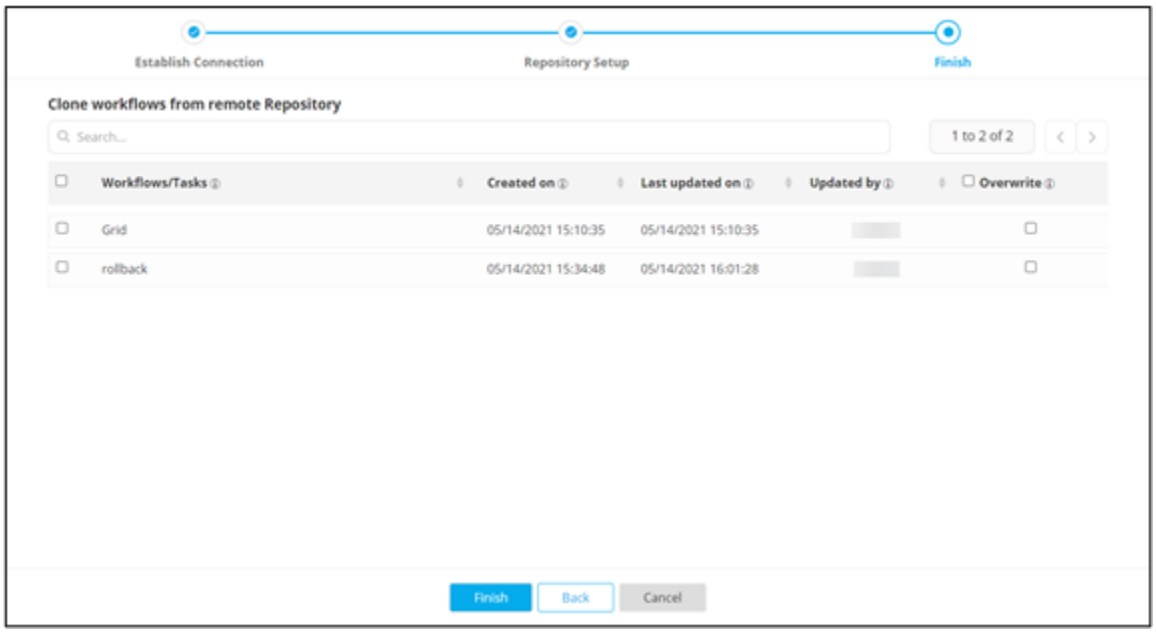
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Note: Prerequisite for SSH - Public key is to be added to the remote user account.

6. Under **Branches**, select **master**.

7. Click **Next**.

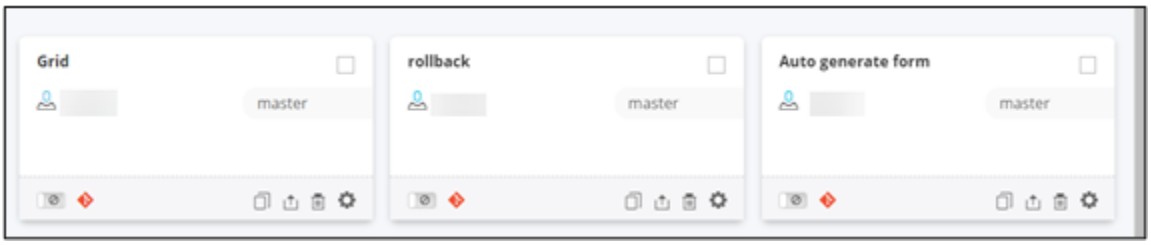
   All available workflows, tasks and subflows in the remote repository are displayed.



8. Select the workflows to be cloned.
9. Click **Finish**.
10. Click **Yes** in the **Confirmation** pop-up window.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

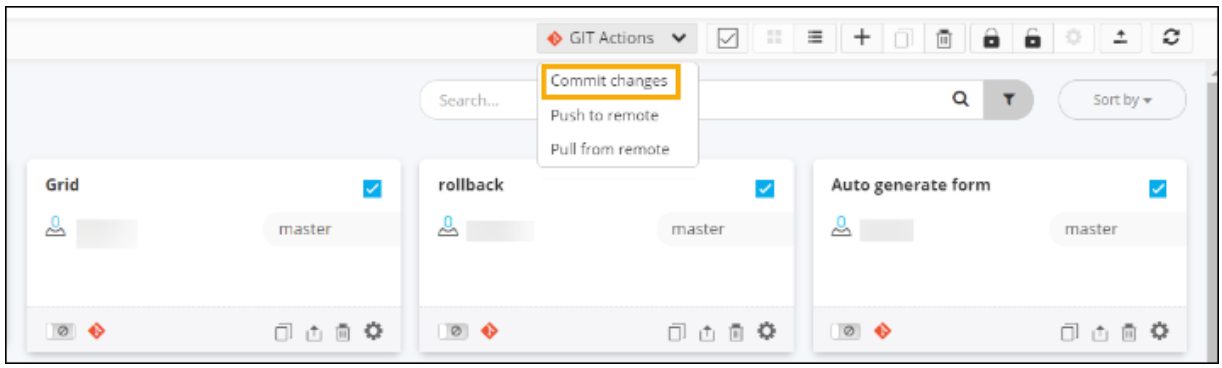
- Cloned tasks and subflows are added to Source Control Directory and Workflows to Inventory. The connected workflows reflect the Source Control logo .



- Cloned Repository is added to the **Manage Repositories** page.

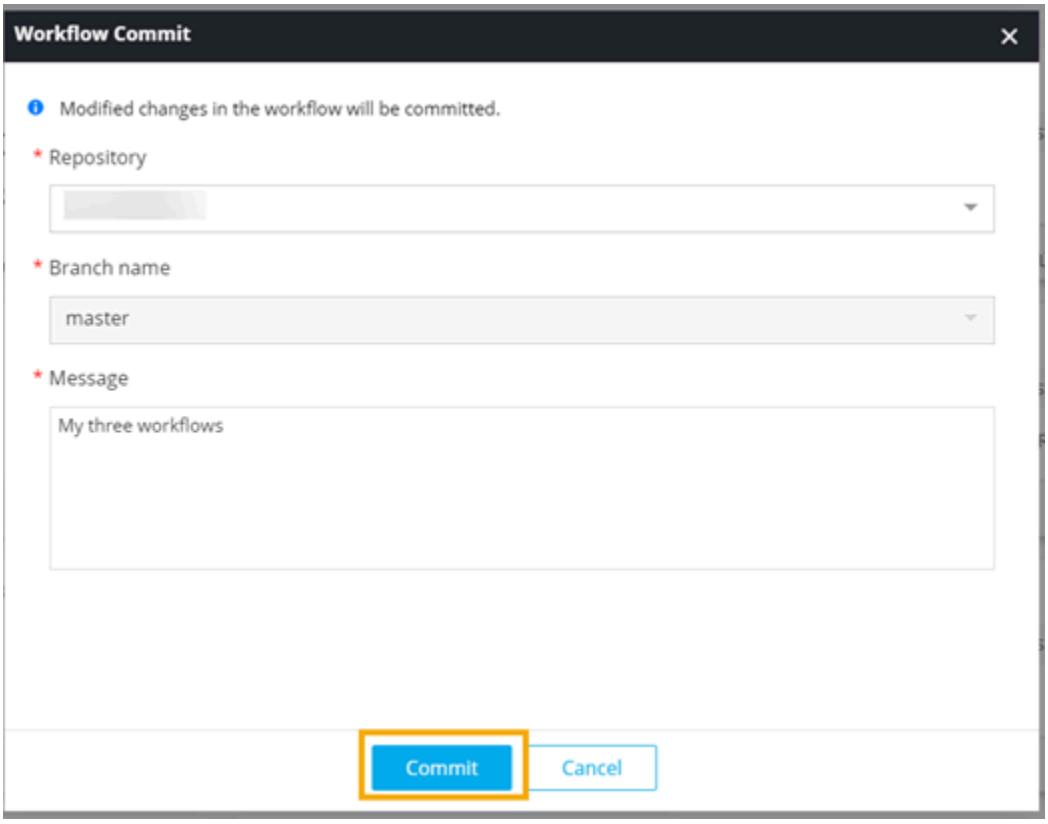
Committing Workflows in Inventory

1. On the **Workflow** inventory page, select the workflows to be committed.
2. From the **GIT Actions** dropdown menu, select **Commit changes**.



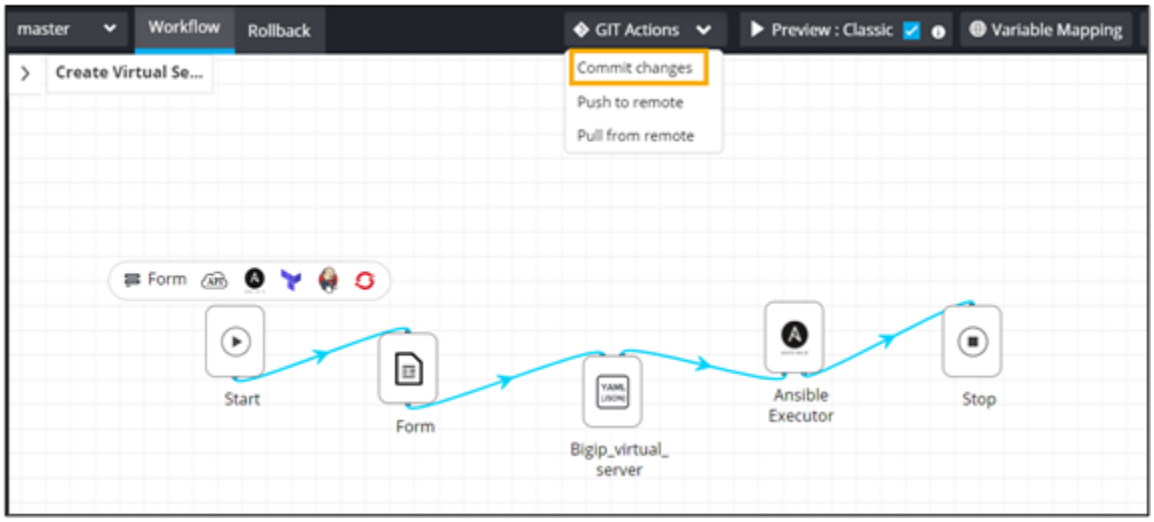
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

3. In the **Workflow Commit** window, enter a commit **Message** and click **Commit**.



Committing Workflows in the Design Section

1. Design a workflow or open an existing workflow.
2. Click **Git Commit**.

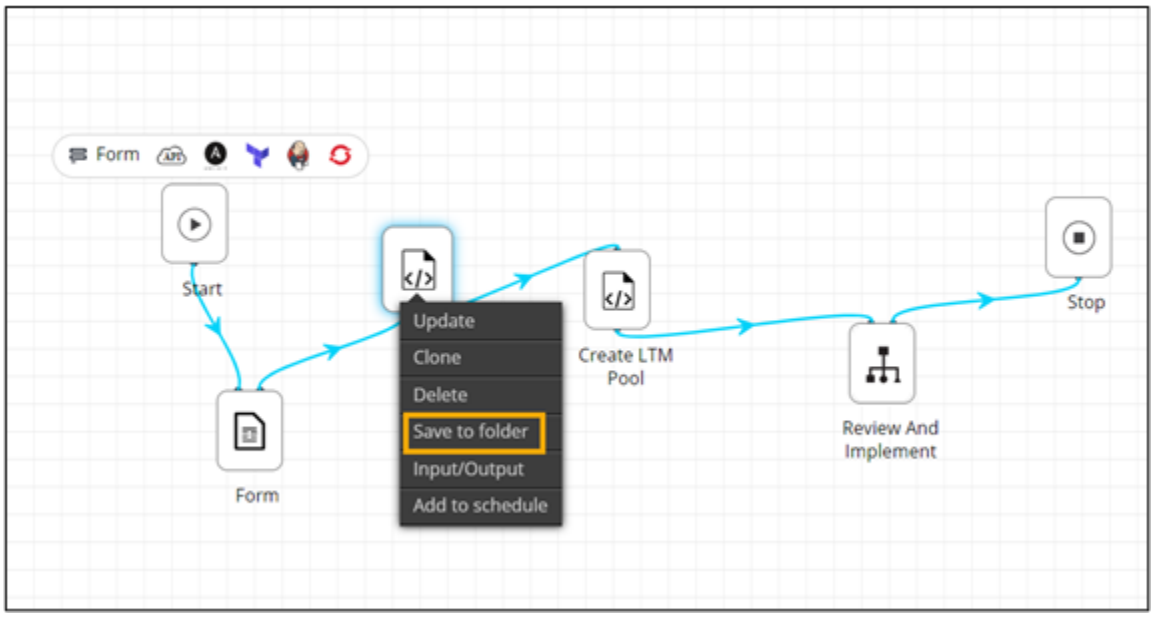


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

3. In the **Workflow Commit** window, enter a commit **Message** and click **Commit**.

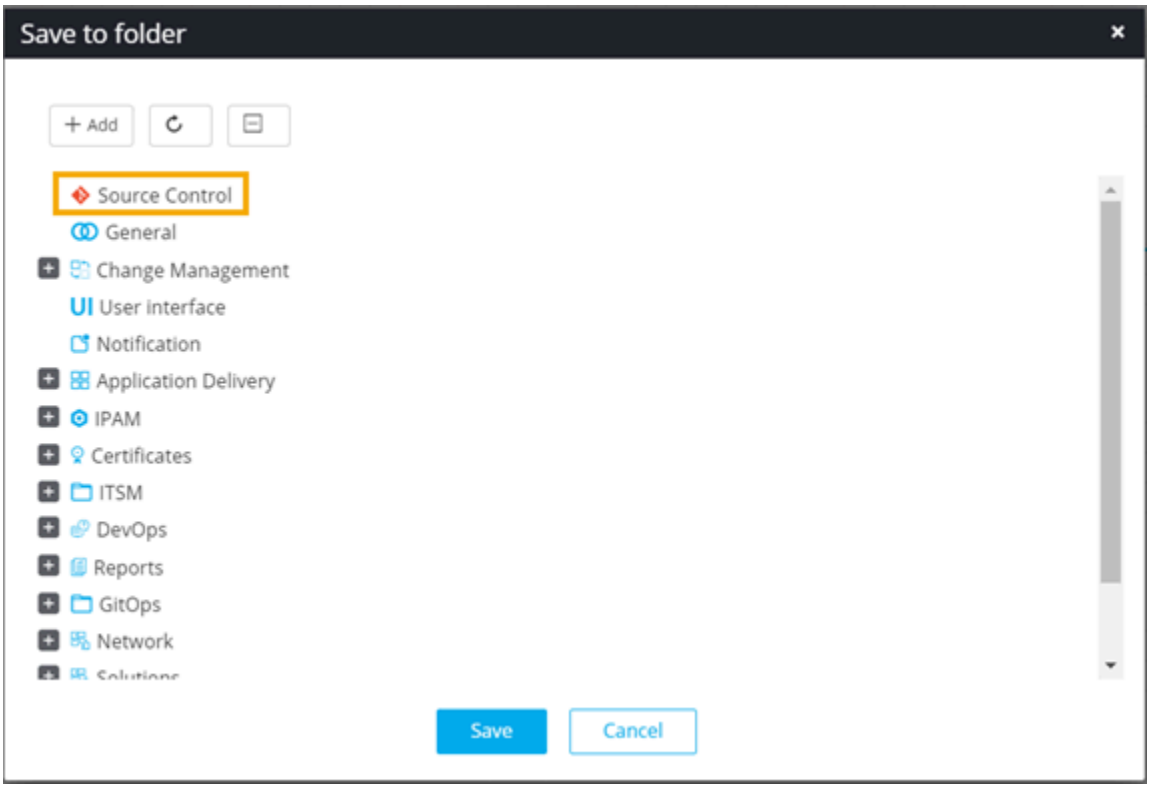
Committing Tasks and Subflows

1. Design a workflow or open an existing workflow.
2. Right-click on a task to see the options.
3. From the options displayed, click **Save to Folder**.

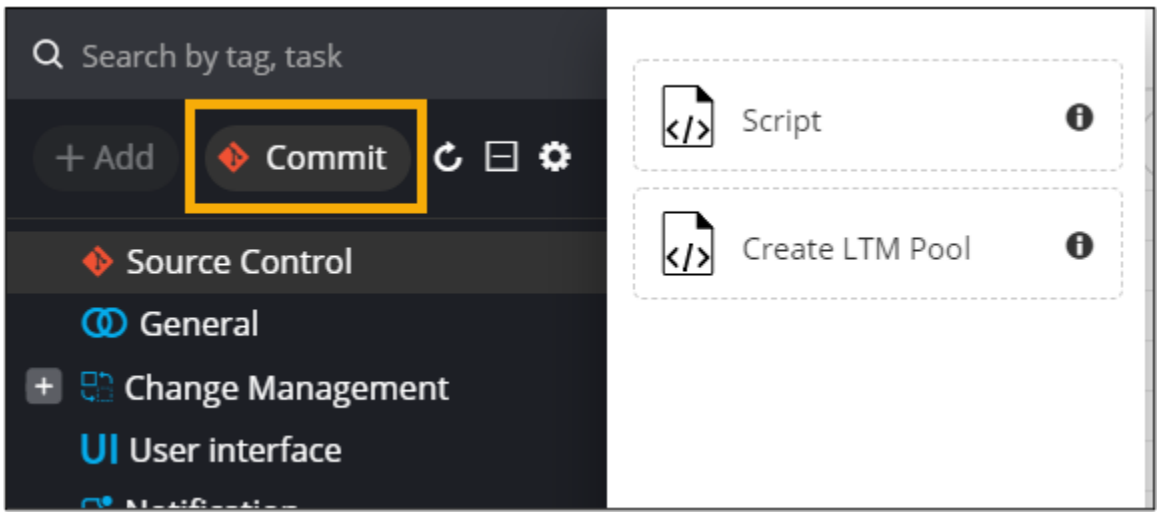


4. In the **Save to folder** window, save the task to the **Source Control** directory.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



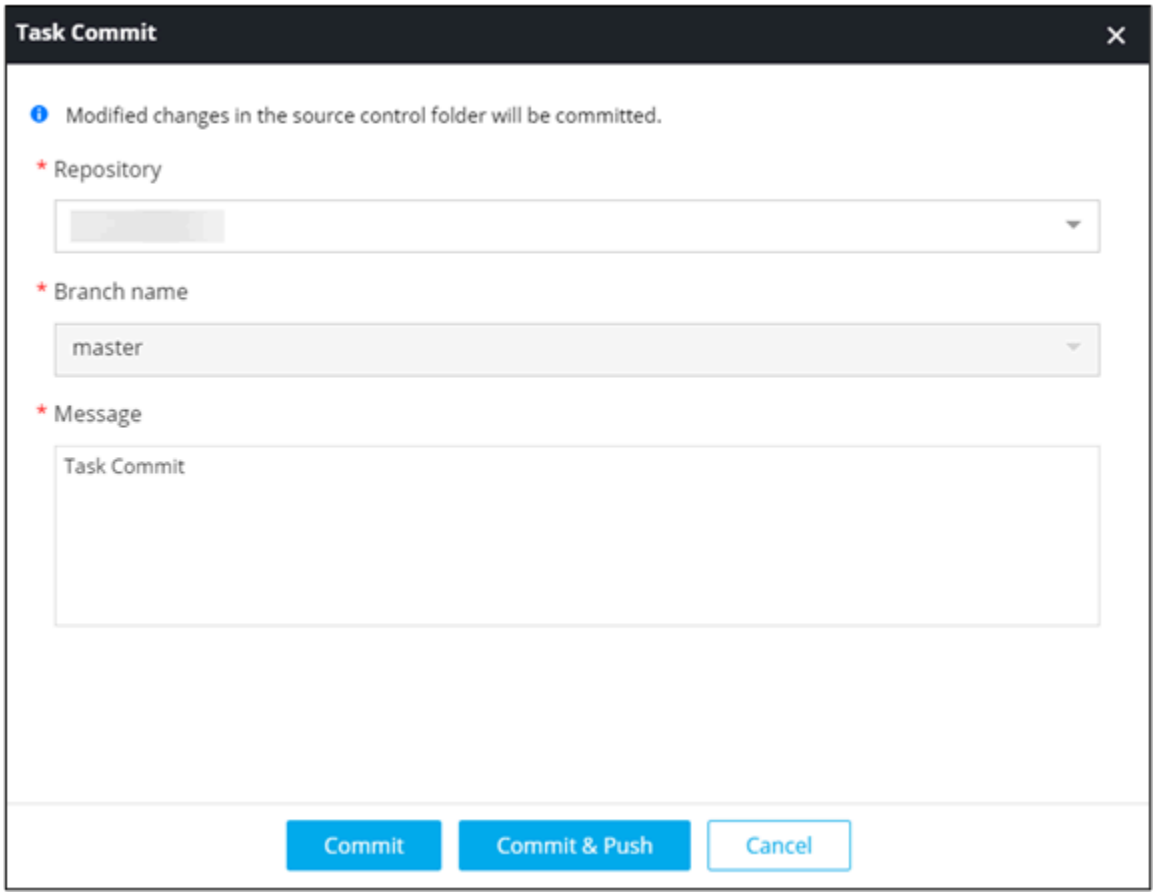
5. Click **Save**.
6. Add the tasks and subflows to be committed to the Source Control Directory.
7. Click **Commit**.



Note: By default, the Commit button is disabled. It is enabled only after selecting the Source Control Directory to save tasks.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

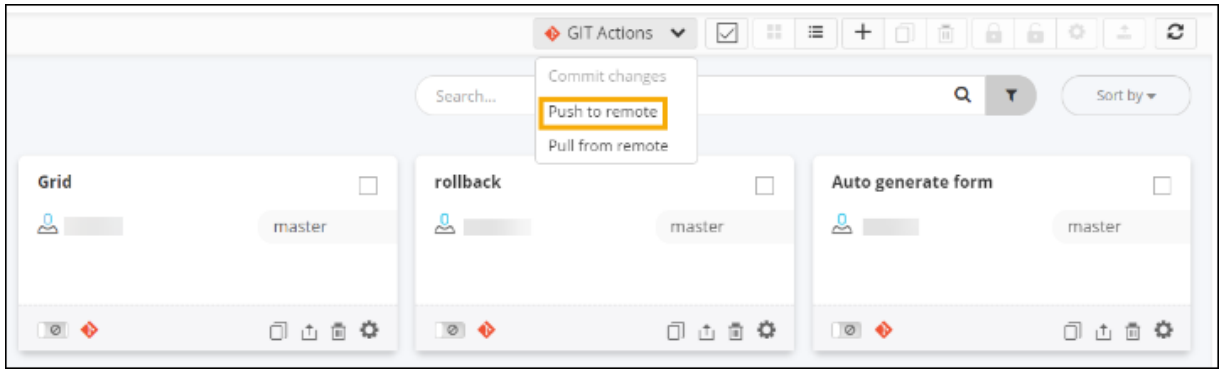
8. In the **Task Commit** window, enter a commit **Message** and click **Commit**.



Note: Only the recently added tasks/subflows will be committed.

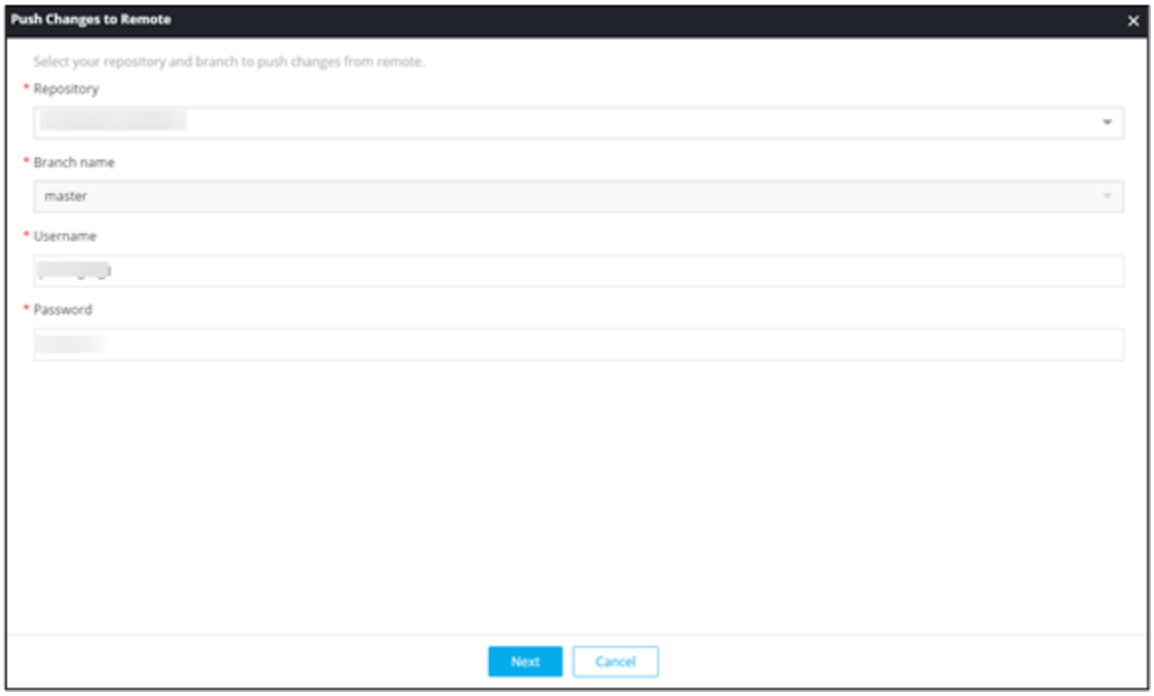
Pushing to a Remote Repository from Workflow Inventory Page

1. On the **Workflow** Inventory page, from the **GIT Actions** dropdown menu, select **Push to remote**.

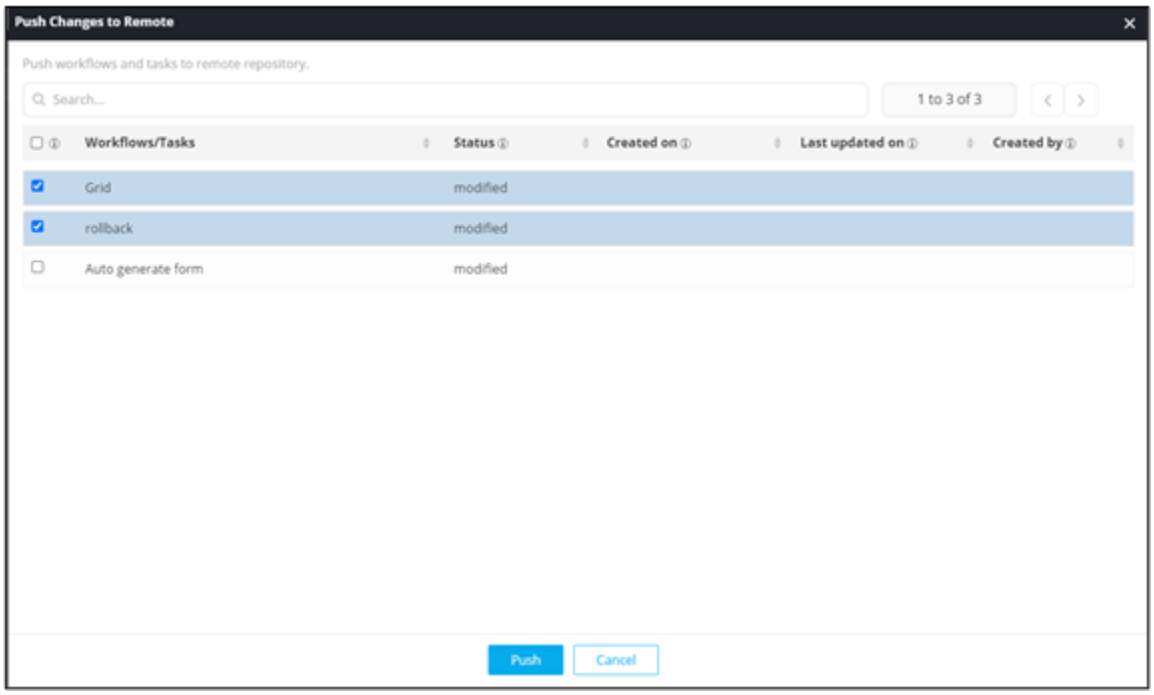


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

2. In the **Push Changes to Remote** window, enter the **Password**.



3. Click **Next**.
4. In the **Push Changes to Remote** window, from the list of committed tasks, select the task(s) to be pushed.



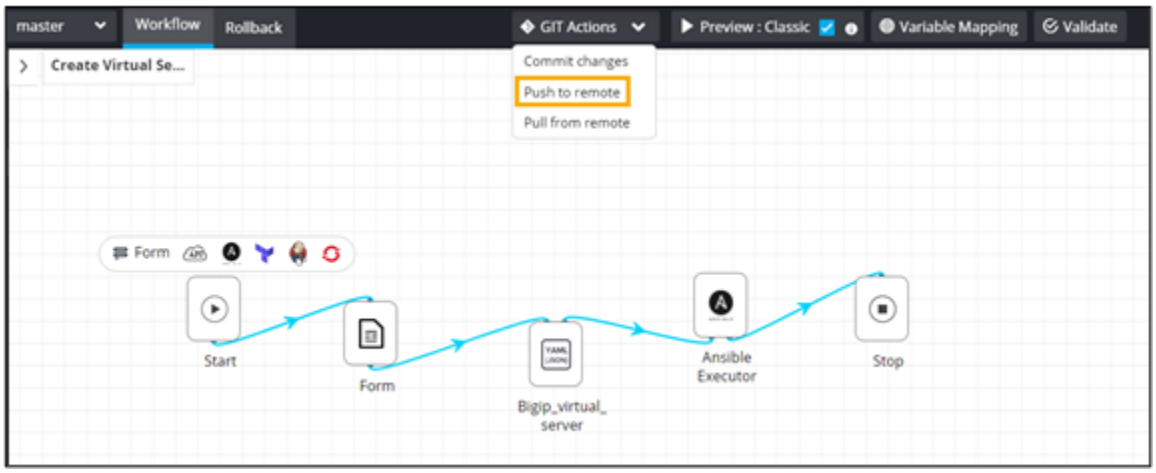
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

5. Click **Push**.

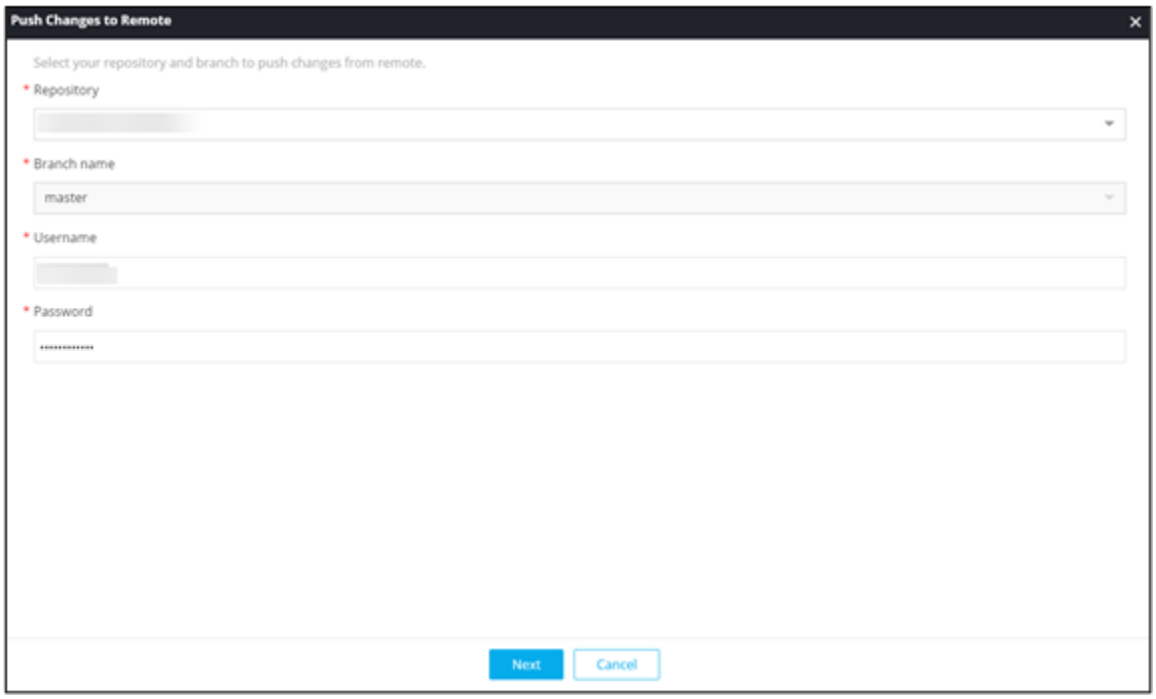
   The selected workflows/tasks are pushed successfully.

Pushing to a Remote Repository from the Design Studio

1. Design a workflow or open an existing workflow.
2. From the **GIT Actions** dropdown menu, select **Push to remote**.

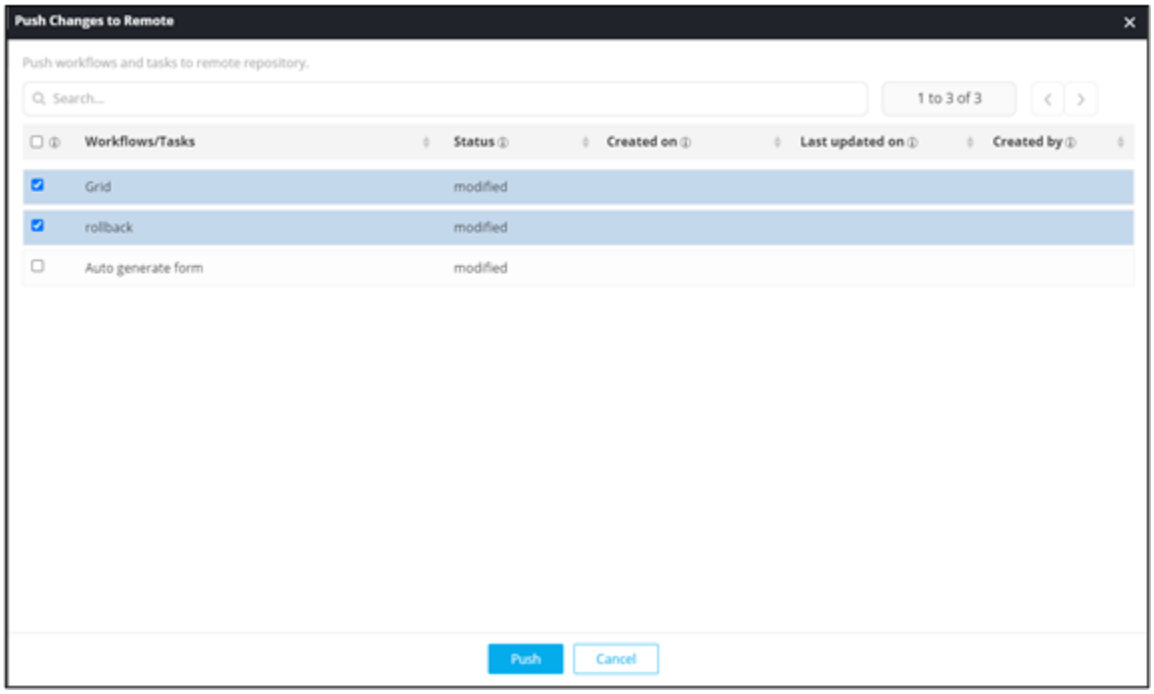


3. In the **Push Changes to Remote** window, enter the **Password**.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

4. In the **Push Changes to Remote** window, from the list of committed tasks, select the task(s) to be pushed.

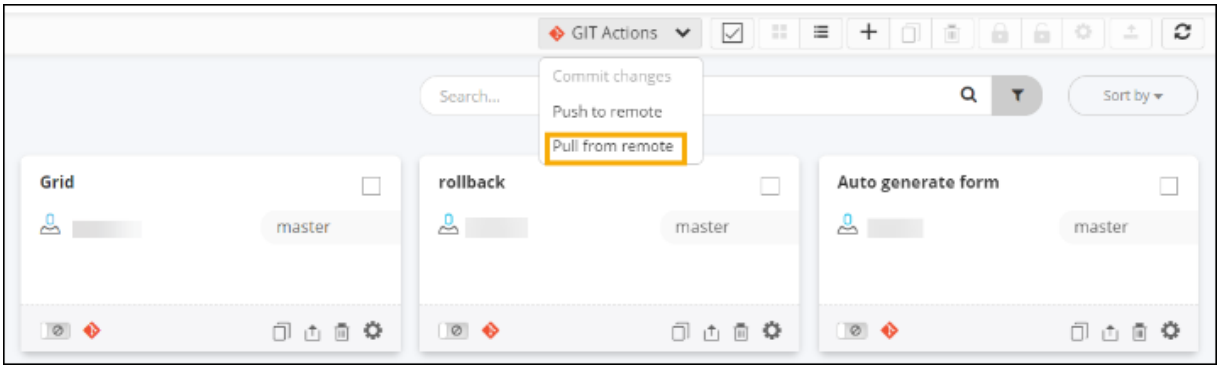


5. Click **Push**.

   The selected workflows/tasks are pushed successfully.

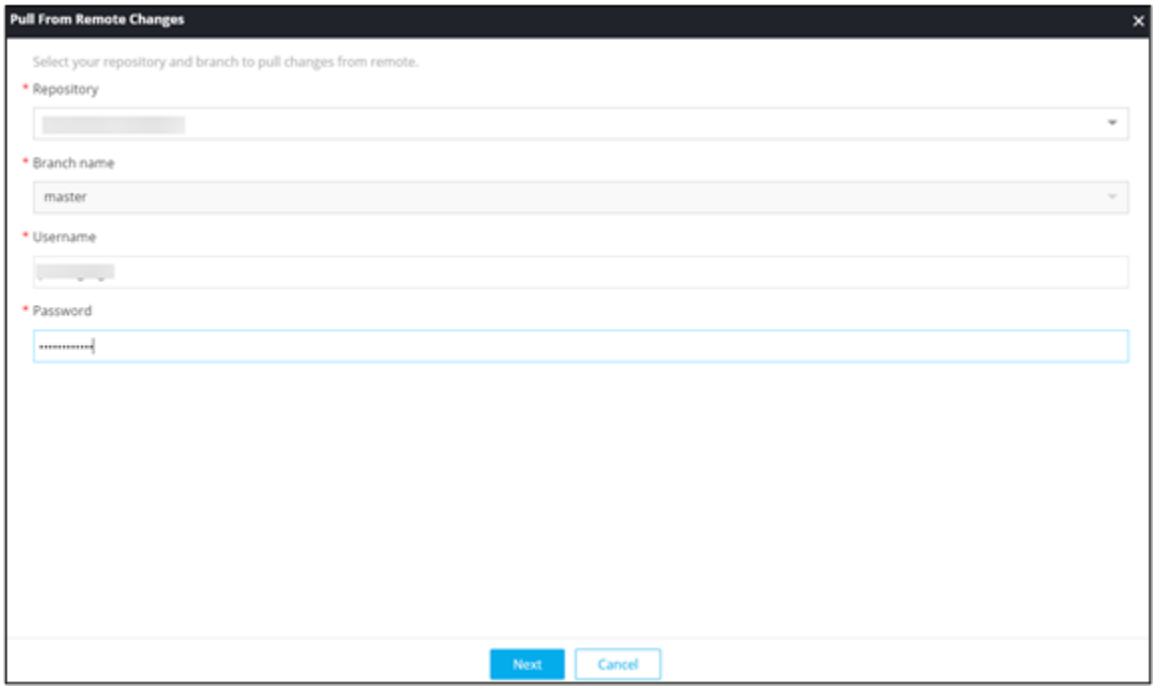
Pulling from the Repository from Workflow Inventory Page

1. On the **Workflow** Inventory page, from the **GIT Actions** dropdown menu, select **Pull from remote**.

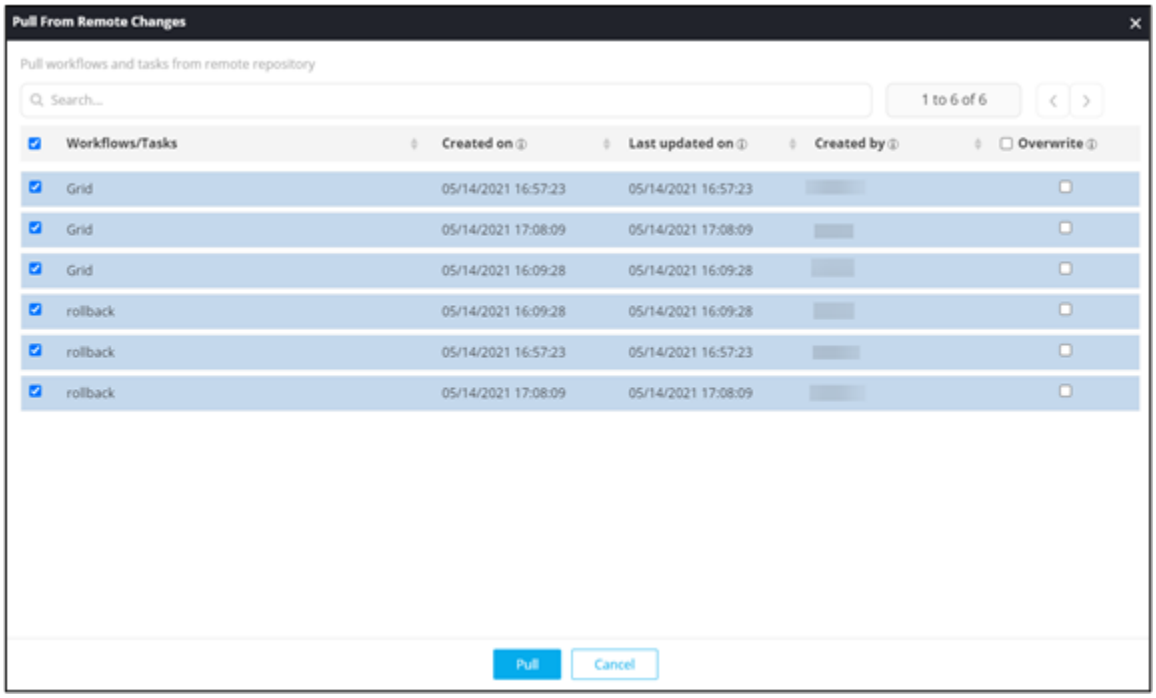


2. In the **Pull From Remote Changes** window, enter the field information.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



3. In the **Pull From Remote Changes** window, select the workflows to be pulled and click **Pull**.

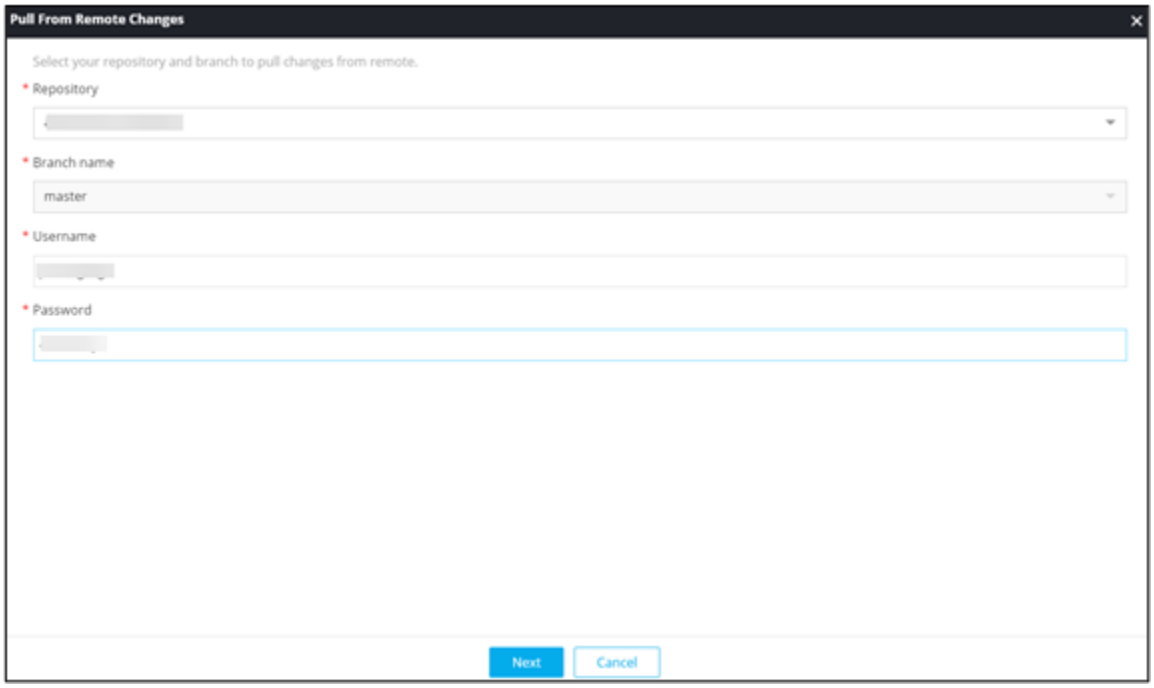


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Pulling from the repository from the Design Studio

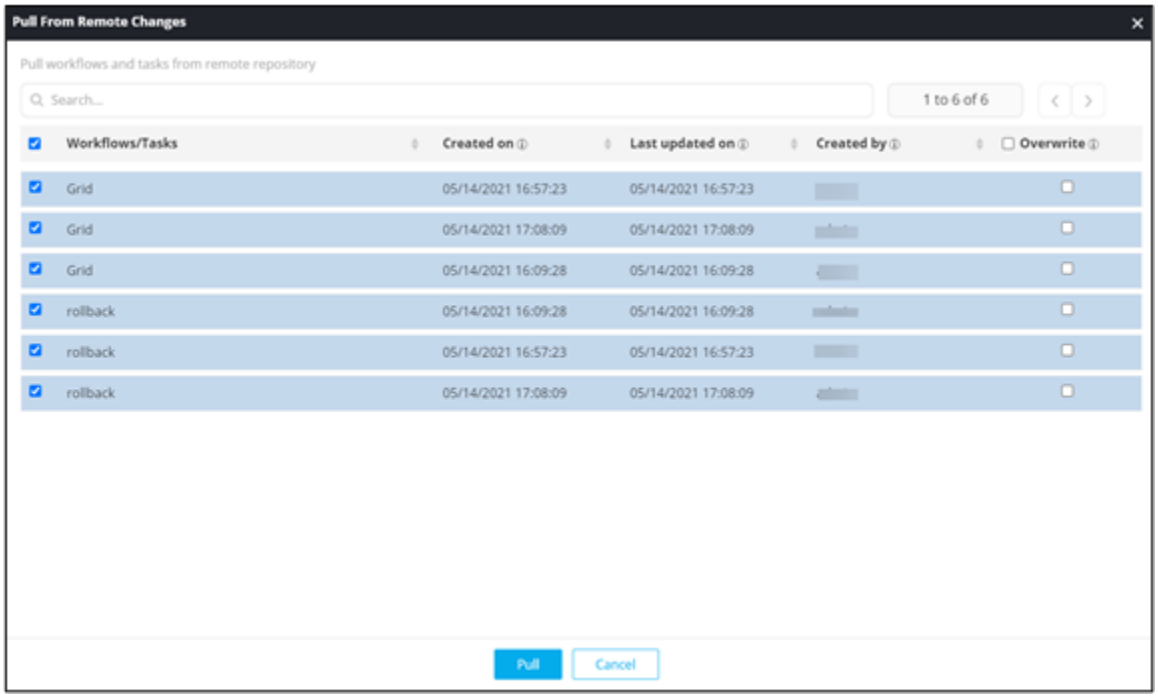
1. Design a workflow or open an existing workflow.
2. From the **GIT Actions** dropdown menu, select **Pull from remote**.

3. In the **Pull from Remote Changes** window, enter the field information.



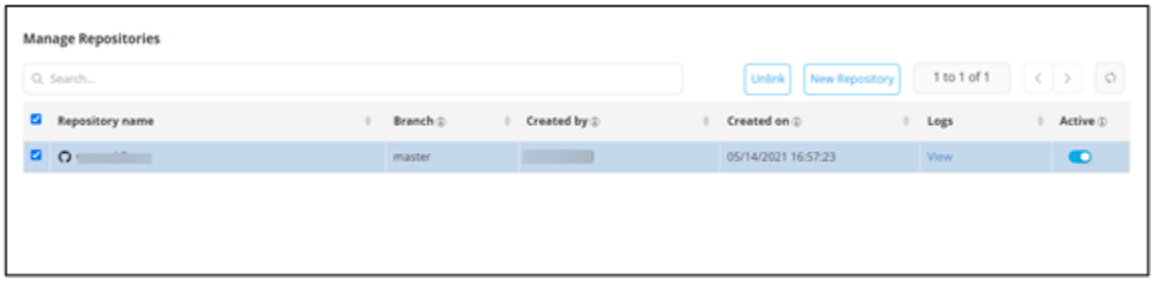
4. In the **Pull from Remote Changes** window, select the workflows to be pulled and click **Pull**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Unlinking a Repository

1. On the Workflow Inventory page, from the navigation pane on the left, click **Source Control**.
2. Under **Manage Repositories**, select the repository to be unlinked.



3. Deactivate the repository by turning off the toggle.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

4. To unlink the repository, click **Unlink**.

5. Click **Ok** in the **Confirmation** pop-up window.

Logs

Logs is an option supported to view the actions performed in the chosen repository. Logs for the latest 10 actions are displayed in the descending order.

The logs can be viewed on the **Manage Repositories** page.

The following actions are **supported** for Source Control:

- Support for Cloning, Pulling, Committing and Pushing Workflows,Tasks, and Subflows.
- Support for both On Premises and Cloud Repositories.
- Support for both public and private repositories.
- Support for SSH and HTTP/HTTPS authentication protocols.
- New branch can be created from AppViewX.
- Support for creating new versions and overwriting existing workflows on selection while importing them.
- Tasks and subflows can be overwritten or duplicated on selection while importing them.
- Support for cloning an empty repository is available.
- Allows switching between active repositories while performing Push, Pull, and Commit actions.
- Support to commit recently added tasks and subflows.
- Support to auto handle the conflicts thus enabling seamless pull and push actions in the repository.
- A concept analogous to Git staging is supported for choosing files to push from a committed list along with the Transaction status.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Renamed workflows can be pushed and duplicates are deleted in the remote repository if the workflow with “Deleted” status is selected in staging.
- Confirmation for actions like, Overwrite, First Commit inside Design, and Cloning without selecting any files.
- Source Control Repository connection establishment based on repo name, source control type, and working branch name.
- Support for other dependencies like Helpers, Regex, Hooks, Pages.
- Commit and Push actions for workflow(s) and task(s) can be performed from the Workflow Inventory page as well from the workflow Design studio.

The following actions are **not supported** for Source Control:

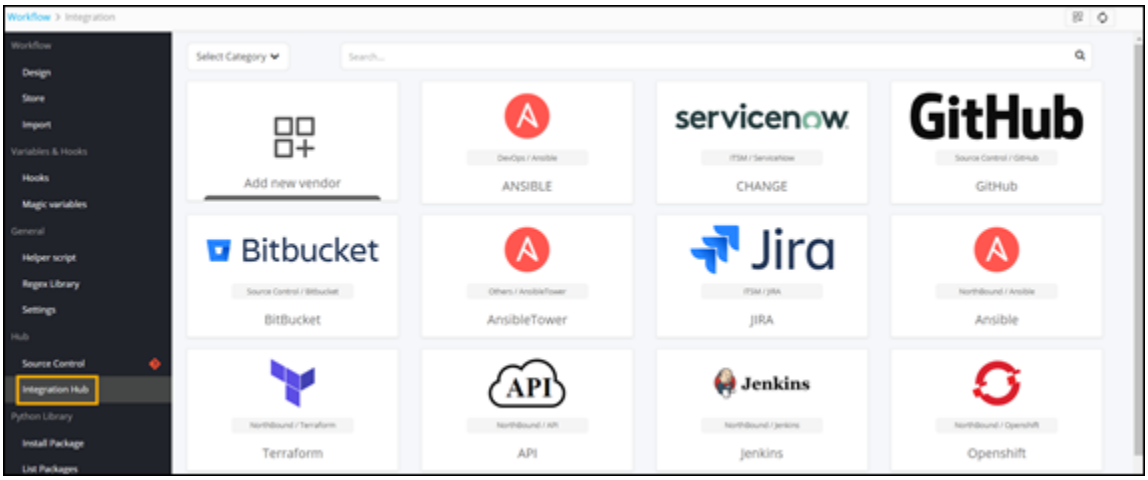
- Option for hiding password in tasks and folder segregation along with filtering field values of exported data in remote is not supported.
- Support is not provided for deleting remote tasks and workflows.

Integration Hub

Integration Hub facilitates integration with different vendors (ITSM, GIT, DevOps, PaaS) and allows you to use the configurations for speedy automation.Relevant vendor configurations, communication endpoints, closure codes and so on must be pre-configured as part of the ITSM vendor configuration plug-in in order to reference them as part of the workflow task(s).

- Provision to select/reference a specific ITSM instance to be integrated with a workflow.
- Provision to perform actions such as create, close, and withdraw an ITSM ticket.
- Provision to get ticket details and validate them as part of the workflow process.
- Provision to push specific configurations and logs as part of the automation process to specific field(s) on an ITSM ticket.
- Provision to get the ITSM ticket details for validation as part of the workflow and change automation process.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



- ITSM Vendor Configuration
- Checking ServiceNow REST API details
- ITSM JSON Configuration
- Using Nested Response in JSON
- Payload Data Mapping
- Validation Parameters
- Tooltip Data Configuration
- Mandatory Headers
- Authorization Configuration
- Using Variables in Visual Workflow Integration

ITSM Vendor Configuration

You can configure the relevant ITSM configuration - communication end points, field mappings, closure codes - required to integrate a workflow automation flow with an IT Service Management (ITSM) tool.

- Provision to configure the ITSM vendor name and the instance (For example, ServiceNow, Redmine, JIRA etc.) which can be referenced within the workflow tasks.
- Provision to define credentials to connect to any ITSM system.
- Provision to define relevant REST API details in order to do the following:
  - Create an ITSM ticket
  - Poll ITSM ticket details

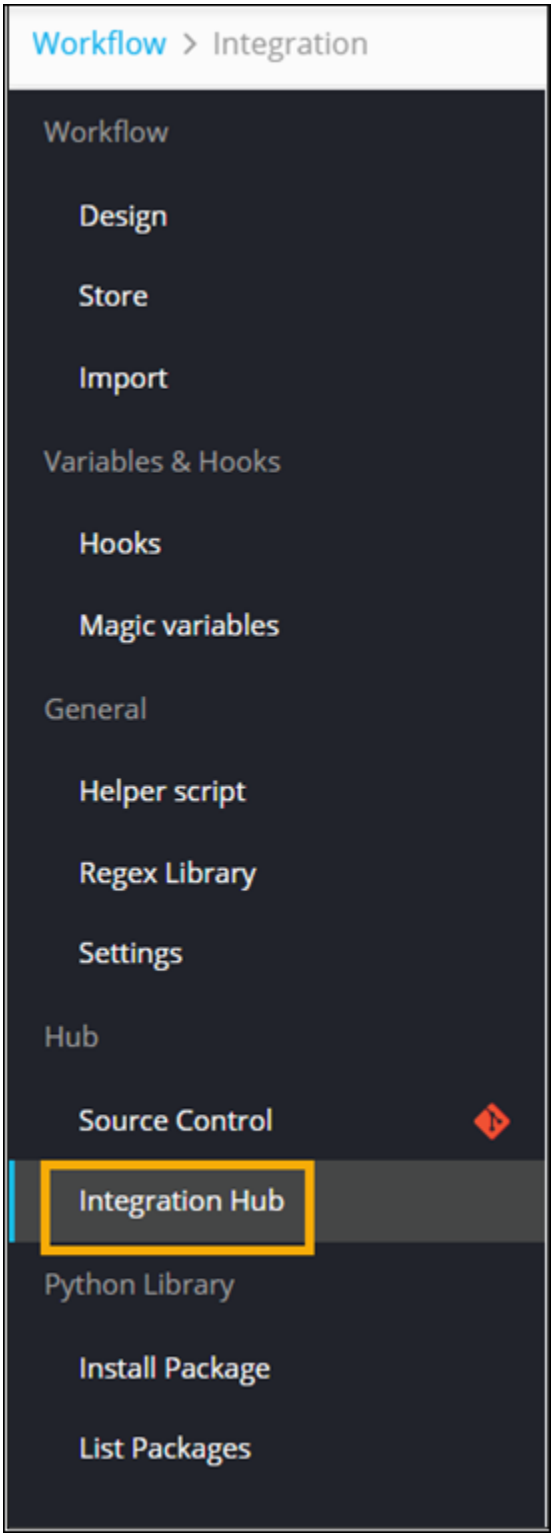
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

  - Update an ITSM ticket
  - Close an ITSM ticket
- Provision to define mapping of custom change ticket closure codes between AppViewX and the ITSM vendor.
- Provision to define custom validation parameters between AppViewX vis-a-vis ITSM system, for example, Device/CI, Implementation date/time, ITSM state/status etc.

To configure a ServiceNow instance:

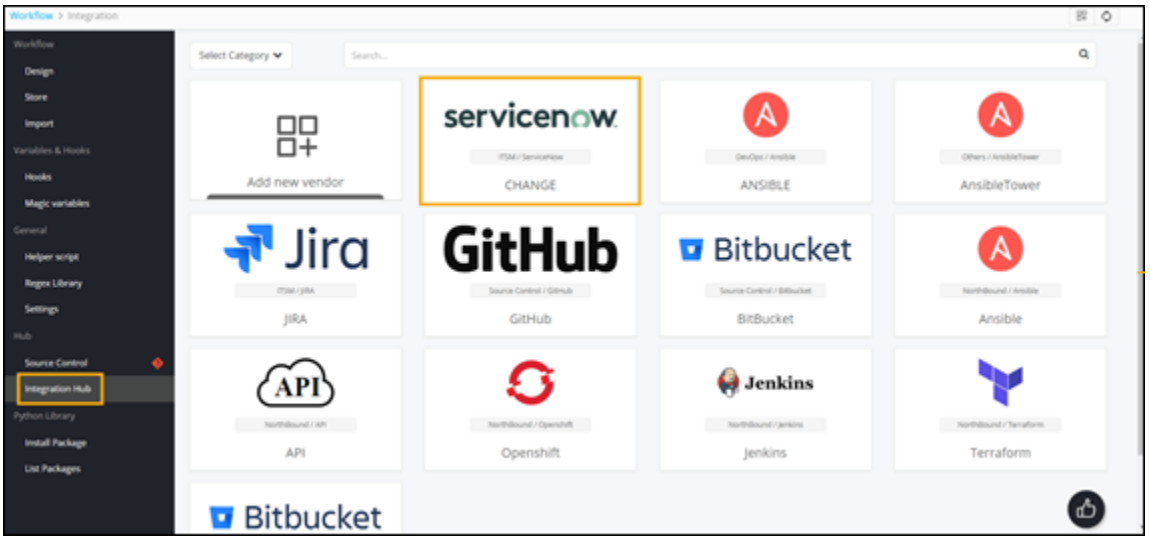
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. On the **Workflow** Inventory page, from the navigation pane on the left, click **Integration Hub**.

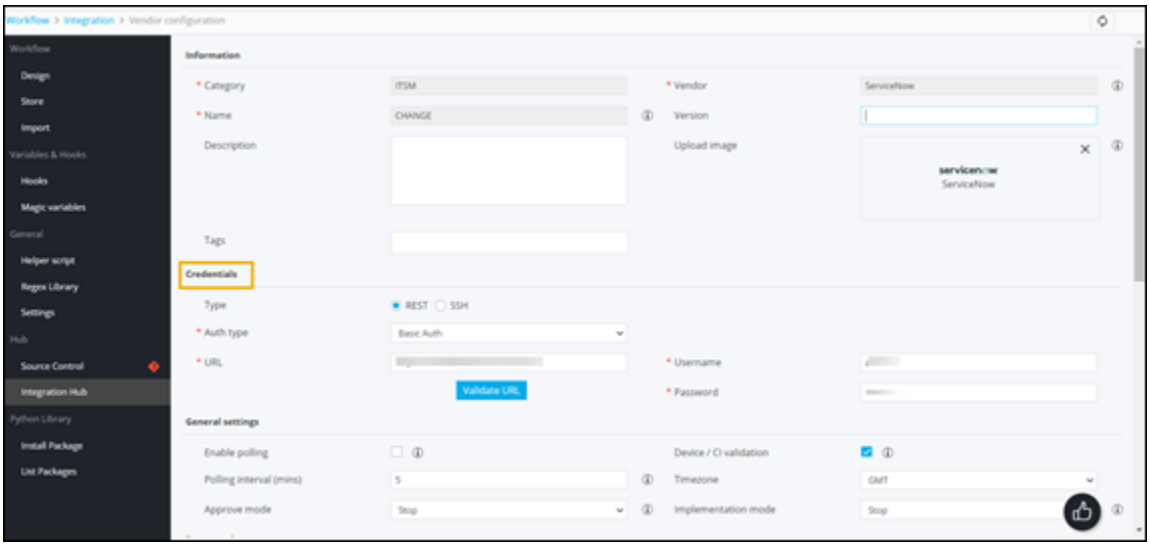


2. On the **Integration** page, click on the **servicenow/change** card.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

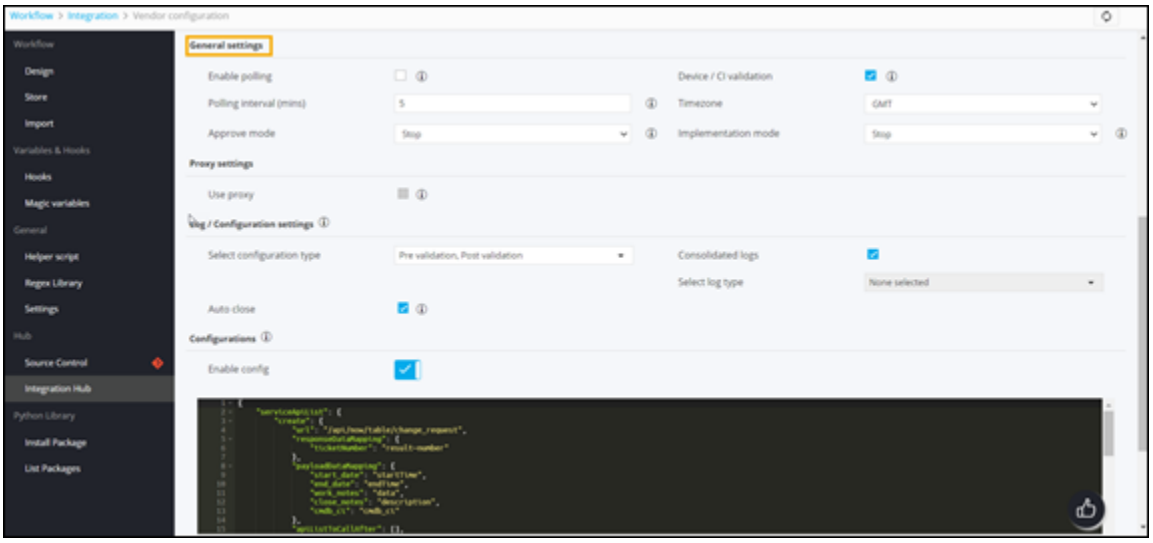


3. On the **Vendor Configuration** page, under **Credentials**, enter the **URL**, **Username**, and **Password**.



4. Under **General Settings**, define the relevant field information.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

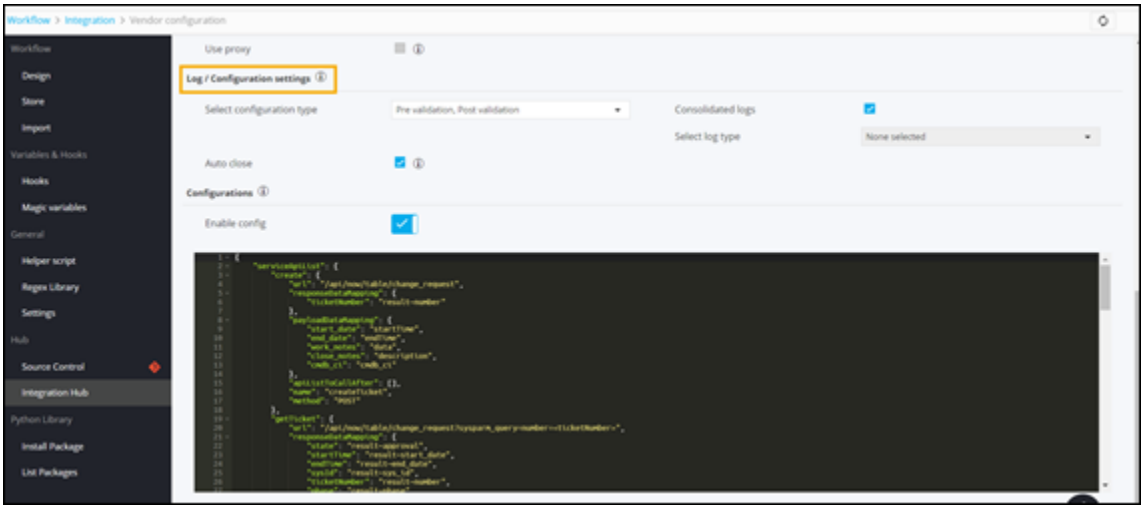


The following table describes the field information in this section:

Field	Description
Enable polling	Allows you to enable periodic polling when integrated with an ITSM system.
Polling Interval (mins)	Allows you to define the frequency of polling interval when integrated with an ITSM system.
Device / CI validation	Allows you to validate for any mismatch between device/CI (configuration item) details between the AppViewX work order and the change ticket at the time of the work order approval and implementation.
Timezone	Allows you to select the timezone from the list.
Approve mode	Allows you to enable either a ‘hard stop’ or ‘override’ (the work order) in the event of any mismatch between the AppViewX work order and the change ticket details at the time of work order approval, peer review.
Implementation mode	Allows you to enable either a ‘hard stop’ or ‘override’ (the work order) in the event of any mismatch between the AppViewX work order and the change ticket details at the time of work order implementation post all peer reviews.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

5. Define the **Log/Configuration** settings.



The following table describes the field information in this section:

Field	Description
Select configuration type	Allows you to enable pushing of configuration commands and/or logs to the ITSM system. For example, pushing config and configurations to the 'journal field' on ServiceNow.
Auto close	Allows you to enable automatic ticket closure or otherwise on the ITSM system once the AppViewX work order is executed.

6. Define relevant API parameters in the JSON configurator.

```
{
 "ServiceNowConfig": {
  "serviceName": "ServiceNow",
  "serviceApiList": {
   "getTicket": {
    "url": "/api/now/table/change_request?sysparm_query=number=<ticketNumber>",
    "responseDataMapping": {
     "state": "result~approval",
     "startTime": "result~start_date",
     "endTime": "result~end_date",
     "sysId": "result~sys_id",
     "ticketNumber": "result~number"
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
 },
 "apiListToCallAfter": [
  "getDeviceList"
 ],
 "name": "getTicket",
 "method": "GET"
},
"getDeviceList": {
 "url": "/api/now/table/task_ci?sysparm_display_value=true&sysparm_query=task=<sysId>",
 "responseDataMapping": {
  "deviceList": "result~ci_item~display_value"
 },
 "name": "getDeviceList",
 "method": "GET"
},
"closeTicket": {
 "url": "/api/now/table/change_request/<sysId>",
 "responseDataMapping": {
  "state": "result~state"
 },
 "payloadDataMapping": {
  "state": "closureCode",
  "close_notes": "comment"
 },
 "name": "closeTicket",
 "method": "PUT"
},
"updateTicket": {
 "url": "/api/now/table/change_request/<sysId>",
 "responseDataMapping": {},
 "payloadDataMapping": {
  "work_notes": "updateData"
 },
 "name": "updateTicket",
 "method": "PUT"
}
},
```


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
"validation": {
 "state": "approved"
},
"tooltipData": {
 "Affected CI's": "deviceList",
 "Start Date": "startDate",
 "End Date": "endDate",
 "Status": "state"
},
"authorizationConfig": {
 "type": "Basic"
},
"mandatoryHeaders": {
 "Content-Type": "application/json"
},
"constants": {
 "ticketCloseStates": {
  "Success": 3,
  "Not implemented": 4,
  "Withdrawn": 7
 },
 "dateFormat": "yyyy-MM-dd HH:mm:ss"
 }
}
}
```

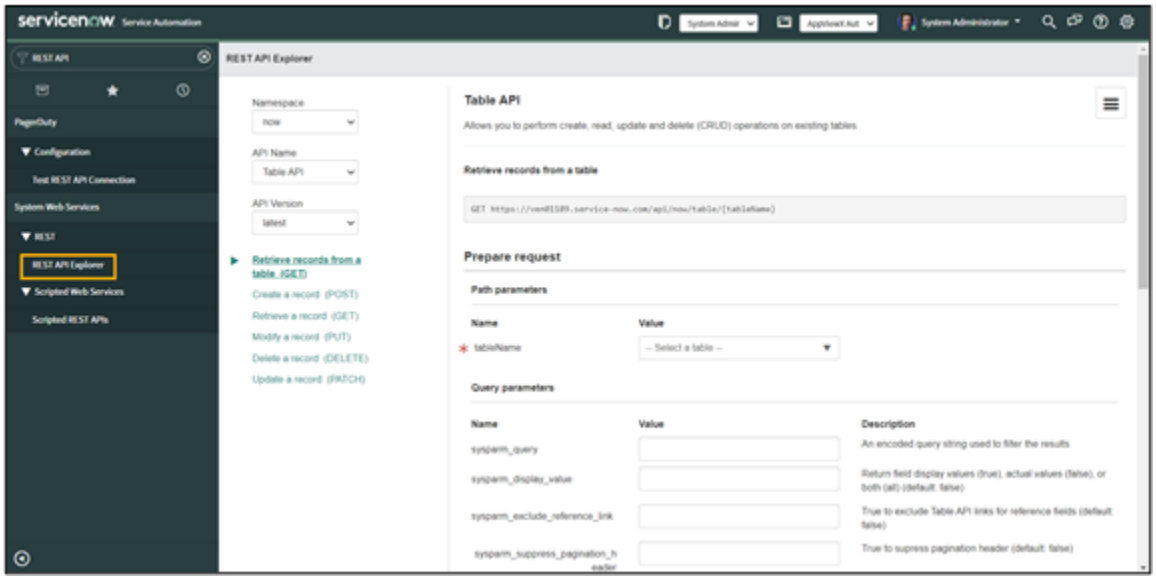

7. Click **Save**.

Checking ServiceNow REST API details

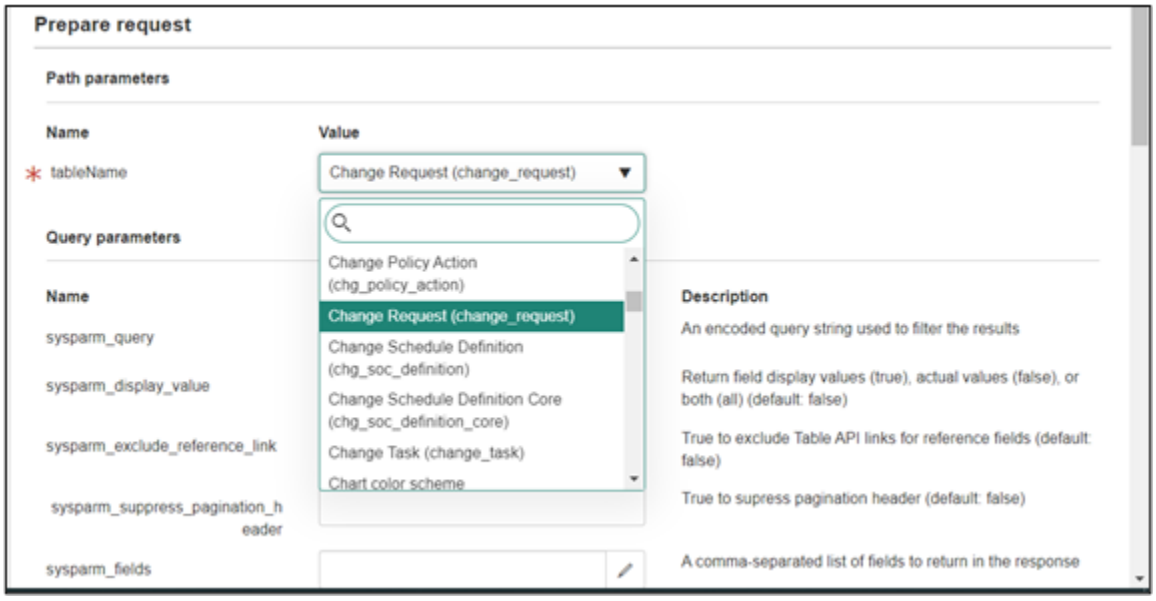
To retrieve details using REST API explorer for a given Change ticket on ServiceNow:

1. Login to ServiceNow using your credentials.
2. Search or navigate to **REST API Explorer**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



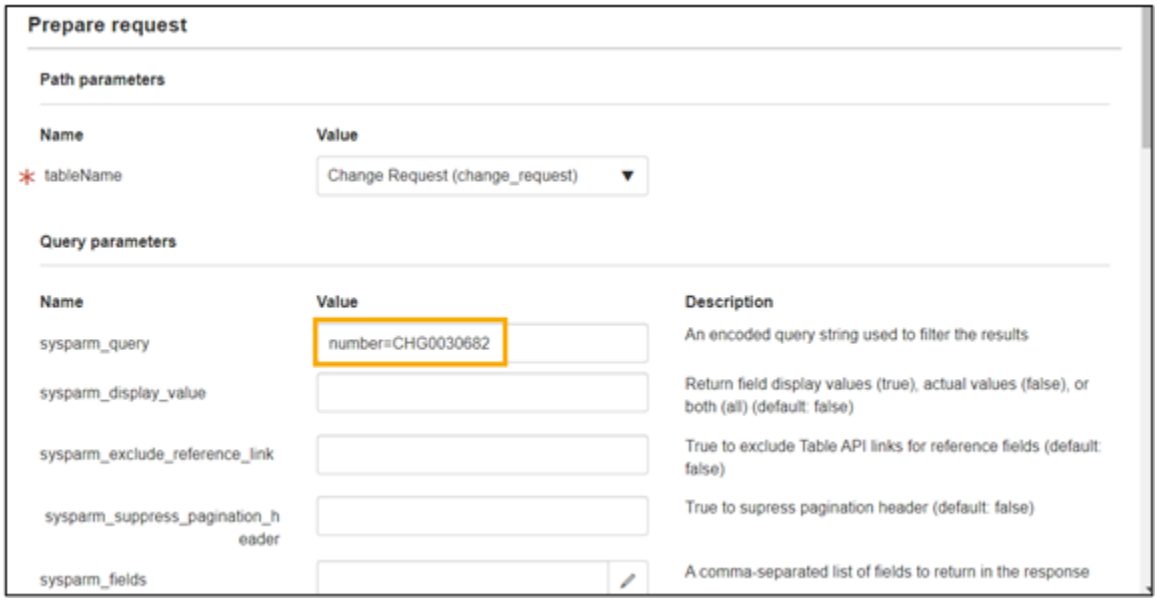
3. Select **tableName** value from the dropdown list.



4. Enter the value for **sysparm_query** (number=<change ticket>).

   For example, number=CHG0030682

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



5. Get ‘sysID’ API response body for the change ticket.

```
{
 "result": [
  {
   "u_glide_date_1": "",
   "parent": "",
   "reason": "",
```


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
"made_sla": "true",
"backout_plan": "",
"watch_list": "",
"u_integer_4": "",
"upon_reject": "cancel",
"sys_updated_on": "2016-11-11 07:58:35",
"type": "Comprehensive",
"conflict_status": "Not Run",
"approval_history": "",
"number": "CHG0030682",
"test_plan": "",
"sys_updated_by": "admin",
"opened_by": {
  "link": "https://ven01189.service-now.com/api/now/table/sys_user/6816f79cc0a8016401c5a33be04be441",
  "value": "6816f79cc0a8016401c5a33be04be441"
},
"user_input": "",
"requested_by_date": "",
"sys_created_on": "2016-11-11 07:08:22",
"sys_domain": {
  "link": "https://ven01189.service-now.com/api/now/table/sys_user_group/global",
  "value": "global"
},
"state": "4",
"sys_created_by": "admin",
"knowledge": "false",
"order": "",
"phase": "requested",
"closed_at": "2016-11-11 07:58:35",
"cmdb_ci": {
  "link": "https://ven01189.service-now.com/api/now/table/cmdb_ci/0ca262a80f4fa200c3e6cd8ce1050e68",
  "value": "0ca262a80f4fa200c3e6cd8ce1050e68"
},
"delivery_plan": "",
"impact": "3",
"active": "false",
"review_comments": "kk_vs1",
```


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
"work_notes_list": "",
"business_service": "",
"priority": "2",
"time_worked": "",
"cab_recommendation": "",
"expected_start": "",
"production_system": "false",
"rejection_goto": "",
"opened_at": "2016-11-11 07:08:22",
"review_date": "",
"business_duration": "",
"group_list": "",
"requested_by": {
  "link": "https://ven01189.service-now.com/api/now/table/sys_user/kk",
  "value": "kk"
},
"work_end": "",
"change_plan": "",
"phase_state": "open",
"approval_set": "",
"cab_date": "",
"wf_activity": "",
"work_notes": "",
"implementation_plan": "",
"u_table_name_1": "",
"end_date": "2016-11-12 00:00:00",
"short_description": "Create VS on UAT",
"correlation_display": "",
"delivery_task": "",
"work_start": "",
"assignment_group": "",
"additional_assignee_list": "",
"outside_maintenance_schedule": "false",
"description": "Create VS",
"calendar_duration": "",
"close_notes": "Closing the ticket from AppViewX.",
"sys_class_name": "change_request",
```

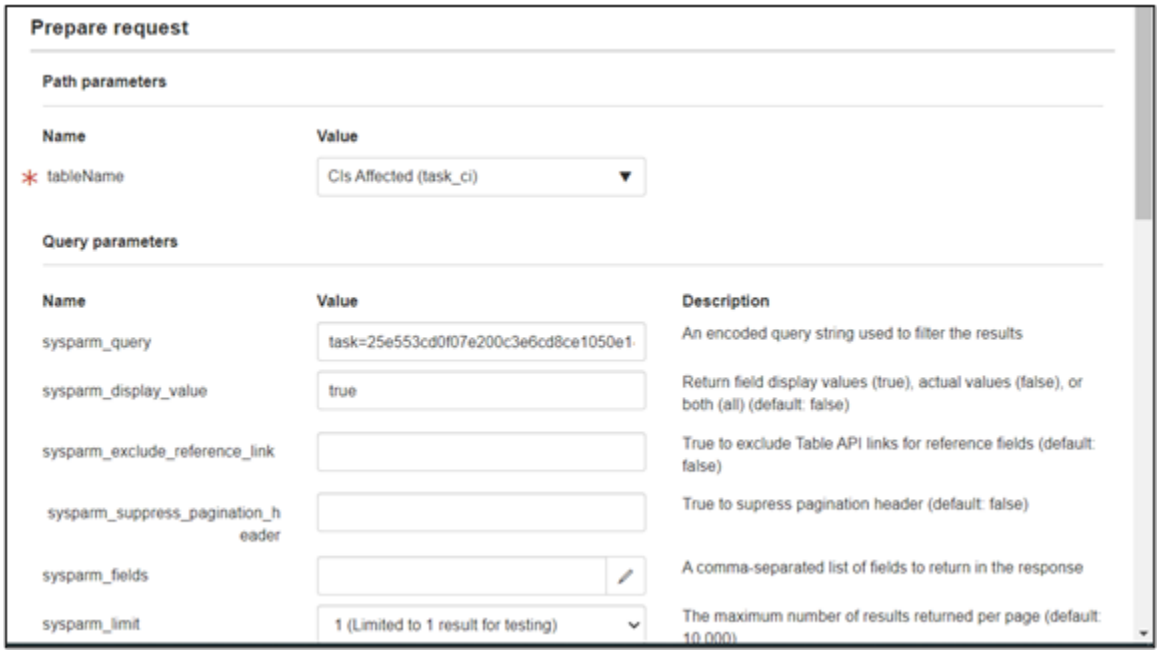

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
    "closed_by": {
      "link": "https://ven01189.service-now.com/api/now/table/sys_user/6816f79cc0a8016401c5a33be04be441",
      "value": "6816f79cc0a8016401c5a33be04be441"
    },
    "follow_up": "",
    "sys_id": "25e553cd0f07e200c3e6cd8ce1050e14",
    "contact_type": "phone",
    "urgency": "3",
    "scope": "3",
    "company": "",
    "justification": "",
    "reassignment_count": "0",
    "review_status": "3",
    "activity_due": "",
    "assigned_to": "",
    "start_date": "2016-11-07 00:00:00",
    "u_field_name_3": "",
    "u_field_name_1": "",
    "comments": "",
    "u_field_name_2": "",
    "u_journal_1": "",
    "approval": "approved",
    "sla_due": "",
    "comments_and_work_notes": "",
    "due_date": "",
    "sys_mod_count": "4",
    "sys_tags": "",
    "conflict_last_run": "",
    "escalation": "0",
    "upon_approval": "proceed",
    "correlation_id": "",
    "location": "",
    "risk": "4",
    "category": "Hardware"
  }
 ]
}
```


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

6. Enter or select the following field information to get ‘CI’ specific details.

tableName	CIs Affected (task_ci)
sysparm_query	task=’sysID’ (task=25e553cd0f07e200c3e6cd8ce1050e14)
sysparm_display_value	true



Response for sysid from task_ci table.


```
{
 "result": [
  {
   "sys_id": "39e593010f47e200c3e6cd8ce1050e8a",
   "sys_updated_by": "admin",
   "task": {
    "display_value": "CHG0030682",
    "link": "https://ven01189.service-now.com/api/now/table/task/25e553cd0f07e200c3e6cd8ce1050e14"
   },
   "applied": "false",
   "sys_created_on": "2016-11-11 07:08:23",
   "xml": "",
   "sys_mod_count": "0",
```


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
    "sys_updated_on": "2016-11-11 07:08:23",
    "sys_tags": "",
    "applied_date": "",
    "ci_item": {
      "display_value": "117",
      "link": "https://ven01189.service-now.com/api/now/table/cmdb_ci/0ca262a80f4fa200c3e6cd8ce1050e68"
    },
    "sys_created_by": "admin"
  }
 ]
}
```


ITSM JSON Configuration

The IT Service Management (ITSM) JavaScript Object Notation (JSON) configurator is a generic method through which an ITSM vendor that supports a JSON based REST communication can integrate with AppViewX. This allows AppViewX to establish communication with an external tool and achieve the necessary change management integration.

The key components of the ITSM JSON configuration are:

- Application Programming Interface (API) Definitions
- Validation parameter configuration
- Tooltip data configuration
- Mandatory headers configuration
- Authorization configuration
- Constants
  - Change ticket closure states.
  - Date format used by the vendor

The API definition can have the following configurations:

- **Name**: The API Name
- **Method**: The HTTP method - **GET, POST, PUT, DELETE**
- **URL**: The relative URL for the REST endpoint of the API
- responseDataMapping
- payloadDataMapping

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- apiListToCallAfter
- apiListToCallBefore
- ITSM APIs
- Defining ITSM Instance
- URL and Response Data Mapping

ITSM APIs

The following APIs need to be defined within the configurator in order to integrate Change management vendor with AppViewX:

- **Create ticket**: This API is used to create an ITSM change ticket and pass some basic ticket details defined in the payloadDataMapping.
- **Get ticket**: This API is used to fetch the ITSM change ticket for a given RFC ID (Request for change). All the fields mapped in the responseDataMapping will be stored internally in AVX for reference.
- **Close ticket**: This API is used to close the ITSM change ticket based on the AppViewX workorder status. The corresponding state in the ITSM system has to be mapped in the constants, under ticketCloseStates.
  - The value of “closureCode” is picked from ticketCloseStates based on the workorder status.

```
"close": {
  "url": "/api/now/table/change_request/<sysId>",
  "responseDataMapping": {
   "state": "result~state"
  },
  "payloadDataMapping": {
   "state": "closureCode",
   "close_notes": "comment"
  },
  "name": "closeTicket",
  "method": "PUT"
 },
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- **Update ticket**: This API is used to update data back to the change ticket. Commonly the update data are workorder configurations (Implementation, Rollback, Pre-validation, and Post-validation) and workorder logs.
  - Specific configurations such as Implementation, Pre-validation, and Post-validation can be mapped against the relevant ITSM field, and pushed.

```
“update”: {
  "url": "/api/now/table/change_request/<sysId>",
  "responseDataMapping": {},
  "payloadDataMapping": {
   "work_notes": "updateData"
  },
  "name": "update",
  "method": "PUT"
 },
 "updateConfig": {
  "responseDataMapping": {},
  "payloadDataMapping": {
   "user_input": [
    "rollback_config"
   ],
   "close_notes": [
    "postvalidation_config"
   ],
   "work_notes": [
    "prevalidation_config"
   ]
  },
  "url": "/api/now/table/change_request/<sysId>",
  "name": "updateConfig",
  "method": "PUT"
 },
 "updateLog": {
  "responseDataMapping": {},
  "payloadDataMapping": {
   "user_input": [
    "rollback"
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
 ],
 "close_notes": [
  "postvalidation"
 ],
 "work_notes": [
  "prevalidation"
 ]
},
"url": "/api/now/table/change_request/<sysId>",
"name": "updateLog",
"method": "PUT"
```

If the API for withdrawal of the ticket is different from the close API, it can be defined under the name “withdrawTicket”.

```
"method": "GET"
 },
 "withdraw": {
  "responseDataMapping": {
   "status": "status"
  },
  "payloadDataMapping": {
   "change_number": "ticketNumber",
   "comment": ""
  },
  "url": "/withdraw",
  "name": "withdrawTicket",
  "method": "POST"
 },
```

Defining ITSM Instance

Define a unique ITSM instance.

Note: For more information, refer to the section on ITSM Vendor Configuration.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

URL and Response Data Mapping

The relative URL for the REST endpoint of the API and the relevant query parameters for the URL can be specified within <> representation.

For example, In the getTicket API only <ticketNumber> can be used. But for other API any mapped data can be used with the <> notation.

- Data that can be Mapped
- Response Data Mapping

Data that can be Mapped

The following keys can be mapped in the ‘get ticket’ API for use in any other API or for other internal purposes:

- deviceList
- phase
- state
- endTime
- startTime
- ticketNumber

In addition any vendor specific ticket data can be mapped with a custom key and can be used under validation config or mapped in the payload data of an API definition.

Note: Any data that is to be mapped from the external system has to be mapped in the getTicket API. This is the entry point for any ticket data from the external system into AVX.

Response Data Mapping

This is a guideline for the data to be mapped for internal use by AppViewX based on the JSON response obtained from the ITSM system. Response data mapping (key-

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

value pair) is used to define all the relevant fields from the ITSM system into AppViewX.

Configuration command

```
{
    "serviceApiList": {
        "create": {
            "url": "/api/now/table/change_request",
            "responseDataMapping": {
                "ticketNumber": "result~number"
            },
            "payloadDataMapping": {
                "start_date": "startTime",
                "end_date": "endTime",
                "work_notes": "data",
                "close_notes": "description",
                "cmdb_ci": "cmdb_ci"
            },
            "apiListToCallAfter": [],
            "name": "createTicket",
            "method": "POST"
```

Configuration command

```
{
    "ServiceNowConfig": {
        "serviceName": "ServiceNow",
        "serviceApiList": {
            "getTicket": {
                "url": "/api/now/table/u_vip_create?sysparm_query=u_number=<ticketNumber>",
                "responseDataMapping": {
                    "state": "result~u_approval",
                    "startTime": "result~u_start_date",
                    "endTime": "result~u_end_date",
                    "sysId": "result~sys_id",
                    "ticketNumber": "result~u_number"
                },
                "apiListToCallAfter": [
                    "getDeviceList"
                ],
                "name": "getTicket",
                "method": "GET"
```

```
{
 "ServiceNowConfig": {
  "serviceName": "ServiceNow",
  "serviceApiList": {
   "getTicket": {
    "url": "/api/now/table/u_vip_create?sysparm_query=u_number=<ticketNumber>",
    "responseDataMapping": {
     "state": "result~u_approval",
     "startTime": "result~u_start_date",
     "endTime": "result~u_end_date",
     "sysId": "result~sys_id",
     "ticketNumber": "result~u_number"
    },
    "apiListToCallAfter": [
     "getDeviceList"
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- The key on the left indicates the values that are used in AppViewX as part of the provisioning, ticket validation process.
- The values on the right indicate the response received from an external tool and used within AppViewX as part of the provisioning, ticket validation process.

Using Nested Response in JSON

If the data is in the root of the JSON then the entire key is specified. However, if the data in the response is nested within another JSON or array, then a tilde '~' symbol is used to denote the path to the data to be mapped.

- Nested JSON response from ServiceNow referenced in AppViewX


```
{
 "result": [
  {
   "u_glide_date_1": "",
   "parent": "",
   "reason": "",
   "u_start_date": "2016-11-21 07:58:35",
   "u_end_date": "2016-11-23 07:58:35",
   "backout_plan": "",
   "u_approval": " Approved",
   "upon_reject": "cancel",
   "sys_updated_on": "2016-11-11 07:58:35",
   "type": "Comprehensive",
   "conflict_status": "Not Run",
   "approval_history": "",
   "u_number": "CHG0030682",
   "test_plan": "",
   "sys_updated_by": "admin",
   "opened_by": {
    "link": "https://ven01189.service-now.com/api/now/table/sys_user/6816f79cc0a8016401c5a33be04be441",
    "value": "6816f79cc0a8016401c5a33be04be441"
   }]
}
```


- Mapping response from ServiceNow into AppViewX using tilde '~'

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
{
 "ServiceNowConfig": {
  "serviceName": "ServiceNow",
  "serviceApiList": {
   "getTicket": {
    "url": "/api/now/table/u_vip_create?sysparm_query=u_number=<ticketNumber>",
    "responseDataMapping": {
     "state": "result~u_approval",
     "startTime": "result~u_start_date",
     "endTime": "result~u_end_date",
     "sysId": "result~sys_id",
     "ticketNumber": "result~u_number"
    },
    "apiListToCallAfter": [
     "getDeviceList"
```


Payload Data Mapping

These are attributes required to be sent to an ITSM system as part of the payload data (via REST)

```
"closeTicket": {
   "url": "/api/now/table/u_vip_create/<sysId>",
   "responseDataMapping": {
    "state": "result~u_state"
   },
   "payloadDataMapping": {
```


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
  "u_state": "closureCode",
  "u_notes": "comment"
 },
```

- The ‘key’ on the left indicates the specific field values on the ITSM system.
- The ‘values’ on the right indicate the specific values with which the fields on ITSM have to be updated by AppViewX
- API list to call Before (Optional)
- API list to call After (Optional)

API list to call Before (Optional)

This is a list of names of defined API which will be called prior to making a call to the actual URL defined in this API definition.

For example, API to get the session ID and use it as part of subsequent web service calls, payload and so on.

API list to call After (Optional)

This is a list of names of defined API that will be called after making a call to the actual URL defined in this API definition.

Example API Definition:

```
"getTicket": {
 "url": "/api/now/table/u_vip_create?sysparm_query=u_number=<ticketNumber>",
 "responseDataMapping": {
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

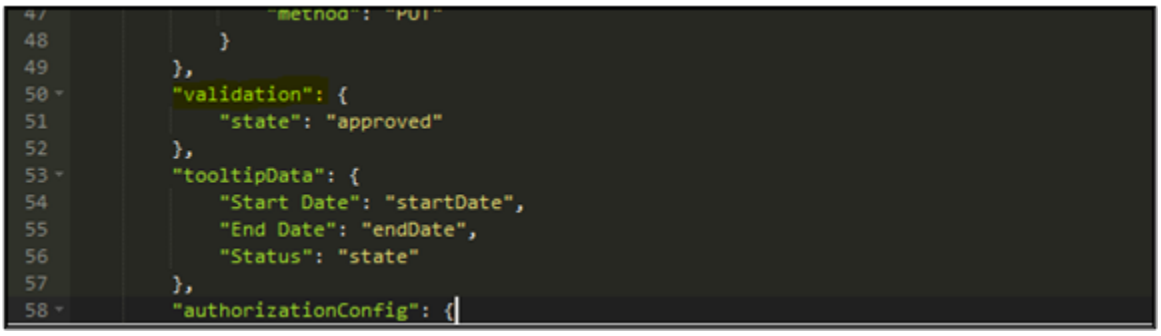
```
 "state": "result~u_approval",
 "startTime": "result~u_start_date",
 "endTime": "result~u_end_date",
 "sysId": "result~sys_id",
 "ticketNumber": "result~u_number"
},
"apiListToCallAfter": ["getDeviceList"],
"name": "getTicket",
"method": "GET"
},
"getDeviceList": {
"url": "/api/now/table/task_ci?sysparm_display_value=true&sysparm_query=task=<sysId>",
"responseDataMapping": {
 "deviceList": "result~ci_item~display_value"
},
```

Validation Parameters

This is a set of JSON key value pairs that define what attributes of a change ticket needs to be validated within AppViewX. Any custom attributes can be defined as part of the validation. The values on the left side of the assignment denote the attribute to be validated and the right side of the assignment denotes the expected value for the parameter to be considered as valid.

Note: In case an attribute that is not mapped to the system is validated, these will be ignored and will be treated as invalid. The validation message on the GUI will be displayed accordingly.

Example validation parameter config:

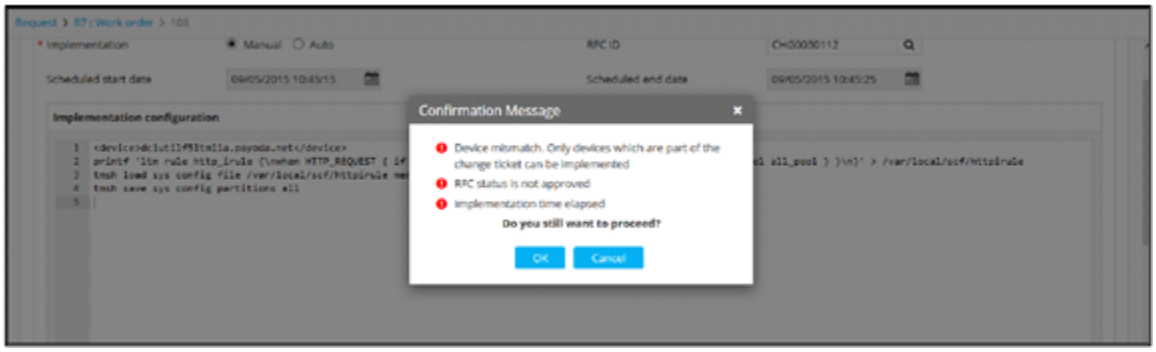


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
"name": "updateTicket",
  "method": "PUT"
 }
},
 "validation": {
 "state": "approved"
},
 "tooltipData": {
 "Start Date": "startDate",
 "End Date": "endDate",
```

• Validation message sample

In this case, AppViewX executes a work order based on the validation parameters. For example, the state of the change ticket on the ITSM tool must be ‘approved’, Device/CI must be valid, Change window must be valid, and so on.



Tooltip Data Configuration

This placeholder gives you the flexibility to customize the tooltip data that is to be displayed in the workorder review page against a specific work order (on hovering over the RFC ID field).

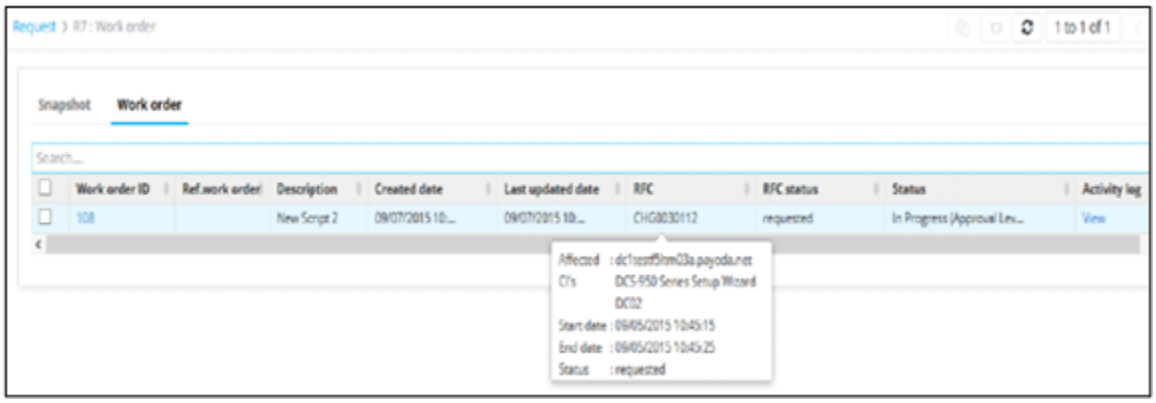
The left side of the assignment denotes the Label to be displayed in the tooltip, and the right side of the assignment denotes the data that is to be shown in the tooltip.

Example tooltip config:

```
"tooltipData":{
    "Affected CI's":"deviceList",
    "Start Date":"startDate",
    "End Date":"endDate",
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
    "Status":"state"
}
```



Mandatory Headers

These are key-value pairs that are needed for every HTTP request sent to the defined APIs. Support is available for “Content-type”:”application/json”. There can be additional headers depending on the system.

For example:

```
"mandatoryHeaders":{
    "Content-Type":"application/json"
}
```

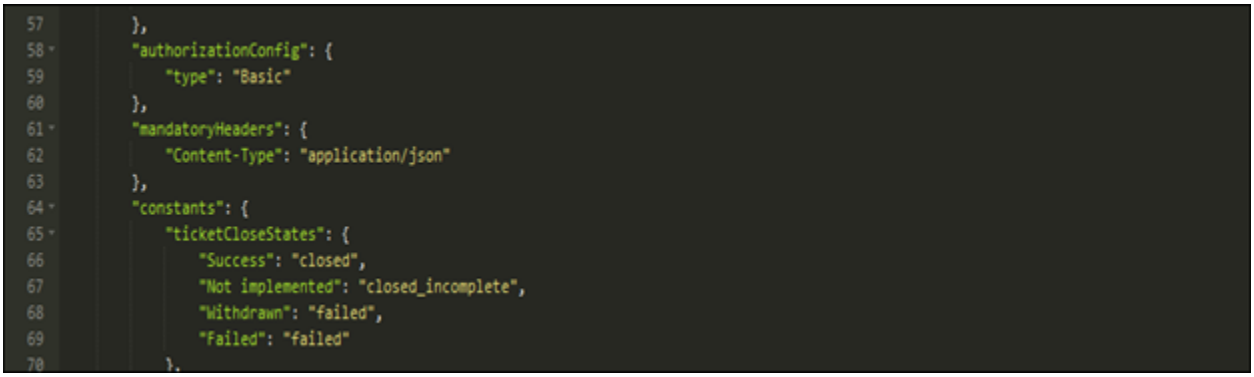
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Authorization Configuration

This part of the configuration contains the authentication details to be performed in order to communicate with the ITSM system. Support is available for “no authentication” and “BASIC authentication”. This is to be specified in the “type” key under in configuration.

For example:

```
"authorizationConfig":{
        "type":"Basic"
    }
```



- Ticket close states: A placeholder to map the AppViewX work order states - Completed, Failed, Not Implemented - with the appropriate ITSM change ticket closure states. This can be defined either as a String or Integer format depending on the ITSM system.
- Vendor date format: Date format is mandatory for AppViewX to interpret the date values from the ITSM system.

Example: Ticket closure states in integer format

```
"constants":{
            "ticketCloseStates":{
             "Success":3,
             "Not implemented":4,
             "Withdrawn":7
      },
     "dateFormat":"yyyy-MM-dd HH:mm:ss"
    }
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Example: Ticket closure states in string format

```
"constants":{
            "ticketCloseStates":{
             "Success":3,
             "Not implemented":4,
             "Withdrawn":7
    },
    "dateFormat":"yyyy-MM-dd HH:mm:ss"
  }
```

Using Variables in Visual Workflow Integration

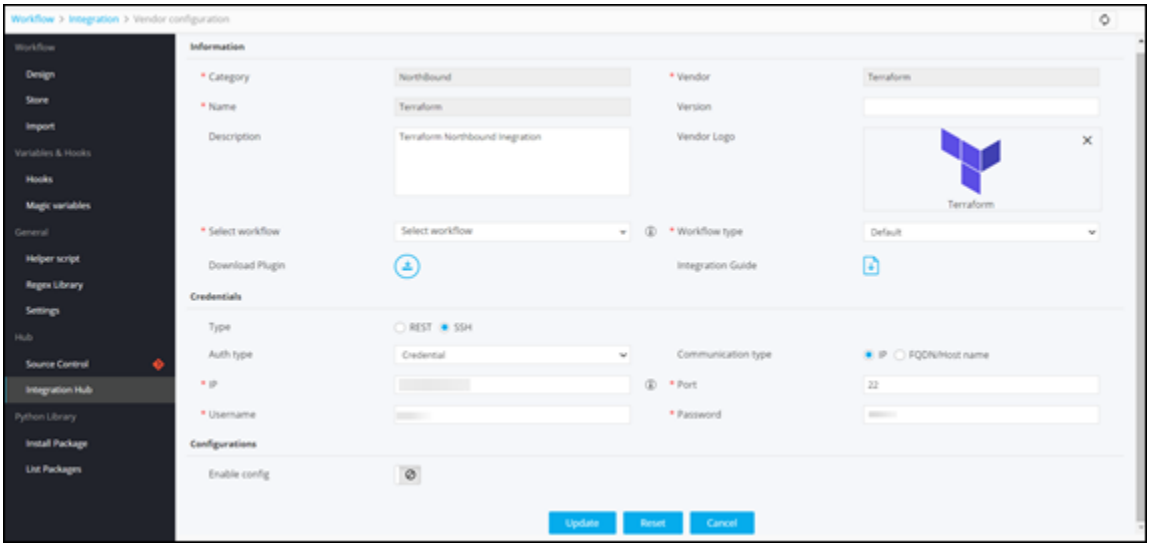
By integrating Visual Workflow with third-party vendors such as Ansible, Bitbucket, Terraform and so on, you can define and use variables for your automation workflows.

The following syntax must be used to refer integration variables within a workflow:

`<%Integrations.<integration name>.<key name>%>`

To refer a Terraform integration variable, such as the IP address, in any task within the workflow:

1. On the Workflow Inventory page, from the navigation pane on the left, click **Integration Hub**.
2. On the **Integration** page, click **Terraform**.
3. Enter the field information to configure the Terraform integration.



4. To refer any of the configurations, for example the IP address, as a variable in any task within a workflow, use the syntax: `<%Integrations.<Terraform>.<key name>%>`

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Python Library

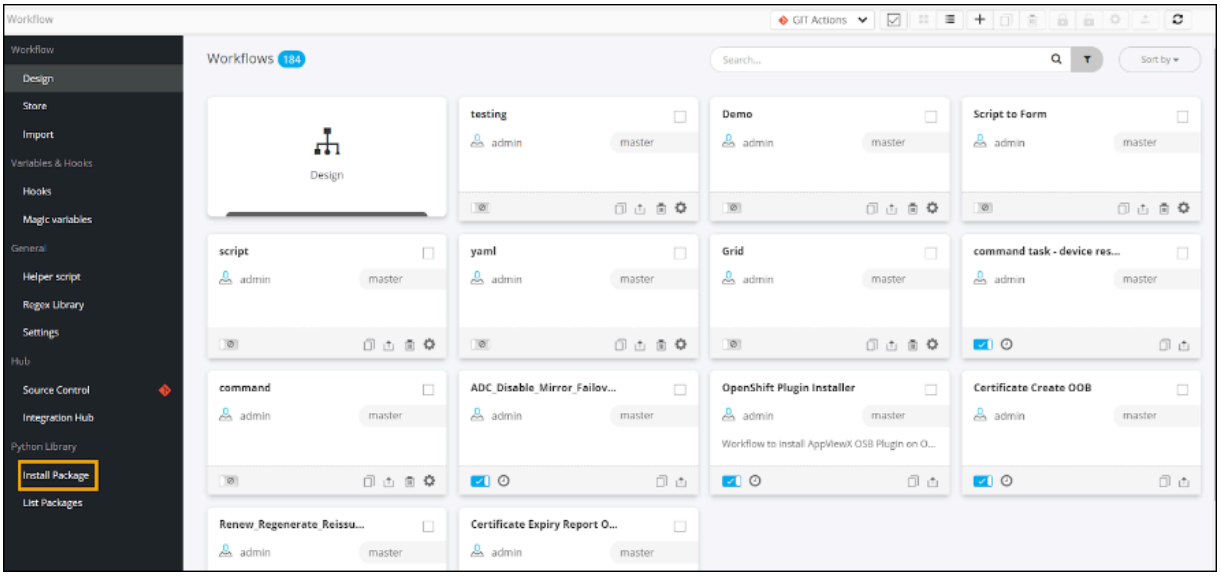
This feature allows you to explore the inventory of Python packages available within AppViewX and also install a Python package from a Python repository, such as Python Package Index (PyPi).

- Installing Python Package
- Python Package List

Installing Python Package

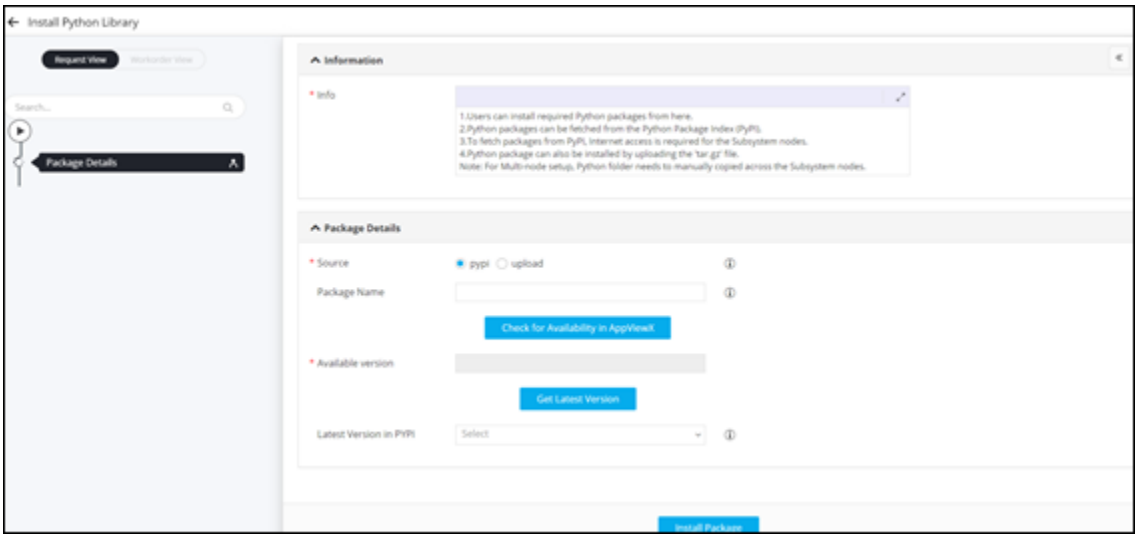
To trigger the workflow for dynamically installing a Python package from Pypi:

1. On the Workflow Inventory page, from the navigation pane on the left, under **Python Library**, click **Install Package**.

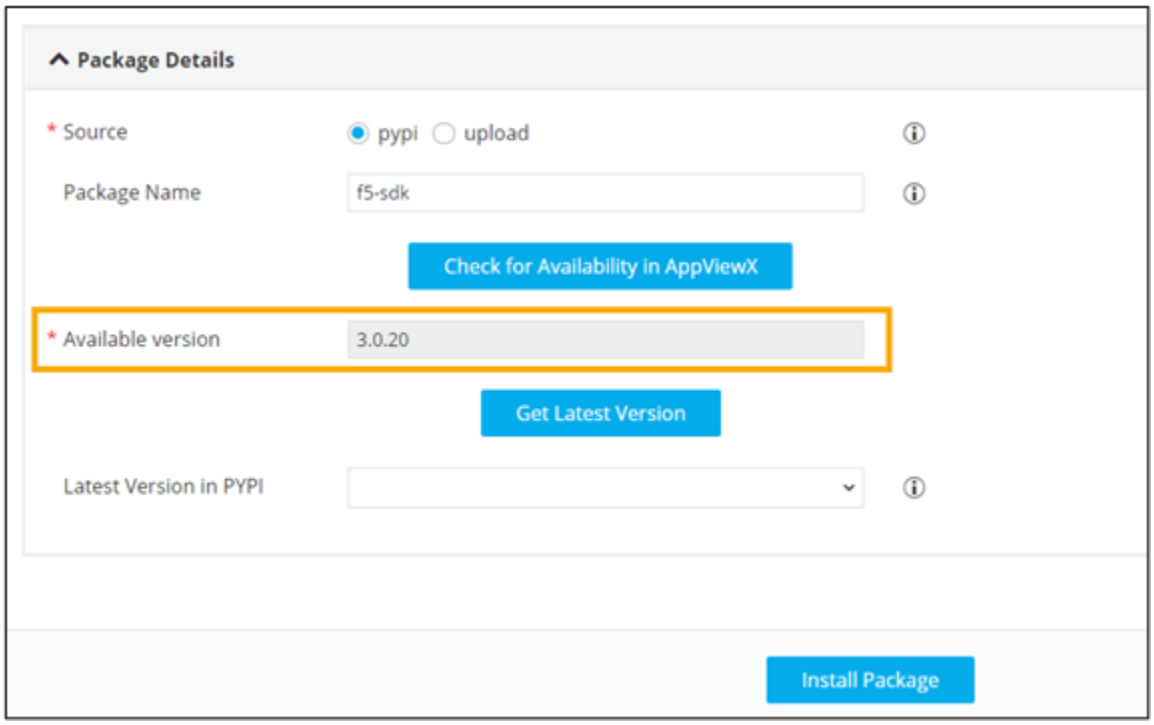


The **Install Python Library** page is displayed.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

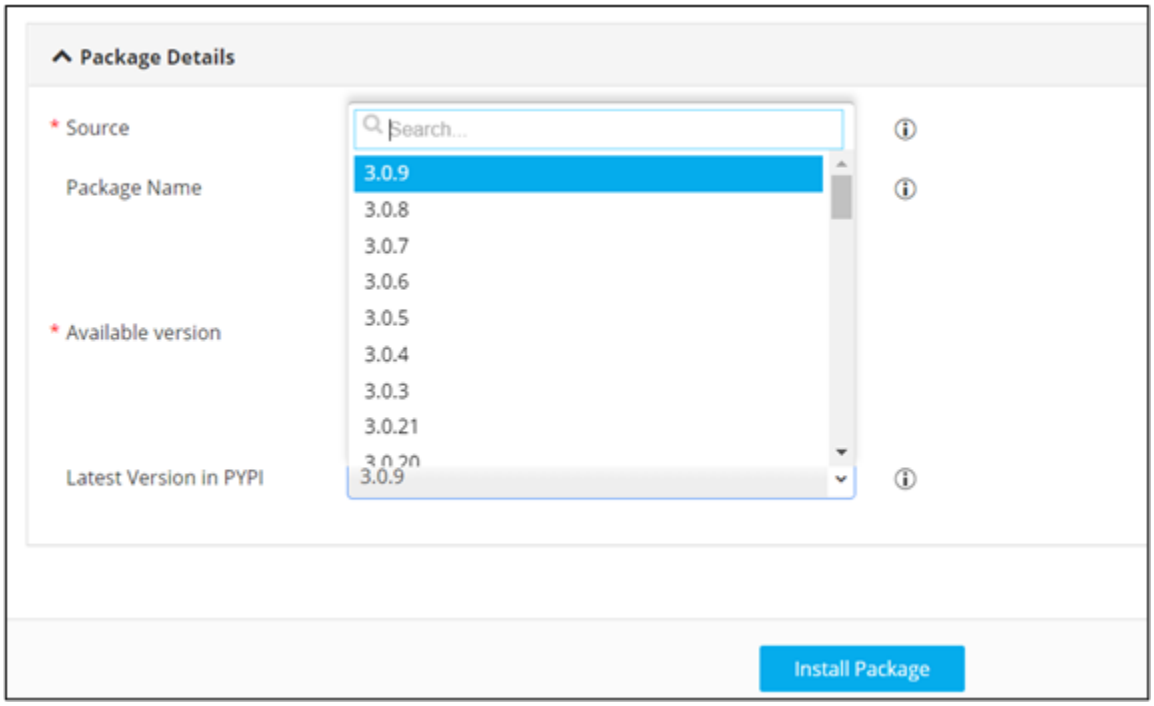


2. On the **Install Python Library** page, under **Package Details**, select **Source** as **pypi**.
3. Enter the name of the Python package that you wish to install, for example, f5-sdk.
4. To check if the package is already available in AppViewX, click **Check for Availability in AppViewX**. The **Available version** field is populated with the name of the Python package available in the AppViewX Python library.



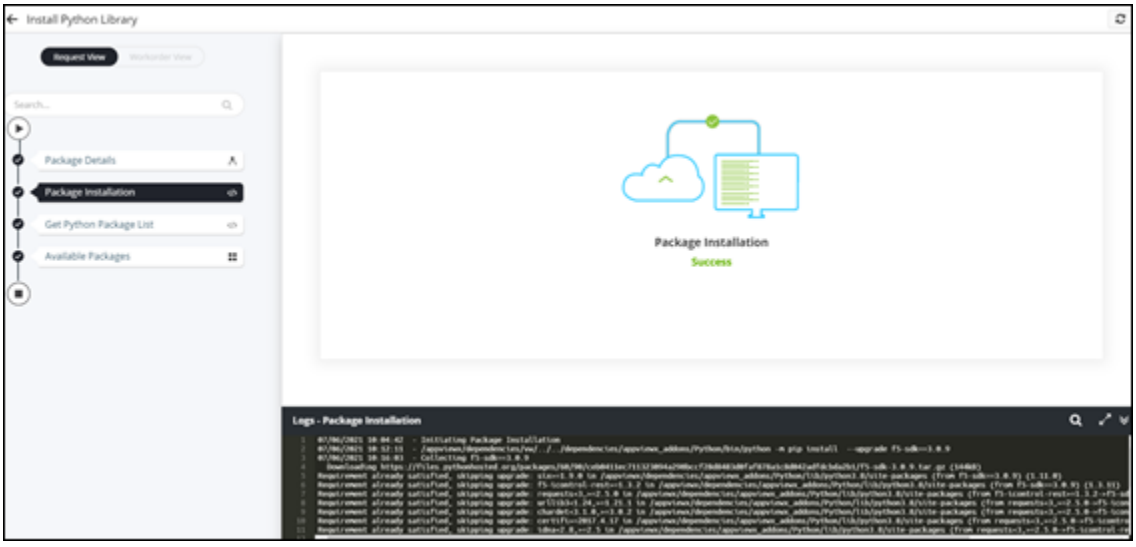
5. To get the latest version of the package from pypi, click **Get Latest Version**.
6. Select the **Latest version in pypi**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

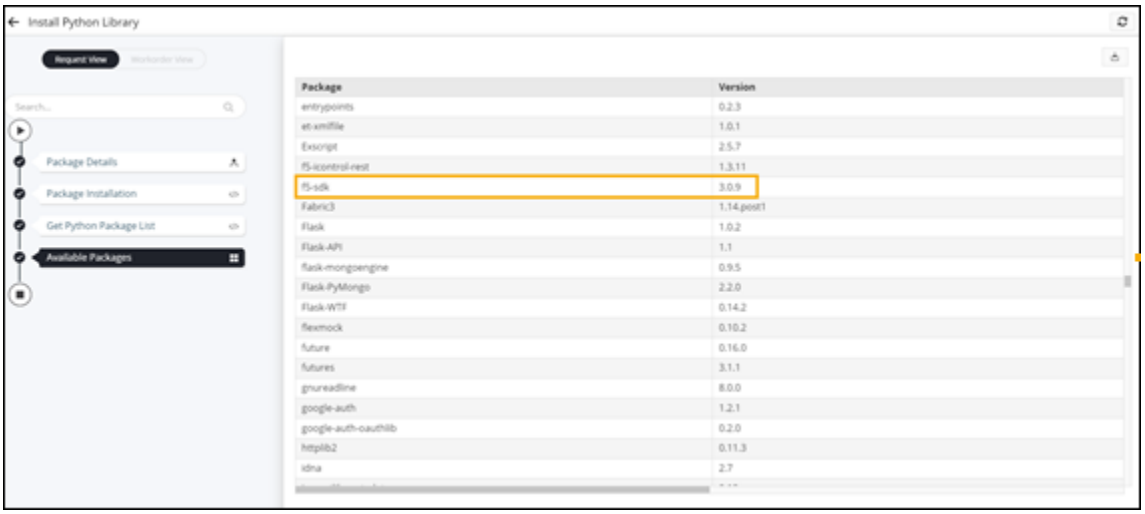


7. Click **Install Package**.

   The selected Python package is installed successfully.



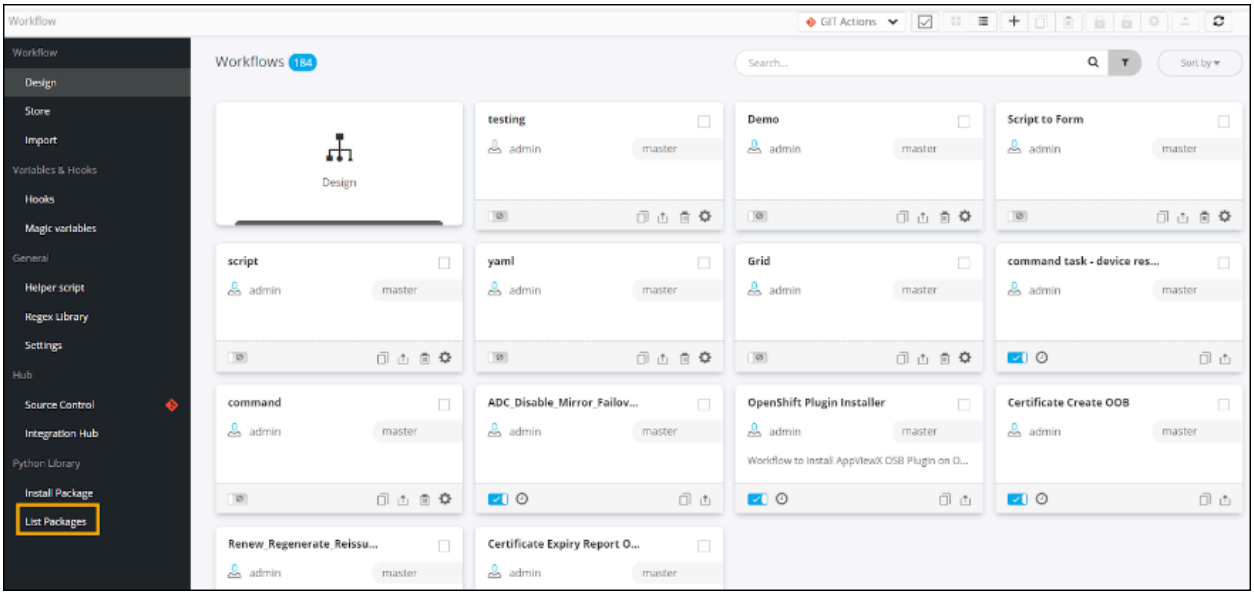
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Python Package List

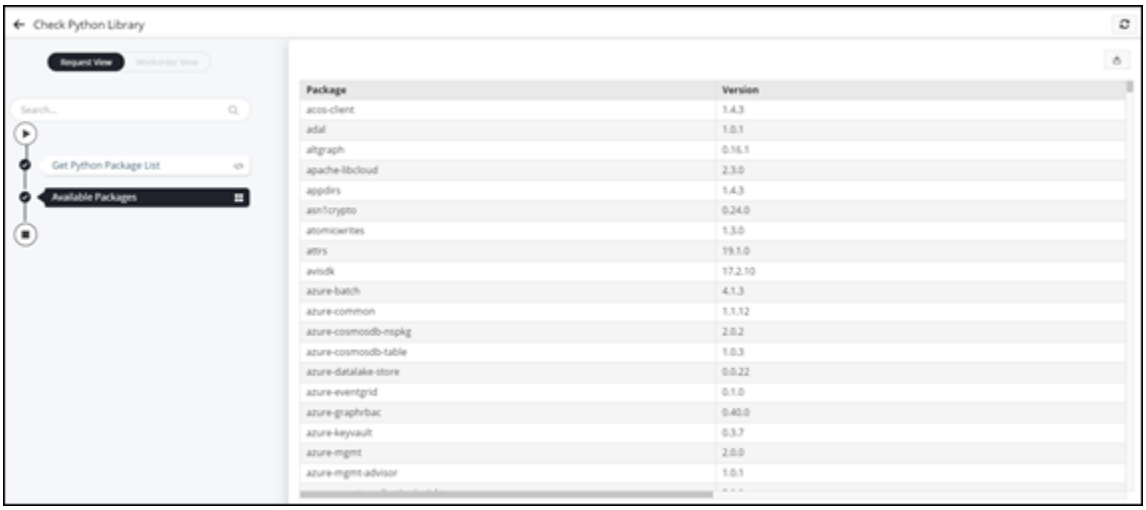
To trigger the workflow for getting the list of Python packages available within AppViewX:

On the Workflow Inventory page, from the navigation pane on the left, under **Python Library**, click **List Packages**.



The workflow is executed and a list of packages with their versions is displayed on the screen.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Chapter 9: Workflow Tasks

- Overview
- Folders
- Task Category - General
- Task Category - User Interface
- Task Category - ChatOps and Notifications
- Task Category - Change Management
- Integrations
- Automation Collision
- Workflow Task Actions
- Variable Mapping
- Rollback Workflow
- Task Scheduler
- Workflow Cart
- Workflow Options
- Connecting tasks in Workflow Studio
- Connecting Workflow Tasks with Failover - RGF Flow

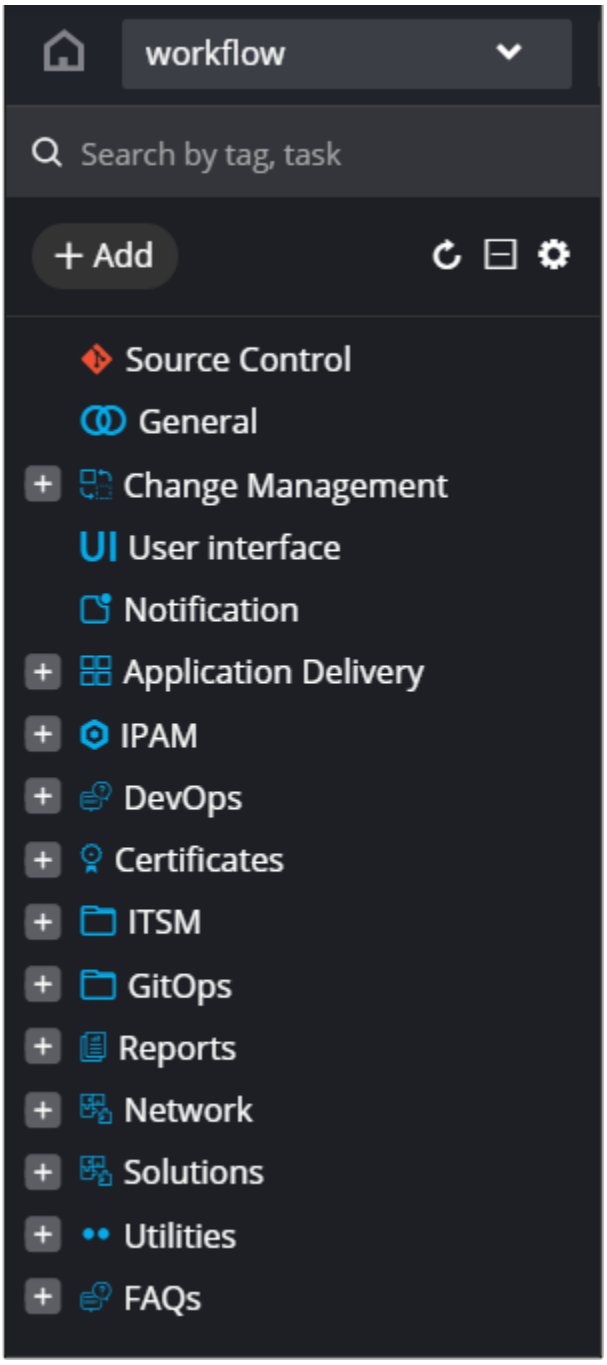
Overview

A workflow palette or task is an independent business logic block required to automate and stitch together a workflow.

- Tasks are objective workflow elements required to automate and design a workflow
- Tasks can have custom and/or pre-defined business logic to aid in workflow automation
- Tasks can be defined as either user tasks or automated tasks
- Tasks can be cloned and reused with the workflow
- Tasks will have an Input/output (I/O operation) to pass data (in the form of variables) from one stage of a workflow to another

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Folders



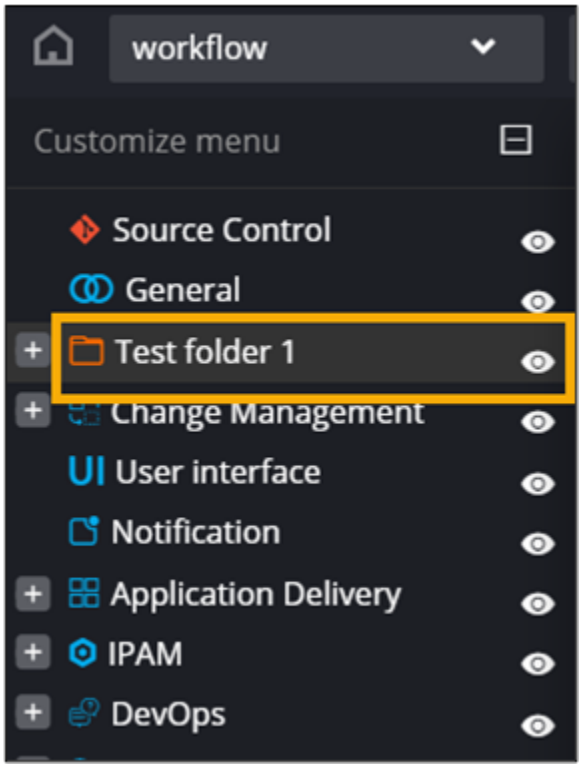
The following options are available in the Workflow Tasks folder menu:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Options	Description
Search	Allows you to search for tasks by typing keyword(s)
Add	Allows you to add a new folder
	Refreshes the tasks folders menu
	Allows you to expand and collapse the task folders
	Allows you to hide folders from the workflow tasks menu and customize task sequence within a folder

To hide a folder from the workflow tasks menu:

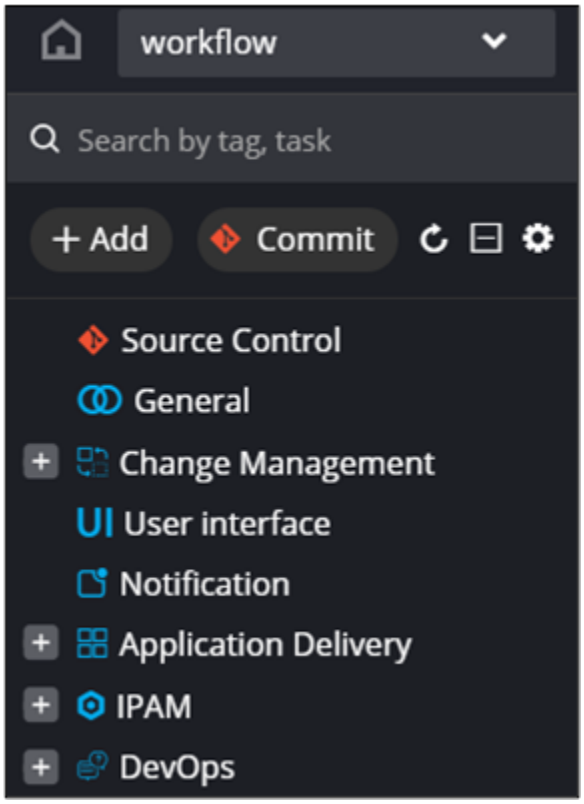
1. From the top right corner of the menu, click .
2. Click next to the folder you want to hide from the menu.



3. Click **Save**.

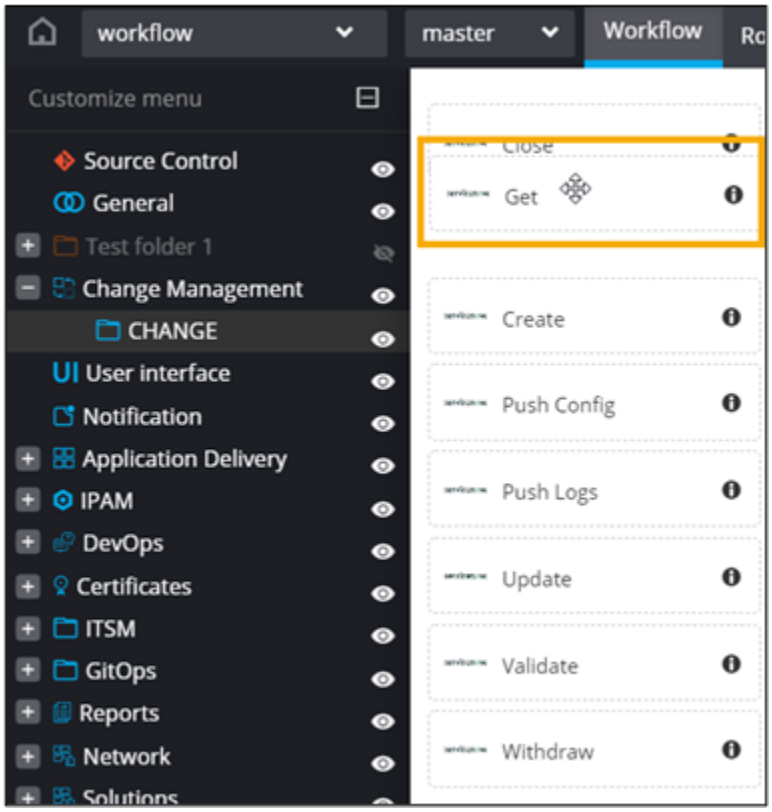
   The folder is no longer visible in the menu.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



4. To reorder the sequence of tasks within a folder, from the top right corner of the menu, click .
5. Select a folder from the menu.
6. Hold and drag tasks to change their sequence in the folder.

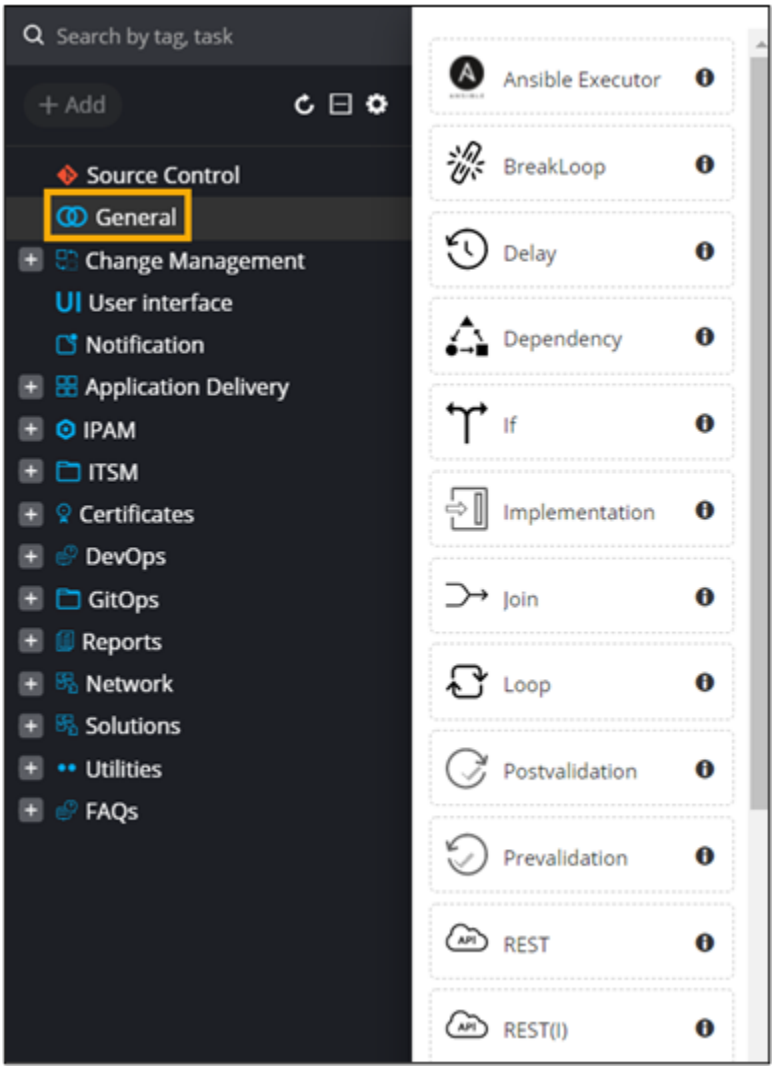
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Task Category - General

This folder allows you to incorporate all flow control tasks within a workflow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



- Ansible Executor
- Break Loop
- Delay
- Dependency
- If
- Implementation
- Join
- Loop
- Postvaildation
- Prevalidation
- REST

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

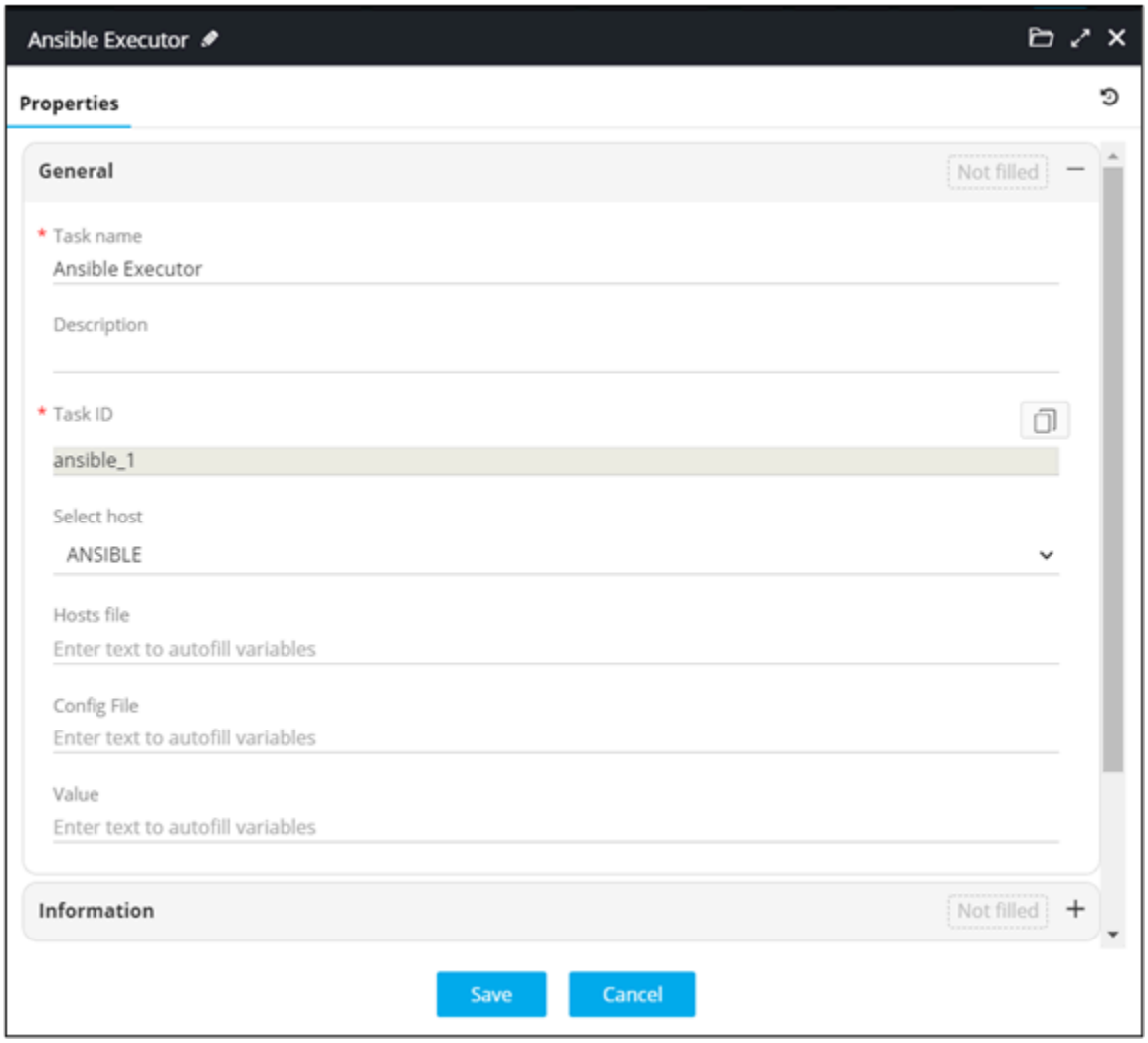
- REST(I)
- Retry
- Rollback
- Schedule
- Script
- Split
- Switch
- WorkOrder

Ansible Executor

This task allows you to integrate with Ansible from an AppViewX workflow. The Ansible executor task is used when there is a need for automating network configurations through Ansible as the southbound. This task must be used when passing input data from AppViewX to Ansible and while executing the Ansible modules on a specific host.

- Provision to input data in either YAML or JSON format
- Provision to refer YAML or JSON samples
- Provision to use the YAML task as either a user interface task or as a service task
- Provision to select the ‘host’ on which to execute

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



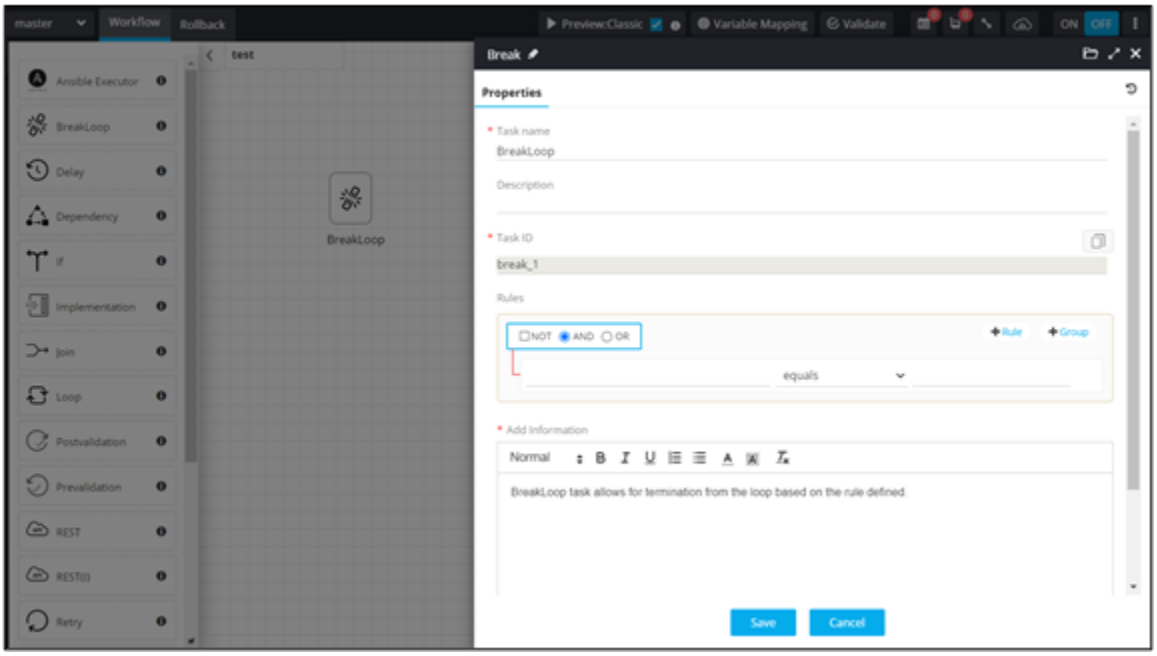
Note: In the event of integrating with an existing Ansible setup, the host instance can be defined under **AppViewX Menu** > **Settings** > **Integration** > **Ansible configuration**. For more information, refer to the section on Ansible Southbound Integration.

Break Loop

BreakLoop task allows for termination from the loop based on the rule defined.

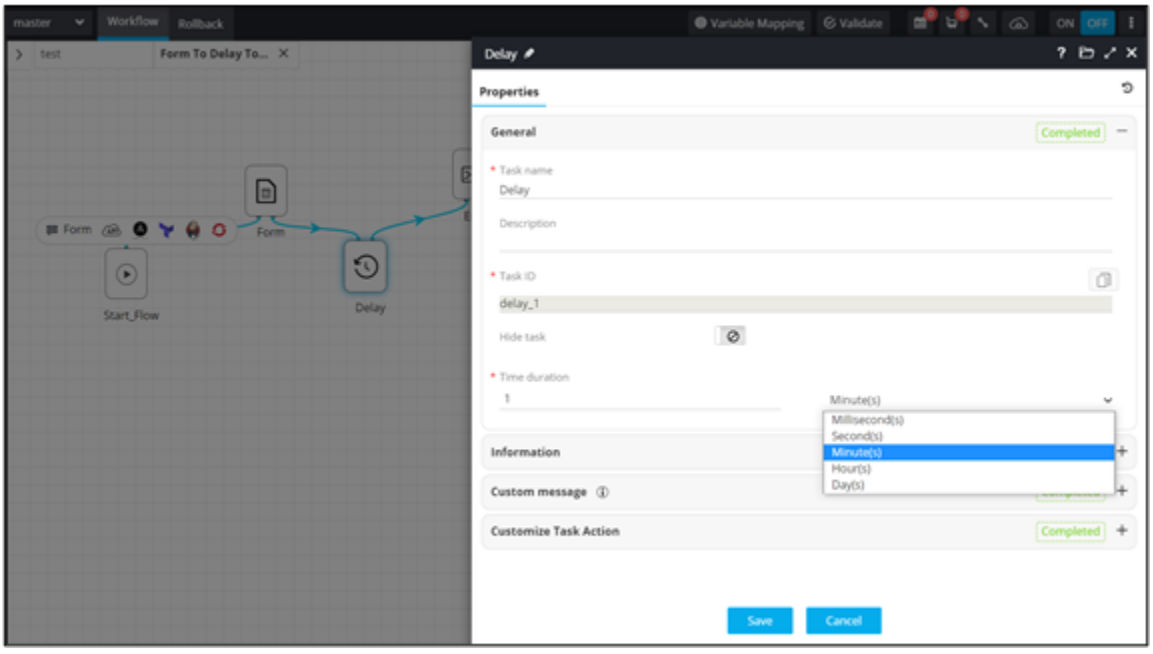
- Provision to define a break condition to terminate the loop.
- Provision to define rule or Boolean condition (AND, OR, NOT) to test the loop variable.
  - If the condition holds true, the loop terminates.
  - If the condition holds false, control is passed to the loop.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Delay

Delay task allows you to configure delays between workflow tasks. Delay can be configured for Milliseconds, Seconds, Minutes, Hours and Days.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

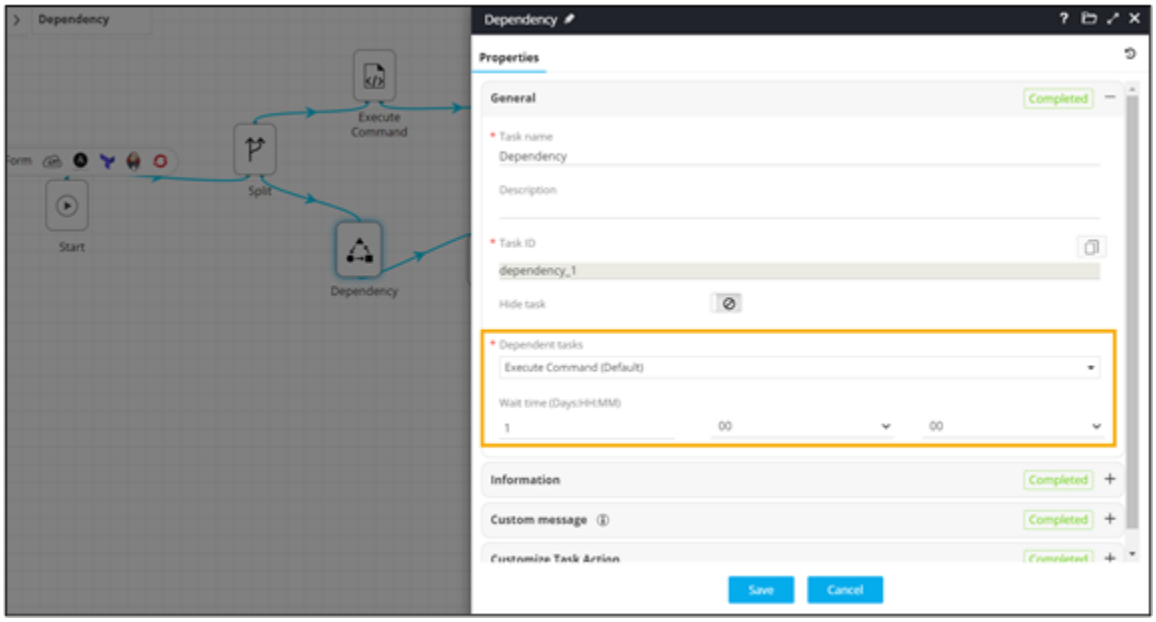
Dependency

This task allows you to explicitly define dependencies on previous tasks prior to executing the next task in the workflow sequence. This is helpful when you want to check whether a specific task has completed or not before going on to the next task.
You can assign a wait time or duration for which the next task in the flow can wait for the dependent task to be completed. If the dependent task is completed within the wait time, the next task is executed. However, if the dependent task does not get completed with the assigned wait time, the workflow stops.

- Provision to select the dependent tasks which are defined within the workflow.
- Provision to assign a wait time or the duration for which the next task in the flow can wait for the dependent task to be completed.
- Task dependency will check for the successful completion of the dependent tasks.

To define dependency on a task prior to executing the next task in the flow:

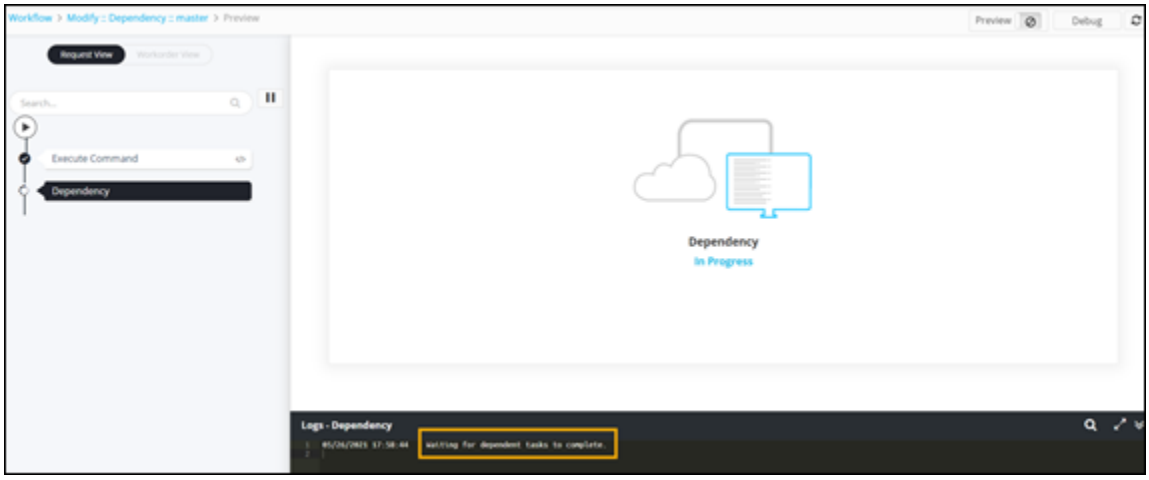
1. Design a workflow.
2. Drag and drop necessary tasks within the workflow.
3. Add the dependent tasks and assign wait time.



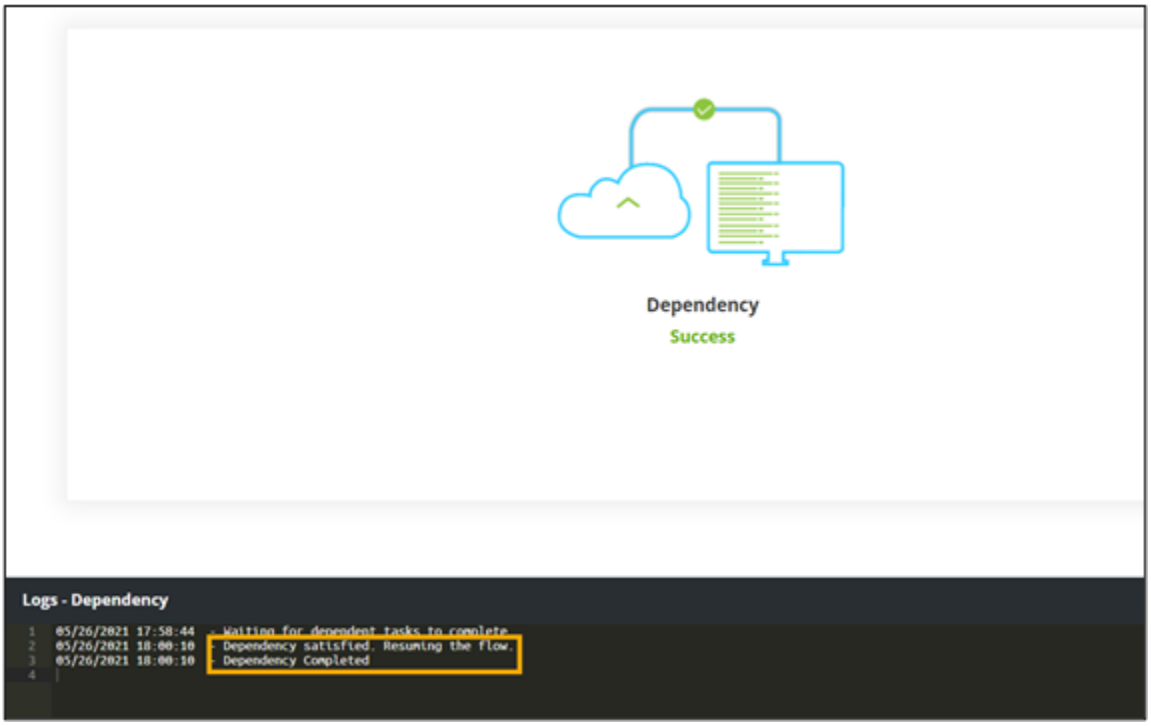
4. Click **Preview**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Dependency in progress.

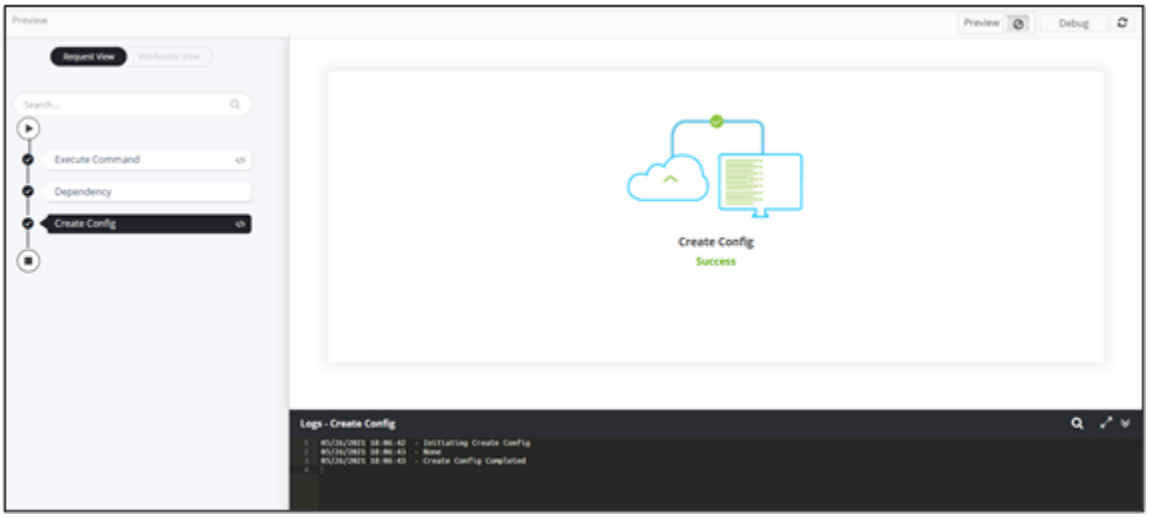


- Dependency completed.



- Next task initiated and completed.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

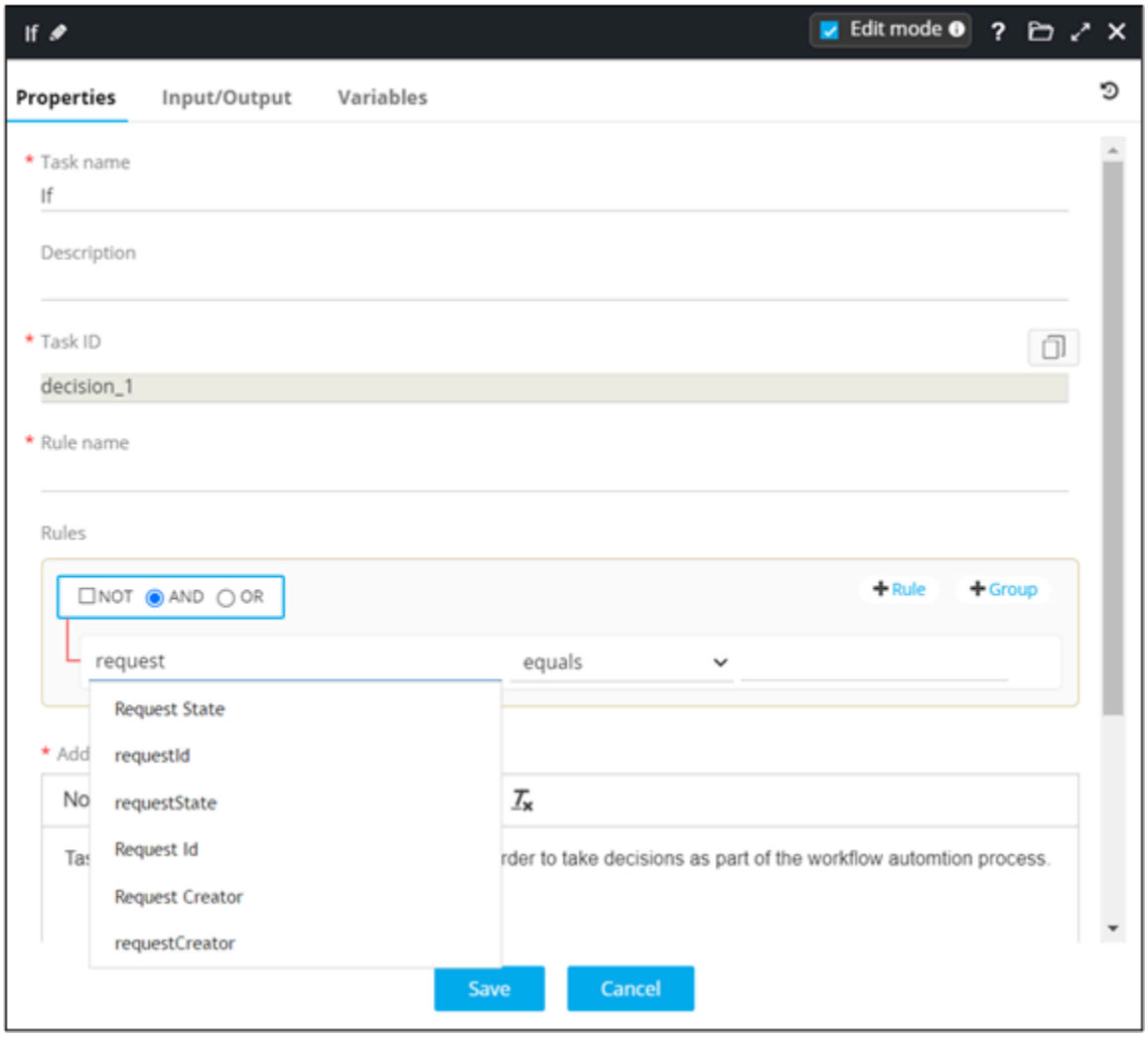


If

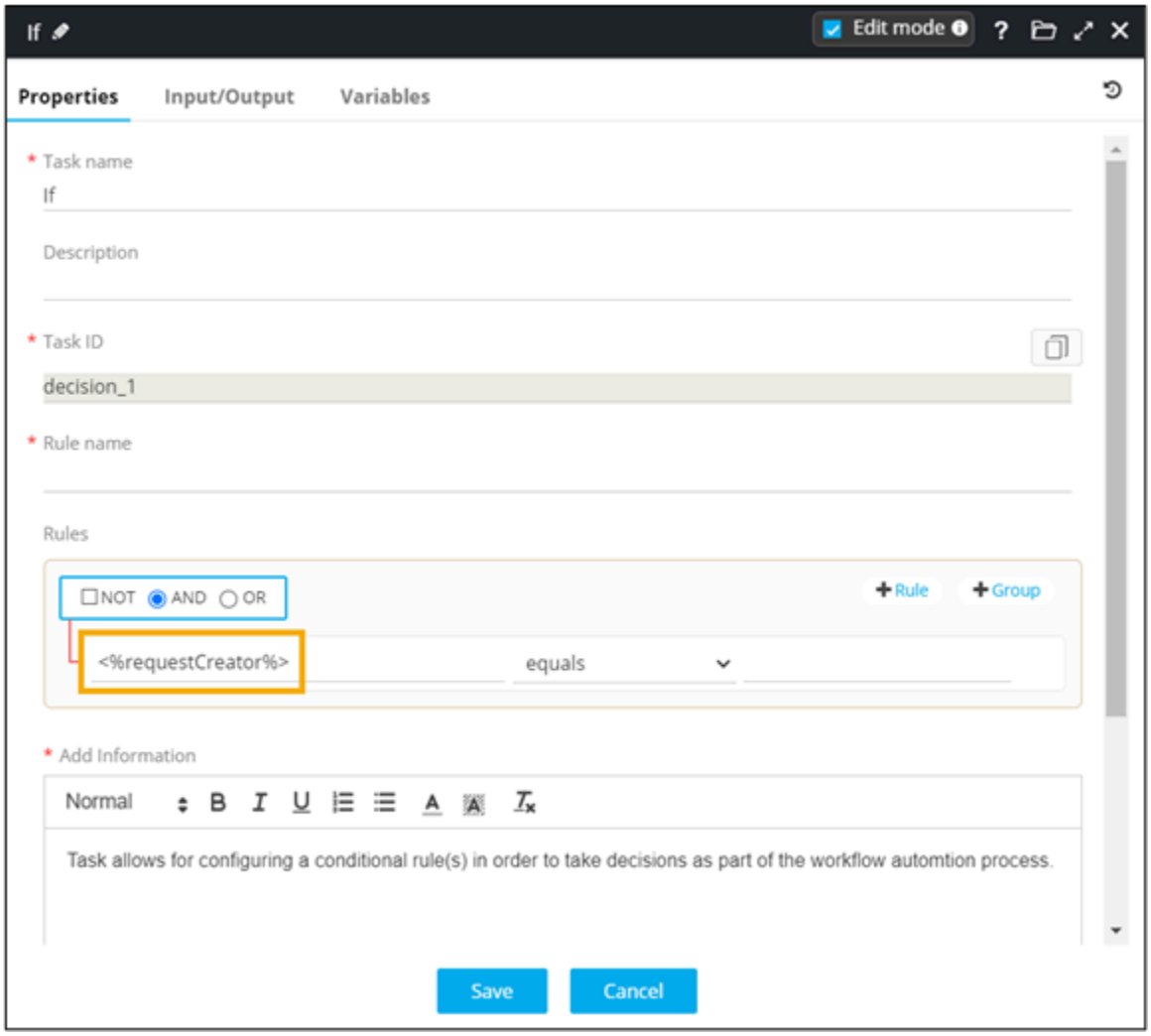
The If task allows you to configure a conditional rule(s) in order to take decisions as part of the workflow automation process.

- Provision to define multiple rules based on which a workflow can be routed (maximum of two decisions can be taken).
- Provision to reference variables from a previous task as input.
- Provision to define conditional rule logic as a truth table.
- Provision to copy the task id and use it to reference across workflows.
- Provision to auto-populate the global variables.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



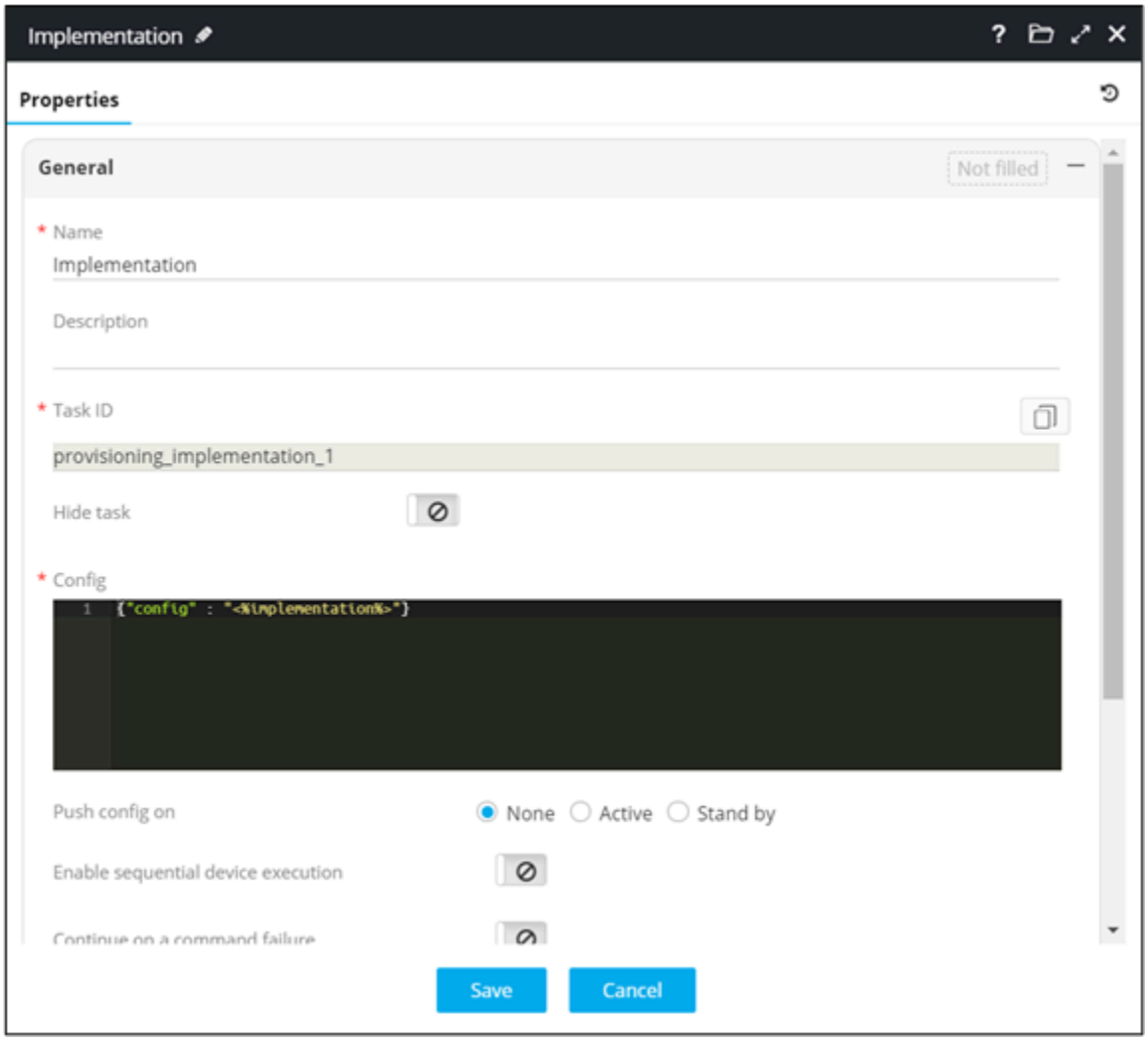
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Implementation

This task allows you to perform Implementation changes on the end point.

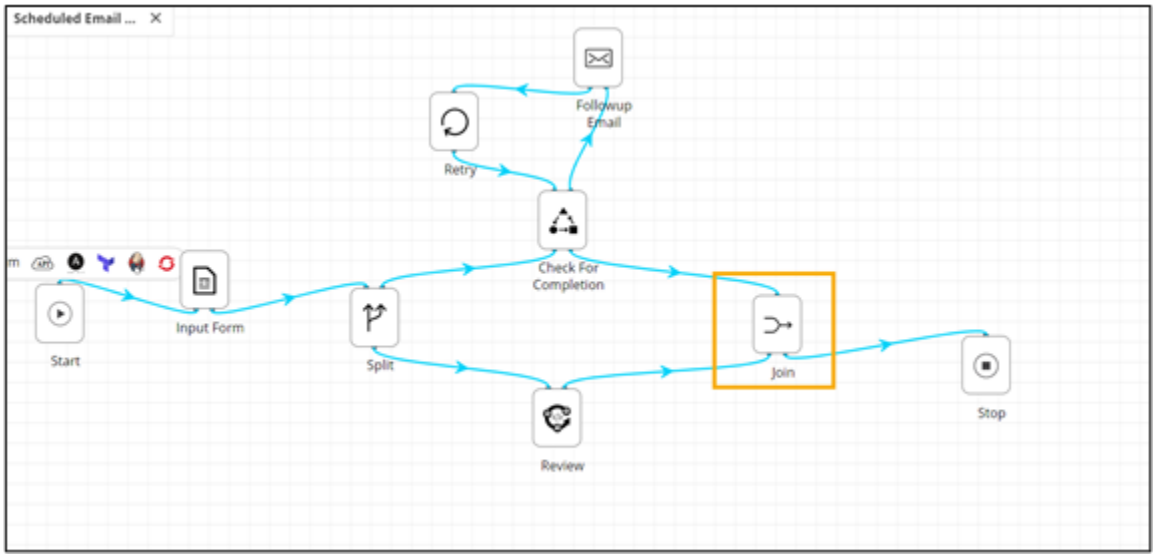
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Join

The Join task is used when multiple flows are branched out and executed parallely. This task allows you to merge one or more branches of a workflow into a logical end.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

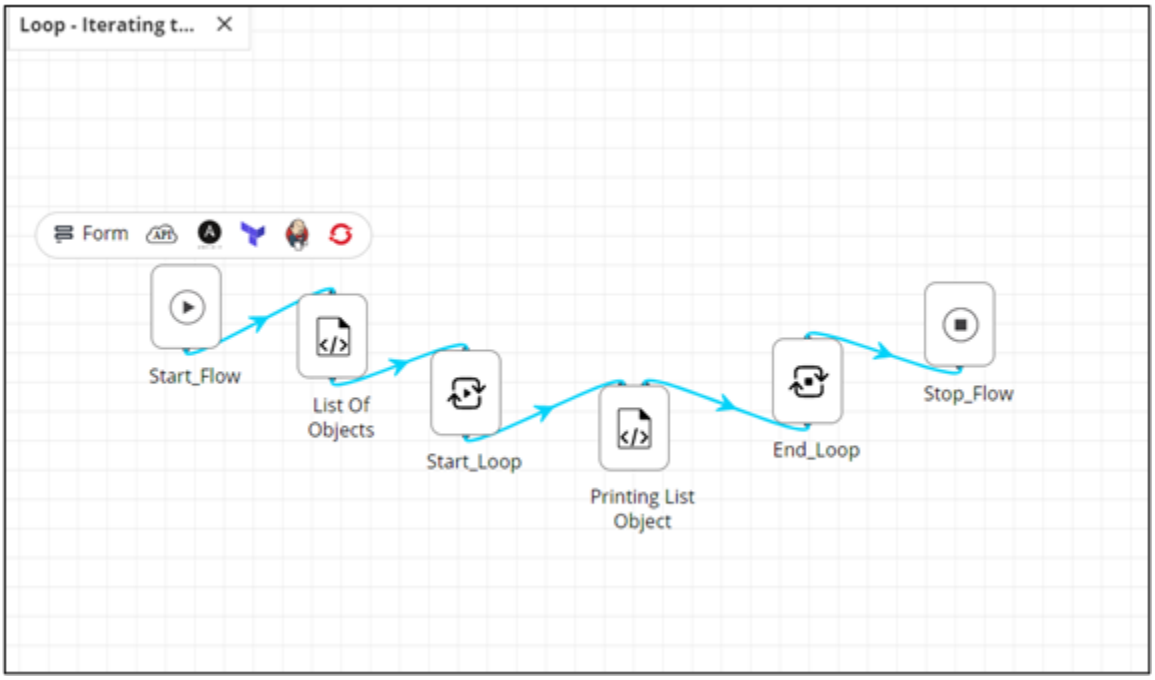


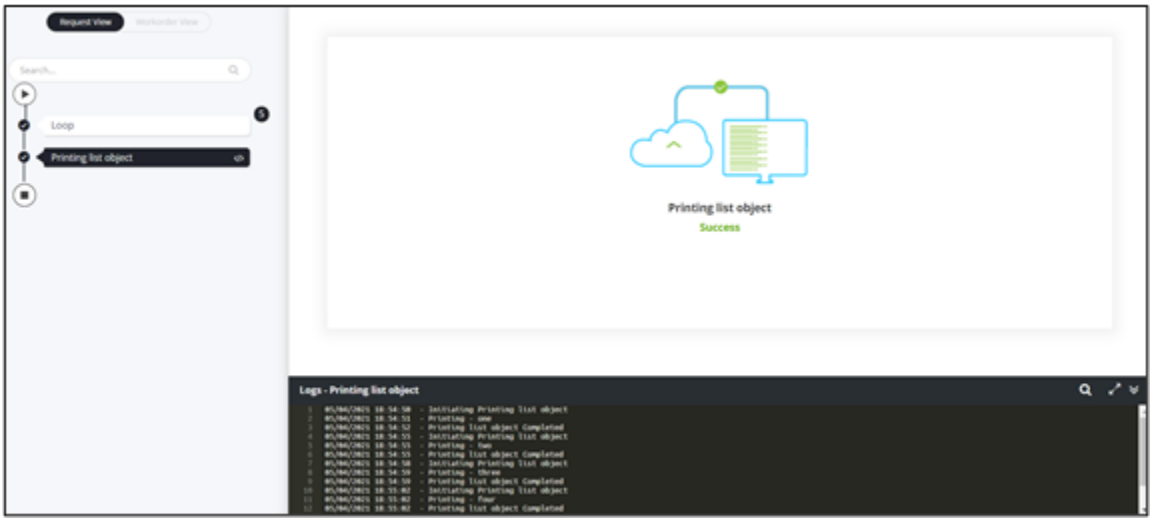
Loop

Loop task allows for repeated execution of a task or sub-process based on a specified condition. The Loop task consists of two parts – Start loop and End loop.

- Provision to define a loop for iterative execution of a task or sub-process.
- Provision to resume control to the next iteration of the loop in the event of failure of any task within the loop.
- Provision to define rule or Boolean condition (AND, OR, NOT) to execute the loop variable. If the condition holds true, the loop continues to iterate. If the condition holds false, the loop skips or exists.
- Provision to display the count of iteration in the stage view

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

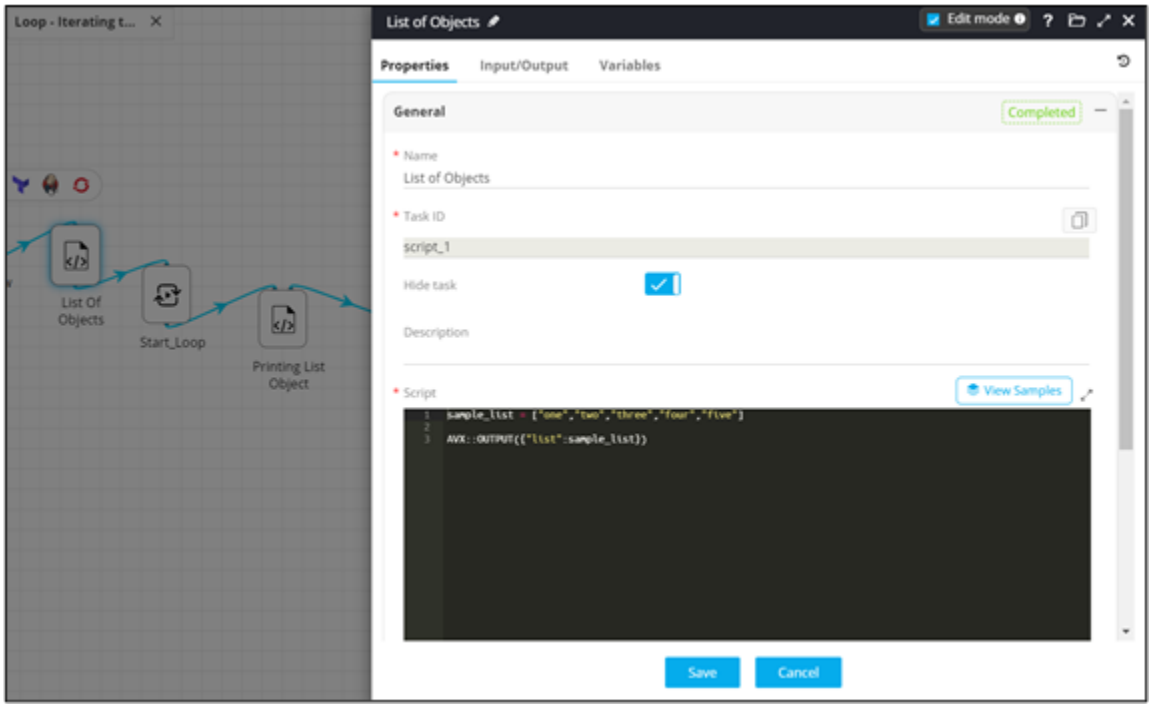




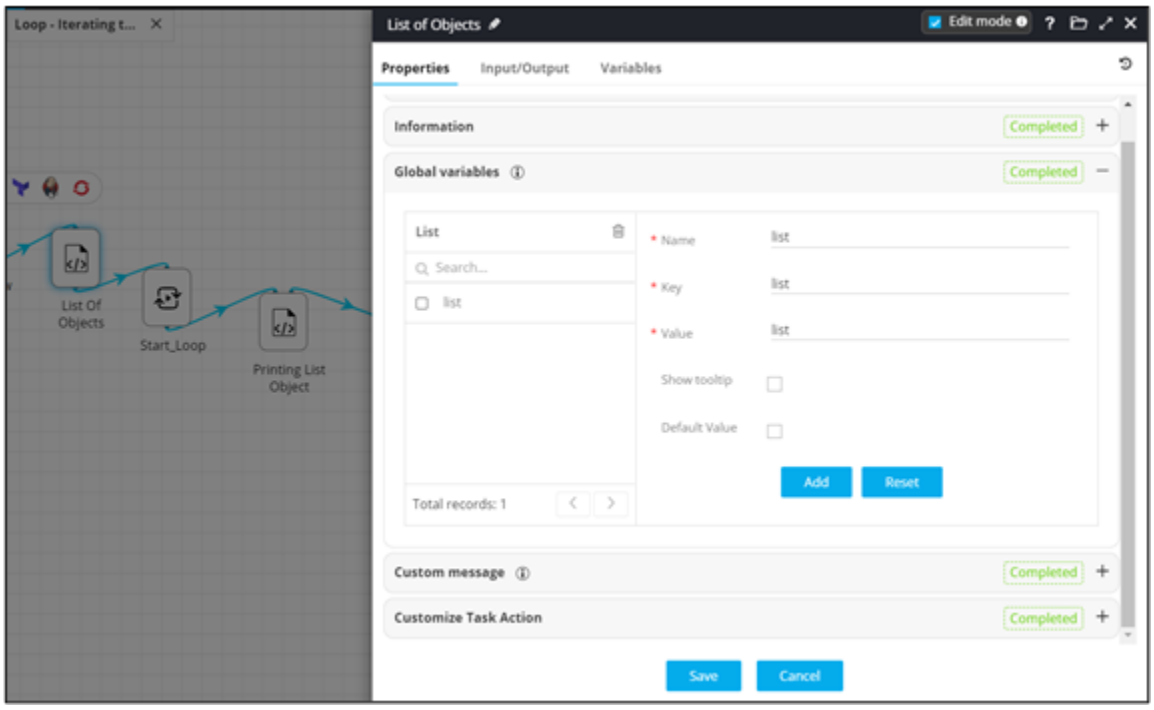
To add a Loop task to your workflow:

1. Design a workflow.
2. From the **General** section, drag and drop the **Loop** task.
3. Define a script to get a list of values.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

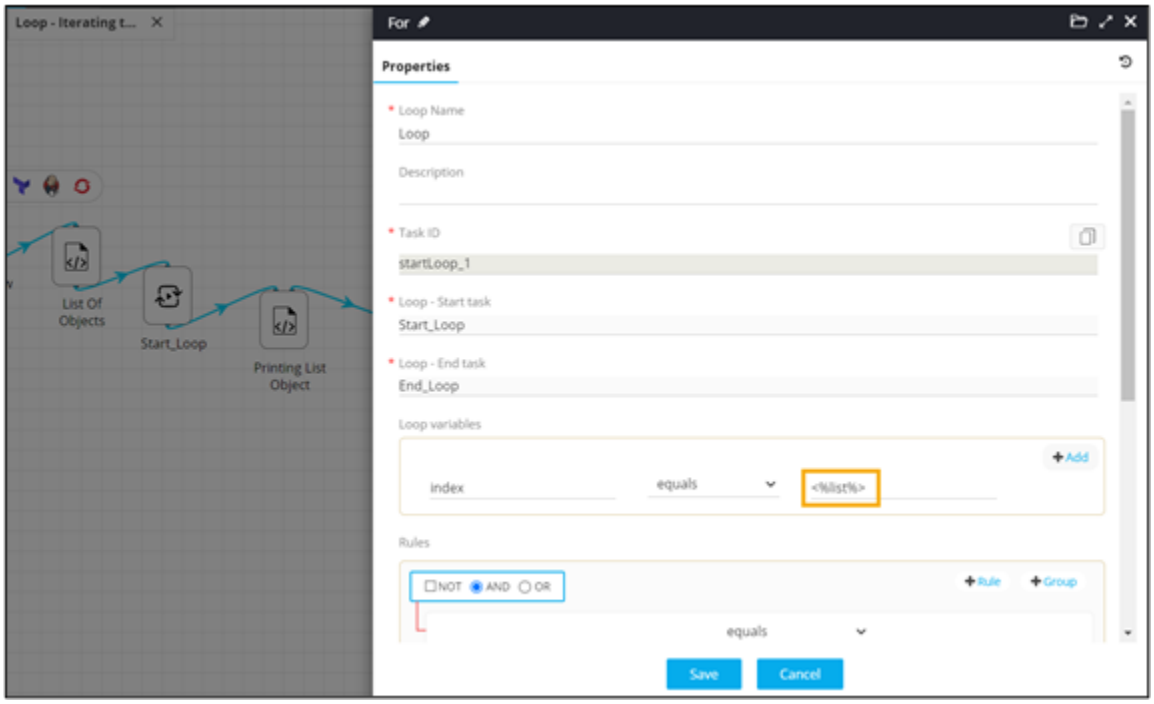


4. Define a sample list of values and declare it as a global variable.

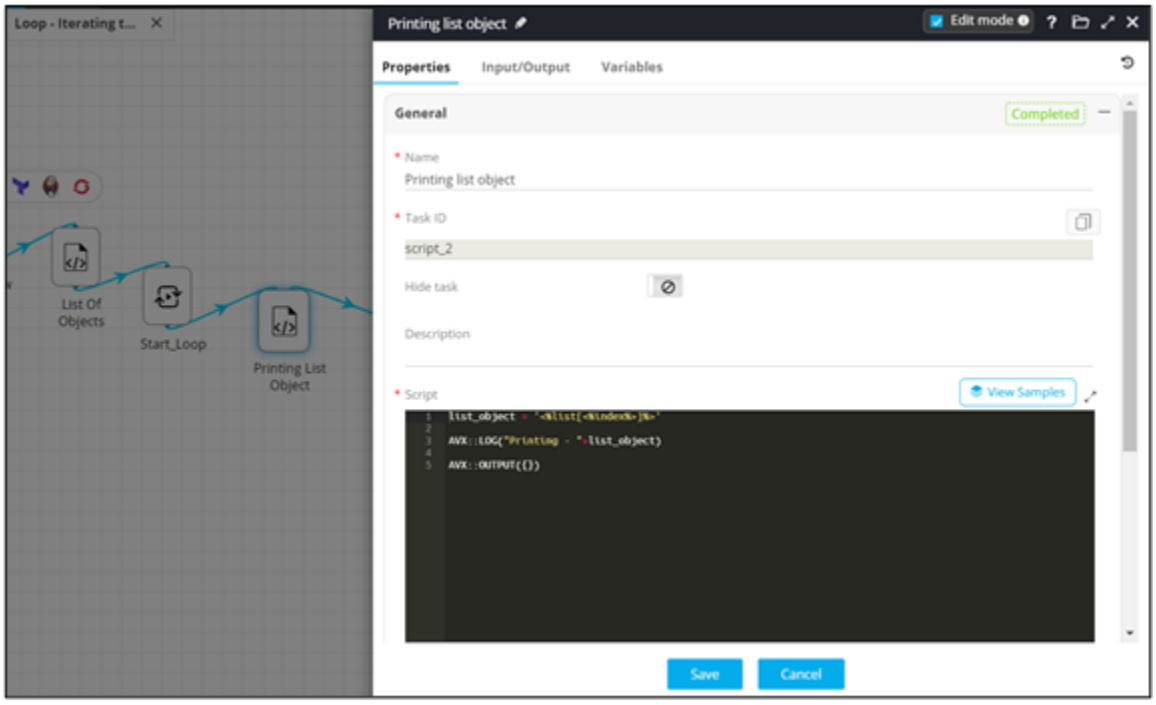


5. Under **Loop variables**, reference the variable from the previous task to be used for iteration.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



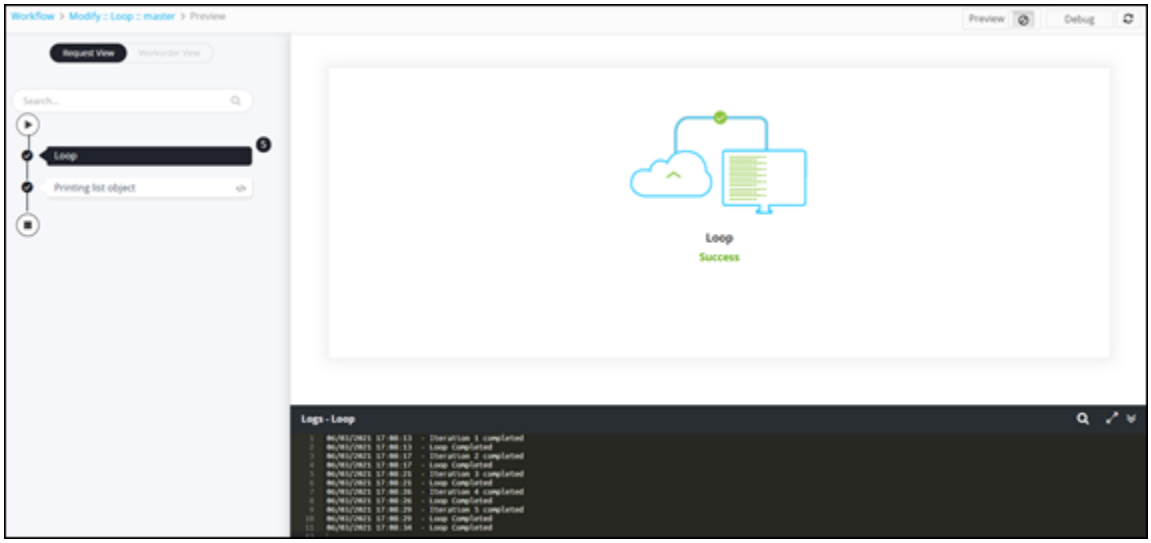
6. Define a **Script** task for printing list objects.



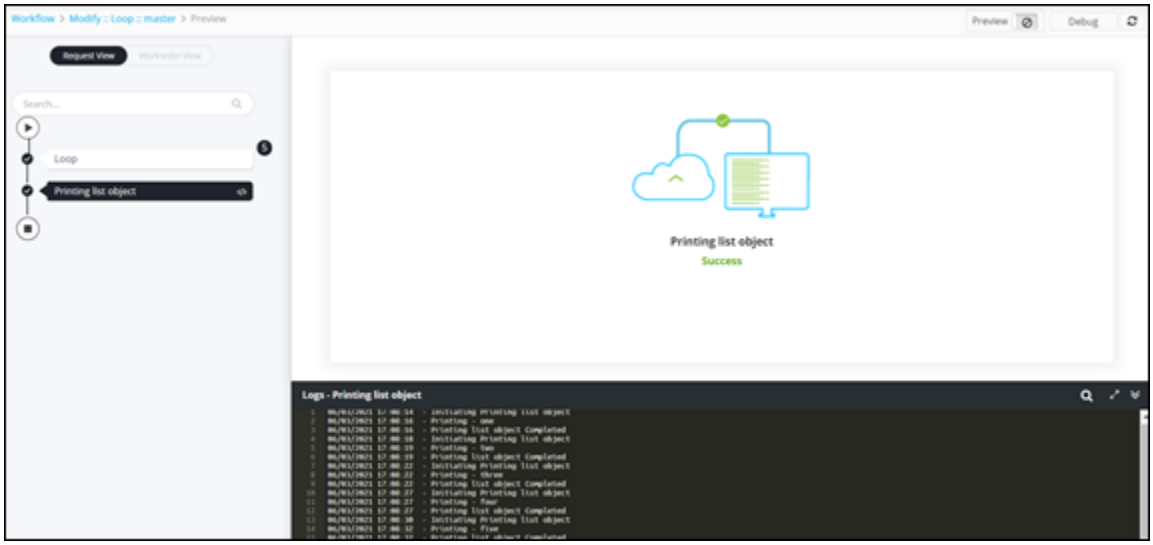
7. Connect the workflow tasks.
8. Click **Preview**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Loop execution with iteration count displayed.



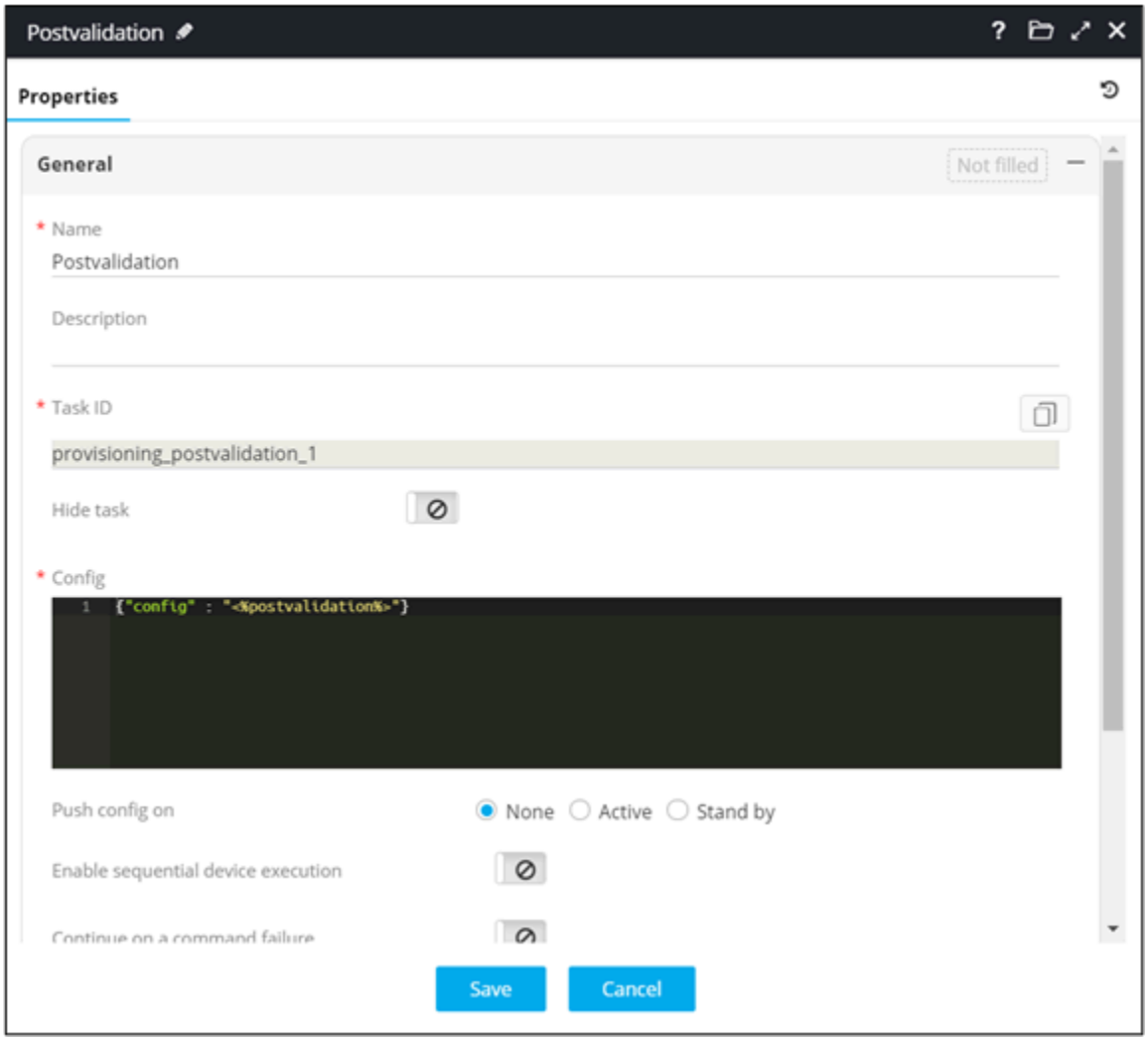
- Printing list objects completed.



Postvaildation

This task allows you to perform post-validation changes on the end point.

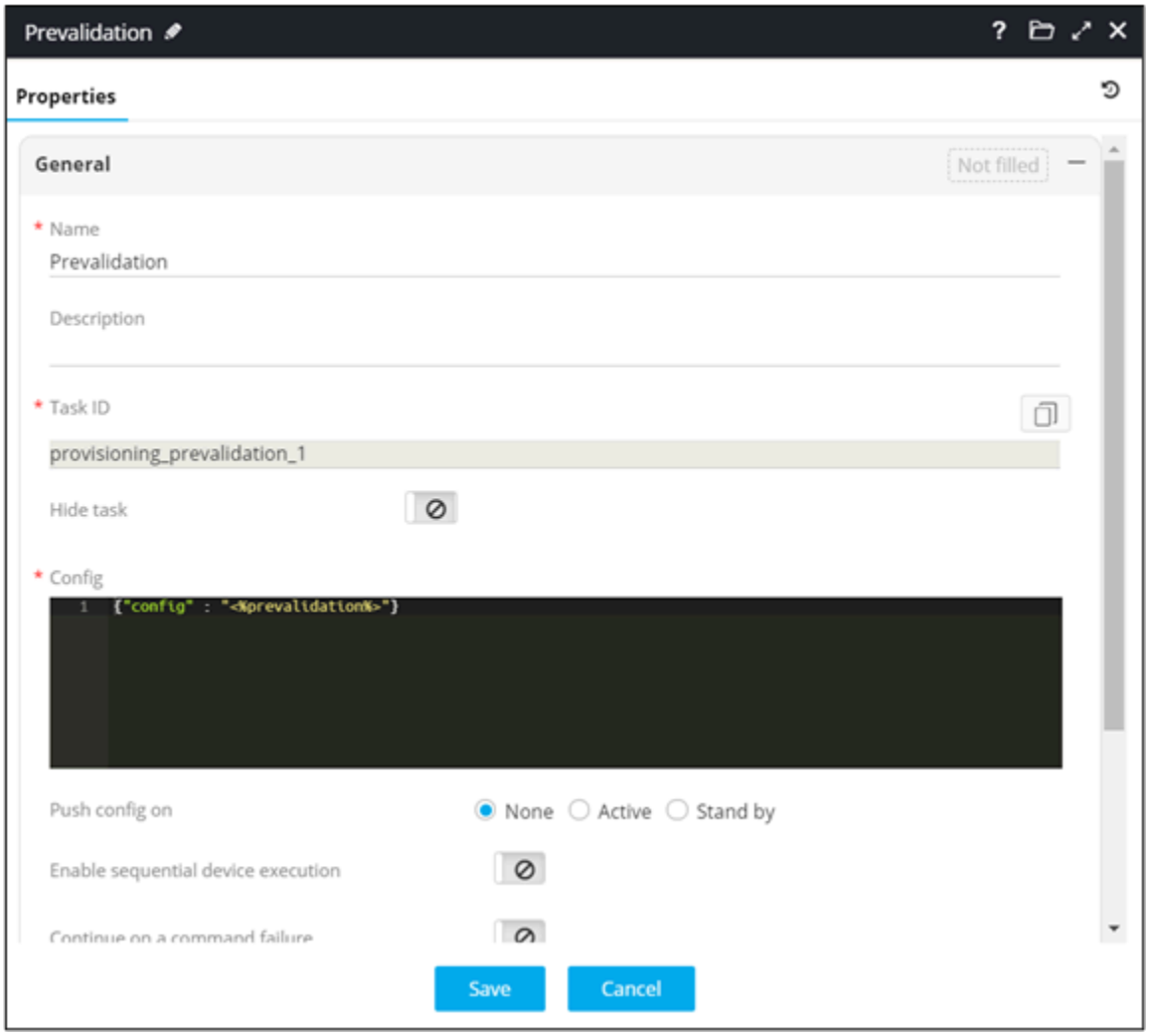
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Prevalidation

This task allows you to perform pre-validation changes on the end point.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



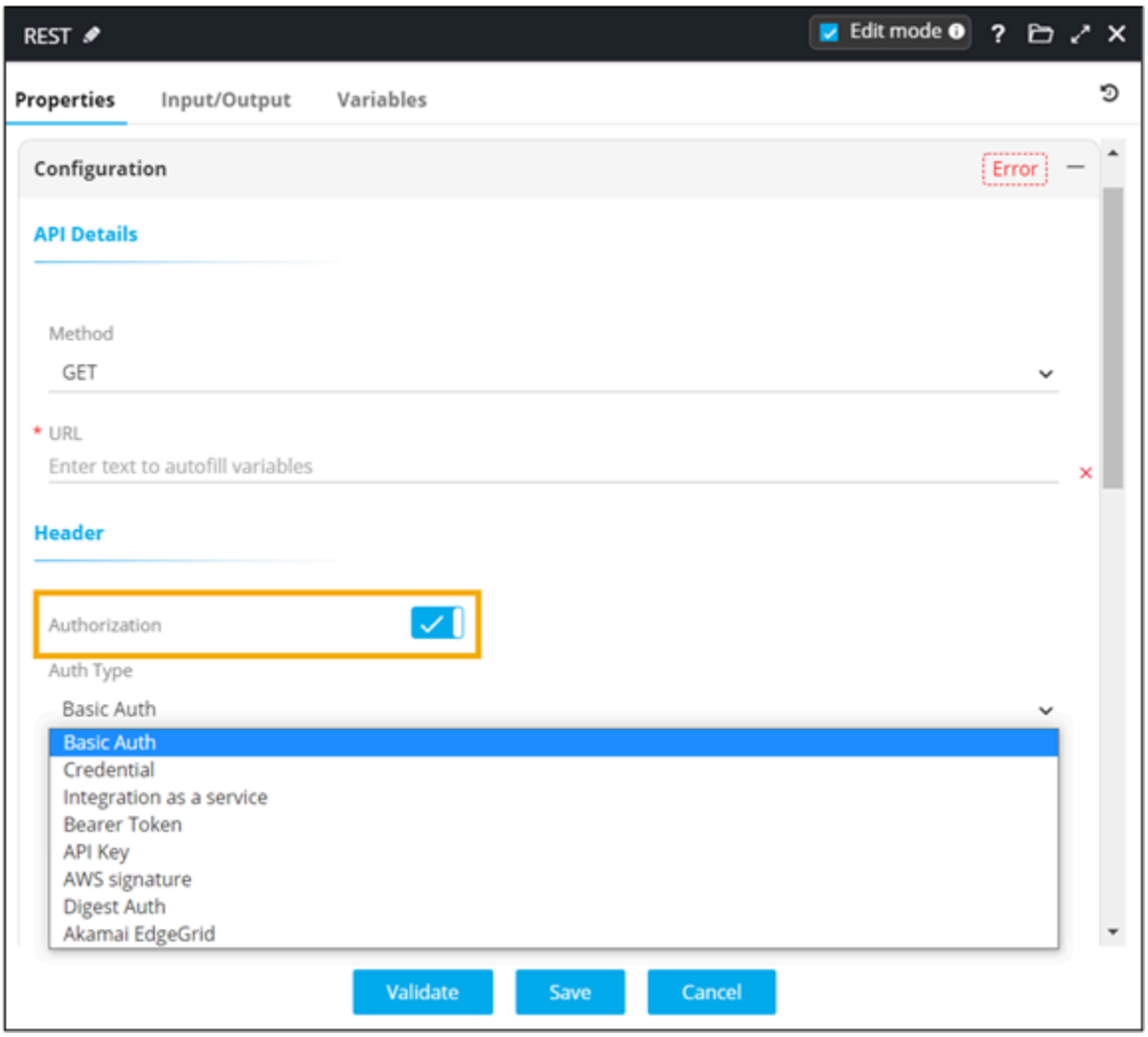
REST

REST API task is used for external web service communications as part of the workflow automation process.

- Enhanced API auth support:
  - Basic Auth
  - Bearer Token
  - API Key
  - Digest Auth
  - AWS Signature
  - Akamai EdgeGrid

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Support for ‘GET’, POST’, ‘PUT’, ‘DELETE’ and ‘PATCH’ methods
- Support for multiple headers



- Authorization Modes

Authorization Modes

Provision to support different types of authentication in the visual workflow Rest API task.

- Basic Auth
- Bearer Token
- API Key

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

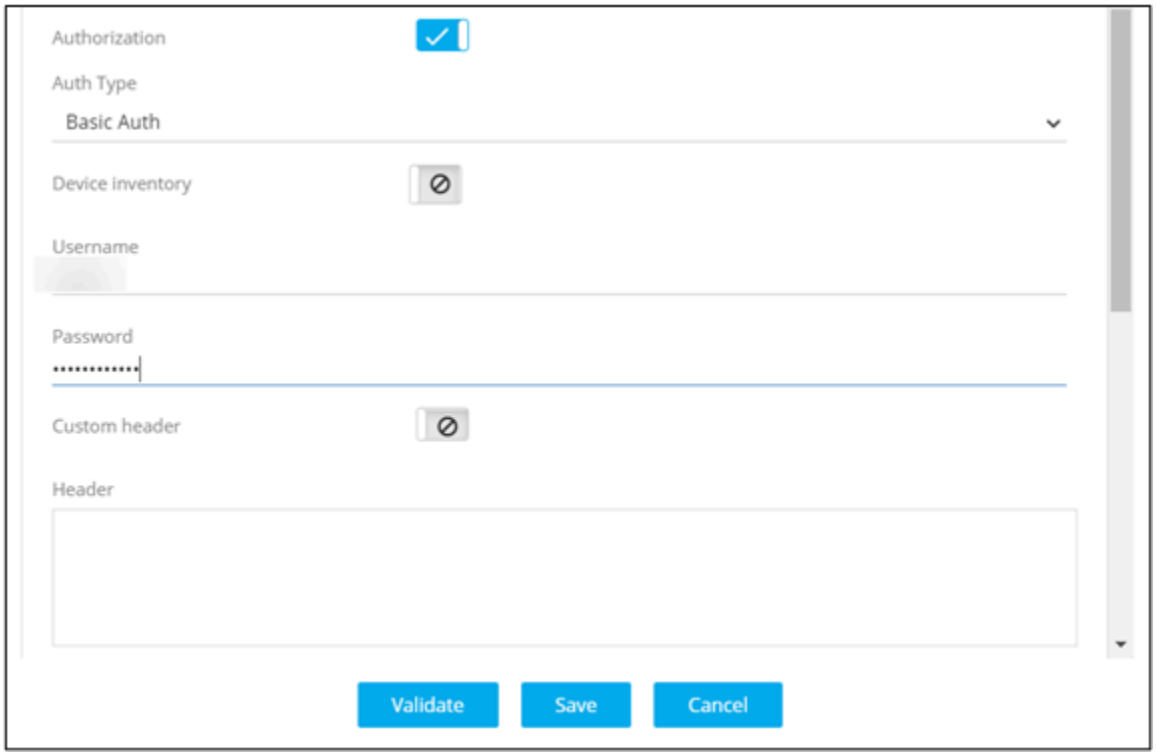
- Digest Auth
- AWS Signature
- Akamai EdgeGrid

Basic Auth

The client sends HTTP requests with the Authorization header that contains a base64-encoded value of Username and Password.

1. In the **REST** task, under **Authorization**, select **Auth Type** as **Basic Auth**.

2. Provide the **Username** and **Password**.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

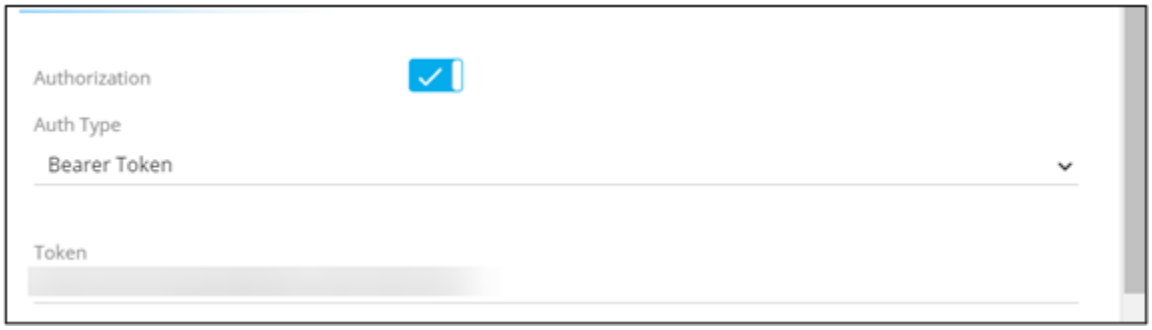
3. Click **Save**.

Bearer Token

A bearer token is an HTTP authentication scheme that involves authentication with security tokens. The client can define this token in the Authorization header while making requests to protected resources.

1. In the **REST** task, select the **Authorization type** as **Bearer Token**.

2. Provide the **Token** value.



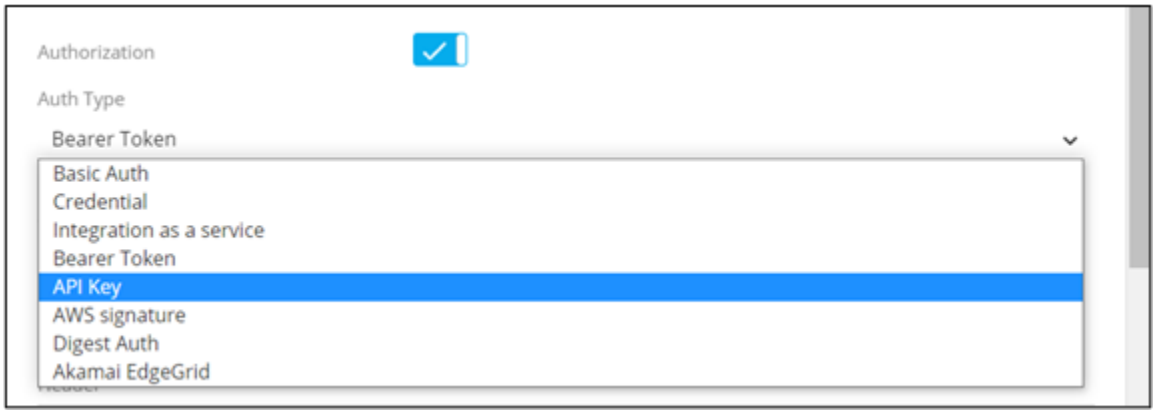
3. Click **Save**.

API Key

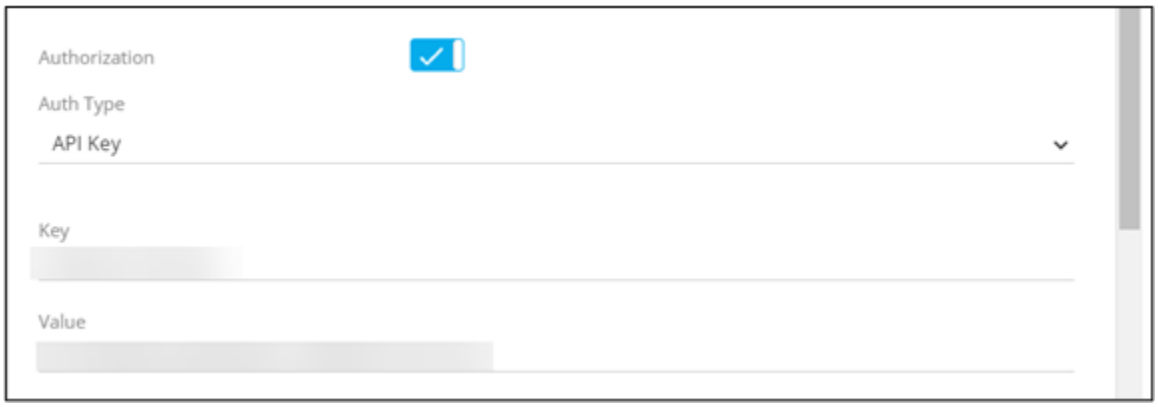
A unique generated value is assigned to all users to signify that the user is known. API keys can be sent in the query string as part of the URL or through headers.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. In the **REST** task, under **Authorization**, select **Auth Type** as **API Key**.



2. Provide the **Key** and **Value**.



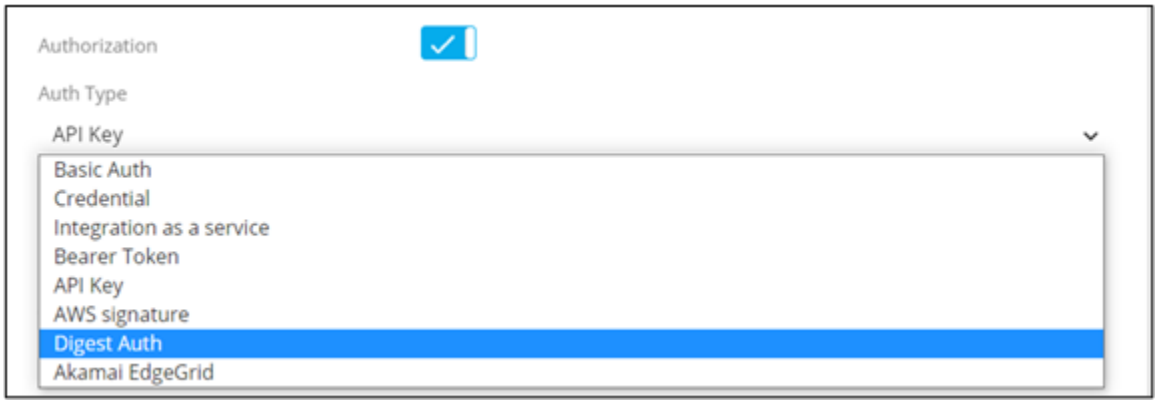
3. Click **Save**.

Digest Auth

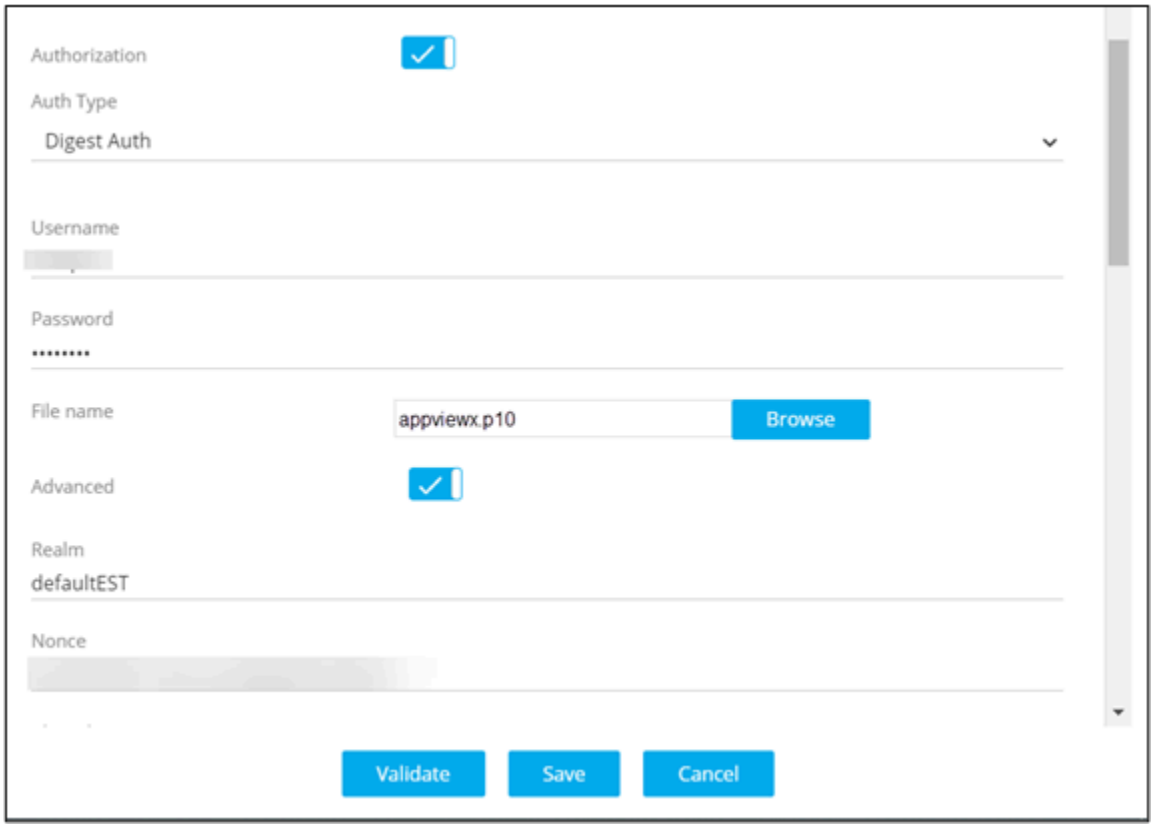
This authorization is immune to replay attacks as it uses a one time number (a nonce) from the server. The same method is used to generate a hash-key on the server side.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

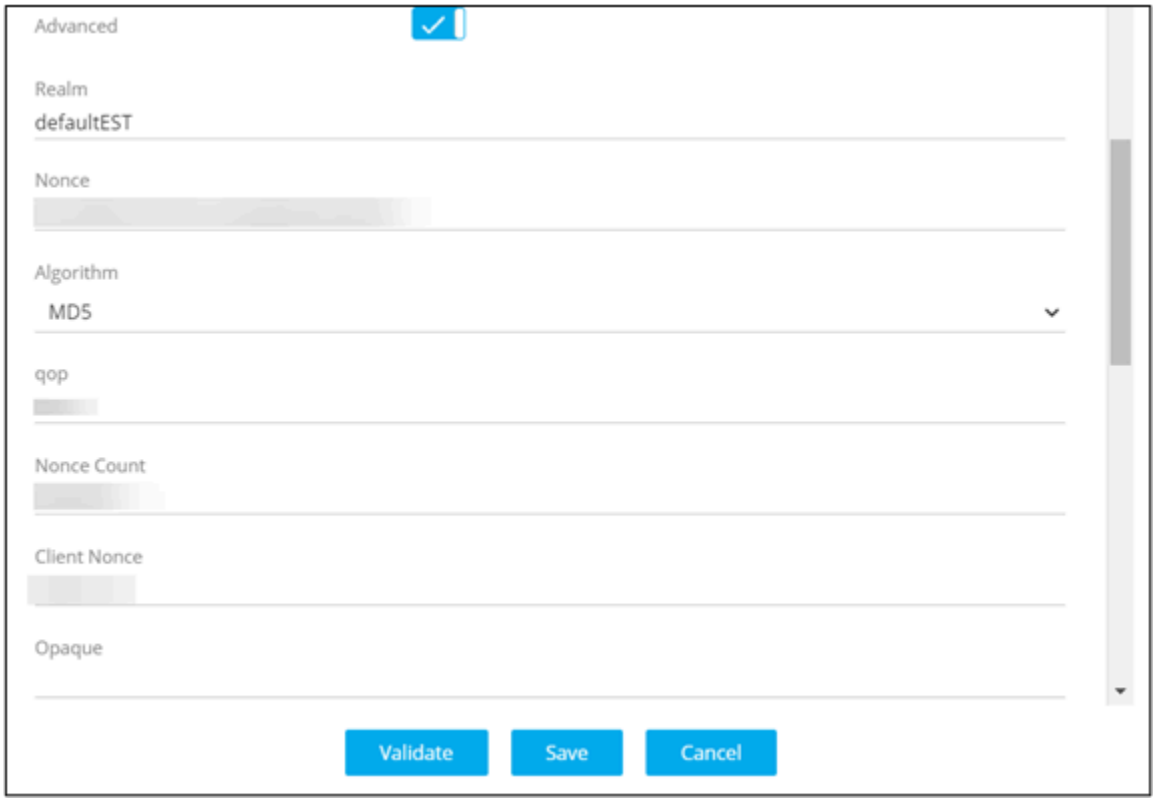
1. In the **REST** task, under **Authorization**, select **Auth Type** as **Digest Auth**.



2. Enter all the field information.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.



3. Click **Save**.

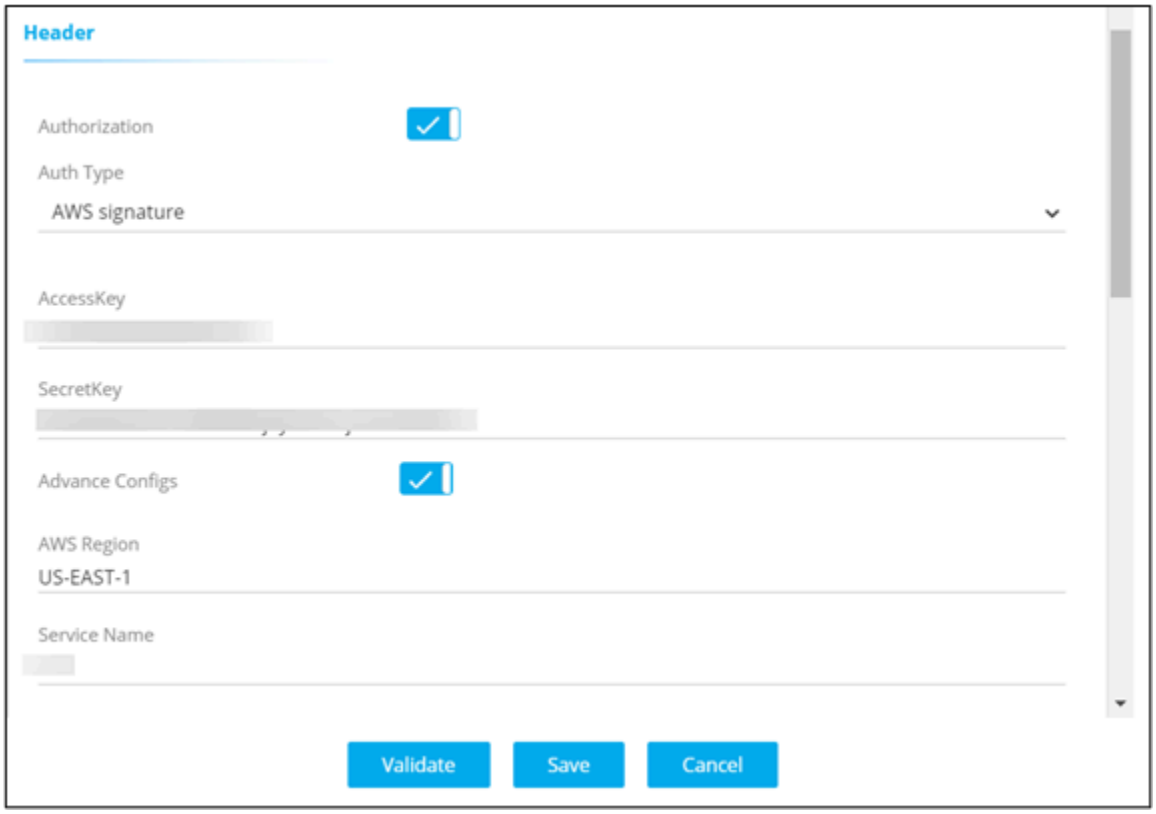
AWS Signature

Authentication information you send must include a signature.

1. In the **REST** task, under **Authorization**, select **Auth Type** as **AWS Signature**.

2. Under **Properties**, enter the field information.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

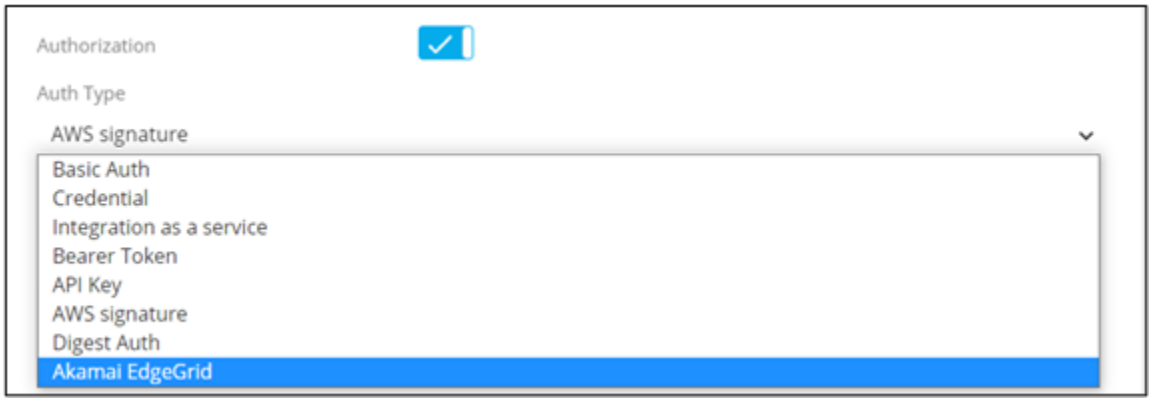


3. Click **Save**.

Akamai EdgeGrid

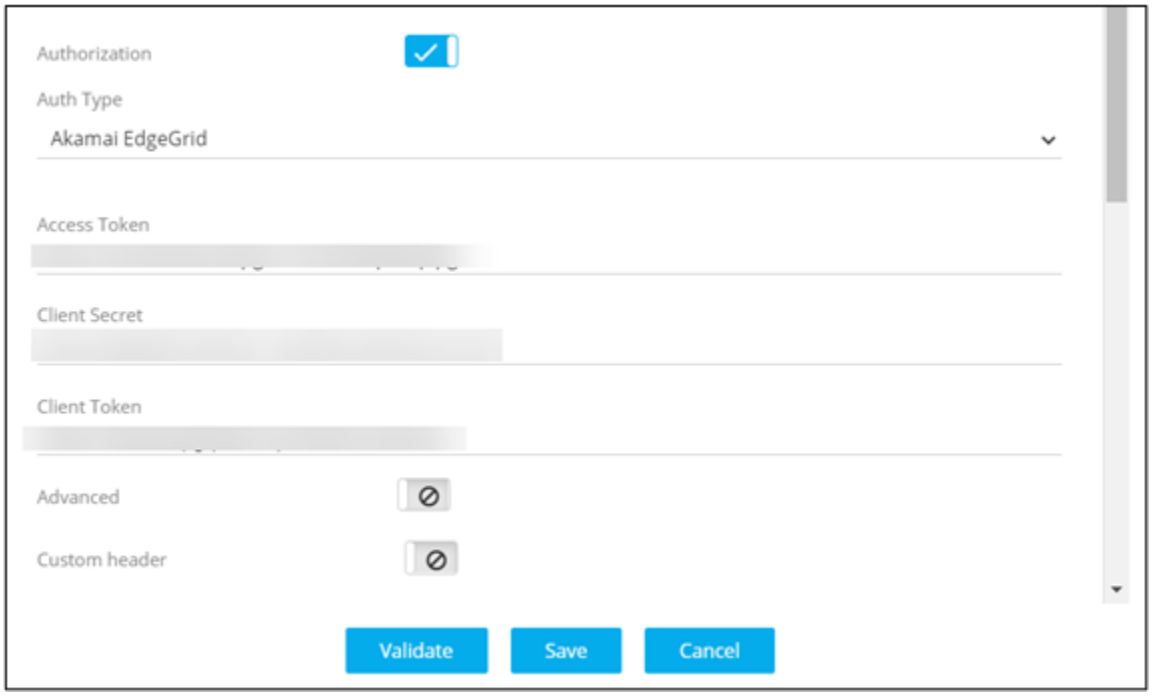
Akamai EdgeGrid uses a signature which is the base-64 encoding of the SHA-256 HMAC of the data to sign with the signing key.

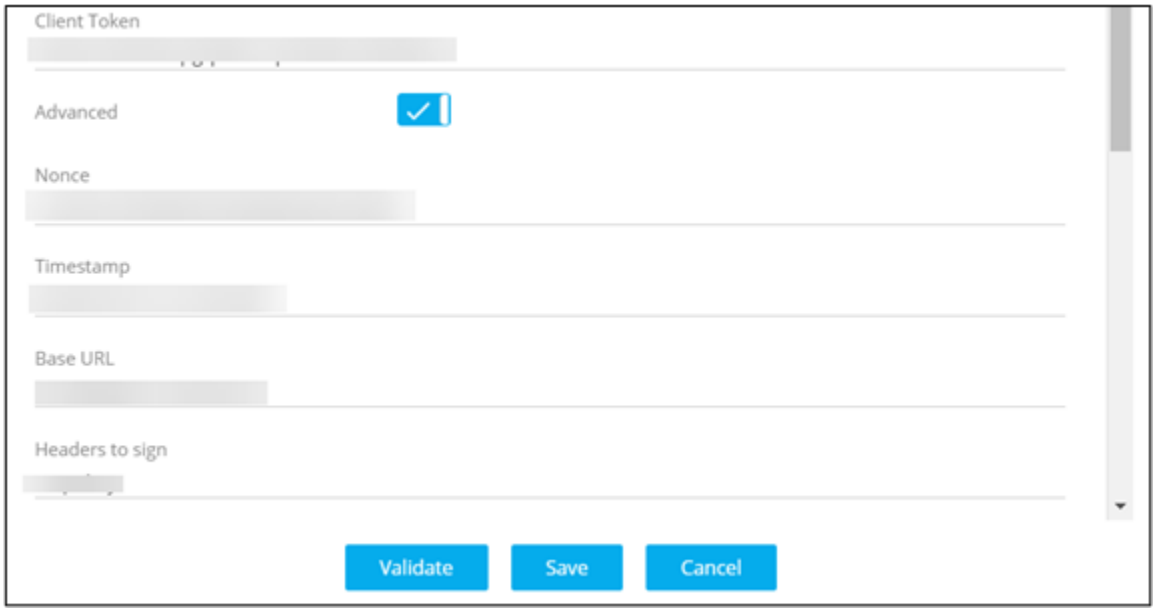
1. In the **REST** task, under **Authorization**, select **Auth Type** as **Akamai EdgeGrid**.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

2. Enter all the field information.





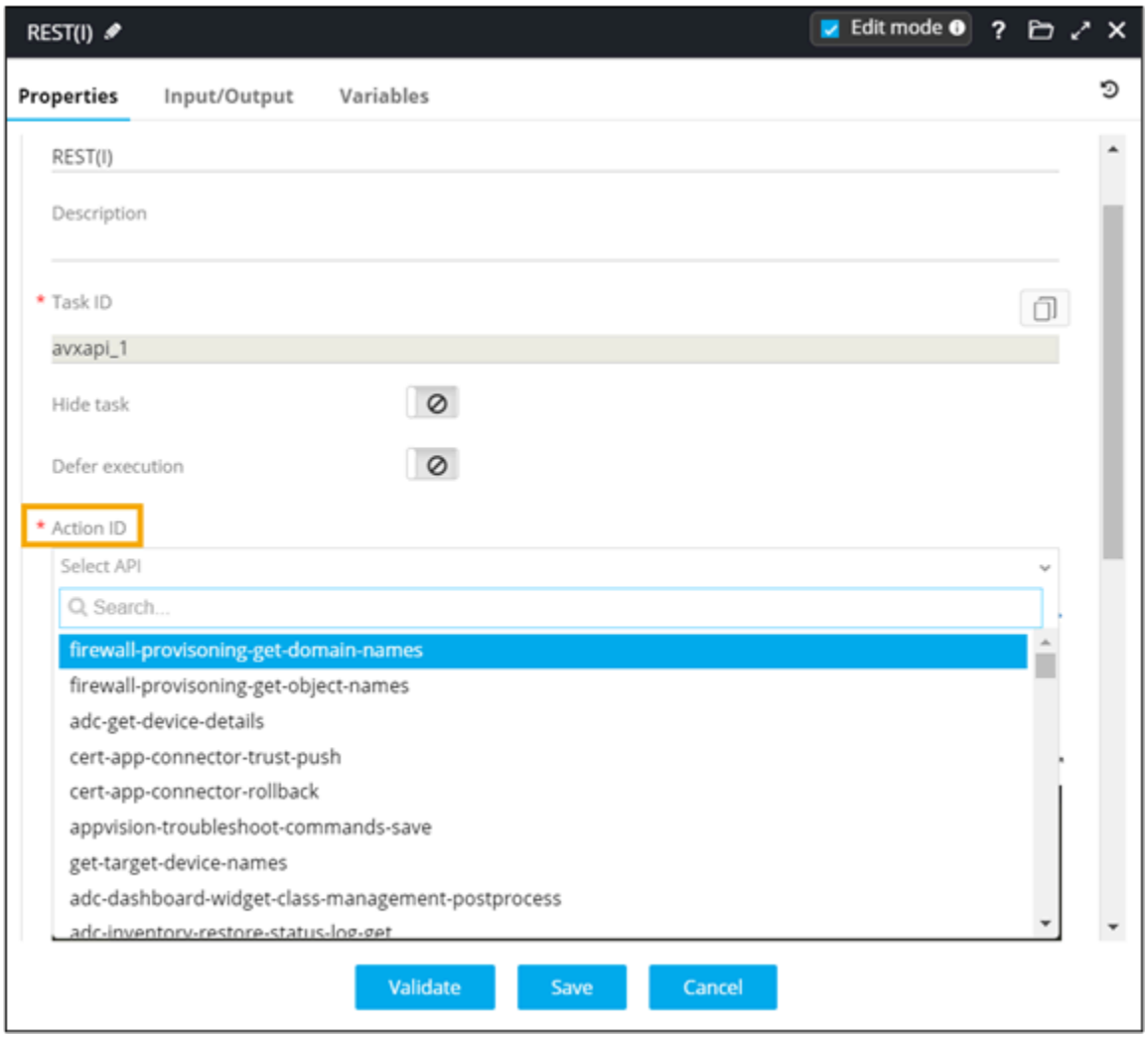
3. Click **Save**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

REST(I)

REST (I) is an API Task used for internal communication within AppViewX modules such as Device Management, Dashboard, Certificate, Firewall, Statistics, and Accounts etc. This involves using in-house AppViewX APIs.

- Uses the session of the logged in user to make internal API calls
- Option to define Action ID - an internal API that has to be defined in the database collection
- Provision to pass the required payload
- Examples include:
  - Internal communication using ‘Add-device-API’ in order to add a new device through workflow
  - Internal communication using ‘Get unused Objects’ in order to retrieve unused object details from Insight through workflow



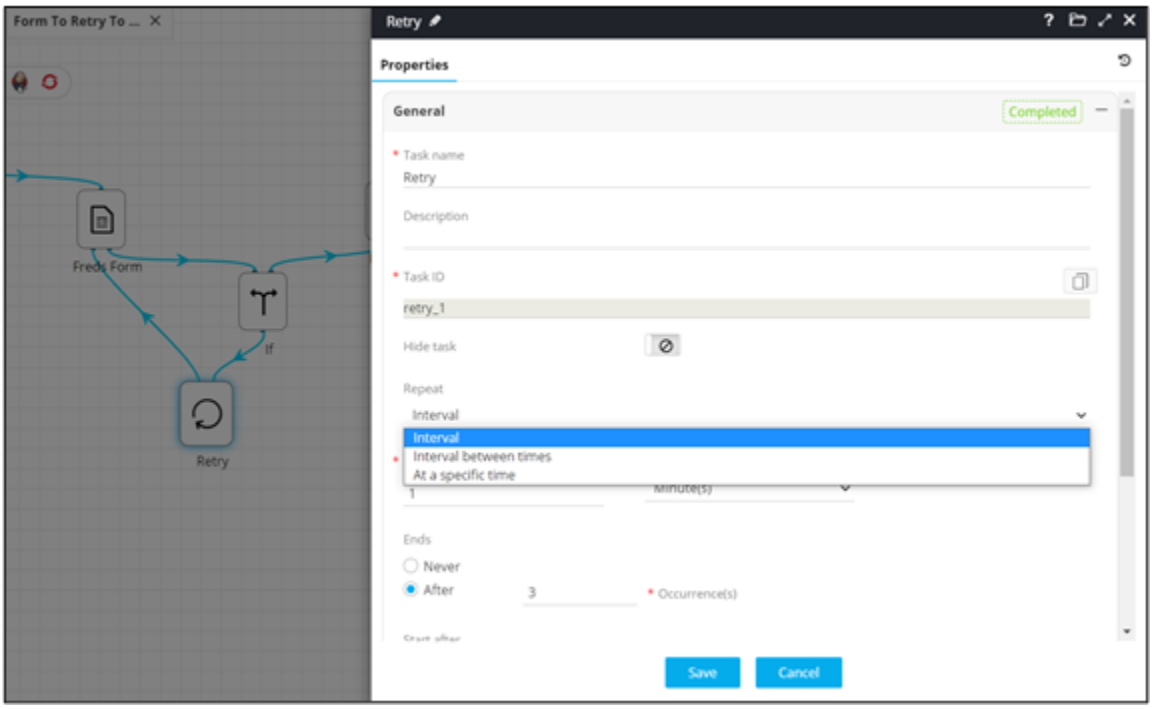
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Note: Relevant APIs have to be added into the database collection (visualworkflow_avxapi) in order to be consumed as Action ID via the REST (I) task.

Retry

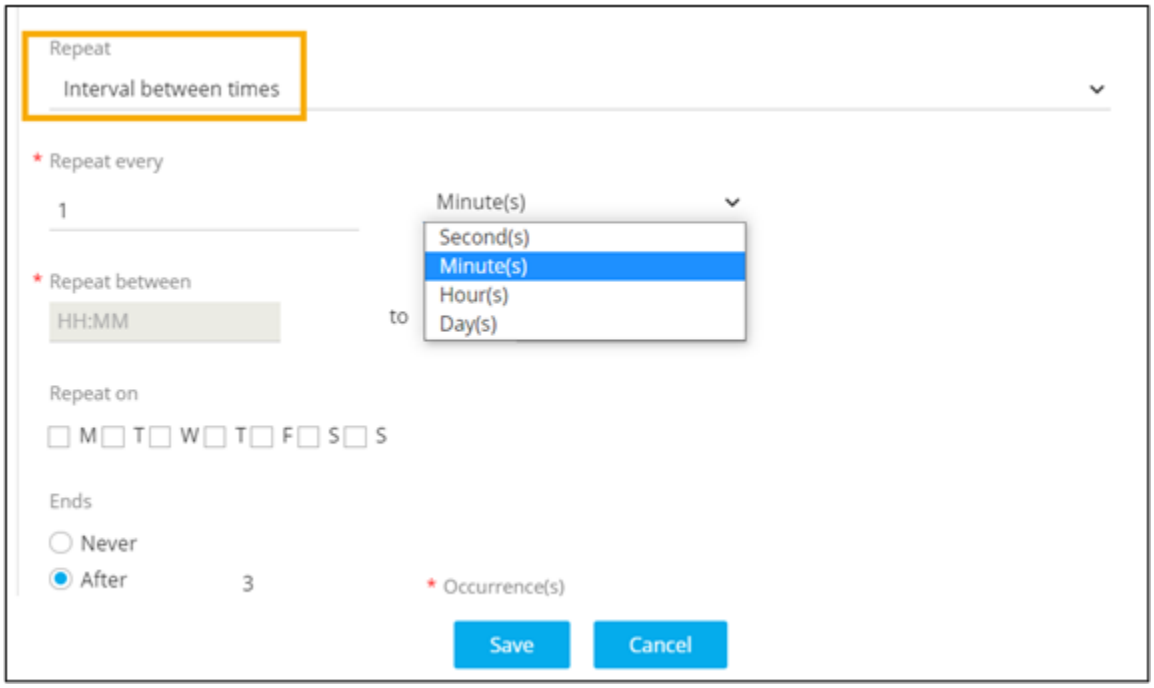
Retry task allows you to define custom retry intervals between workflow task(s) during automation.

- **Interval**: Provision to retry for a certain time interval (seconds, minutes, hours, days) before exit



- **Interval between times**: Provision to retry for a time between specific time intervals before exit

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

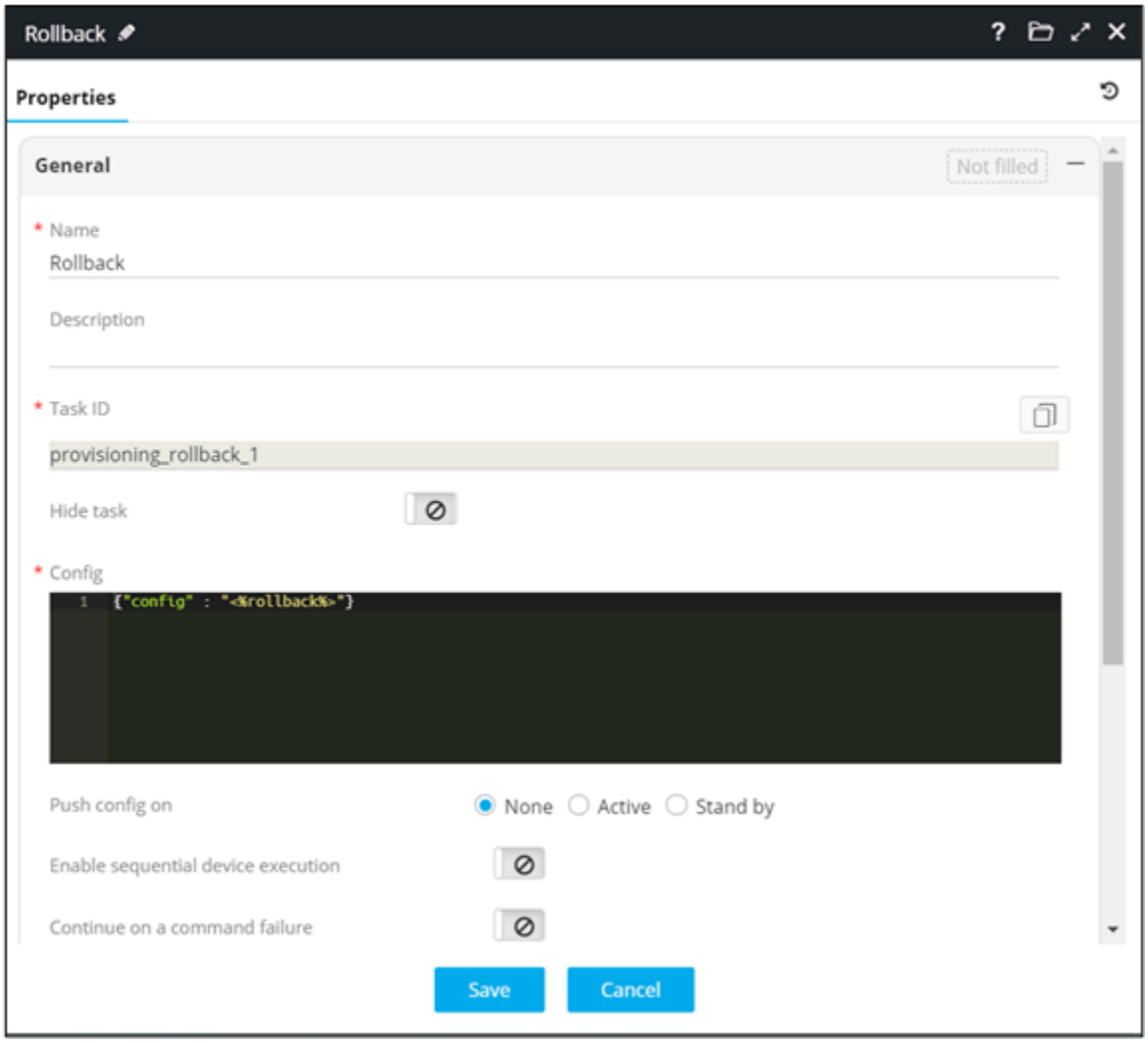


- **At a specific time**: Provision to retry only at a specific time, day before exit

Rollback

This task allows you to rollback implemented changes.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Note: For more information, refer to the section on Rollback Workflow.

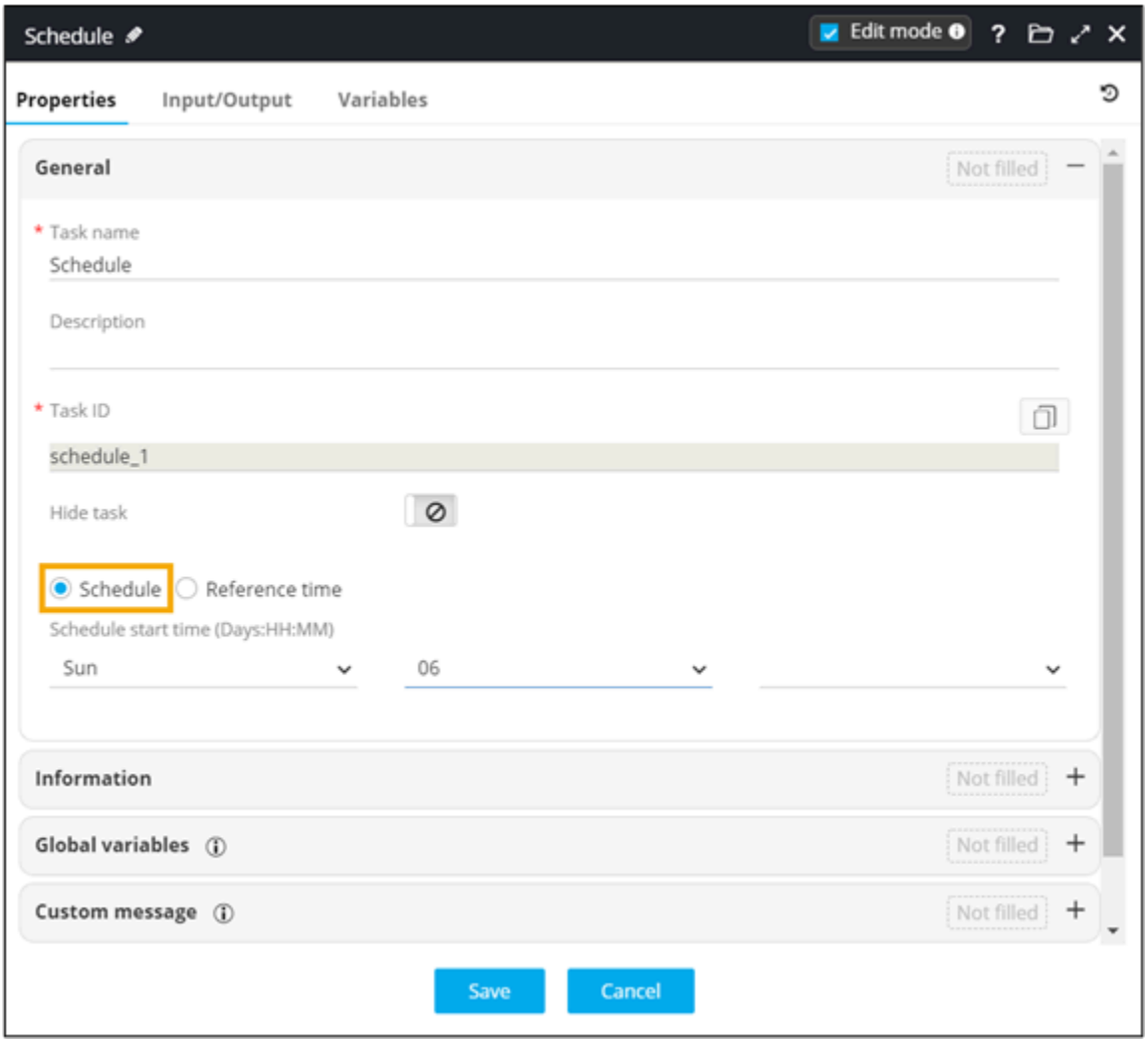
Schedule

The scheduled task allows you to schedule a specific task in the workflow sequence.

- Provision to add a scheduler job between different workflow tasks
- Provision to schedule the job by a specific day of the week, hour and minute
- Provision to pass schedule date, time as variable across workflow tasks

For example, in order to auto implement or schedule the implementation of a work order, you can define a Schedule task after a Review task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Script

Script task allows for defining of custom scripts (using python), business logic as part of the workflow automation process.

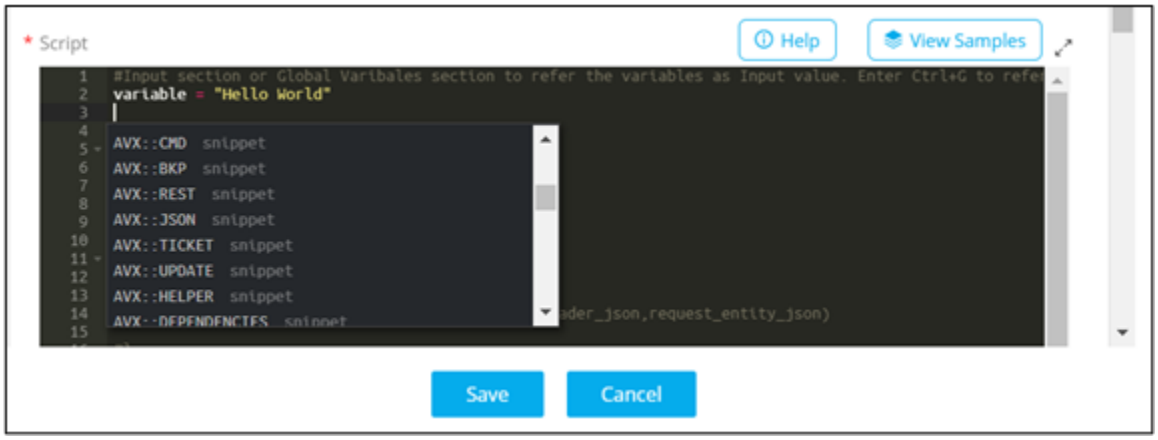
- Supported versions: Python 3.x
- Actual version: Python 3.5
- Provision to debug script with variables
- Allows for comprehensive debugging of the scripts, syntax, and error handling as part of the workflow automation process
- Script tasks can be used with two connections - success and failover

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

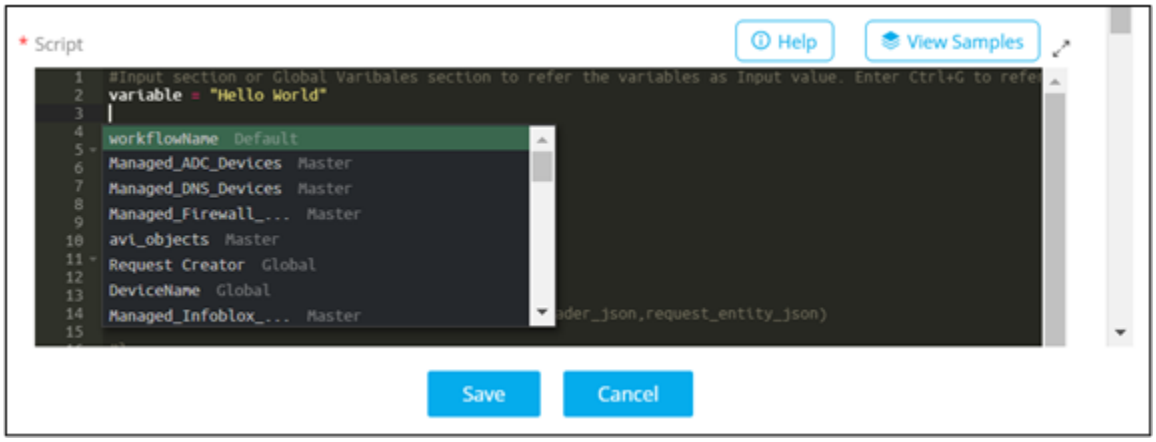
- Provision to select a list of custom commands from the script editor such as AVX::CMD, AVX::OUTPUT, AVX::LOG, AVX::CONFIG and AVX::REST
- Provision to reference global variables
- Using Keyboard Shortcuts
- Custom Commands

Using Keyboard Shortcuts

- Provision to use keyboard shortcut **Ctrl+Space** to get a list of custom commands.

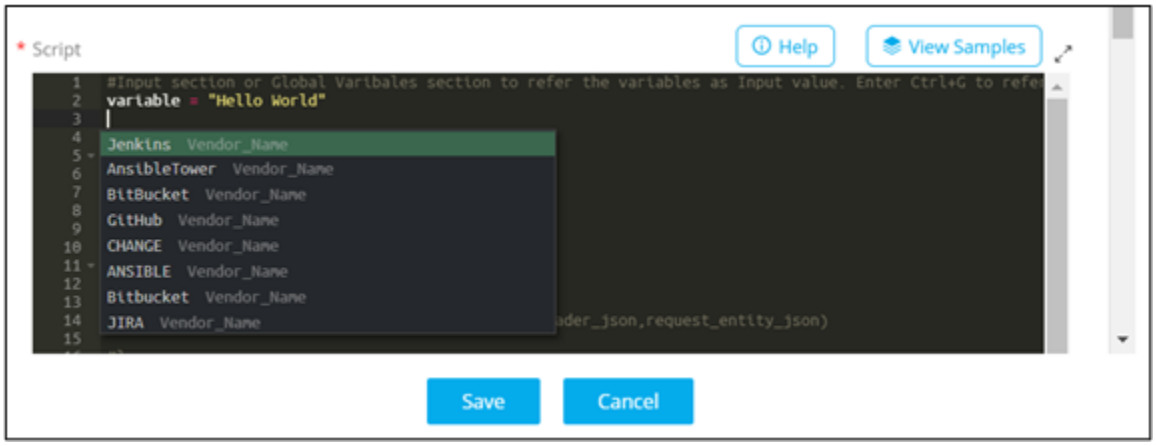


- Provision to use the keyboard shortcut **Ctrl+G** to reference global variable(s).



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Provision to use the keyboard shortcut **Ctrl+K** to load integration variables.



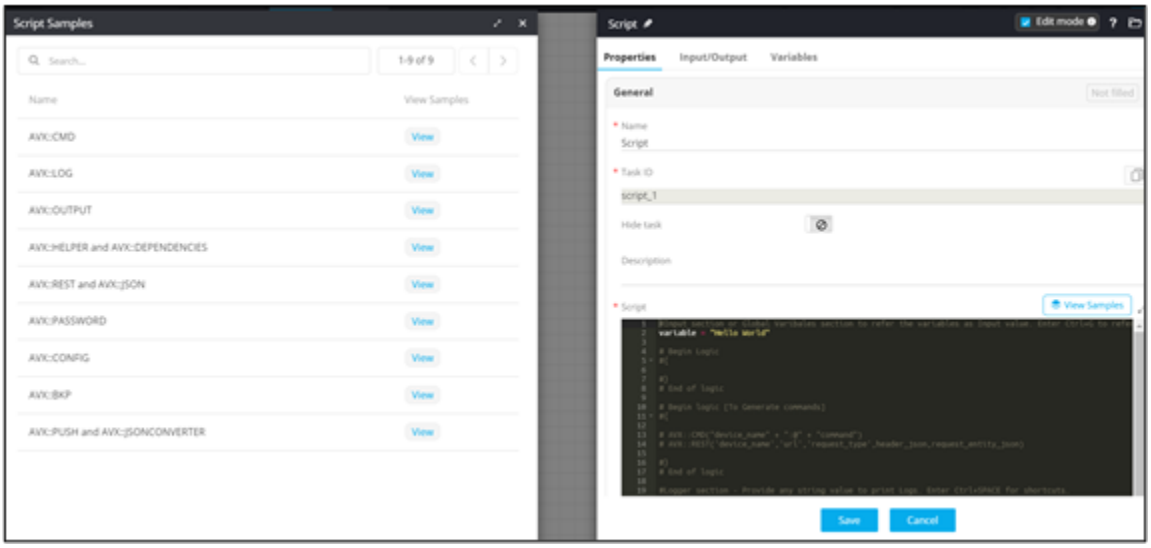
Custom Commands

The visual workflow design process allows you to natively use a few custom commands within Tasks in order to cater to specific use cases. This is an extension to the custom commands used natively in APS such as AVX::CMD, AVX::REST, and AVX::JSON etc.

Following are the command generating functions available in the Visual Workflow Studio:

- AVX::CONFIG
- AVX::LOG
- AVX::OUTPUT
- AVX::CMD
- AVX::REST
- AVX::PUSH
- AVX::JSONCONVERTER
- AVX::MERGECONFIG
- AVX::PASSWORD
- AVX::JSON
- AVX::UPDATE
- AVX::FILEDOWNLOAD

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



- AVX::CONFIG
- AVX::PUSH and AVX::JSONCONVERTER
- AVX::LOG
- AVX::CMD
- AVX::REST and AVX::JSON
- AVX::PASSWORD
- AVX::UPDATE
- AVX::FILEDOWNLOAD
- AVX::OUTPUT
- AVX::MERGECONFIG

AVX::CONFIG

This command is used when other commands like AVX::CMD, AVX::REST and AVX::JSON are used within a workflow task such as Script. AVX::CONFIG is used to assign the commands generated script under a specific key in a python dictionary or a variable. In a workflow, the commands to be generated can be assigned using AVX::CONFIG (within a single script) and can be referenced anywhere across the workflow.

Usage:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

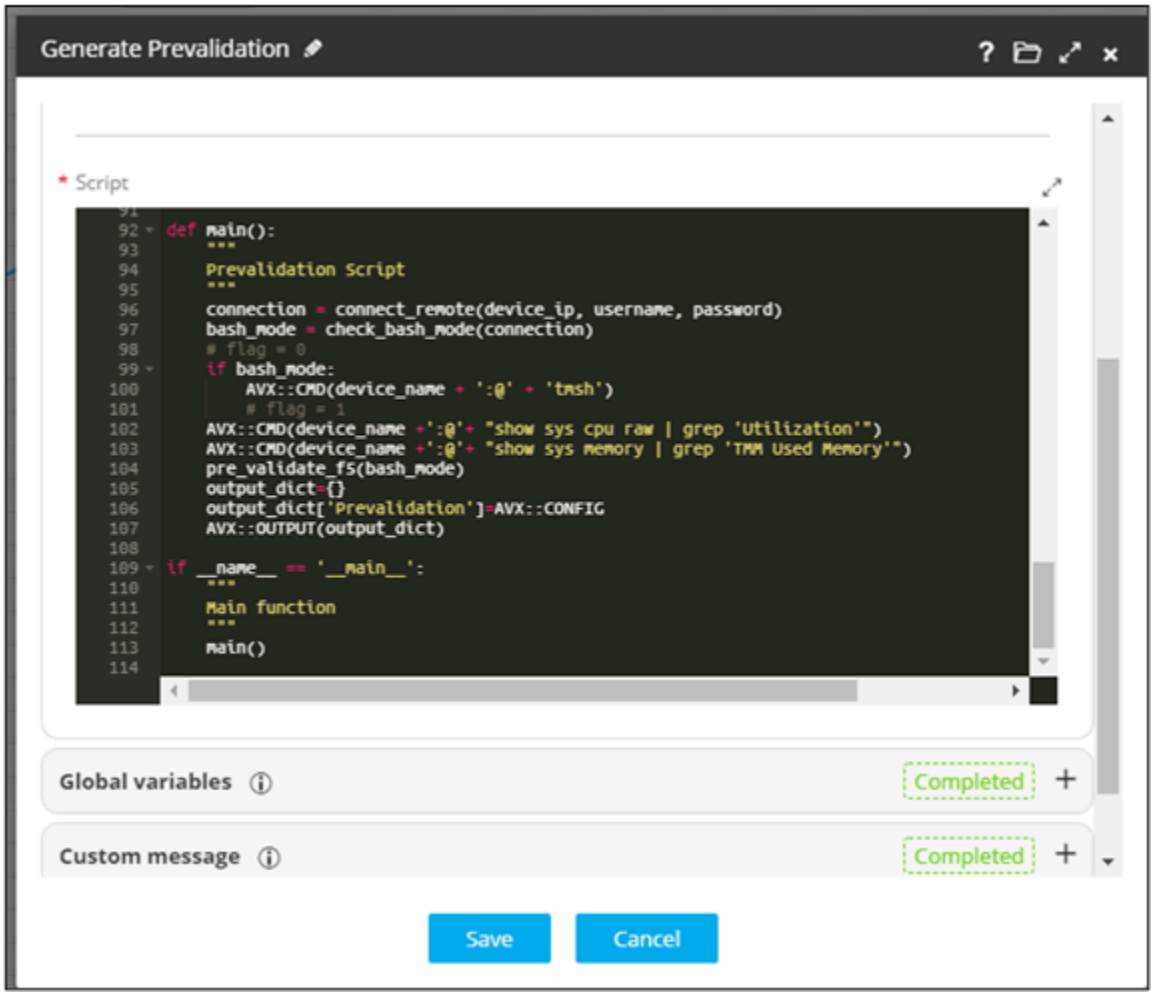
AVX::CMD(“testdevice”+':@'+”implementation command 1”) AVX::CMD(“testdevice”+':@'+”implementation command 2”) commands ={} commands[“implementation”] = AVX::CONFIG	• Commands 1 and 2 will get added to the implementation key. • This will add up the commands generated in the script to the provided key.

Example: Generating F5 VIP prevalidation commands

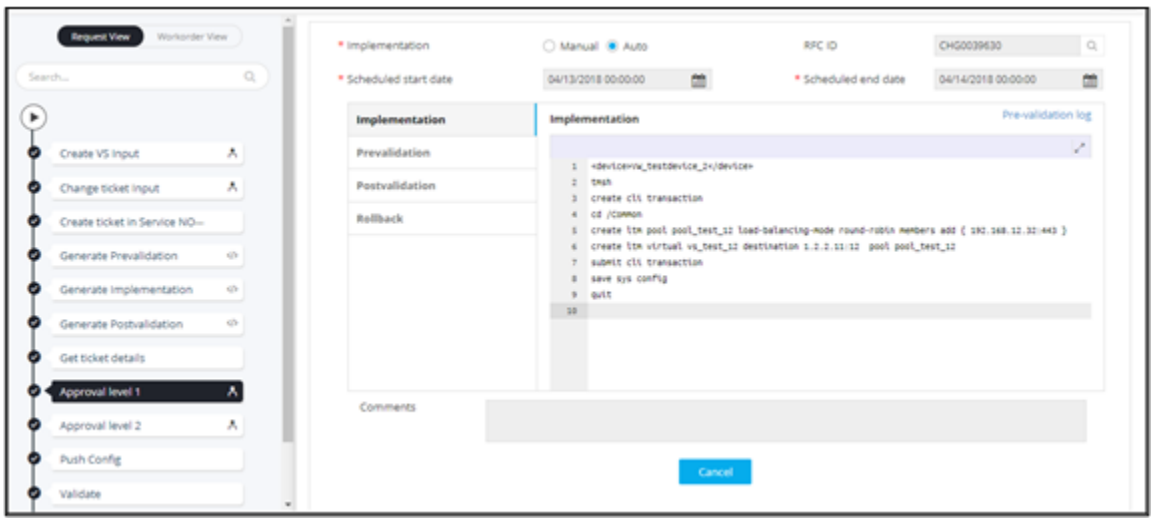
```
AVX::CMD(device + ':@' + 'tmsh')
AVX::CMD(device + ':@' + 'list ltm pool ' + pool_name)
AVX::CMD(device + ':@' + 'list ltm virtual ' + vip_name)
AVX::CMD(device + ':@' + 'quit')


config = {}
config['prevalidation_config'] = AVX::CONFIG
AVX::LOG('Configurations generated successfully.')
```

- AVX::CONFIG within script

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Prevalidation commands generated during workflow execution:



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Create F5 Sample Config:

```
import sys

import json

sys.path.insert(0,AVX::DEPENDENCIES)

sys.path.insert(0,AVX::HELPER)

import os

import f5

import appviewx

import Decrypt_Python3 as Decrypt

from f5 import *

from Decrypt_Python3 import getpassword, getencoded

import paramiko

null=None

connect_db=appviewx.db_connection()

db = connect_db.appviewx

vendor = 'F5'

device_name = '<%Device%>'

fqdn = '<%AppName%>'.lstrip('www.').split('.')[0]

vip_ip = '<%IP%>'

vip_port = '<%Port%>'

persistence = '<%Persistence%>'

create_http_profile = '<%CreateHTTPProfile%>'

create_http_monitor = '<%CreateHTTPMonitor%>'

lb_method = '<%LoadBalancingMethod%>'

defaults_from = '<%DefaultsFrom%>'

ratio = '<%Ratio%>'

interval = '<%Interval%>'

timeout = '<%Timeout%>'

send_string = '<%SendString%>'

receive_string = '<%ReceiveString%>'

existing_monitor = '<%ExistingMonitor%>'

irulecheck = '<%irulecheck%>'

irulefile = '<%uploadedfile%>'

pool_members = <%PoolMembers%>

partition='Common'

output_dict = {}

cmd2 = []
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
deviceInfo = db.device
devVersion = deviceInfo.find_one({'name':device_name})['version']
device_ip = deviceInfo.find_one({'name':device_name} , {"ip": 1})['ip']
username, password = Decrypt.getpassword(device_name)
try:
    tmos_version =deviceInfo.find_one({"name": device_name}, {'detailedVersion': 1})['detailedVersion'].split(" ")[0]
except  Exception:
    tmos_version = None
def connect_remote(IP,username,password):
    try:
        remote_conn=paramiko.SSHClient()
        remote_conn.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        remote_conn.connect(IP,username=username,password=password,look_for_keys=False,allow_agent=False)
        return remote_conn
    except Exception as e:
        print (str(e))
def check_bash_mode(connection) :
    stdin, stdout, err=connection.exec_command('echo "testing"')
    err = err.read().decode()
    if "Error" not in err:
        return True
    else:
        return False
def create_f5():
    '''
    Create F5 vip
    '''
    monitors = []
    cmd = []
    connection = connect_remote(device_ip, username, password)
    bash_mode = check_bash_mode(connection)
    if bash_mode:
        cmd.append('tmsh')
        cmd2.append('tmsh')
    if tmos_version and tmos_version not in ('11.4.1','11.4.3'):
        cmd.append("create cli transaction")
        cmd2.append("create cli transaction")
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
if devVersion in ['v11', 'v12']:
    cmd.append('cd /{0}'.format(partition))
    cmd2.append('cd /{0}'.format(partition))
else:
    cmd.append('modify cli admin-partitions update-partition {0}'.format(partition))
    cmd2.append('modify cli admin-partitions update-partition {0}'.format(partition))
if create_http_monitor == 'Yes':
    http_mon = F5Monitor(**{'type': 'http',
                'app_name': fqdn,
                'vip_port': vip_port,
                'interval': interval,
                'timeout': timeout,
                'send': send_string,
                'recv': receive_string})
    cmd.append(http_mon.create)
    cmd2.append(http_mon.delete)
    mon_index = len(cmd2) - 1
    monitors.append(http_mon)
    output_dict['monitor'] = http_mon.name
if existing_monitor != 'None':
    ex_mon = F5Monitor(**{'type': existing_monitor,
                    'associate': True})
    monitors.append(ex_mon)
    output_dict['monitor'] = ex_mon.name
profile_list = []
if create_http_profile == 'Yes':
    prof = F5Profile(**{'type': 'http',
                'app_name': fqdn,
                'vip_port': vip_port,
                'defaults-from': defaults_from})
    if defaults_from and defaults_from.lower() != "select" and defaults_from.lower() != "none":
        prof_create_cmd = prof.create
    else:
        prof_create_cmd = prof.create.split("defaults-from")[0]
    cmd.append(prof_create_cmd)
    cmd2.append(prof.delete)
    profile_list.append(prof)
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
    output_dict['profile'] = prof.name
else:
    output_dict['profile'] = ''
pool = F5Pool(**{'lb_mode': lb_method,
                 'app_name': fqdn,
                 'port': vip_port,
                 'ratio': ratio,
                 'monitors': monitors,
                 'pool_members': {'keys': ['PoolMemberIP', 'PoolMemberPort', 'Ratio'],
                                  'values': pool_members}})
cmd.append(pool.create)
cmd2.append(pool.delete)
output_dict['pool'] = pool.name
per = F5Persistence(**{'persistence': persistence})
output_dict['persistance'] = per.name
vip = F5VirtualServer(**{'app_name': fqdn,
                 'vip_ip': vip_ip,
                 'vip_port': vip_port,
                 'pool': pool,
                 'profiles': profile_list,
                 'persistence': per})
cmd.append(vip.create)
cmd2.append(vip.delete)
vip_index = len(cmd2) - 1
try:
    cmd2[vip_index], cmd2[mon_index] = cmd2[mon_index], cmd2[vip_index]
except:
    pass
output_dict['vip'] = vip.name
if irulecheck == "Yes":
    # AVX::LOG(str(type(irulefile))+" "+str(irulefile))
    fileName = irulefile['fileName']
    #filename = 'irulee1.txt'
    try:
        # file_location = '/home/'+username+os.sep
        file_location = '/config/'
        destination_json = AVX::JSONCONVERTER(device_name,'$$username$$', '$$password$$', file_location)
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
            AVX::PUSH('N/A','None', destination_json,'sftp', irulefile)
            # AVX::LOG("iRule Successfully pushed to device")
        except:
            pass
    if tmos_version and tmos_version not in ('11.4.1','11.4.3'):
        cmd.append('submit cli transaction')
        cmd2.append('submit cli transaction')
    if irulecheck == 'Yes':
        # fileName = irulefile['fileName']
        # file_location = '/home/'+username+os.sep
        cmd.append("load sys config file "+file_location+fileName+" merge")
    cmd.extend(['save sys config', 'quit'])
    cmd2.extend(['save sys config', 'quit'])
    for c in cmd:
        AVX::CMD(device_name+':@'+c)
def main():
    '''
    Main method
    '''
            if vendor == 'F5':
        create_f5()

    output_dict['Implementation']=AVX::CONFIG

    for c in cmd2:
        AVX::CMD(device_name+':@'+c)

    output_dict['Rollback']=AVX::CONFIG

    AVX::OUTPUT(output_dict)

if __name__ == '__main__':
    '''
    Main function
    '''
main()
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

AVX::PUSH and AVX::JSONCONVERTER

These commands are used to push file(s) between destinations - local to remote, remote to remote.

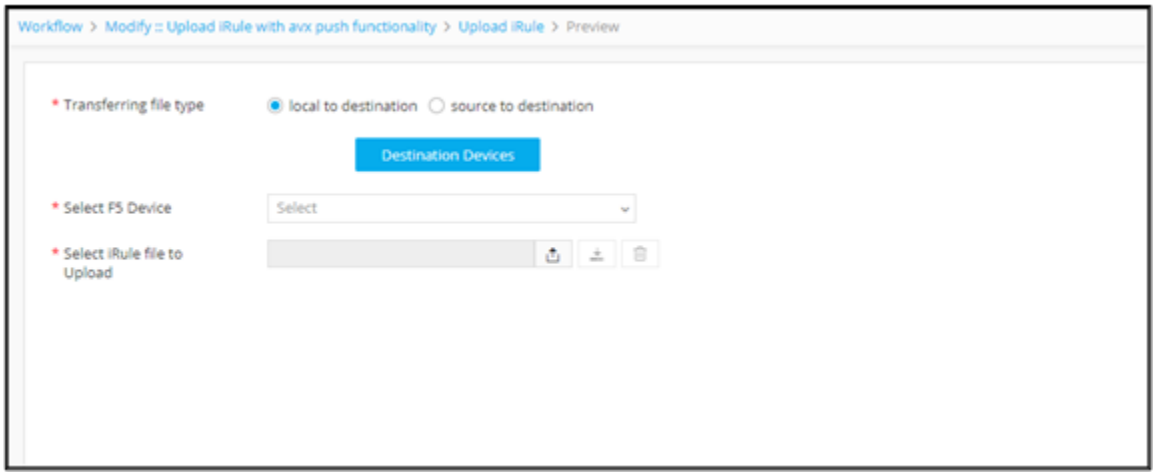
Usage:

```
AVX::PUSH ('device_name',source_json, destination_json,protocol, fieldId , port_number, recursive, seq_no=0, properties={'sleep':0})
```

- Device_name: Name of the device from where the scp/sftp command to be executed.
- Source_json: AVX::JSONCONVERTER('ipAddress', 'username', 'password', 'location').
- Destination_json: AVX::JSONCONVERTER('ipAddress', 'username', 'password', 'location').
- Protocol: “scp” or “sftp”
- fieldId: Field Id of the uploaded file in the form
- Port_number: SCP port (Default is 22)
- Recursive: “True” for folders
- Seq_no and other properties remains the same as AVX::CMD

To use these commands:

1. Design a workflow.
2. To allow users to select the source of the destination device and upload file, drag and drop a **Form**task.
3. Configure the form fields as shown.



4. To get inputs from the form and push file(s) to a destination (AVX::PUSH logic), drag and drop a **Script**task.

```
#load sys config file <irulepath> verify merge


import sys
```

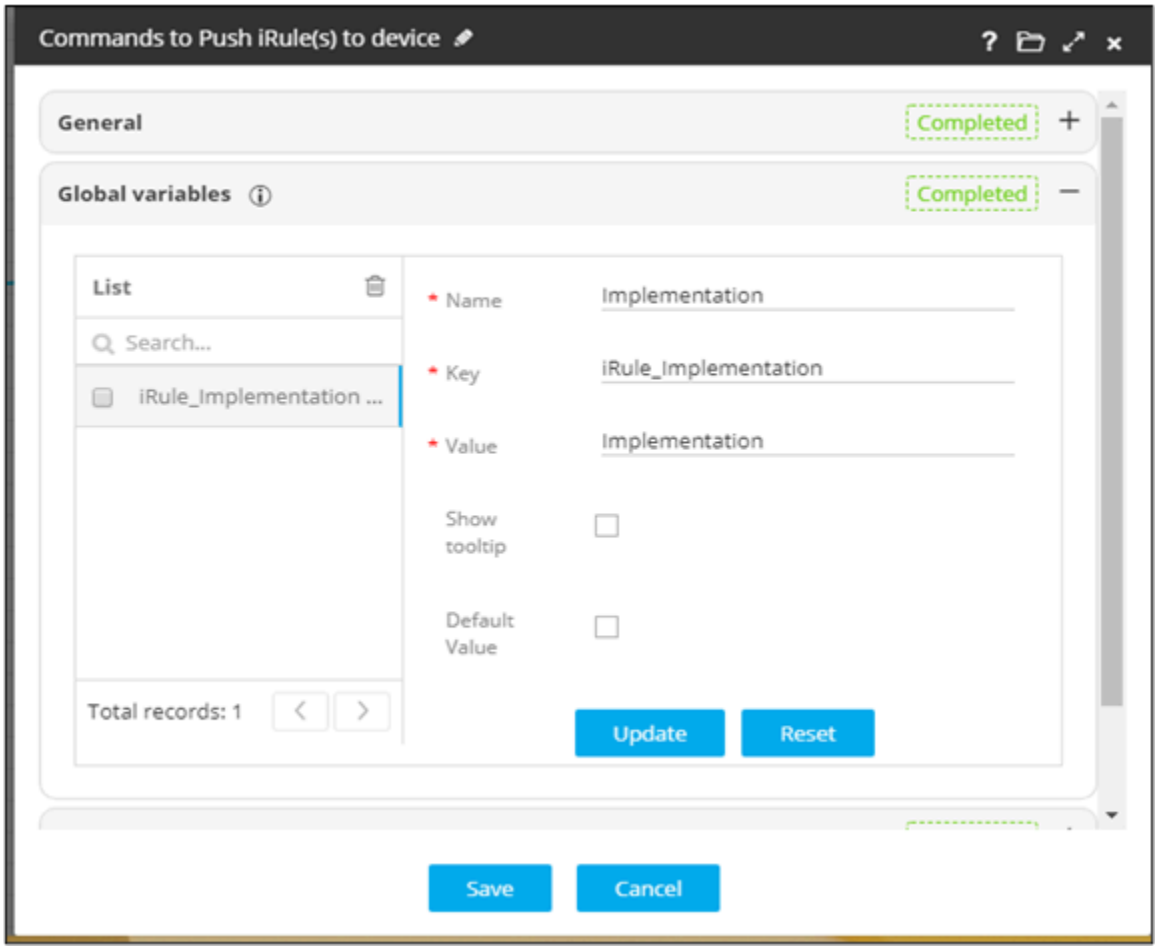
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
import os
sys.path.insert(0,AVX::HELPER)
sys.path.insert(0,AVX::DEPENDENCIES)
import paramiko
import appviewx
import Decrypt_Python3 as Decrypt
connection = appviewx.db_connection()
db = connection.appviewx
device_name = "<%device%>"
deviceInfo = db.device
transfer_type = '<%transfer_type%>'


AVX::LOG(transfer_type)
if transfer_type == "local to destination":
    try:
        #device_ip,username,password = get_device_details(device_name)
        irulefile = '<%uploadedfile%>'
        fileName = irulefile['fileName']
        file_location = '<%destination_path%>'
        destination_json = AVX::JSONCONVERTER(device_name,'$$username$$', '$$password$$', file_location)
        AVX::PUSH('N/A','None', destination_json,'sftp', irulefile)
        AVX::LOG("iRule Successfully pushed to device")
        AVX::CMD(device_name+":@tmsh")
        AVX::CMD(device_name+":@load sys config file "+file_location+fileName+" merge")
        AVX::CMD(device_name+":@quit")
    except Exception as e:
        AVX::LOG(str(e))
else:
    try:
        #AVX::PUSH('device_name',source_json, destination_json,protocol, fieldId , port_number, recursive, seq_no=0, properties={'sleep':0})
        source_path = '<%source_path%>'
        source_device = "<%source_device%>"
        source_json = AVX::JSONCONVERTER(source_device,'$$username$$', '$$password$$', source_path)
        file_location = '<%destination_path%>'
        file_name = source_path.split('/')[-1]
        destination_json = AVX::JSONCONVERTER(device_name,'$$username$$', '$$password$$', file_location)
        AVX::PUSH(source_device,source_json, destination_json,'scp')
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

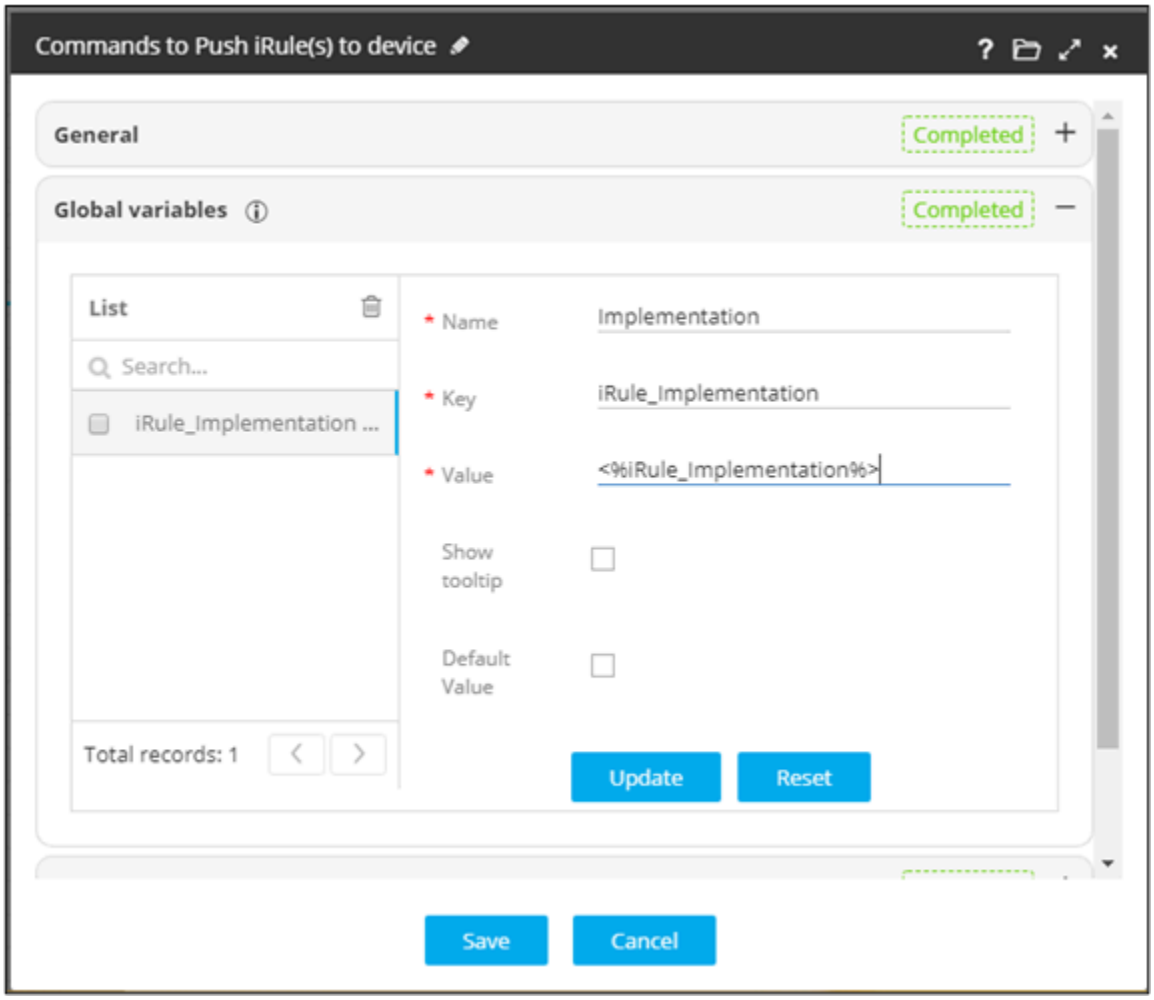
```
        AVX::LOG("iRule Successfully pushed to device")
        AVX::CMD(device_name+":@tmsh")
        AVX::CMD(device_name+":@load sys config file "+file_location+source_path.split('/')[-1]+" merge")
        AVX::CMD(device_name+":@quit")
    except Exception as e:
        AVX::LOG(str(e))


config = {}
config["Implementation"] = AVX::CONFIG


AVX::OUTPUT(config)
```

5. Define the configuration to be generated from the script as a global variable value.



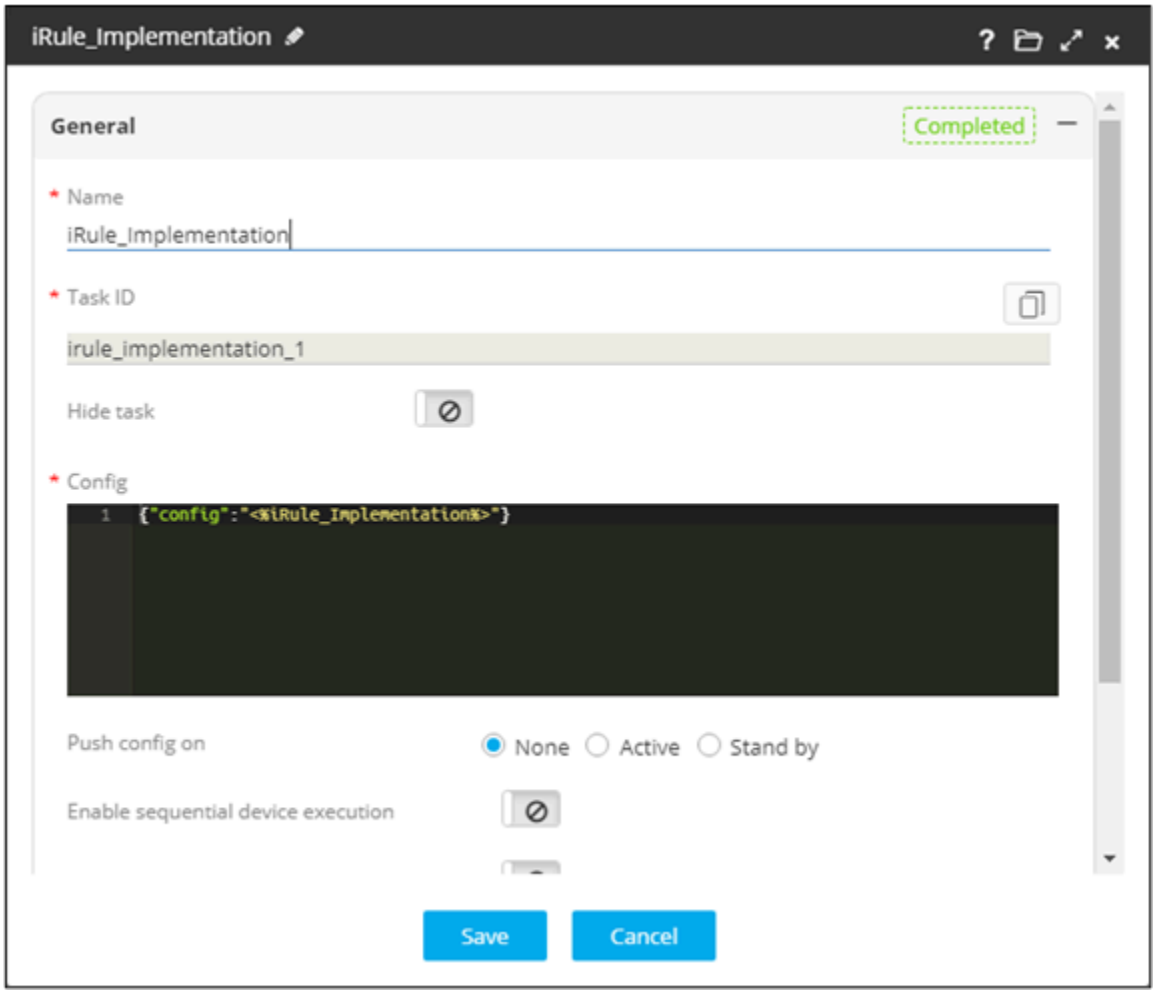
6. To display the configurations to be pushed to a device, drag and drop a **Review** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



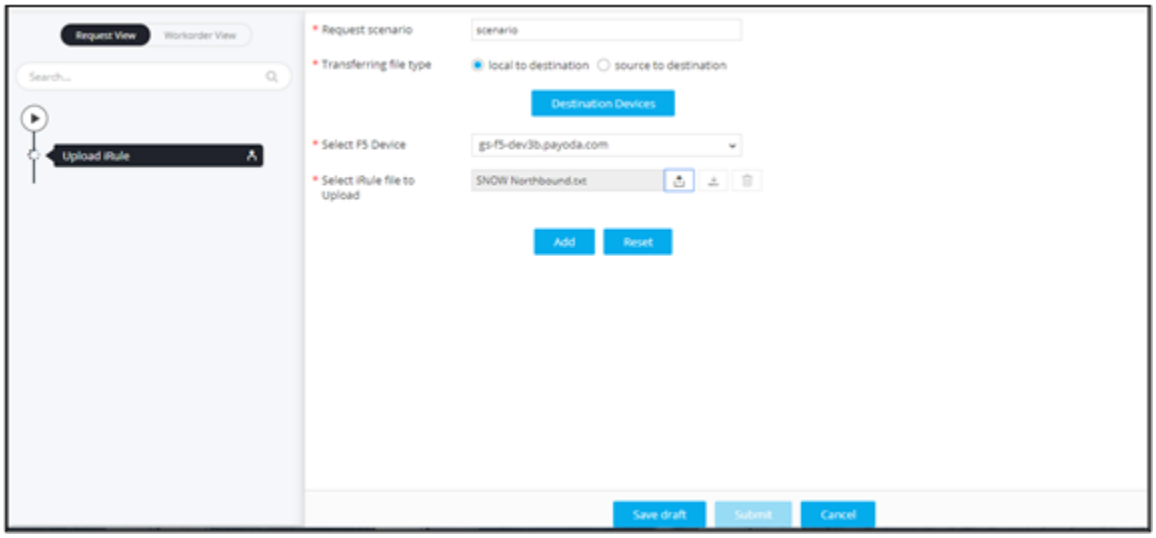
7. To push the configuration on the end device, drag and drop the **Implementation** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

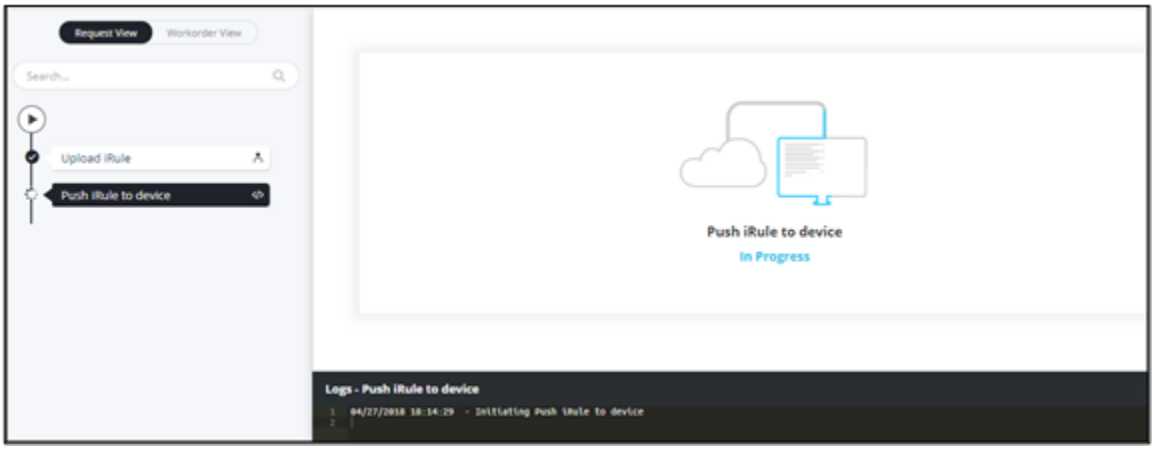


8. Refer the global variable from the Script task.
9. Enable and trigger the workflow.
10. Select the destination device and upload the file.

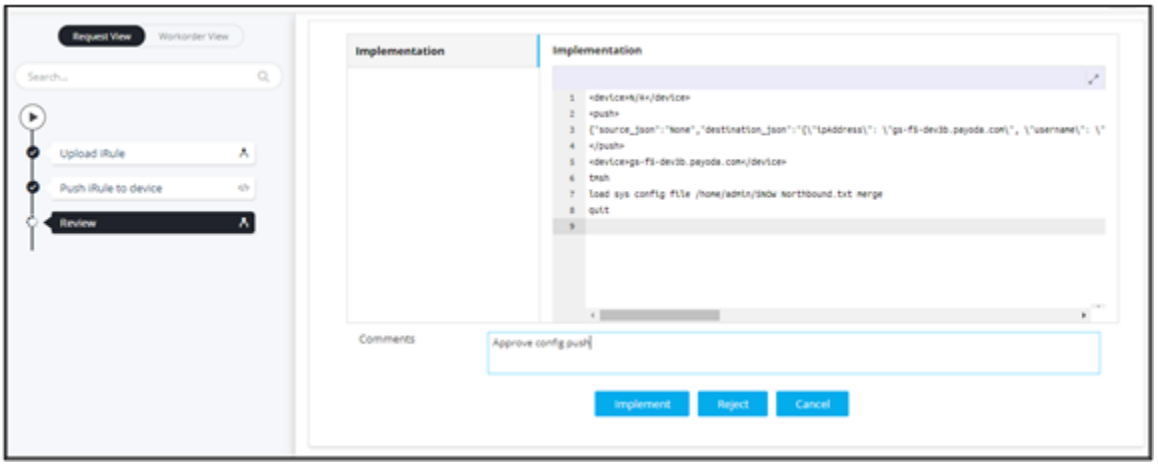
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



- Push iRule to device task in in progress.

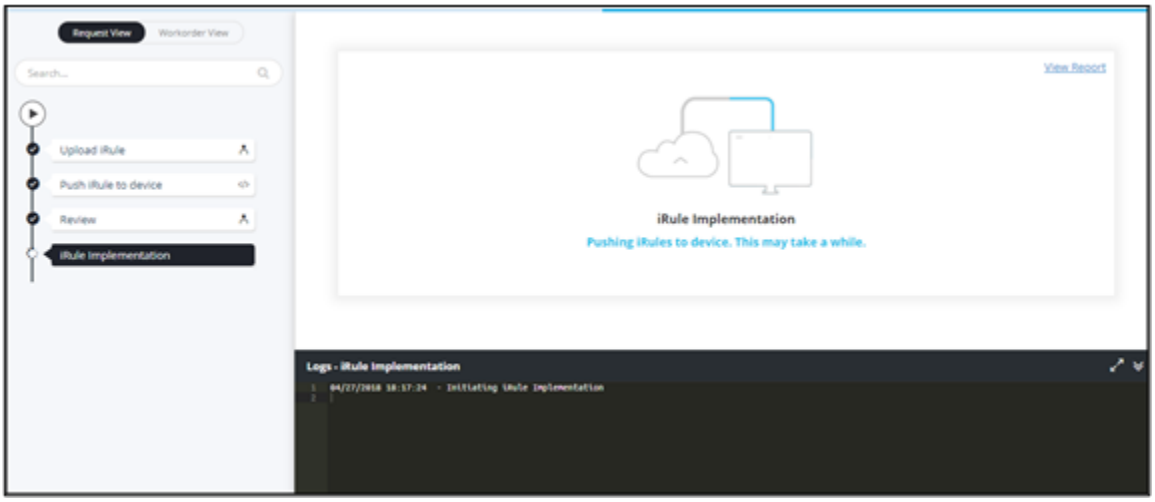


- Implementation stage.

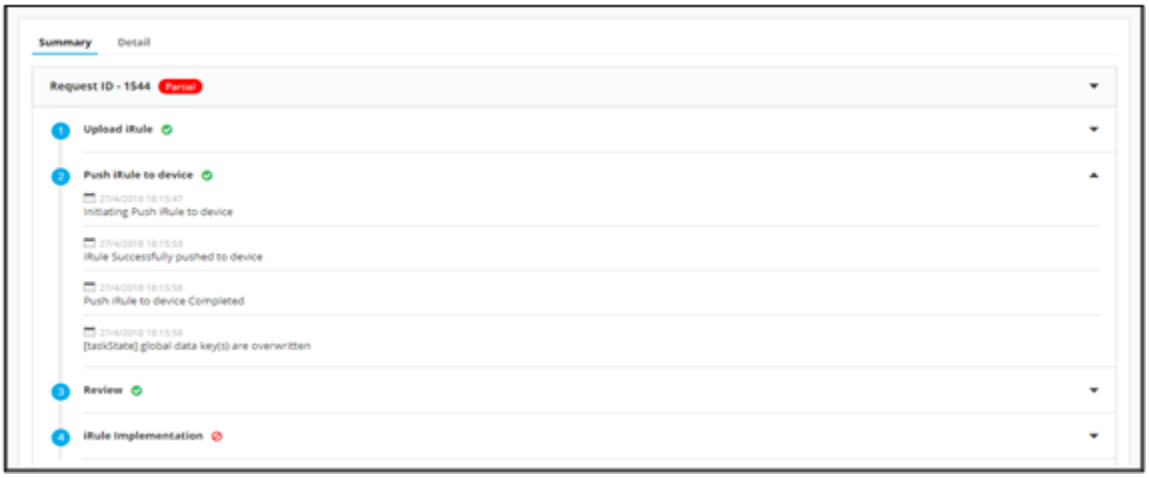


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Implementation in progress.



- Summary of workflow execution after file push.



AVX::LOG

This command allows you to record and print custom messages in the logs (in the request stage view). AVX::LOG takes a string value (i.e. the message to be logged) as an argument and prints it in the log. AVX::OUTPUT collates all the logs and finally prints them.

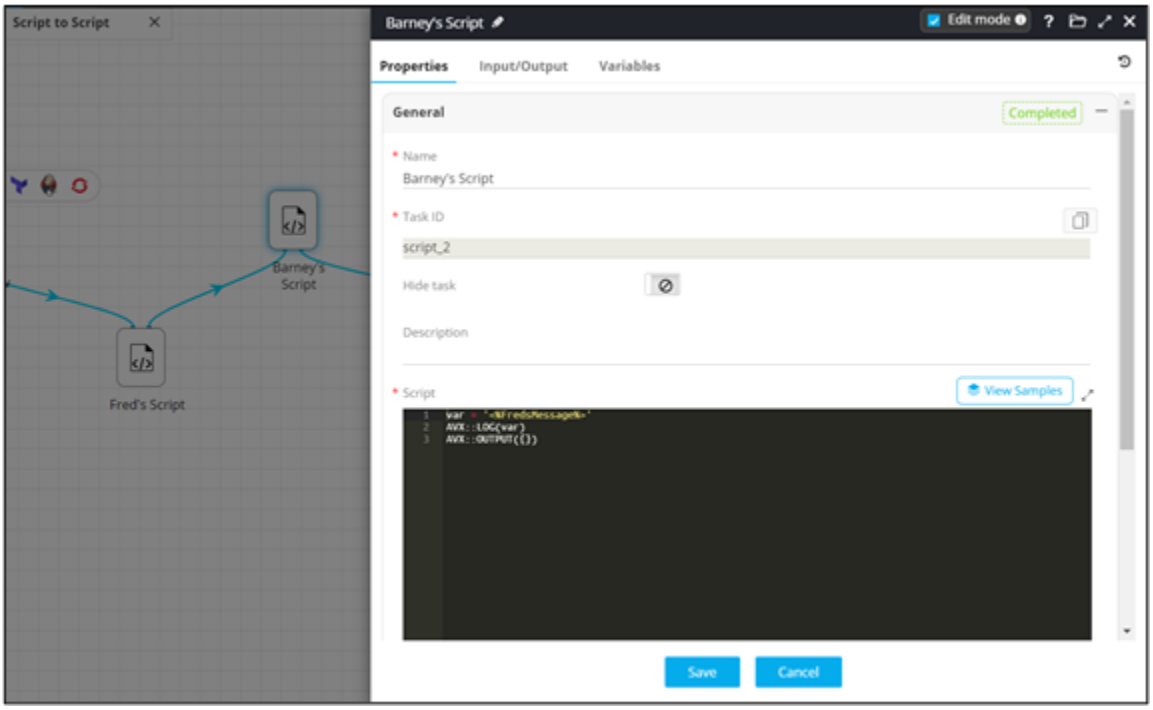
Usage:

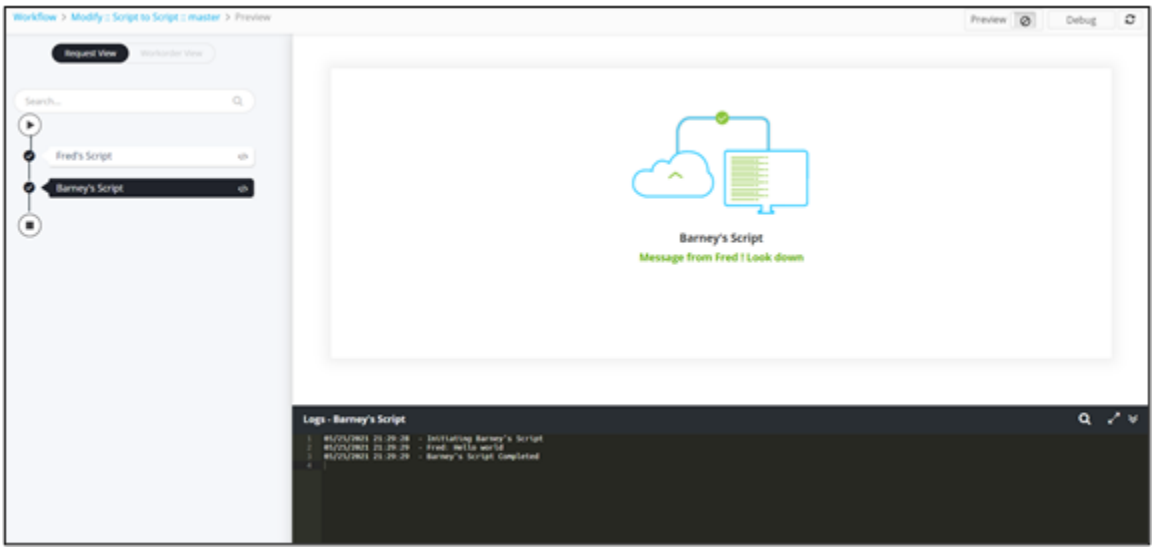
```
AVX::LOG(“Print a simple log message”)

AVX::OUTPUT({})
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Example 1:

```
message_from_fred = '<%FredsMessage%>'

AVX::LOG("Fred's Message is "+message_from_fred)

AVX::OUTPUT({})
```

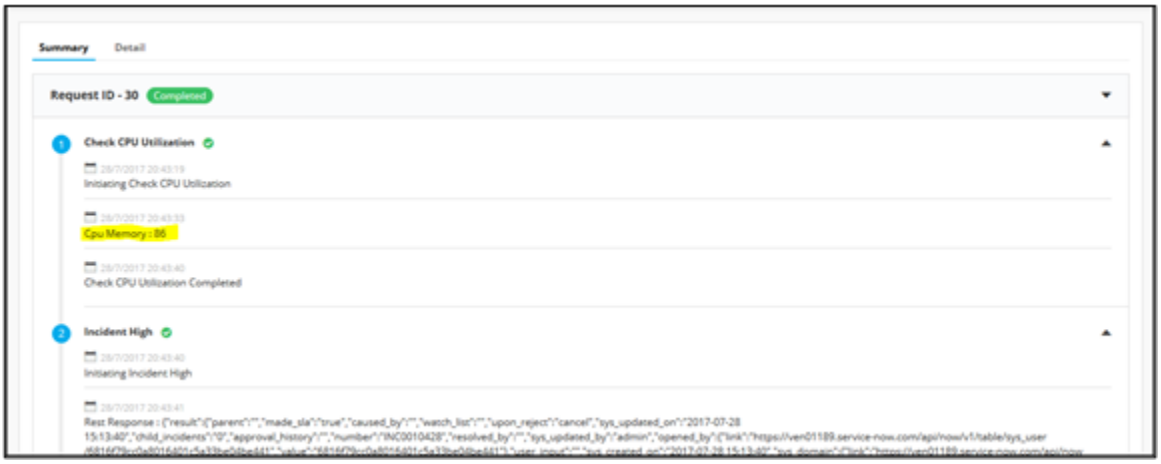




Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Example 2:

```
AVX::LOG("Creation of Virtual Machine Completed Successfully")

cpu_mem = int(stdout.read())

 outjson={"cpu_mem":cpu_mem , "ci" : "F5_192_41_101","description":"CPU utilization exceeded 40 % | Current cpu utilization is {} %".format(cpu_mem) }

AVX::LOG("Cpu Memory : "+str(cpu_mem))
```



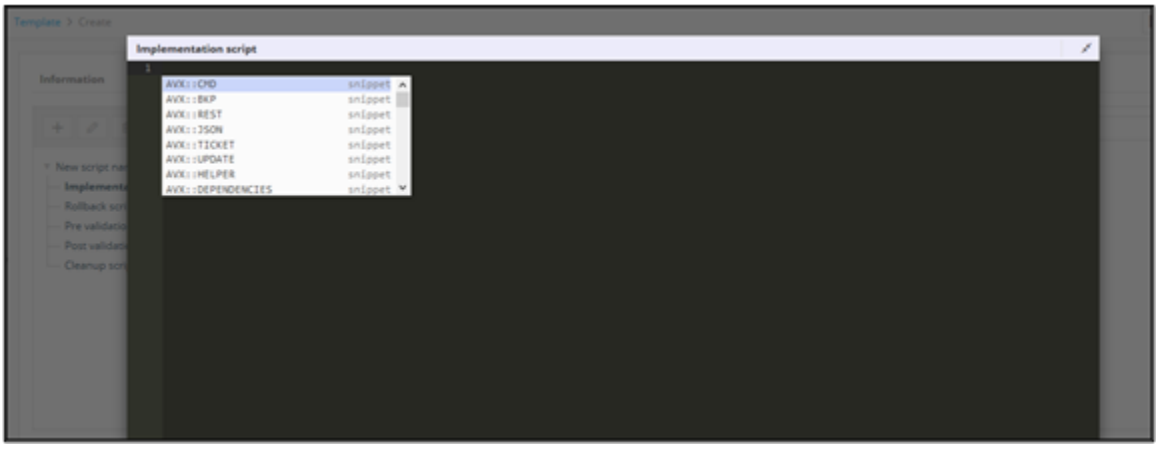
AVX::CMD

```
AVX::CMD('device_name' +':@'+ 'command',interactive='None|command|input',seq_no=0,properties={'sleep':0,'wait':0})
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

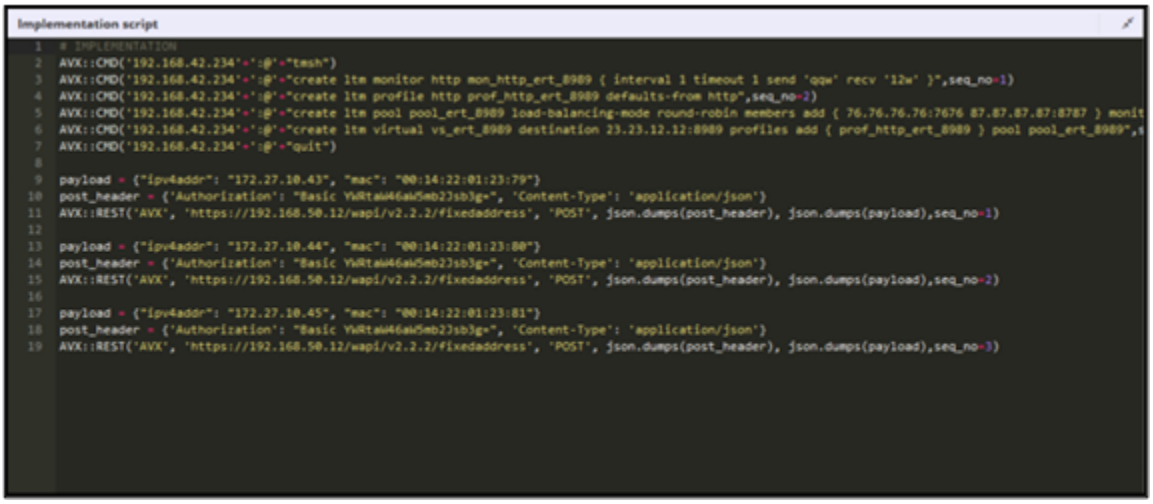
‘device name’	Name of the device on which the command needs to be executed
‘command’	Actual command to be executed on the device
‘interactive’	• Used if the command needs to be executed where devices require interactive prompts (such as A10) • Similar to providing a password for SSH
‘Seq_no’	• Provision to define sequence numbers/tags against configurations (Implementation, Rollback, Prevalidation, Postvalidation, Cleanup) • Sequence number ensures a one on one mapping of the Implementation versus Rollback configurations • Used as a provision to rollback ONLY specific configuration(s) that get implemented
‘Sleep’	• Provision to define command specific properties • Specifying ‘sleep duration’ will ensure an explicit wait (for the specified time) after a specific command is executed
‘Wait’	• Provision to define command specific properties • Wait for the specified time to get the response for the implemented command • Should there be no response, it will timeout

Syntax to define Sequence number/Tags for SSH and REST:



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

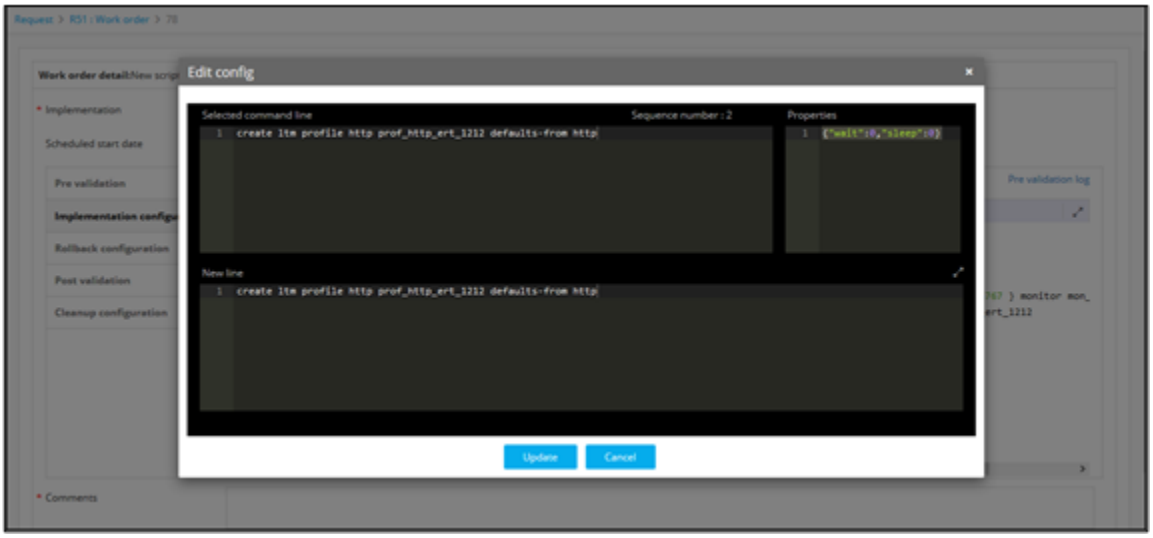
Script with Sequence number/Tags - Implementation Config:



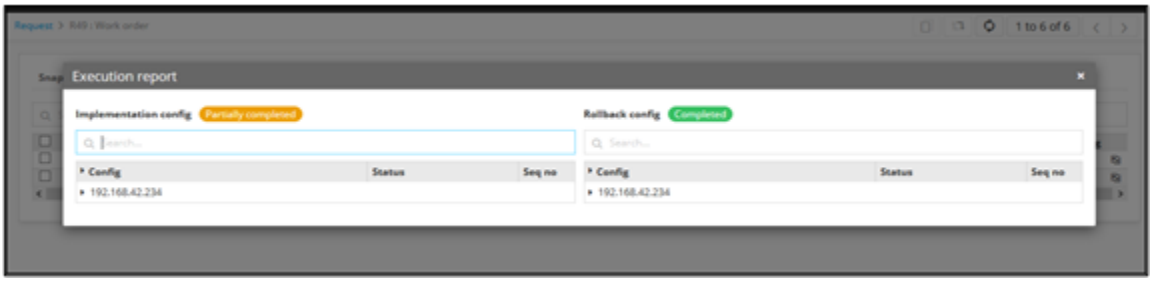
Script with Sequence number/Tags - Rollback Config:

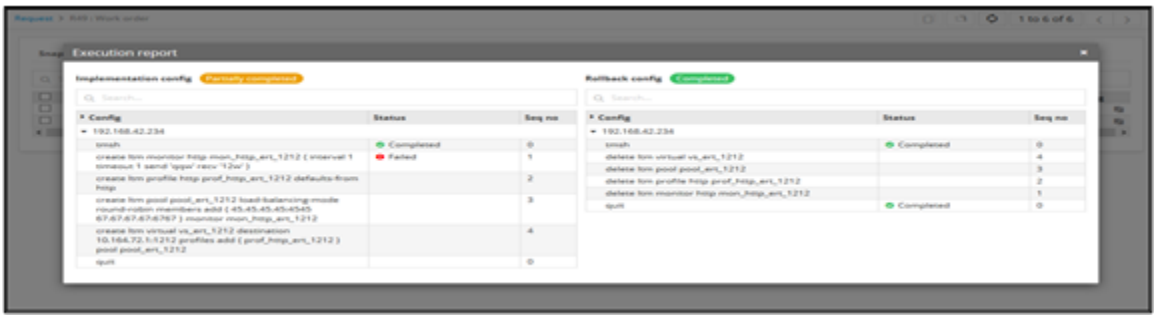
Work order Review pane with option to edit command and properties:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



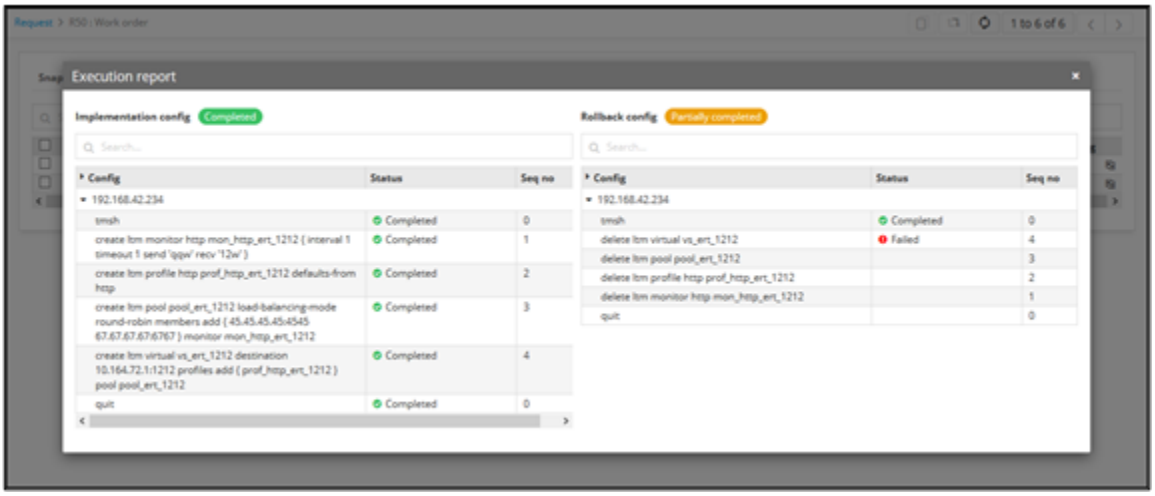
Execution Report - Rollback completion for Partial Implementation:



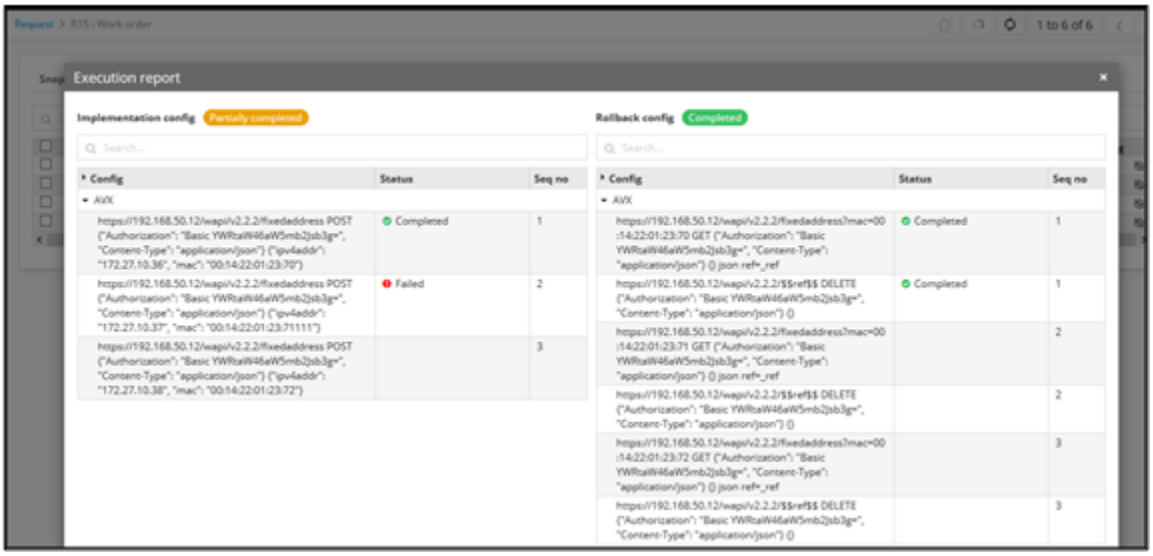


Execution Report for Commands:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Execution Report - REST:



AVX::REST and AVX::JSON

These commands are used via scripts in order to communicate and push files to any REST endpoint.

URL	Endpoint URL to hit the REST
Request_type	“GET”, “PUT”, “POST” or “DELETE”
Header_json	AVX::JSON(json_data,seq_no=0,properties={‘sleep’:0,’wait’:0})

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

URL	Endpoint URL to hit the REST
	Json_data is the json value to be passed to REST
request_entity_json	AVX::JSON(json_data,seq_no=0,properties={‘sleep’:0,’wait’:0})
response_type? required_param	Path_of_the_param is the query and other params which can be provided as string itself

To pass variables when using AVX::REST:

1. Execute REST call.
2. Store the value called “Signature”.

```
auth_url  = "https://{}/axapi/v3/auth".format(deviceip)
    headerjson_auth = {"Content-Type": "application/json"}
    bodyjson_auth = {"credentials": {"username": username ,"password": '@$avx_password@$' }}
    AVX::REST(dev_name,auth_url,'POST',json.dumps(headerjson_auth),json.dumps(bodyjson_auth),'json?sig=authresponse/signature')
```

```
{
  "authresponse" : {
     "signature":"6181d42578cc03dd1088d58884637e",
     "description":"the signature should be set in Authorization header for following request."
   }
}
```

The value is stored in ’json?sig=authresponse/signature’. In the above, json signature lies under authresponse, hence the response_type?required_param will be json?sig=authresponse/signature.

3. The variable signature (sig) is reused in the following REST call’s header.

```
url = 'https://{}/axapi/v3/slb/service-group/{}'.format(deviceip,value)
headerjson = {"Authorization": "A10 $$sig$$", "Content-Type": "application/json"}
bodyjson = {"service-group": { "member-list":  state_pool_dict[value] }}
AVX::REST(dev_name,url,'POST',json.dumps(headerjson),json.dumps(bodyjson))
```

AVX::PASSWORD

This command is used in the event of having to hide any password credentials from being displayed to the user. A key can be used to map the password and the password value can be retrieved with the key.

```
password = 'ninjaturtle@hello'
bodyjson_auth = {"credentials": {"username": username ,"password": '@$avx_password@$' }}
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

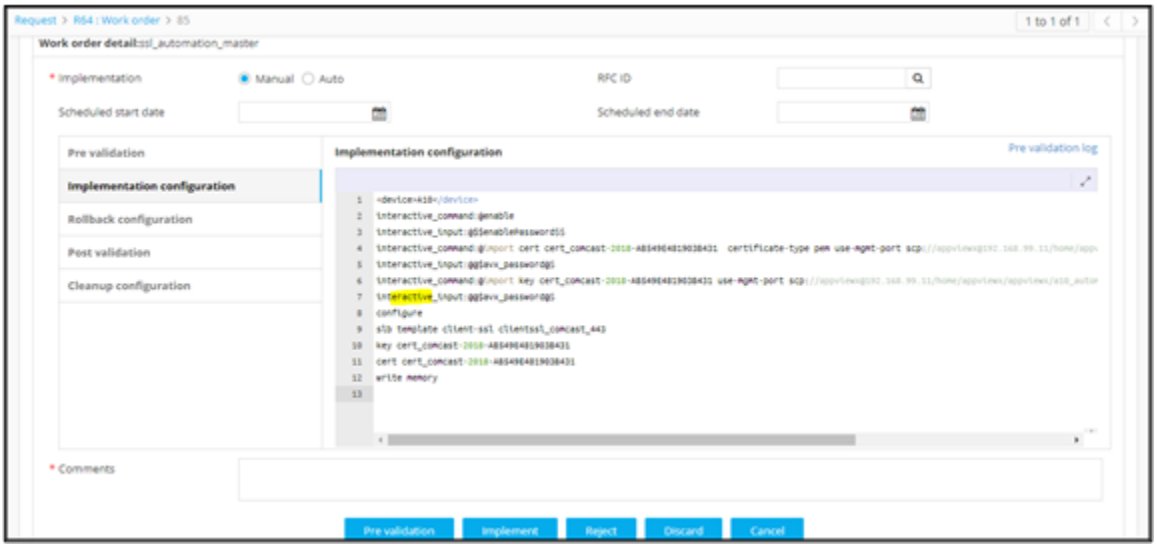
```
AVX::PASSWORD('@$avx_password@$',password)
```

Password	The variable which holds the password For example, ninjaturtle@hello
'@$avx_password@$'	This is to mask the declared password.
AVX::PASSWOR('@$avx_password@$',password)	Mapping of the masked password

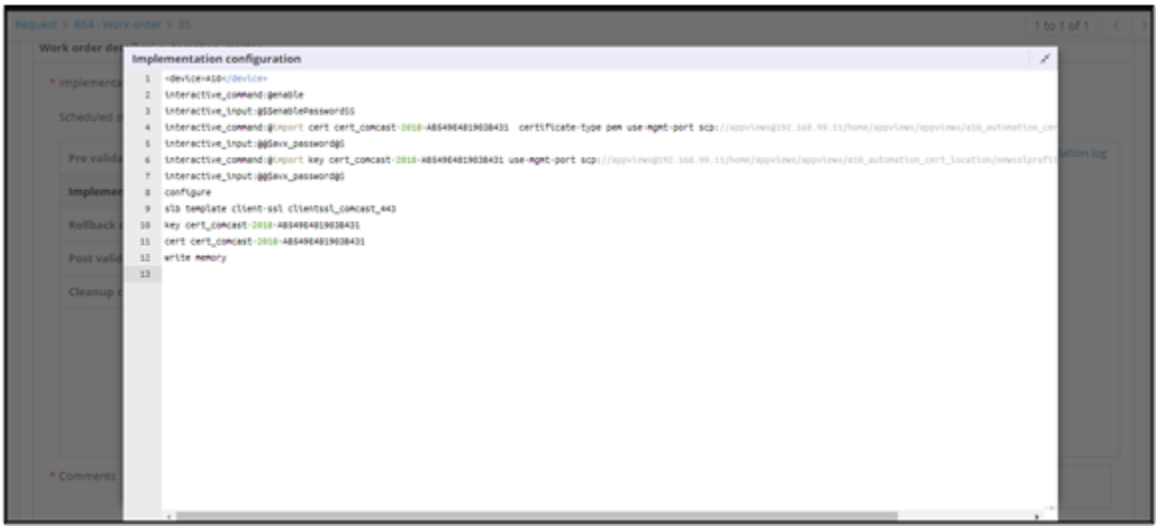
AVX::PASSWORD usage with configuration sample:

```
<device>A10</device>

<rest>

<url>https://172.16.30.110/axapi/v3/auth</url>

<type>POST</type>

<header>{"Content-Type": "application/json"}</header>

<request_entity>{"credentials": {"password": "@$avx_password@$", "username": "admin"}}</request_entity>

<response_parse_type>json</response_parse_type>

<response_param>sig=authresponse/signature</response_param>

</rest>
```

Masking of password from users during Review:



Copyright © 2020 AppViewX, Inc. All Rights Reserved.



AVX::UPDATE

This command is used to update specific parameters within Collections. It is used to filter out the documents from the collection named ‘Collections’.

```
AVX::UPDATE (query_dict,'key','value','action')
```

Query_dict	The dictionary value to be provided as a querying parameter in the Mongo db
Key	Key for which the value needs to be updated in the queried document
Value	The value which needs to be modified for the provided key
Action	It can be either • Add - To add the provided value in the array of values • Remove - To remove the provided value from the array of values

AVX::FILEDOWNLOAD

This command is used to generate a .csv file on runtime based on a dictionary of values and download the file output.

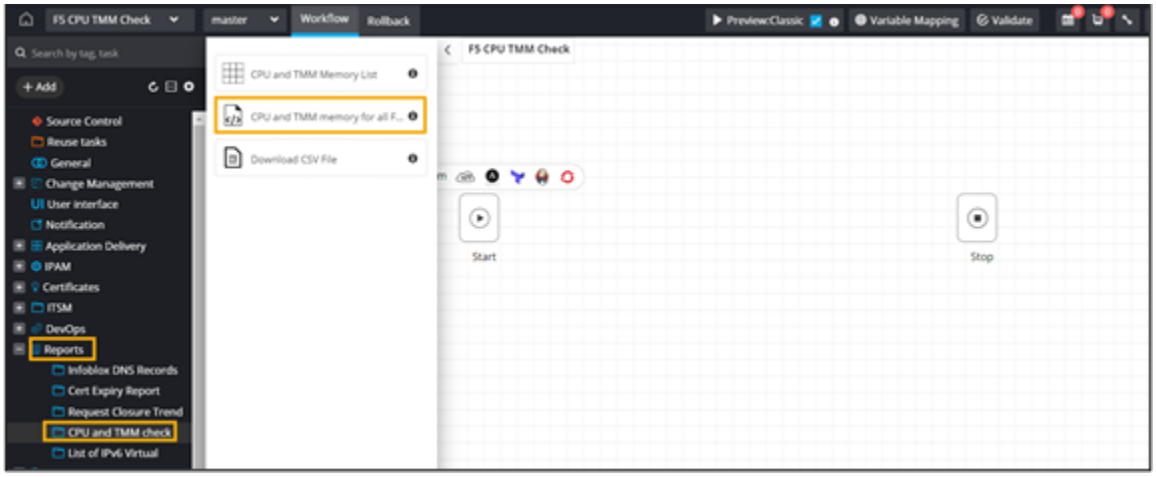
Syntax:

```
output = {"AVX::FileDownload" :{"linkData": <dictionary value> ,"fileName":'<excel file name>', "columnHeaders":<list of column headers to be added>}}
```

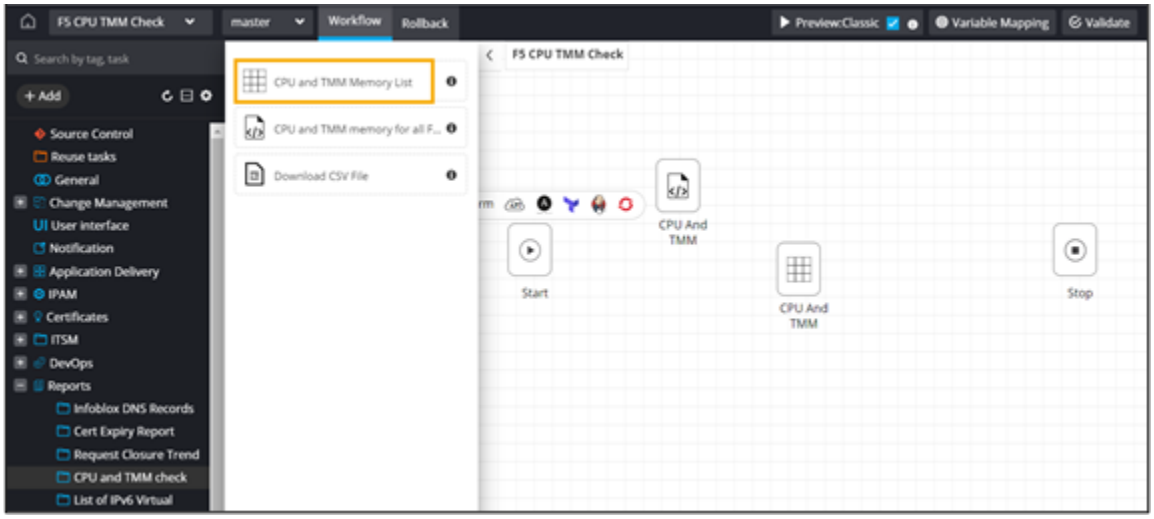
To use this command:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. Design a new workflow.
2. Under **Reports**, from the **CPU and TMM** folder, drag and drop the **CPU and TMM Memory** task.
   This script will scan all the F5 devices managed for the CPU and TMM memory.



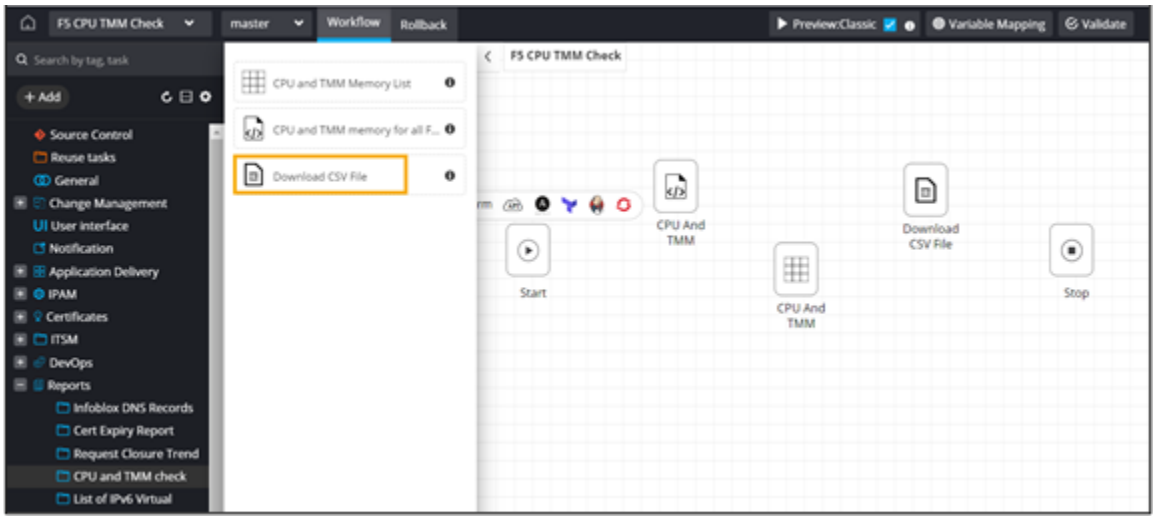
3. Drag and drop the **CPU and TMM memory List** task.
   This task will display the Device name, CPU (%), and TMM (%) in a grid view.



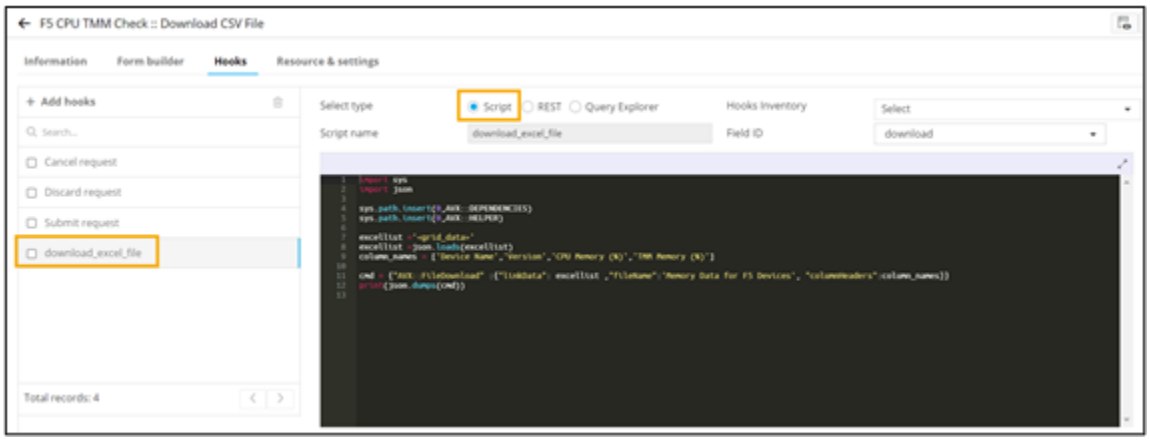
Note: Assign RBAC to users who can access this task.

4. Drag and drop the **Download CSV file** task. This task contains the device and CPU/TMM details from the Grid task in a .csv file.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



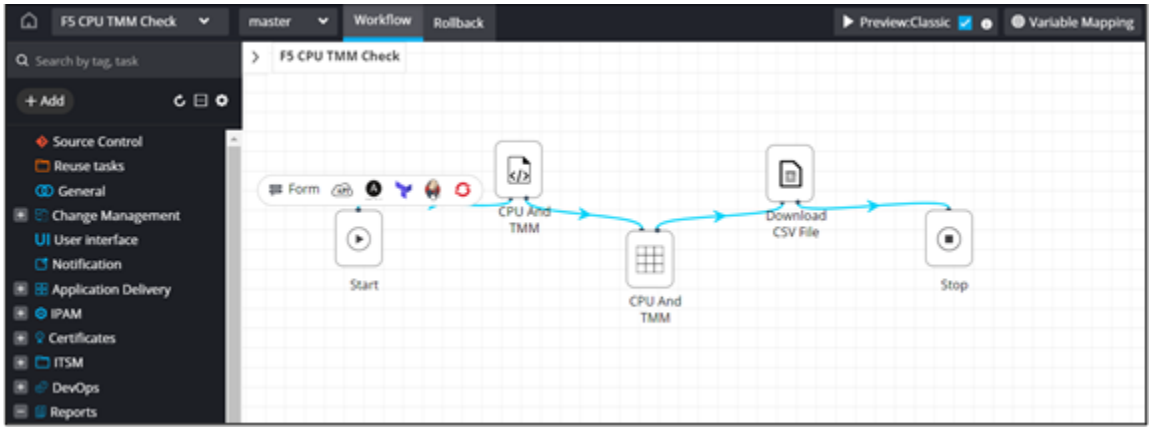
5. Define the file download logic to generate a .csv file on run time.



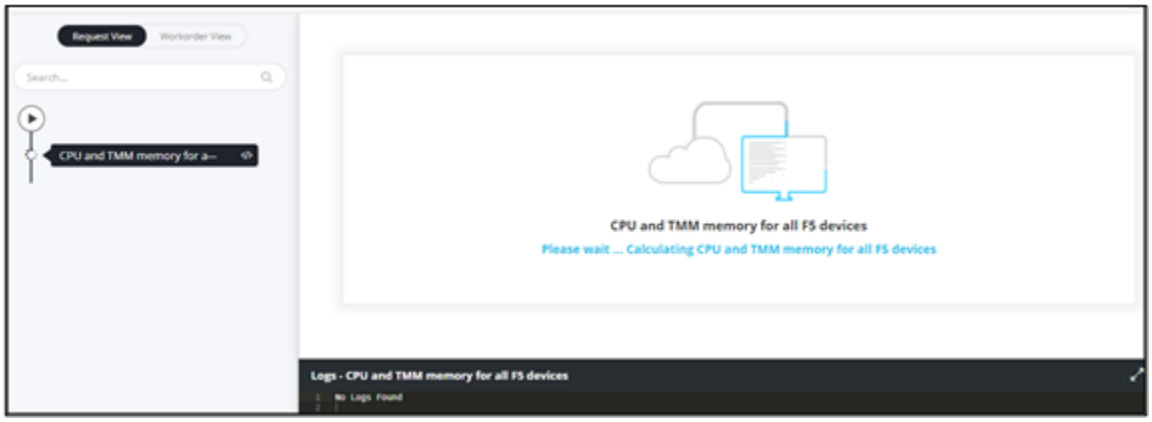

```
import sys

import json


sys.path.insert(0,AVX::DEPENDENCIES)

sys.path.insert(0,AVX::HELPER)


excellist ='<grid_data>'

excellist =json.loads(excellist)

column_names = ['Device Name','Version','CPU Memory (%)','TMM Memory (%)']


cmd = {"AVX::FileDownload" :{"linkData": excellist ,"fileName":'Memory Data for F5 Devices', "columnHeaders":column_names}}

print(json.dumps(cmd))
```

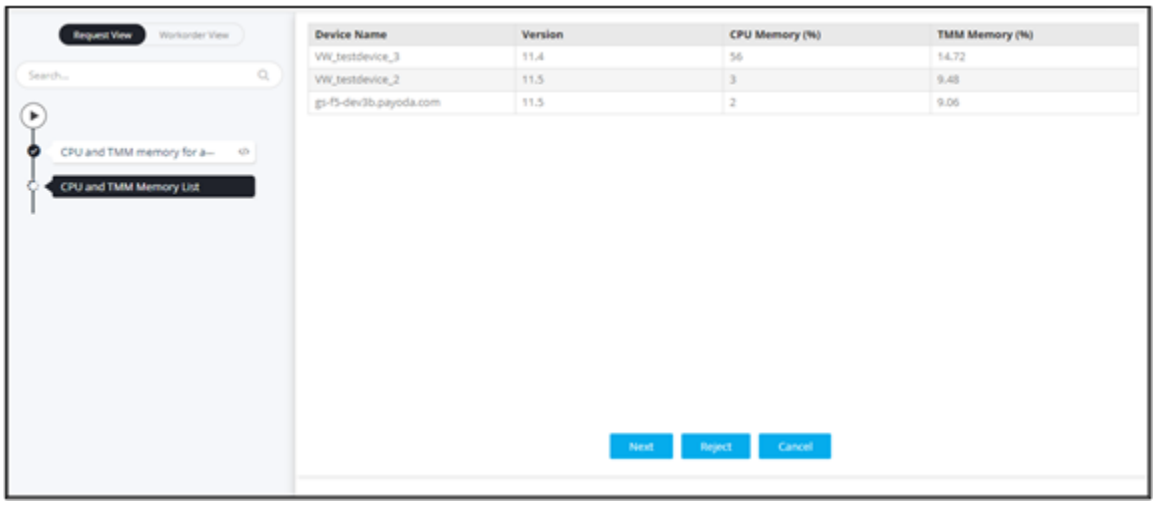
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

6. Connect and enable the workflow.



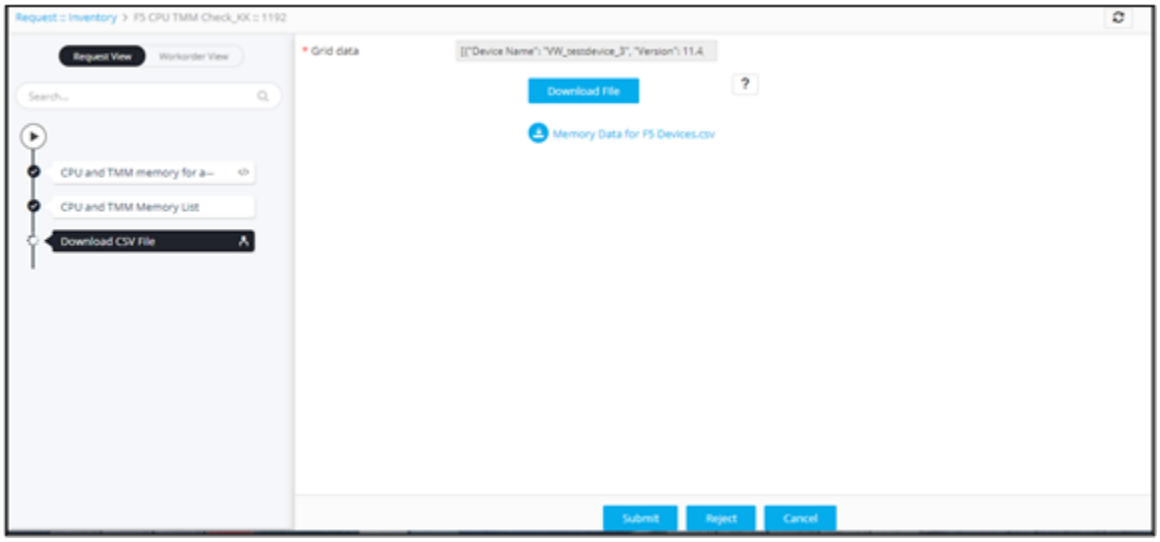
7. Trigger the workflow from the Request :: View/Run page.

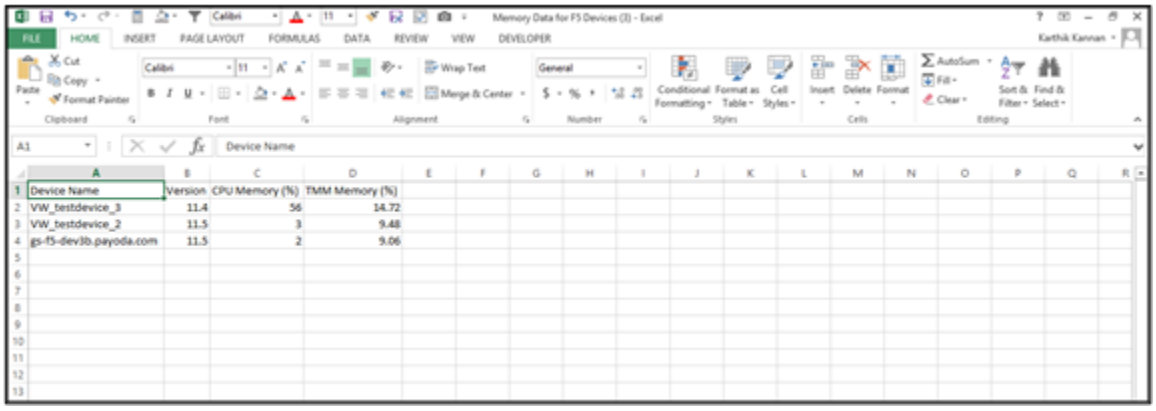




8. Click **Download file** to generate .csv file.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.





AVX::OUTPUT

This task is used within the Script task to record the output of the script execution logic.

- AVX::OUTPUT takes the following arguments - dictionary, a string
- Another usage takes an additional ‘integer’ argument which represents the execution state (1-Success, 2-Failed)

Usage:

```
AVX::OUTPUT(config_dict) / AVX::OUTPUT(error_data, 2)
```

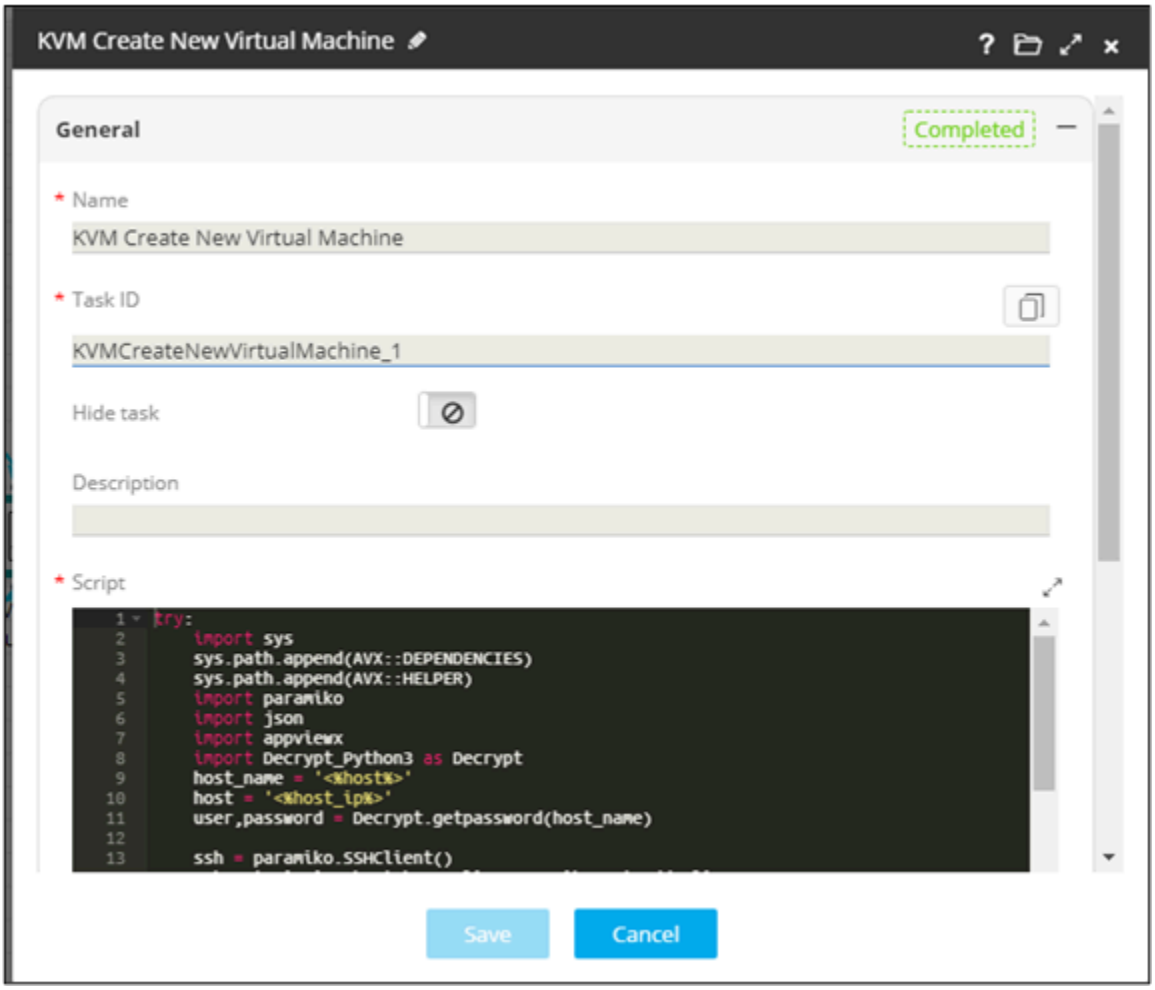
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
mgmt_interface=mgmt_interface,add_interface=add_interface)


    #print(cmd)
    AVX::LOG(cmd)
    stdin_vm, stdout_vm, stderr_vm = ssh.exec_command(cmd)
    outJson = {"mgmt_interface":mgmt_interface,'add_interface':add_interface}
    if (stdout_vm.channel.recv_exit_status() == 0):
        AVX::LOG("Creation of Virtual Machine Completed Successfully")
        AVX::OUTPUT(outJson)


    else:
        AVX::LOG("Disk and VM Creation Failed"+str(stderr_vm.read().decode()))
        AVX::OUTPUT(outJson,2)
    ssh.close()
```

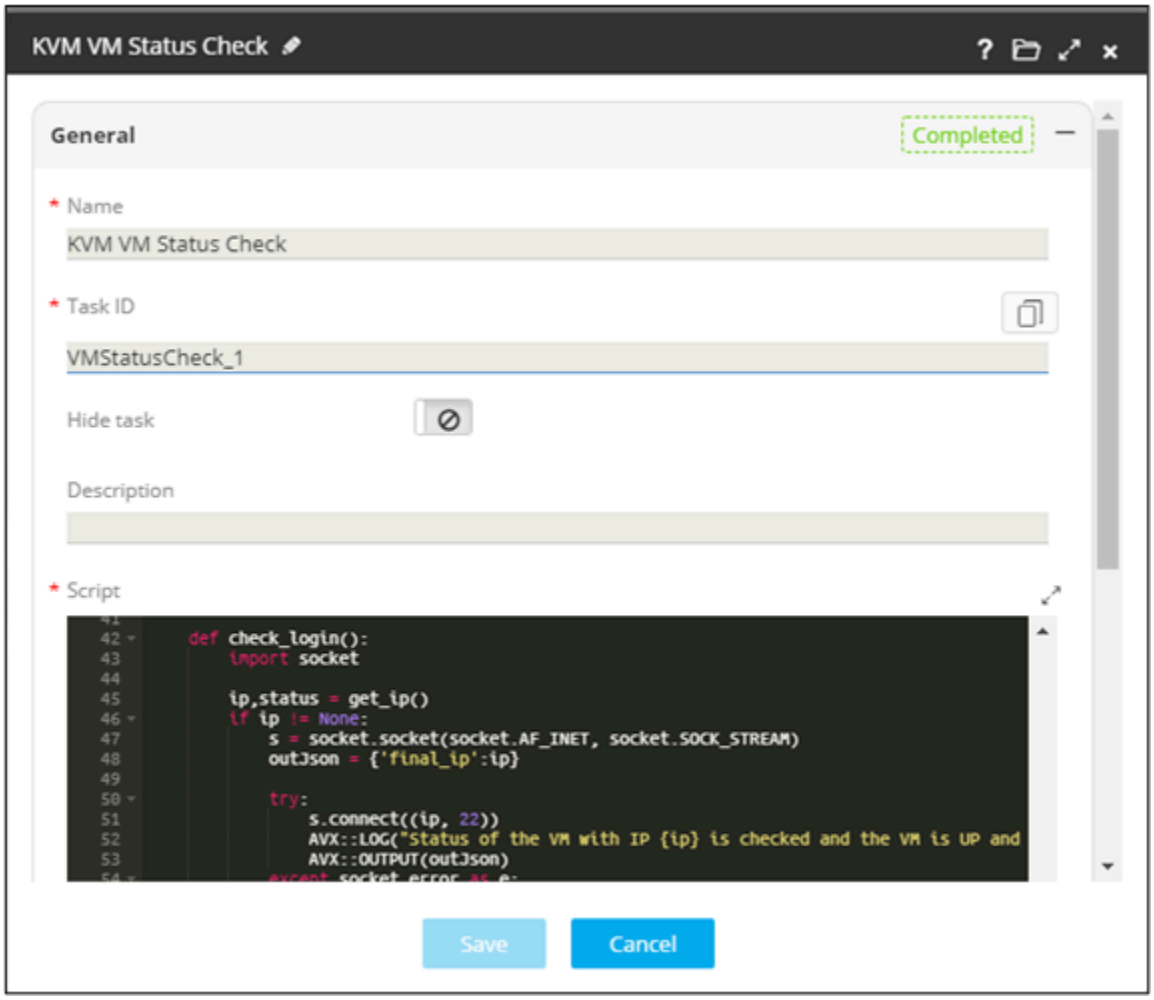
```
if __name__ == '__main__':
    ip,cmd=execute_command()
    outJson = {'ip':ip,'prov_modules':prov_modules,'commands':cmd,'host':host_name,'timezone':time_zone,'vlan':vlan_name}
    AVX::LOG("Device has been provisioned with the information given as the input")
    AVX::OUTPUT(outJson)
```

AVX::OUTPUT usage with failover options:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Script with 'try-except':

```
def check_login():
    import socket
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    outJson={'Success':'Success'}
    try:
        s.connect((ip, 22))
        #print "Port 22 reachable"
        #print json.dumps({"status":"success", "conditions":{ "login_success": "login_success"}})
        # dataJson = {}
        # dataJson["data"] = {"outputData":"Success"}
        # dataJson["status"] = "success"
        # dataJson['state'] = 1
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
    # dataJson["logs"] = [{"message": "Status of the VM with IP {ip} is checked and the VM is UP and RUNNING".format(ip=ip)}]
    # print(json.dumps(dataJson))
    AVX::LOG("Status of the VM with IP {ip} is checked and the VM is UP and RUNNING".format(ip=ip))
    AVX::OUTPUT(outJson)
except socket.error as e:
    #print json.dumps({"status":"failure", "conditions":{ "failure": "failure"}})
    # dataJson = {}
    # dataJson["data"] = {"outputData":"Failure"}
    # dataJson["status"] = "failure"
    # dataJson['state'] = 2
    # dataJson["logs"] = [{"message": "Status of the VM with IP {ip} is checked and the VM is still not UP".format(ip=ip)}]
    # print(json.dumps(dataJson))
    AVX::LOG( "Status of the VM with IP {ip} is checked and the VM is still not UP".format(ip=ip))
    AVX::OUTPUT(outJson,2)
s.close()
```

Note: In case of using ‘try-except’ in the python script, you must include ‘AVX::OUTPUT with state 2’ to record script failure actions. Try-except is used to handle errors and exceptions.

AVX::MERGECONFIG

This command is used to merge multiple configurations in a workflow to be generated as a single entity. The parameters must have the config object generated using the AVX custom commands only. It is useful in cases where a few commands have to be added at runtime prior to implementation depending on the outcome of certain prevalidation commands.

- Initial Implementation config generated for device version v11 scheduled for 10th March.
- Run pre-validation on 9th March to check for device versions.
- If there is a change in device version, generate and add additional commands, and merge Implementation configuration into one.

Usage:

```
Conf1 = <%Script_1.output %>
Conf2 = <%Script_3.output %>
Conf3 = <%Script_4.output %>
AVX::OUTPUT (AVS::MERGECONFIG(config, [conf1, conf2, conf3]))
```

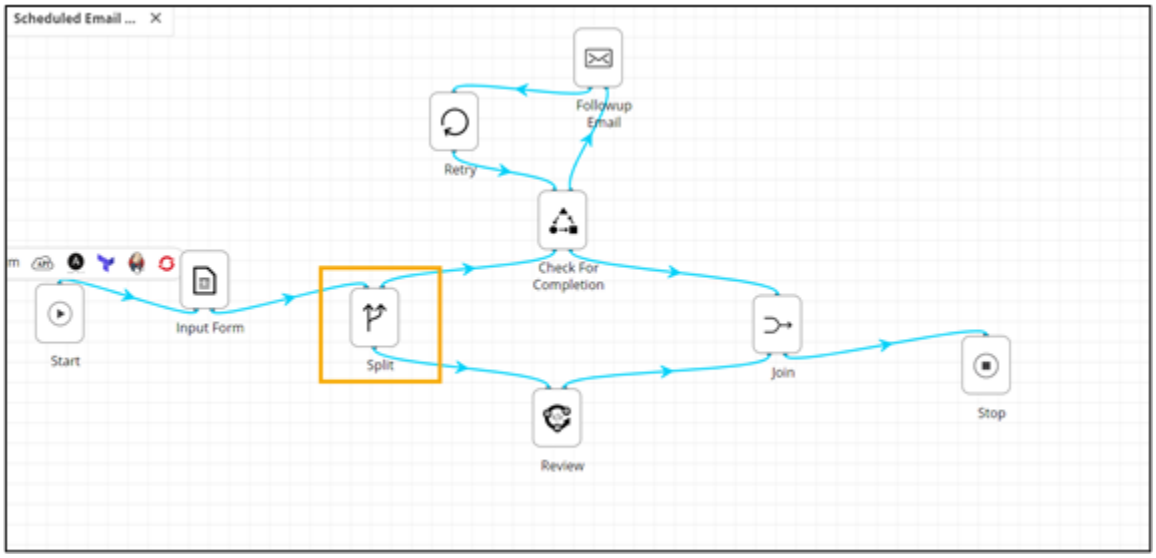
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

OR

```
Merged_config = AVX::MERGECONFIG(config,[conf1, conf2, conf3]))
```

Split

Split task is used when parallel flows have to be spawned as part of a parent workflow.



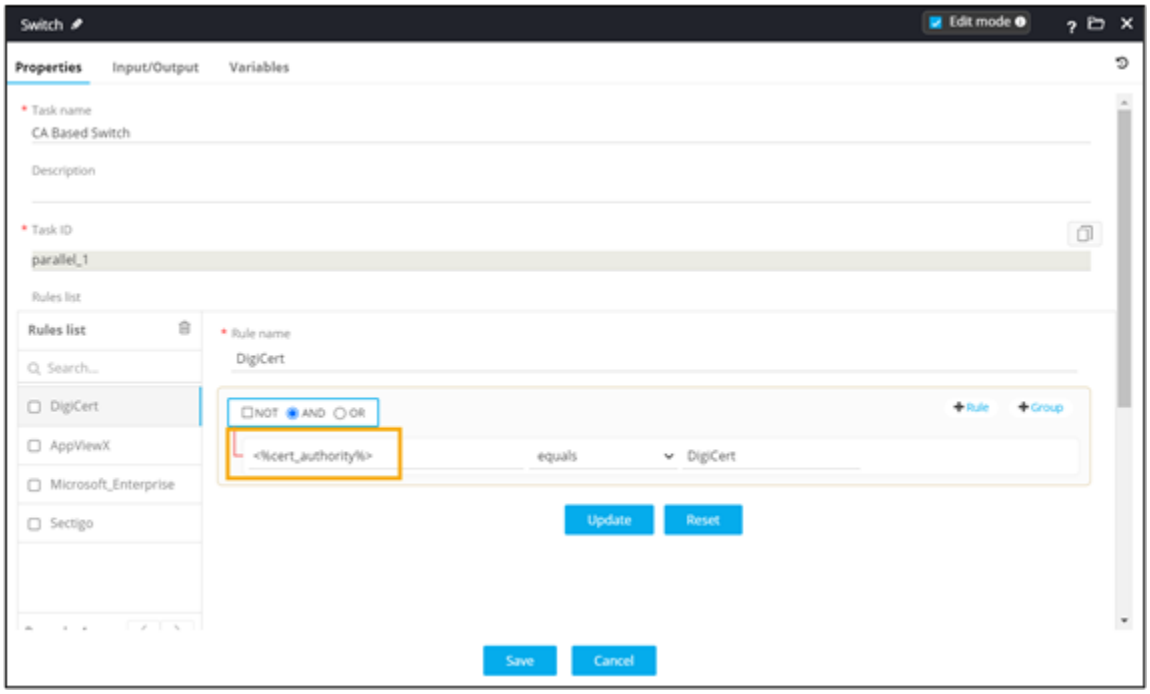
Note: The Join task must be used as a common connector when a Split task is used.

Switch

The Switch task allows users to configure multiple conditional rules in order to take decisions as part of the workflow automation process.

- Provision to define rules with multiple decisions based on which workflow can be routed.
- Provision to reference variables from a previous task.
- Provision to define multi-boolean conditional rule logic as a truth table.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



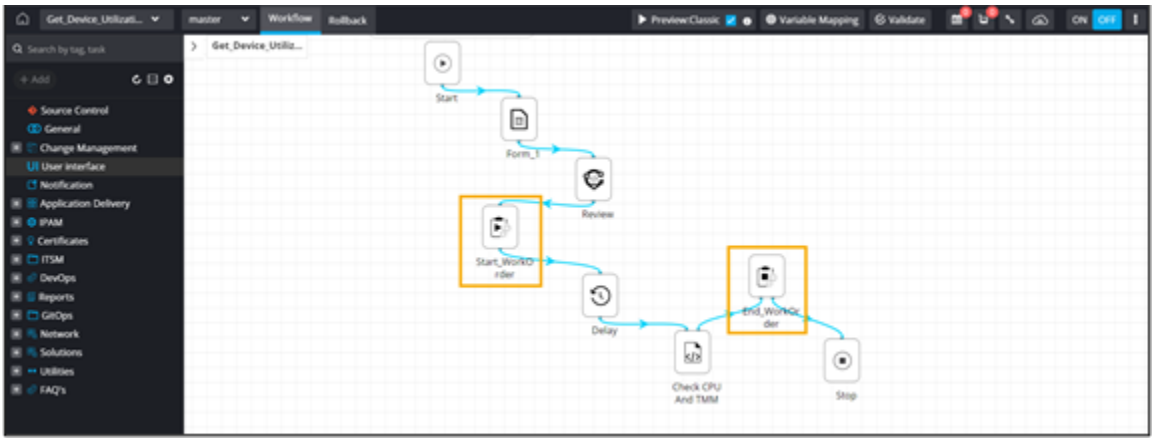
WorkOrder

This task allows you to generate a work order (aka Child request) under a main workflow request.

- Work order is an optional task
- Work order is akin to a mini workflow under a parent workflow
- Work order can be used to mimic the existing APS (Application Provisioning System) capability where different configurations can be tracked as part of a work order under a single workflow request (For example, Create VIP, Create DNS, and New CSR)
- On workflow execution, the work order task renders the work order review page for the user
- Each work order triggers a unique work order ID that can be used for audit
- A work order can have a specific Rollback option defined and associated

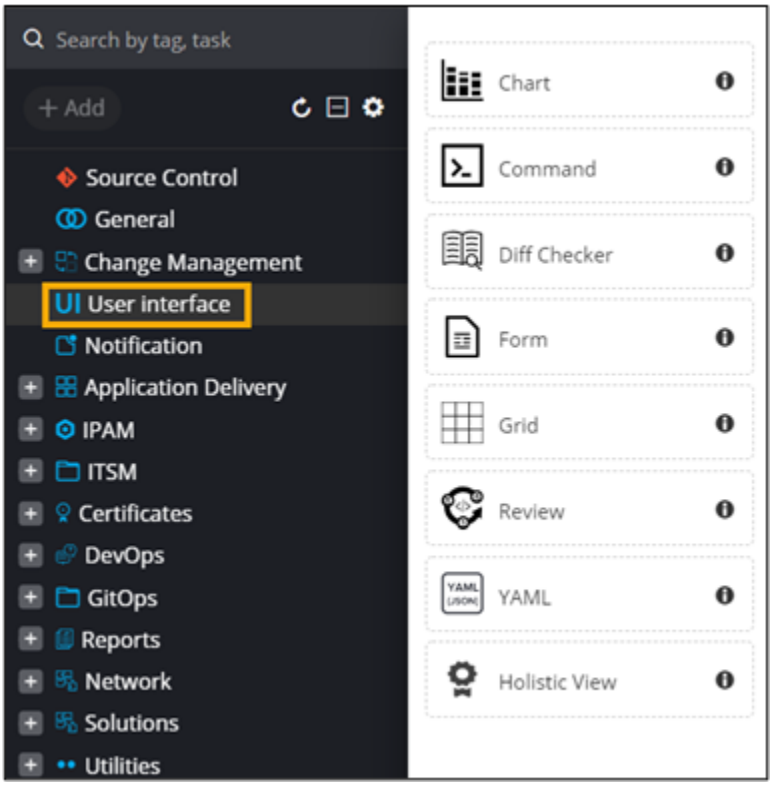
For example, you are executing two implementations, one for a firewall device and the other for a certificate. Now if we just want to rollback the firewall because of some issue in its implementation, then a work order can be used in this situation.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Task Category - User Interface

This category comprises tasks that require a certain amount of user intervention.



- Chart
- Command
- Diff Checker

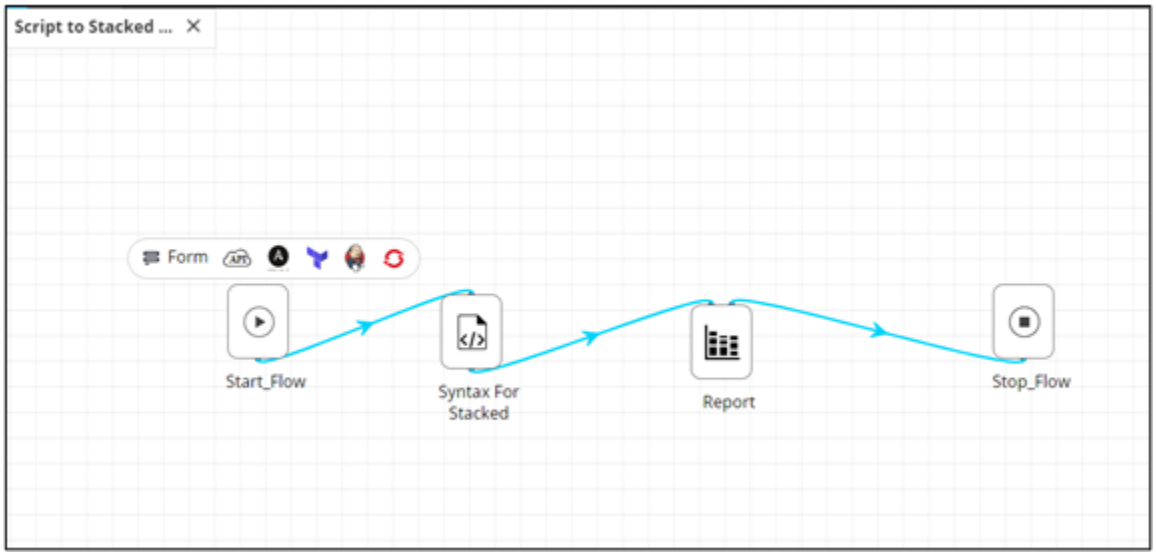
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Form
- Grid
- Review
- YAML

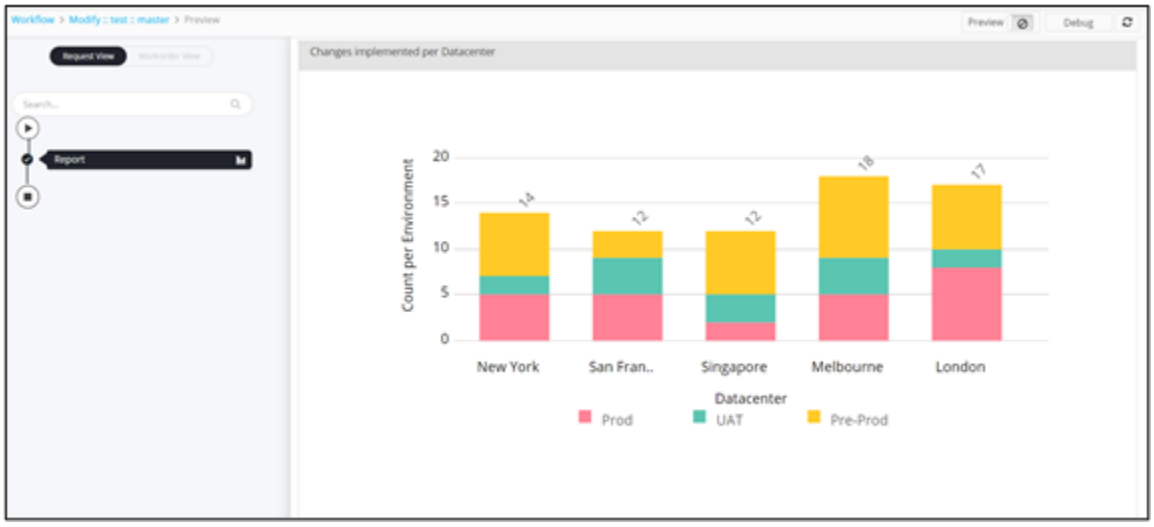
Chart

Chart Task allows for designing and visualizing data in the form of pie charts, bar charts or stacked bar charts.

- Provision to represent data using pie, bar and stacked charts
- Provision to define charts manually or by passing data from a previous source (such as script, form) using global variables
- Provision to define custom data set for the chart - sequence number, labels and values
- Provision to dynamically configure the chart elements by passing the global variable defined in the script in the “source” field
- Provision to assign user role (RBAC) access to the chart task in the request stage
- Provision to preview chart types with sample data



Copyright © 2020 AppViewX, Inc. All Rights Reserved.



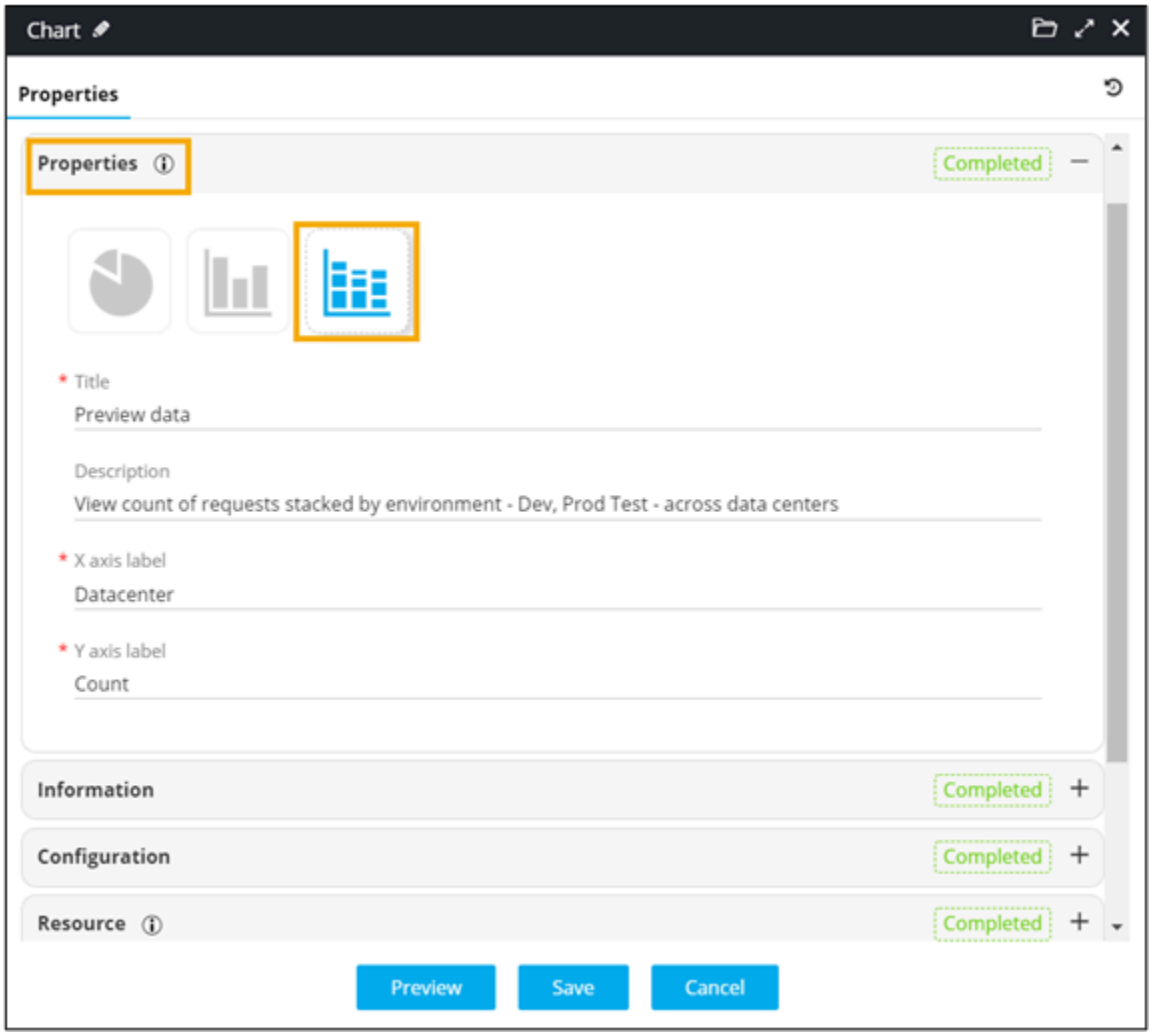
- Designing a Chart with Static Data
- Designing a Chart with Dynamic Values

Designing a Chart with Static Data

To design a stacked chart using static data,

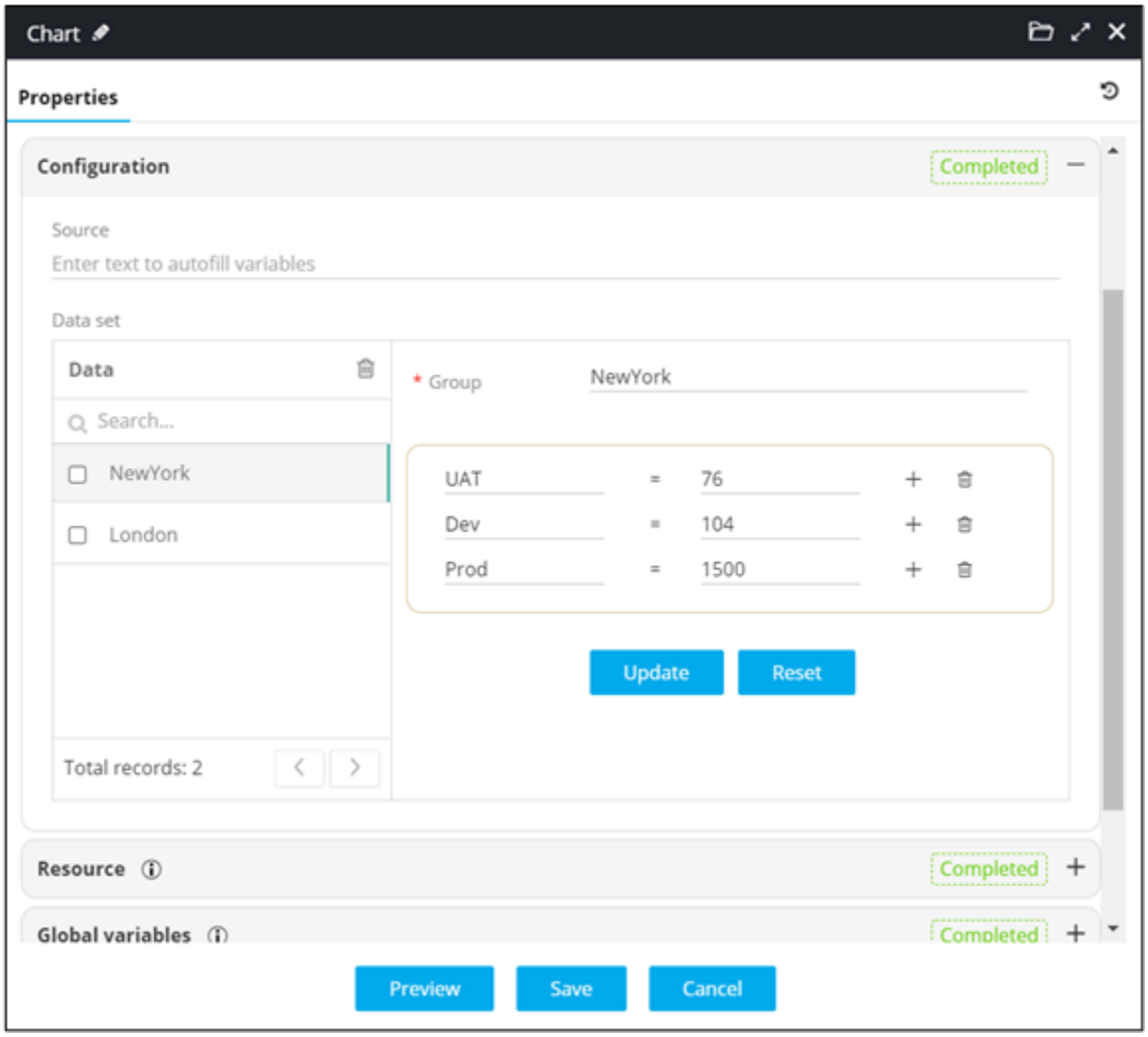
1. Design a workflow.
2. From the **User Interface** section, drag and drop the **Chart** task.
3. In the **Chart** task window, under **Properties**, select the chart type as stacked chart .
4. In the **Chart** task window, under **Properties**, enter values for chart **Title** and X-axis, Y-axis labels.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



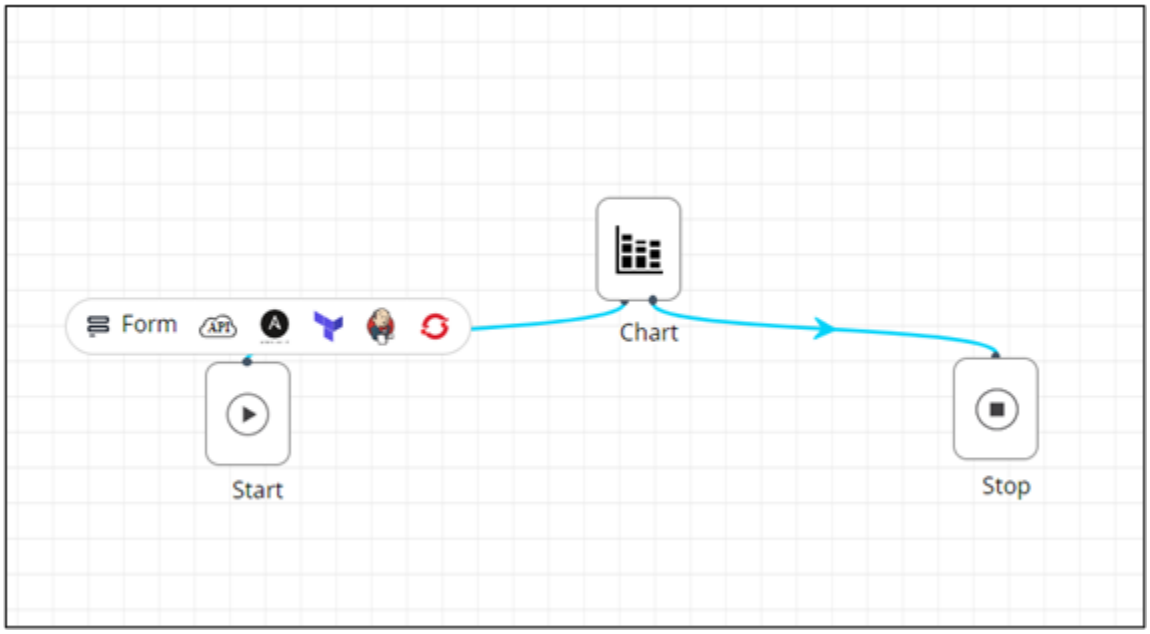
5. In the **Chart** task window, under **Configuration**, define the static data to stack values in the chart.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

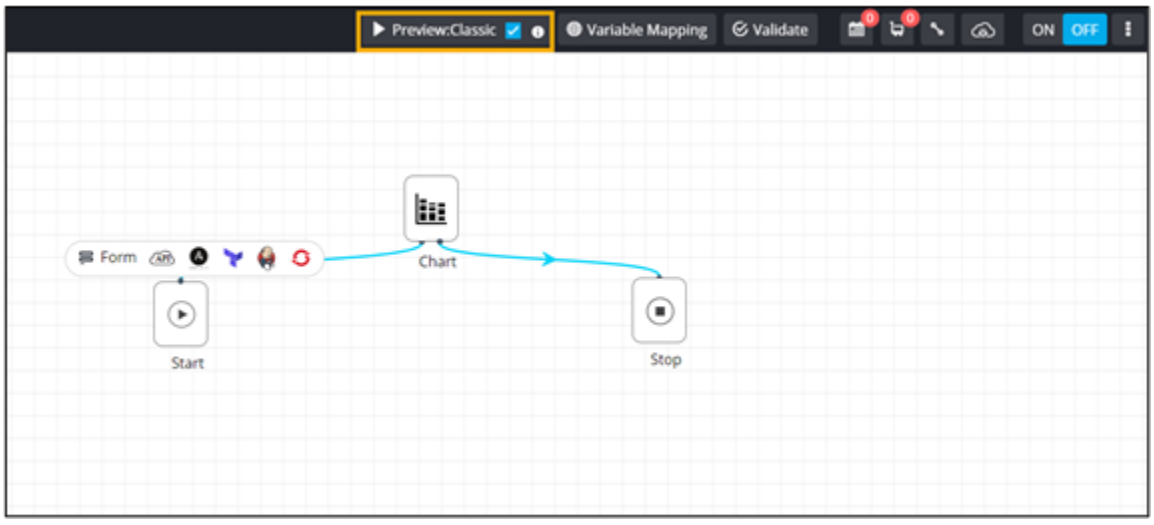


6. Click **Save**.
7. Connect the workflow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



8. Click **Preview**.



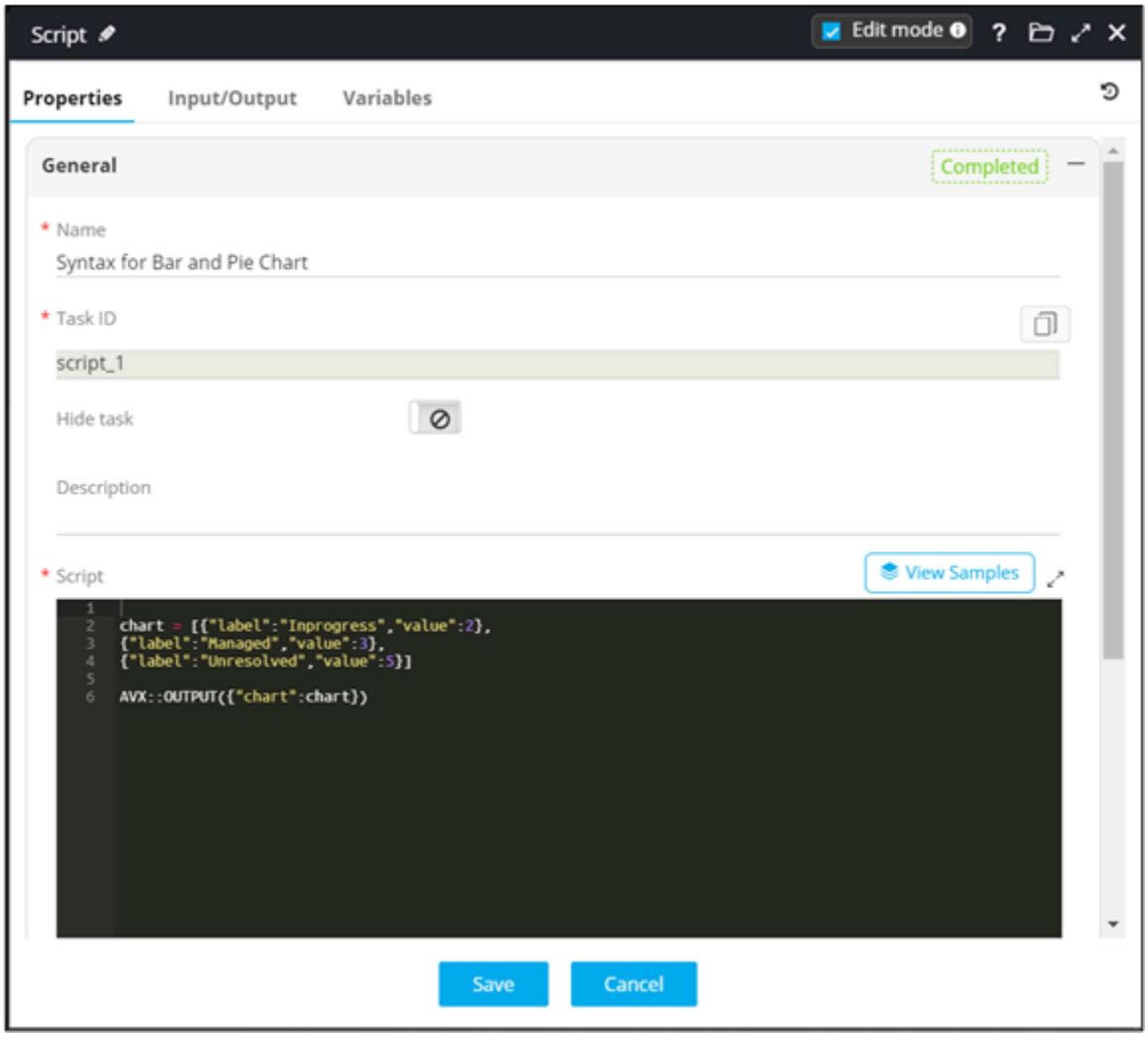
Stacked chart with static data is displayed.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Designing a Chart with Dynamic Values

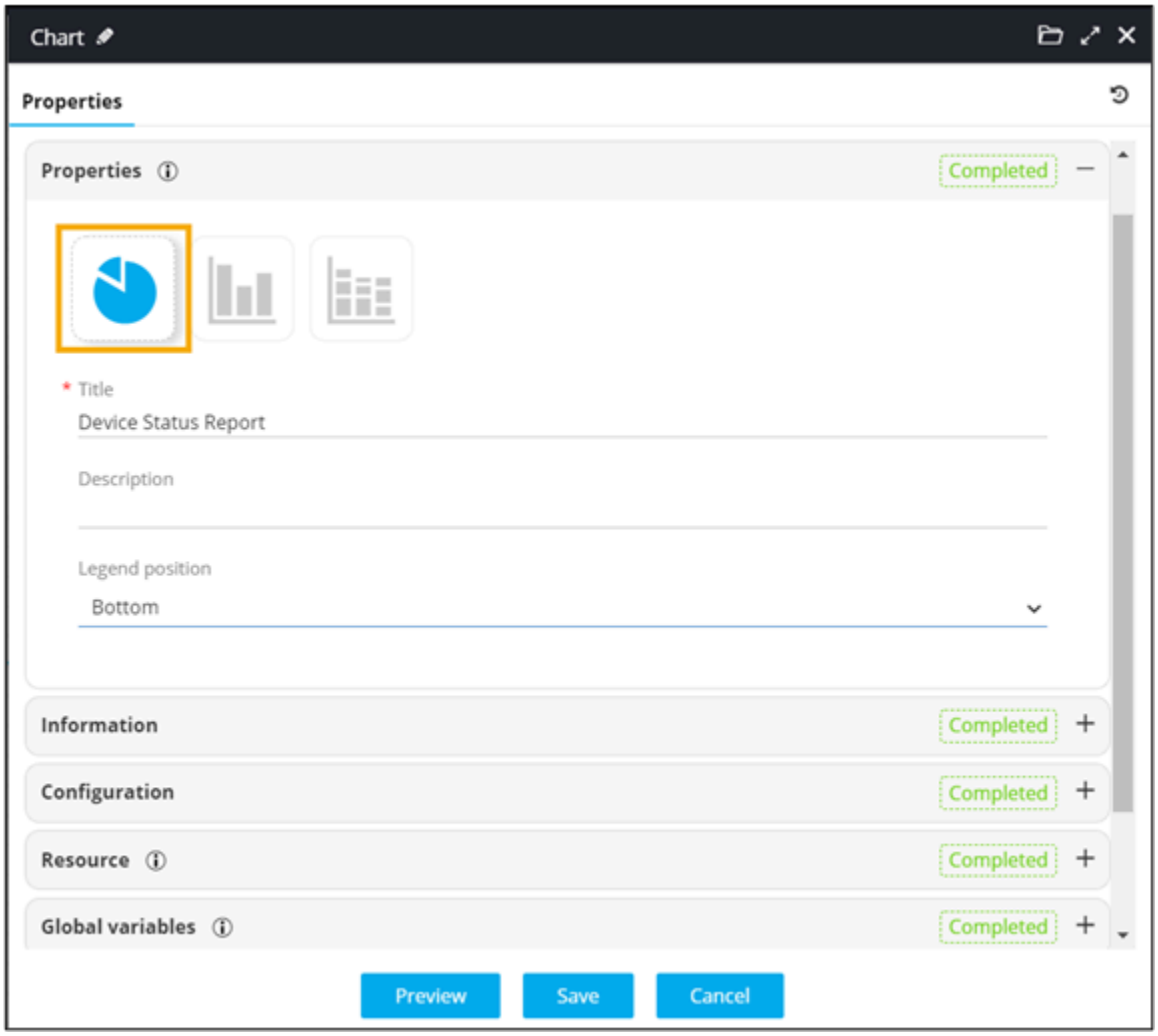
1. Design a workflow.
2. From the **General** section, drag and drop a **Script** task.
3. In the **Script** task window, under **Properties**,define a **Script** to get a list of values for the chart.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



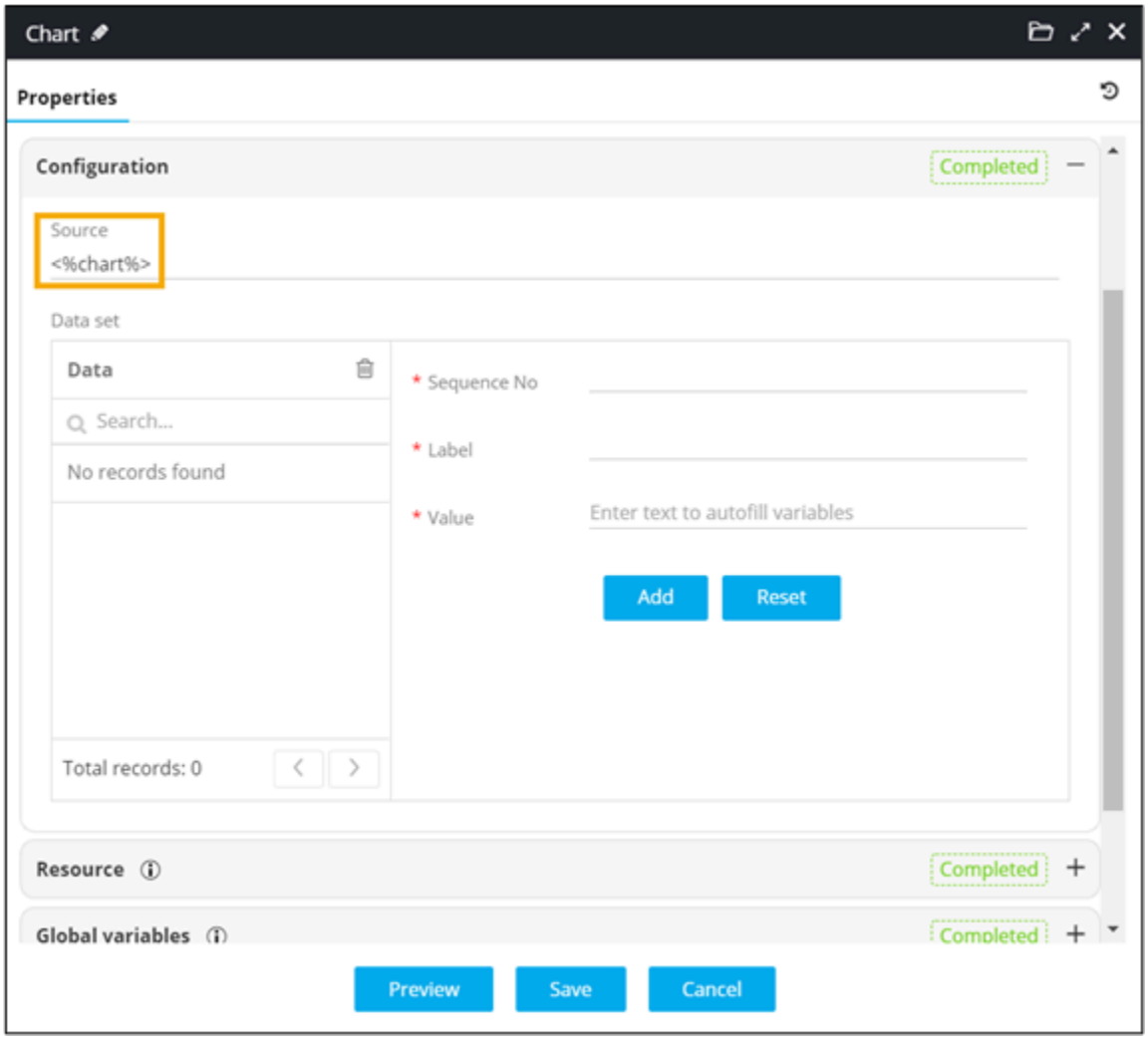
4. From the **User Interface** section, drag and drop a **Chart** task.
5. In the **Chart** task window, under **Properties**, select the chart type as Pie .
6. In the **Chart** task window, under **Properties**, enter or select the field information.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



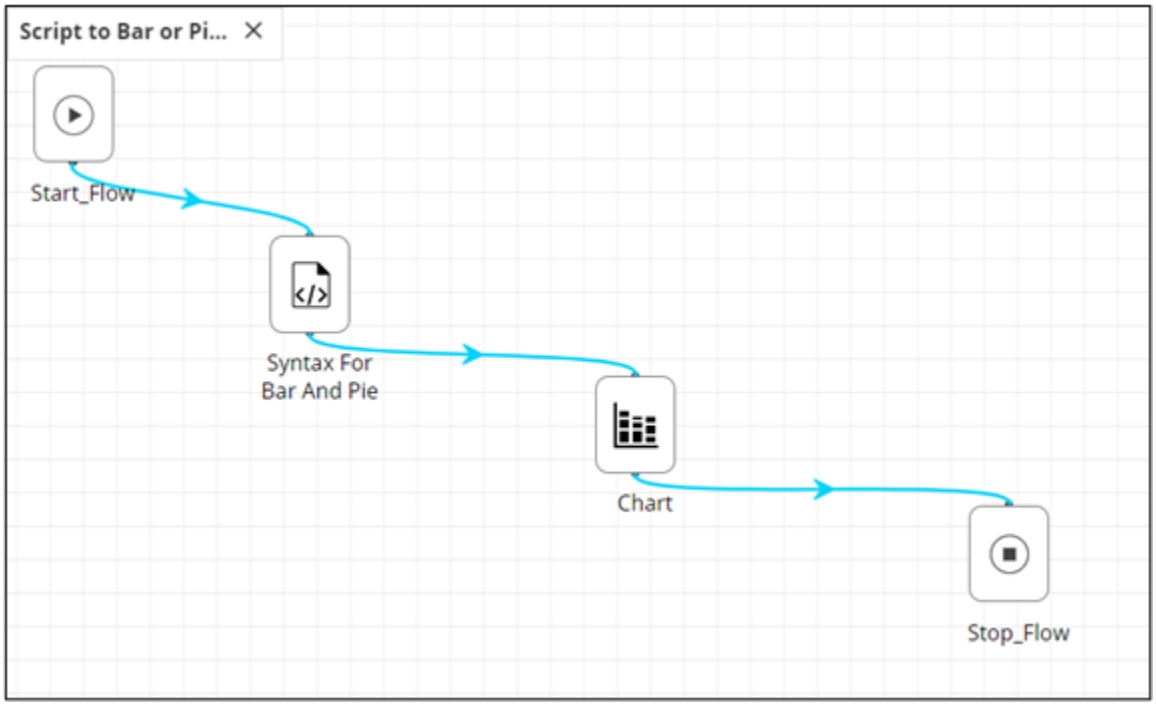
7. In the **Chart** task window, under **Configuration**, refer the global variable to render data.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



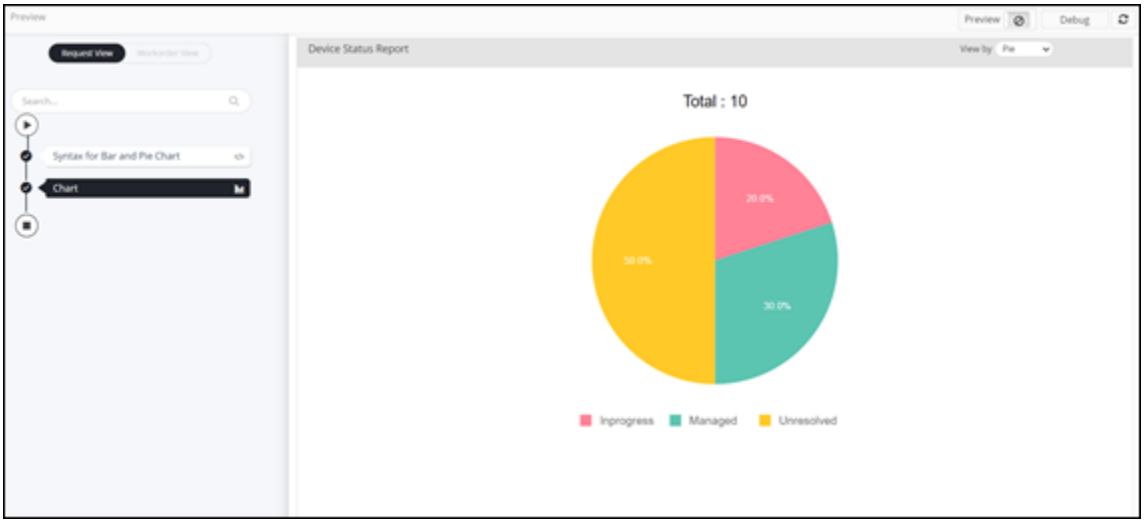
8. Connect the workflow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



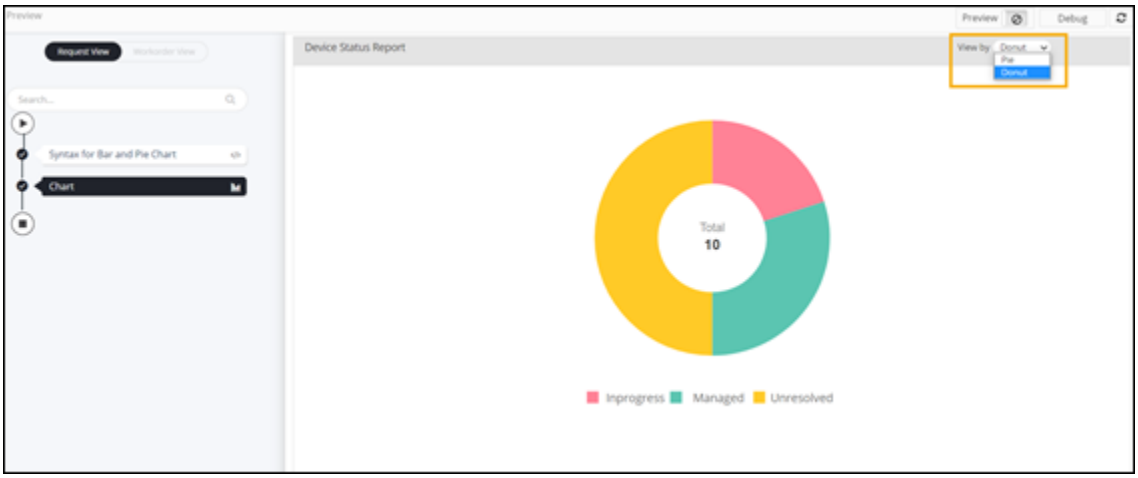
9. Click **Preview**.

   Chart is displayed.



10. Under **View by**, to view the data as a Donut, select **Donut**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Command

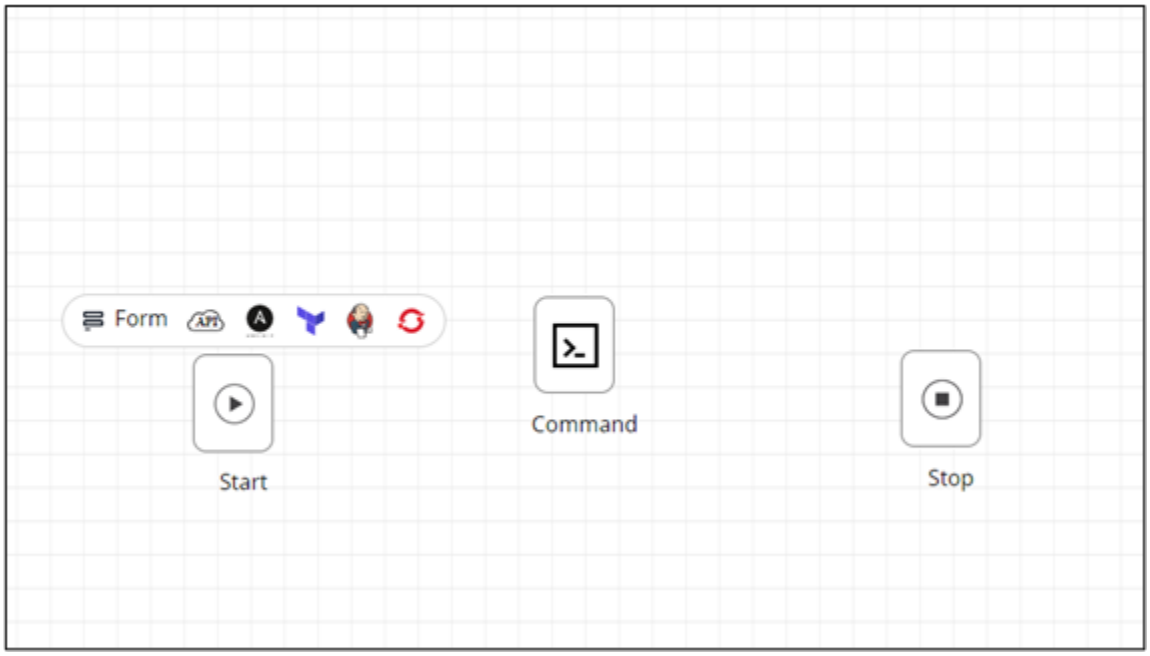
The Command task allows you to execute command configurations. You can either define a YAML structure or use prebuilt Command samples. The Command task simplifies config automation using SSH and CLI.

- YAML - declarative command execution
- Parallel, sequential config execution
- Intelligent command execution with in-built rules (Multi-boolean conditions operators with shortcuts)
- Event handling and dynamic variable support
- Modify configs and reviews
- Pre-validation and post-validation of configs
- Automate config automation for ADC devices, routers, and switches

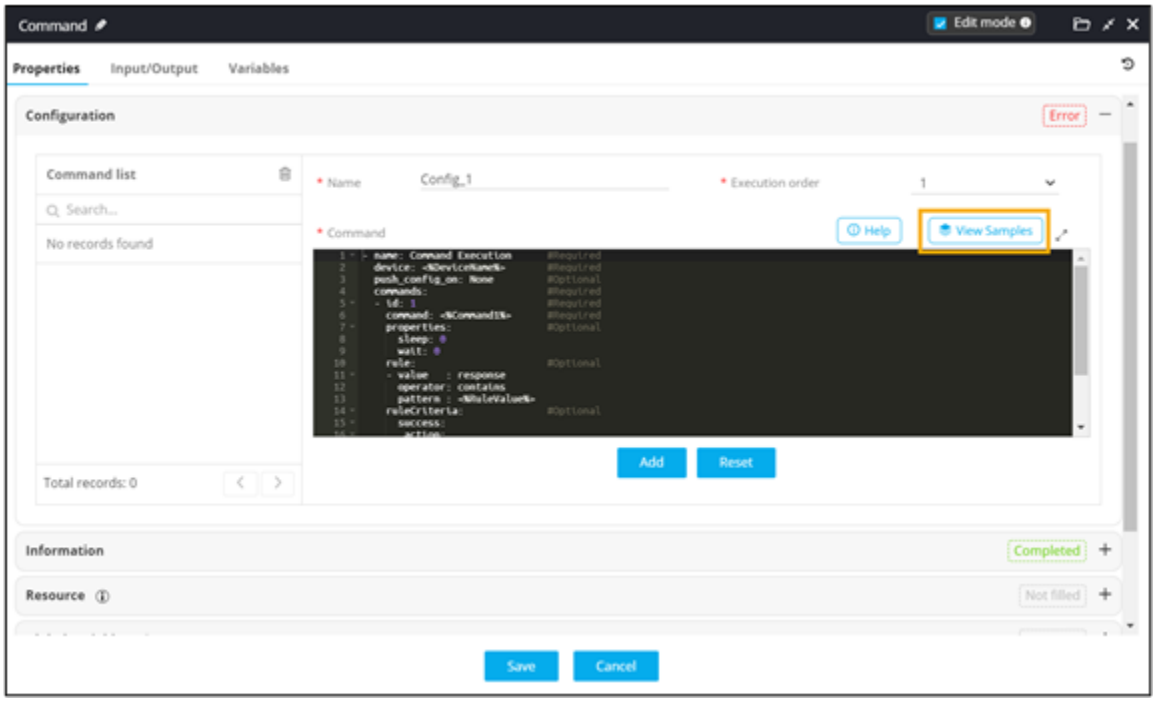
To use a Command task within your workflow:

1. Design a workflow.
2. From the **User Interface** section, drag and drop the **Command** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

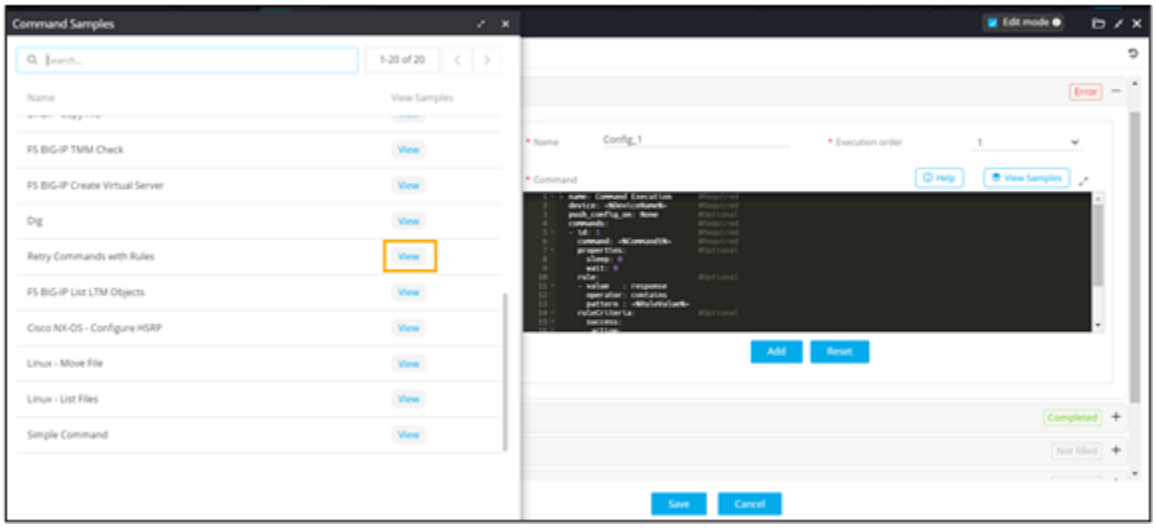


3. To use a prebuilt command sample, in the **Command** task window, under **Properties**, in the **Configuration** section, click **View Samples**.

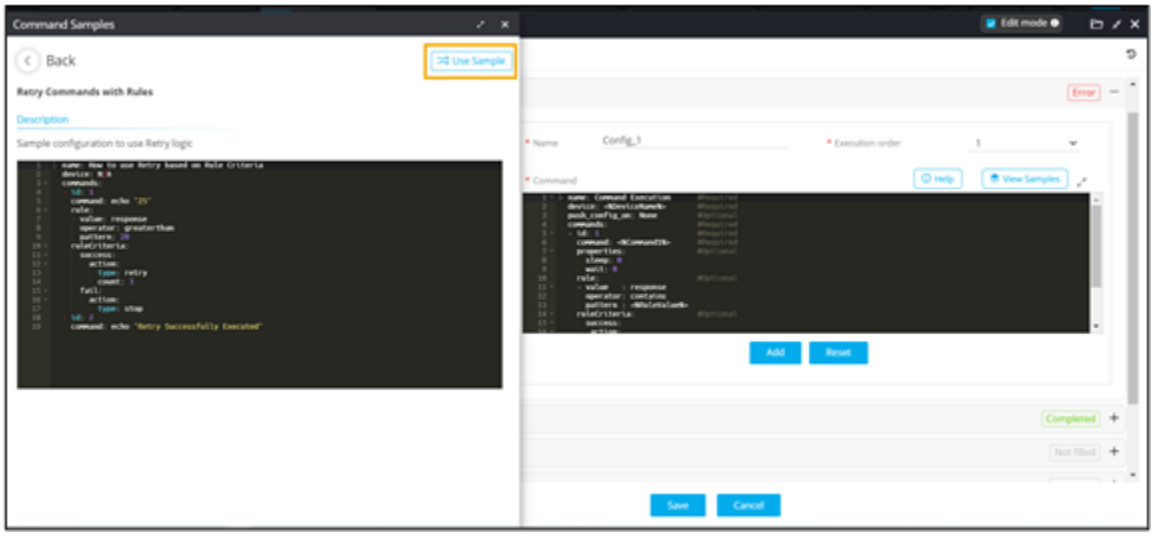


4. In the **Command Samples** window, click **View** next to the Command Sample.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

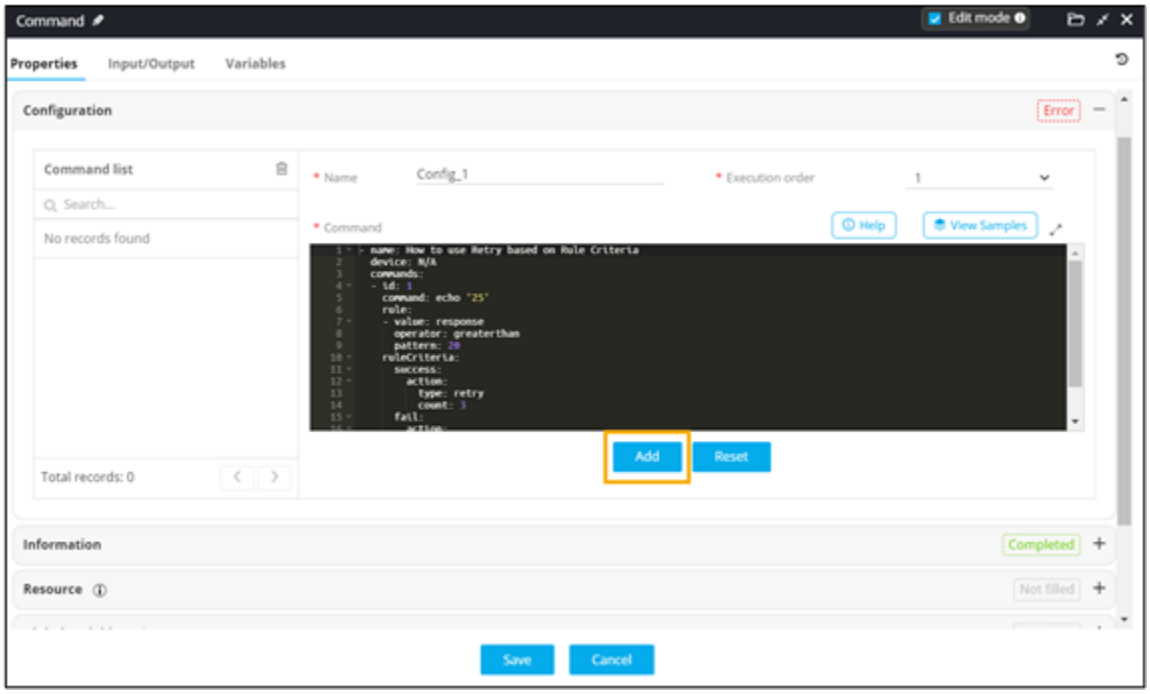


5. To use this configuration, click **Use Sample**.

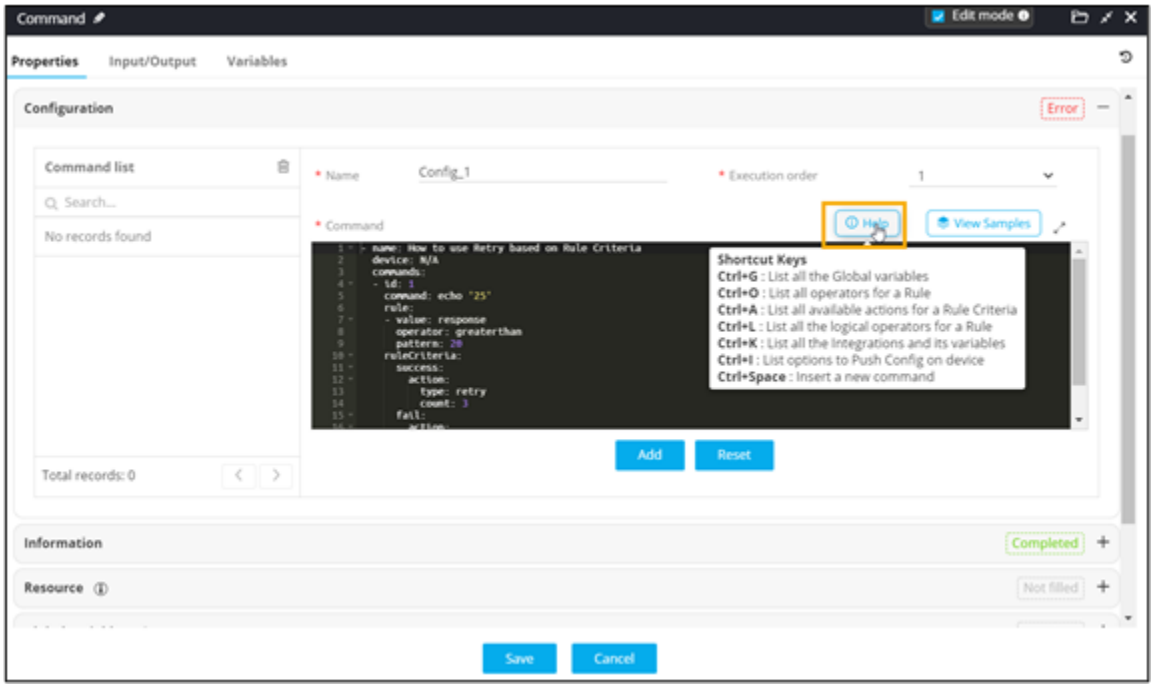


6. To save the configuration settings, in the **Command** task window, under **Properties**, in the **Configuration** section,click **Add**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

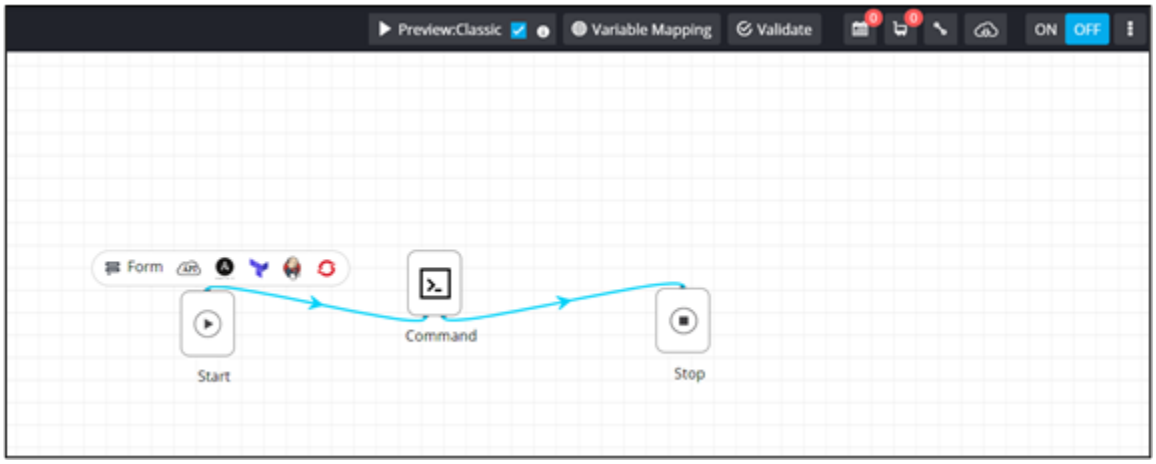


Tip: To see a list of keyboard shortcuts, hover your mouse over **Help**.

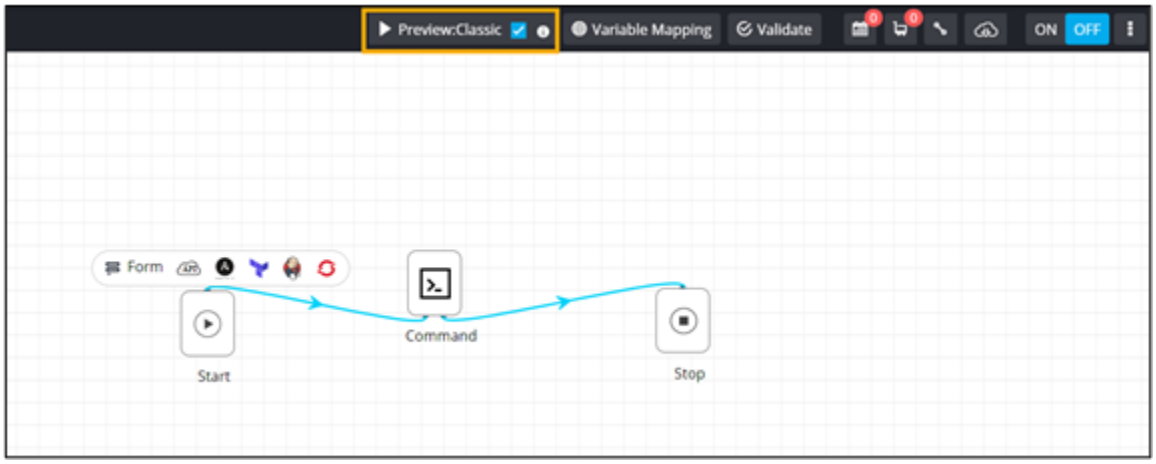


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

7. Click **Save**.
8. Connect the workflow.

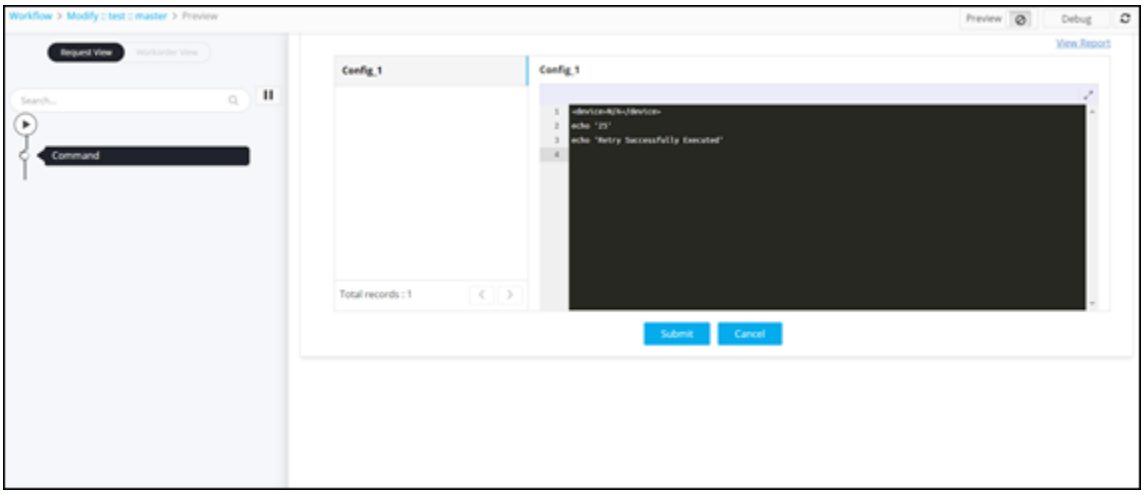


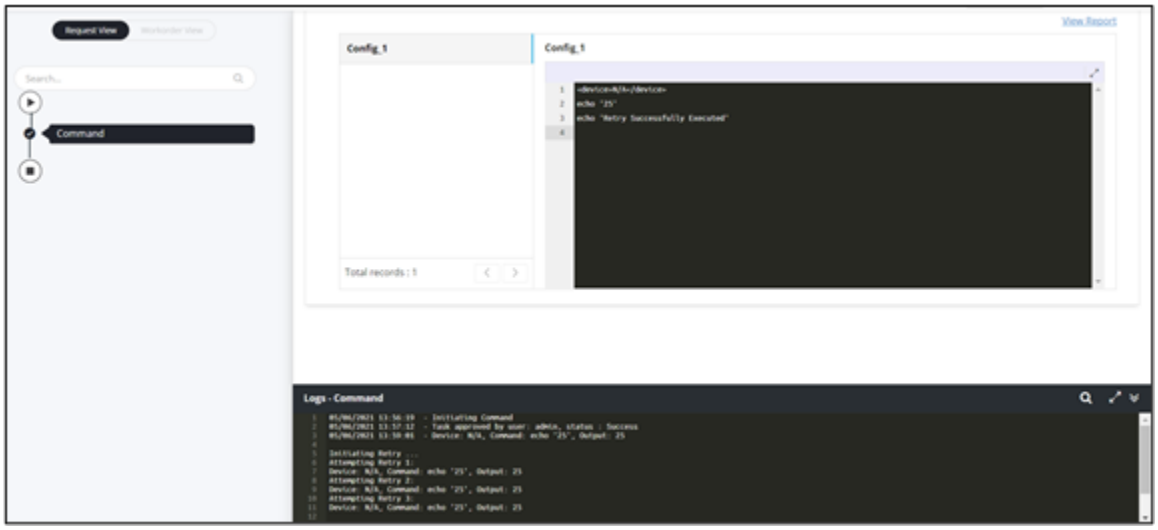
9. Click **Preview**.



Command task is executed.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.





- Using Command task to pass the task response as a Global Variable

Using Command task to pass the task response as a Global Variable

The Command task can be used to pass the response of a task as a global variable for the next task in the workflow. The global variable can be declared in the following ways:

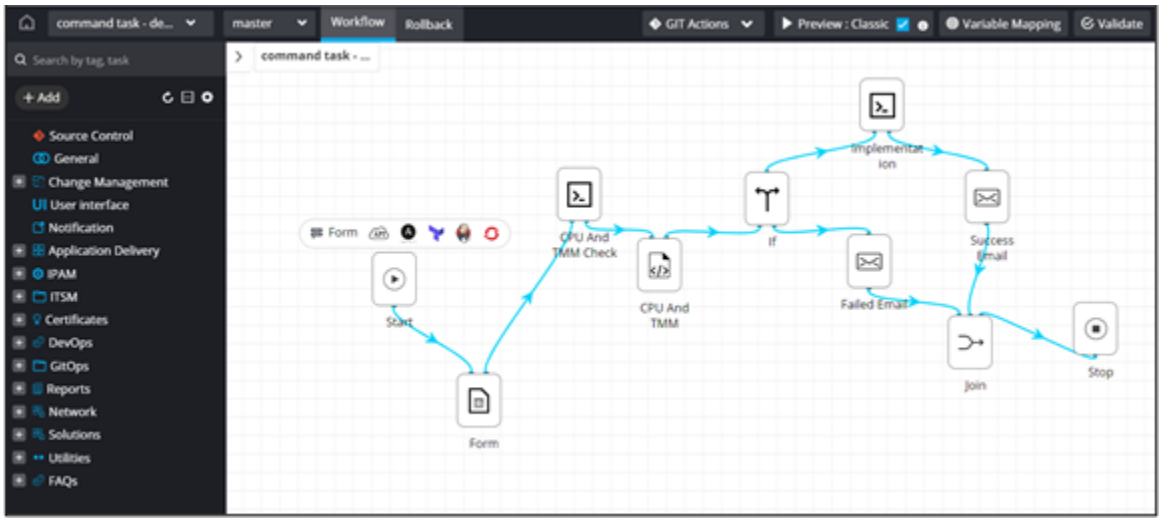
- `$$<Config Name>$$` - This returns the value of all the Commands with the Device Name and Response.
- `$$<Config Name>.<Command ID>$$` - This returns the value of the specific Command with the Device Name and Response.
- `$$<Config Name>.<Command ID>.output$$` - This returns the response of the Command ID.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

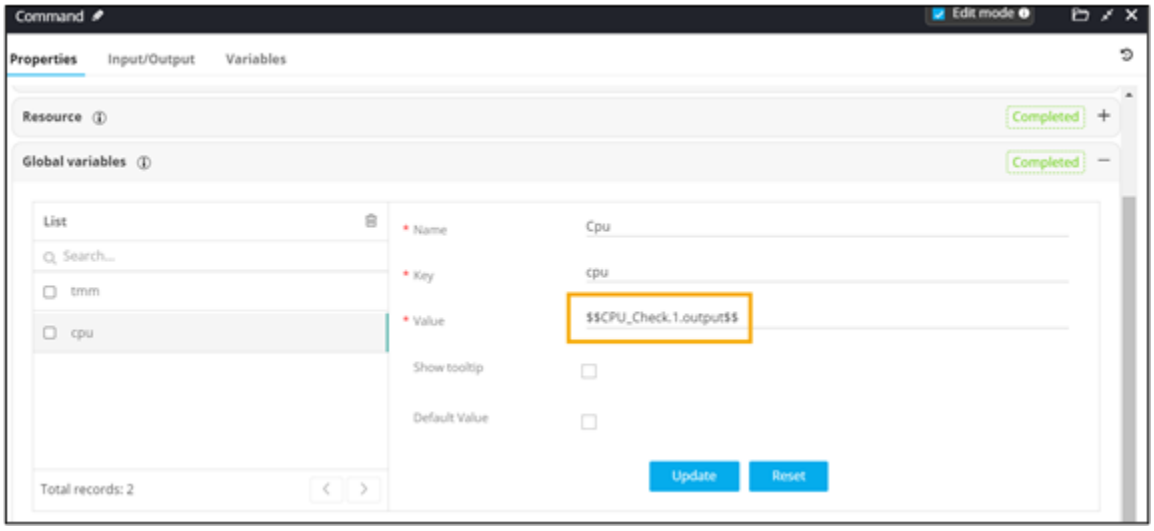
- `$$<Config Name>.<Command ID>.command$$` - This returns the value of the command of the provided Command ID.
- `$$<Config Name>.<Command ID>.deviceName$$` - This returns the Device Name of the provided Command ID.

To use the output from the Command task to perform a CPU/TMM check and create a virtual LTM server on a F5 BIG-IP device:

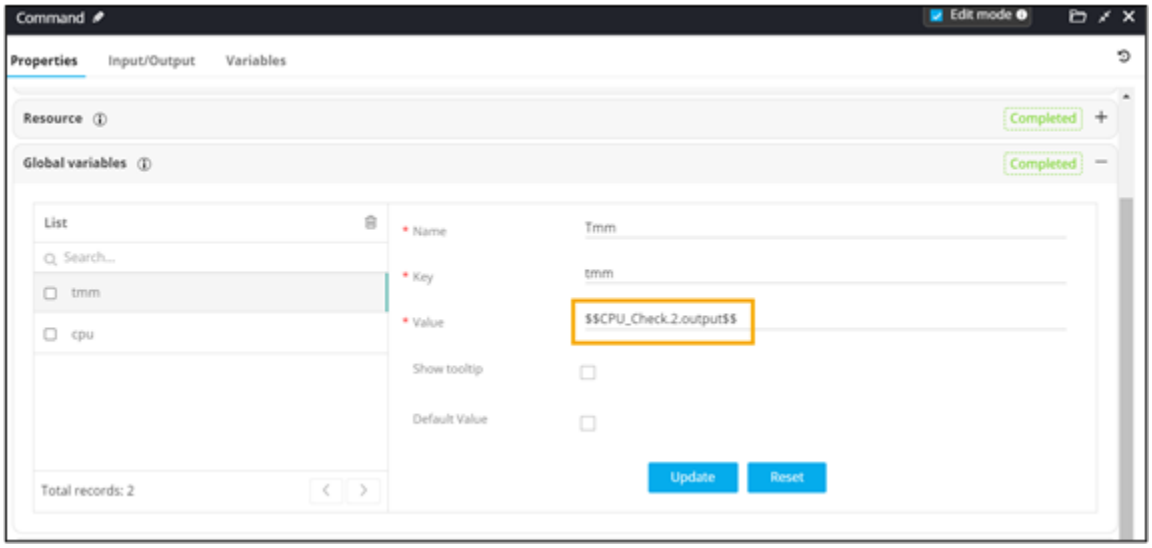
1. Design a workflow.



2. To pass the output of the Command task as a global variable, in the **Command** task window, under **Properties**, in the **Global variables** section, define the value to be declared as a global variables using the following syntax: `$$Cpu.1.output$$` `$$Cpu.2.output$$`



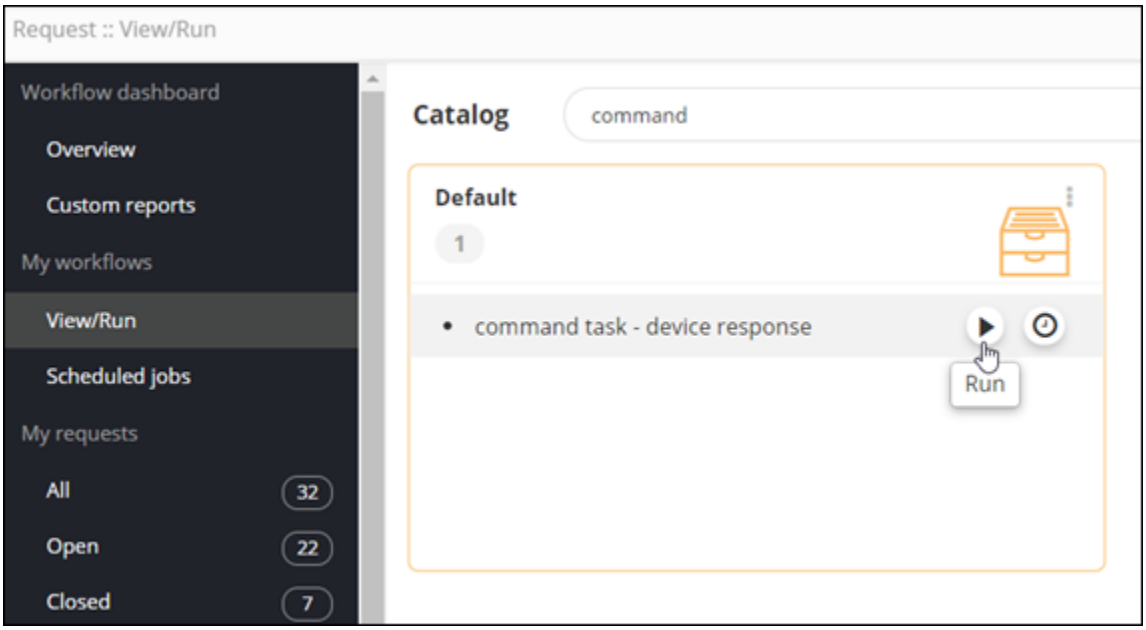
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



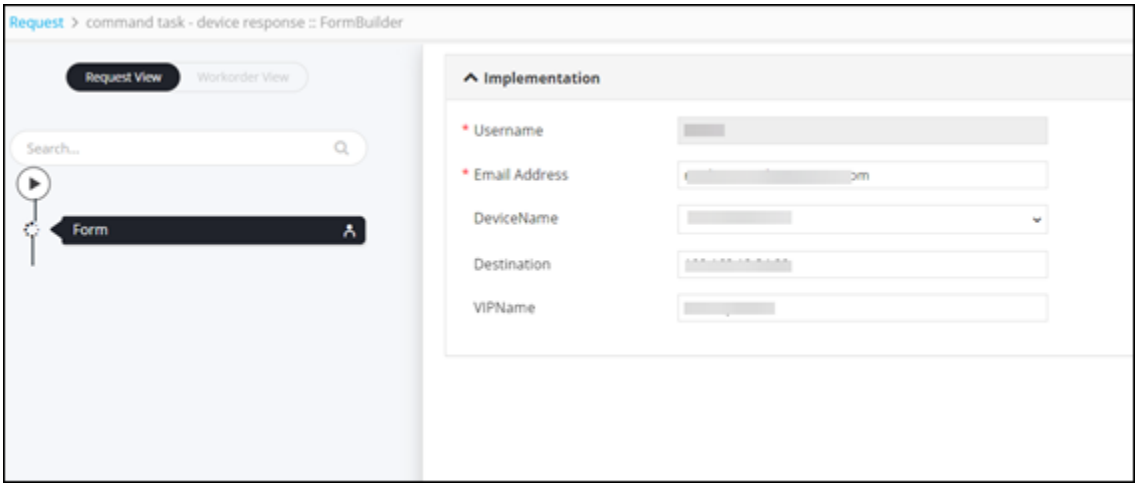
3. In the **If** task window, define the rules for executing the next task in the workflow using the Global variables defined in the Command task.
   For example, create a device if the CPU/TMM values are less than 25.

4. Trigger the workflow from the Request :: View/Run page.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

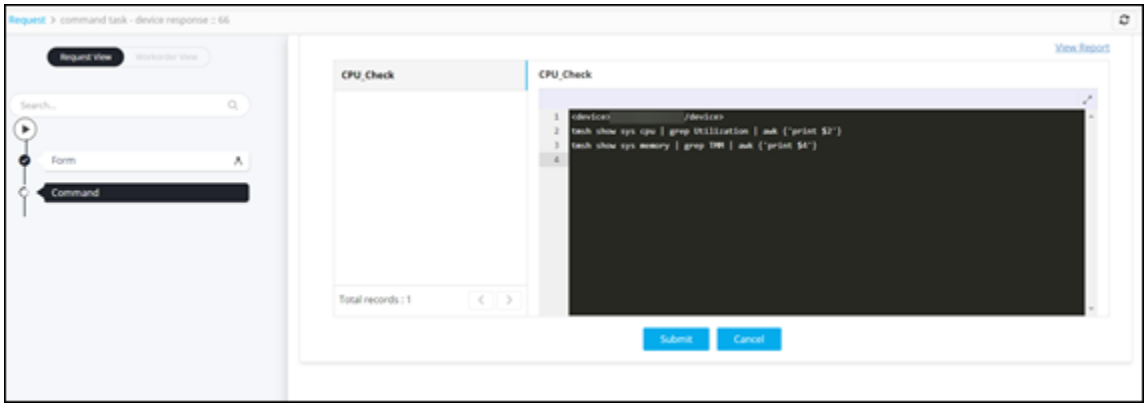


5. Enter the Destination and VIP Name in the Input form.

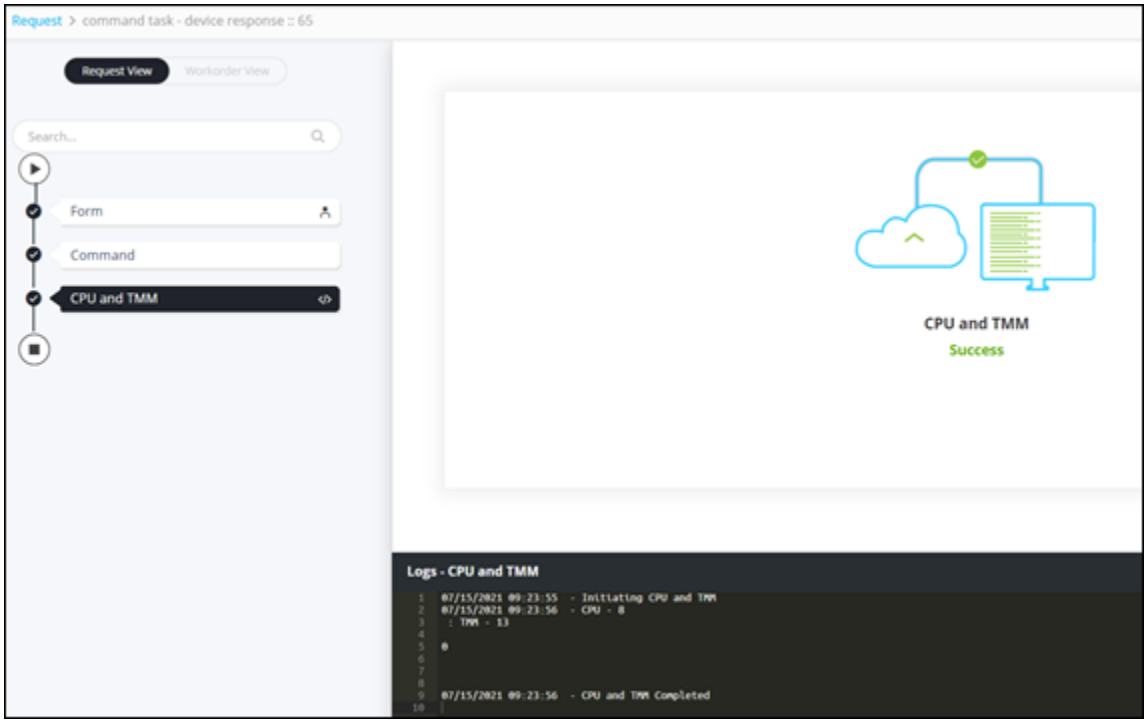


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Command Task initiated.



- CPU and TMM check completed.

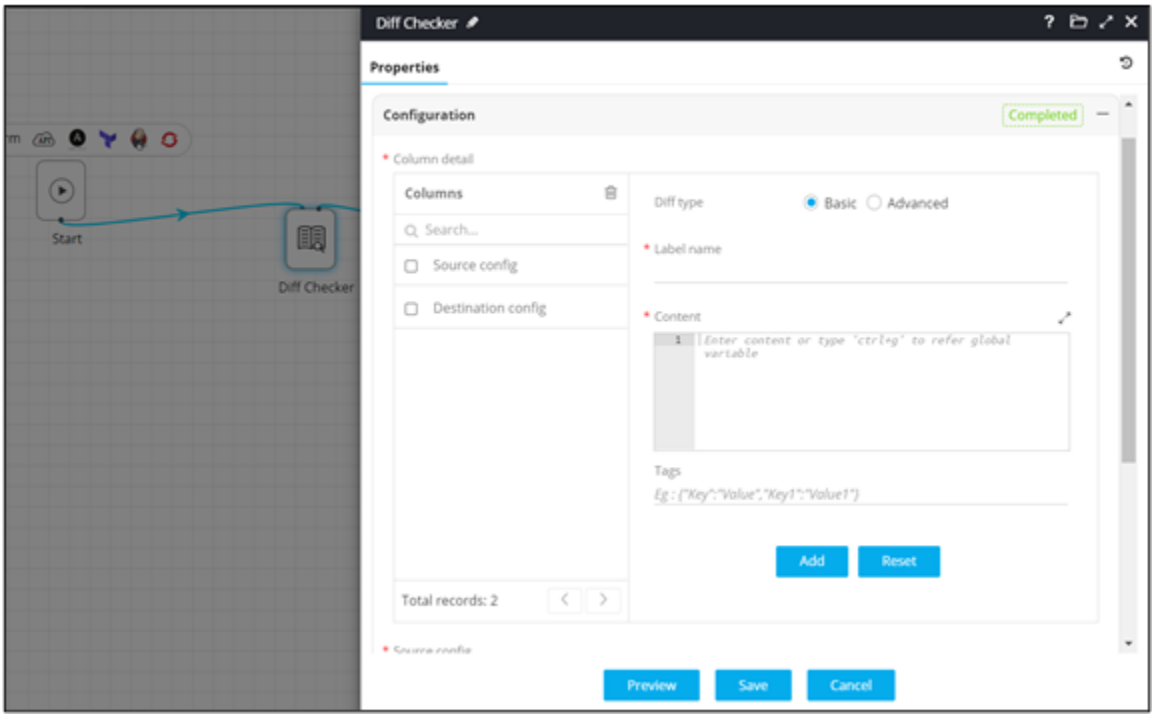


Diff Checker

The Diff Checker task allows you to perform a side-by-side comparison of configuration files. You can automate network configuration compliance and ensure that network device configurations adhere to best practices and standards, thus avoiding manual configuration errors.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Easy to build low-code diff checker task.
- Perform static and dynamic comparison of configurations across two to five devices/objects.
- Provision for basic and advanced diff checker options.
- Provision to reference variables from any workflow task for comparison.
- Provision to define custom hooks (scripts, REST API) within the advanced diff checker.
- Provision to compare differences in configuration before pushing to the end device.
- Provision to compare changes in peer review and approval process.

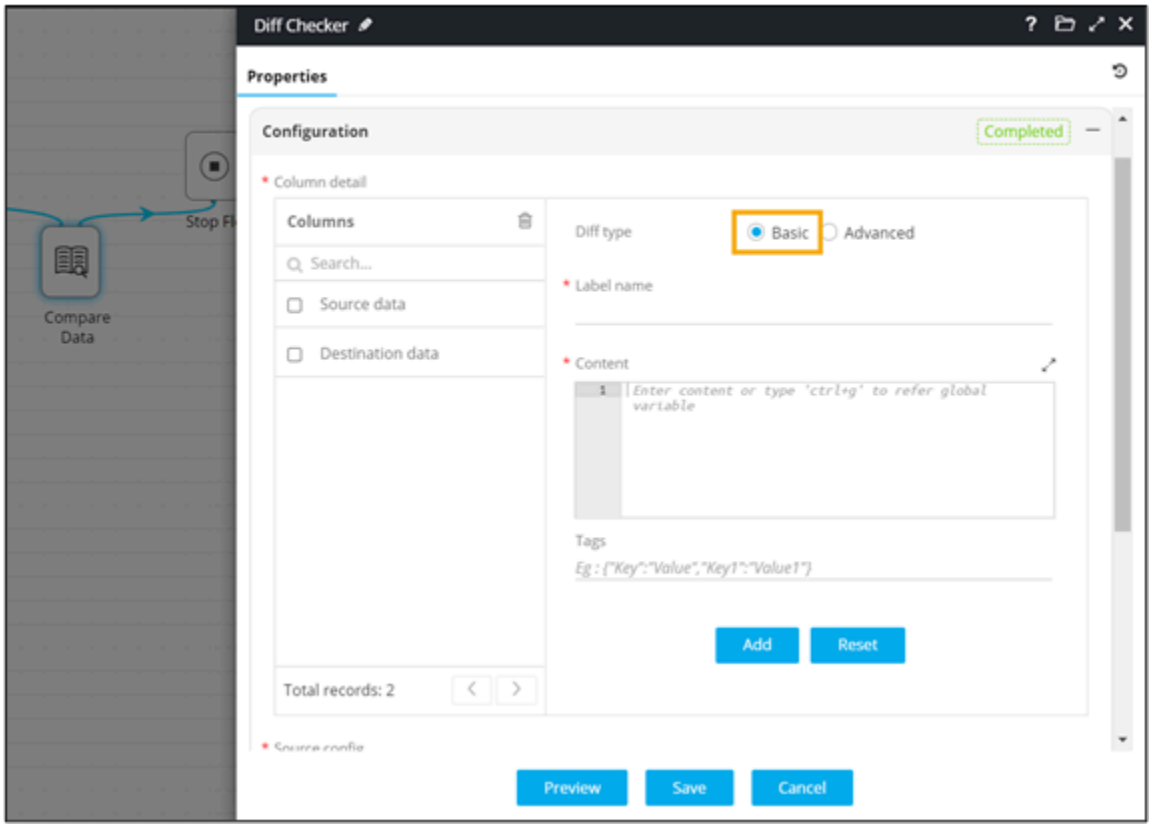


- Configuring a Basic Diff Checker
- Configuring an Advanced Diff Checker

Configuring a Basic Diff Checker

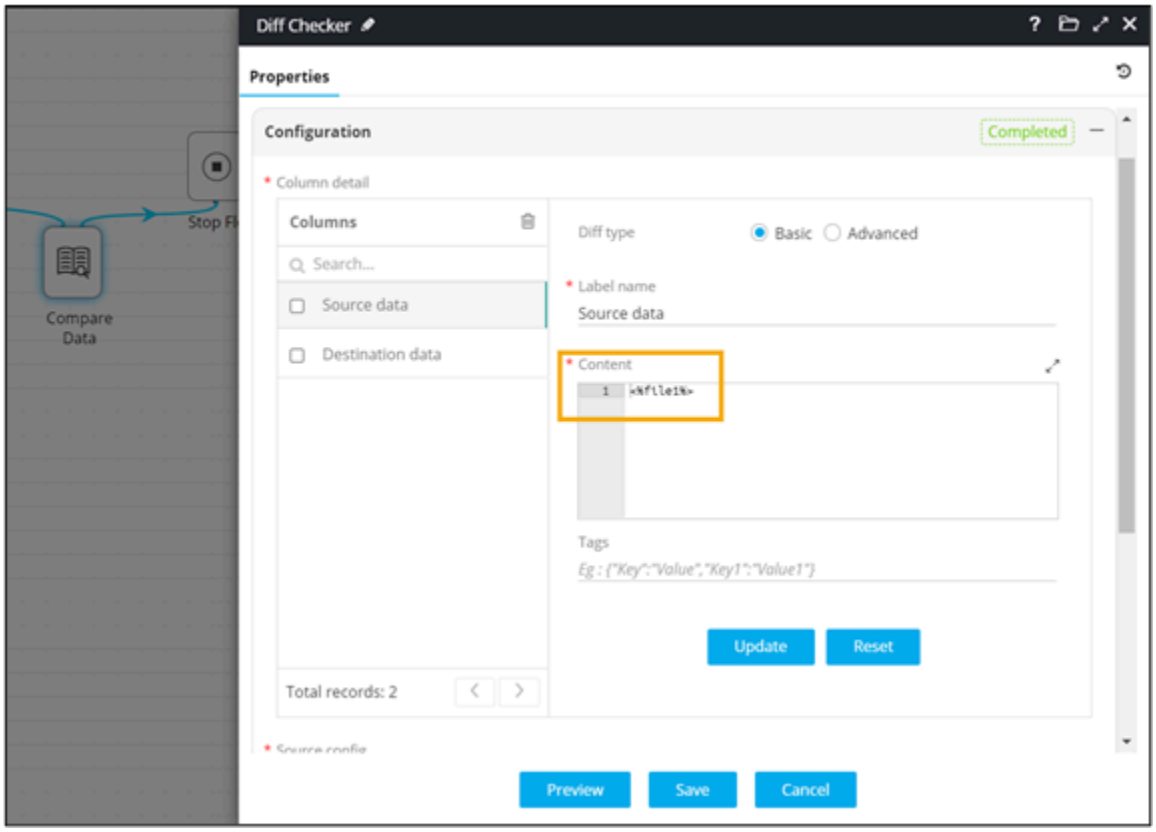
1. Design a workflow.
2. Drag and drop the **Diff Checker** task from the **User Interface** section.
3. In the **Diff Checker** task window, under **Properties**, in the **Configuration**section, select **Diff type** as **Basic**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

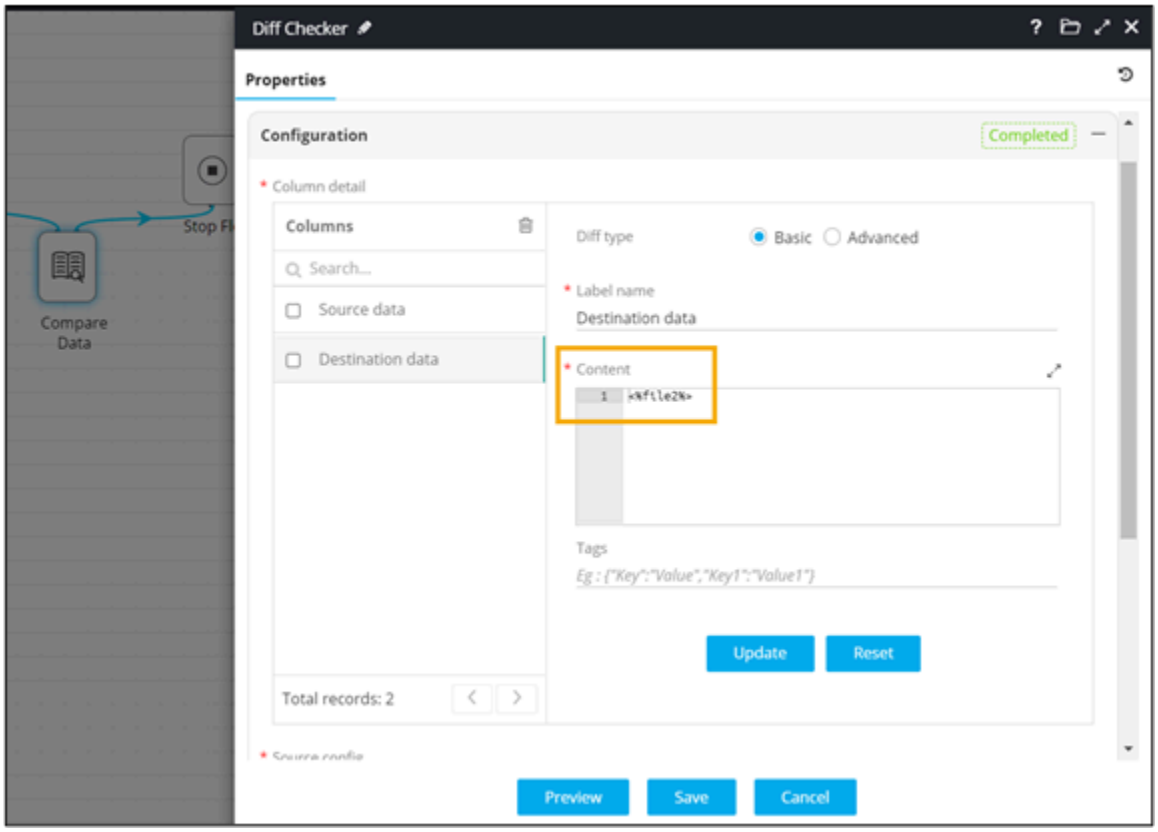


4. Enter the **Content** to be compared or reference content from any previous workflow task as a variable.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

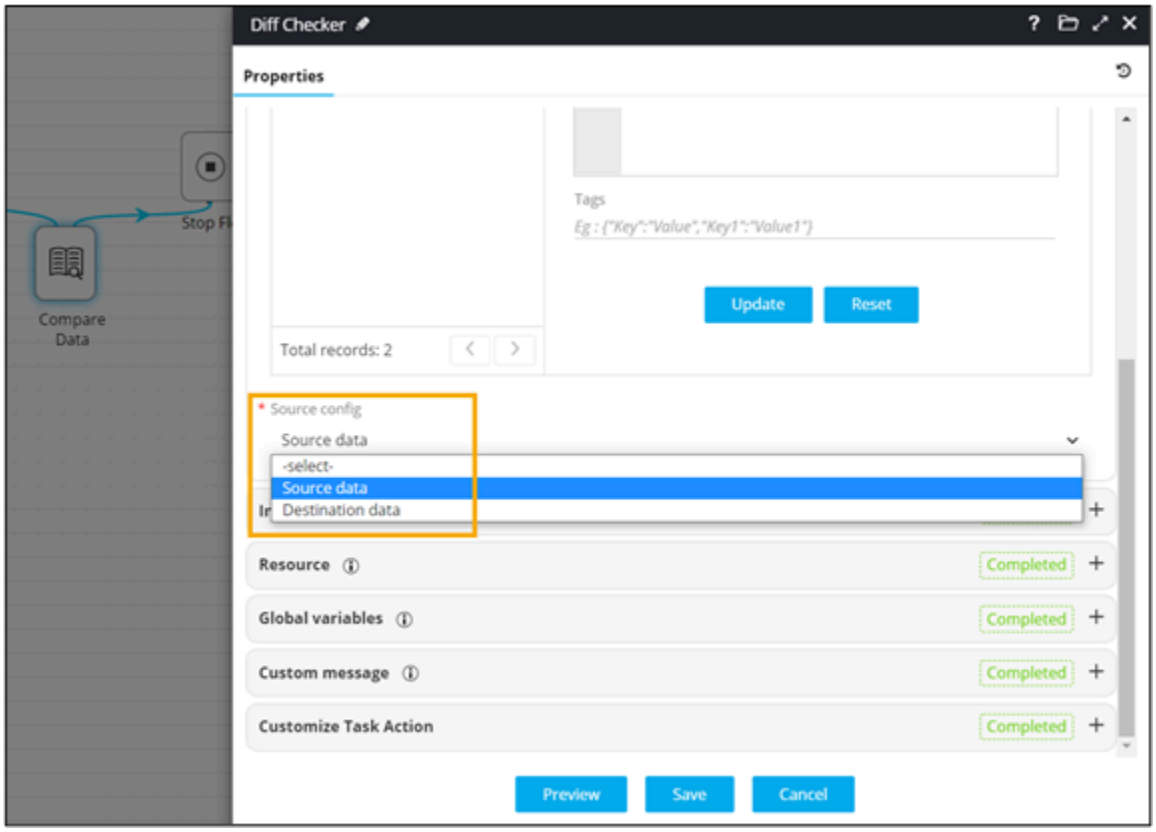


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

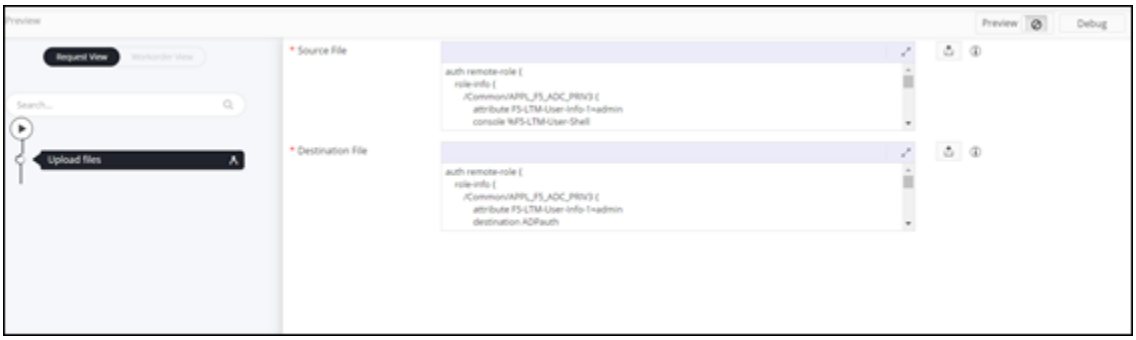


5. Under **Source config**, select the source column against which the diff checker comparison must be made.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



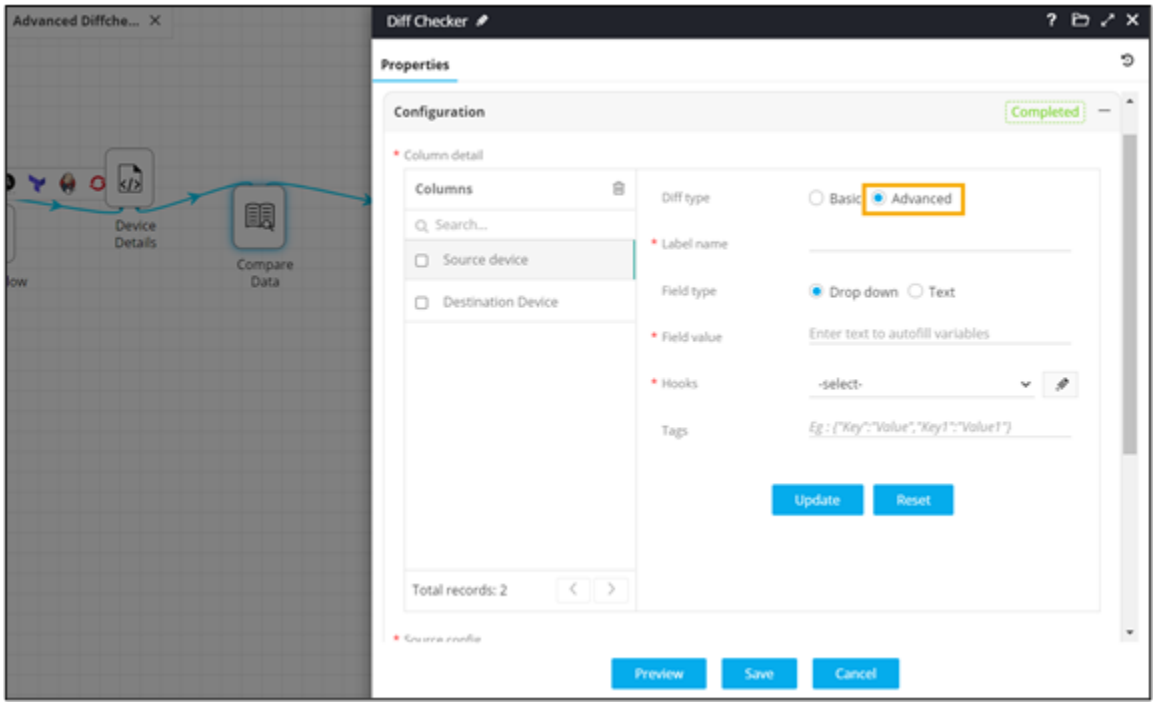
6. Click **Preview**.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

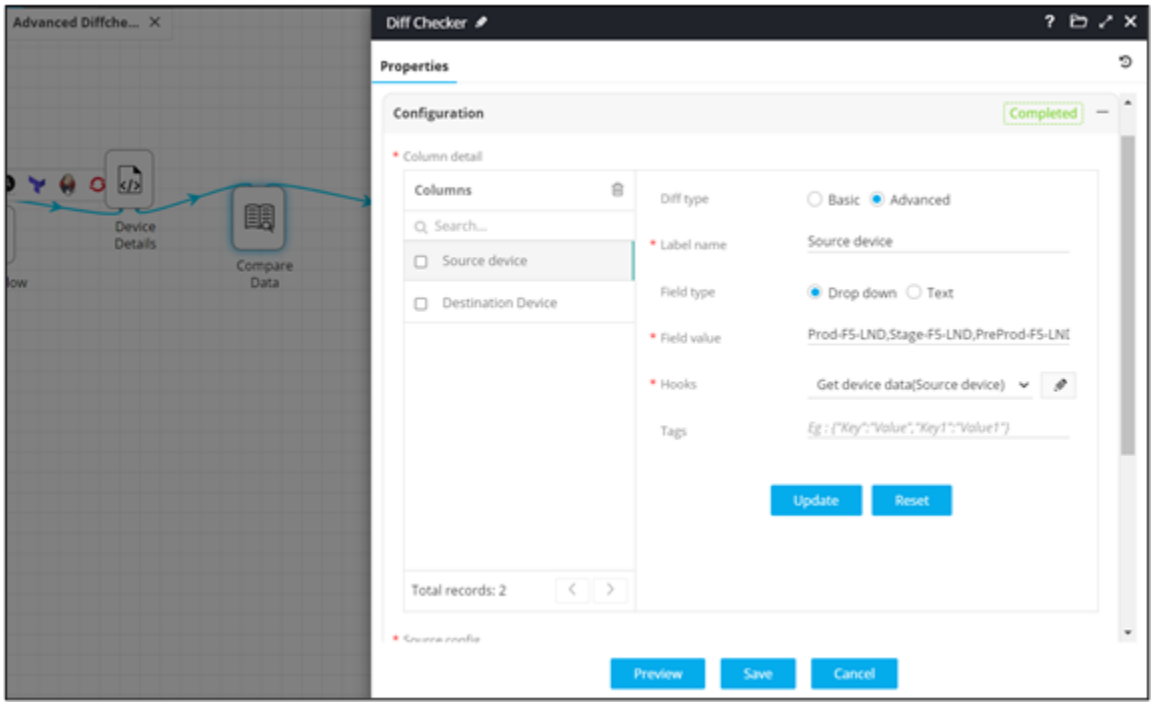
Configuring an Advanced Diff Checker

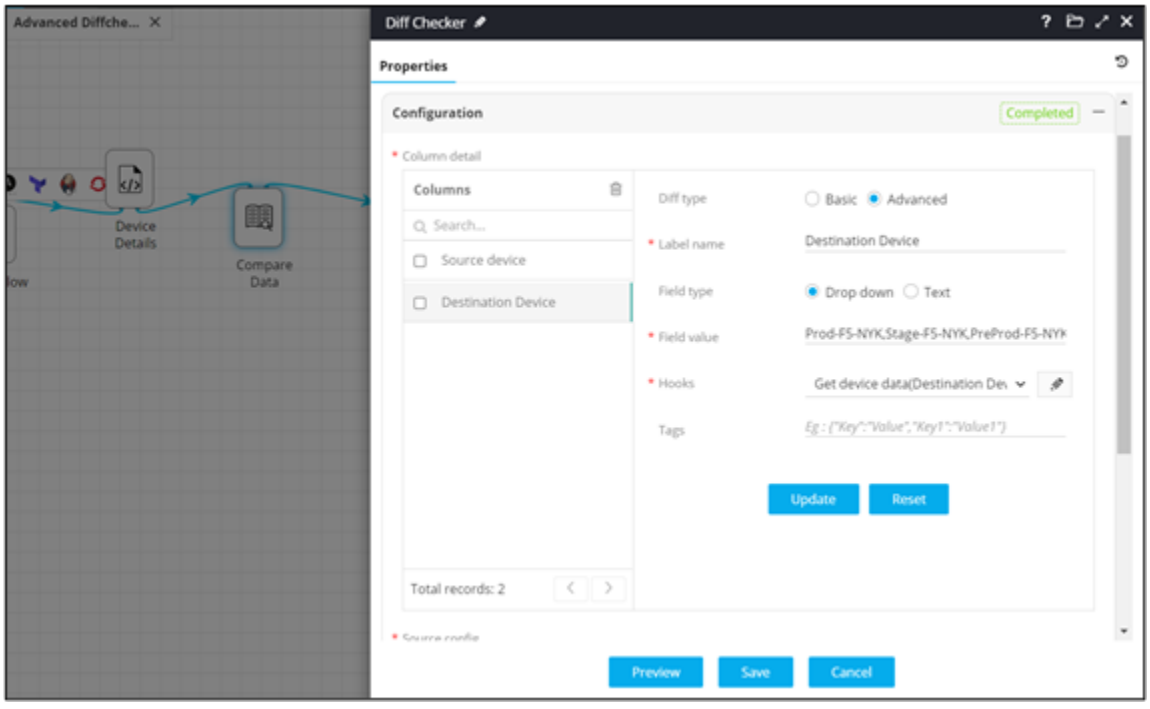
1. Design a workflow.
2. From the **User Interface** section, drag and drop the **Diff Checker** task.
3. In the **Diff Checker** task window, under **Properties**, in the **Configuration**section, select **Diff type** as **Advanced**.



4. Enter or select the field information as required.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



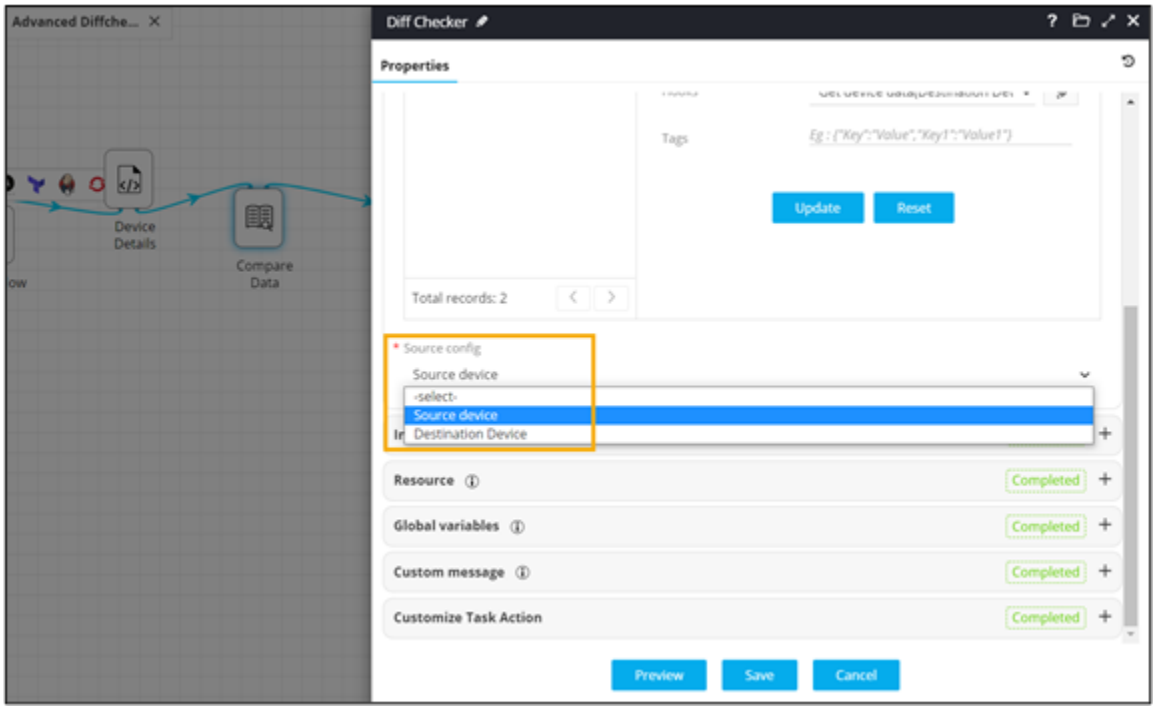


This table describes the field information in this section:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

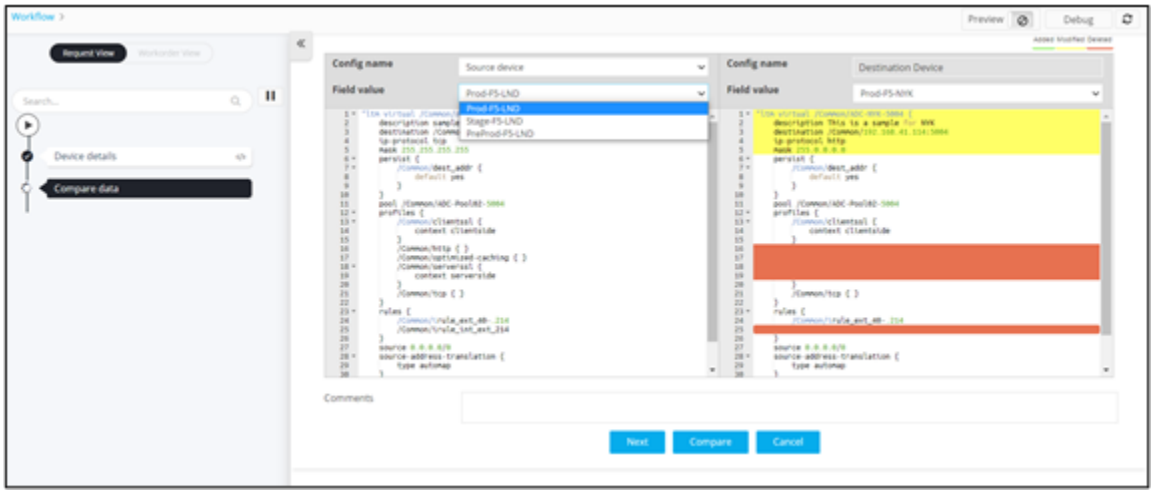
Field	Description
***Label Name**	Define custom label names as headings for every diff checker column.
Field Type	Allows you to define the input required to execute a hook dynamically during the diff checker view in real time. It allows for either providing the inputs dynamically as: • **Text**: Values can be entered within a text box which will be taken as an input variable to execute the hook. • **Dropdown**: Inputs to execute the hooks can be defined as dropdown values allowing hooks to trigger dynamically.
***Field Value**	Define either a static or dynamic value through a variable to be displayed in the dropdown field.
***Hooks**	Select an existing hook from the Hooks Inventory or define a custom hook specific to the diff checker task.
All asterisk (*) marked fields are mandatory.	

5. Under **Source config**, select the source column against which the diff checker comparison must be made.



6. Click **Preview**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

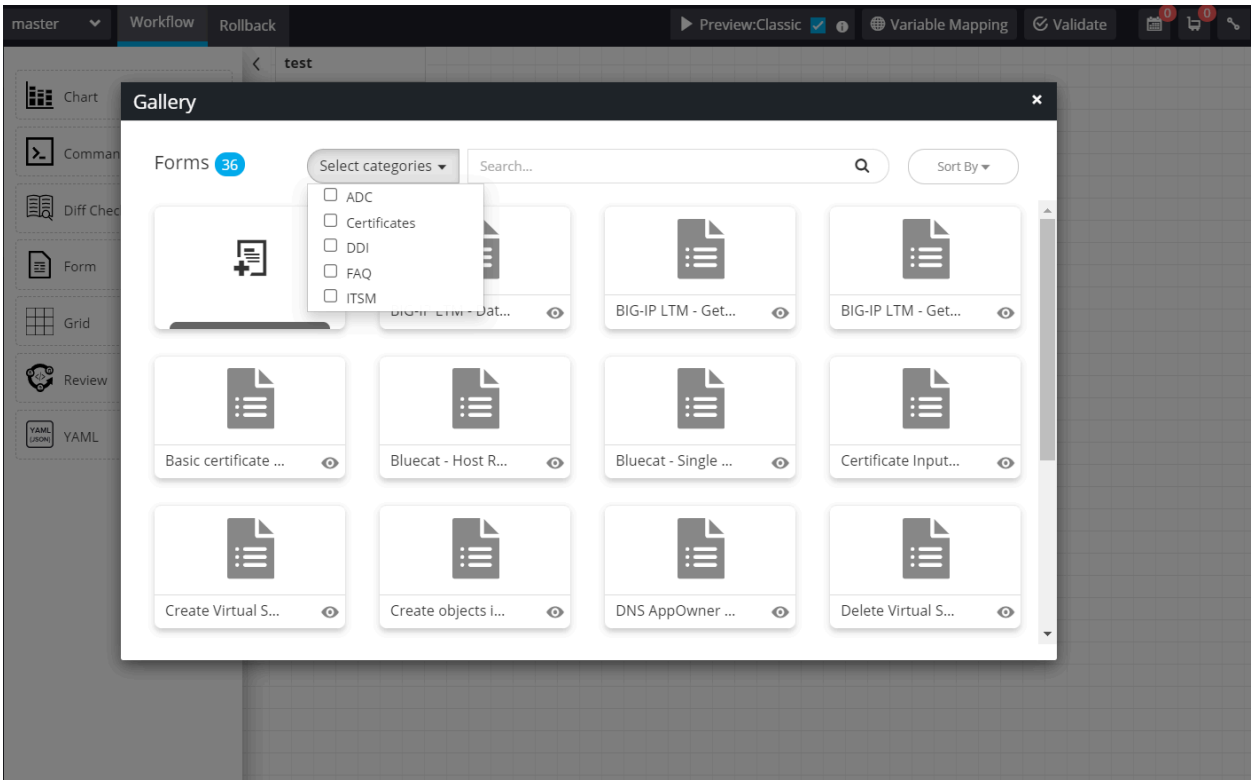


Form

Forms enable easy self-servicing capabilities to the Line of Business (Application owners, NetOps and so on) via an intuitive GUI-based interface, thus abstracting the end user from underlying infrastructure configurations.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

The form can either be used out of the box form from the inventory for quick self-servicing or customized using the Form builder based on your automation requirements to build simple and complex workflows.



- Out of the box Forms
- Designing a Custom Form
- Adding Form Fields in the Form Builder
- Using Hooks in a Form
- Defining Resource & Settings

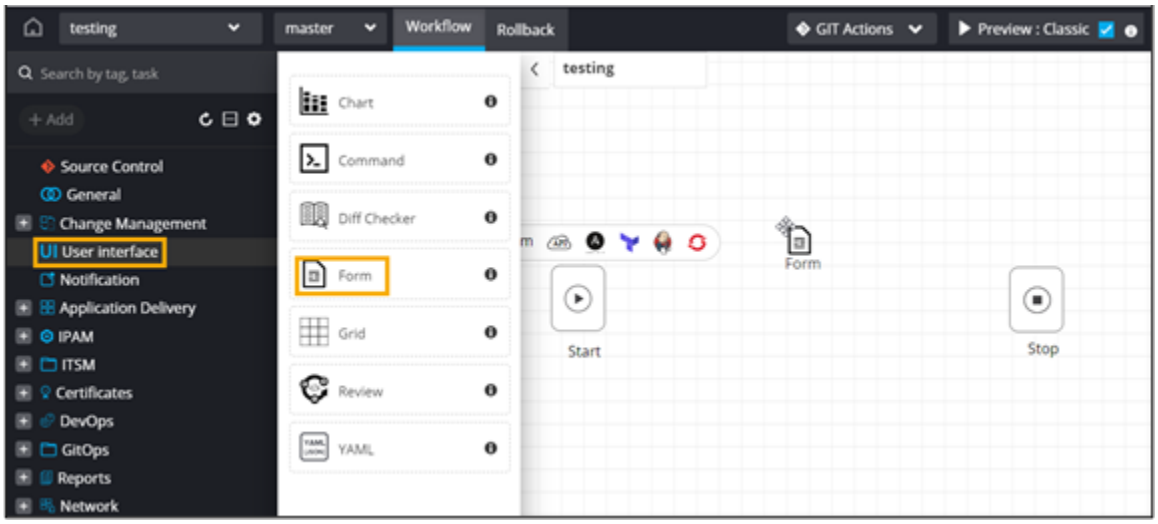
Out of the box Forms

You can customize out of the box (OOB) forms to build workflows.

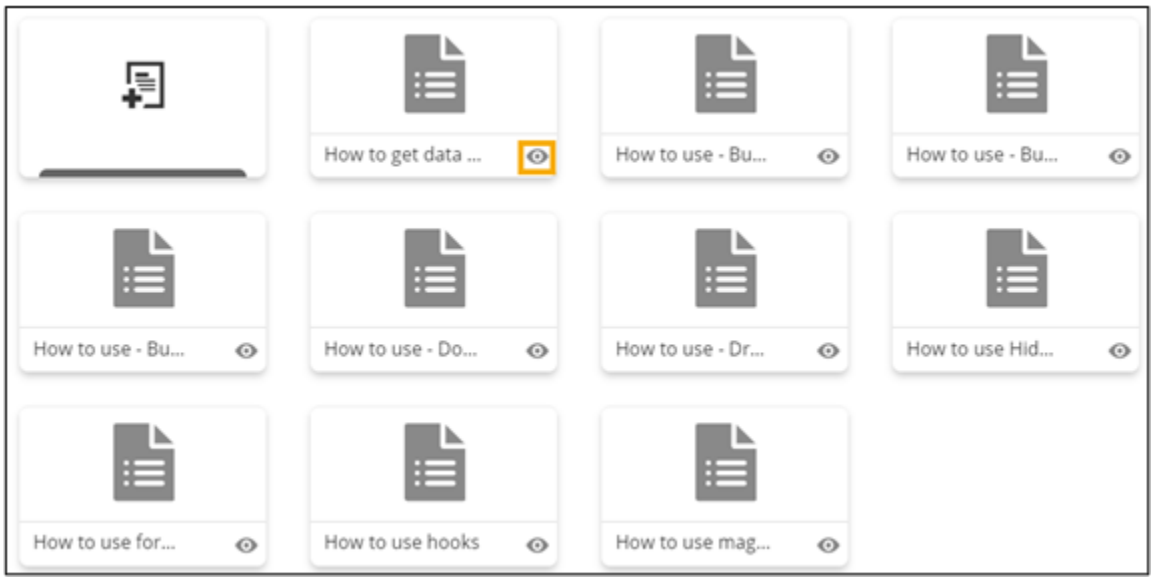
To use an OOB form:

1. Design a workflow.
2. From the **User Interface** section, drag and drop the **Form** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

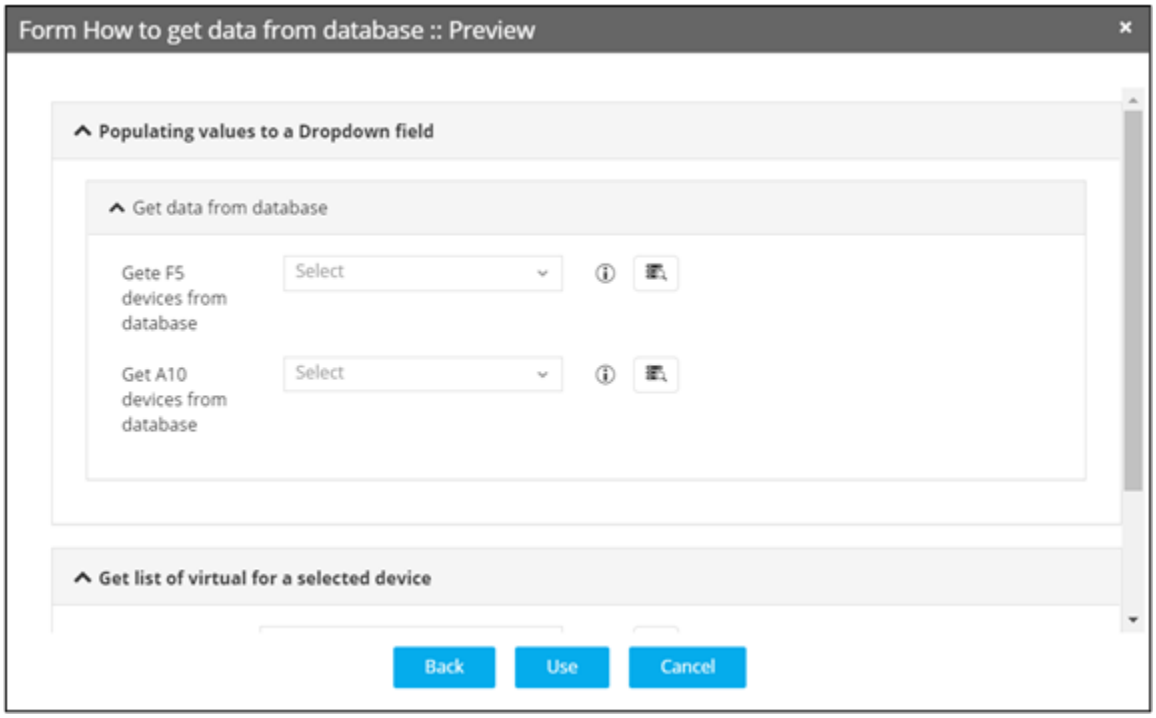


3. In the **Gallery** window, to preview the form, click on any of the forms.



Form preview.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



4. To use this form, click **Use**.
5. To go back and preview another form, click **Back**.

Designing a Custom Form

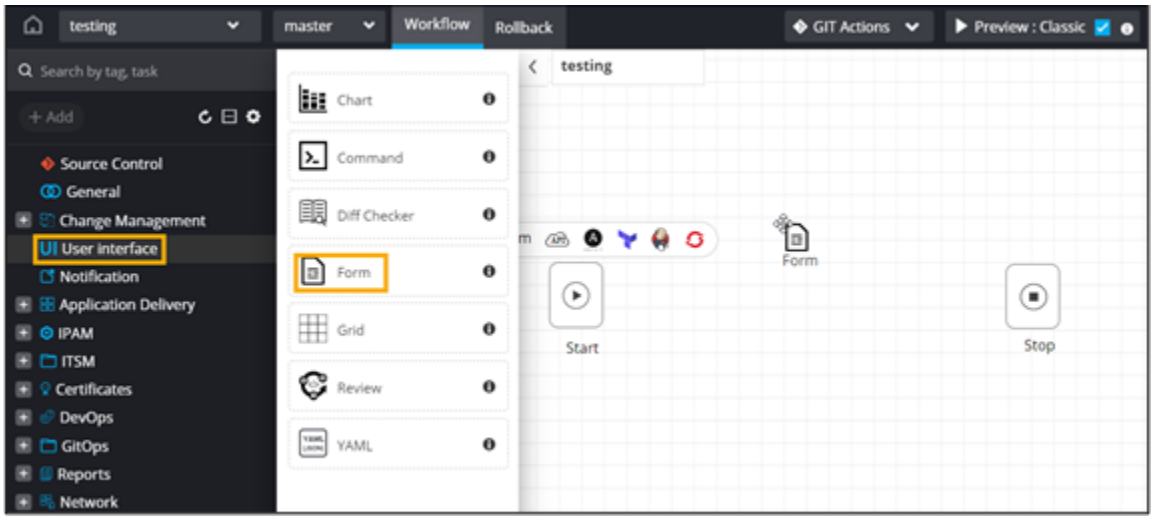
You can build a dynamic self-service form or catalog using the form builder GUI in order to capture user inputs as part of the workflow execution process. The Form builder allows you to:

- Build a custom form.
- Associate custom script(s) within the form builder.
- Assign form access to specific user role(s).
- Declare one or more form field values as ‘global variables’.

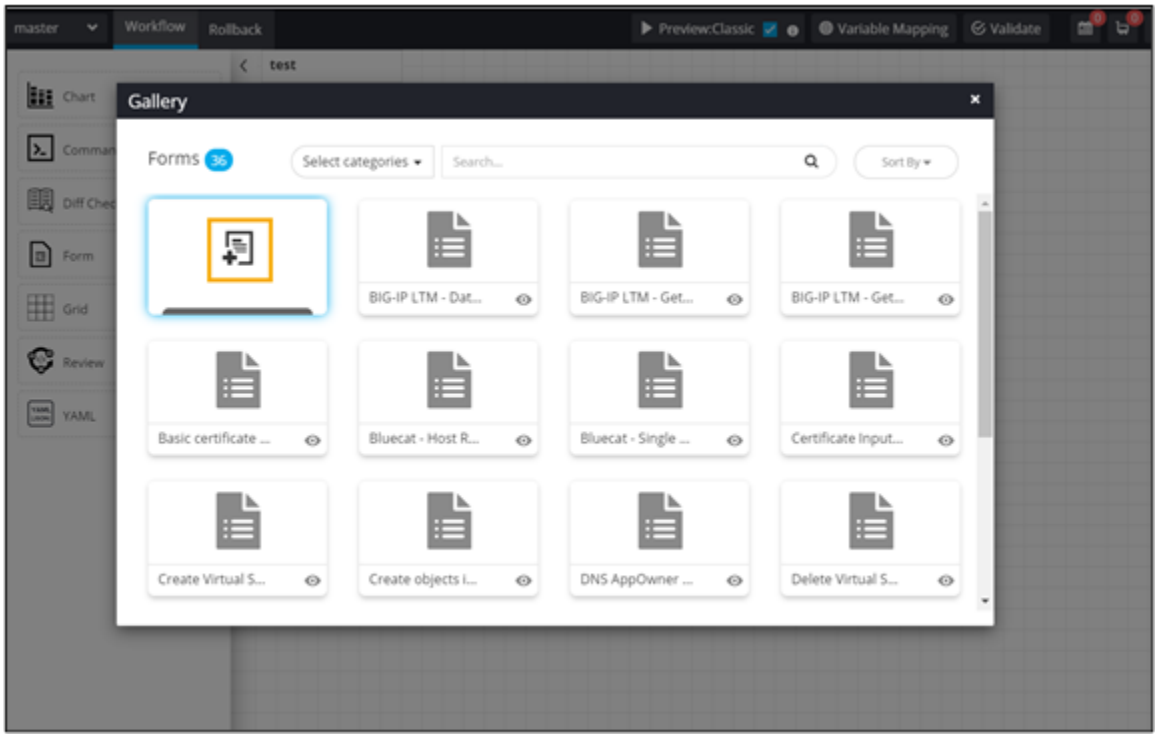
To design a custom form:

1. Design a workflow.
2. From the **User Interface** section, drag and drop the **Form** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

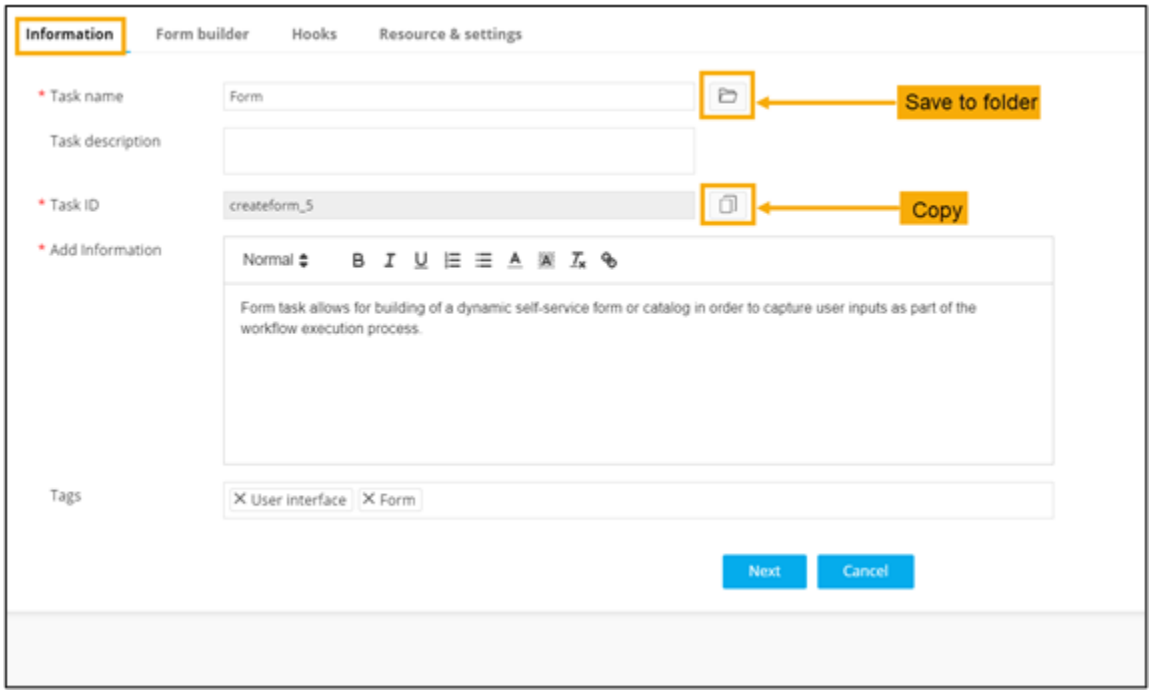


3. To design a new form, in the **Gallery** window, click .



4. Under **Information**, enter the necessary field values.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

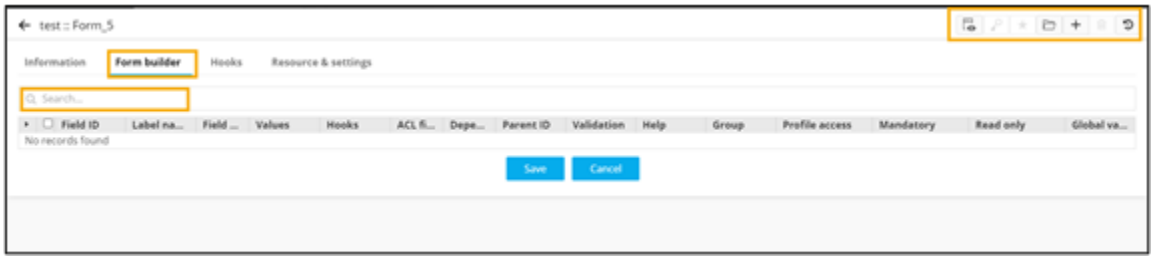


This table describes the fields under the **Information** tab:

Field	Description
***Task name**	Provide a unique task name.
	Provision to bookmark the task within a custom folder in the workflow studio.
Task description	Enter a valid task description.
***Task ID**	Enter a unique task ID of the form task.
	Provision to copy the task ID and use it across the workflow.
***Add Information**	Provide information about the form task.
All asterisk (*) marked fields are mandatory.	

5. Under **Form builder**, add the form fields.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



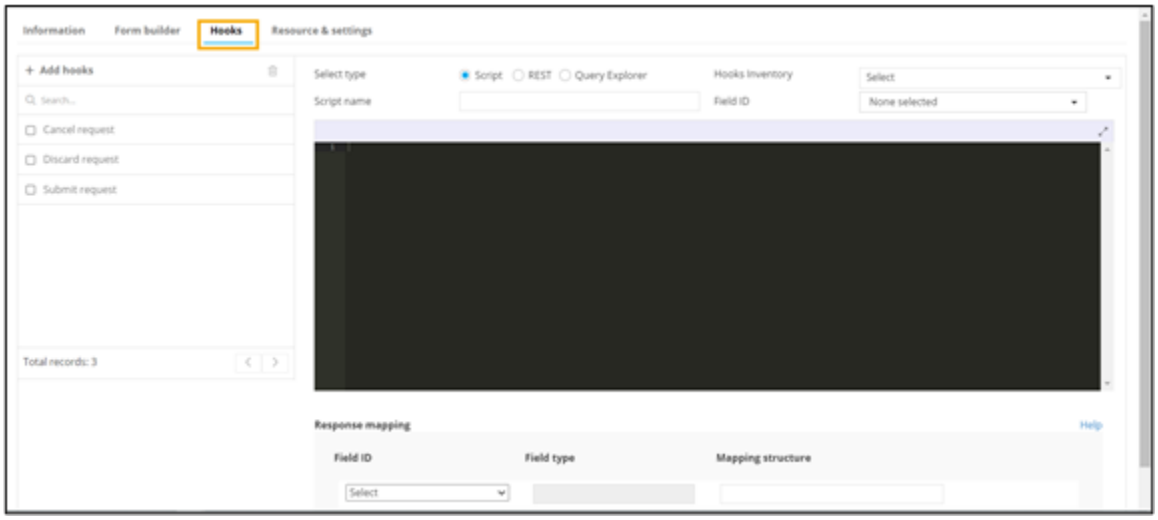
This table describes all the options available under the **Form builder** tab:

Option	Description
Search	Allows you to search for a specific form field.
	Allows you to preview the form.
	Allows you to select one or more form fields and assign to a user profile.
	Allows you to select one or more form fields and mark them as mandatory or otherwise.
	Allows you to select multiple form fields and group them logically.
	Allows you to add multiple form fields.
	Allows you to select one or more form fields and delete them.
	Allows you to view form history.

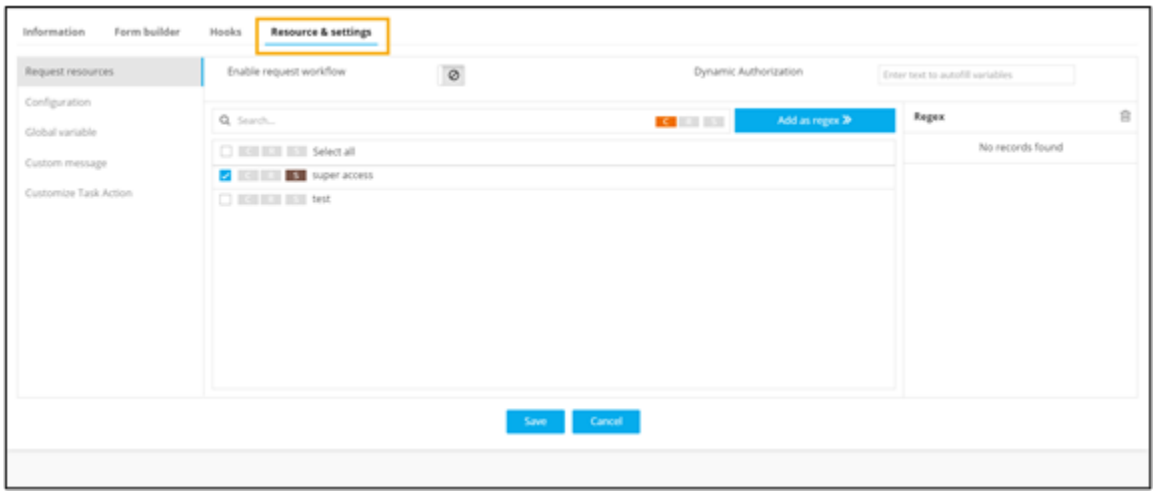
Note: For more information on adding form fields, click here.

6. To query different data sources such as database, device, or external systems and aid in mapping the response values to a form field, click **Hooks**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



7. To assign access to specific roles(s), click **Resource & Settings**.

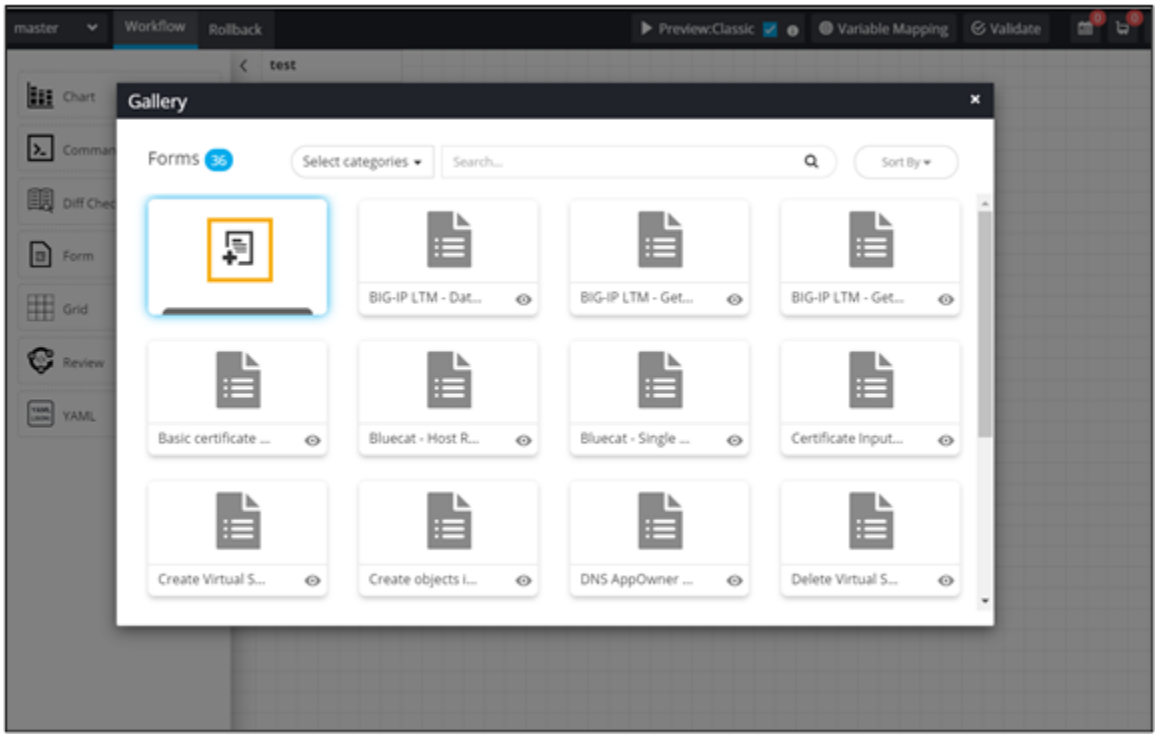


Adding Form Fields in the Form Builder

To add a form field:

1. Design a workflow.
2. From the **User Interface** section, drag and drop the **Form** task.
3. To design a new form, in the **Gallery** window, click .

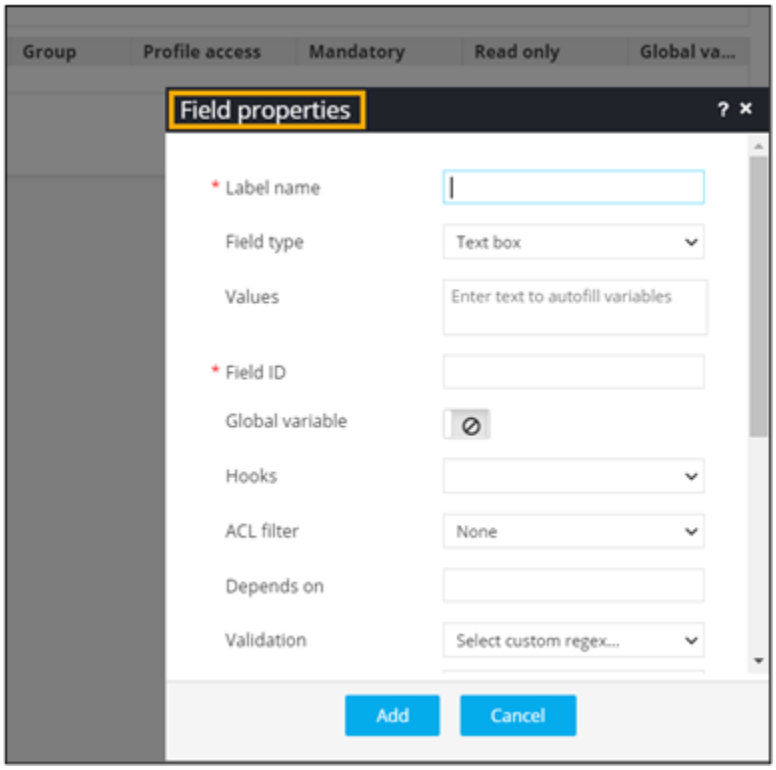
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

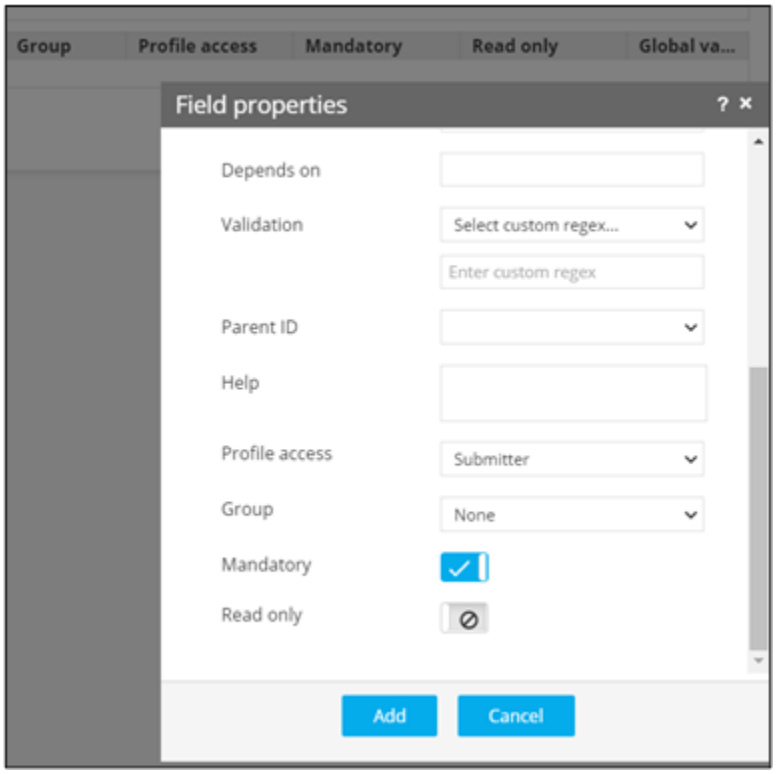


4. Under the **Form Builder** section, from the upper right corner of the screen, click +.

5. In the **Field properties** window, enter or select the field information.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.





Copyright © 2020 AppViewX, Inc. All Rights Reserved.

The following table describes the form fields:

Field	Description
***Label name**	• Enter a valid label or field name such as Username, Enter IP address, Environment Type, Environment Info etc. • Char Length: Min (2), Max (120)
Field type	Select field type from the dropdown.
Values	• Define field values relevant to a form field. • Values can be delimited by a comma. • Chars not allowed: '&' and '|'
***Field ID**	• Enter a unique field ID to reference the field name. • Allowed characters: '-' and '_' • Character Length: Min (1), Max (50)
Global variable	Assign the field as a global variable.
Hooks	• Select a hook that is relevant to the from field. • Hooks can be either Script or REST API. • Hooks are used to fetch details against a specific form field from the device and/or database and map them to the respective field on the user interface.
ACL filter	• This field is used to validate, filter information displayed in the provisioning request page based on RBAC defined through 'authorized control' under account management. • Options: Device, Device Objects, Cert Groups, None (default)
Depends on	• Define dependencies between form fields based on 'Field ID' and 'Values'. • Operators used: '&' and '|' • Max character: 550 • Format: [Field Id1: value1 & Field Id2: value1 & Field Id3: value3], [Field Id1: value1 | Field Id2: value1]
Parent ID	• Select the relevant tabular component (Field ID) against which the form field must be mapped to. • Used only when defining a tabular element in the form.
Help	Enter the help instructions that need to be defined against the field for the user's reference.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

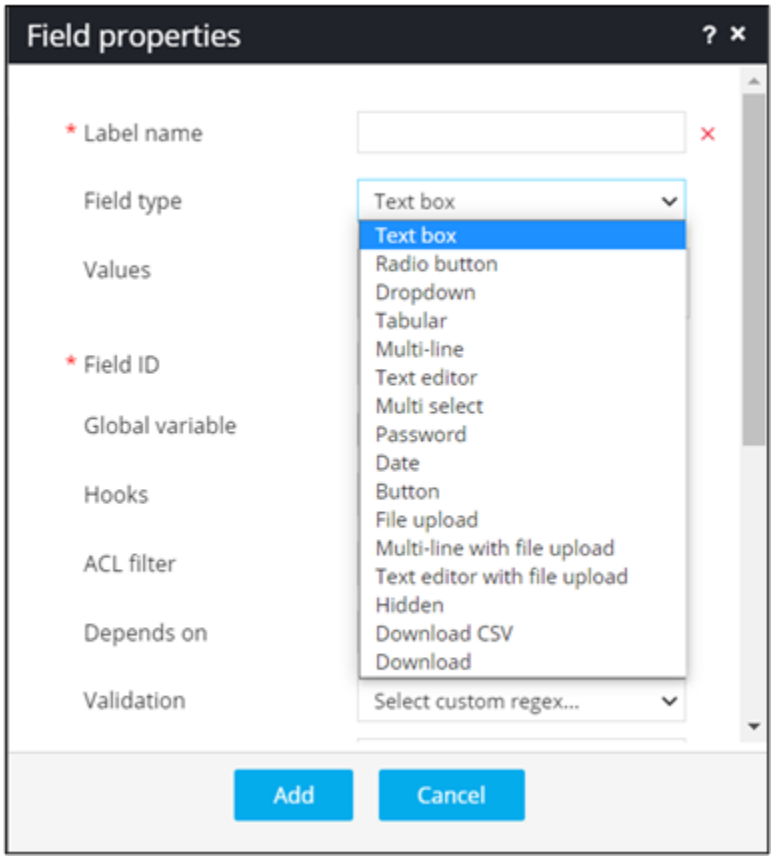
Field	Description
Profile access	Select the profile from the dropdown to restrict form field display within a template based on the relevant profile.
Group	• Group fields logically within the form. • Associate an existing group or create a new group.
Mandatory	Provision to mark the field as mandatory or otherwise.
Read only	Provision to mark the field as read only or otherwise.
All asterisk (*) marked fields are mandatory.	

- Field Types
- Validating Form Fields
- Defining Dependencies between Form Fields
- Defining Tasks as Read Only
- Grouping Multiple Form Fields

Field Types

The form builder allows for specific customization of the form fields using various field types.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



- Field Type - Text Box
- Field Type - Radio button
- Field Type - Dropdown
- Field Type - Tabular
- Field Type - Multi-line
- Field Type - Text editor
- Field Type - Multi select
- Field Type - Password
- Field Type - Date
- Field Type - Button
- Field Type - File upload
- Field Type - Multi-line with file upload
- Field Type - Text editor with file upload
- Field Type - Hidden

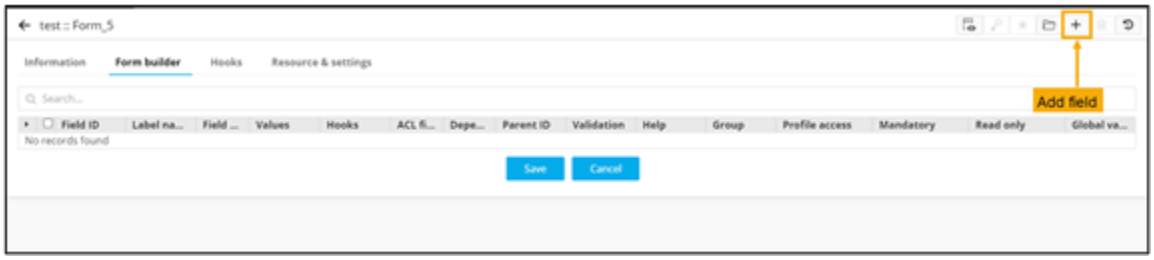
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Field Type - Download CSV
- Field Type - Download

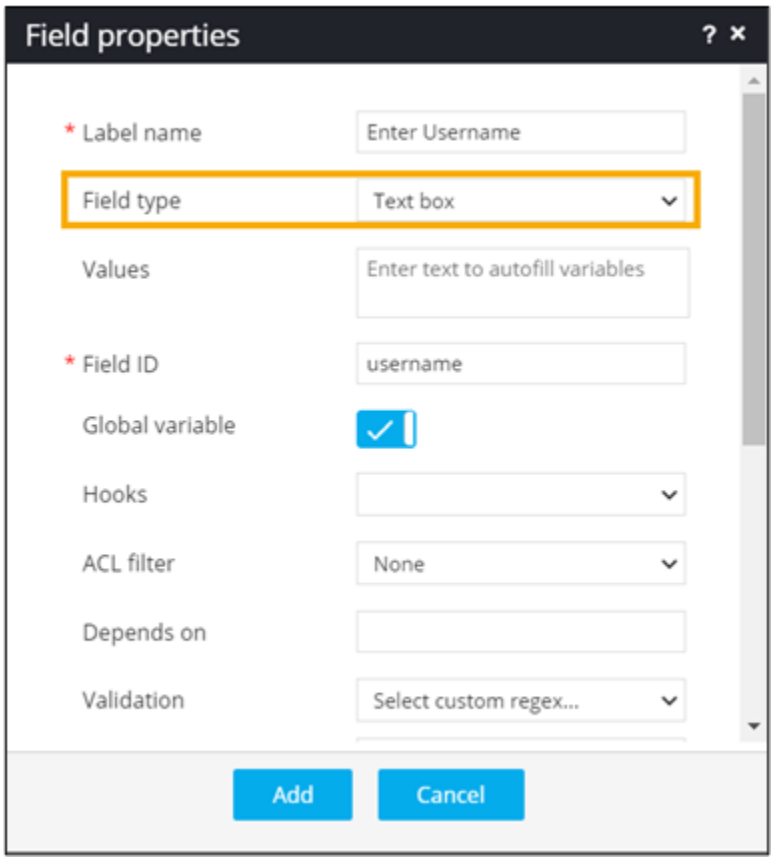
Field Type - Text Box

You can define a text box element on the form.

1. To add a **Text box** to the form, under **Form builder**, click + in the command bar.



2. In the **Field properties** window, enter or select the field information as shown.



3. To add this field to the form, click **Add**.

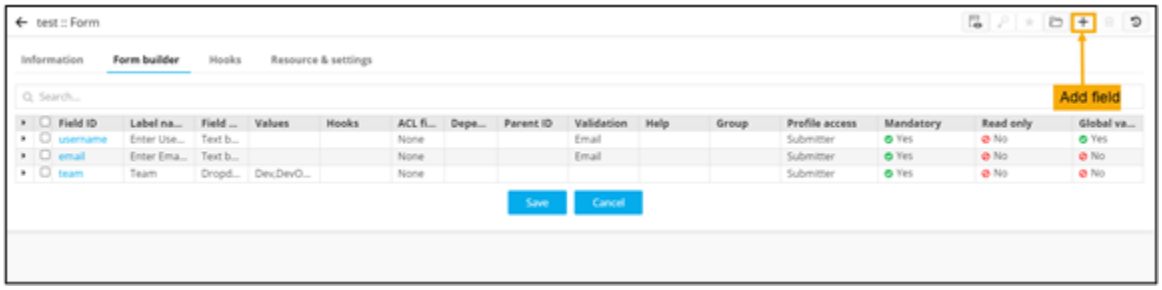
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

4. To see the text box added to the form, click .

Field Type - Radio button

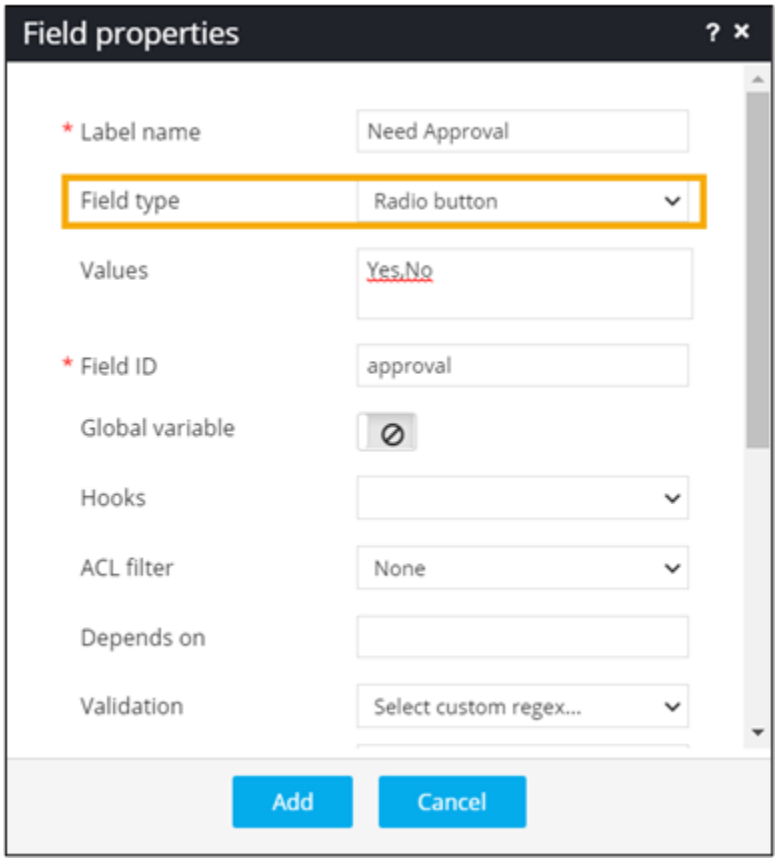
You can define a multi-select option using a radio button on the form.

1. To add a **Radio button** to the form, under **Form builder**, click in the command bar.

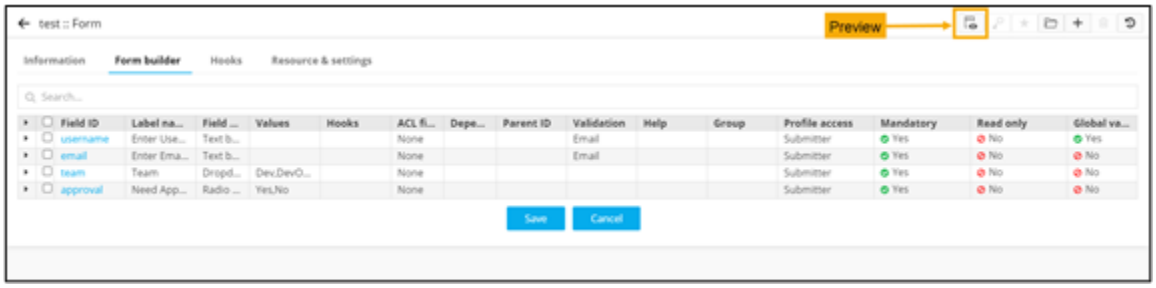


2. In the **Field properties** window, enter or select the field information as shown.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



3. To add this field to the form, click **Add**.
4. To see the Radio button added to the form, click .

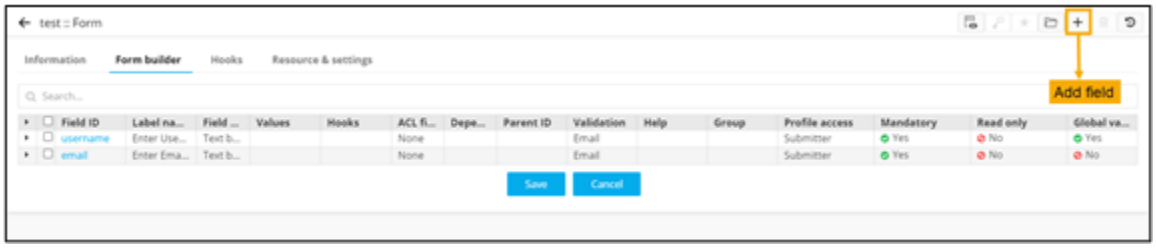


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Field Type - Dropdown

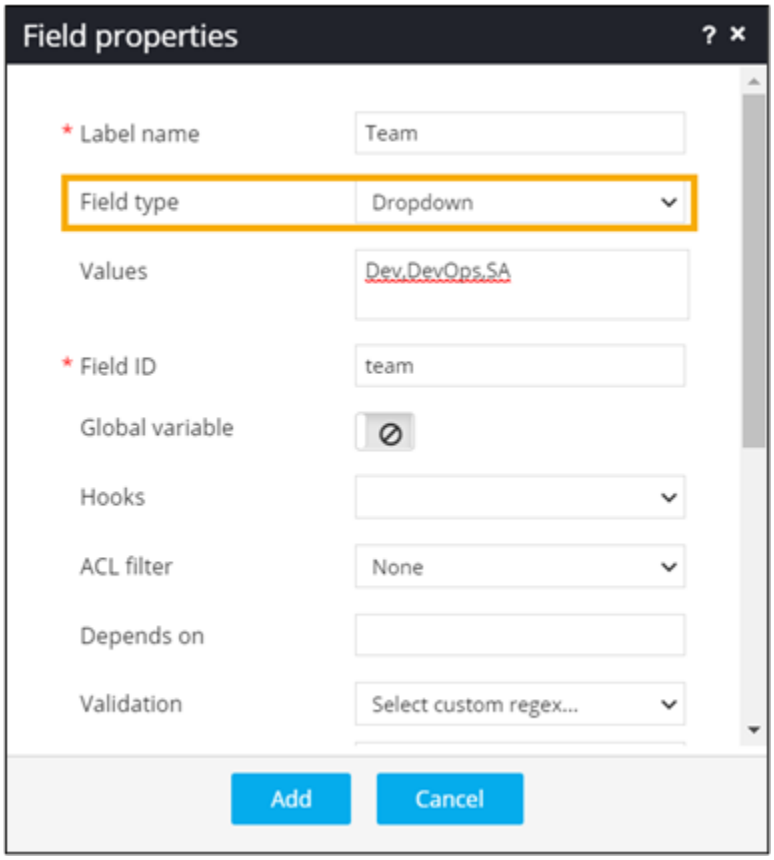
You can define a dropdown element with multiple values to select from in the form.

1. To add a **Dropdown** button to the form, under **Form builder**, click + in the command bar.



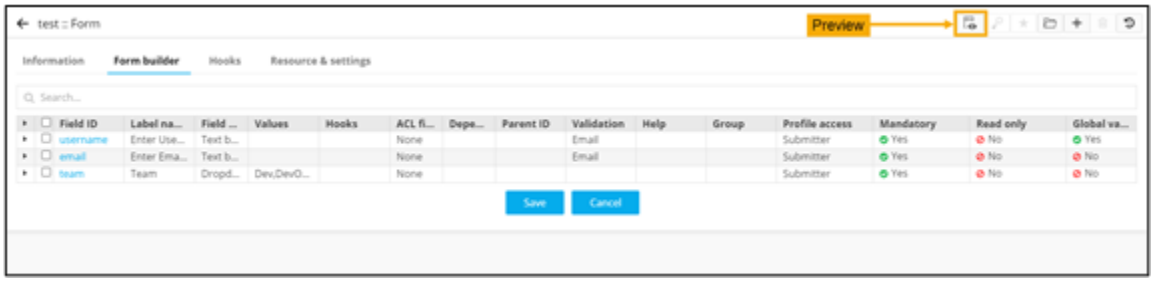
2. In the **Field properties** window, enter or select the information as shown.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

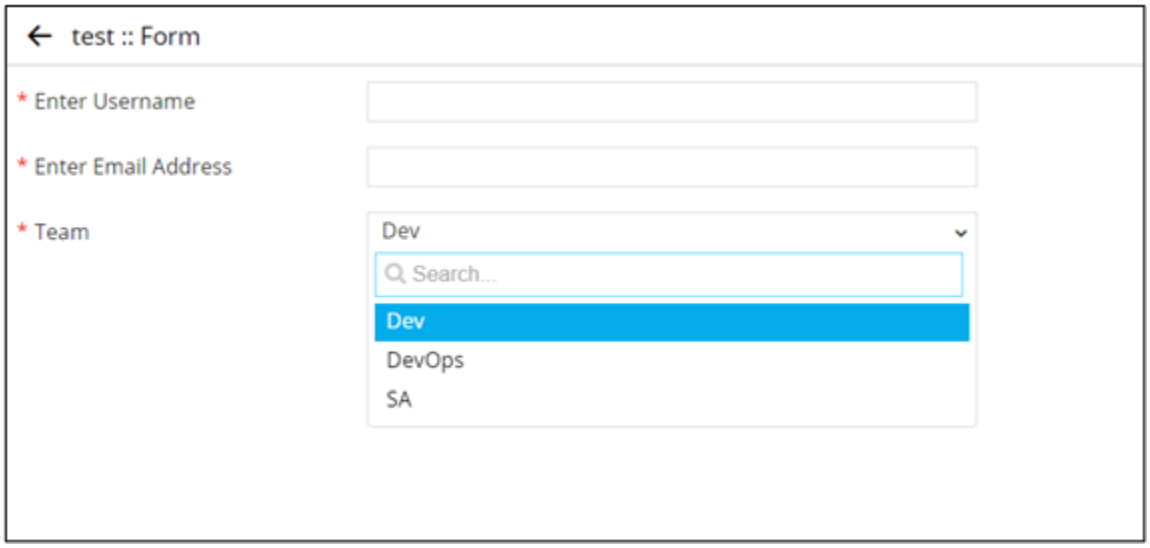


3. To add this field to the form, click **Add**.

4. To see the Dropdown field added to the form, click .



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

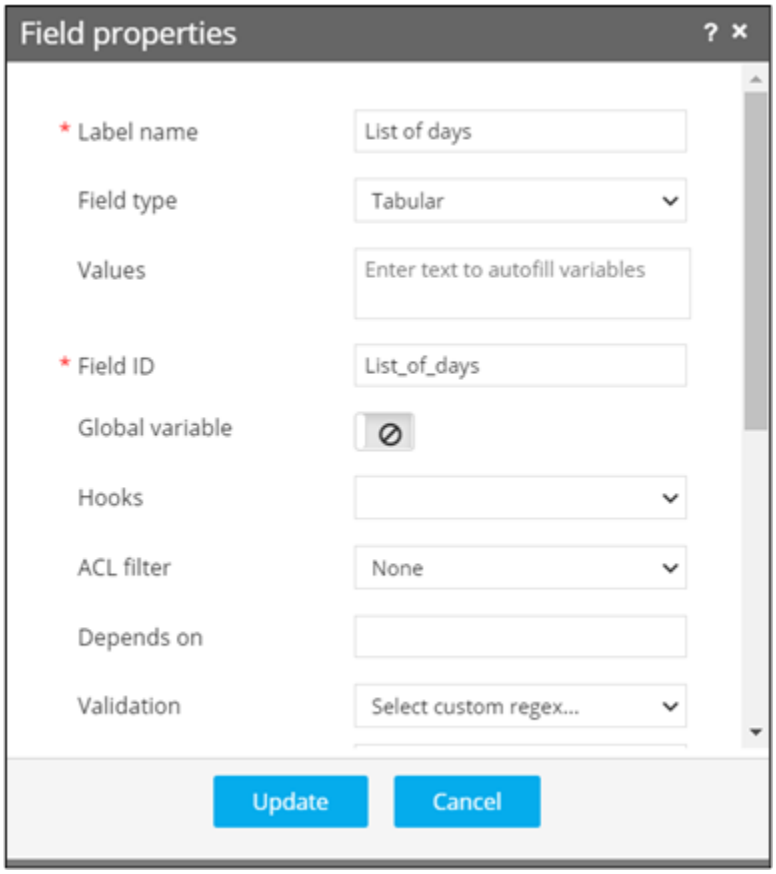


Field Type - Tabular

This field type in the form builder allows you to define a tabular component (grid) and add,modify, delete data from the grid.

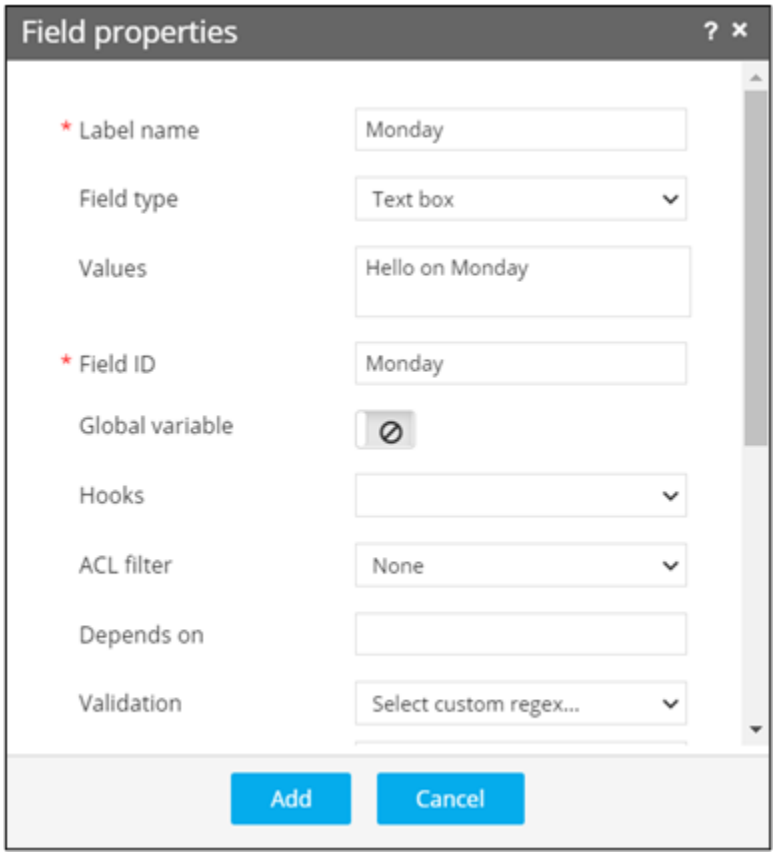
1. To add a **Tabular** field to the form, under **Form builder**, click + in the command bar.
2. In the **Field properties** window, define a form field with field type as **Tabular**, with the respective Field ID.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



3. Define other form fields that are required for the tabular component.
   For example: Label name: Monday, Field type: Text box, Values: Hello on Monday.

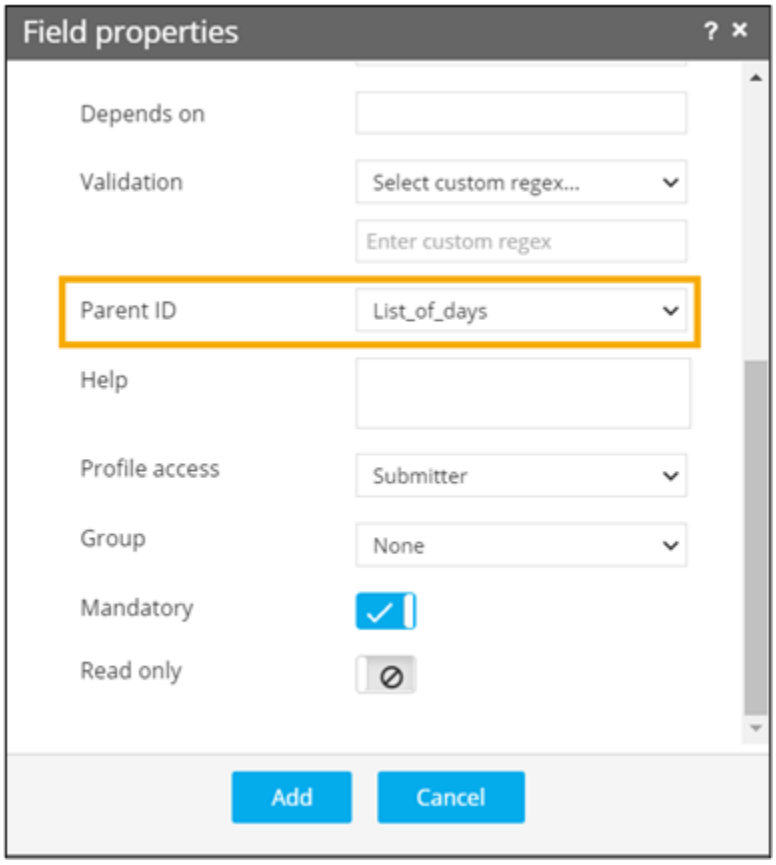
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



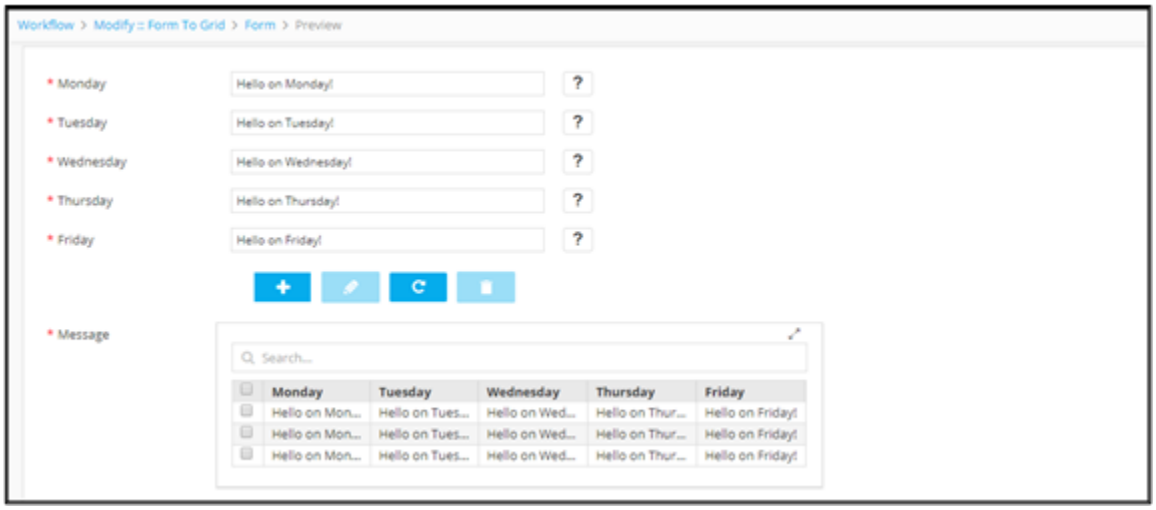
4. Link the Parent ID (the field ID of the parent tabular element) against these fields.

Label name	Field type	Values	Field ID	Parent ID
Monday	Textbox	Hello on Monday	Monday	List_of_days
Tuesday	Textbox	Hello on Tuesday	Tuesday	List_of_days
Wednesday	Textbox	Hello on Wednesday	Wednesday	List_of_days
List of days	Tabular		List_of_days	

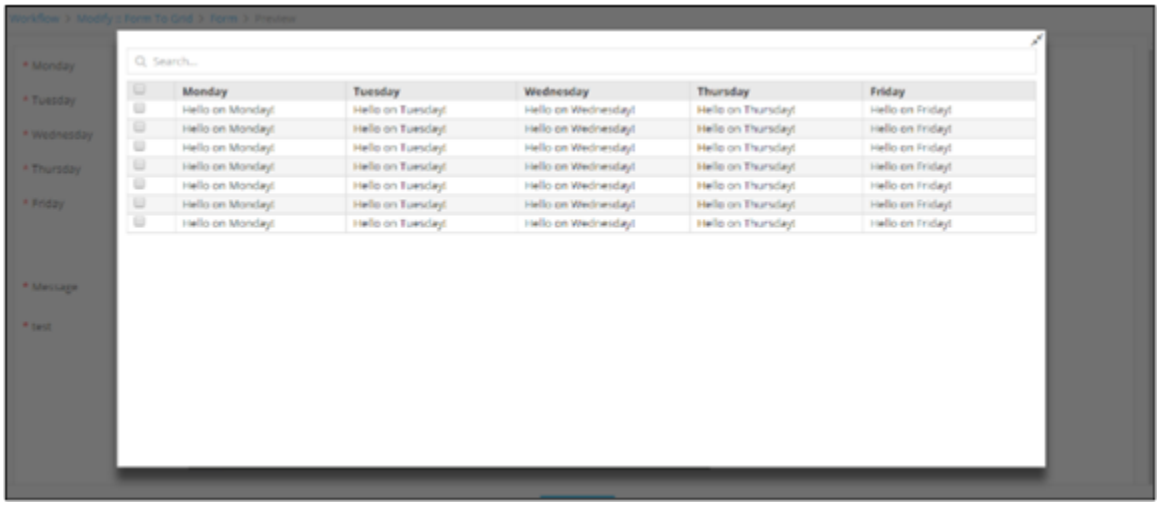
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



5. To see the Tabular field added to the form, click .



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

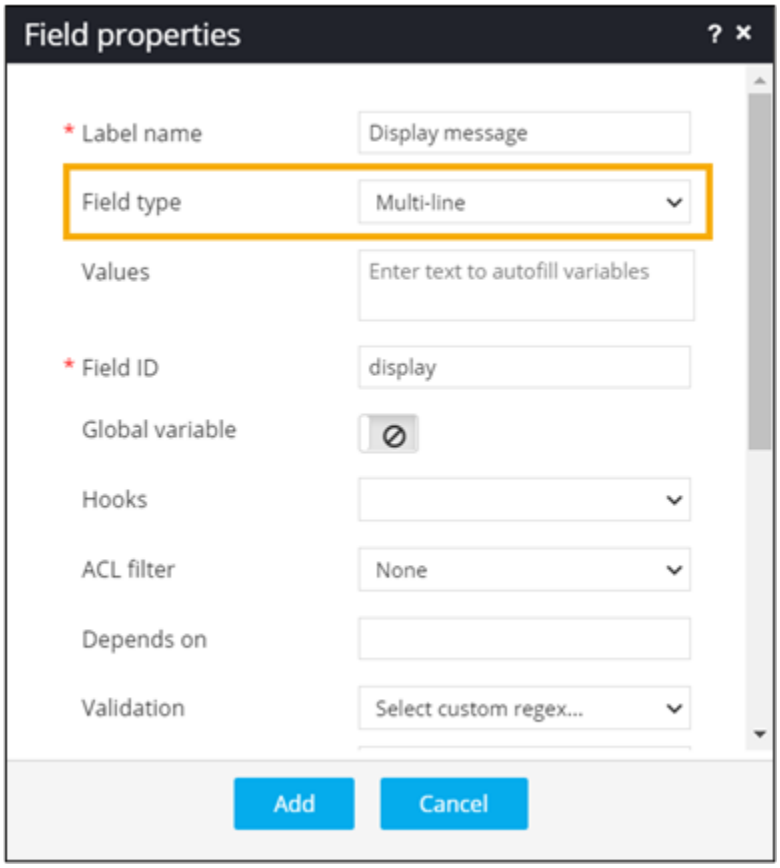


Field Type - Multi-line

This field type in the form builder allows you to capture multiple lines of data within a form.

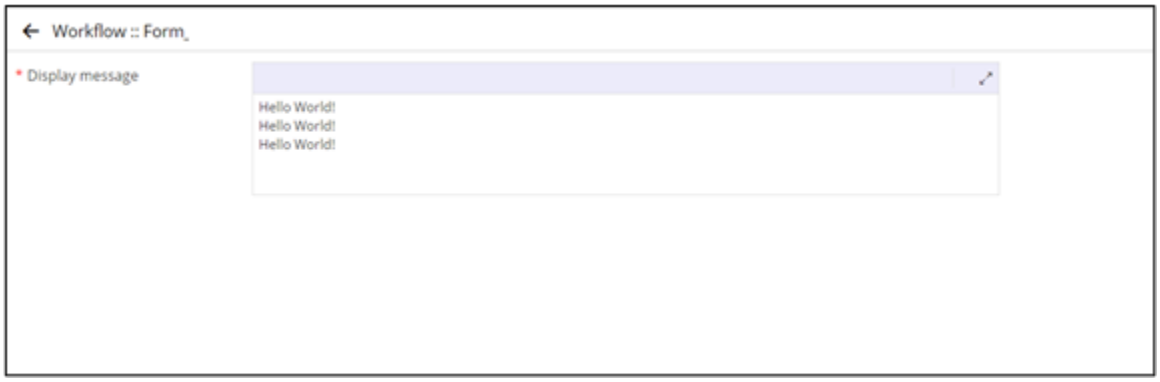
1. To add a **Multi-line** field to the form, under **Form builder**, click + in the command bar.
2. In the **Field properties** window, enter or select the field information as shown.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



3. To add the Multi-line field to the form, click **Add**.
4. To see this field added to the form, click .

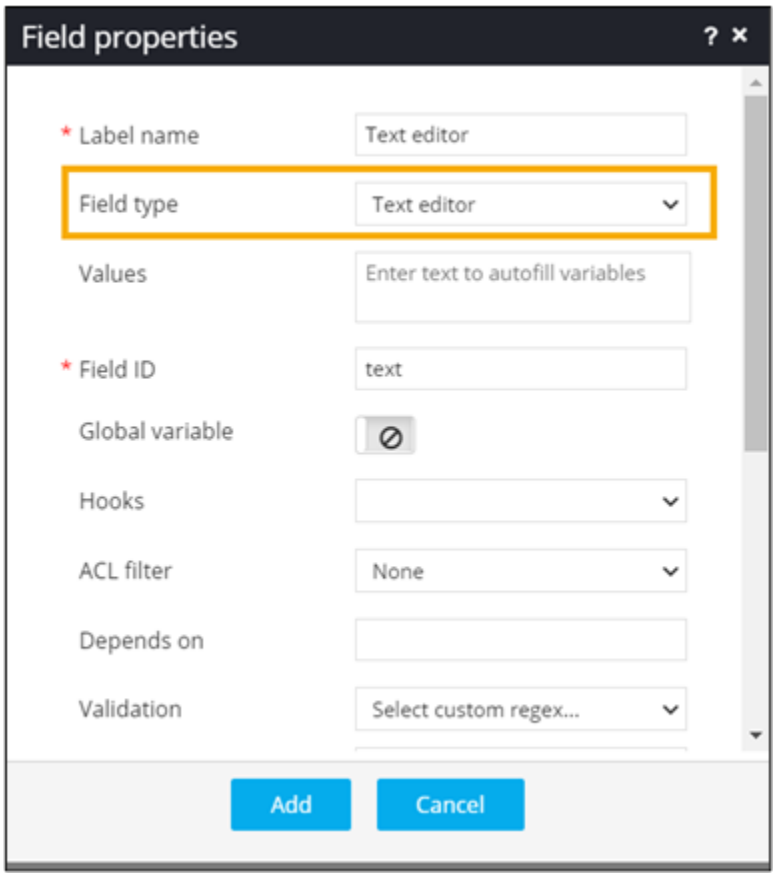
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Field Type - Text editor

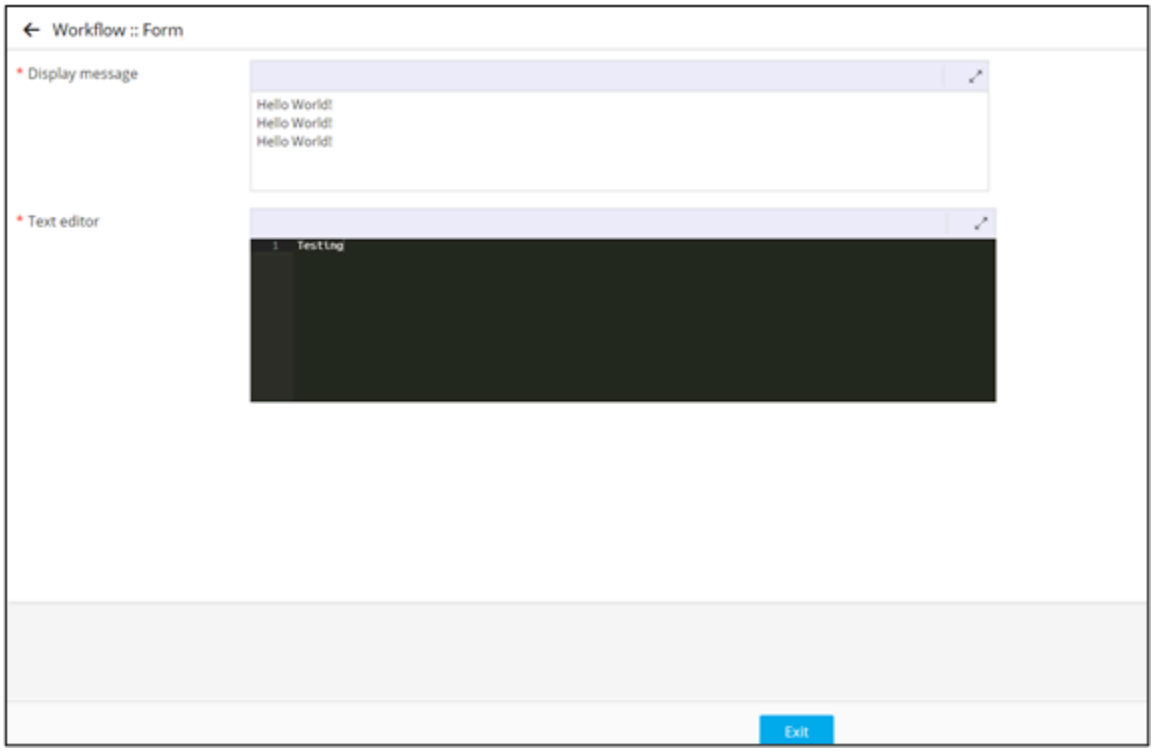
This field type allows you to add a text editor to their form.

1. To add a **Text editor** field to the form, under **Form builder**, click + in the command bar.
2. In the **Field properties** window, enter or select information as shown.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

3. To add the form field, click **Add**.
4. To see the field added to the form, click .

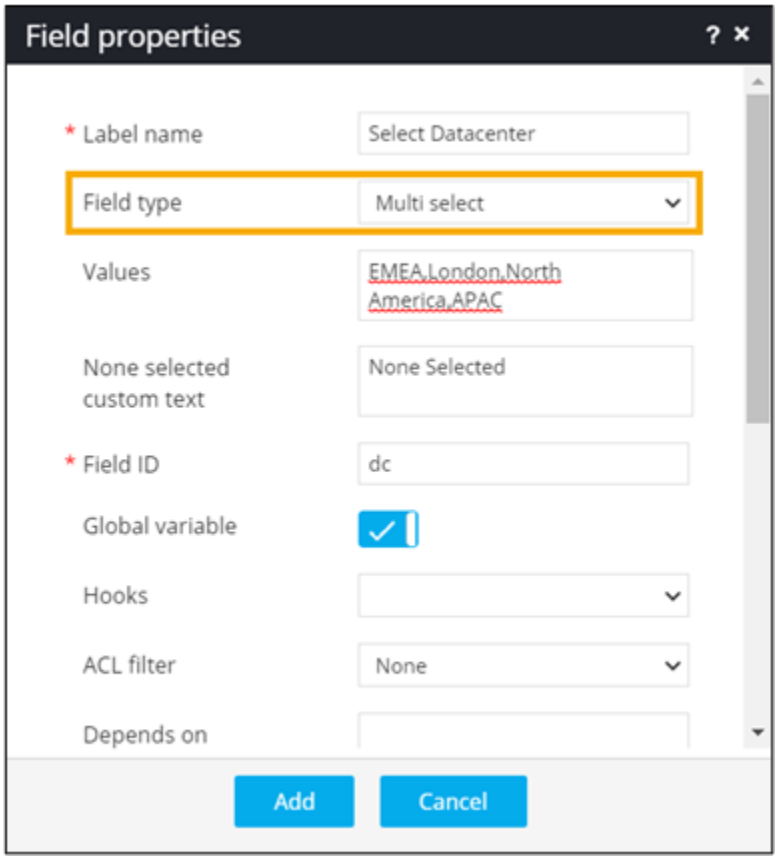


Field Type - Multi select

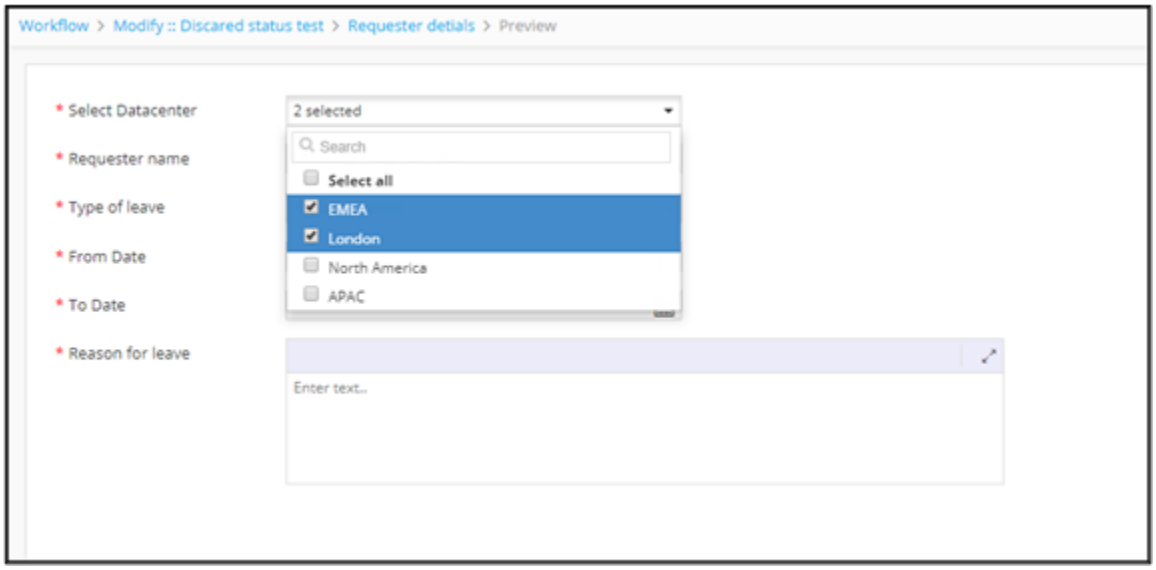
This field type in the form builder allows you to search and select multiple values from a specific dropdown form field within the request form.

1. To add a **Multi-select** field to the form, under **Form builder**, click + in the command bar.
2. In the **Field properties** window, enter or select field information as shown here.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



3. To add this field to the form, click **Add**.
4. To see the field added to the form, click .

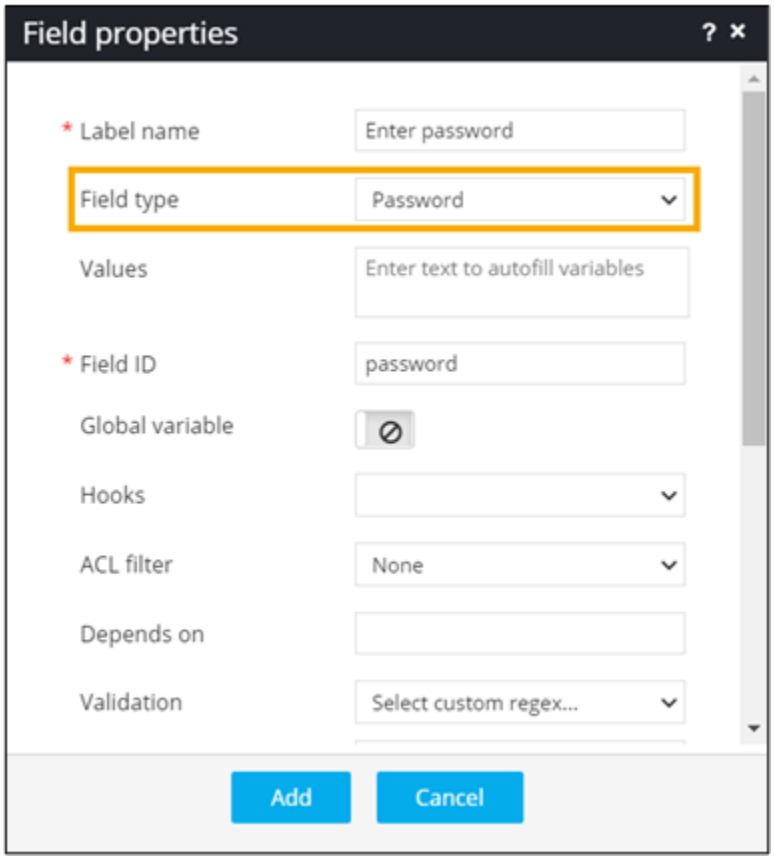


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Field Type - Password

This field type in the form builder allows you to enter passwords if necessary.

1. To add a **Password** field to the form, under **Form builder**, click + in the command bar.
2. In the **Field properties** window, enter or select the field information as shown here.



3. To add this field to the form, click **Add**.
4. To see the field added to the form, click .

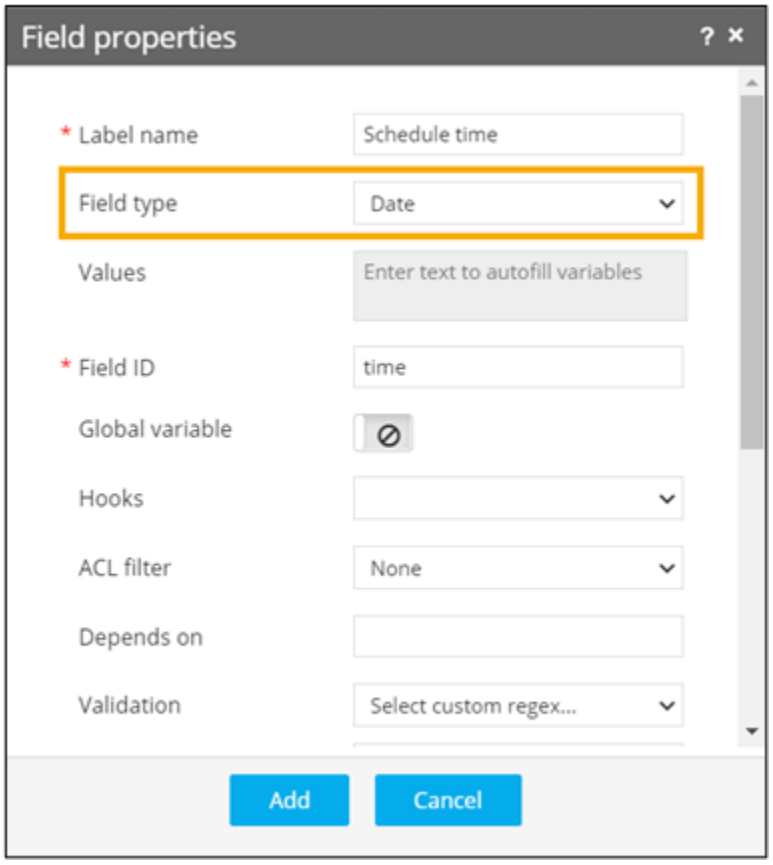
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Field Type - Date

You can add this field type to your form to choose a specific date and time for scheduling tasks within the form.

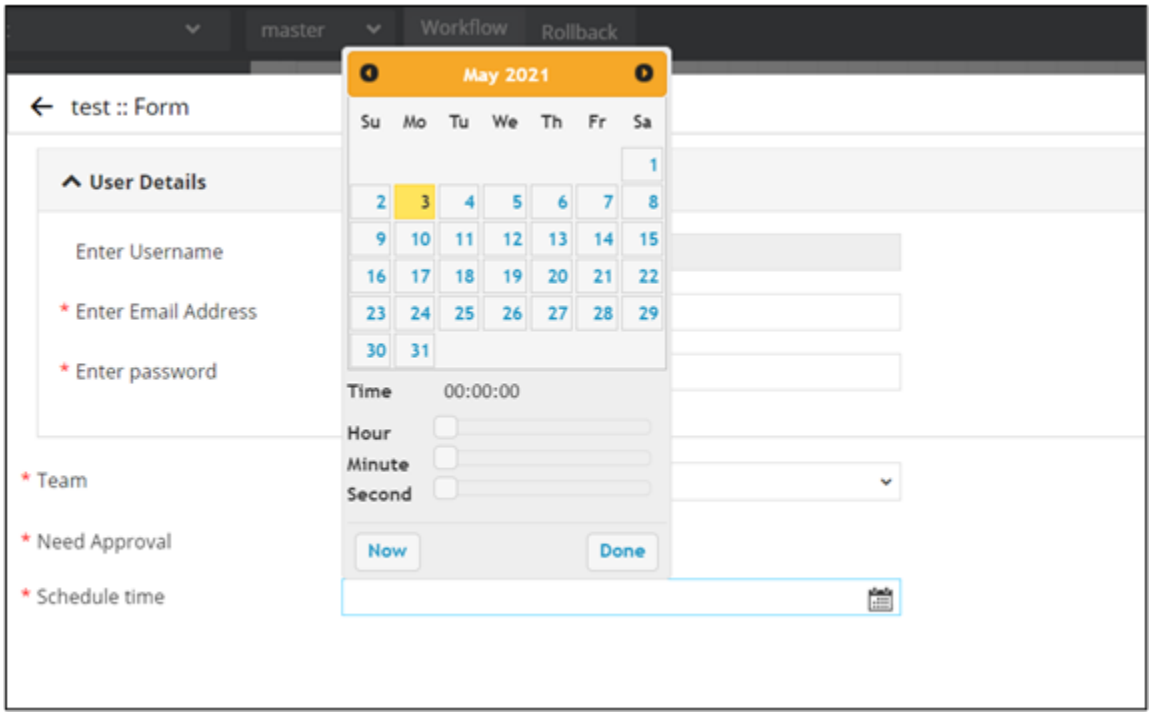
1. To add a **Date** field to the form, under **Form builder**, click + in the command bar.
2. In the **Field properties** window, enter or select the field information as shown here.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



3. Click **Add**.
4. To see the field added to the form, click .

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



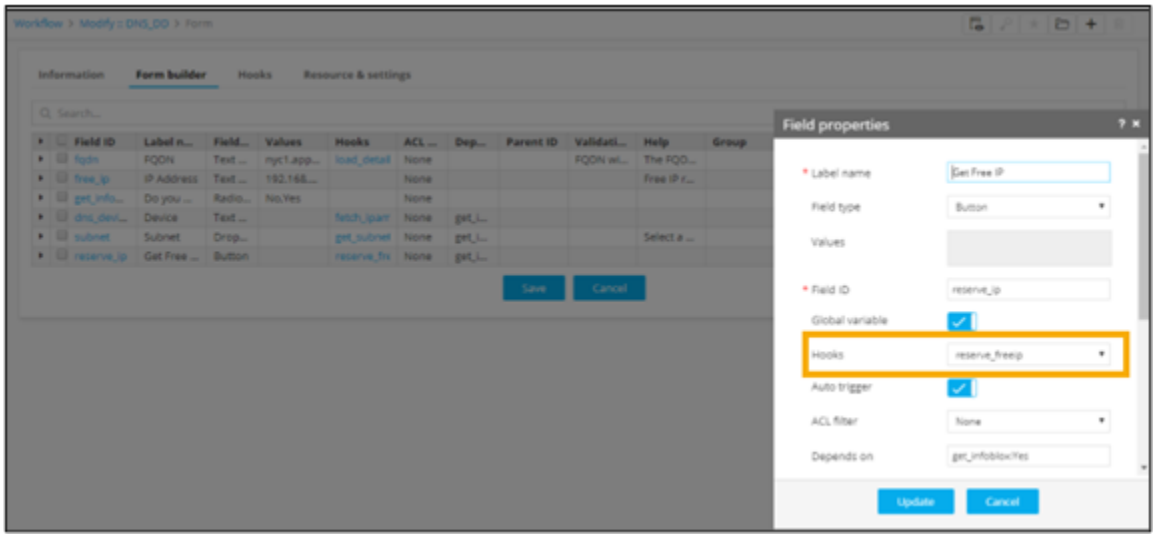
Field Type - Button

You can define custom buttons with label names using the button field. Button fields can be used to retrieve specific details from the device, database or used to perform an action by communicating to an external system via API.

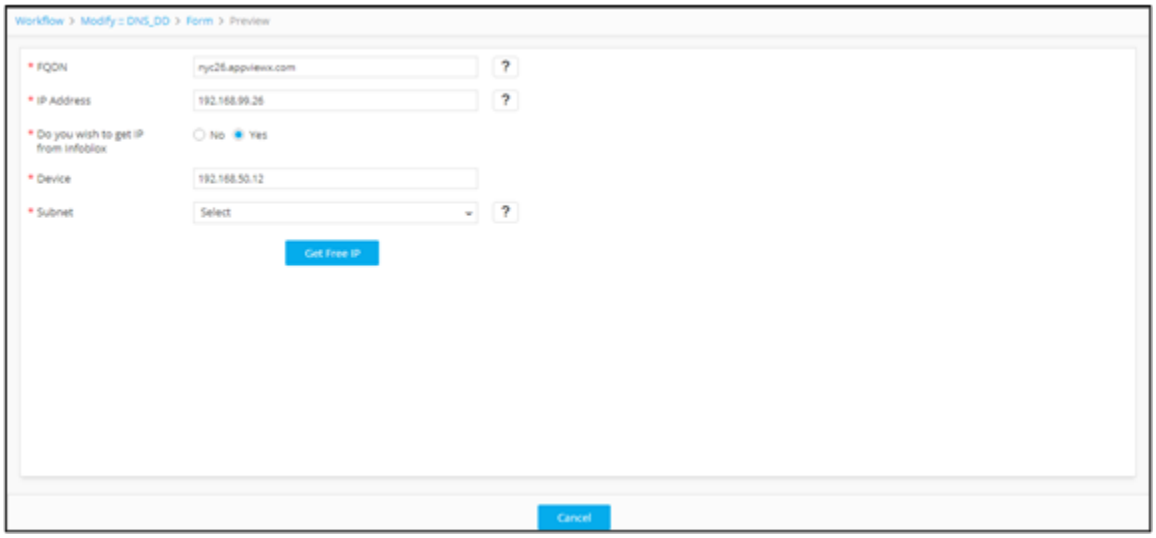
A relevant associate script must be mapped against the button field for it to perform a specific operation or redirect the output to a dependent field.

1. To add a **Button** to the form, under **Form builder**, click + in the command bar.
2. In the **Field properties** window, define a form field with field type as **Button** with the respective Field ID.
3. Map the relevant associate script that needs to be invoked on clicking the button.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



4. To see the button added to the form, click .

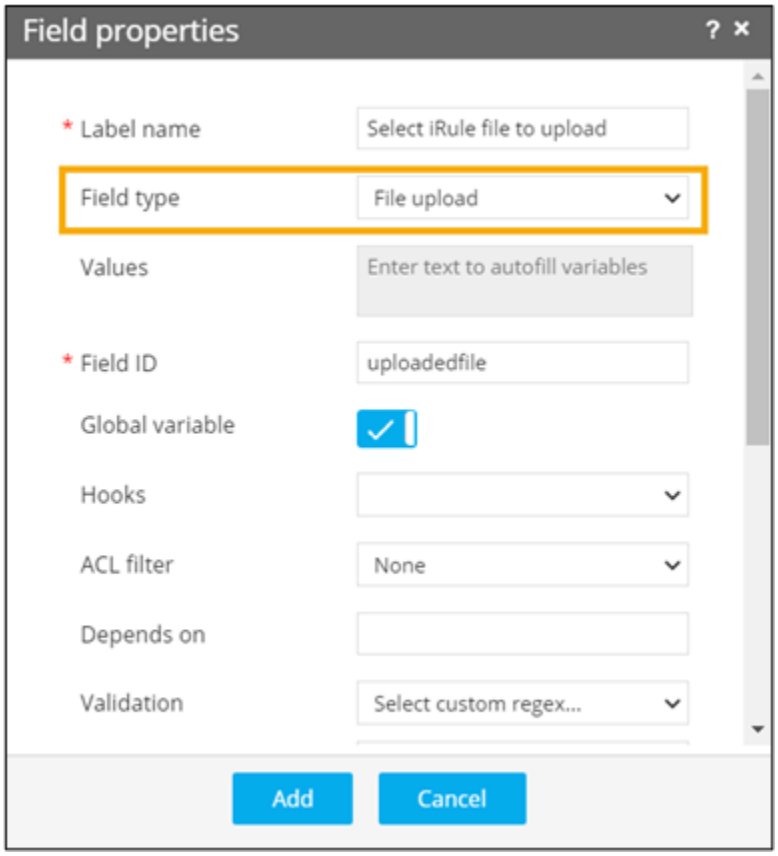


Field Type - File upload

This form field allows you to add a file upload element to your form. You can upload files of maximum 2 MB in .txt, .pem, .crt, .csv formats.

1. To add a **File Upload** field to the form, under **Form builder**, click in the command bar.
2. In the **Field properties** window, enter or select the field information as shown here.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



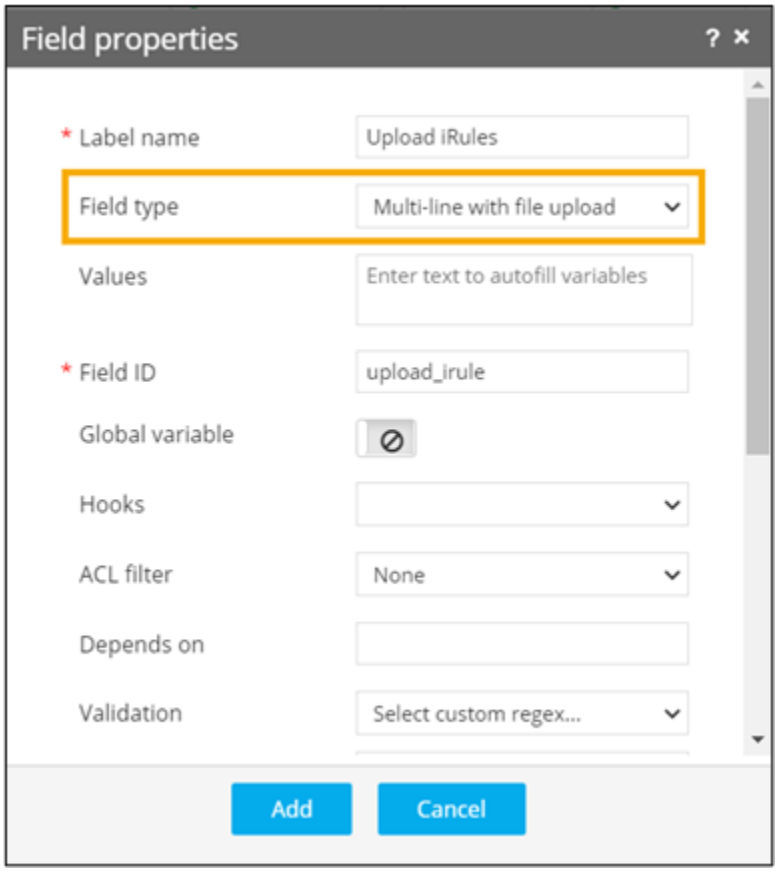
3. To add this field to the form, click **Add**.
4. To see the field added to the form, click .

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Field Type - Multi-line with file upload

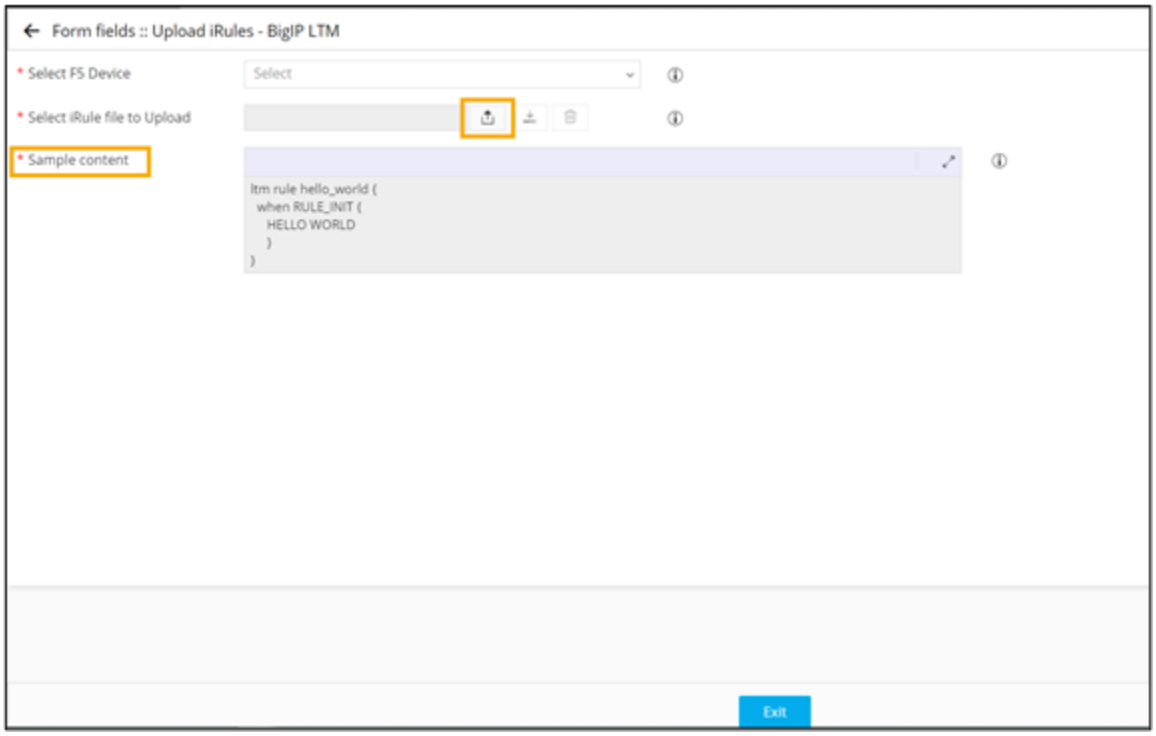
This form field allows you to use the 'file upload' element with a multi-line element and convert them into configurations in order to push to the end device. Used to push file(s) between destinations – local to remote, remote to remote. It supports .txt, .pem, .crt, .csv files and allows maximum file size of 2 MB.

1. To add a **Multi-line with file upload** field to the form, under **Form builder**, click + in the command bar.
2. In the **Field properties** window, enter or select the field information as shown.



3. To add this field to the form, click **Add**.
4. To see the field added to the form, click .

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

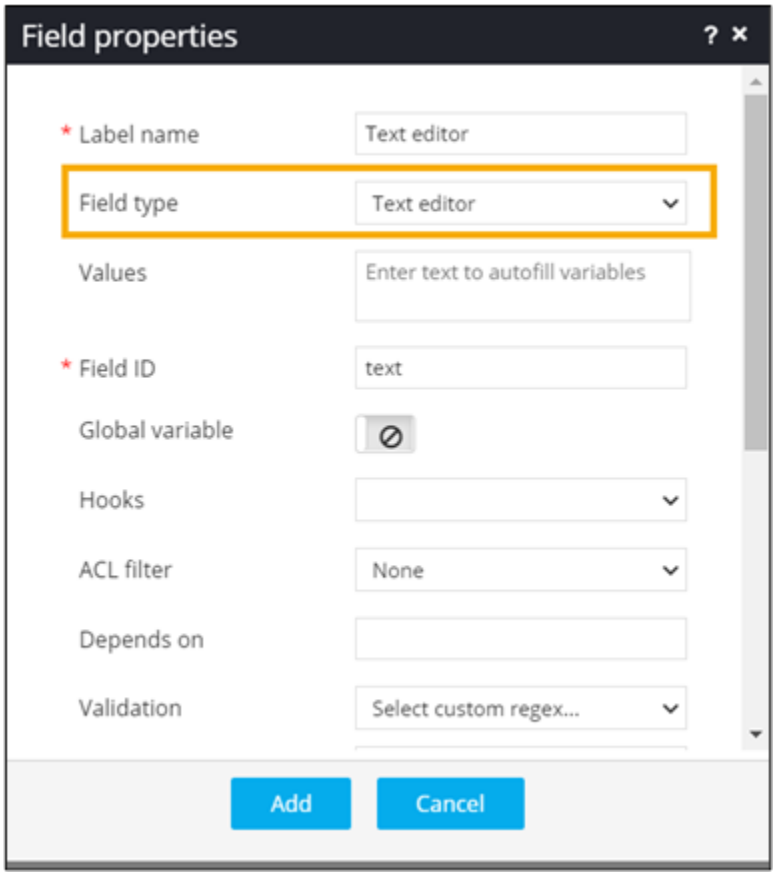


Field Type - Text editor with file upload

This form field allows you to use the file upload element with a text editor element and convert them into configurations in order to push to the end device. Used to push file(s) between destinations – local to remote, remote to remote. It supports .txt, .pem, .crt, .csv files and allows maximum file size of 2 MB.

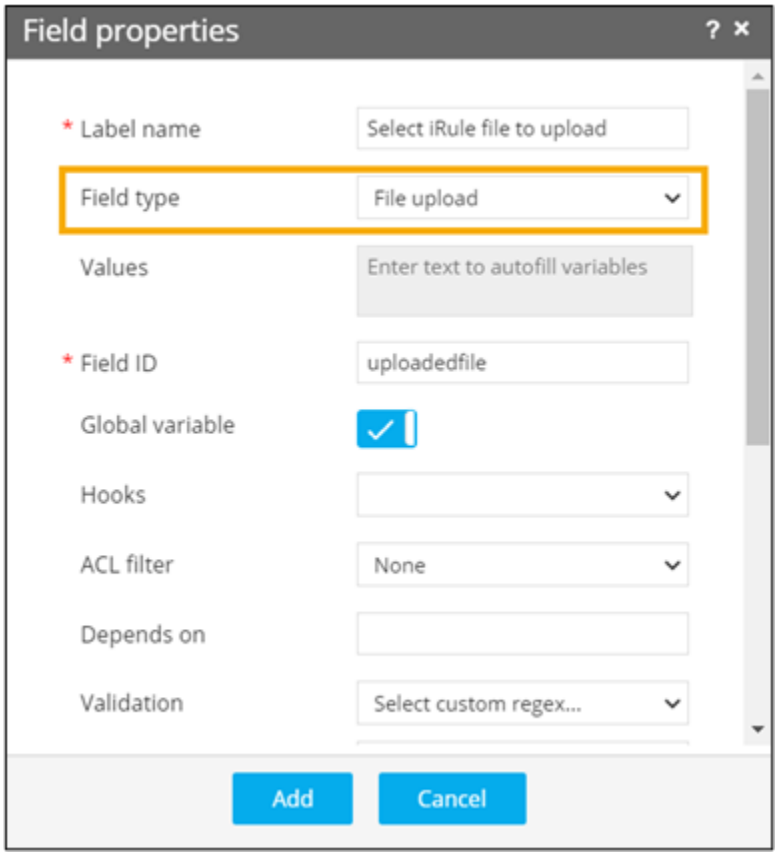
1. To add a **Text editor with file upload** field to the form, under **Form builder**, click + in the command bar.
2. In the **Field properties** window, enter or select information as shown here.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



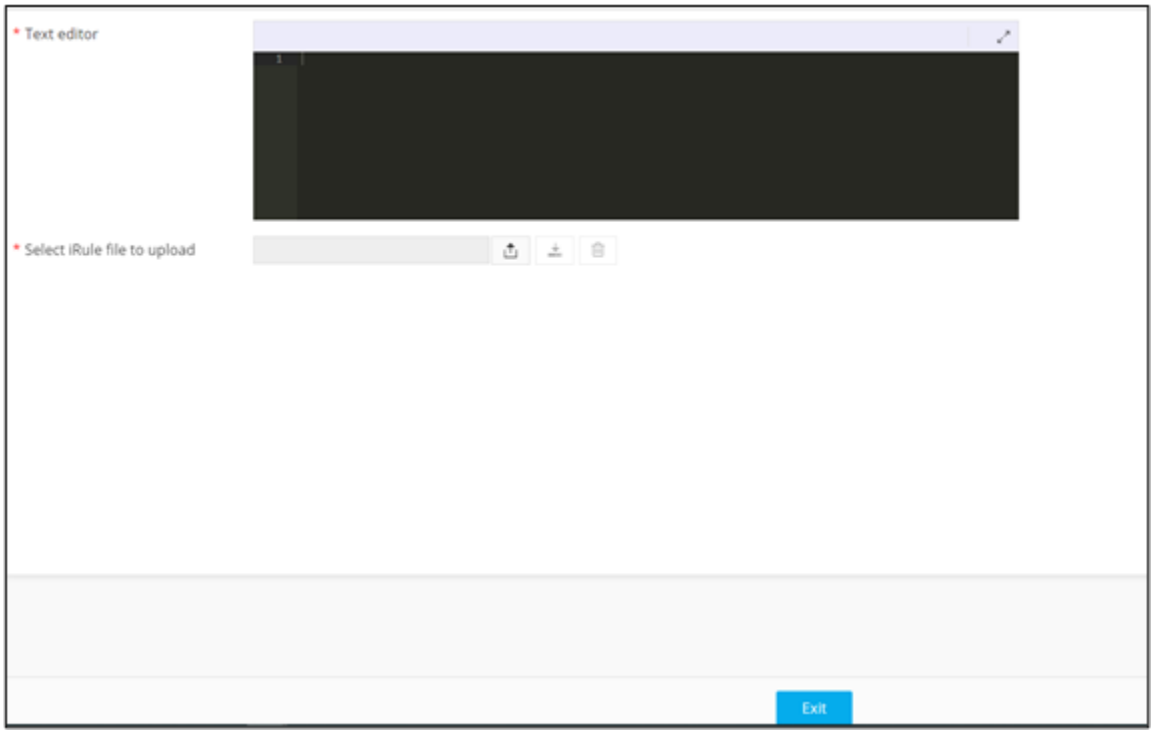
3. To add the form field, click **Add**.
4. Add the **File upload** form field.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



5. To see the field added to the form, click .

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

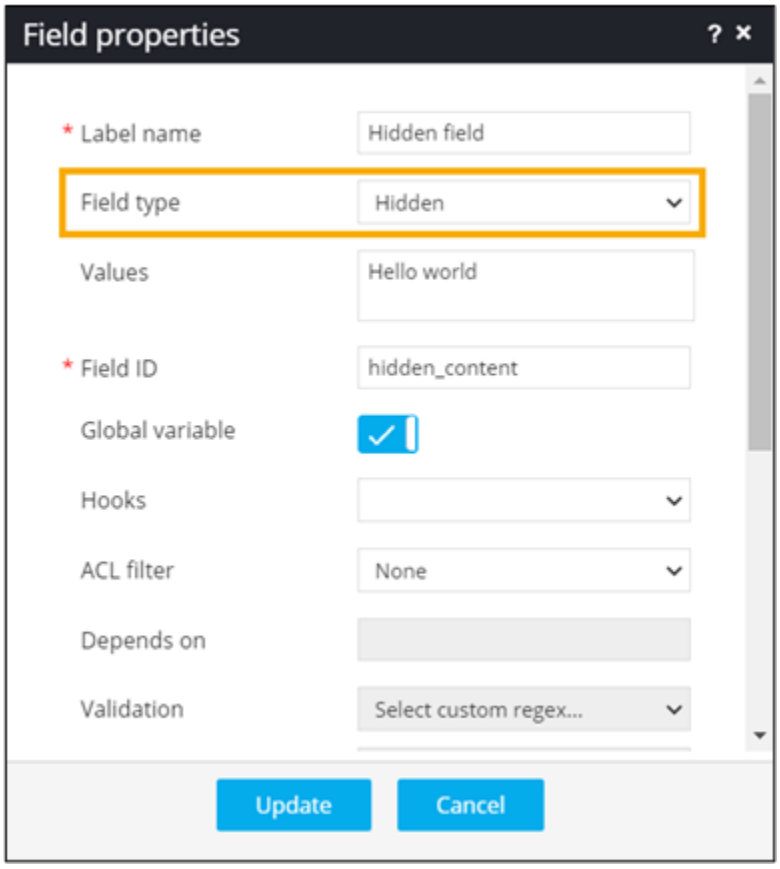


Field Type - Hidden

You can use the Hidden field to hide certain values or content in the form and fetch it only when required.

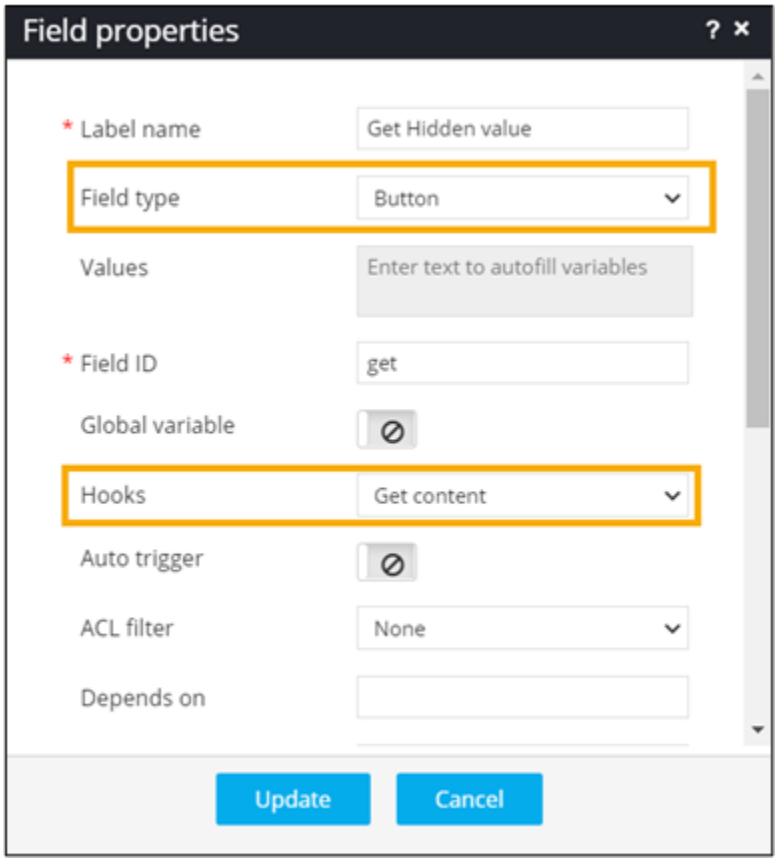
1. To add a **Hidden** field to the form, under **Form builder**, click + in the command bar.
2. In the **Field properties** window, enter/select the field information as shown here.

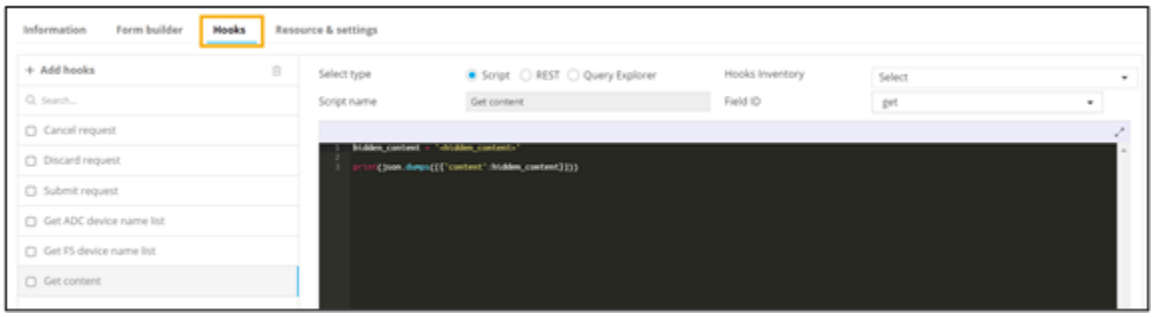
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



3. To fetch hidden data, add the **Button** field and define the relevant hook.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

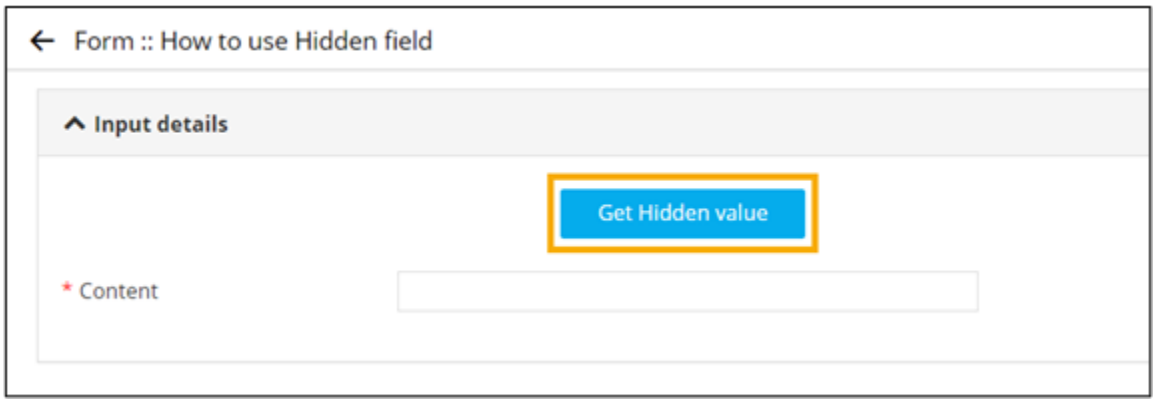


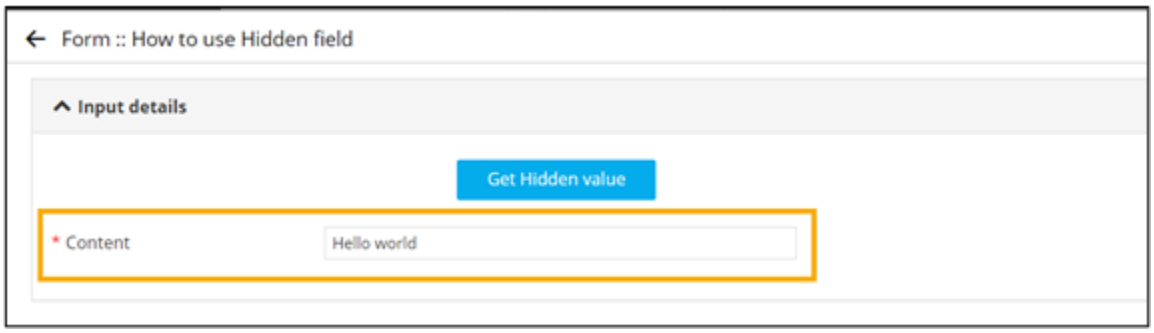


4. Add a **Text box** field.
5. To see the field added to the form, click .

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

6. In the form preview, click **Get Hidden value**.



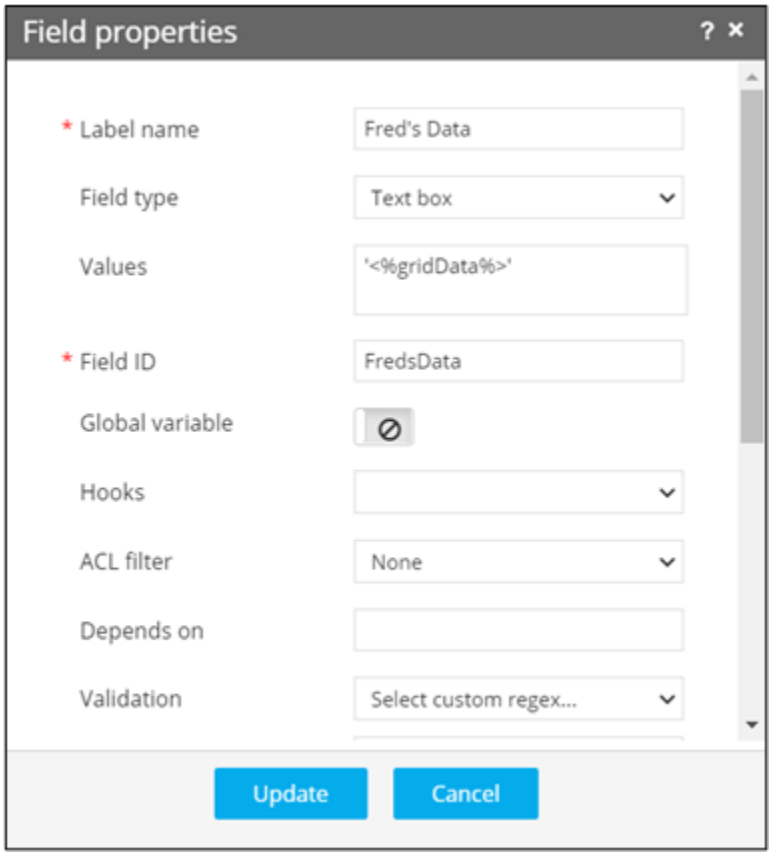


Field Type - Download CSV

This form field allows you to download the data displayed in a grid to your machine as a .csv file.

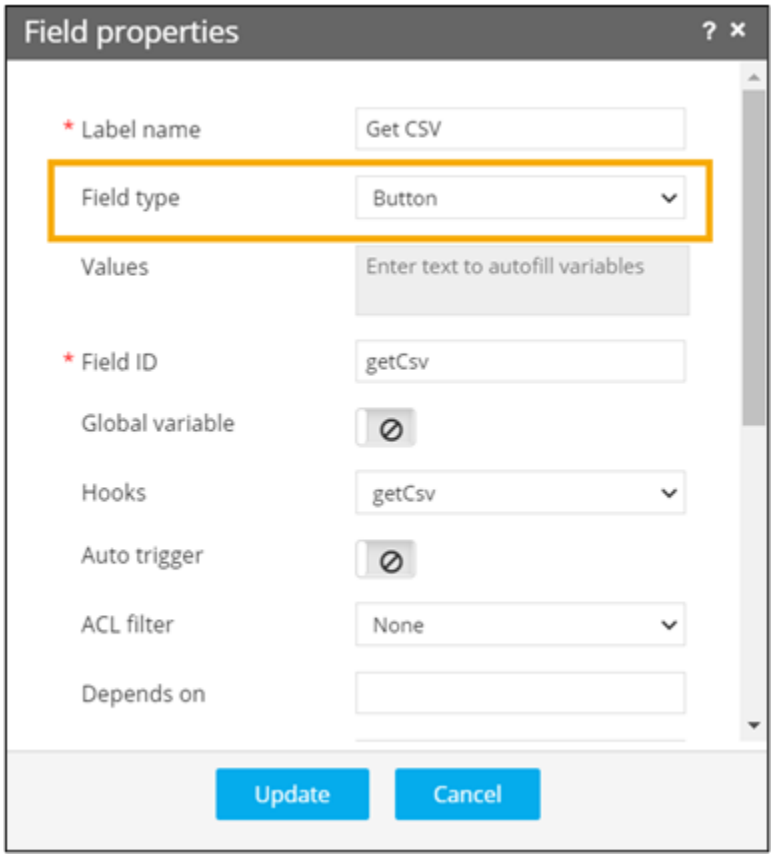
1. To add a **Download CSV** button to the form, under **Form builder**, click + in the command bar.
2. In the **Field properties** window, enter or select the information as shown.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

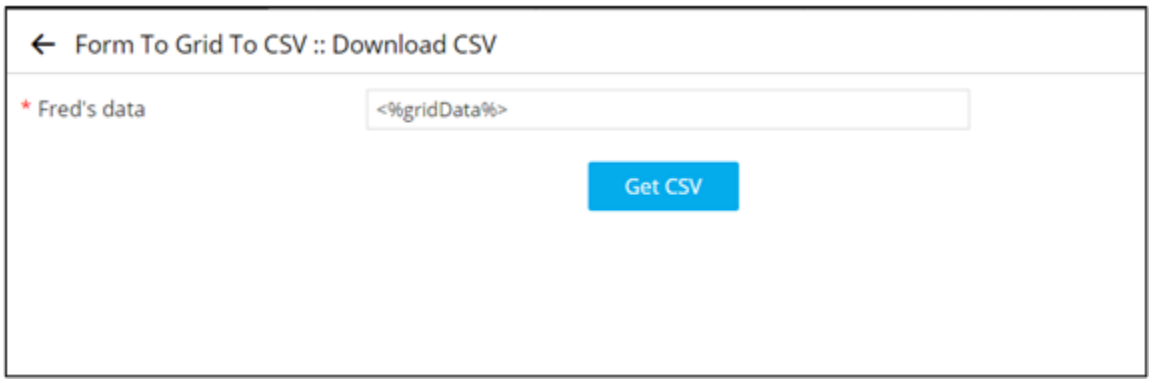


3. Add another form field (Get CSV) with field type as **Button**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



4. To see the field added to the form, click .



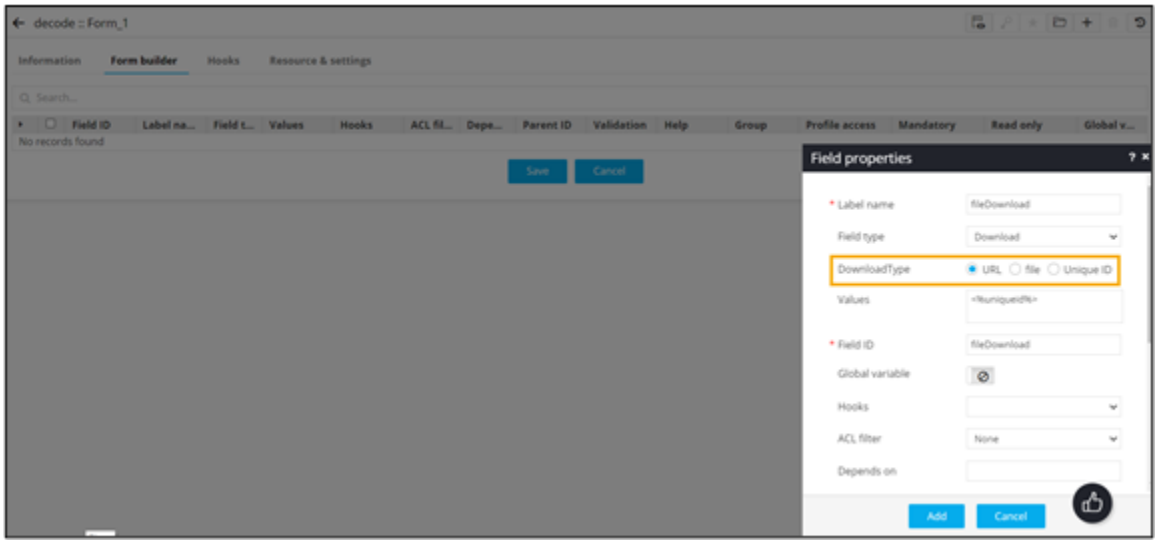
Field Type - Download

This form field allows you to download the data from the file to your machine.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Provision to mention the unique file ID that is used to retrieve files that are stored within the database as part of visual workflow upload file API.
- Provision to choose the Decode file option to retrieve the saved encoded file data.

1. To add a **Download** field, enter or select the field information as shown. The **Download Type** field displays three options:
   - **URL**
   - **File**
   - **Unique ID**

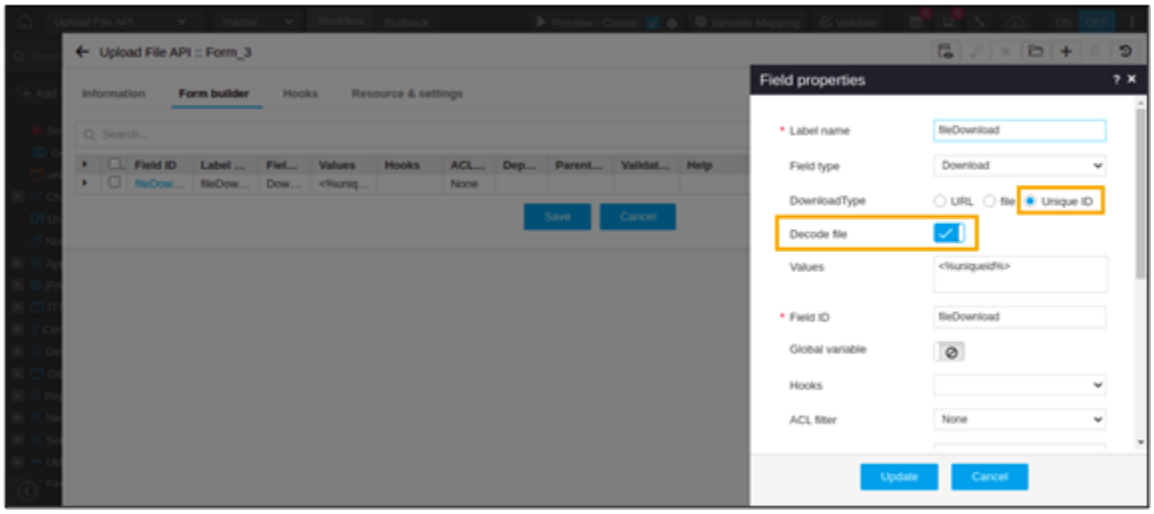


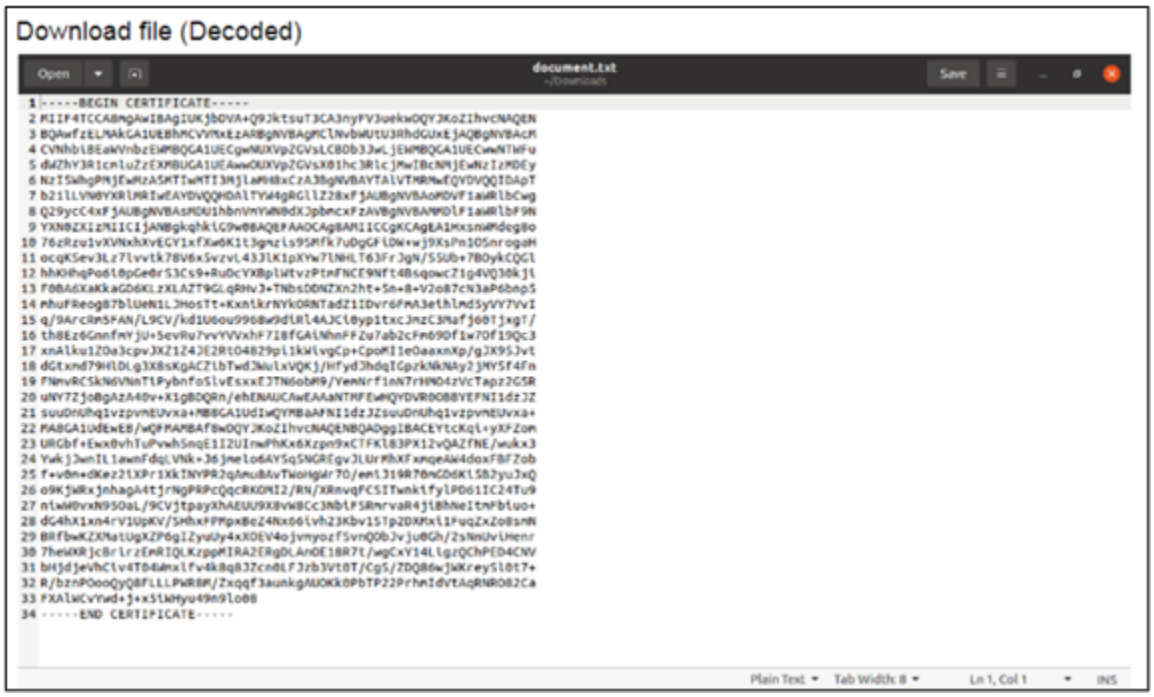
2. Select the **Unique ID** option to display the **Decode** file field.

> **Note:** The Decode file field is an optional field and is disabled by default.

3. To enable the **Decode file** option, turn on the toggle.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



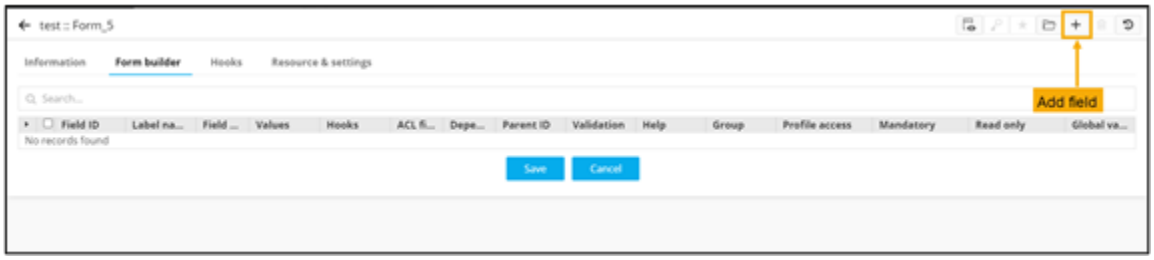


Validating Form Fields

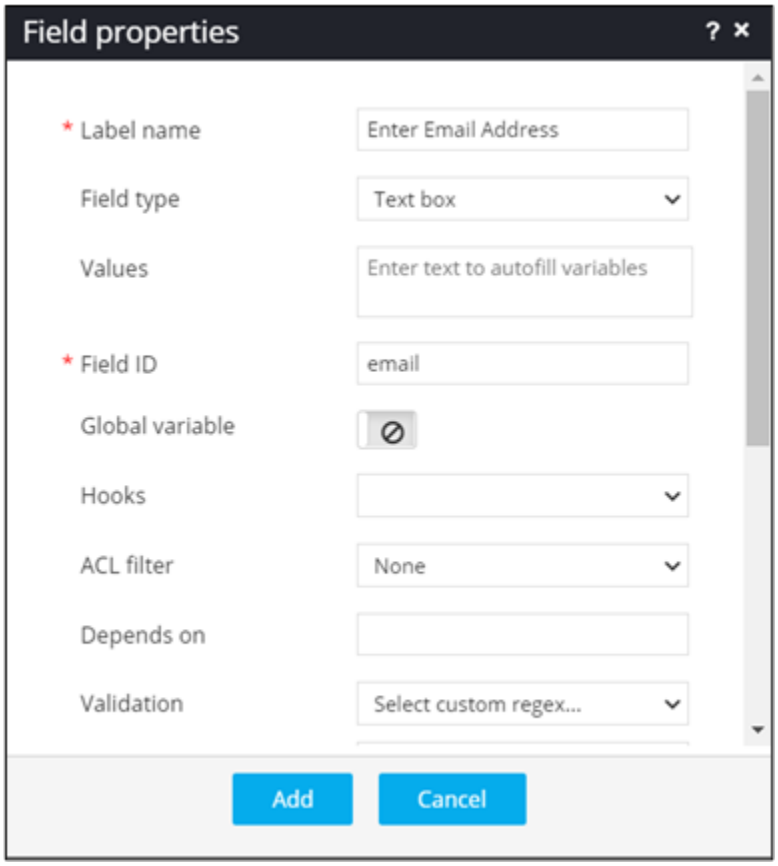
You can define custom regex validation for certain form fields. Regex validations can be referenced against a form field with specific notifications to the user. You can add or edit any regex in the inventory.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. Design a workflow.
2. From the **User Interface** section, drag and drop a **Form** task.
3. To add a form field, in the **Form builder** section, click + in the command bar.

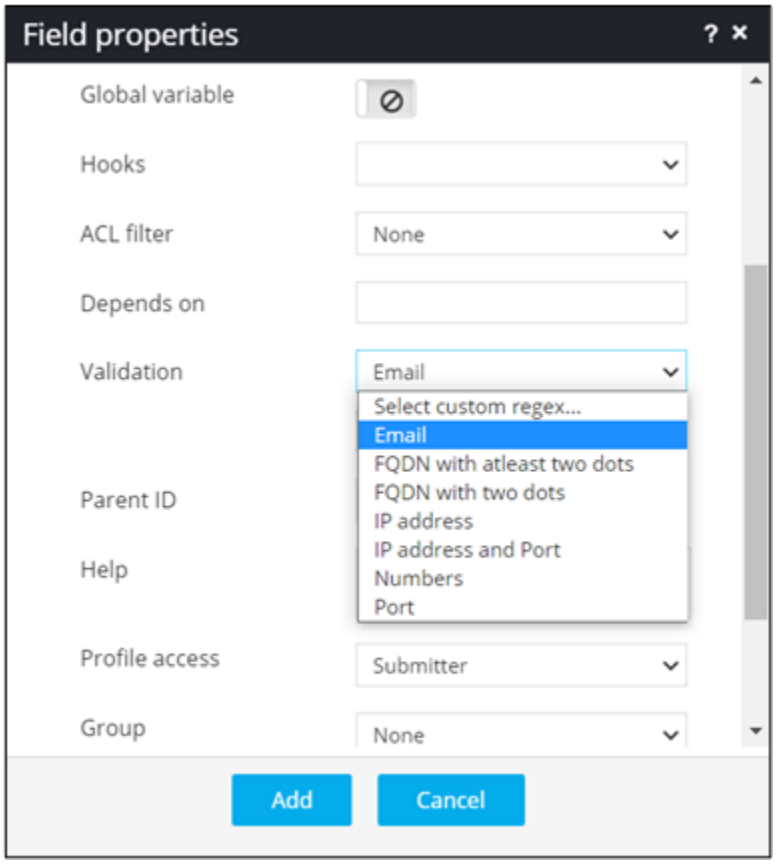


4. In the **Field properties** window, enter or select the field information.



5. From the prebuilt regex patterns, select **Email**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Note: For more information, refer to the section on Regex Library.

6. To add this form field to the form, click **Add**.
7. To see the field added to the form, click .

Only values that match the defined regex are accepted.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

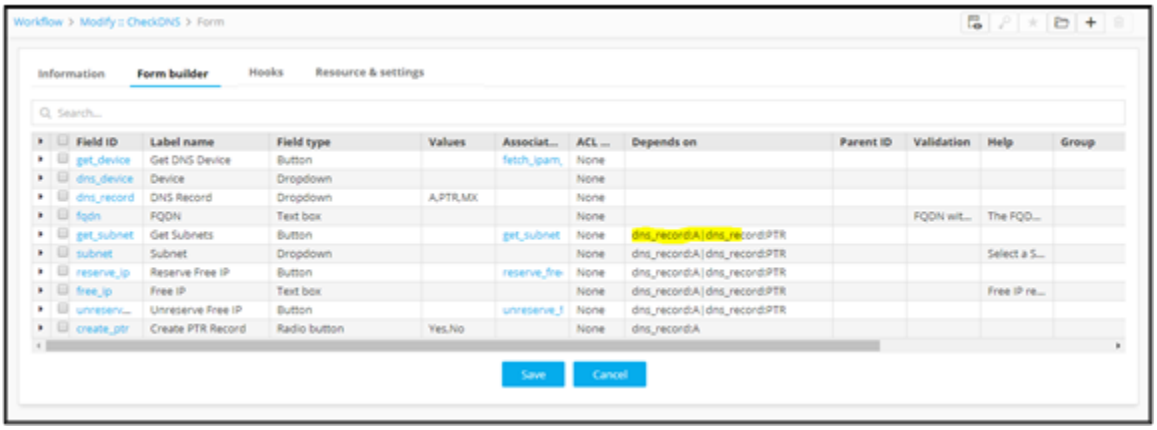
Note: A is displayed next to the form field if the entered value does not match the defined regex.

Defining Dependencies between Form Fields

In order to make the request form dynamic, the form builder allows you to define dependencies between multiple form fields using conditional operators. This can be achieved using the 'Depends On' field through a combination of 'Field ID' and 'Values' parameters in tandem with either the 'logical AND' (&); and 'logical OR' (|) operators.

- Operators supported: '&' (AND); and '|' (OR)
- Defining dependency based on single field and value: [Field ID: Value]
- Defining dependency based on multiple fields and values: [Field ID1: Value1 & Field ID2: Value1], [Field ID1: Value1 | Field ID2: Value1]

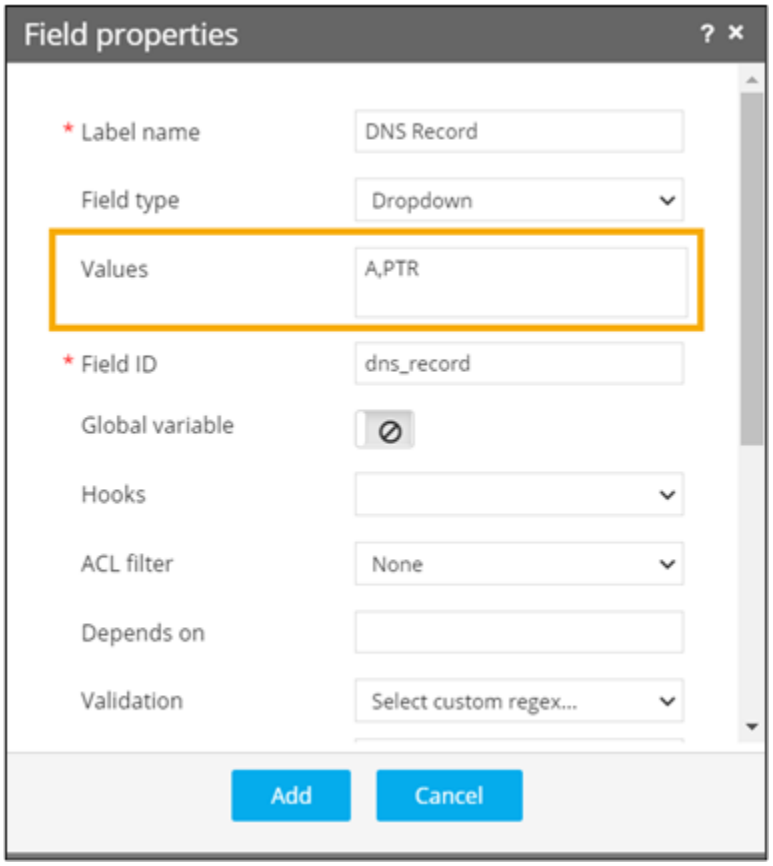
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



To be able to display the Create PTR Record field based on the type of selection of the DNS Record:

1. Design a workflow.
2. From the **User Interface** section, drag and drop a **Form** task.
3. To add form field(s), in the **Form builder** section, click + in the command bar.
4. In the **Field properties** window, define a field called DNS Record with a dropdown that shows two values to select from - A and PTR.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

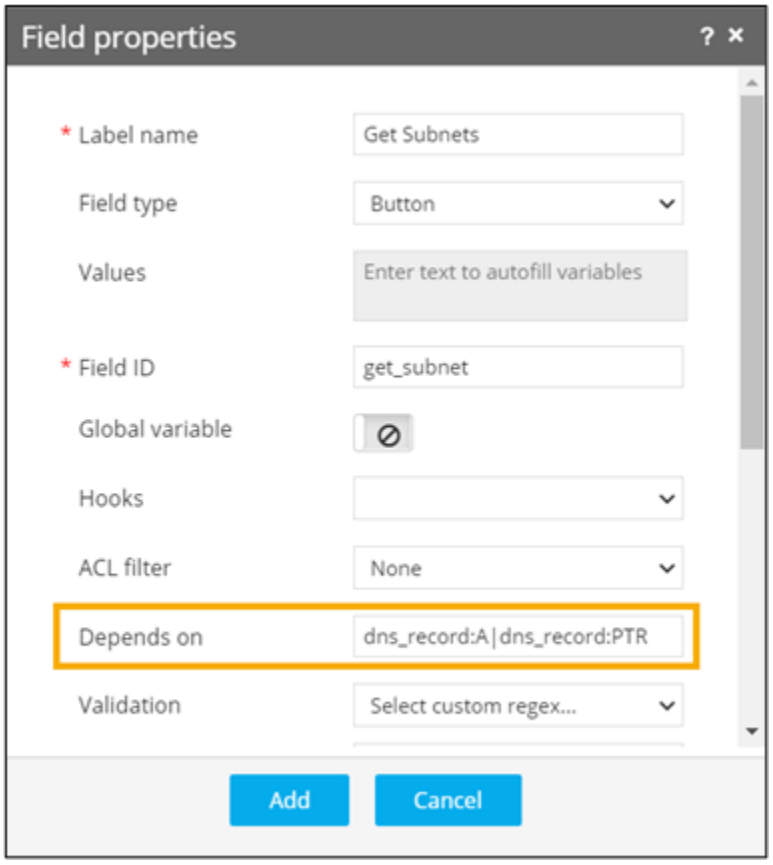


5. Define the parameters for the conditional form field - Create PTR Record based on which this field must be shown.

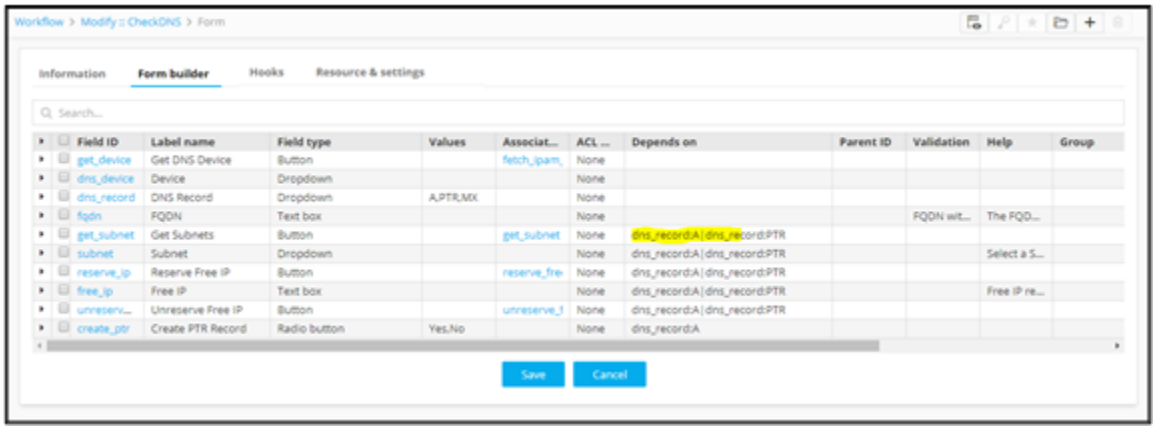
   dns_record:A|dns_record:PTR

   The field Get Subnets will be displayed according to the selection of the DNS record as A or PTR.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

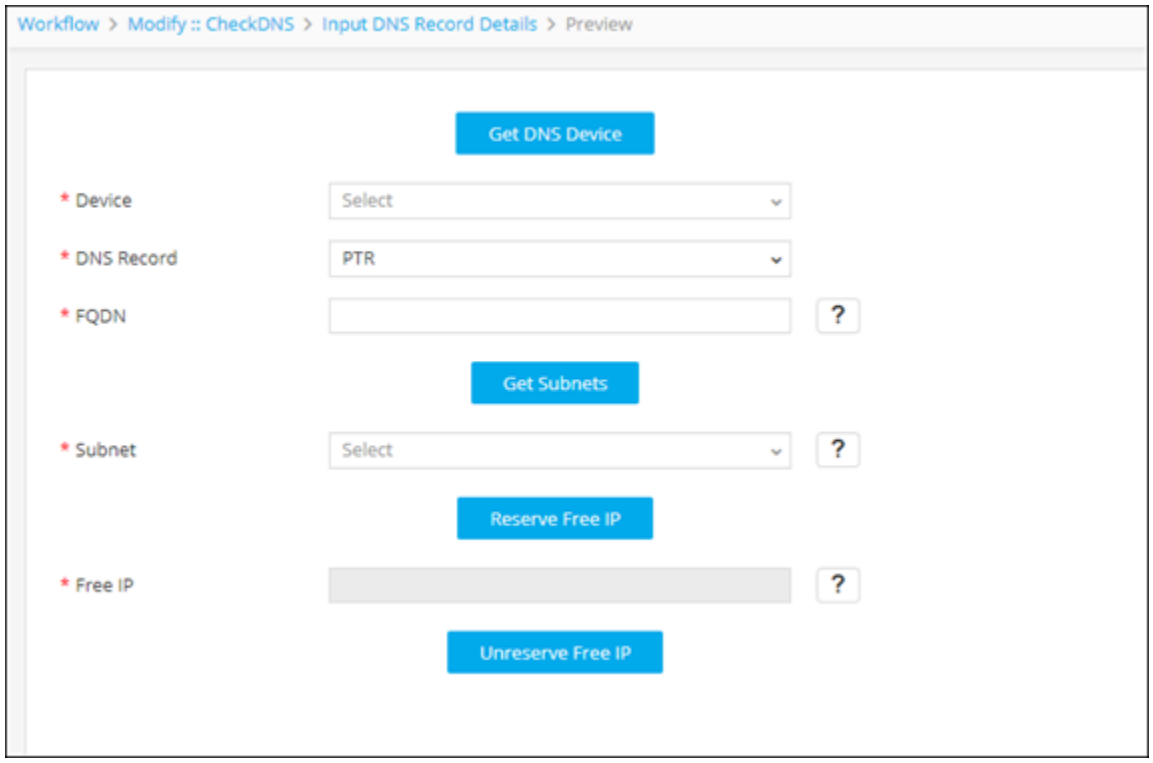


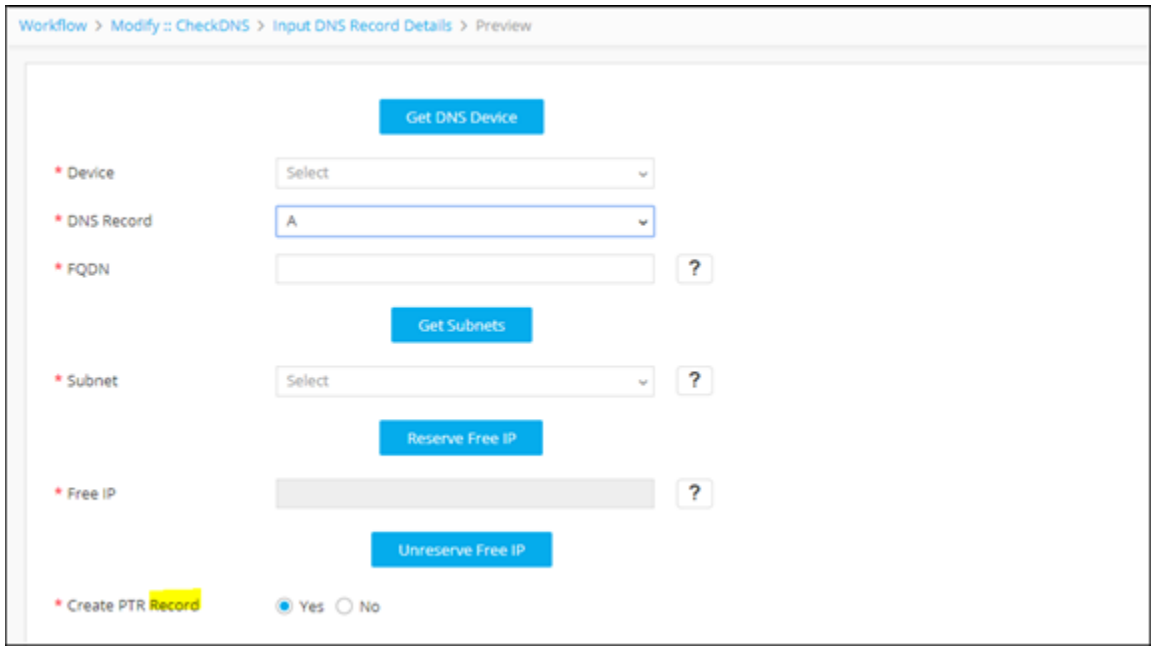
6. Add other form fields and define dependencies as required.



7. To see the form fields added to the form, click .

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

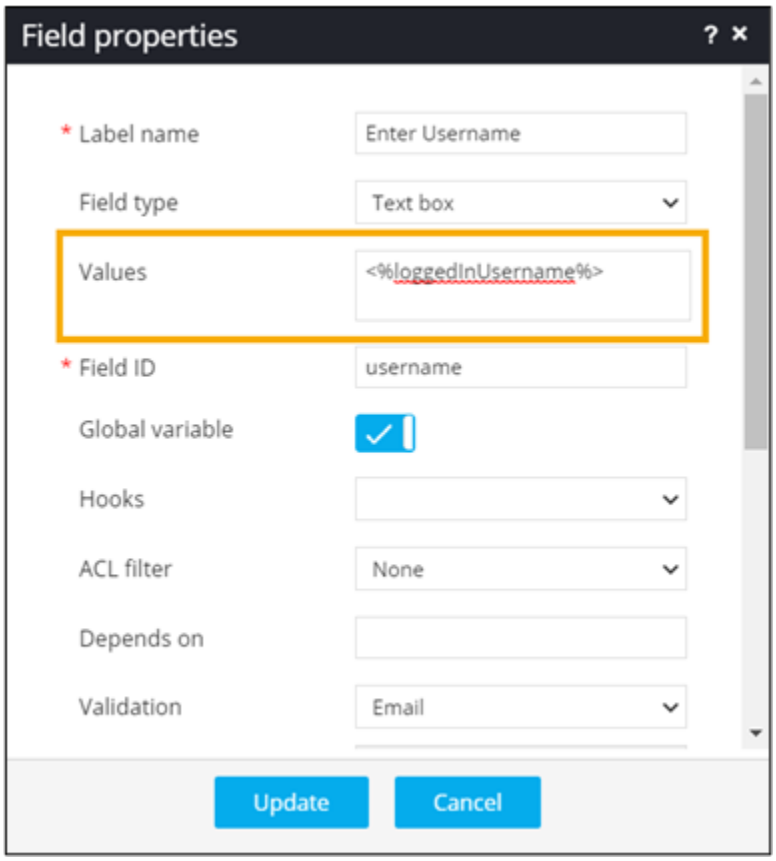




Copyright © 2020 AppViewX, Inc. All Rights Reserved.

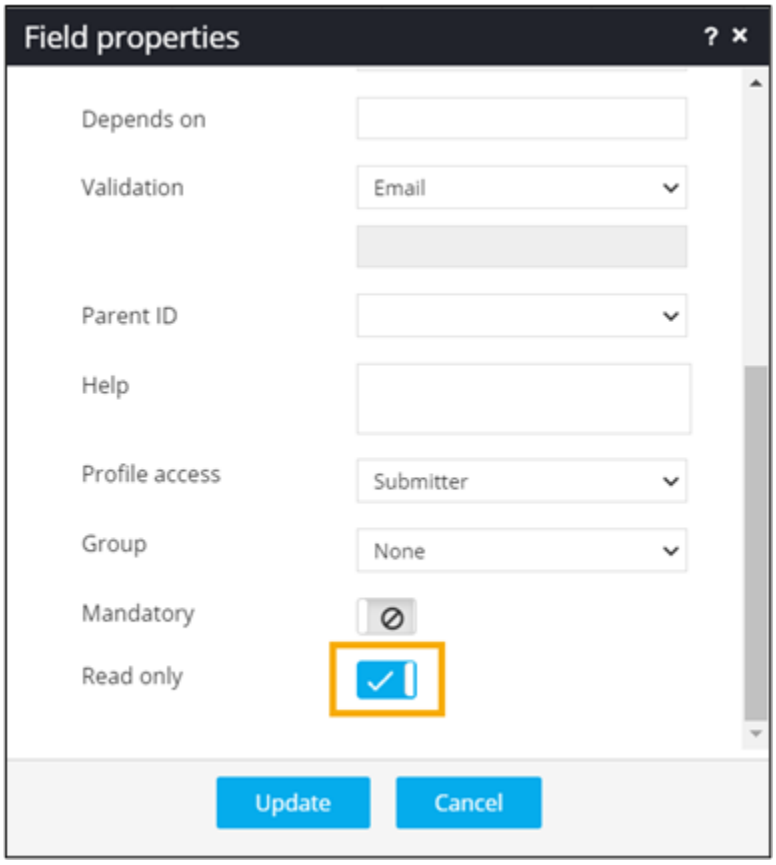
Defining Tasks as Read Only

1. Design a workflow.
2. From the **User Interface** section, drag and drop a **Form** task.
3. To add form fields, click + in the command bar.
4. In the **Field properties** window, enter or select the field information to define the form field.

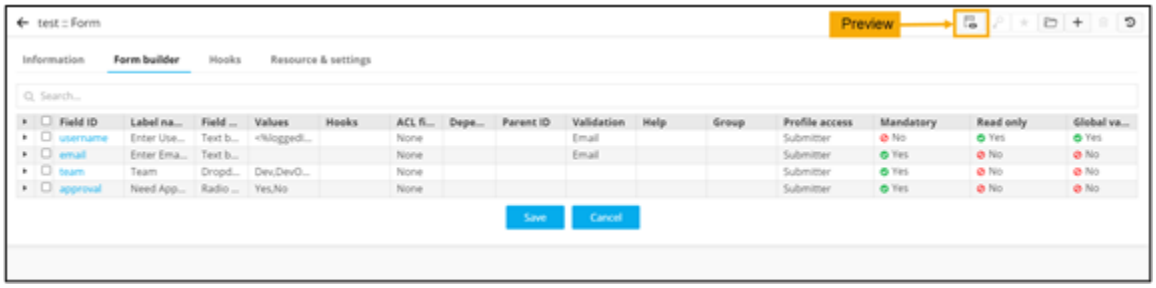


5. To define the field as Read Only, turn on the toggle.

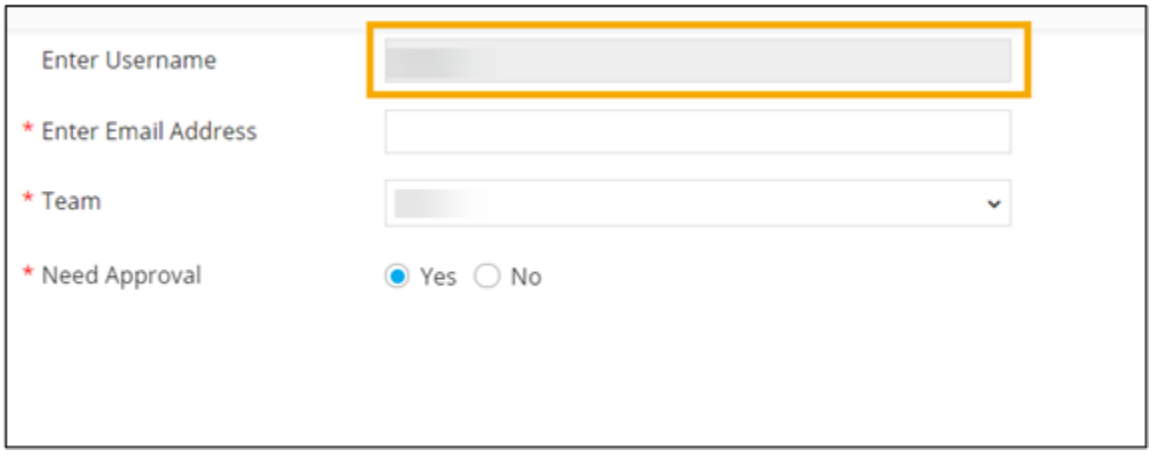
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



6. To see the Read only username field added to the form, click .



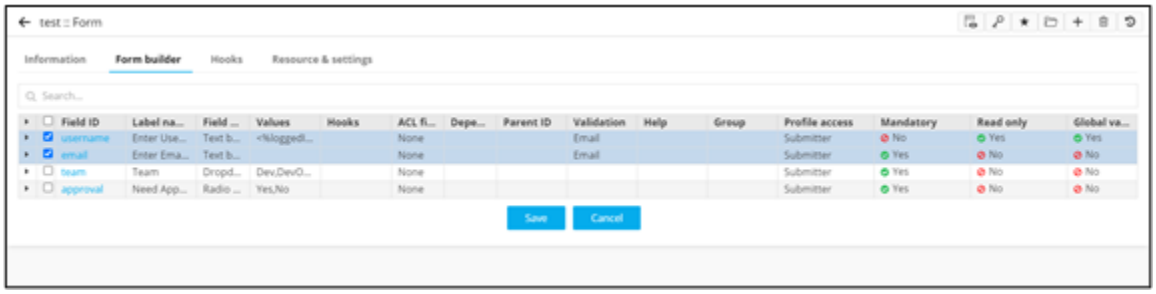
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



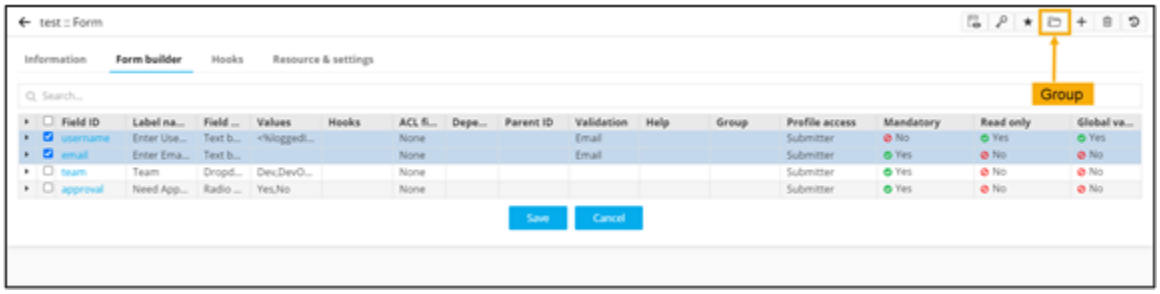
Grouping Multiple Form Fields

You can group multiple fields logically within the form.

1. Design a workflow.
2. From the **User Interface** section, drag and drop a **Form** task.
3. Under **Form builder**, select the form fields to be grouped.

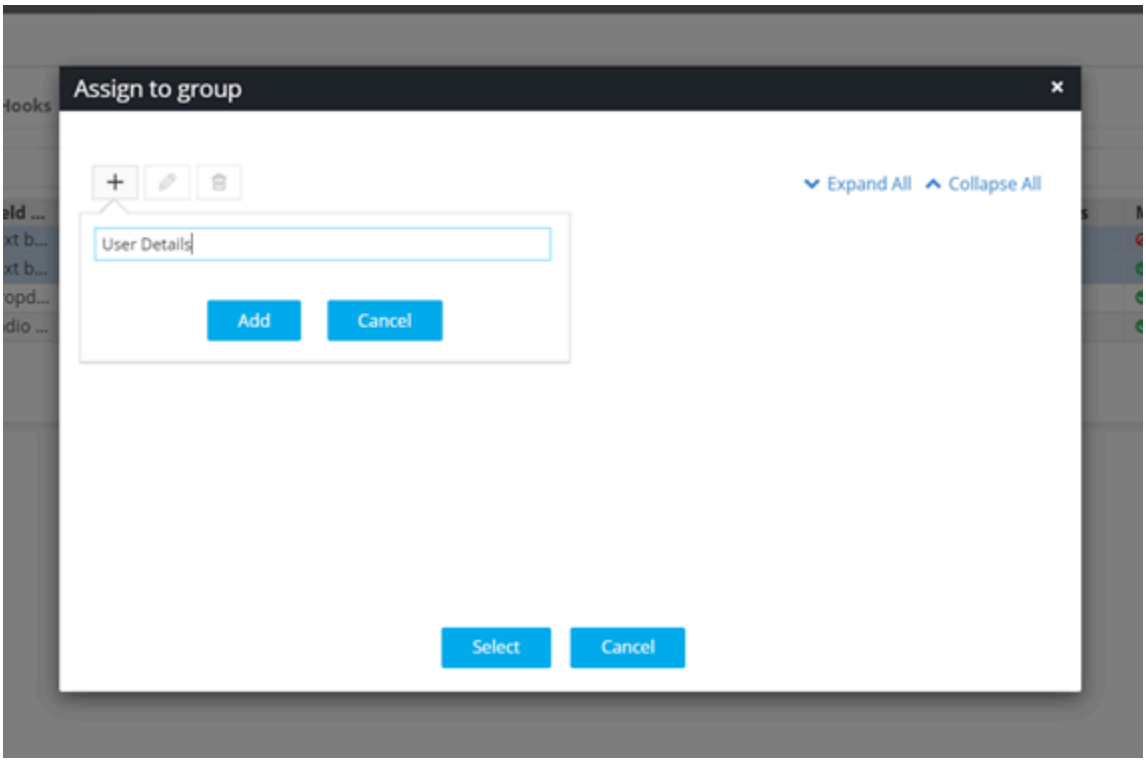


4. To group the selected form fields, click .

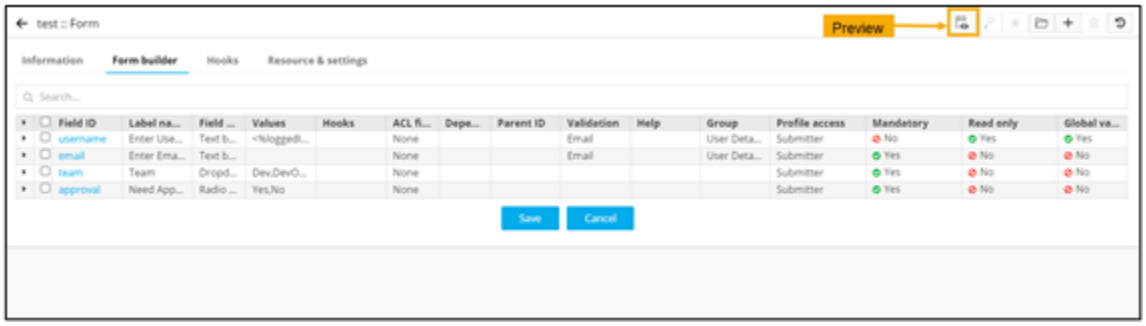


5. To create a new group, in the **Assign to group** window, click .

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



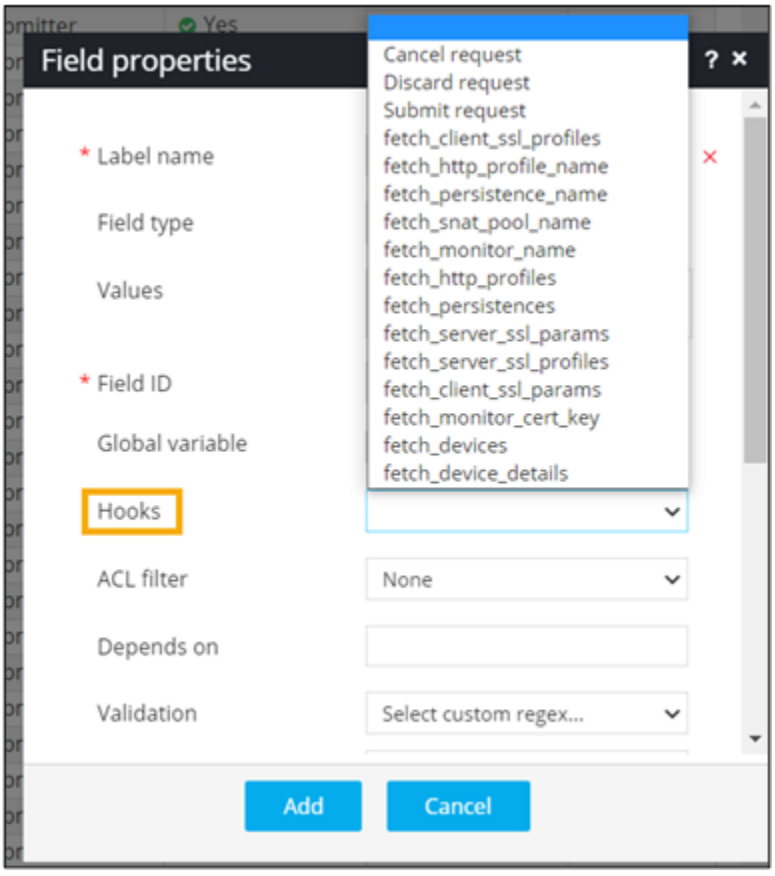
6. To see the form fields listed under the assigned group, click .



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Using Hooks in a Form

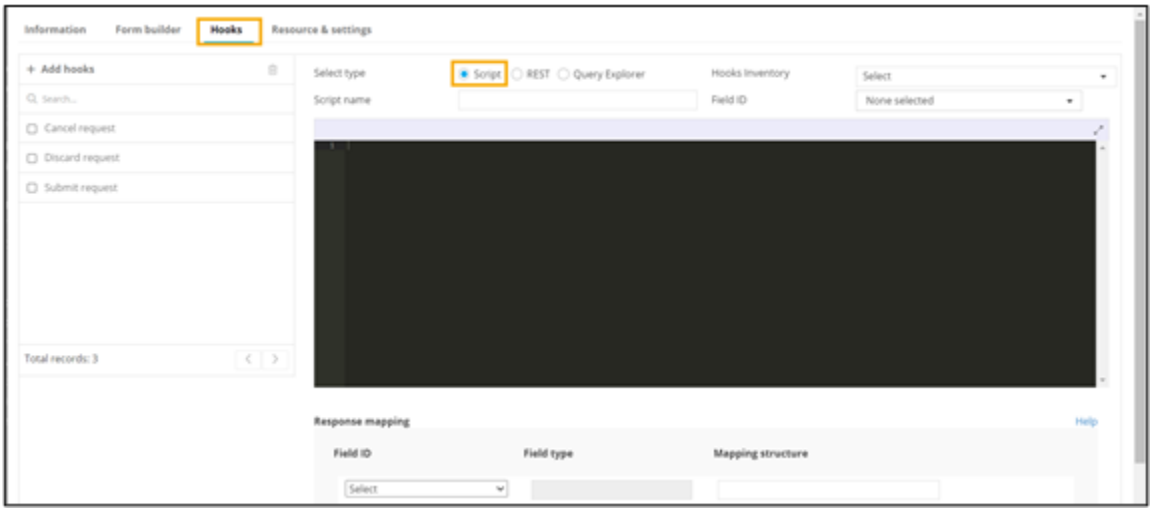
You can fetch values dynamically from a data source such as a database, device or external systems by using hooks. You can either select hooks from the existing list or create a new hook.



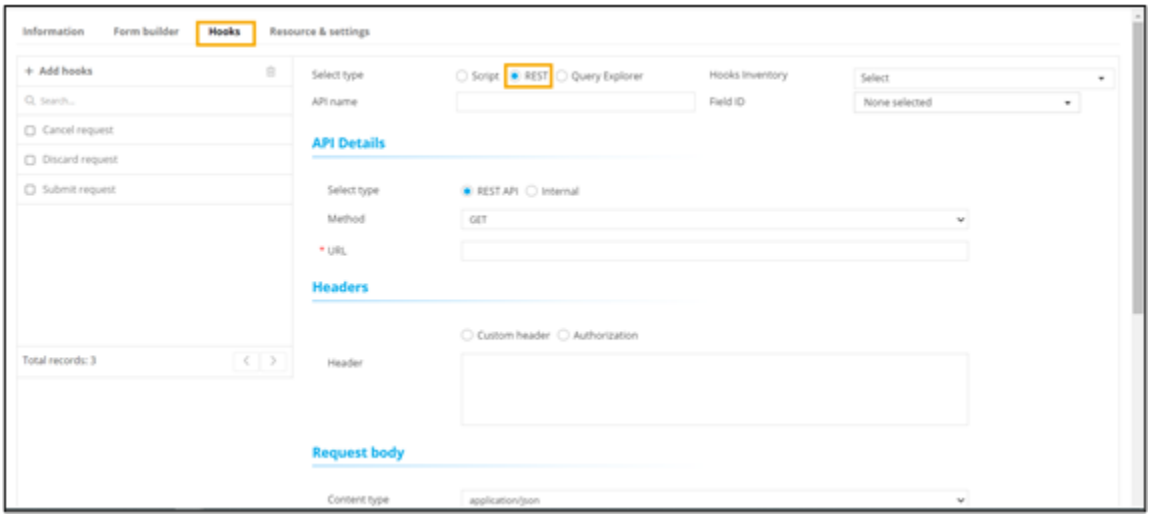
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Hooks are of three types:

- **Script**: The form builder allows you to define one or more ‘associate scripts’ and map them against the relevant form field. The associate script acts as an abstract layer which enables linking of the data retrieved from the query and maps them against appropriate fields within the form.

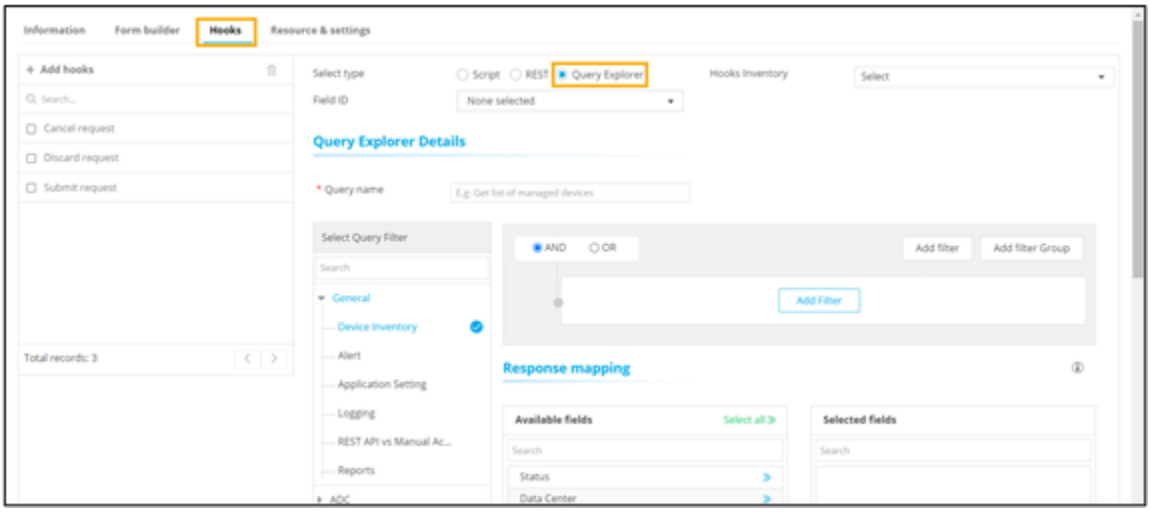


- **REST API** (Internal & External API): REST API allows querying the database, device or external systems



- **Query Explorer**

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



- Using Scripts with Forms
- Auto-trigger scripts
- Default script for Cancel and Discard Actions
- Define ccustom Notification Messages using Scripts
- Defining Query Explorer to Build Forms

Using Scripts with Forms

You can define scripts within a form and leverage them to retrieve values and populate form fields or print output.

- Triggering Script with Button
- Triggering Script to Get Device List
- Triggering Script to Get Device List by Vendor
- Triggering a Script to Print Custom Message
- Using Auto-trigger to Invoke the Script Automatically based on Input
- Triggering a Script Directly from a Form Field to Map its Values to Destination Fields

Triggering Script with Button

To associate a script to a ‘button’ element, trigger and retrieve values and populate them to a form field:

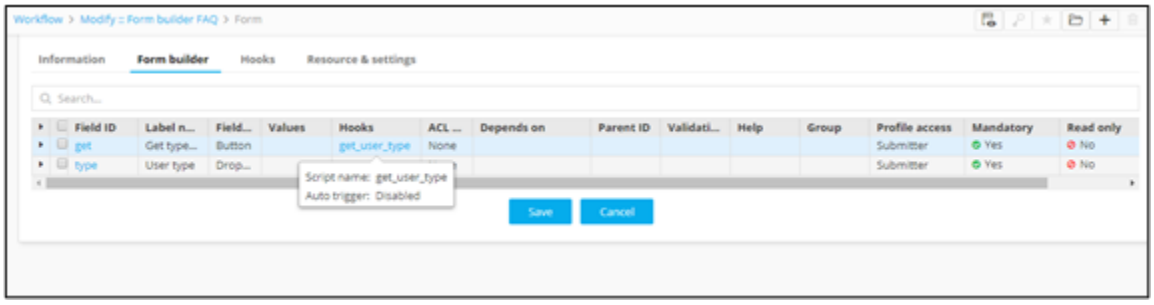
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. Create the following form fields:
   - Field Name: ‘Get types of users’, Field ID: get
   - Field Name: ‘User type’, Field ID: type

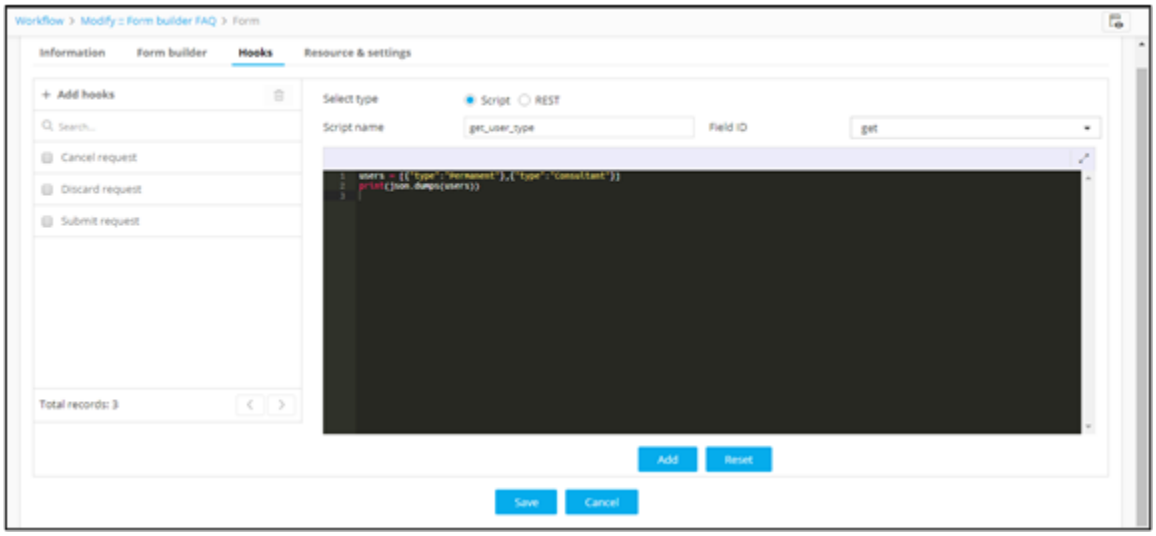
2. Define an associate script to get the ‘user type’ and map the script to field ‘Get types of users’.

   Syntax:

Script	Note
sers=[{“type”:”Permanent”}, {“type”:”Consultant”}]print(json.dumps(users))	Where “type” is the Field ID to which the retrieved values are appended to on the form



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

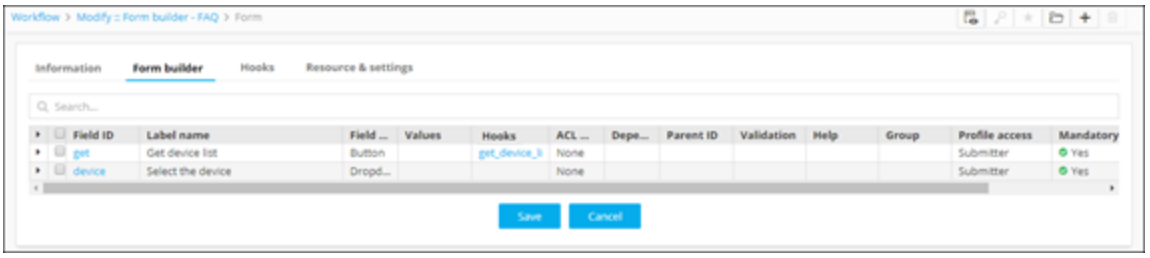


3. Click .

Triggering Script to Get Device List

To query the database by defining an associate script and get a list of F5 devices:

1. Create the relevant form fields:
   - Field Name: ‘Get device list, Field ID: get, Type: Button
   - Field Name: ‘Device’, Field ID: device, Type: Dropdown



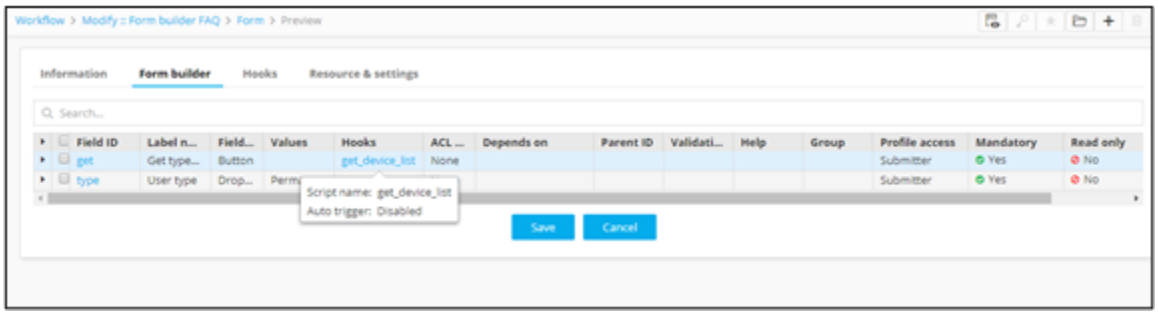
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

2. Define an associate script to get the ‘devices’ and map the script to field ‘Get’.

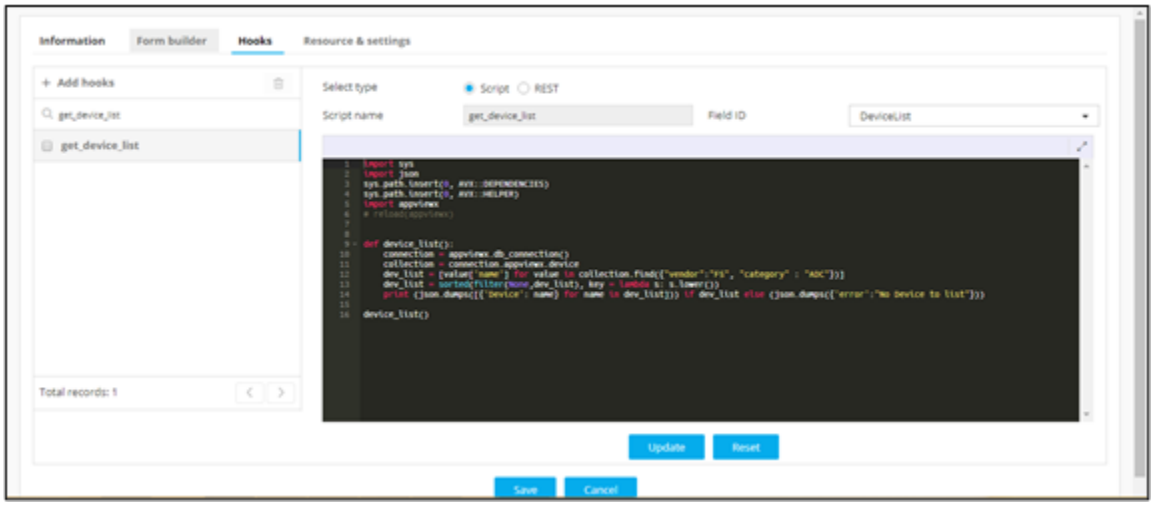
   Syntax:

Script	Note

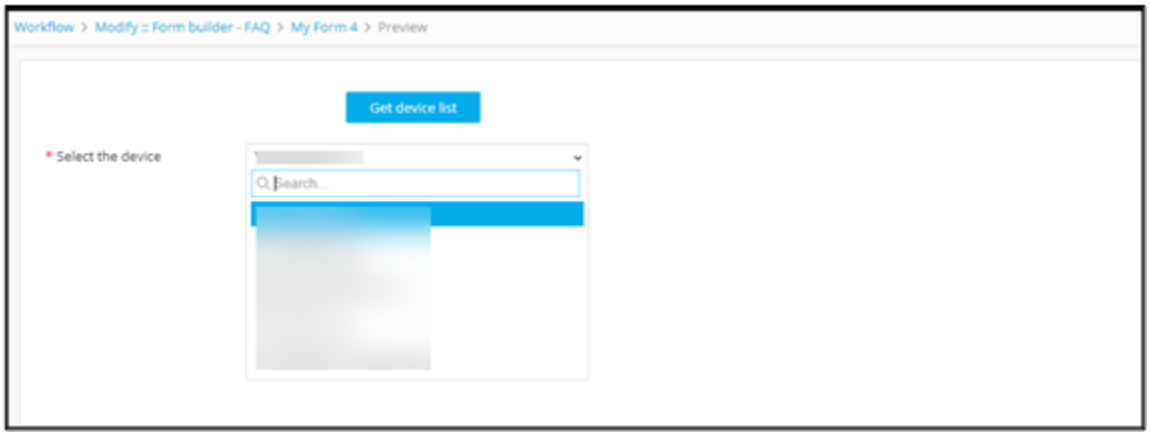
### To make use of the helper script ### 	
sys.path.insert(0,AVX::HELPER)
sys.path.insert(0,AVX::DEPENDENCIES)
### Importing the appviewx helper ### 	
import appviewx
### calling the inbuilt function the helper script to connect with the DB ### 	
`db_connection = appviewx.db_connection() device_collection = db_connection.appviewx.device`
###Creating a list of dictionaries, where each dictionary will have the key as ‘device’. Where device is the form field id to which we need to populate the values###
`device_list = [{"device":value['name']} for value in device_collection.find({"vendor":"F5","category":"ADC"})]`
### Giving an output as part of completion of the script ### 	
`print(json.dumps(device_list))`
	Where “device” is the Field ID to which the retrieved devices are mapped to on the form



Copyright © 2020 AppViewX, Inc. All Rights Reserved.



3. Click .

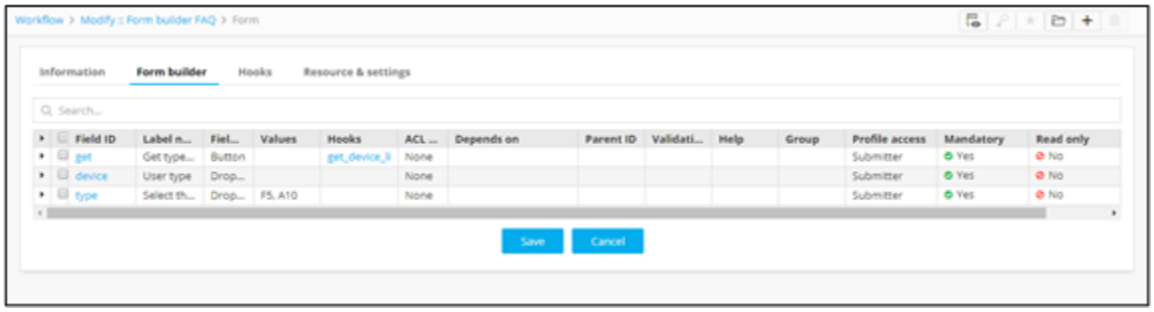


Triggering Script to Get Device List by Vendor

To query the database by defining an associate script and get a list of devices by vendor:

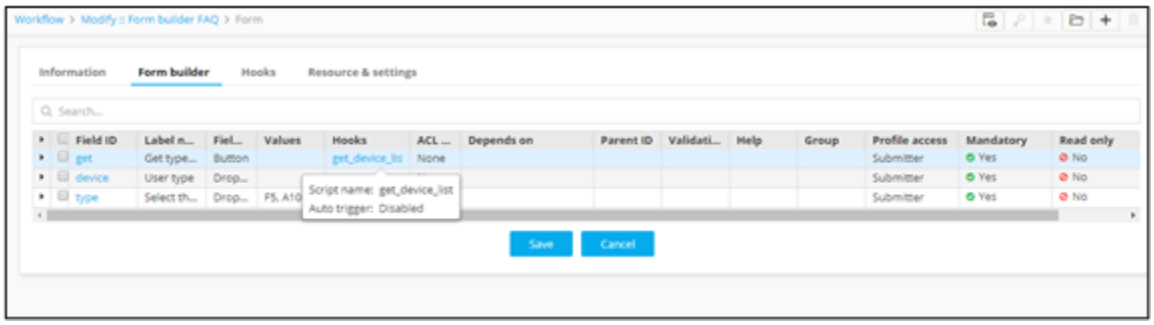
1. Create the relevant form fields:
   - Field Name: ‘Select Vendor, Field ID: type, Type: Dropdown (F5, A10)
   - Field Name: ‘Get Device List’, Field ID: get, Type: Button
   - Field Name: ‘Device’, Field ID: device, Type: Dropdown

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

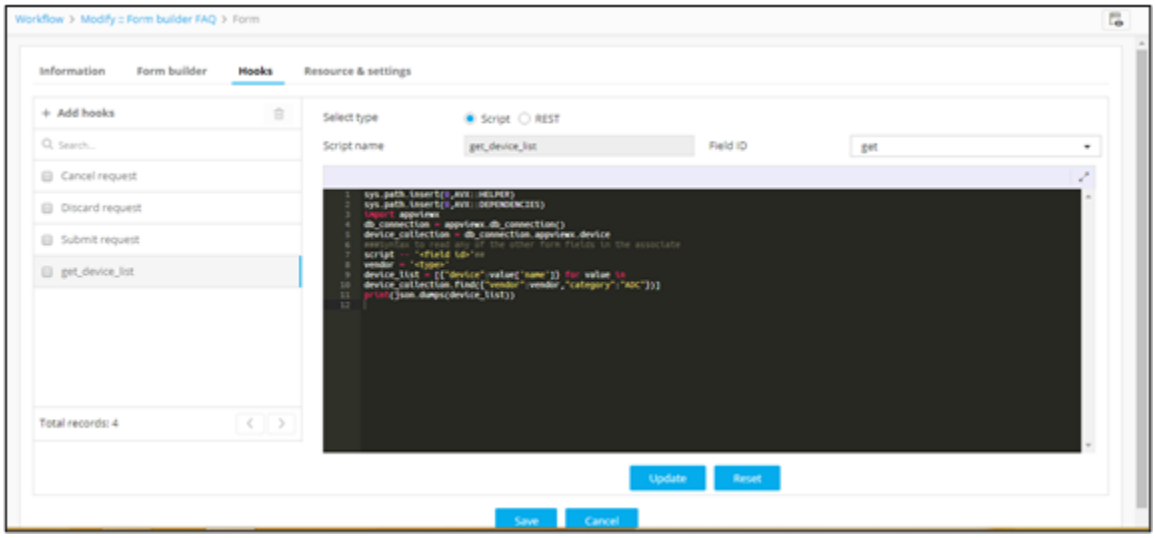


2. Define an associate script to get the ‘devices’ and map the script to field ‘Get’.

Script	Note
sys.path.insert(0,AVX::HELPER) sys.path.insert(0,AVX::DEPENDENCIES) import appviewx db_connection = appviewx.db_connection() device_collection = db_connection.appviewx.device ***###Syntax to read any of the other form fields in the associate script -- '<field id>'##*** vendor = '<type>' device_list = [{"device":value['name']} for value in device_collection.find ({"vendor":vendor,"category":"ADC"})] print(json.dumps(device_list))	• Where ***“<type>”*** is input value from the form based on which the device(s) will be retrieved • *Device”* is the destination Field ID to which the list of devices are mapped back on the form “

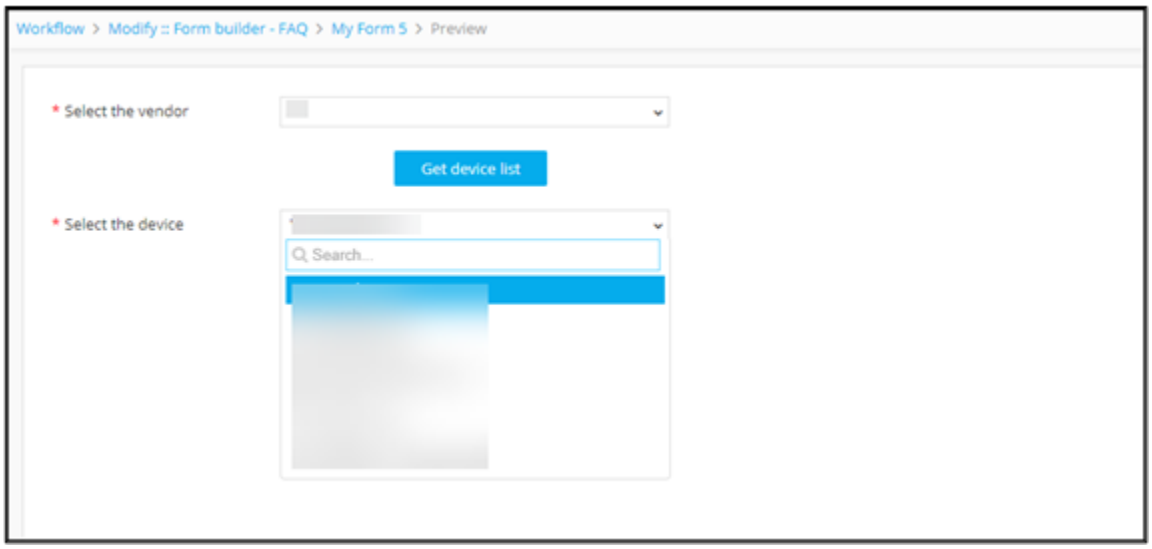


Copyright © 2020 AppViewX, Inc. All Rights Reserved.



3. Click .

   The script is triggered to get the device list by vendor.

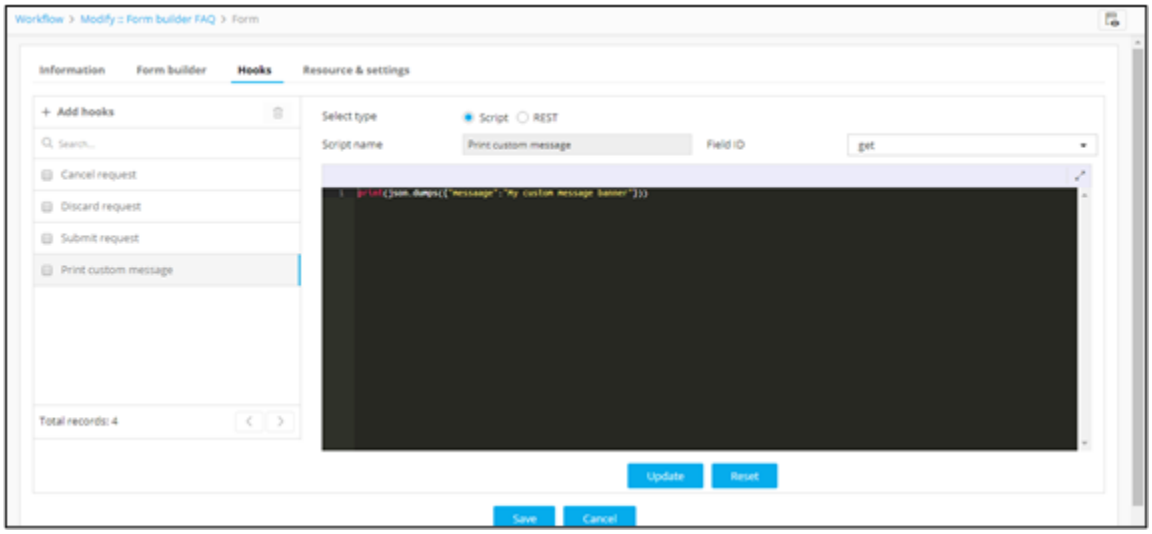


Triggering a Script to Print Custom Message

To print a custom banner message by defining an associate script:

1. Define a form field with Field Name as Print Message, Field ID as print, and Type as Button.
2. Define associate Script and map it to the field ID. The following syntax can be used to print the message by adding the string to the key "message": `print(json.dumps({"message":"My custom message banner"}))`

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



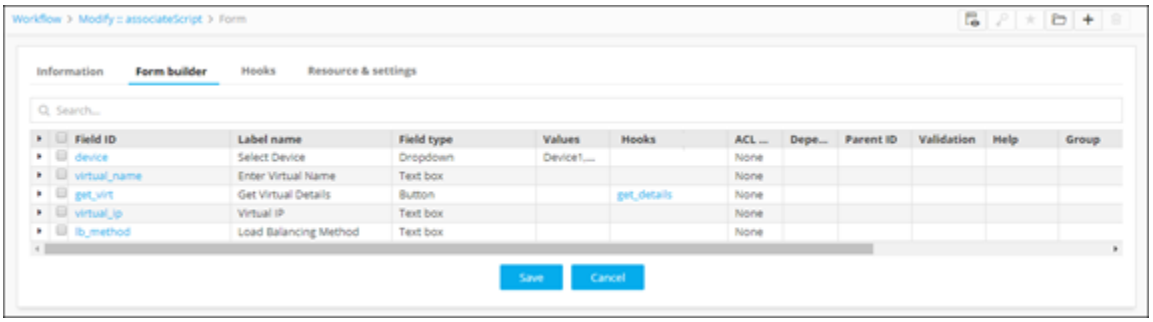
3. Click .

Using Auto-trigger to Invoke the Script Automatically based on Input

To be able to retrieve the IP address and LB method for a selected device:

1. Define a form field with element type Dropdown to select one or more device(s) [Field ID: Device].
2. Define a form field with element type ‘Button’ to retrieve the basic ‘Virtual Details’.
   a. Under the ‘Associate scripts’ tab, define the ‘associate script name’; and the script logic i.e. ‘Get details’ [Field ID: get_virt]
   b. Map the associate script against the relevant form field ID (OR)
   c. Map the field from the list of ‘associate scripts’ displayed against the ‘associate script’ field under ‘Field properties’.
3. Define form field to show the ‘Virtual IP’ [Field ID:virtual_ip].
4. Define form field to show the ‘Virtual IP’ [Field ID:lb_method].

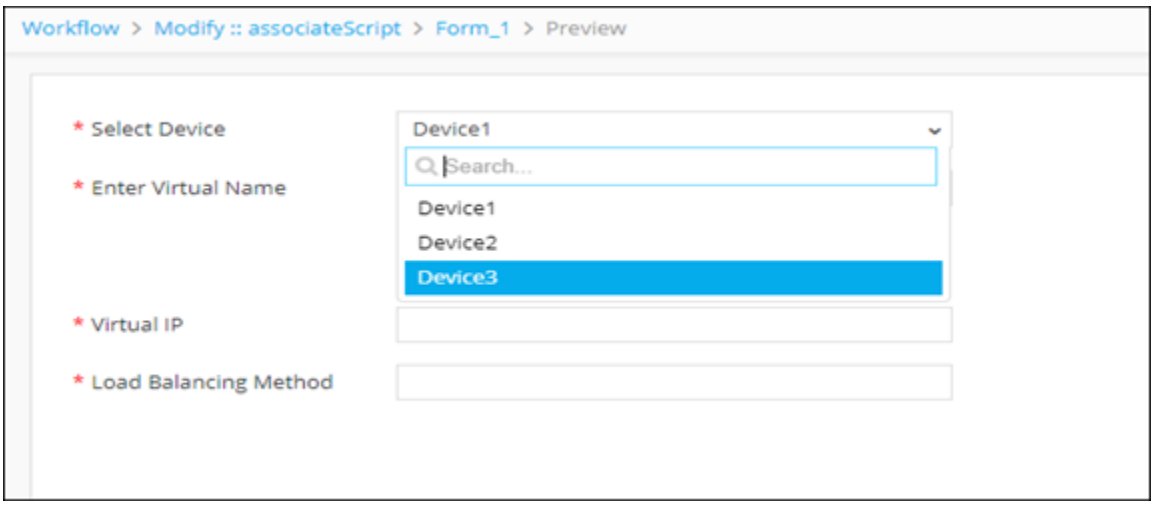
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



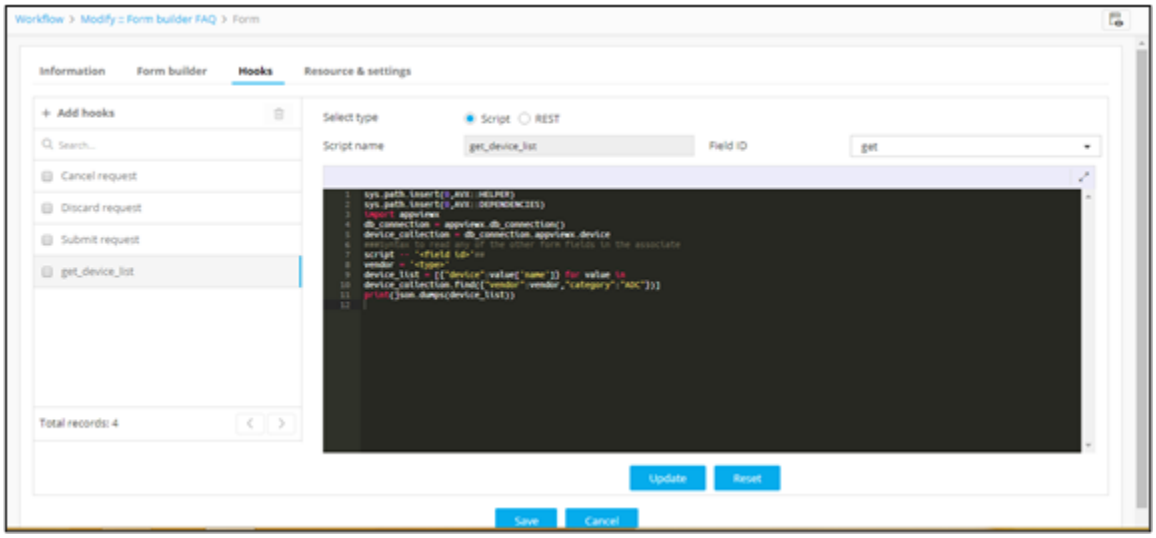
Associate Script logic

```
import sys
import json
device = '<device>'----------Input field used to query
virtual_name = '<virtual_name>'
def get_virtual_details():
    if device == 'Device1':
        ip = '192.168.41.237'
        lb_method = 'ratio'----------Logic to get IP and lb method for a device
    if device == 'Device2':
        ip = '192.168.41.199'
        lb_method = 'round-robin'
        print(json.dumps([{'virtual_ip':ip},{'lb_method':lb_method}])) ----------Return values mapped to the destination field
get_virtual_details()
```

5. Select Device 1 as Input when executing the script via form.

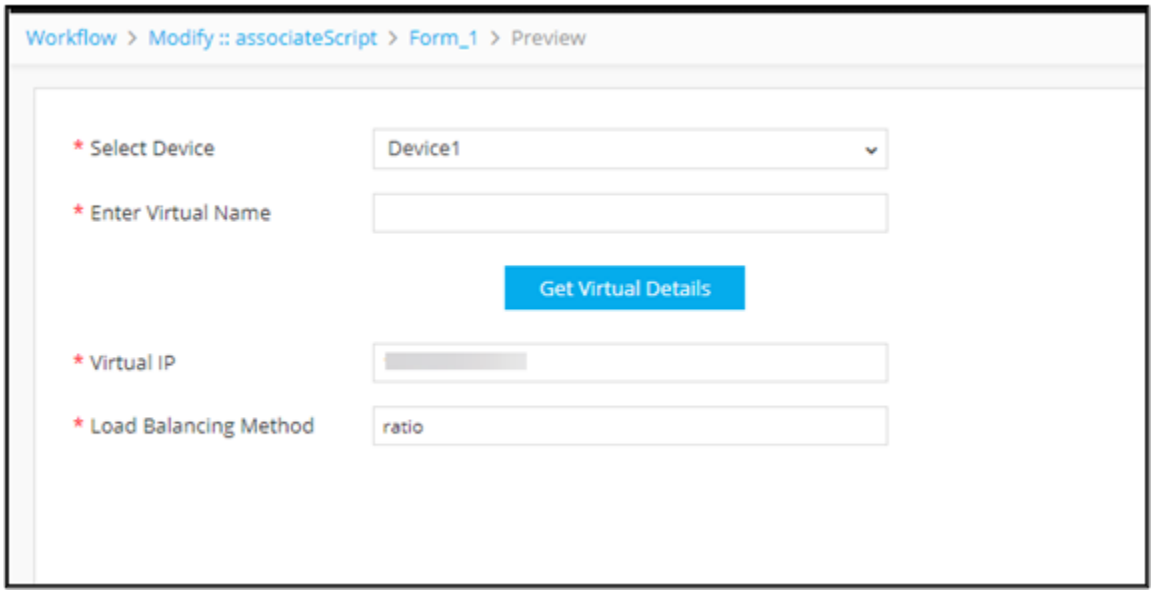


Copyright © 2020 AppViewX, Inc. All Rights Reserved.



6. Click **Get Virtual Details** using Device 1 as Input.

   The response from the query is mapped to the destination fields: Virtual IP and Load Balancing Method.

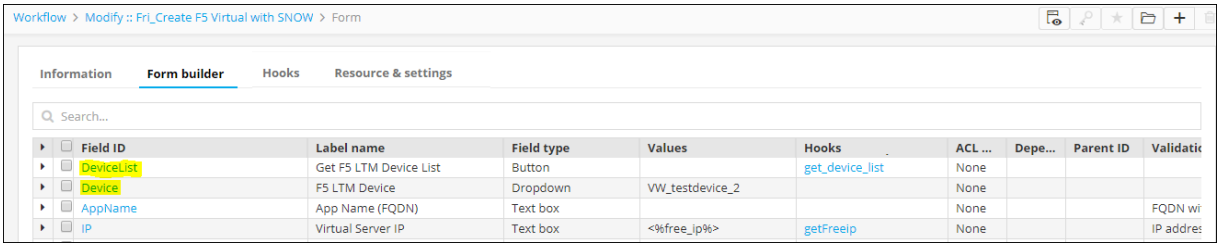


Triggering a Script Directly from a Form Field to Map its Values to Destination Fields

To map destination field values by triggering a script from form fields:

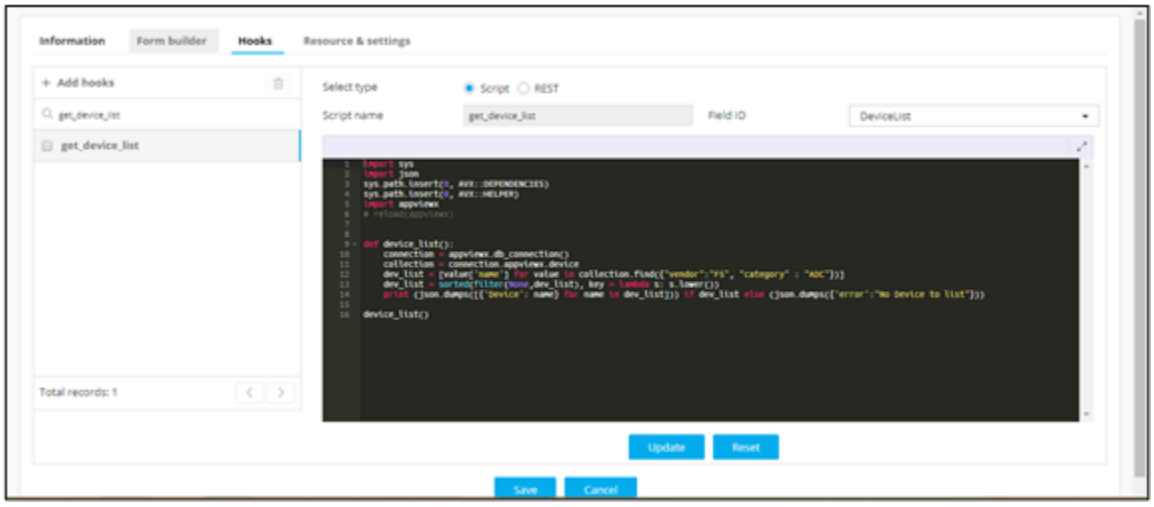
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. Design a workflow.
2. Define a form to Create F5 VIP where one of the parameters is to fetch the ‘list of F5 Devices’.
3. Define the form field called ‘Get Device List’ with Field ID ‘DeviceList’ which will be mapped with an associate script to get the list of devices .
4. Define the destination form field called ‘F5 LTM Device’ with Field ID ‘Device’ where the values from the associate script needs to be mapped to.



Note: For more information on adding form fields, click here.

5. Define Associate Script(s):
   a. Define custom script logic.
   b. Use ‘Ctrl + Space’ in the script editor to list all the pre-defined helper scripts and embed the appropriate helper script(s) within an ‘associate script’.

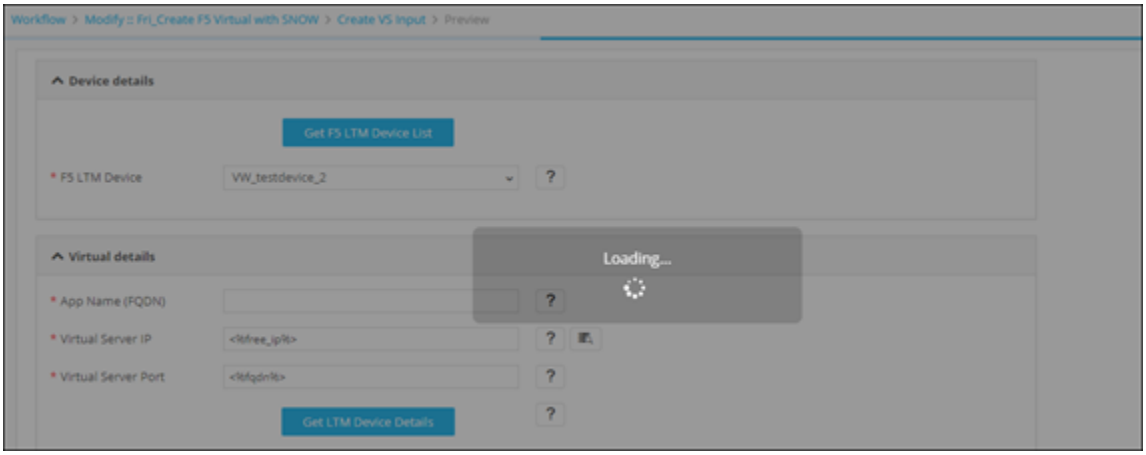


Associate Script

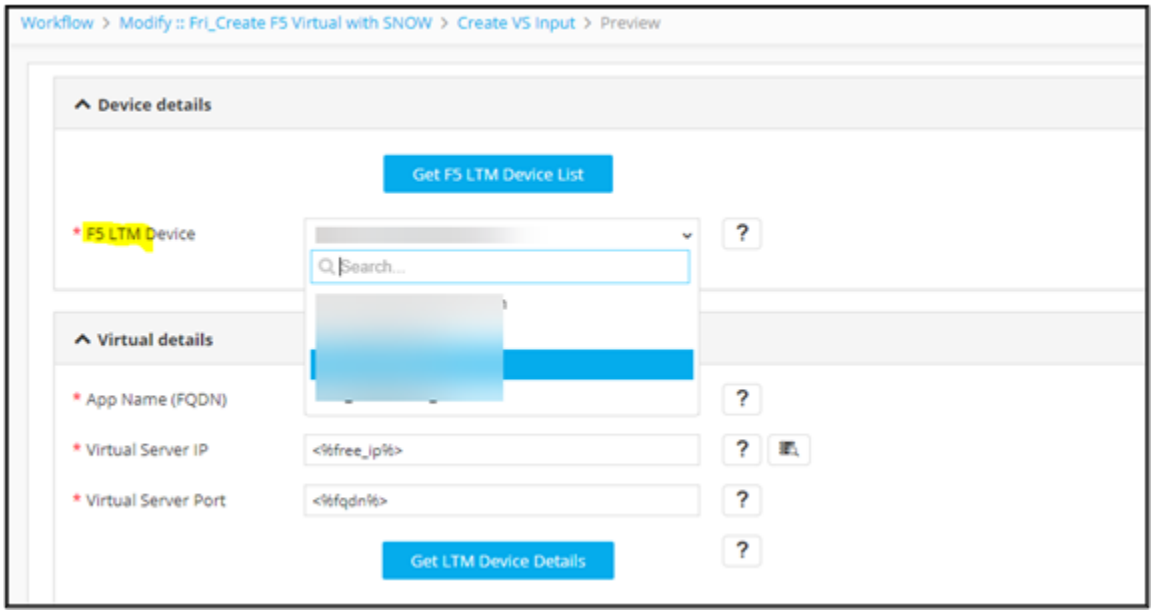
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Associate Script (Get device List)	Mapping Details
import sys *import json* *sys.path.insert(0, AVX::DEPENDENCIES)* *sys.path.insert(0, AVX::HELPER)* *import appviewx* *# reload(appviewx)* *def device_list():* *connection = appviewx.db_connection()* *collection = connection.appviewx.device* *dev_list = [value['name'] for value in collection.find({"vendor":"F5", "category" : "ADC"})]* *dev_list = sorted(filter(None,dev_list), key = lambda s: s.lower())* *print (json.dumps([{'Device': name} for name in dev_list])) if dev_list else (json.dumps({'error':"No Device to list"}))* *device_list()*	• The script queries the database and retrieves the list of F5 devices • Field marked in yellow is the destination field on the form where the queried values from the script are mapped to

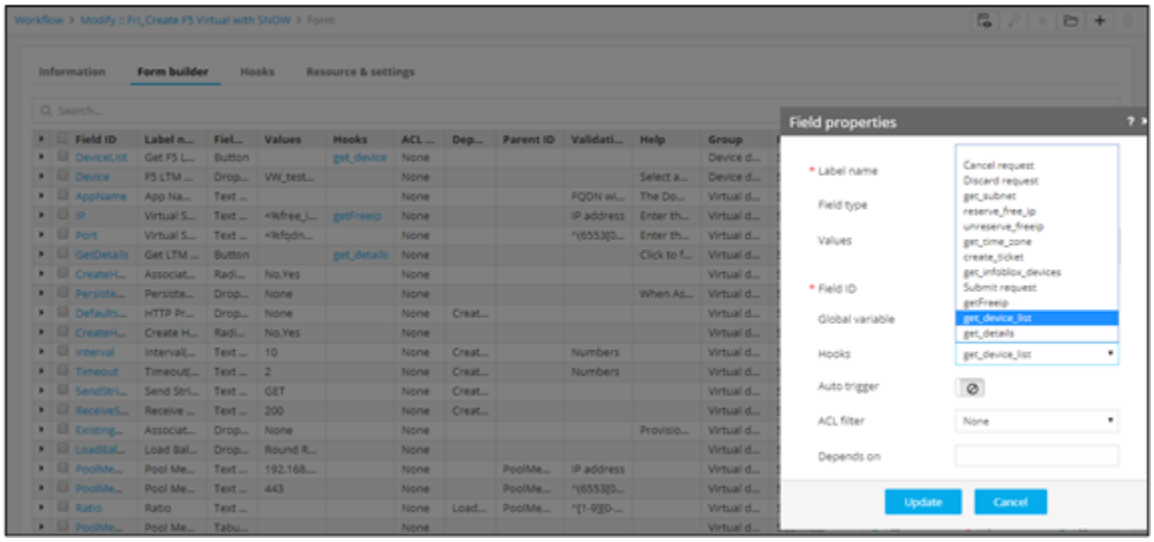
6. To execute the associate script on the form, click **Get F5 LTM Device List**.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.



7. Once the script is defined, it can be referred against the respective form field.



Auto-trigger scripts

Associate scripts in the form can be triggered either manually or automatically. Auto trigger eliminates manual intervention in the form. The input from the form is used to auto trigger an associate script in the backend.

You can choose to enable the auto trigger option against every form field. The scope of auto trigger is limited to the following:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

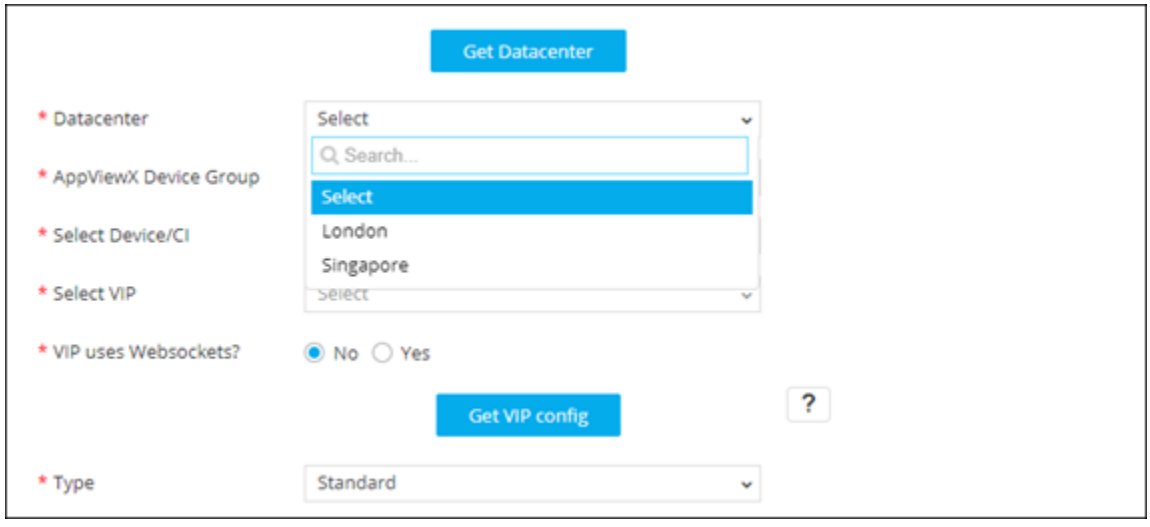
- Auto trigger works for the first form field on loading.
- The input from the form must be from a radio button or dropdown form field.
- Based on the input, auto trigger gets executed ONLY for the next immediate field.

1. Design a form using the following fields and associate scripts:

Field Name	Form Input/Response	Autotrigger script to execute
Get Datacenter	Get Datacenter - Button	getdataCenterlist
Datacenter	Response: Display list of datacenters in dropdown	getDeviceGrouplist
AppViewX Device Group	Response: Display List of Device Groups in dropdown	getDevicelist
Select Device/CI	ResponseL Display list of devices in dropdown	getVIPNames
Select VIP	Response: Display List of VIP Names for the device in dropdown	

Note: For more information on adding form fields, click here.

2. To preview the form, click
3. In the form preview, click **Get Datacenter**.
   The script is auto triggered to populate the dropdown in the next (Datacenter) field.



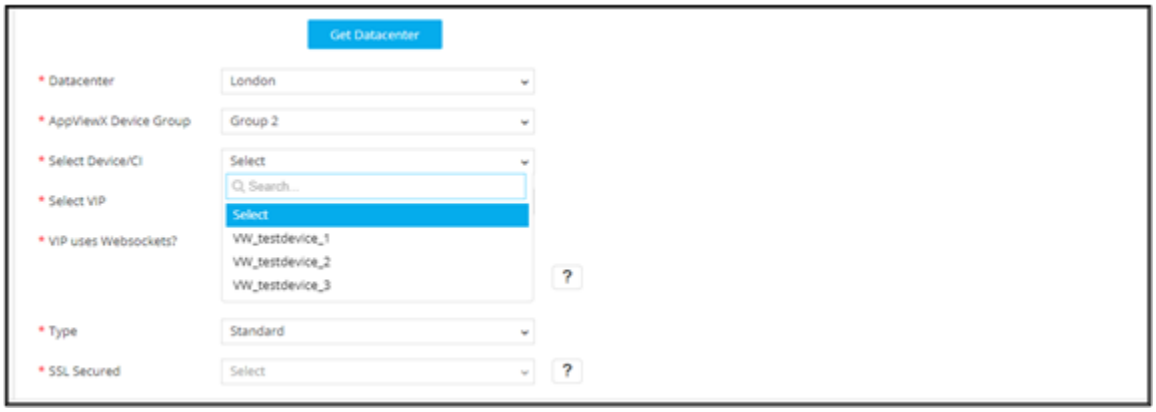
4. From the options displayed in the **Datacenter** dropdown list,select London from the dropdown list.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

The script is auto triggered to populate the next (device group) field.

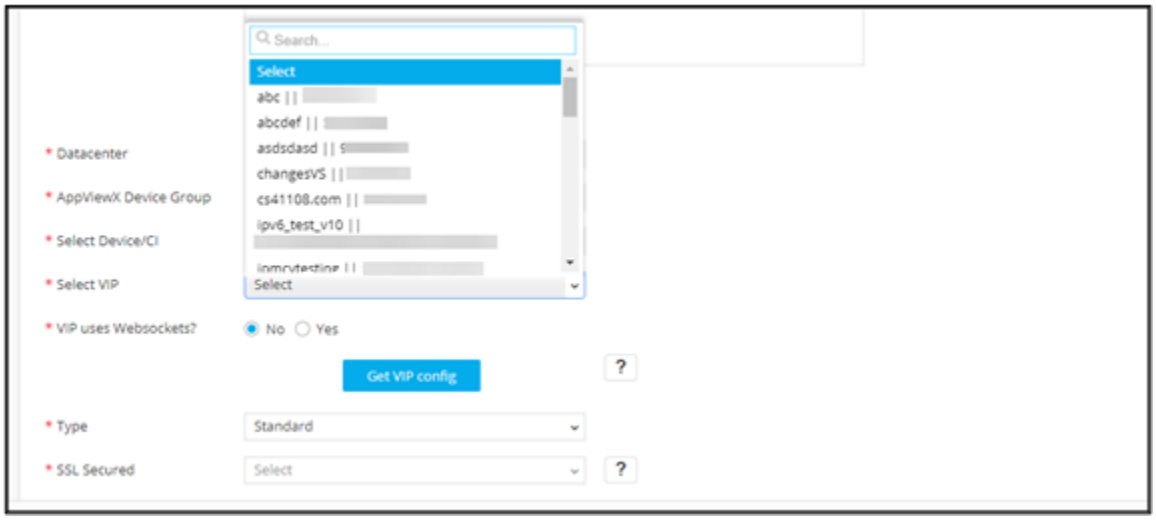
5. Select **Device group** as Group 2.

   The script is auto triggered to populate the next (select device) field.



6. Select the **Device** to auto trigger the script and populate the next (select VIP) field.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



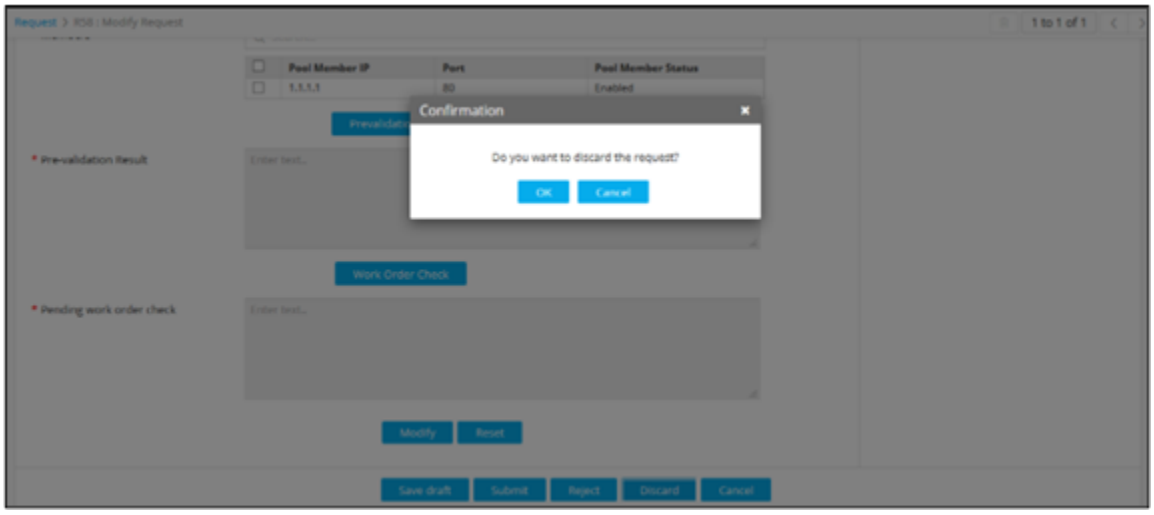
Default script for Cancel and Discard Actions

The request form allows you to either ‘Cancel’ or ‘Discard’ a particular request prior to final submission. Such an event may result in partial transactions which may require a ‘clean-up’.

In order to cater to such operations, default associate scripts for cancel and discard actions are provided within the associate script inventory. You can define a custom logic within the default scripts which will be triggered based on the user’s action on the form.

For example, As part of a Create DNS workflow, a user gets a Free IP, reserves it, and then immediately wants to cancel the request transaction. In this case, the reservation of IP and any other transactions will need to be ‘cleaned up’. The Cancel script Logic releases any temporarily locked free IP and releases any reservation on the IPAM in the event of a cancel or discard action on the form.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

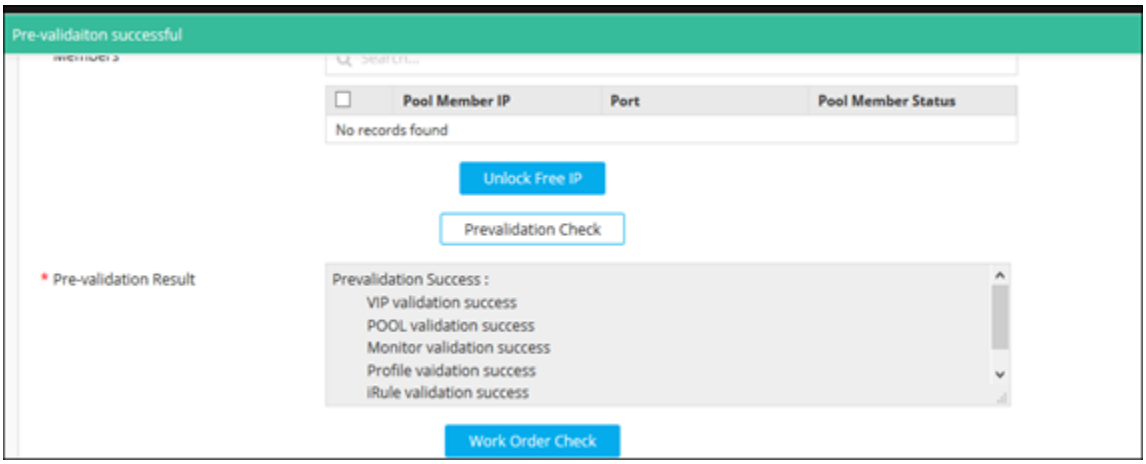


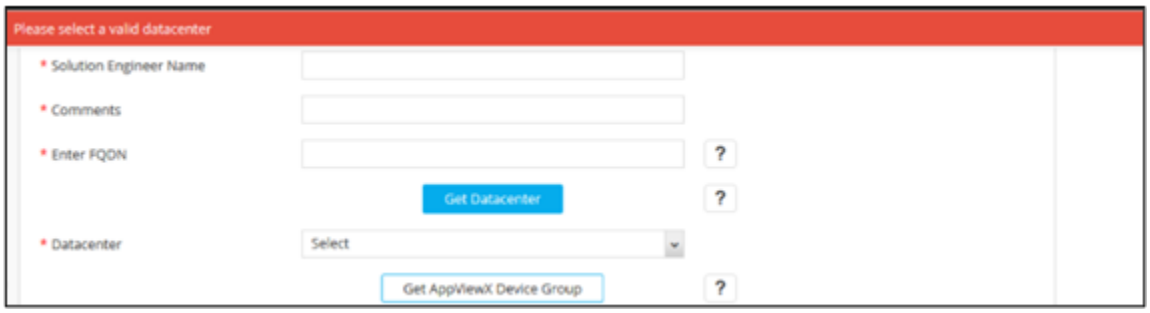
Note: Default associate scripts for ‘Cancel’ and ‘Discard’ cannot be deleted.

Define ccustom Notification Messages using Scripts

For more personalized notifications, you can define custom scripts to display custom notification messages on the request form. Notification messages can be displayed in either ‘red’ or ‘green’ tickers based on the response type - message or error.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.





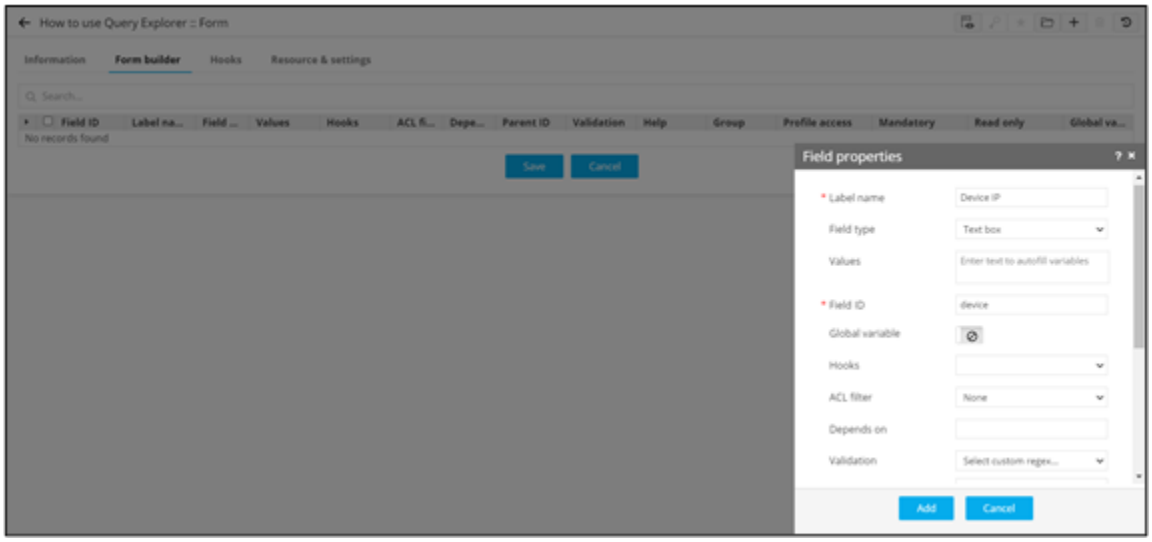
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Defining Query Explorer to Build Forms

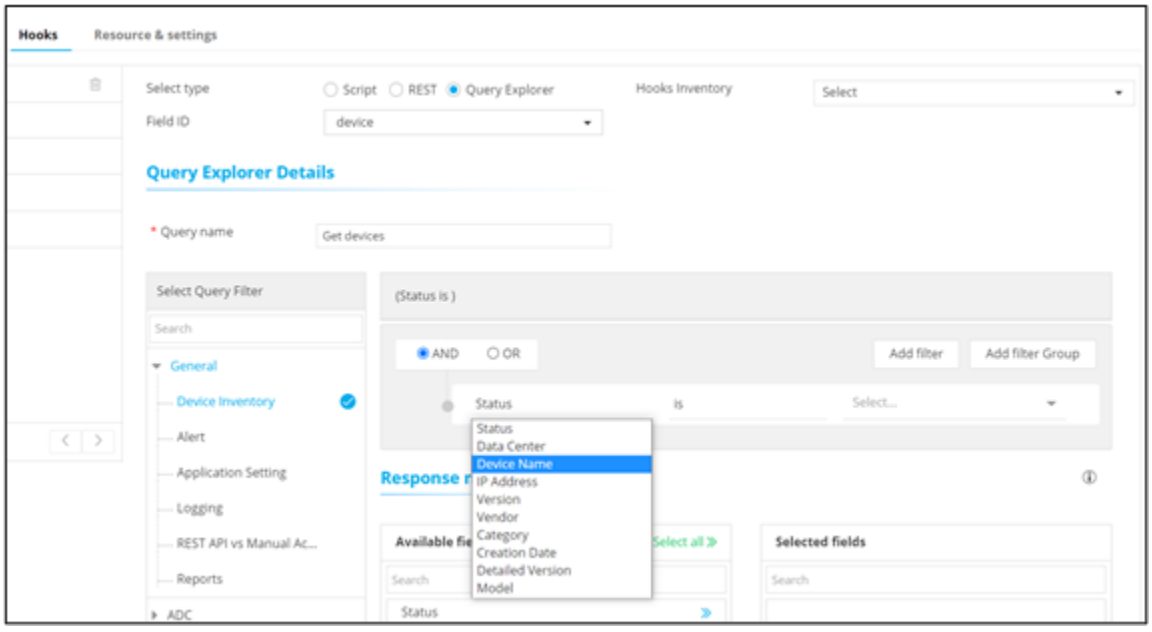
Query explorer allows you to design custom queries, business logic, and build self-service forms quickly.

To design a query to ‘get device details’ using query explorer and build forms quickly:

1. Design/Modify a custom workflow.
2. From the **User Interface** section, drag and drop the **Form** task.
3. Under **Form builder**, define the form field to fetch specific details from the database.

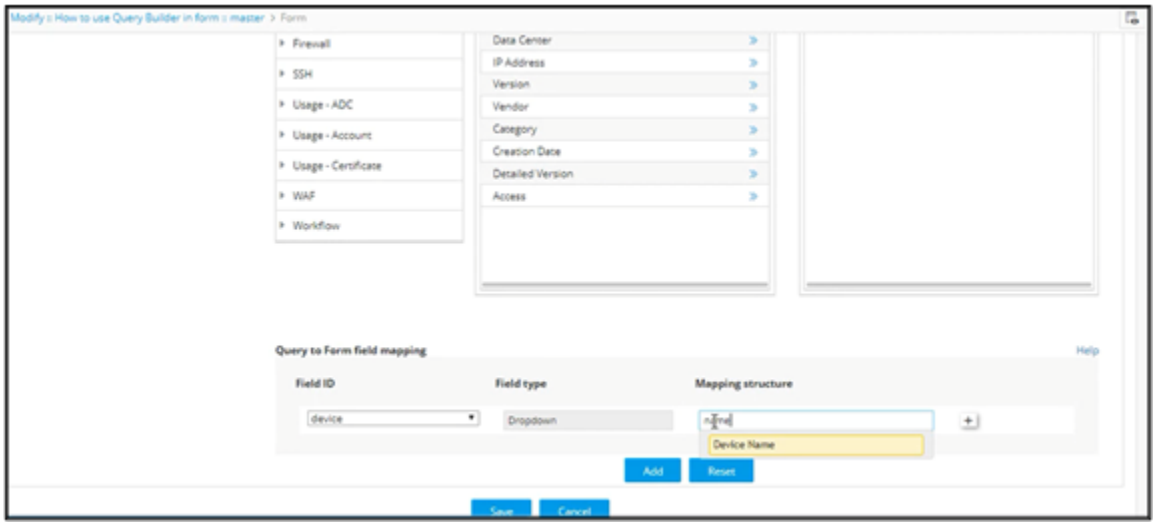


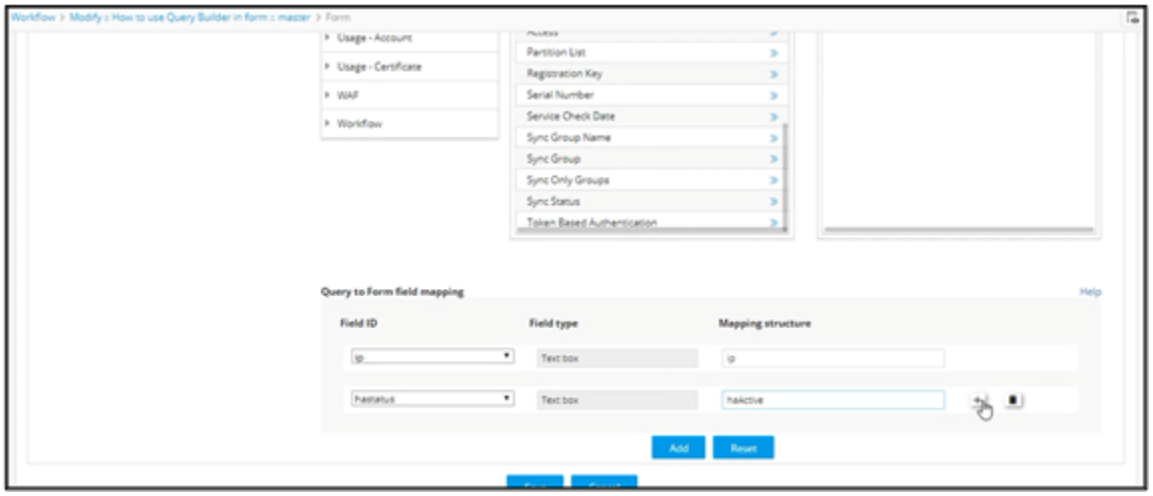
4. Under **Hooks**, define the required Hook using Query explorer.
5. Select the required response from the query to be mapped to the form field.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

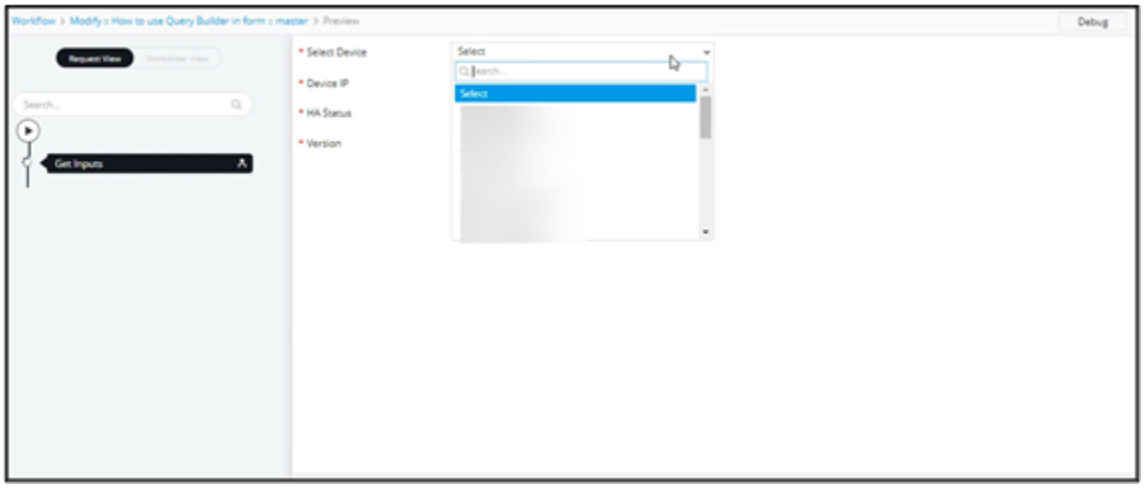
6. Map the query response to the specific form field.





7. To validate the execution, click .

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

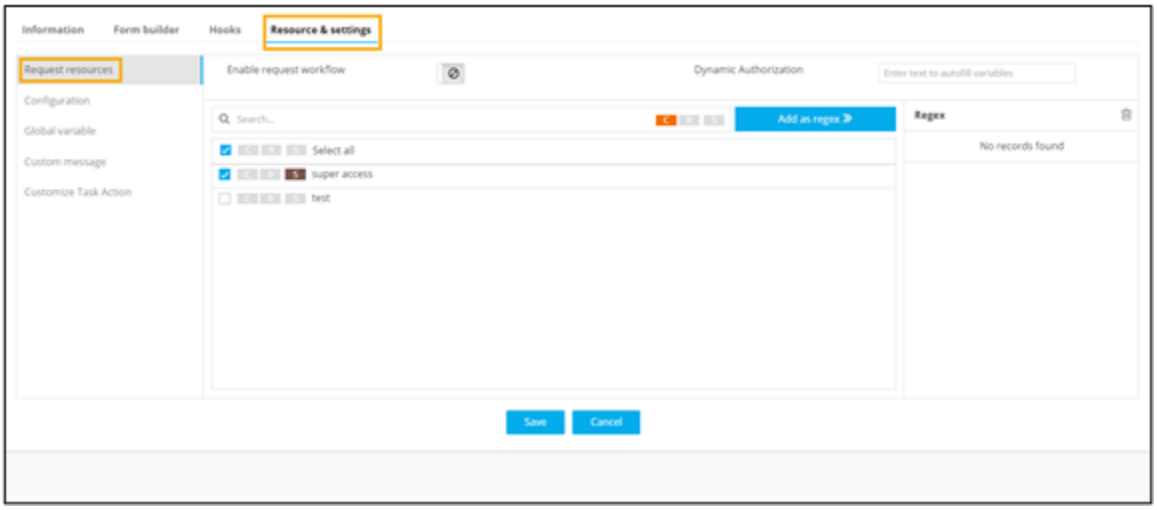


Defining Resource & Settings

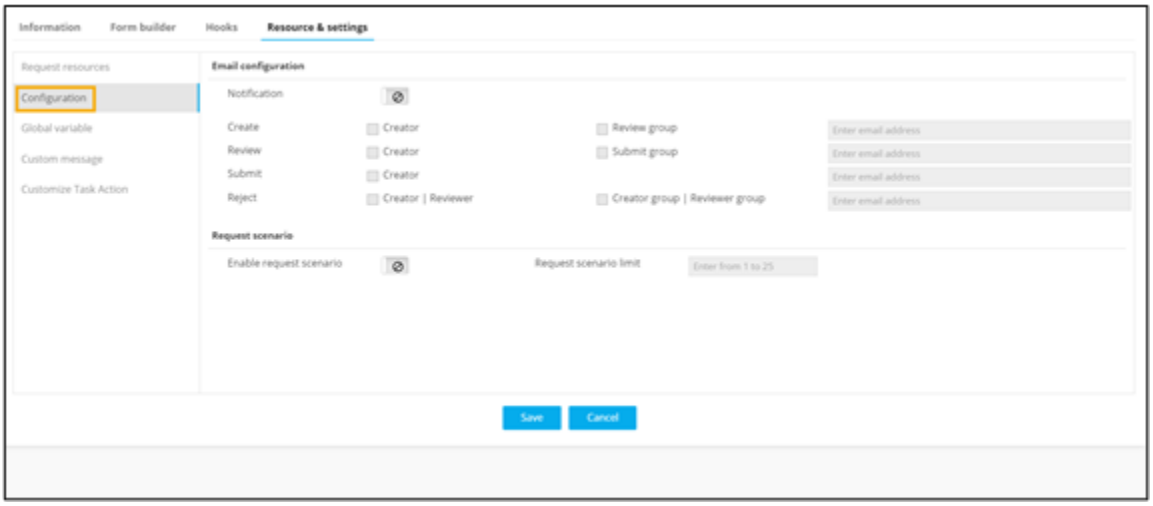
This section supports the following:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- **Request resources**: Provision to assign role based access to the form task.

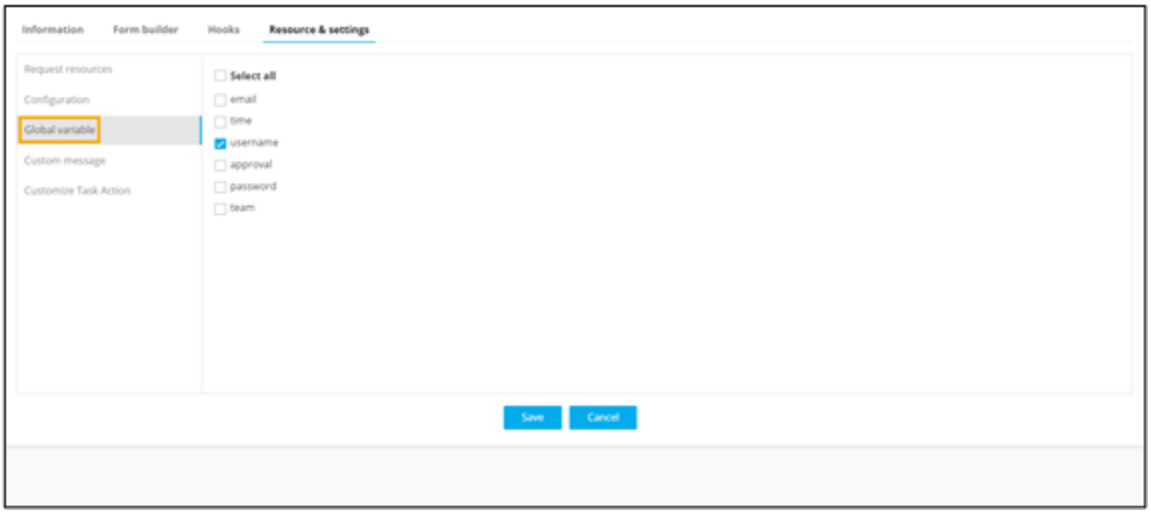


- **Configuration**: Provision to enable multiple approvals of the form task.

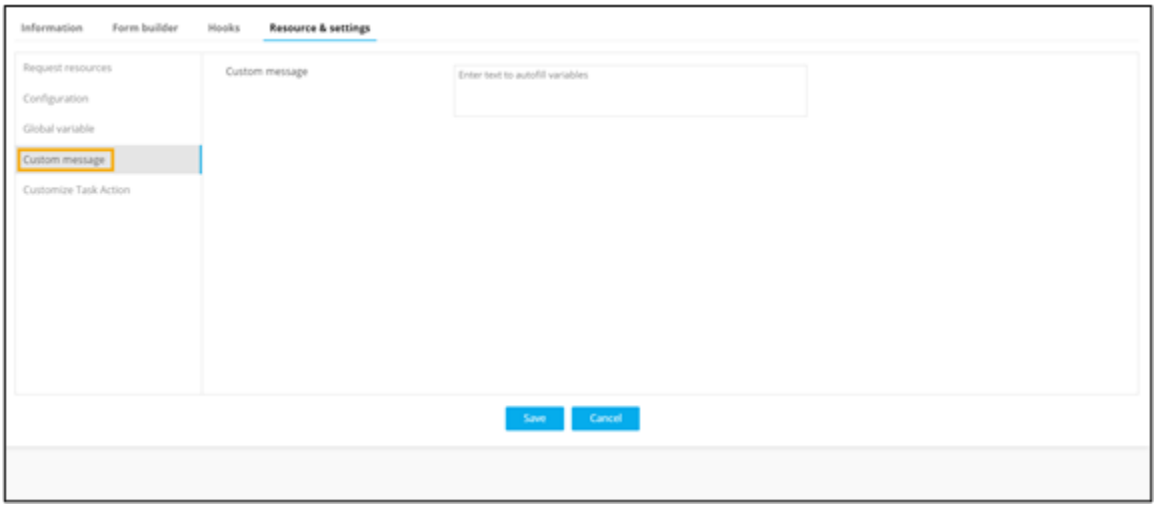


- **Global variable**: Provision to declare one or more form field values as global variables.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

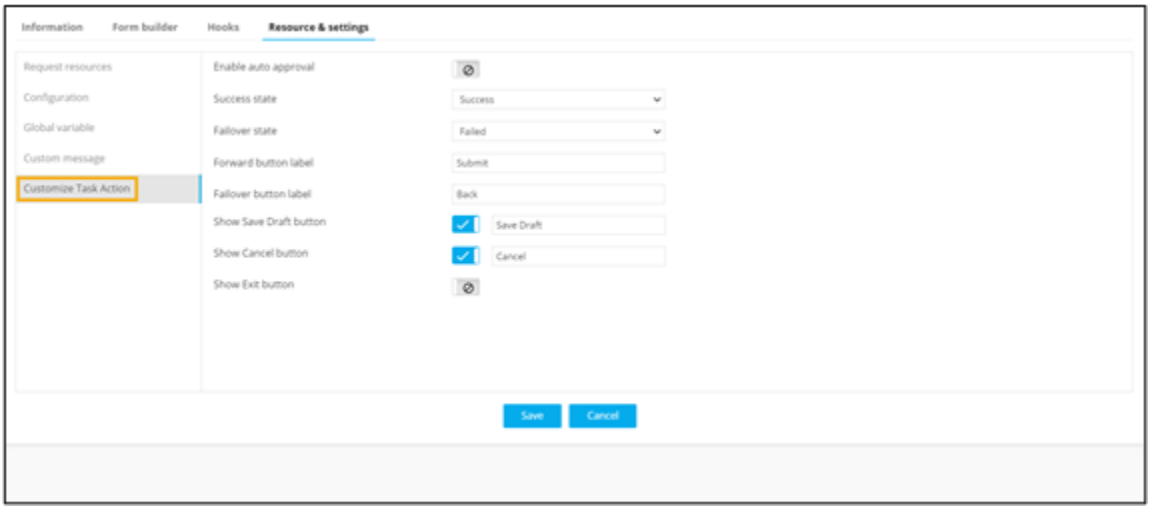


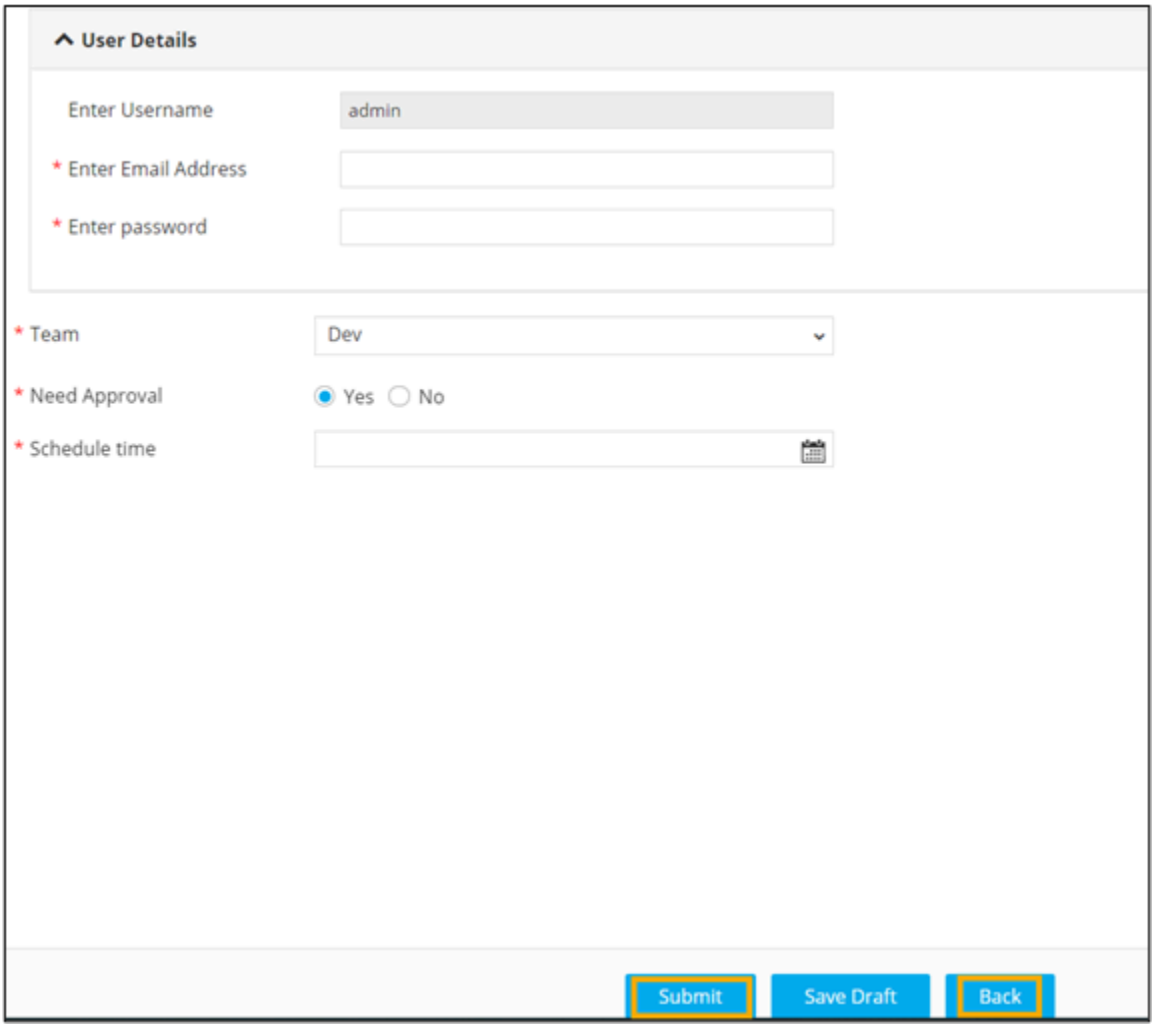
- **Custom Message**: Provision to define context based messages for notifications between workflow stages. For example, User access denied.



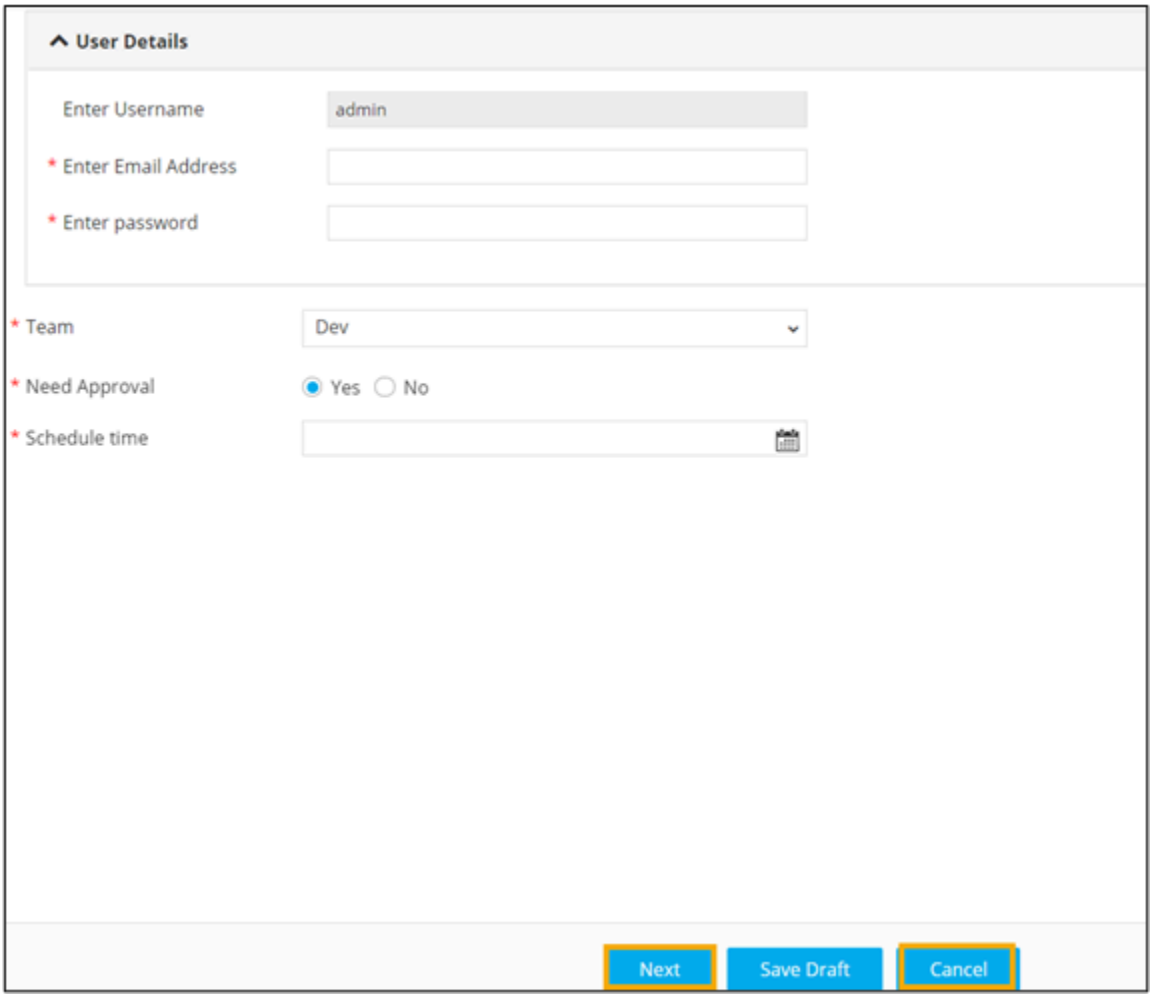
- **Customize Task Action**: Provision to enable auto approval of the form task, provide an ‘alias’ of the form actions. For example, the ‘Submit’ button can be renamed to ‘Next’, ‘Back’ button can be renamed to 'Cancel’.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.





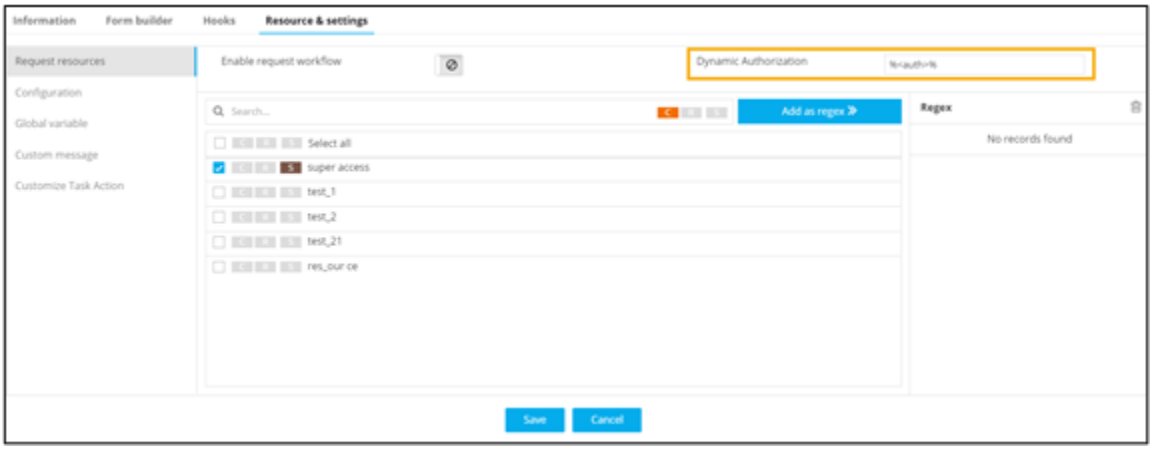
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



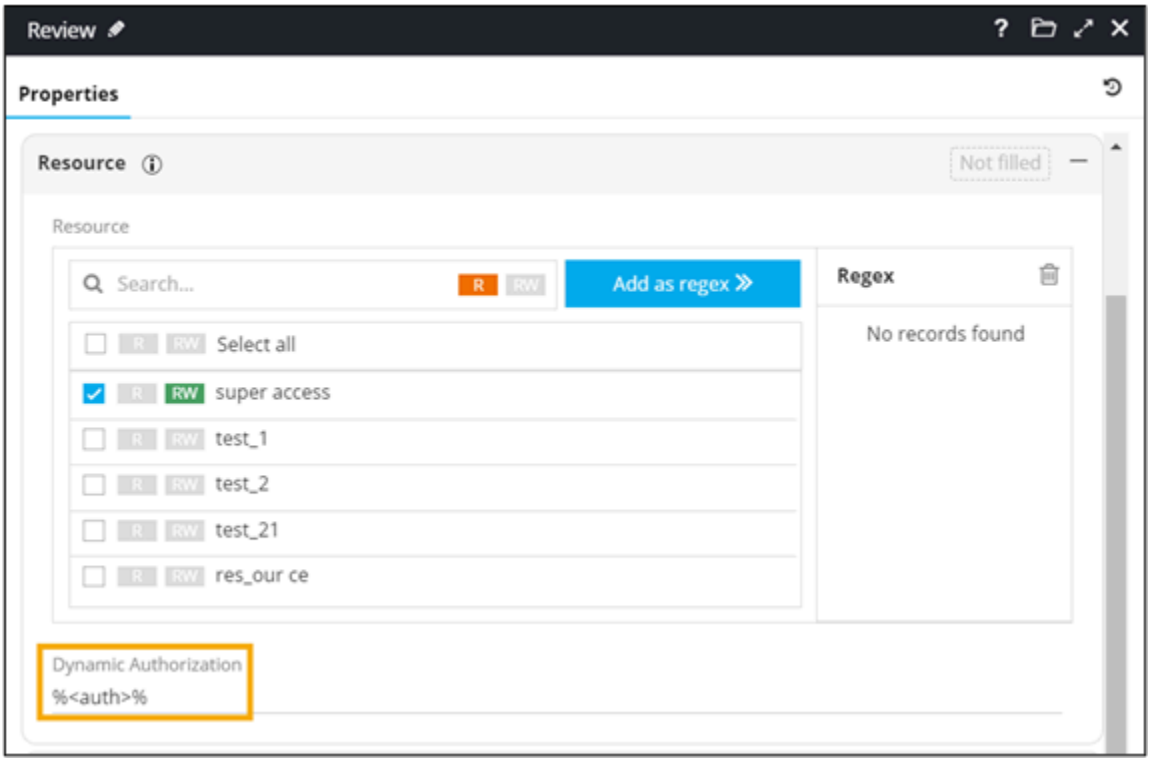
- **Dynamic Authorization**: Provision to enable authorization dynamically for users or user groups by using the global variable `%<auth>%`.
  New users/user groups must be defined in the script logic using the following structure:

```
[{"type":"user","value":["admin","tim"],"access":"R"},
{"type":"role","value":["admin","network-sre"],"access":"RW"},
{"type":"usergroup","value":["admin usergroup"],"access":"RW"}]
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Note: Dynamic Authorization is not limited to the Form task. It can be enabled for all User Interface tasks.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Grid

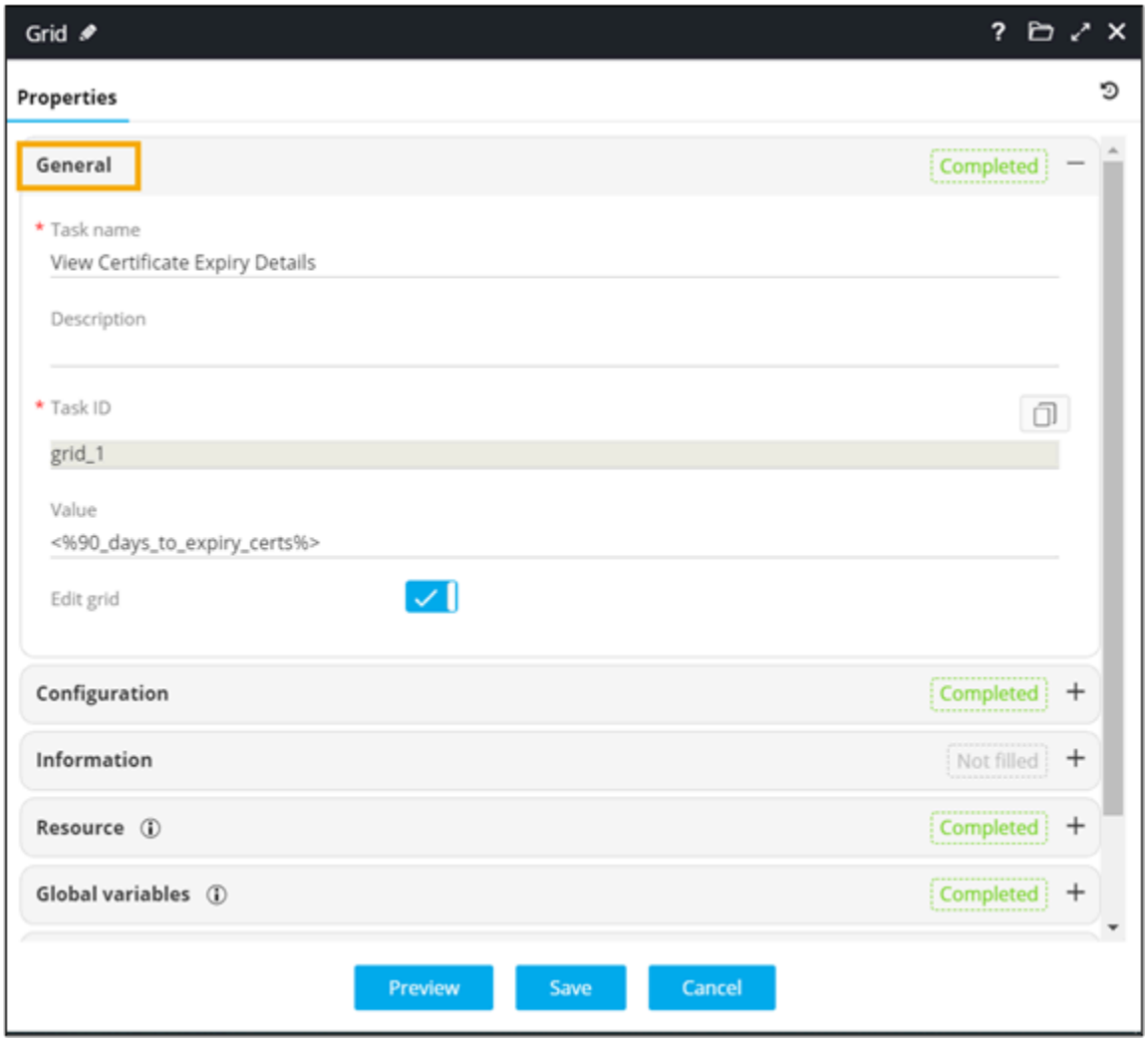
This task allows you to define a custom ‘grid’ component as part of the workflow execution process. The Grid component can be used either to display some data or to gather input from users.

- Provision to define a custom grid component with number of columns.
- Provision to define custom column/label names.
- Provision to dynamically pass data (using variables) from any stage of the workflow into the grid task.
- Provision to assign user role (RBAC) access.
- Provision to define custom button/label names in the event of using the grid task as part of any approval process.
- Provision to download the contents from Grid into .xls or .csv files.
- Provision to dynamically define column/label names from any stage of the workflow into the grid task.
- Provision to define custom messages.
- Provision to declare global variables.

To add a Grid task to the workflow:

1. Design a workflow.
2. From the **User Interface** section, drag and drop the **Grid** task.
3. In the **Grid** task window, under **Properties**, in the **General** section, enter the relevant field information.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

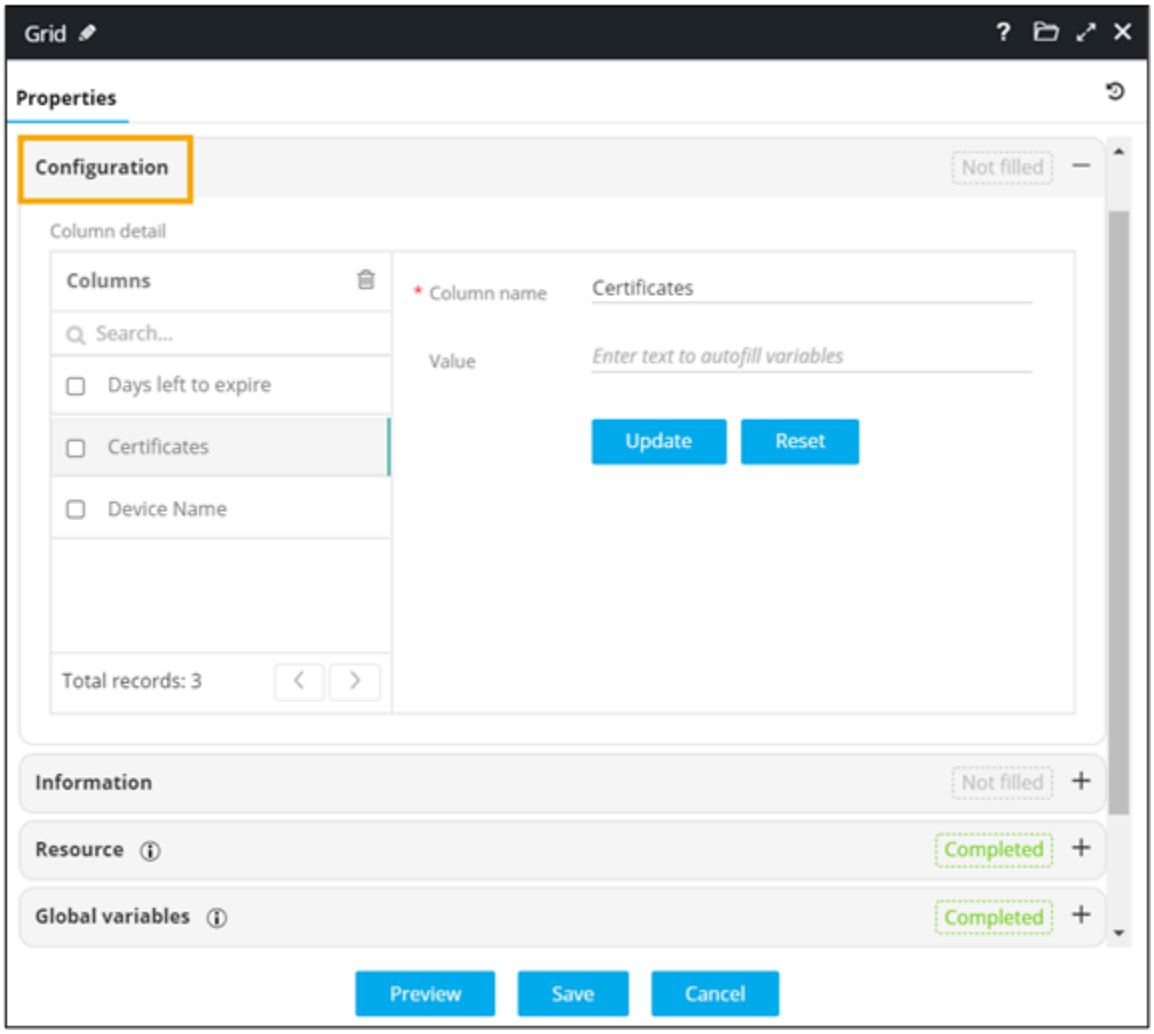


This table describes all the field in this section:

Field	Description
***Task name**	Enter a task name.
Description	Enter a description of the task.
***Task ID**	Enter a unique task value ID for the task.
Value	Refer a global variable from a previous task to render the grid data.
Edit grid	Enable 'Edit grid' to allow you to edit the contents of the grid.
All Asterisk (*) marked fields are mandatory.	

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

4. In the **Grid** task window, under **Properties**, in the **Configuration** section, define the columns for the grid.

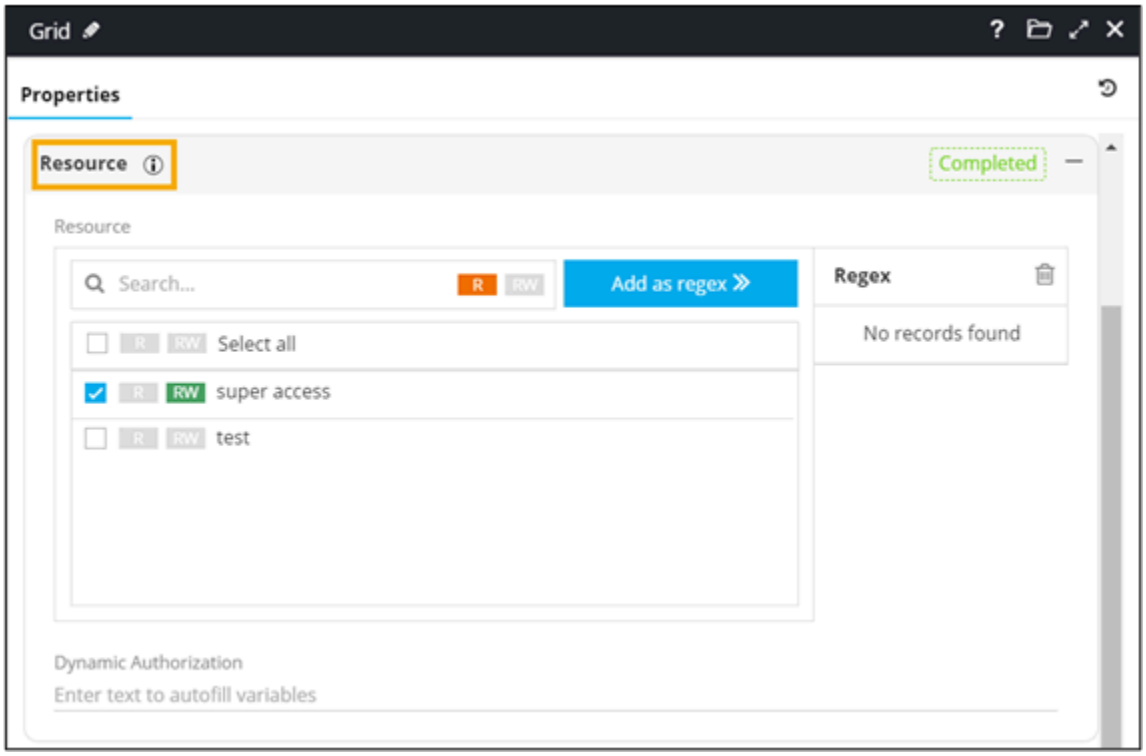


This table describes the fields in this section:

Field	Description
***Column name**	Enter a valid column name.
Value	Enter a static or dynamic value. Value(s) can be referred from any workflow task to render the grid data.
All Asterisk (*) marked fields are mandatory.	

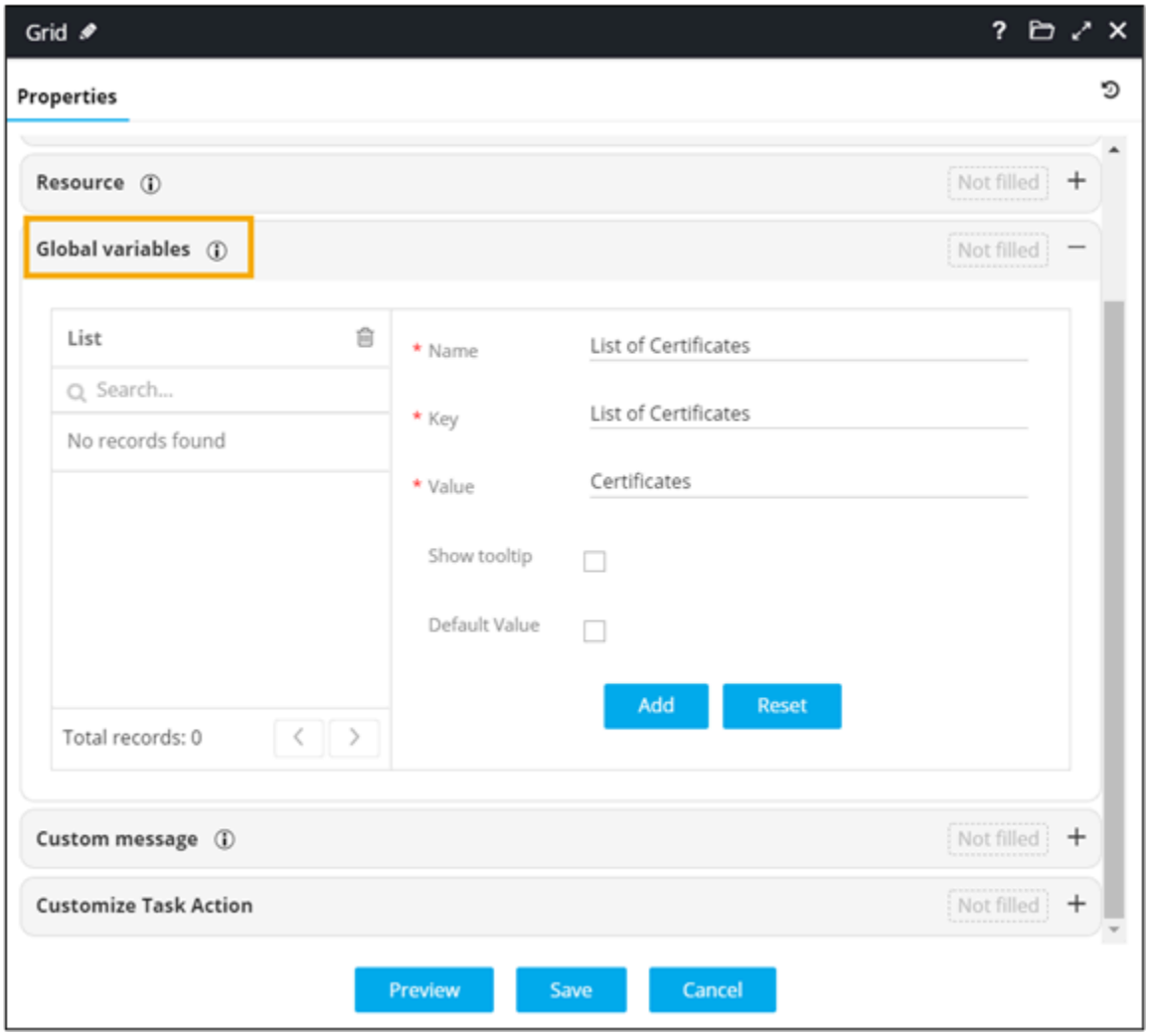
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

5. In the **Grid** task window, under **Properties**, in the **Resource** section, assign resource based access to the task.



6. In the **Grid** task window, under **Properties**, in the **Global variables** section, define values that need to be declared as global variables.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

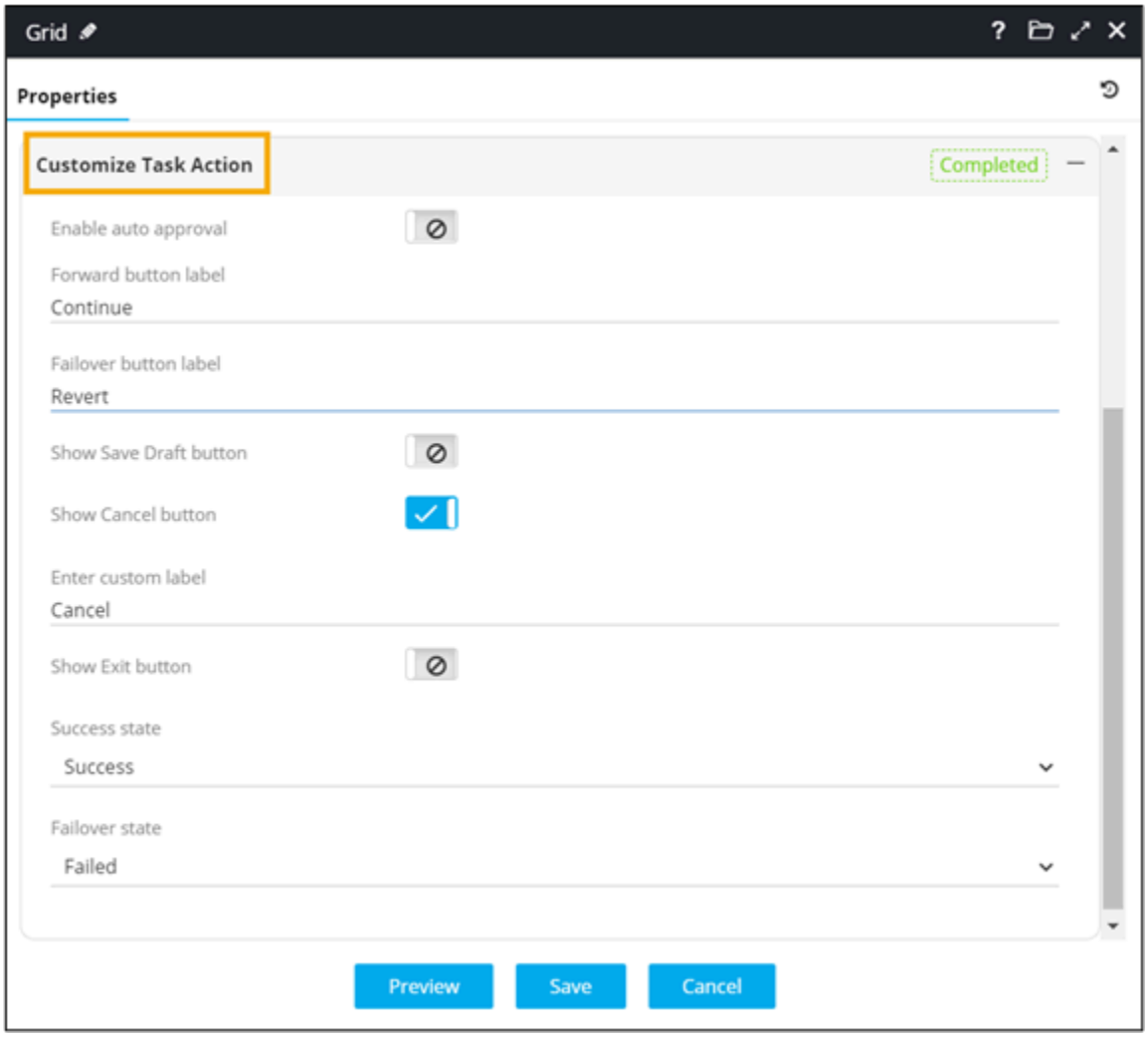


This table describes the fields in this section:

Field	Description
***Name**	Enter a unique display name for the global variable.
***Key**	Enter a unique variable key name.
***Value**	Enter the variable value which is the output or response value of the task.
All asterisk (*) marked fields are mandatory.	

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

7. In the **Grid** task window, under **Properties**, in the **Customize Task Action** section, configure task actions to enable or disable approval.



This table describes the fields in this section:

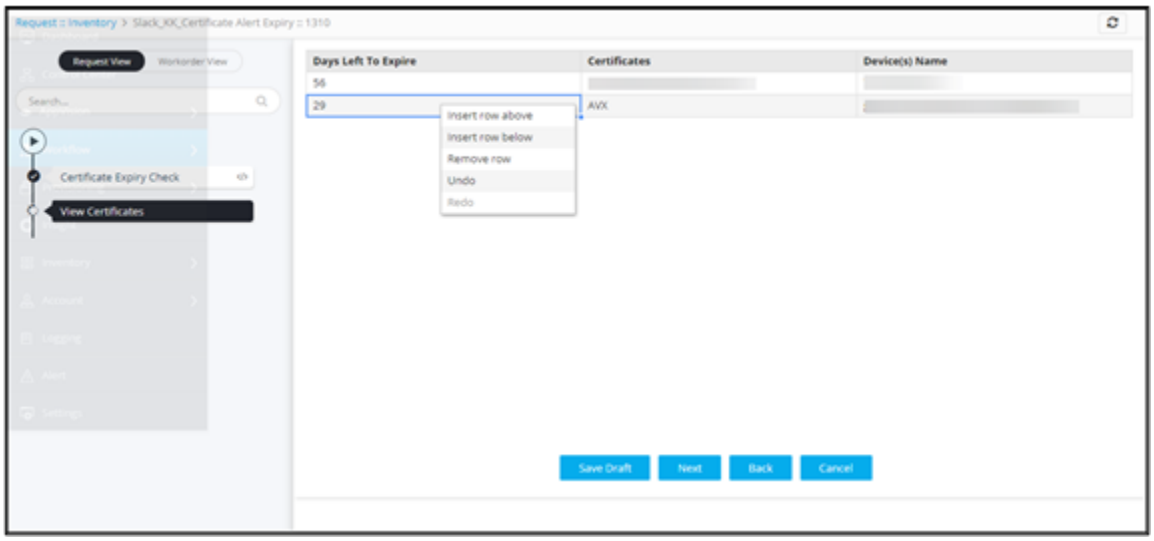
Field	Description
Enable auto approval	Allows you to enable or disable user action on the task.
Forward button label	Allows you to enter a custom label name for the forward button. For example, Next, Continue
Failover button label	Allows you to enter a custom label name for the failover button. For example, Revert, Back

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

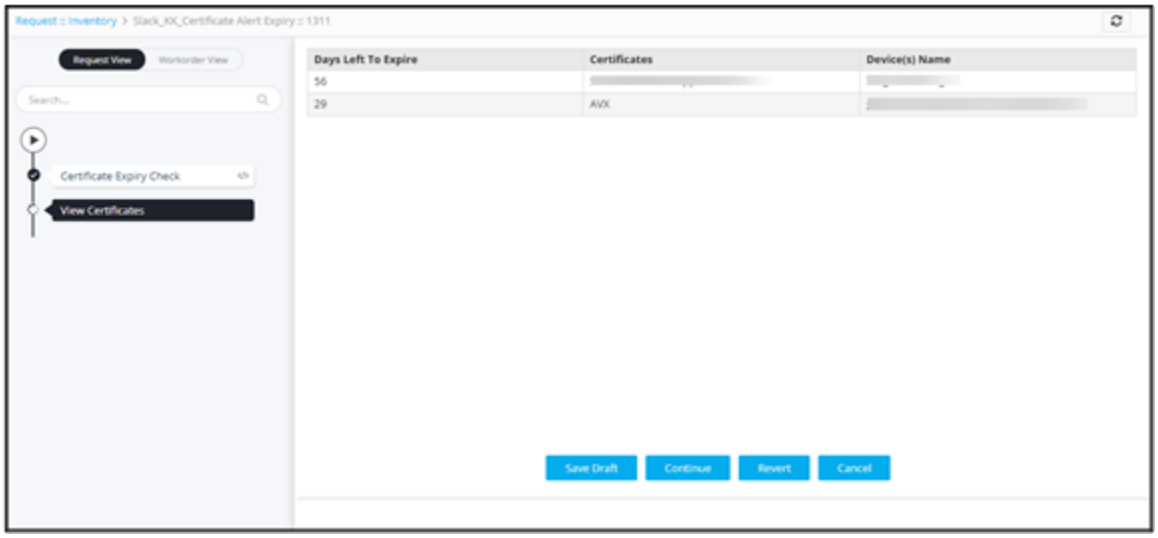
Field	Description
Show Save Draft button	Provision to show or hide the Save draft button which allows you to save the contents of the grid for editing.
Success state	Define custom success state for the task in the event of success. For example, Success, Complete
Failover state	Define custom failover state for the task in the event of failure. For example, Partial, Incomplete

8. Click **Preview**.

   Grid task is executed.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.



- Defining a Grid Component using Variables
- Downloading the Grid

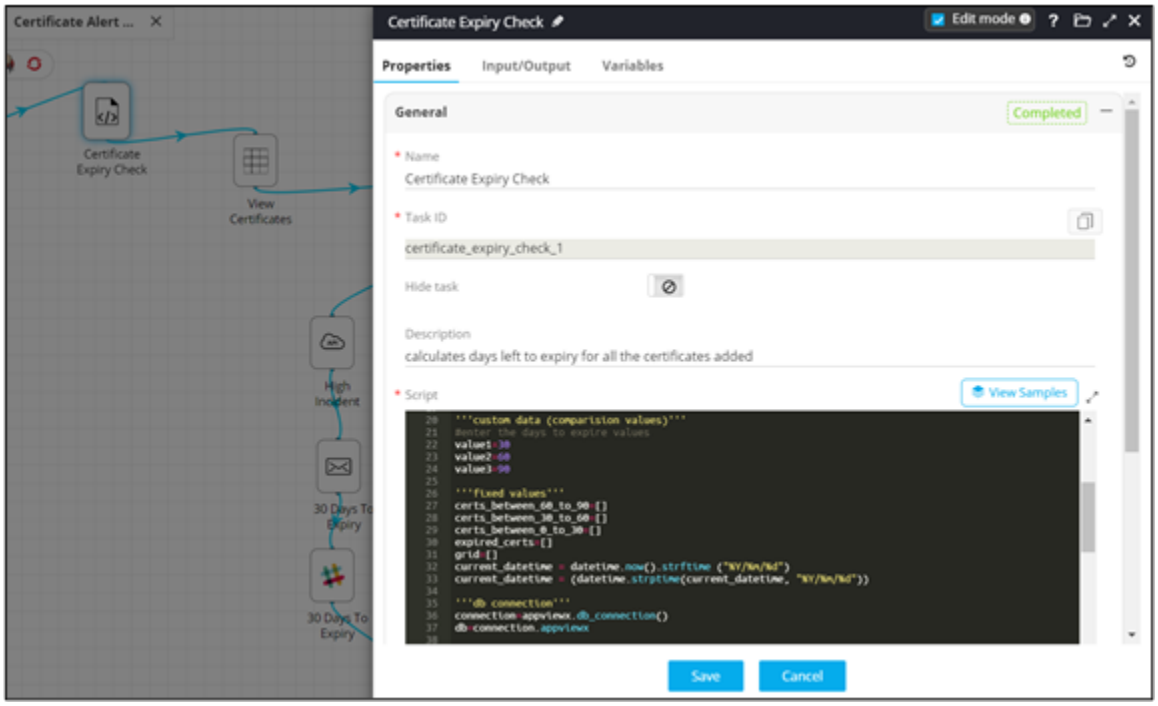
Defining a Grid Component using Variables

A grid component can be defined using either static or dynamic variables.

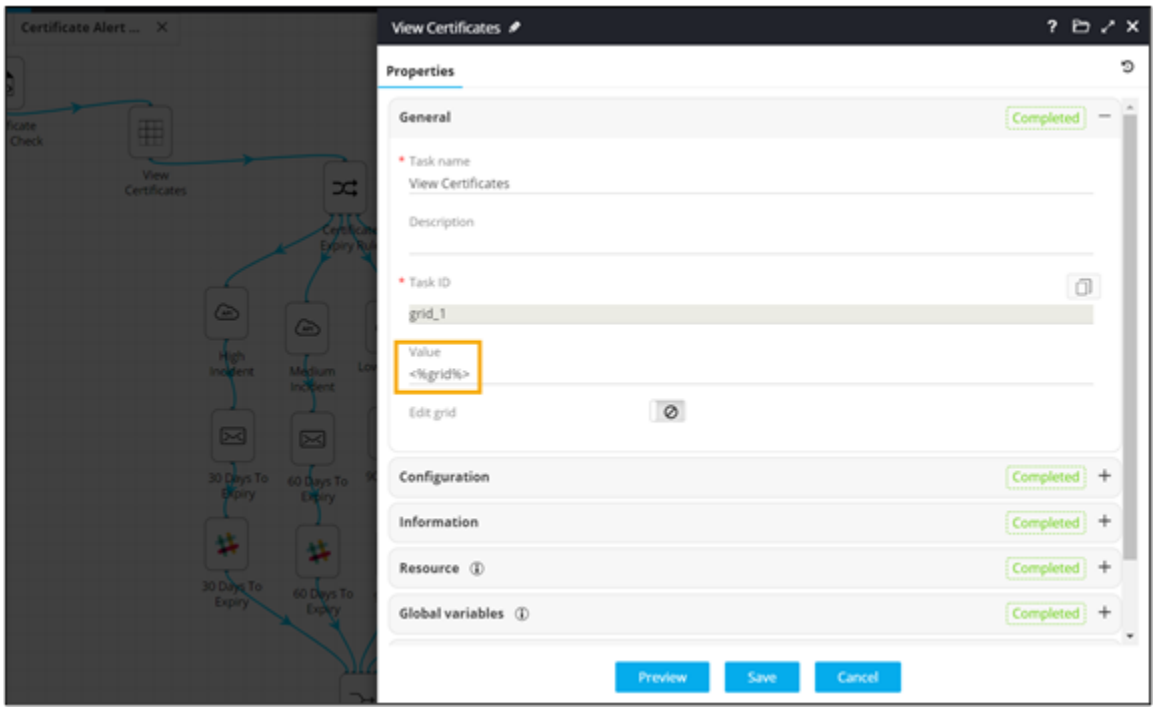
To get a list of certificates expiring within 30, 60, 90 days with device name:

1. Design a workflow.
2. From the **General** section, drag and drop a **Script** task.
3. Define a script to get a list of certificates expiring in 30, 60 , and 90 days, with device name.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



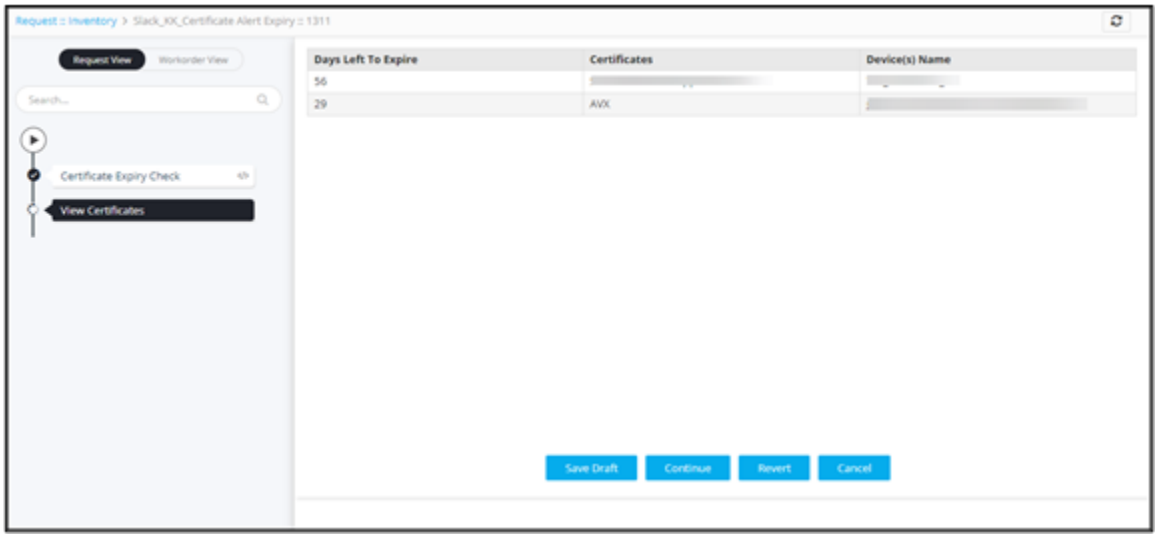
4. From the **User Interface** section, drag and drop the **Grid** task.
5. Refer the variables containing grid values from the previous Script task.



6. Add other relevant workflow tasks.
7. Connect and enable the workflow.

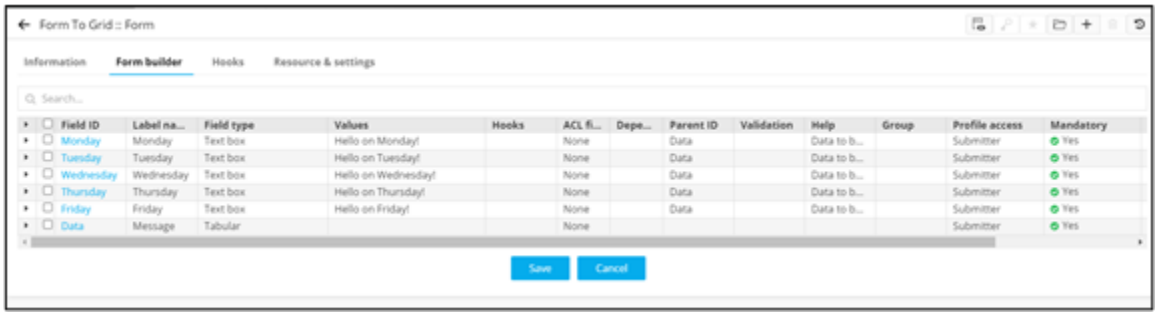
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

8. To view the output in grid view, trigger the workflow from the Request :: View/Runpage.



Downloading the Grid

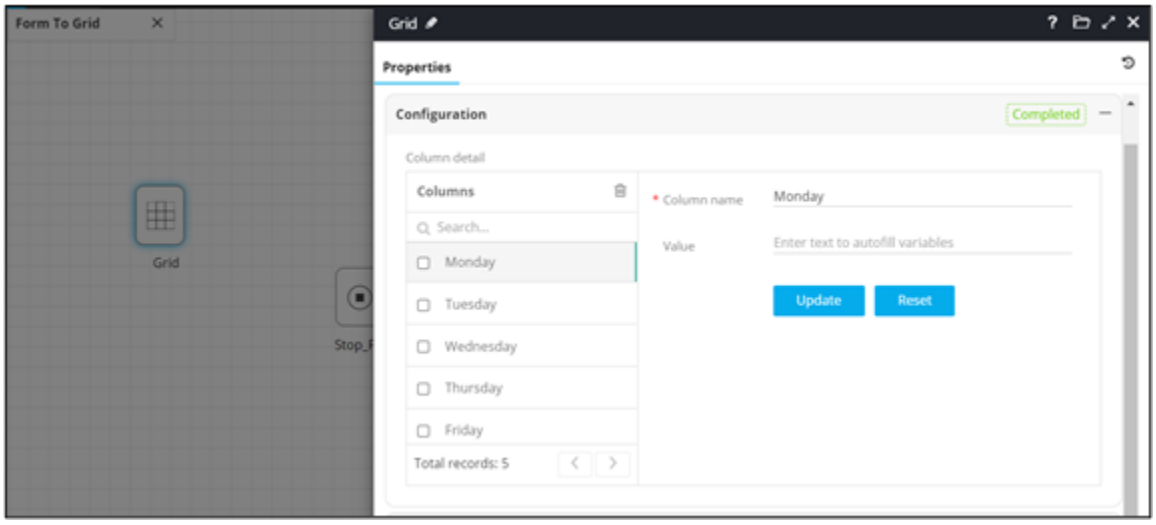
1. Design a new workflow.
2. From the **General** section, drag and drop the **Form** task.
3. Add the required form fields as shown here.



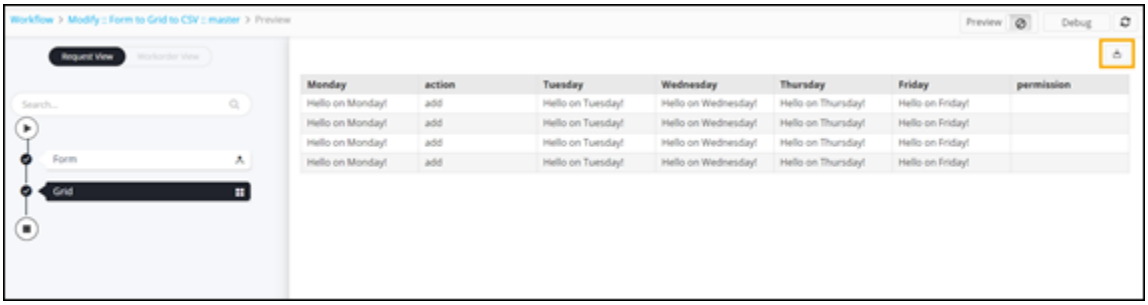
> **Note:** For more information on adding form fields, click here.

4. From the **User Interface** section, drag and drop the **Grid** task.
5. In the **Grid** task window, under **Properties**, in the **Configuration** section, configure the **Columns** for the grid.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



6. Connect the tasks.
7. Click **Preview**.
8. To download the Grid, click in the right upper corner of the screen.



Review

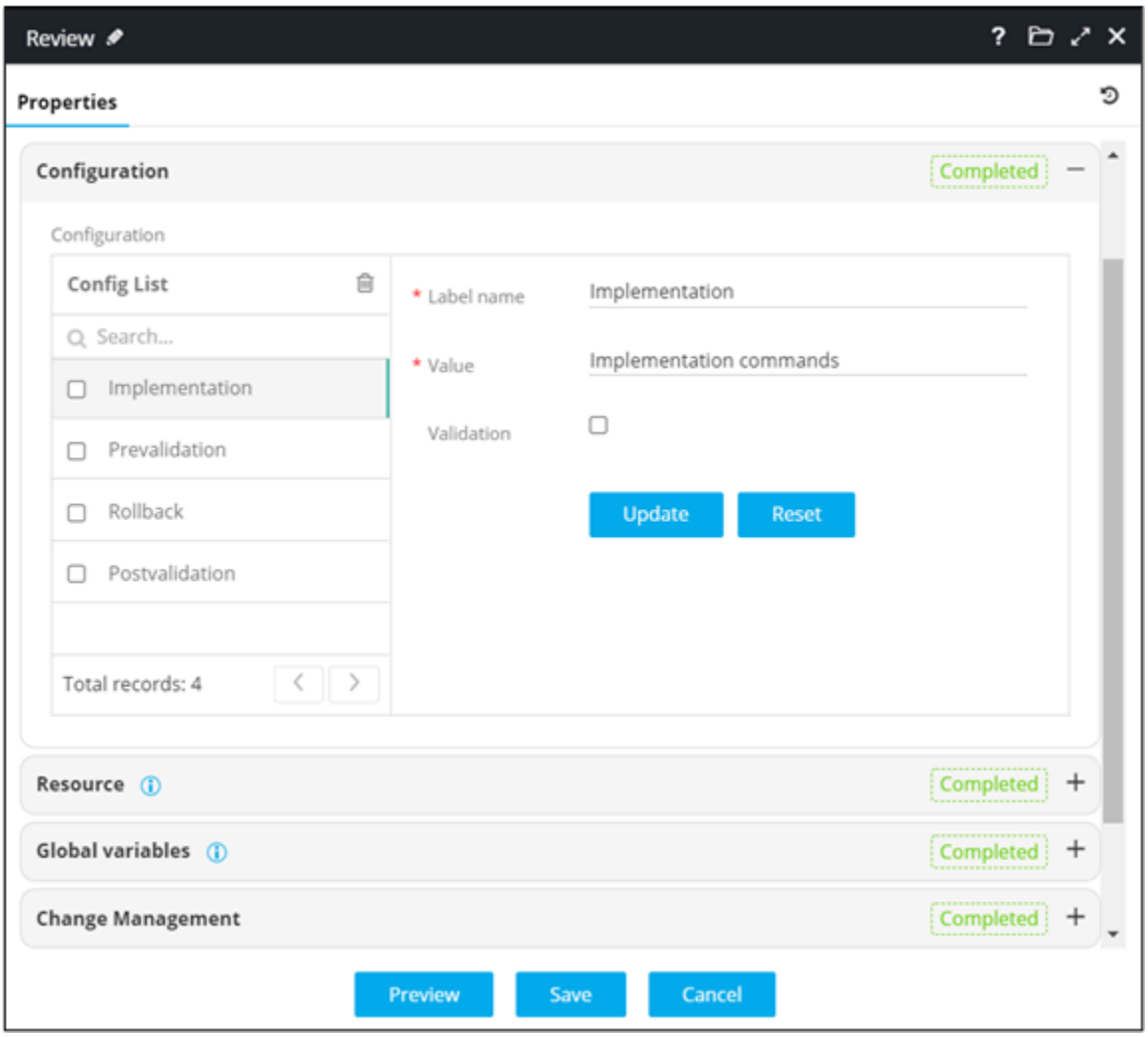
You can define a custom ‘review component’ as part of the workflow automation process. The Review task allows you to enable peer reviews with multiple approval levels. You can review device configurations with Change management integration as part of the approval process.

- Provision to define a custom review or approval component
- Provision to preview the configuration block
- Provision to define custom configuration block and label names in the review component For example, Implementation configuration, Pre-validation configuration, Checkpoint Firewall rule configuration etc.
- Provision to dynamically pass data/value (using variables) from any stage of the workflow into the ‘appropriate configuration block’

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Provision to enable, disable the change management block as part of the request view
- Provision to assign user role (RBAC) access to the review task in the request stage
- Provision to define custom button/label names in the event of using the review task as part of any approval process

For example, Approve, Implement, Reject, Rollback



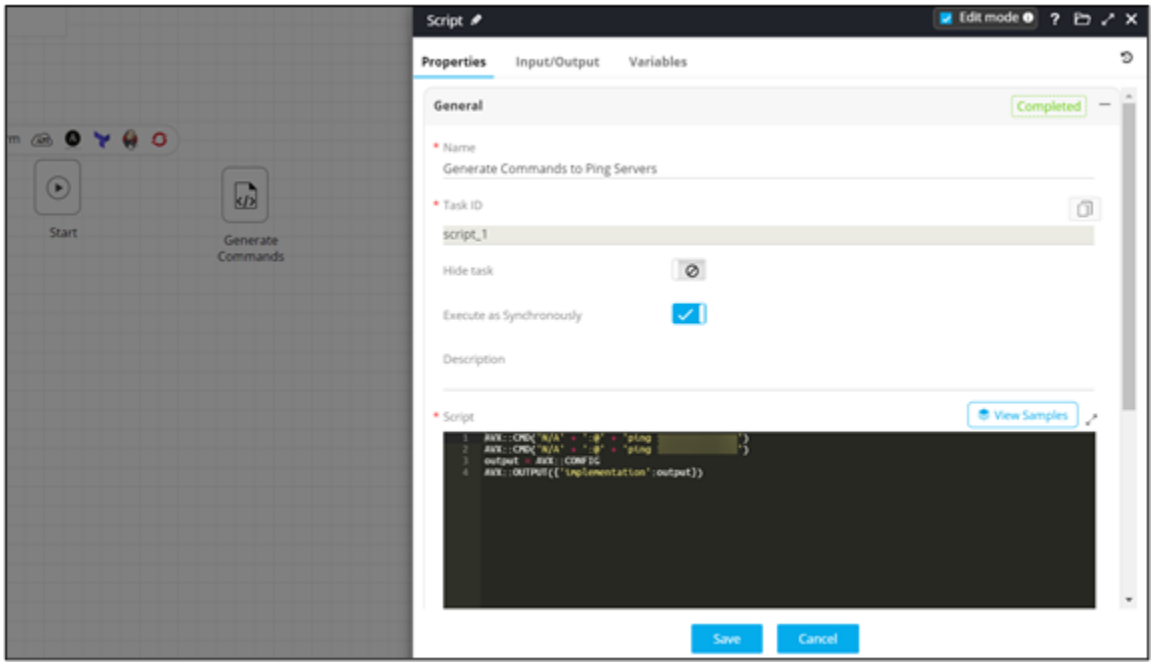
- Defining a Review Task using Variables
- Customizing Configuration Blocks in Review Task

Defining a Review Task using Variables

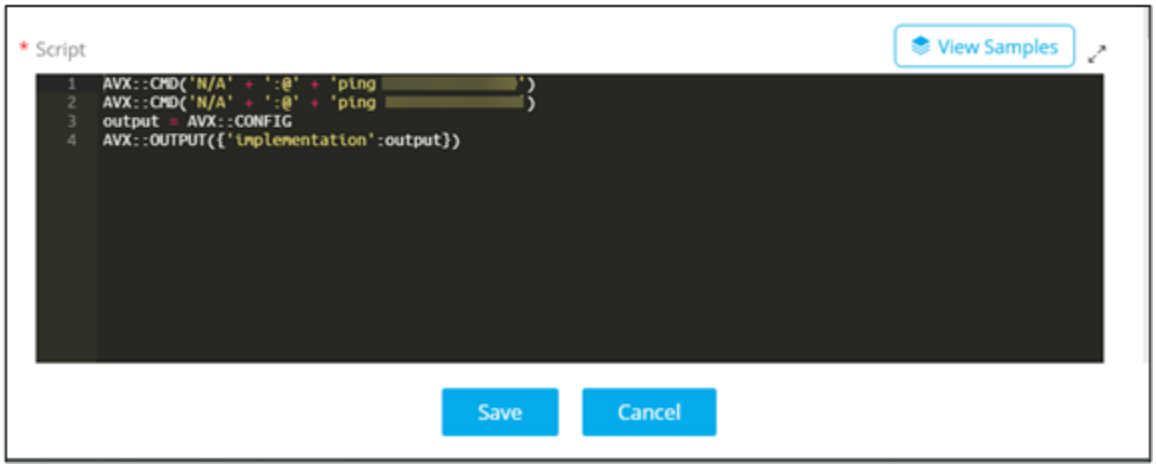
A Review component can be defined using either static or dynamic values.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. Design a workflow.
2. From the **General** section, drag and drop a **Script** task.
3. In the **Script** task window, under **Properties**, in the **General** section, configure the script to generate commands to ping servers.

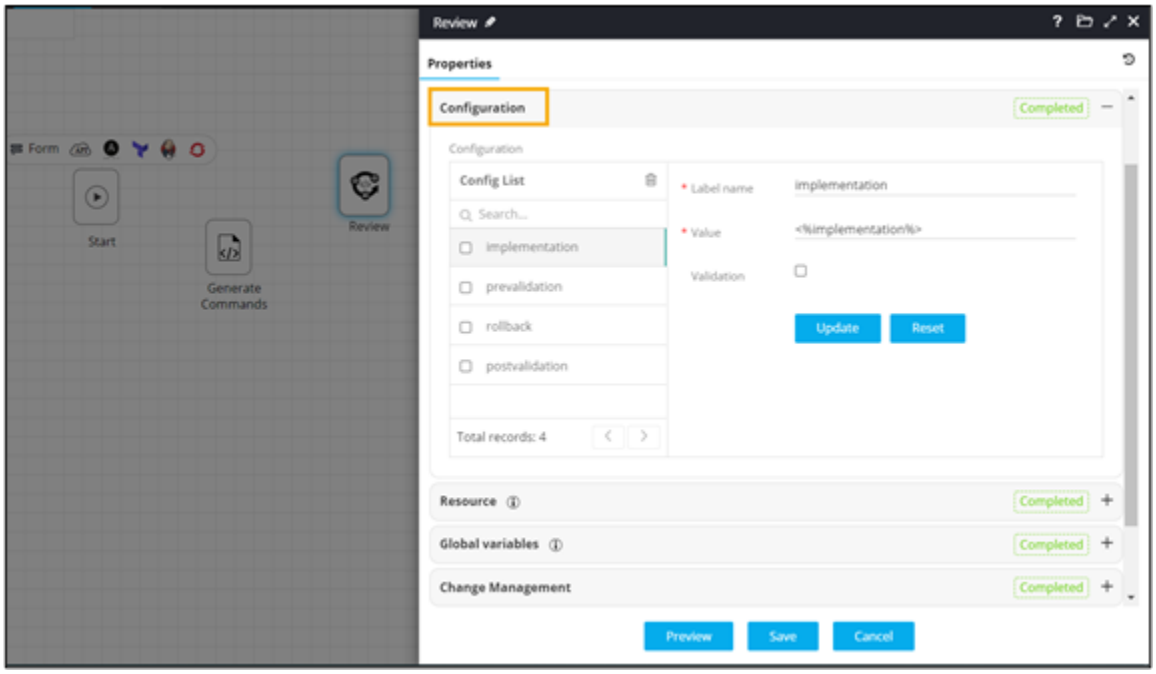


4. Store the config in a variable (AVX::CONFIG).

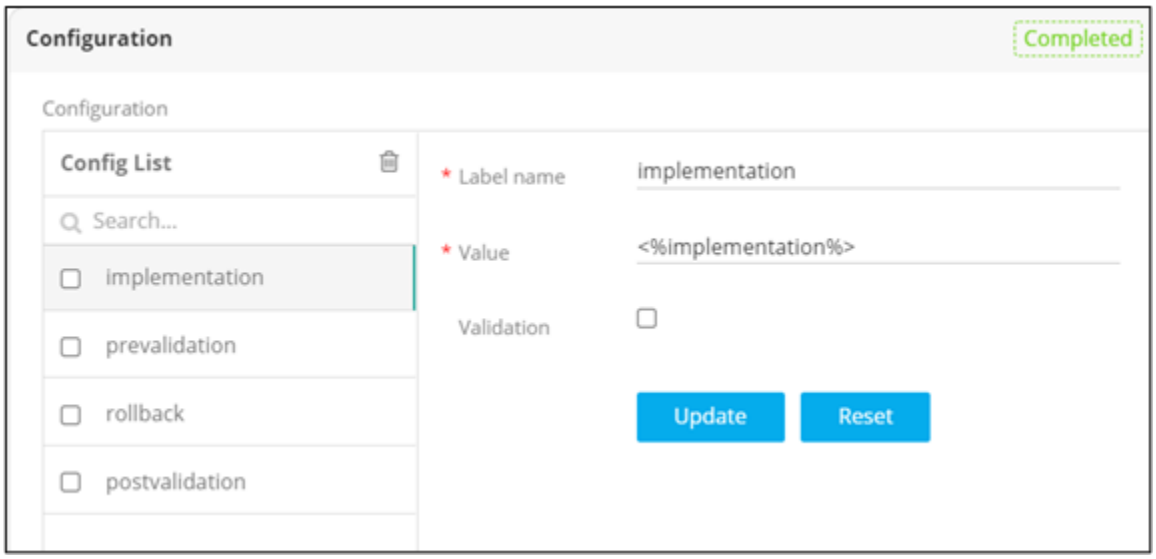


5. From the **User Interface** section, drag and drop the **Review** task.
6. In the **Review** task window, under **Properties**, in the **Configuration** section, define appropriate configuration blocks - Prevalidation, Implementation, Rollback, Postvalidation.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

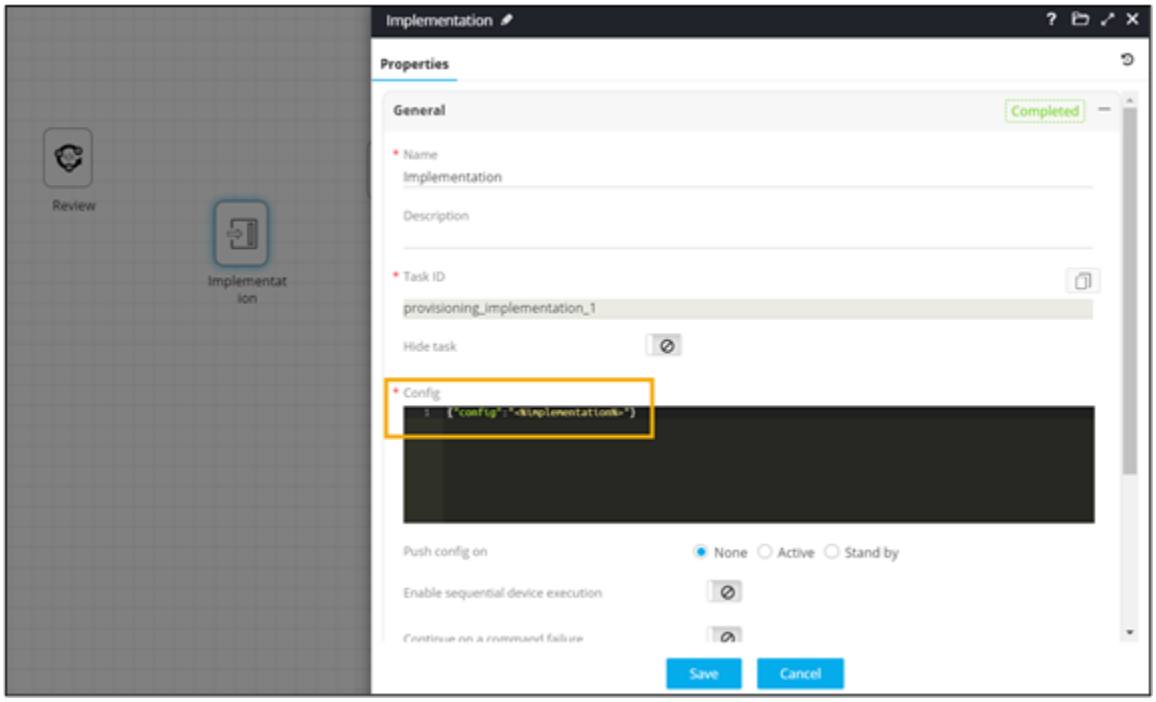


7. Refer the variables from the Script task into the appropriate configuration blocks.

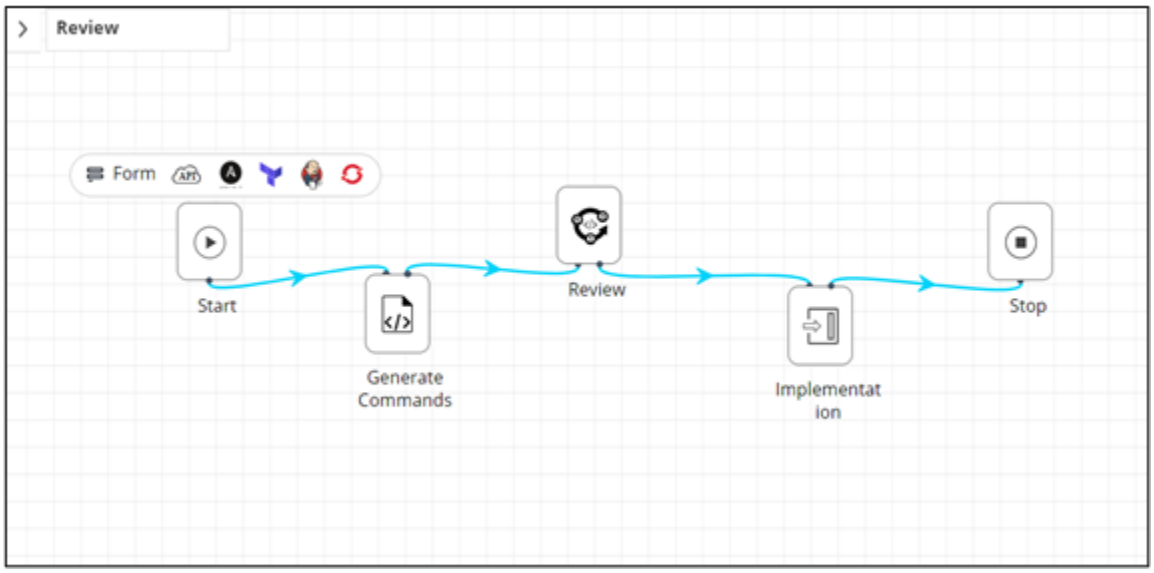


8. From the **General** section, drag and drop the **Implementation** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

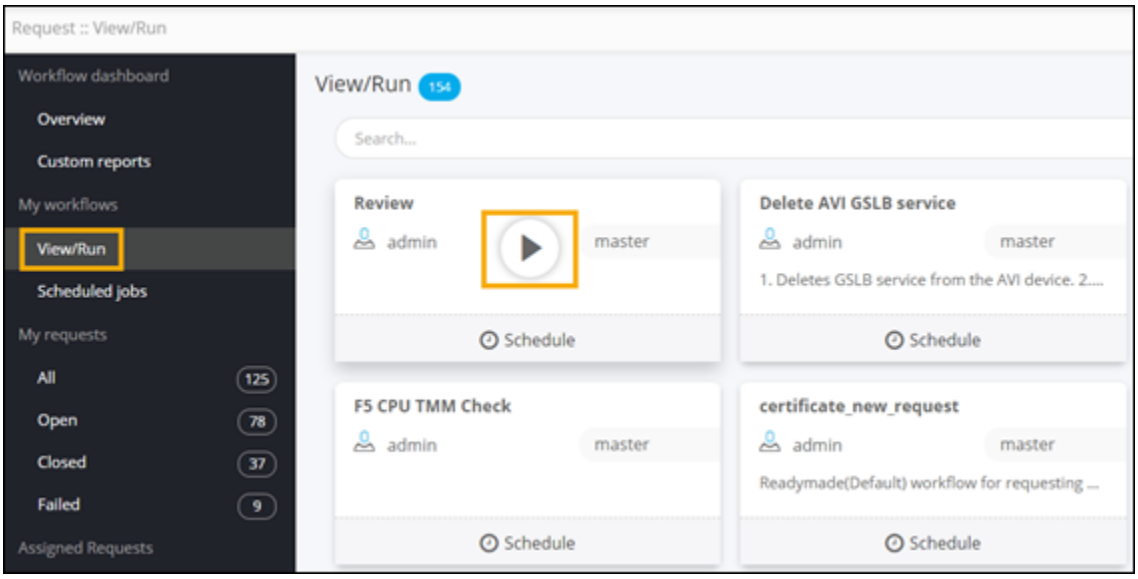


9. Connect and enable the workflow.

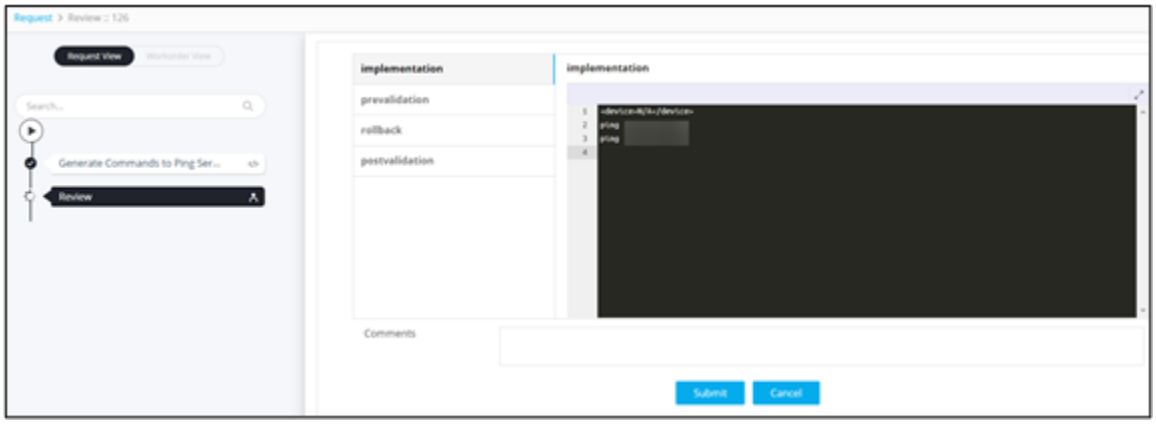


10. Trigger the workflow from the Workflow Request :: View/Run page.

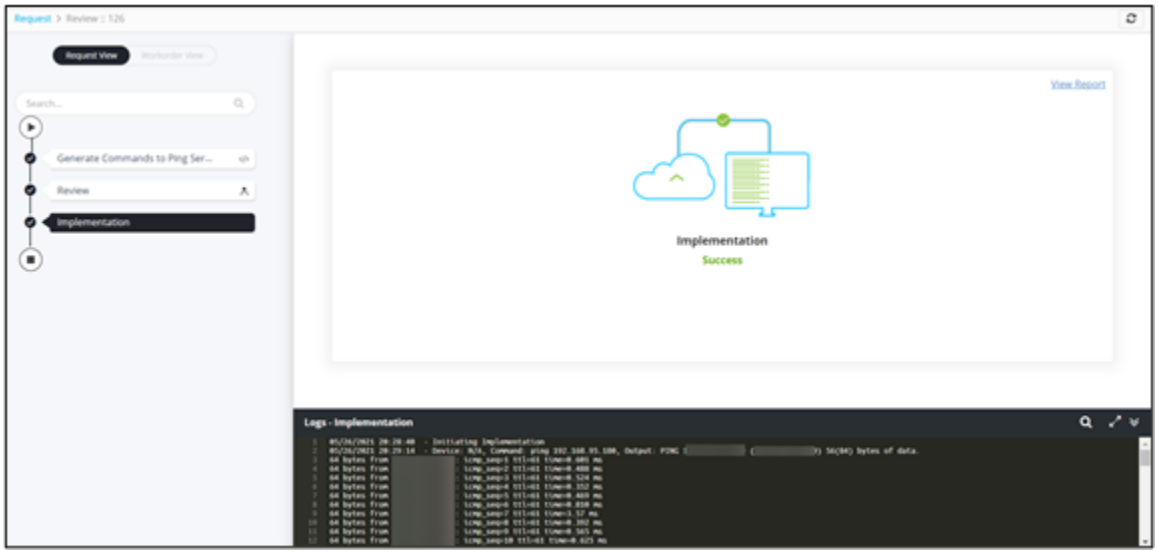
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



View **Review** component with configurations.

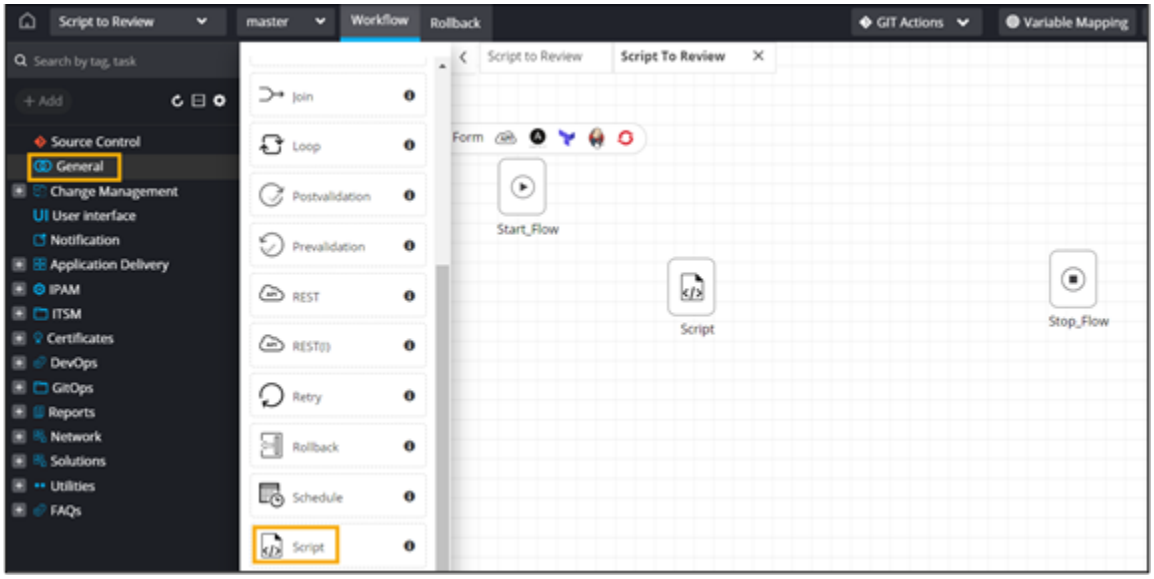


Copyright © 2020 AppViewX, Inc. All Rights Reserved.



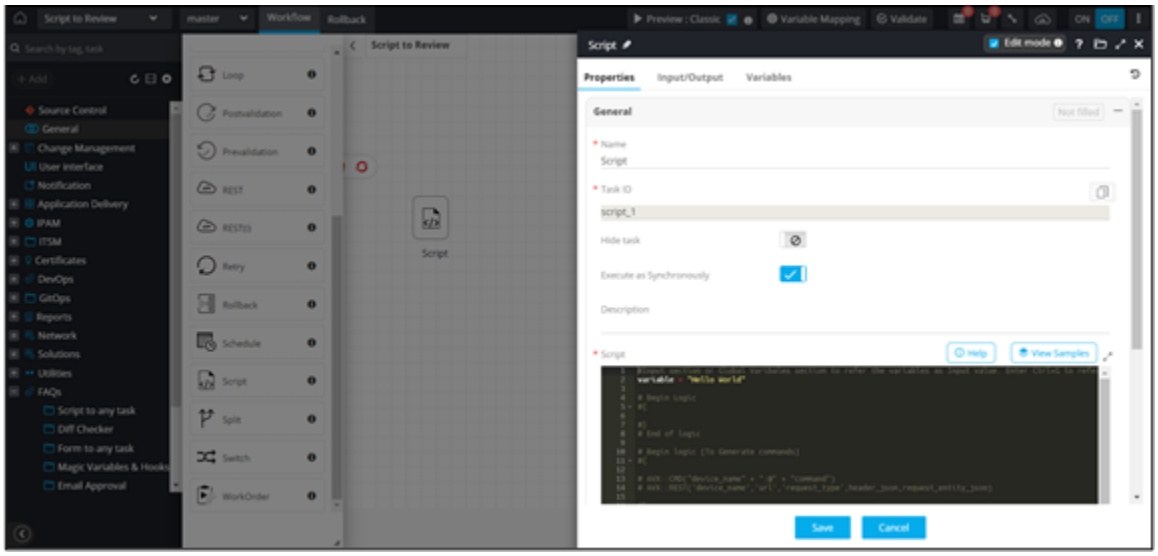
Customizing Configuration Blocks in Review Task

1. Design a workflow.
2. From the **General** section, drag and drop a **Script** task.

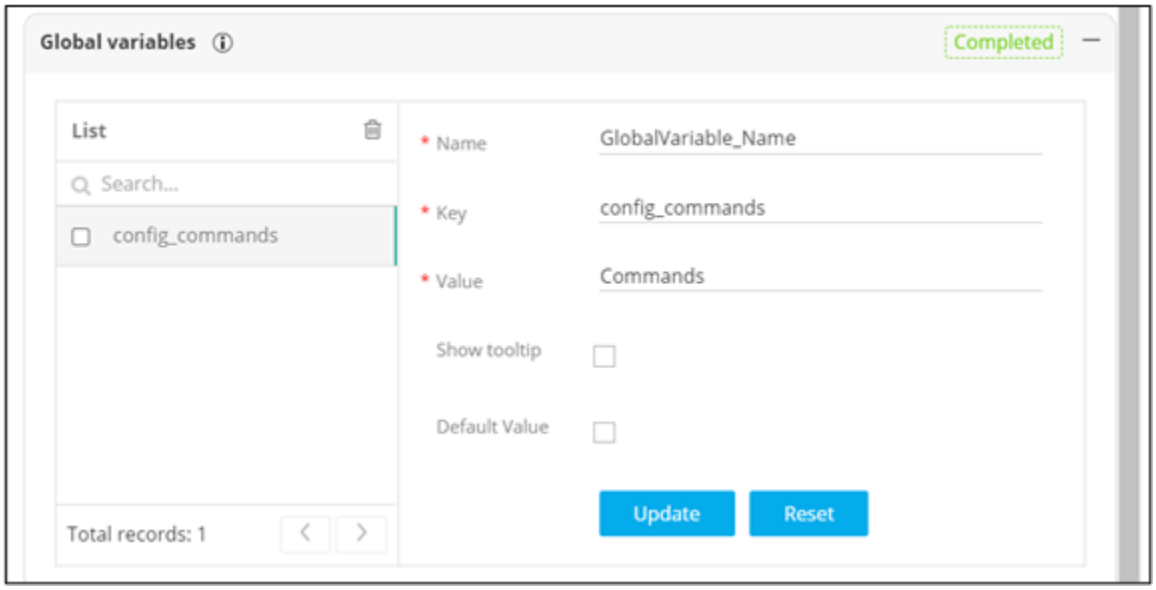


3. Define the **Script** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

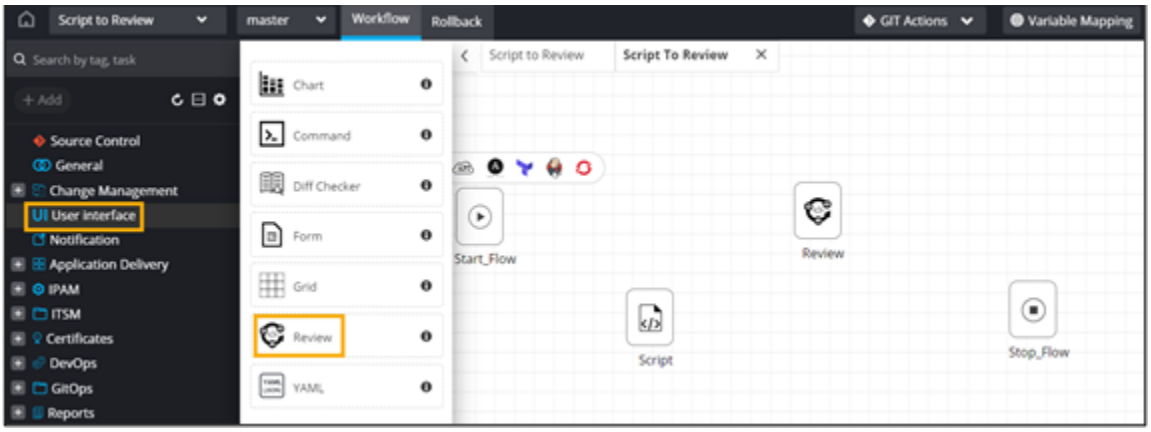


4. In the **Script** task window, under **Global variables**, declare the data to be passed as a global variable to be referenced in the Review task.

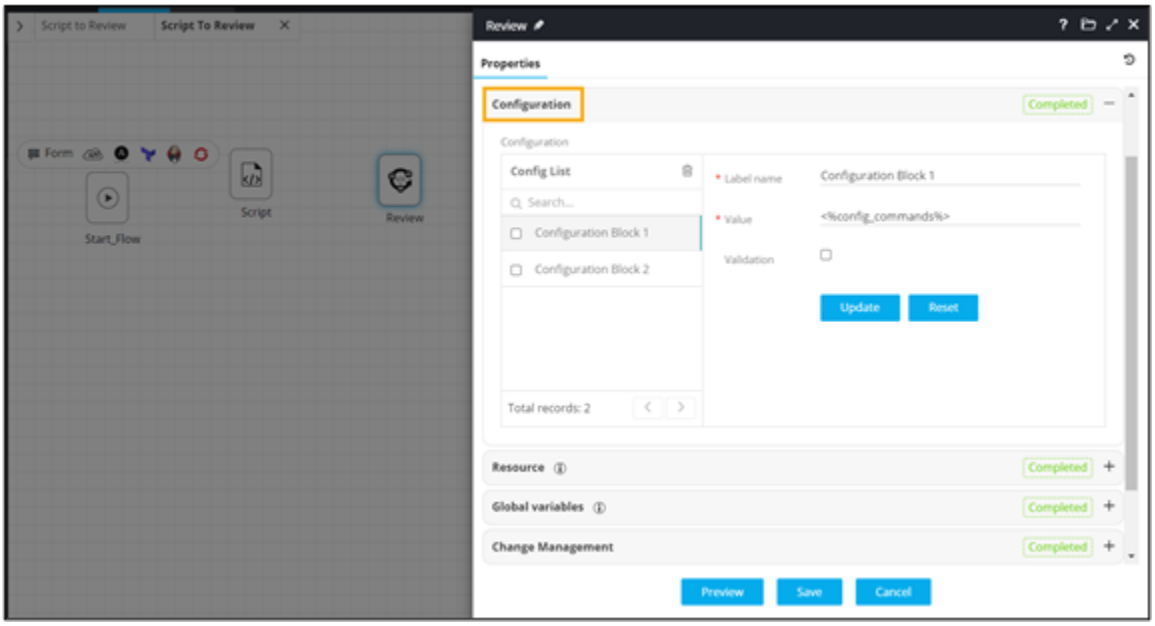


5. From the **User Interface** section, drag and drop the **Review** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

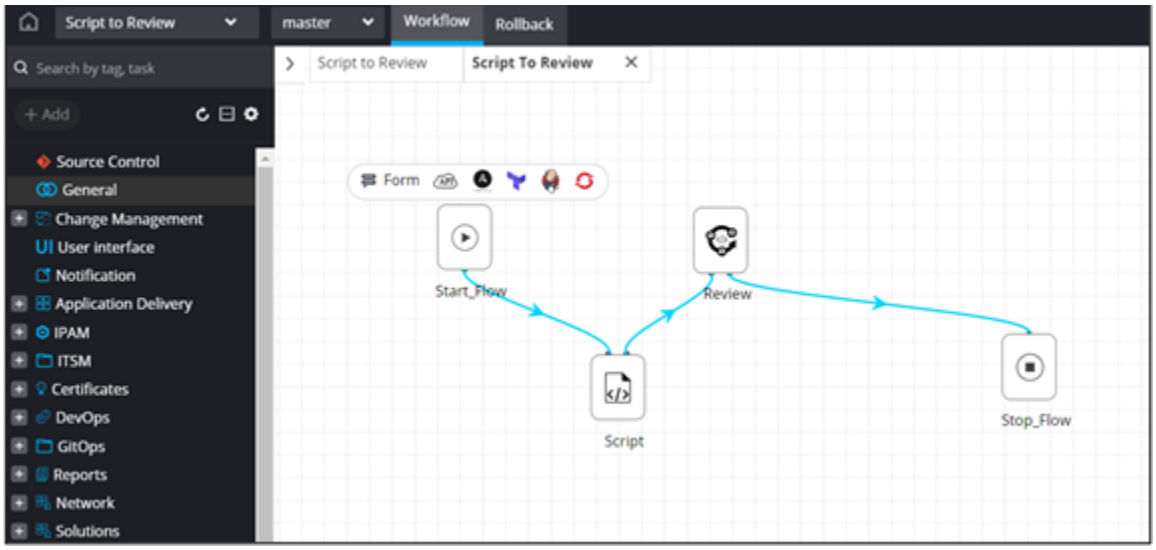


6. In the **Review** task window, under **Configuration** , define custom configuration blocks.
7. Pass variable(s) from the previous task in the Value field against the appropriate configuration block.

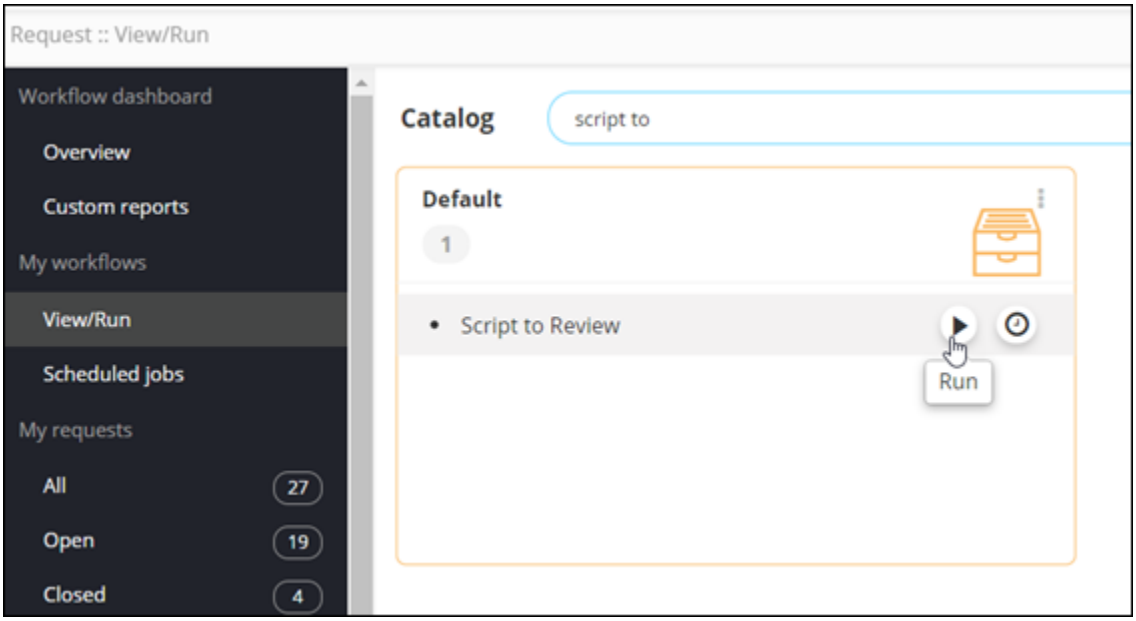


8. Connect and enable the workflow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

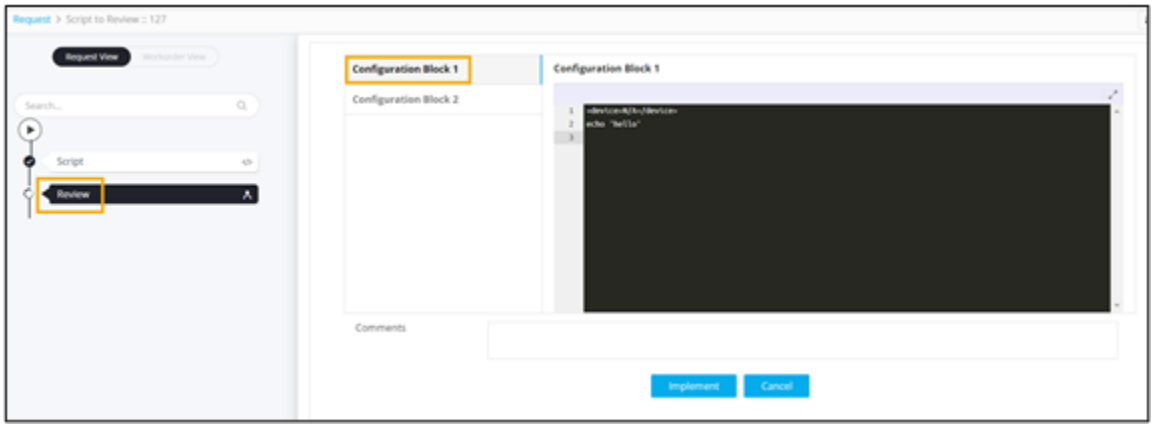


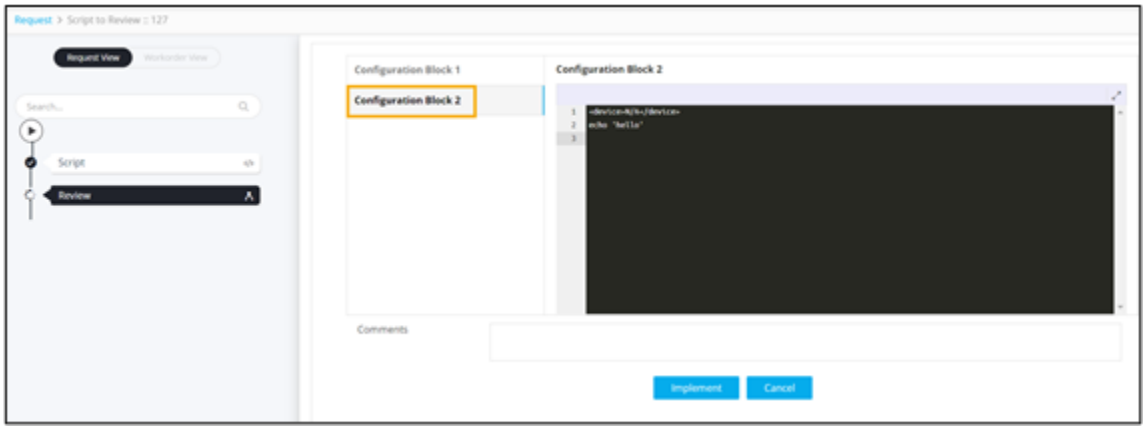
9. Trigger the workflow from the Request :: View/Run page .

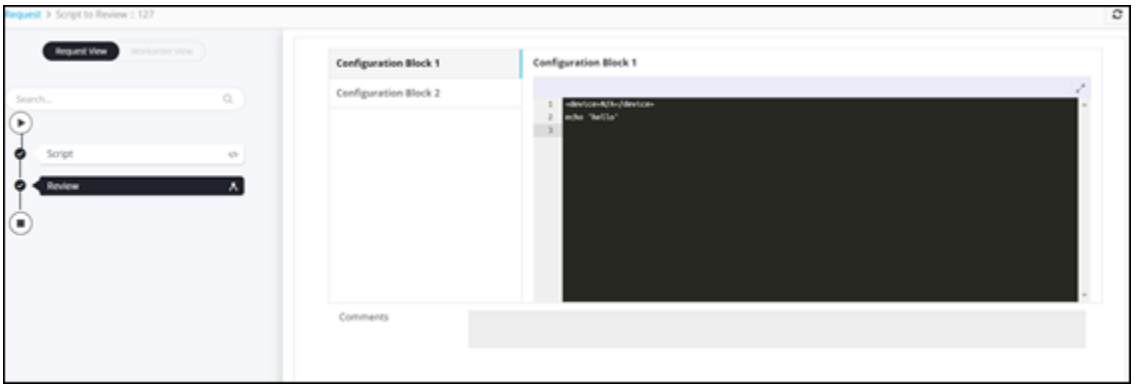


View Review component with variables and configuration blocks.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.





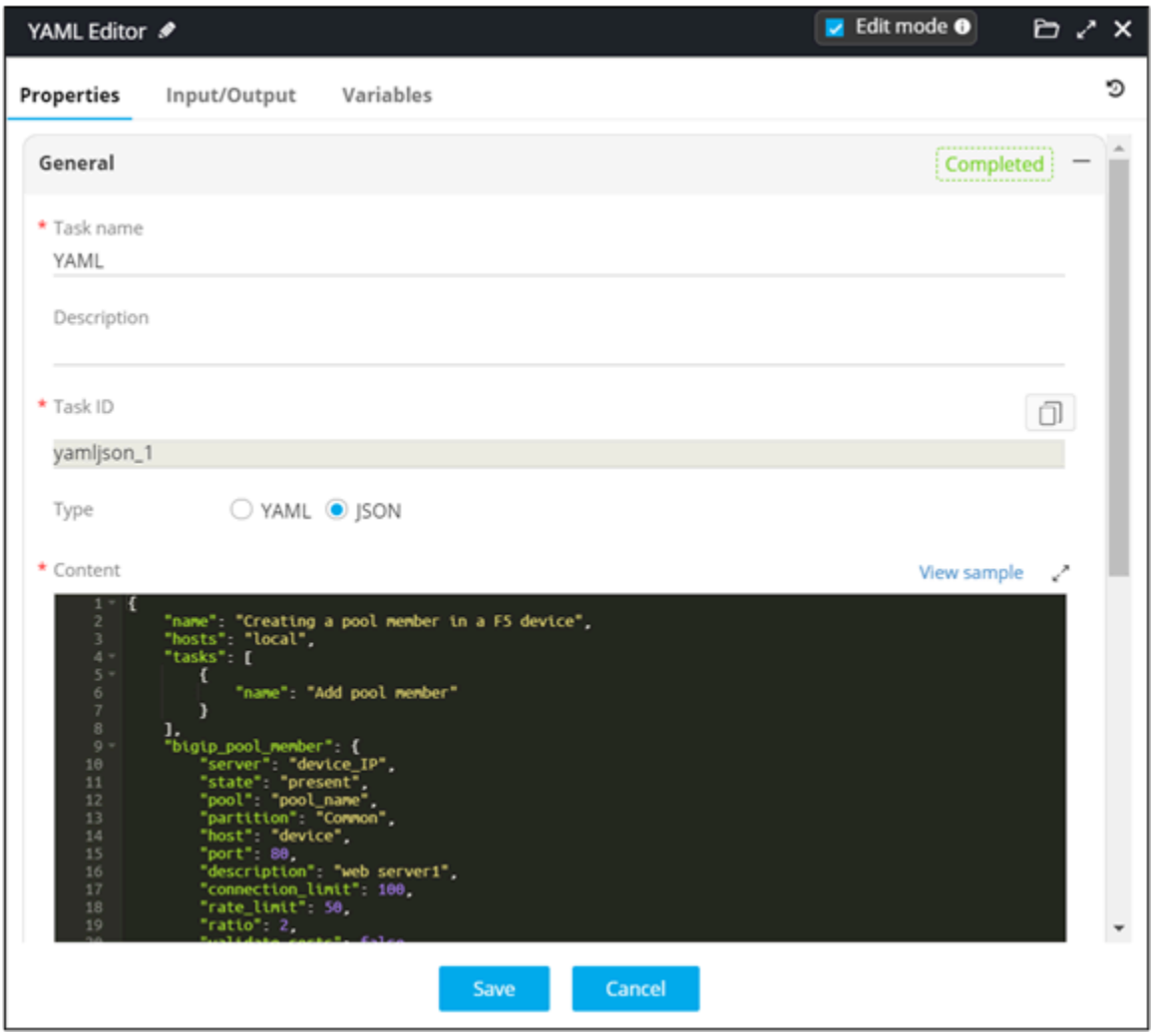


YAML

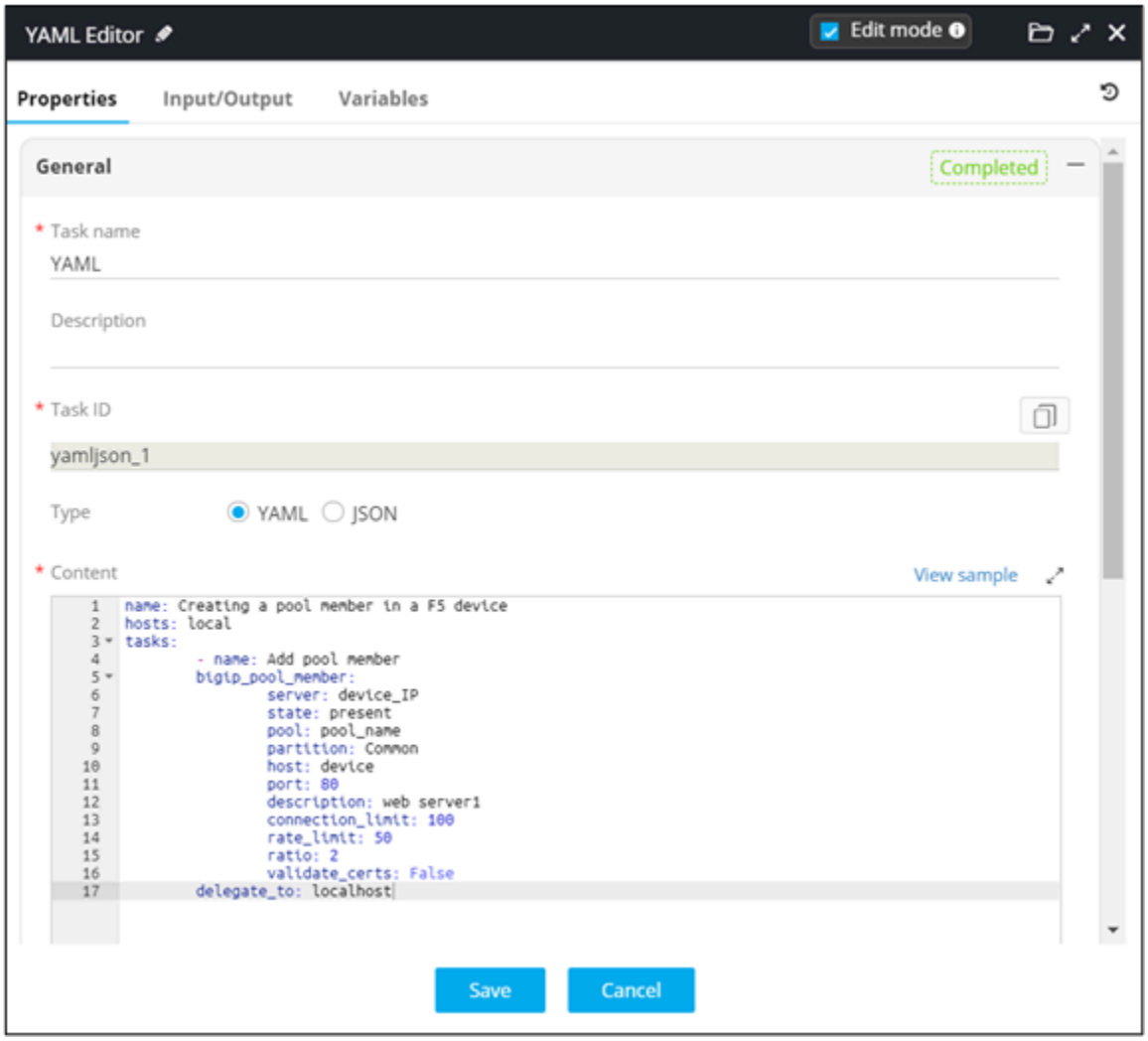
This task allows you to input the workflow data in the form of YAML or JSON content.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Provision to input data in either YAML or JSON format
- Provision to refer YAML or JSON samples
- Provision to use the YAML task as either a user interface task or as a service task
- Provision to edit the contents of the task in the request page
- Provision to assign user role (RBAC) access to the task in the request stage
- Provision to skip manual approvals for the task in the workflow request stage
- Provision to retrieve device credentials dynamically



Copyright © 2020 AppViewX, Inc. All Rights Reserved.



- Using Variables within a YAML Task
- Getting Device Credentials Dynamically from a YAML Task

Using Variables within a YAML Task

Variables from any workflow task can be referred within the YAML task as part of the automation process. Use the following variable syntax in the YAML task to refer value from a previous task: `<%variable_name%>`

Following is an illustration for referring the ‘fqdn’ and ‘FreeIP’ values from an Infoblox task into a YAML task:

```
--
-
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
name: Creating a virtual server in a F5 device
hosts: local
tasks:
    - name: Add virtual server
      bigip_virtual_server:
        server: <%device%>
        user: $$<%device%>.username$$
        password: $$<%device%>.password$$
        state: present
        partition: Common
        name: <%fqdn%>
        destination: <%free_ip%>
        port: 480
        snat: Automap
        description: Test Virtual Server
        validate_certs: no
        all_profiles:
          - http
          - tcp
      delegate_to: localhost
```

Getting Device Credentials Dynamically from a YAML Task

Through the YAML (JSON) task, device credentials can be dynamically retrieved as part of the workflow execution process. The following syntax must be in the task in order to get device credentials:

- Get device detail:

```
$$devicename.username$$
```

- Get device credential:

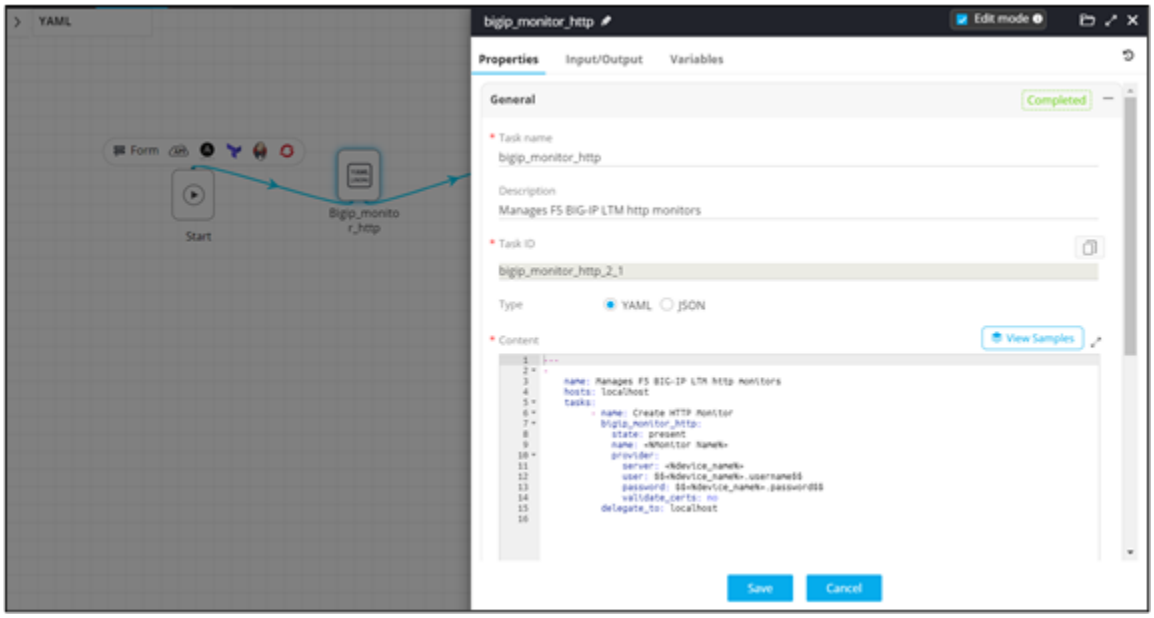
```
$$devicename.password$$
```

Note: Device(s) must be added in the Appviewx inventory.

To get the device credentials dynamically:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

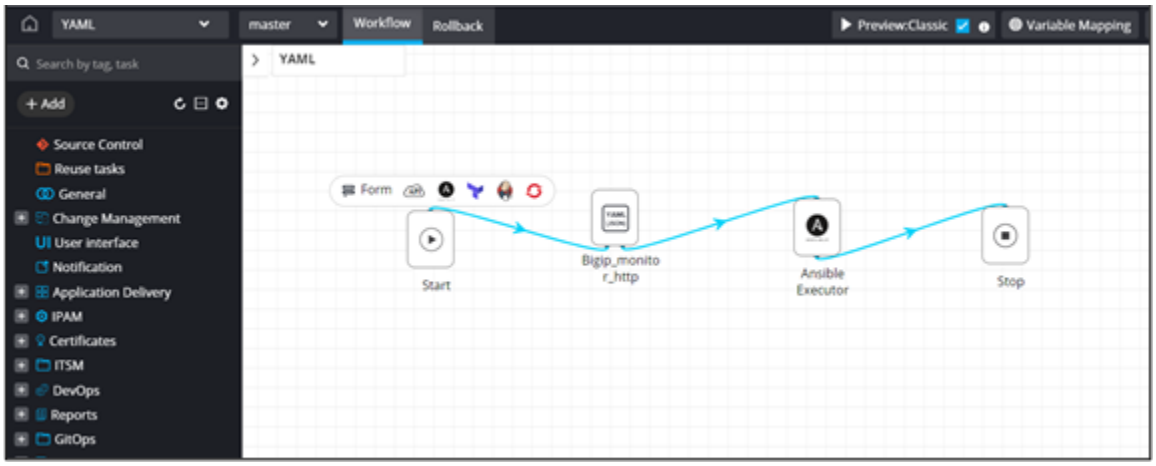
1. Design a new workflow.
2. From the **User Interface** section, drag and drop the **YAML** task.



3. Enter the syntax to get device credentials dynamically.

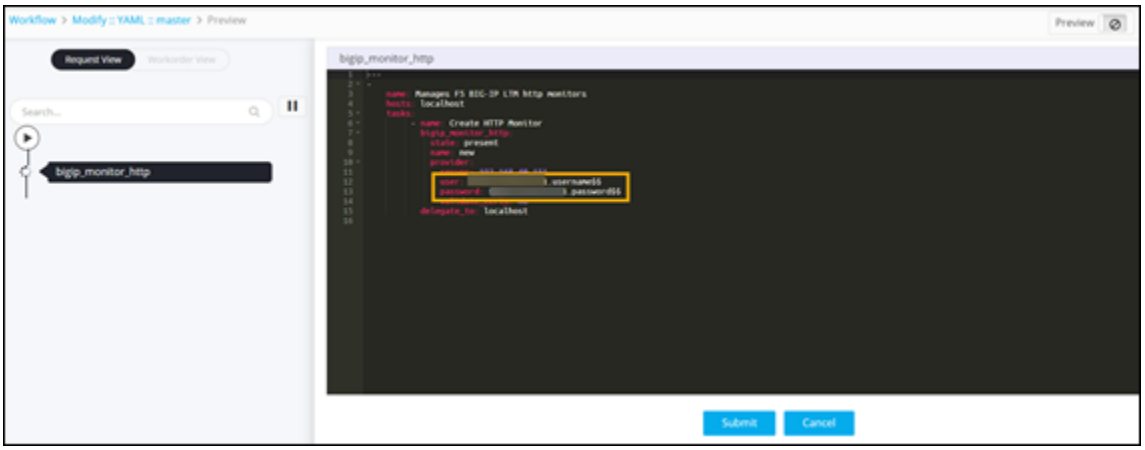
4. Save and enable the workflow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



5. Trigger the workflow from the Request :: View/Runpage.

   Device credentials are dynamically pulled from the Inventory.



Task Category - ChatOps and Notifications

This category comprises tasks that are used to send out notifications to the user for receiving inputs, approvals/rejections, and task completion.

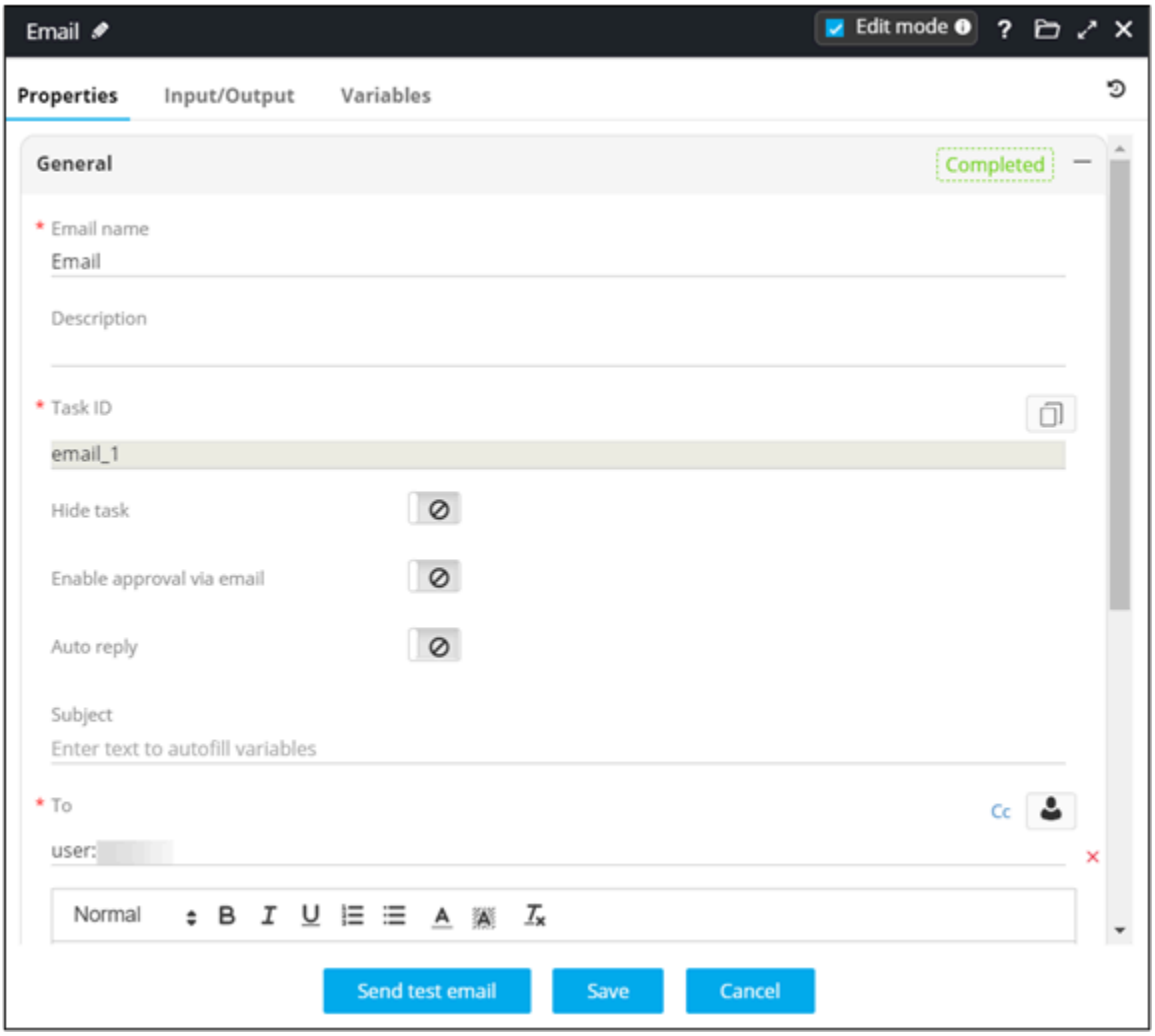
- Email
- Pagerduty
- Slack
- Skype

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Email

Email task allows you to configure static or dynamic email notifications as part of the workflow automation process. You can send email attachments in different formats. This task allows you to configure email approvals to automatically trigger the workflow process.

- Provision to select user(s), user role(s), and user group
- Provision to send test email to ensure that the defined content is relevant
- Provision to reuse email tasks from one workflow into another
- Provision to dynamically render emails by referencing data/value (using variables) from any stage of the workflow into the email task
- Variables can be referenced into the following fields – Subject, To, Cc, Content
- Provision to trigger workflow based on email approvals



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Configuring Dynamic Email Attachments
- Configuring Email Approvals
- Receiving Approvals via Email
- Triggering an Email Notification with Approvals
- Customizing Email Notification

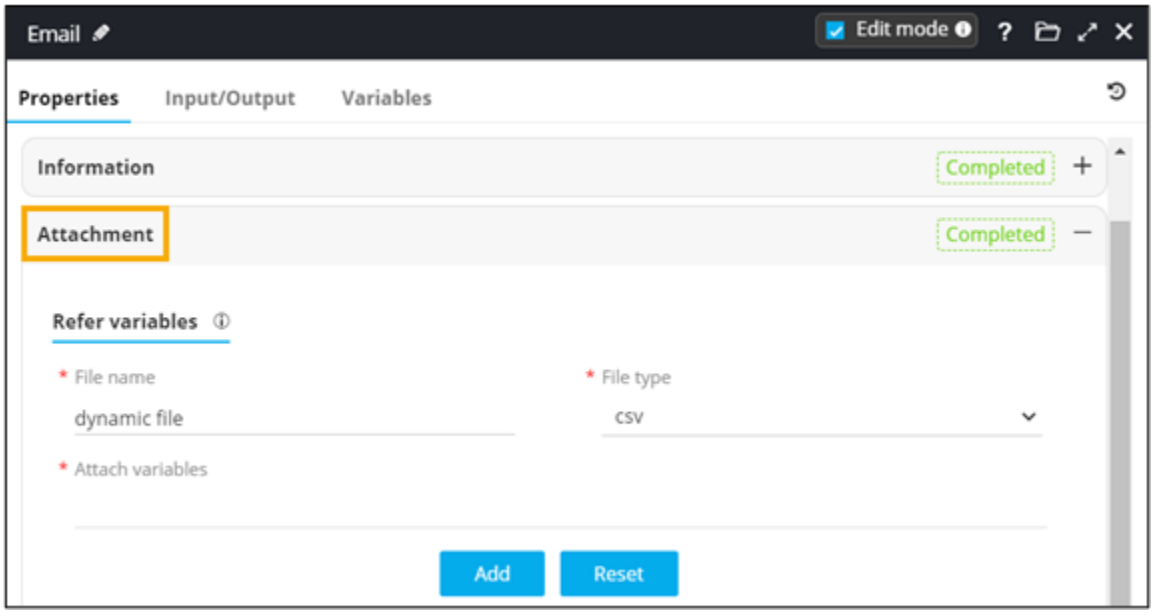
Configuring Dynamic Email Attachments

You can attach static files or dynamically generate file(s) from any workflow stage as part of the email.

- Provision to refer variable(s) from any stage and dynamically generate file attachments.
- Provision to choose the file type for dynamically generating file attachments (Supported file formats are - pdf, csv, txt).
- Provision to upload/add one or more file attachments from the system with a limit up to 25 MB.
- Provision to delete attachment(s) from the email task.

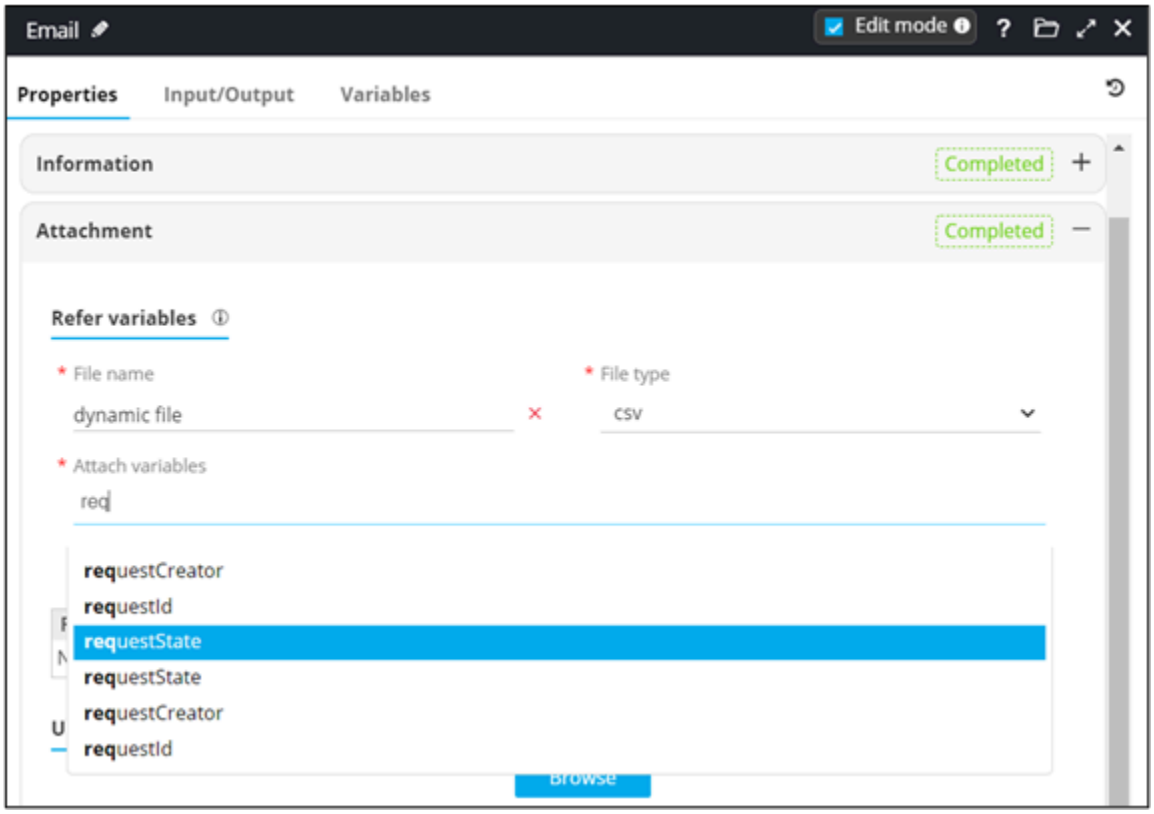
To send an attachment:

1. Design a workflow.
2. From the **Notification** section, drag and drop an **Email** task.
3. In the **Email** task window, under **Properties**, in the **Attachment** section, enter or select the field information.



4. Attach variables to add a dynamic email attachment.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



5. Click **Add**.
   You can use this task within your workflow to send email attachments dynamically.

Configuring Email Approvals

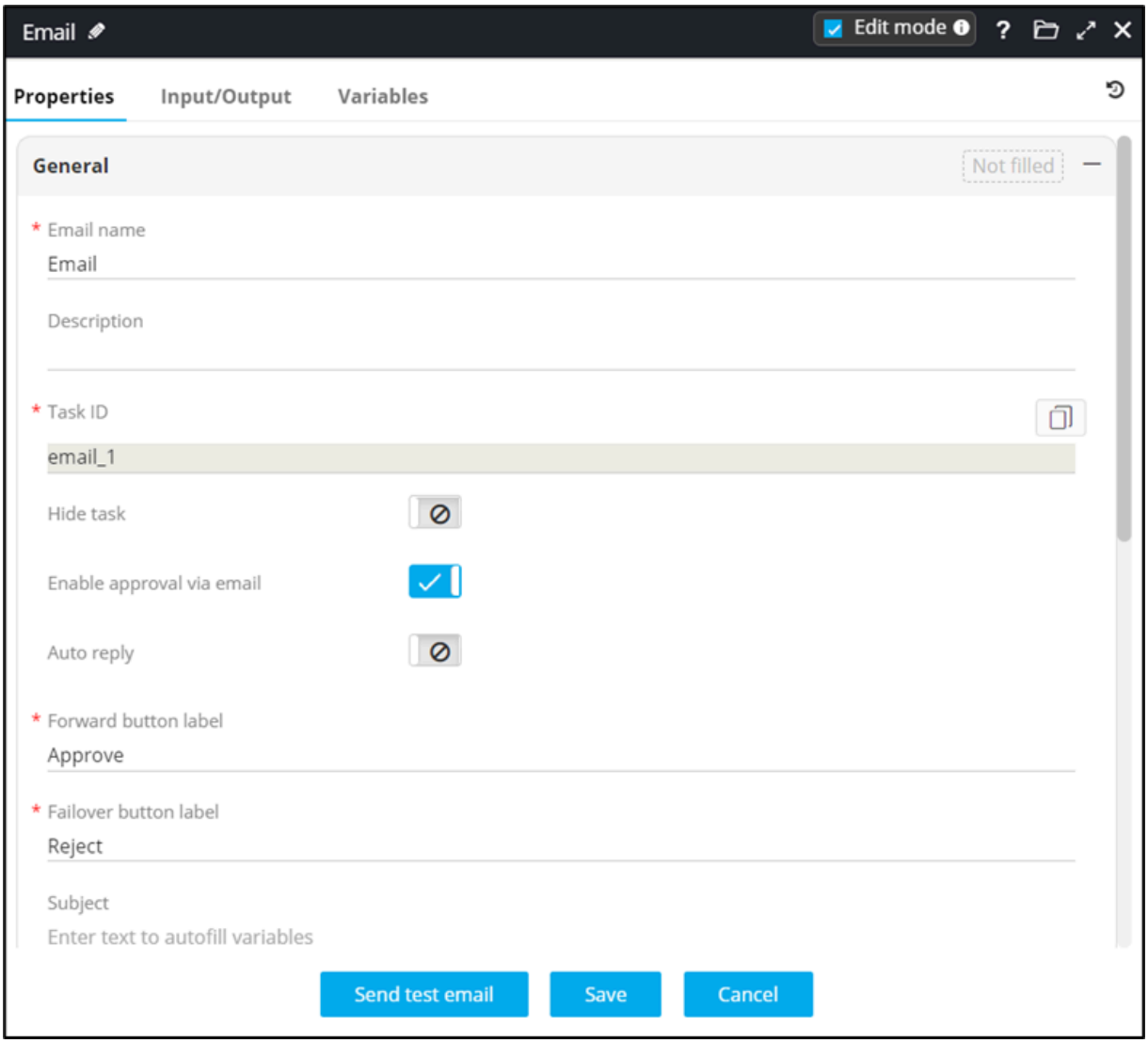
Approvals/peer reviews in the network automation process can be notified and approved directly via email without having to log into AppViewX. You can review config details, and approve or reject based on which the workflow will be triggered further.

- Configure approvals via email and trigger the workflow based on the email event or action.
- Define manual or automatic email approvals.
- Emails include static and dynamic file attachments (pdf, jpeg, doc, xls)

To enable email approval:

1. Design a workflow.
2. From the **Notification** section, drag and drop the **Email** task.
3. In the **Email** task window, under **Properties**, in the **General** section, turn on the **Enable approval via email** toggle.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



On enabling the toggle, the following fields are displayed:

- **Forward button label**
- **Failover button label**

4. Enter values for these two fields.
5. Enter custom email content.
6. Click **Save**.

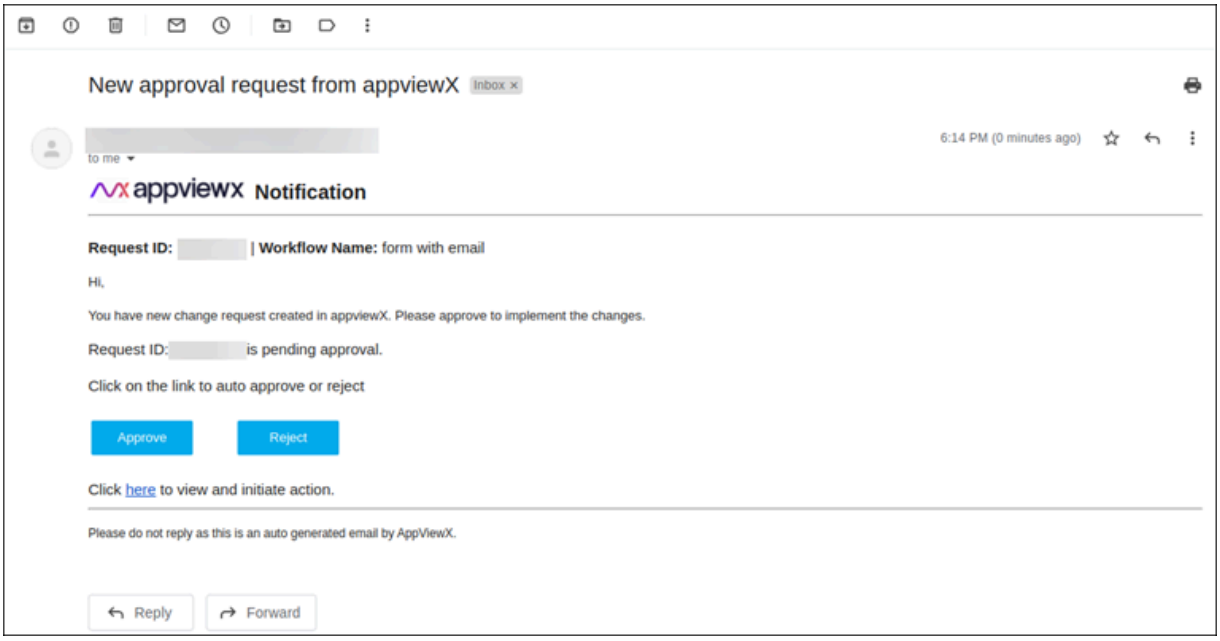
   You can use this task within your workflow to send email approvals.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

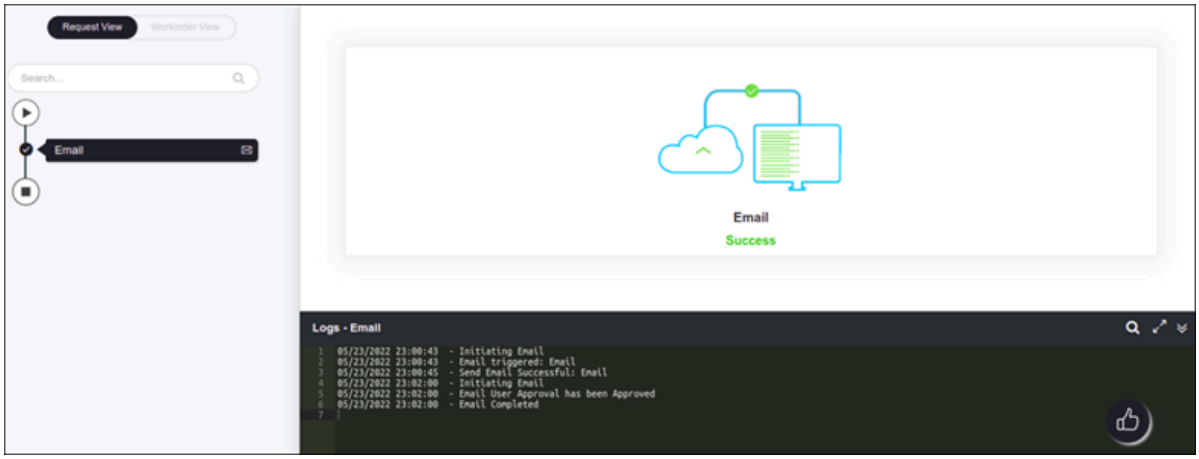
Receiving Approvals via Email

Once the email task to trigger approvals is configured, you can review and approve or reject the workflow automation process directly via email.

1. Click on the link (embedded in the approval email) to navigate to AppViewX.

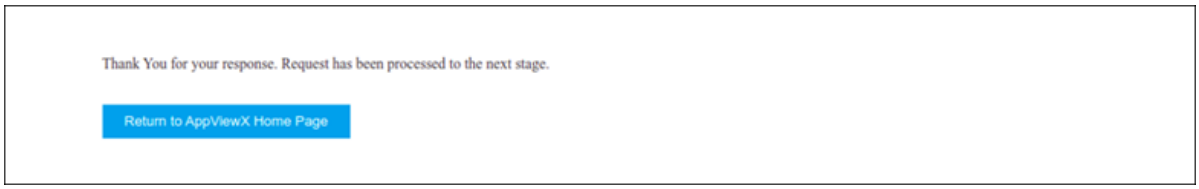


2. Click **Approve**.
   - The following message (HTML text) will be shown in a new tab.

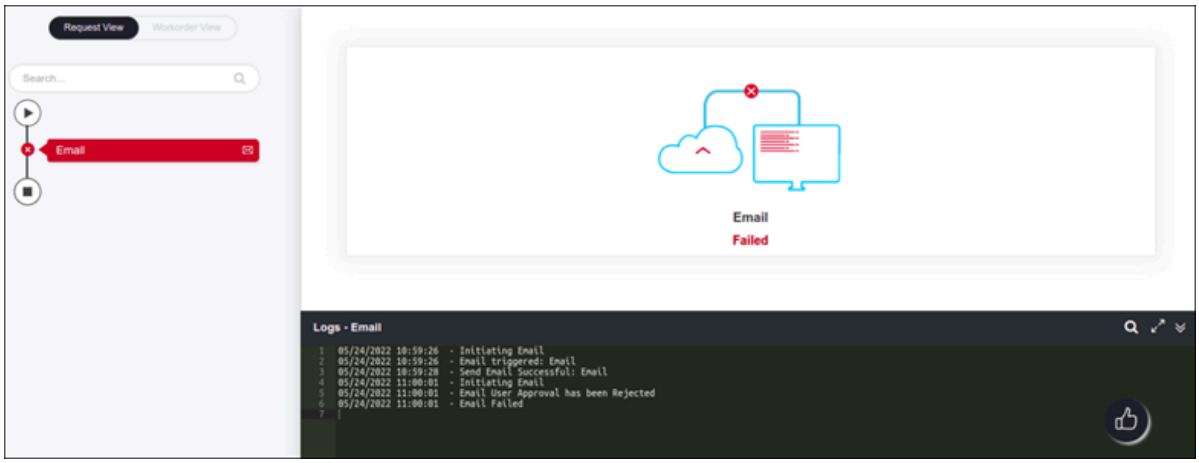


   - If you click **Reject**, the following message (HTML text) will be shown in a new tab.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



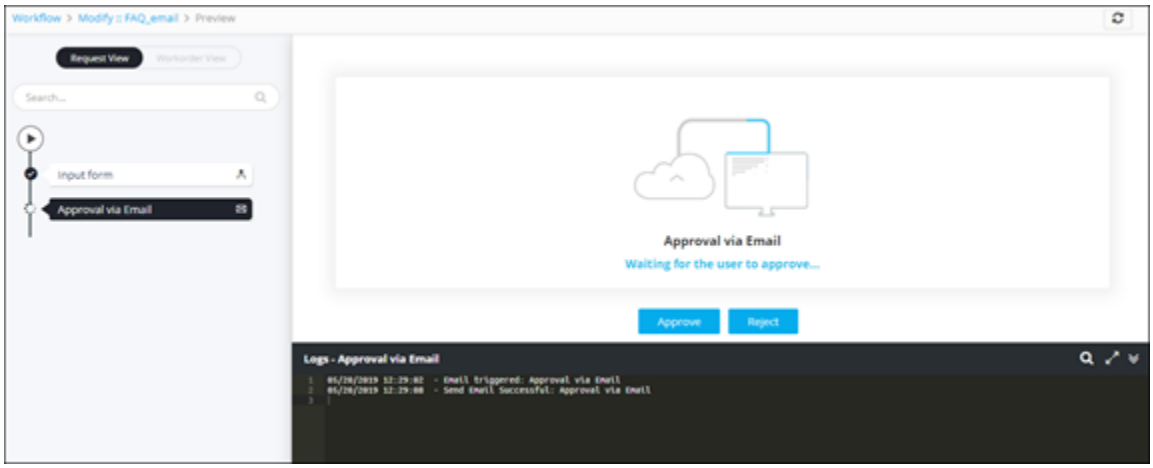
The email is rejected and the task is displayed as failed.



- In the event of clicking the email approval again, the following message indicating that the email response has been processed will be shown in a new tab.

- In the event of not processing the approvals via email on time, users can log into AppViewX, navigate to the pending workflow request and manually approve or reject the workflow task.

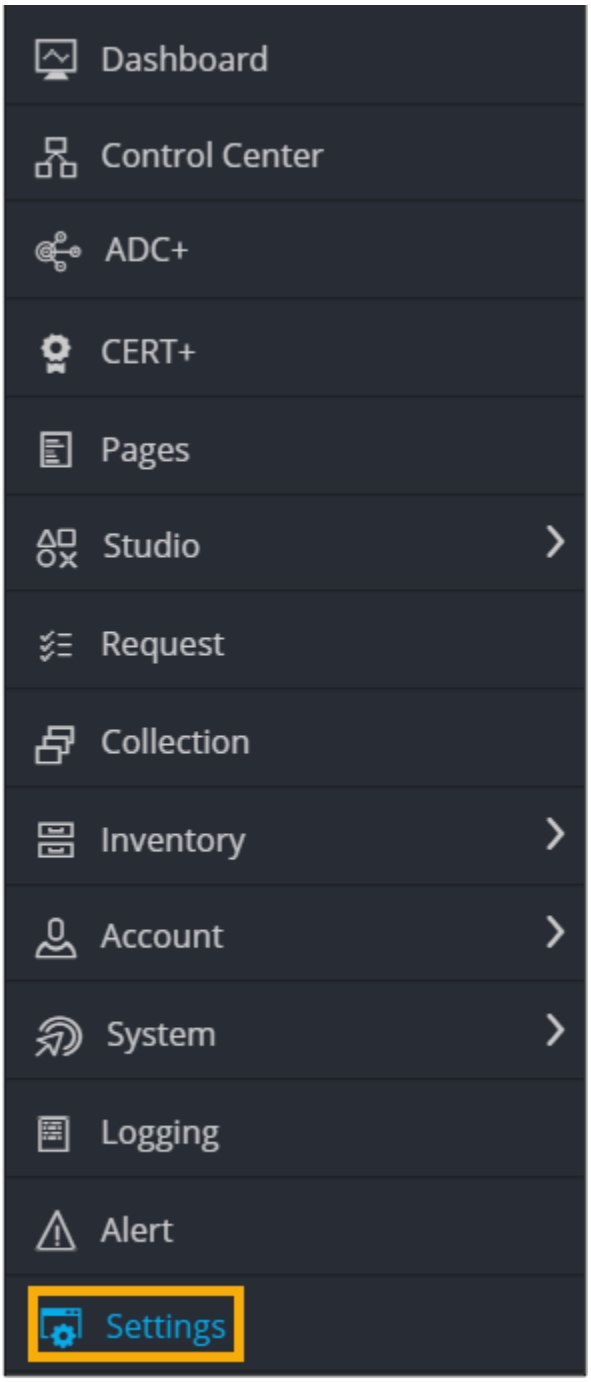
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Note: The SMTP settings must be configured prior to triggering any emails.

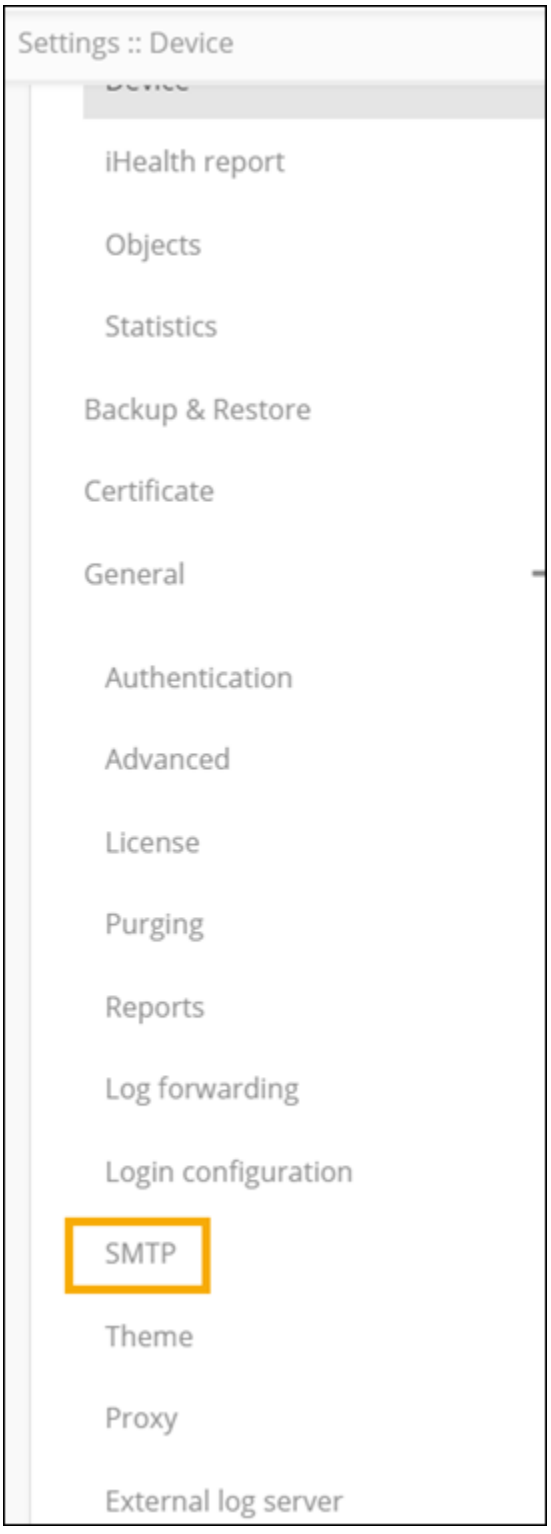
3. To configure SMTP settings, from the upper left corner of the screen, click ☰.
4. From the menu displayed, click **Settings**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



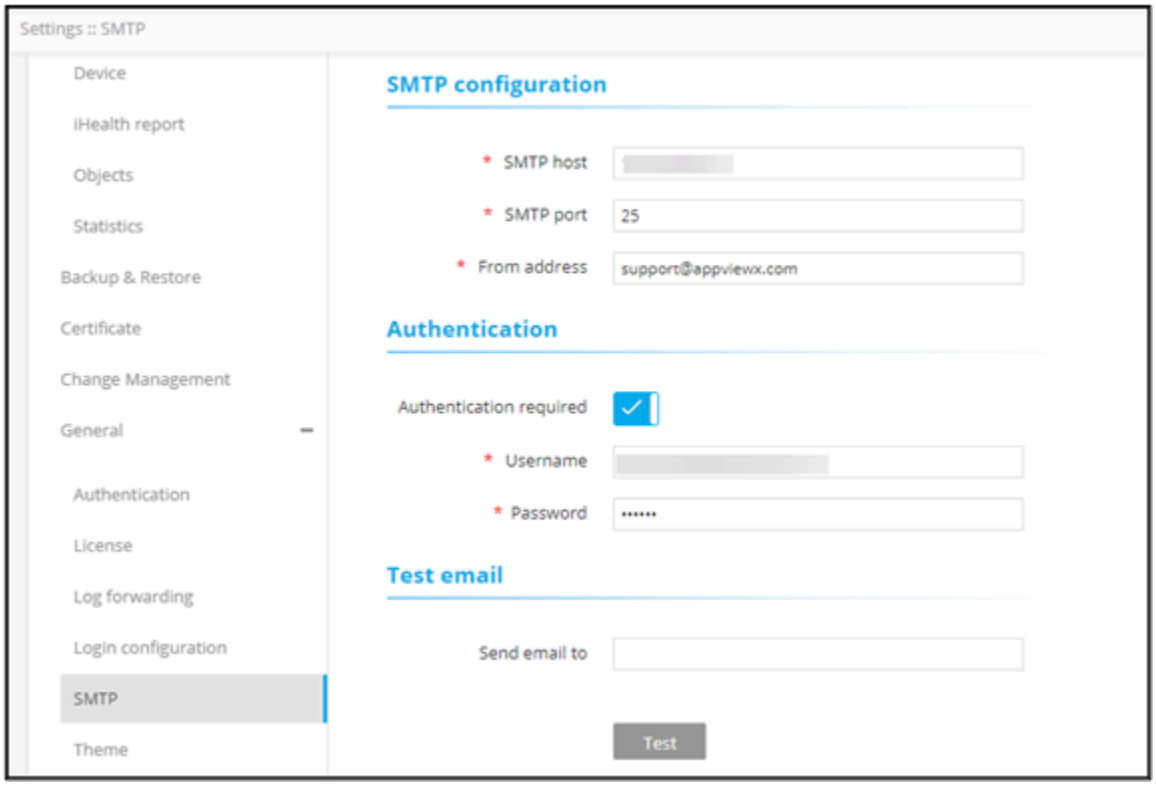
5. On the **Settings** page, from the navigation pane on the left, click **General**.
6. Under **General**, click **SMTP**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



The **Settings::SMTP** page is displayed.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



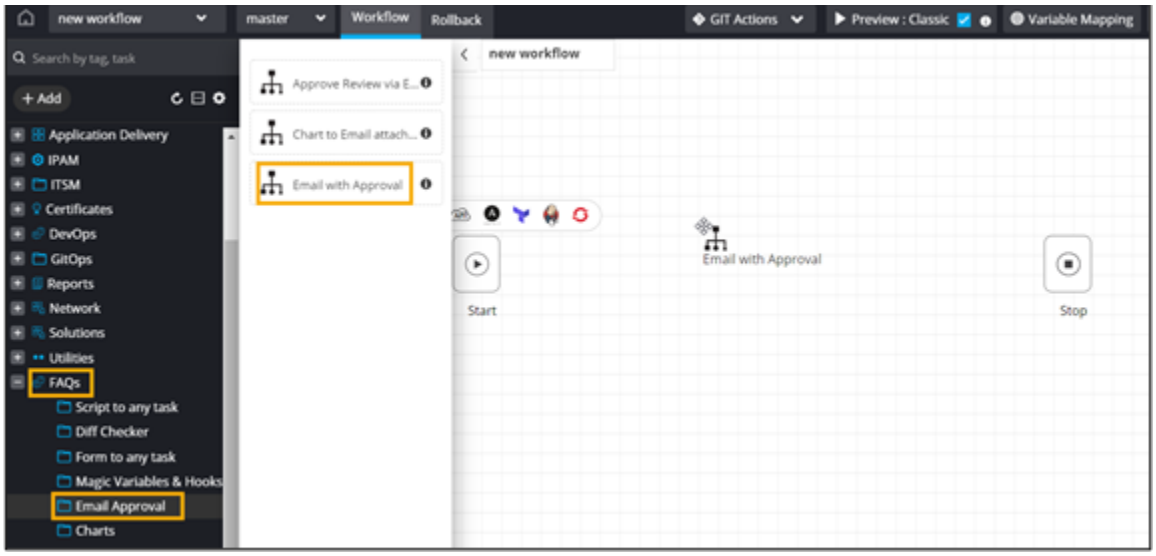
7. Update SMTP details.

Triggering an Email Notification with Approvals

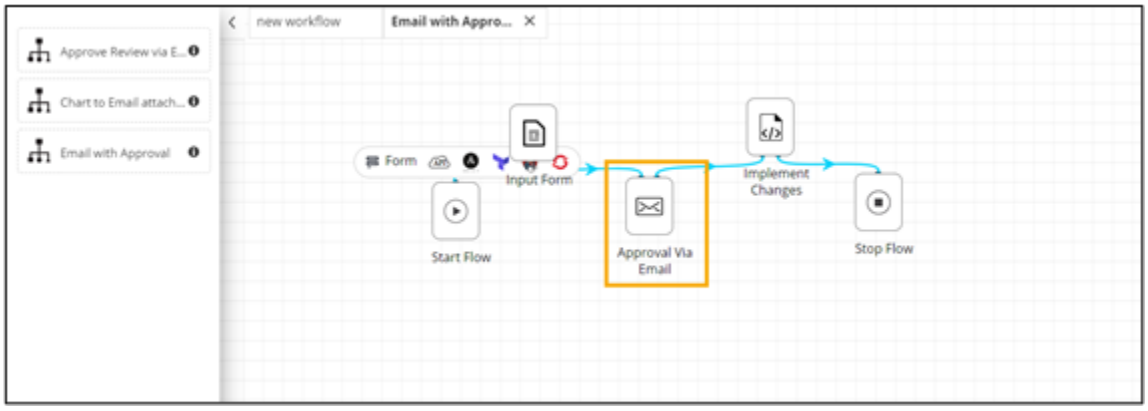
To generate a simple email notification with an option to approve or reject the workflow:

1. Design a new workflow in the workflow studio.
2. Navigate to **FAQs** > **Email with approval**.
3. Drag & drop the **Email with Approval** flow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

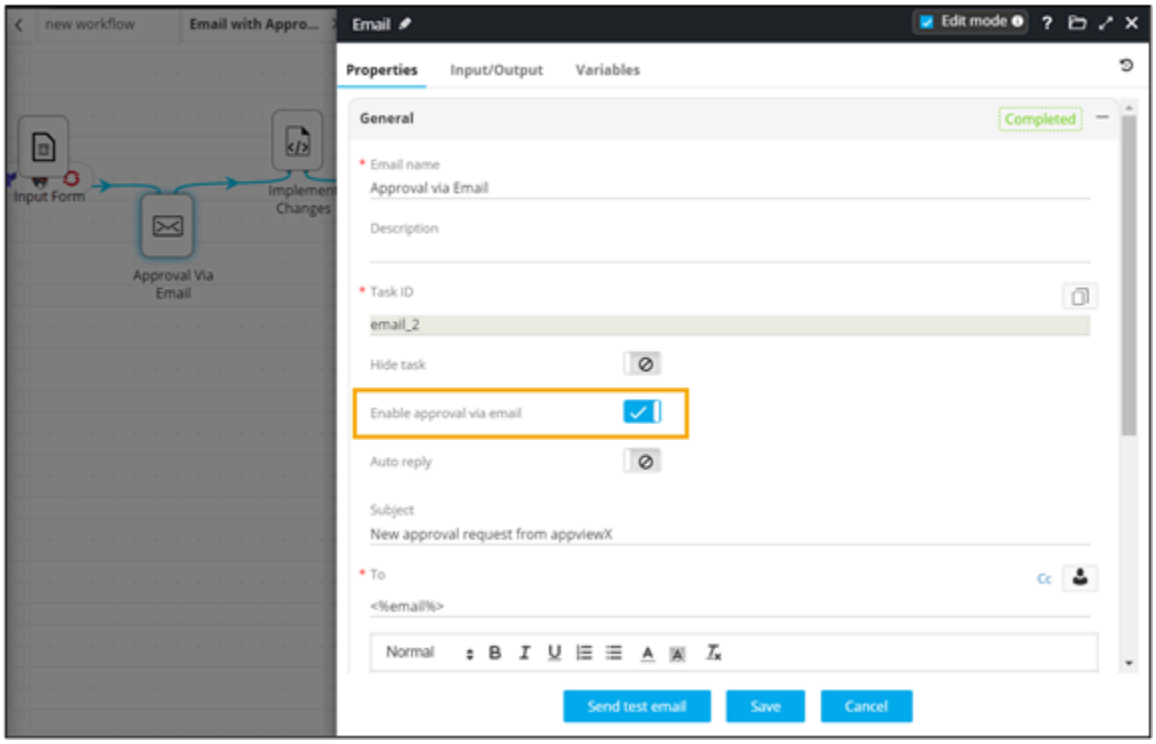


4. Double-click on the flow to see the tasks within the workflow.
5. Click on the **Approval Via Email** task in the workflow.

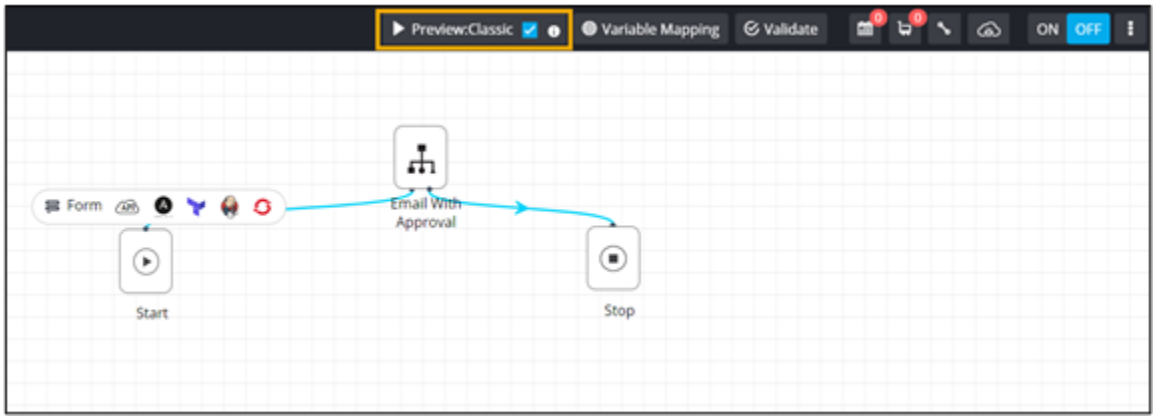


6. In the **Email** task window, under **Properties**, in the **General** section, turn on the **Enable approval via email** toggle.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

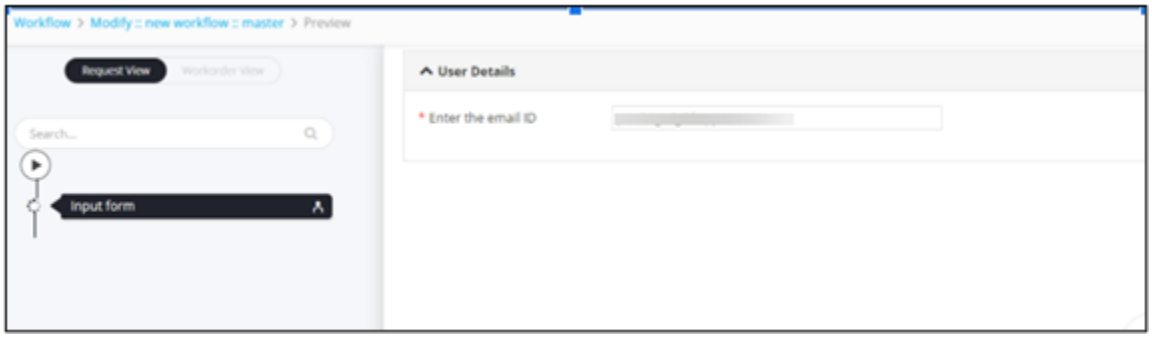


7. Click **Preview**.

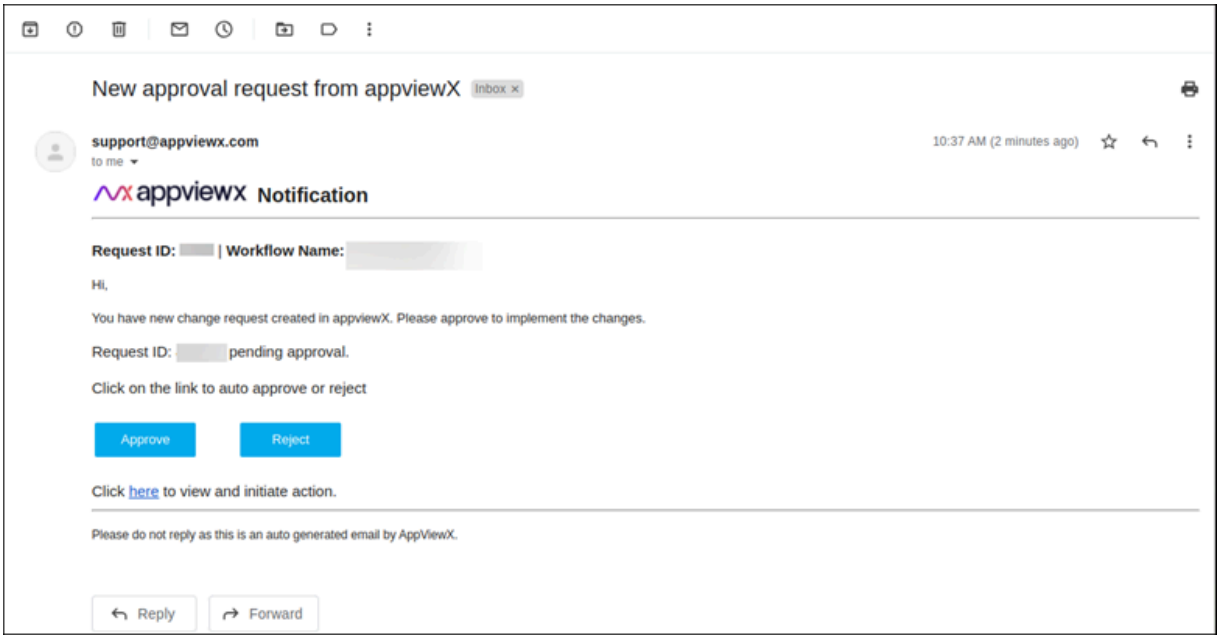


8. Enter the email ID in the user input form.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Approval email notification received via email.

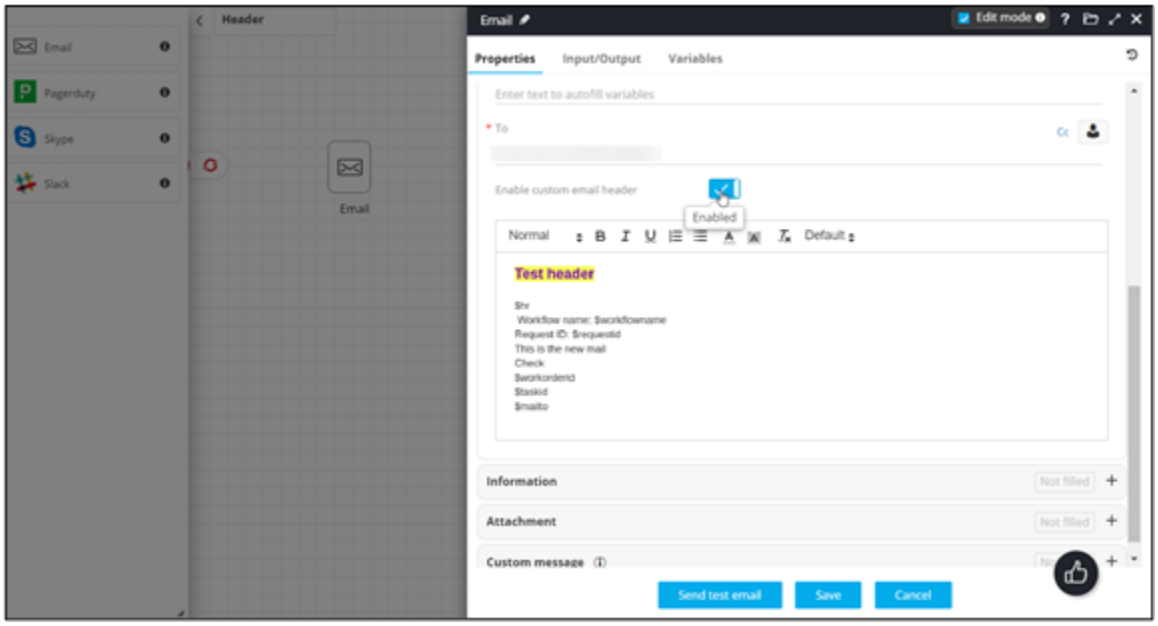


Customizing Email Notification

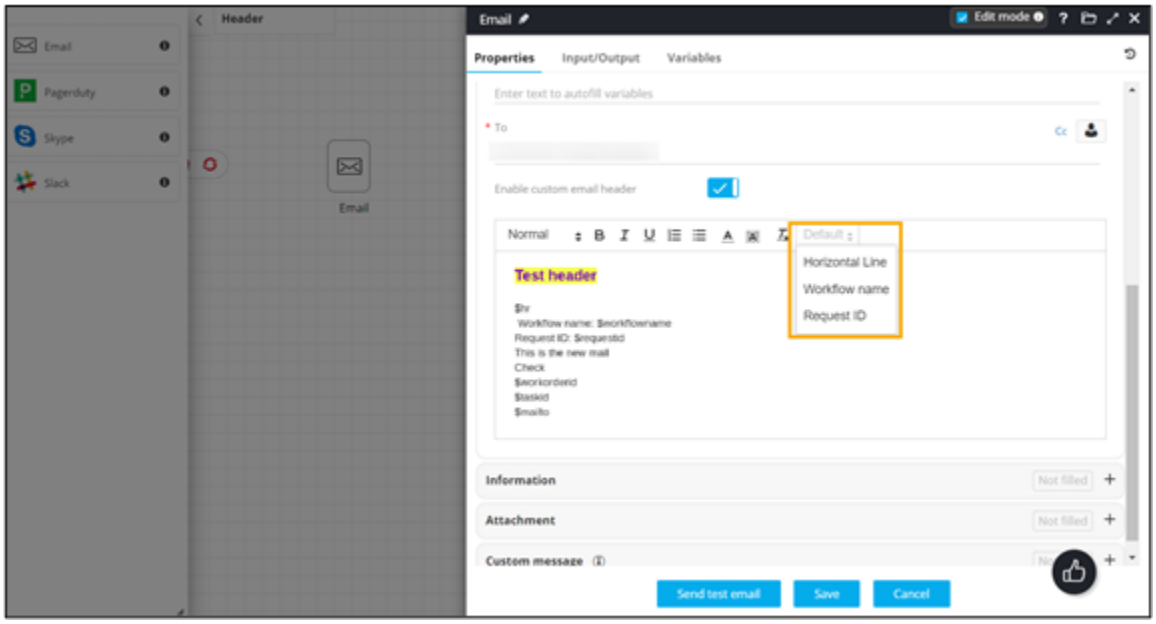
You can customize the email notification by configuring the text in the content box in the email task. You can also customize the email content by including default variables like workflow name, horizontal line and Request ID.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. To **Enable custom email header**, turn on the toggle in the email task.



2. To attach the default variables (Horizontal Line, Workflow Name, Request ID), click **Default** and select the variable from the dropdown.



Custom email notification received.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Note: When the Enable custom email header toggle is disabled, the email content will display appviewX Notification as the header.

Pagerduty

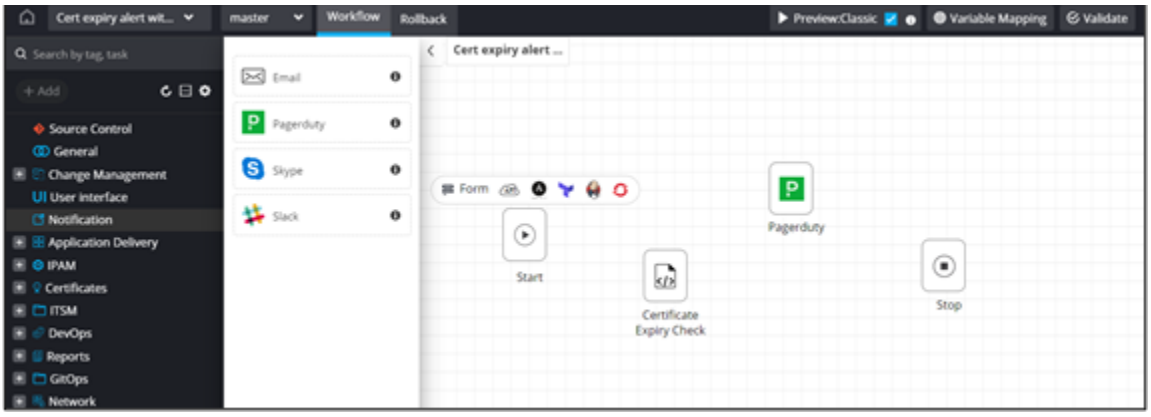
This task allows you to trigger alarm notifications for quick incident remediation through Pagerduty as part of the workflow automation process.

- Provision to drag and drop the Pagerduty Task from the notification section.
- Provision to define custom webhook/URL, method in order to trigger notification.
- Provision to refer global variables and embed them as part of the payload

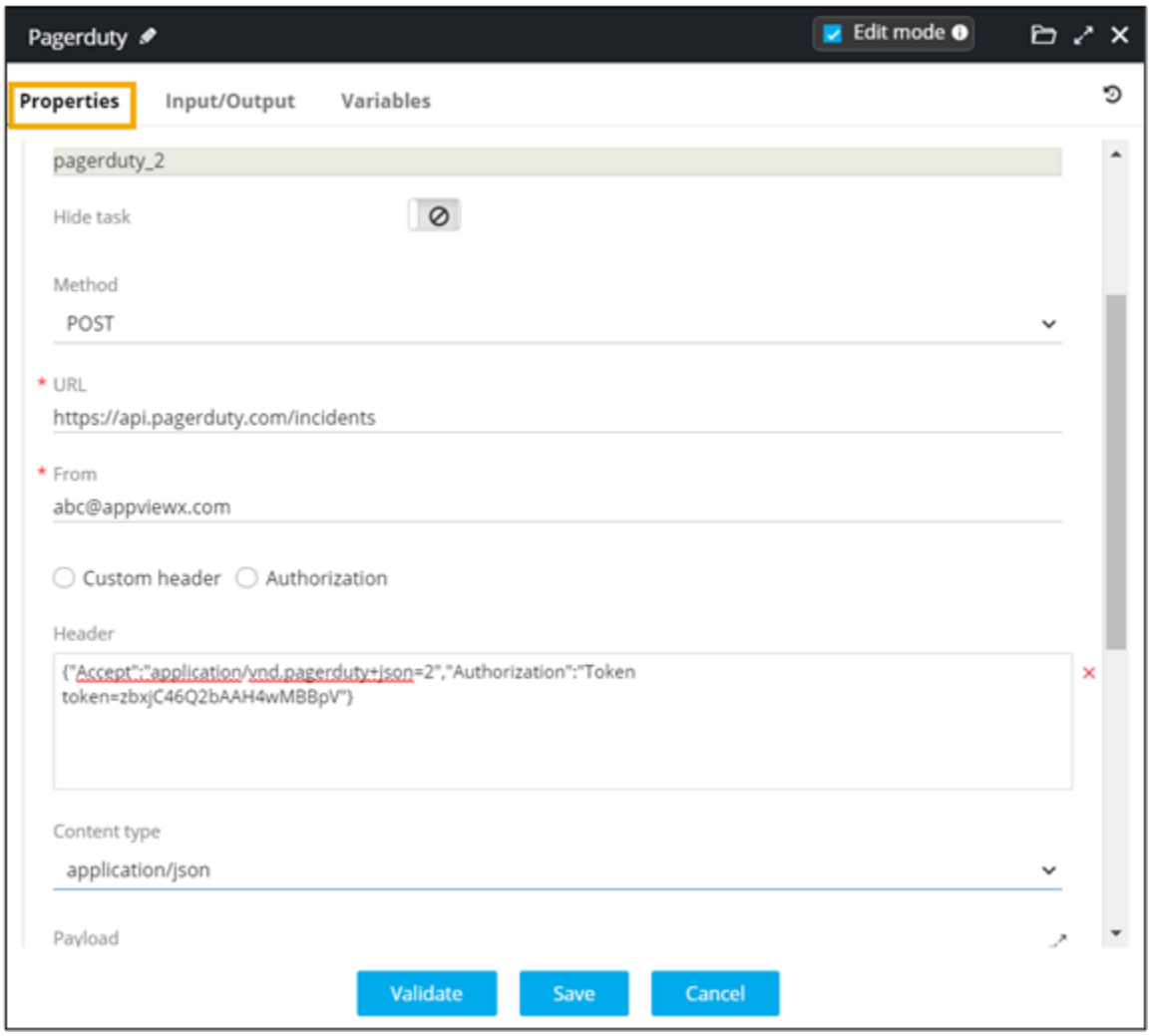
To trigger alarm notifications through Pagerduty:

1. Design a workflow.
2. Drag & drop relevant workflow task(s).

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



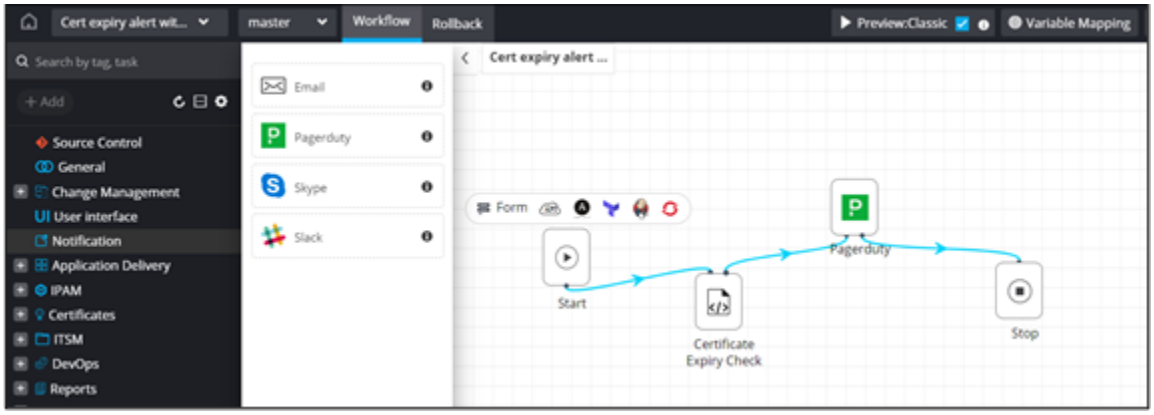
3. Double click on the **Pagerduty** task.
4. In the **Pagerduty** task window, under **Properties**, enter or select the field information as shown.



5. Provide the intended payload, with global variable(s) if any.

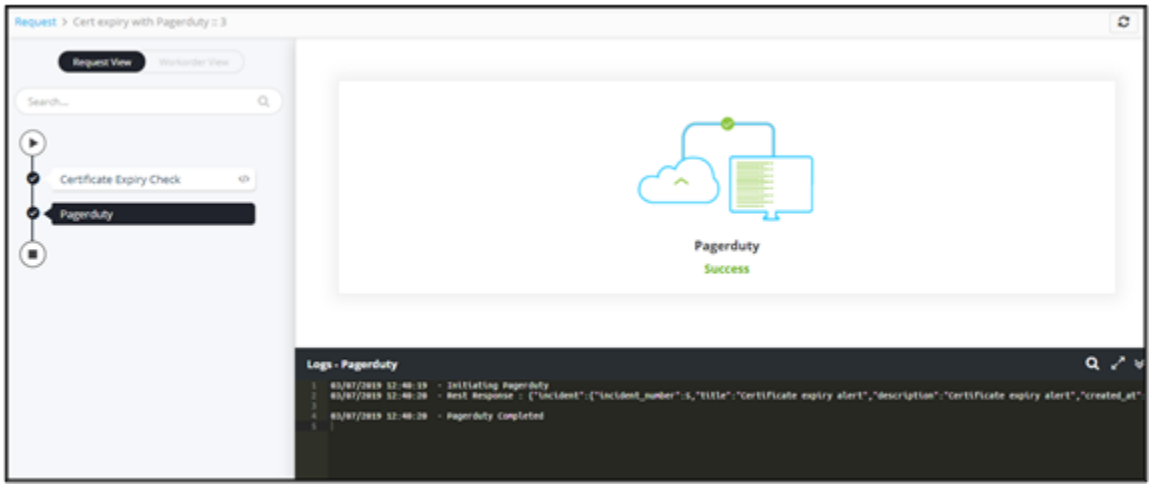
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

6. Connect the workflow.



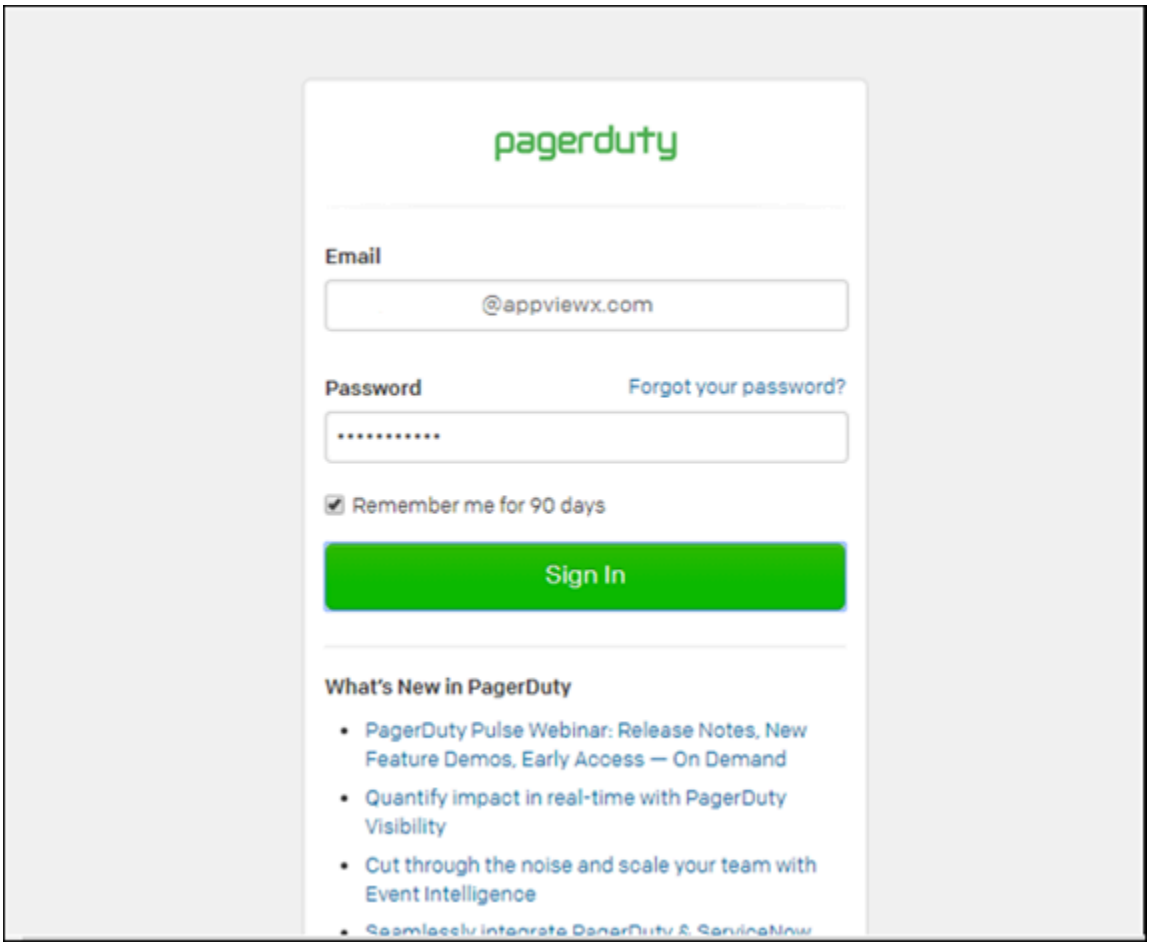
7. Trigger the workflow from the Request :: View/Run page.

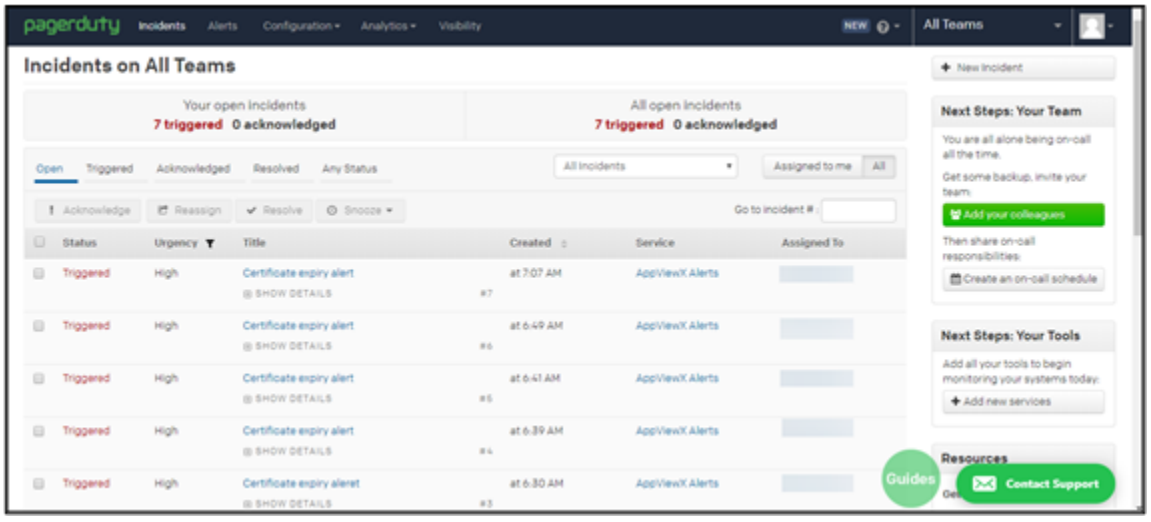
   Workflow is executed.



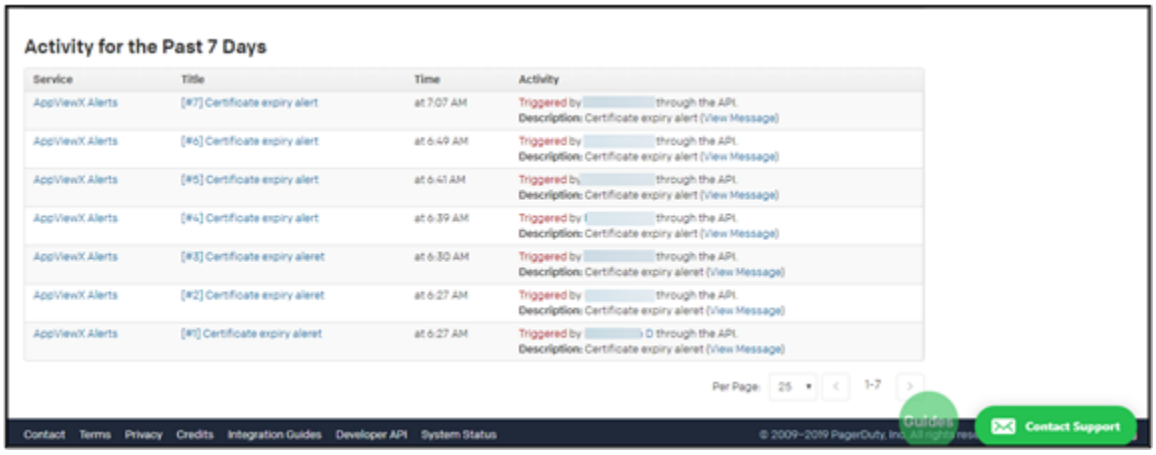
8. To view all Incidents on Pagerduty, log into your account using valid credentials.

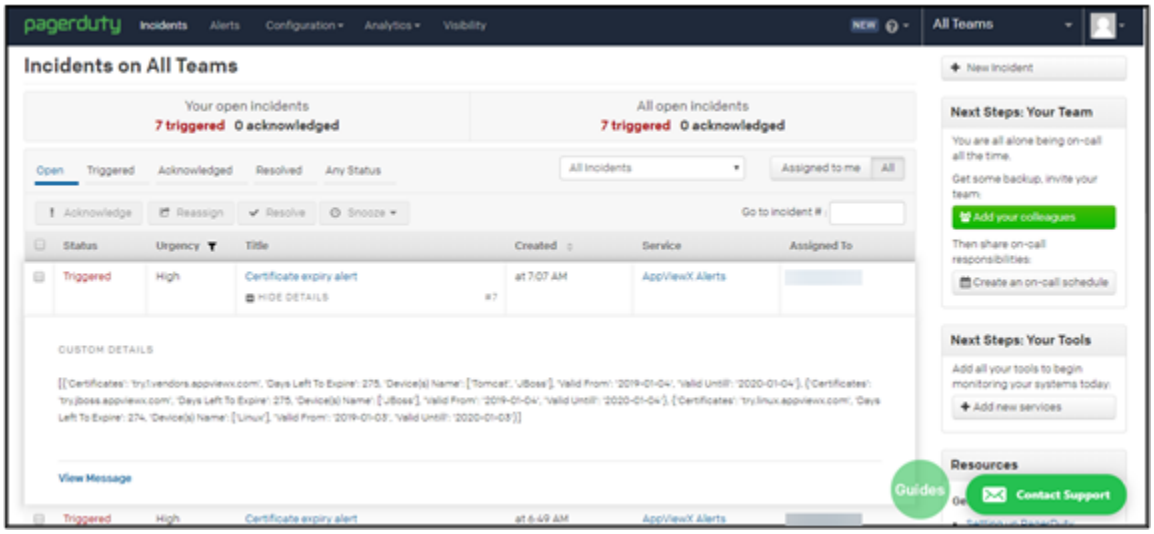
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



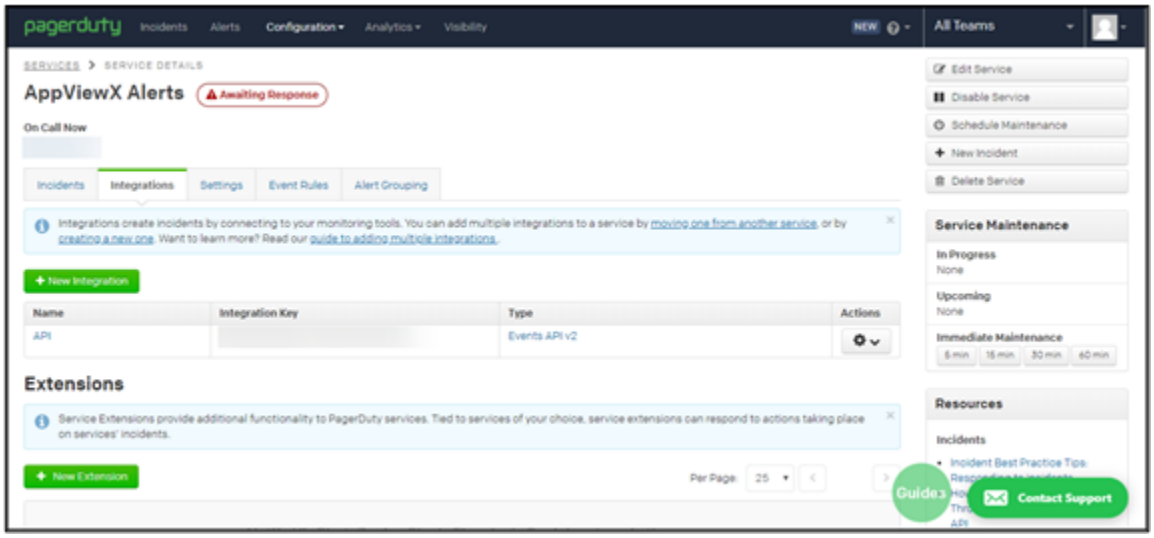


Copyright © 2020 AppViewX, Inc. All Rights Reserved.





Copyright © 2020 AppViewX, Inc. All Rights Reserved.



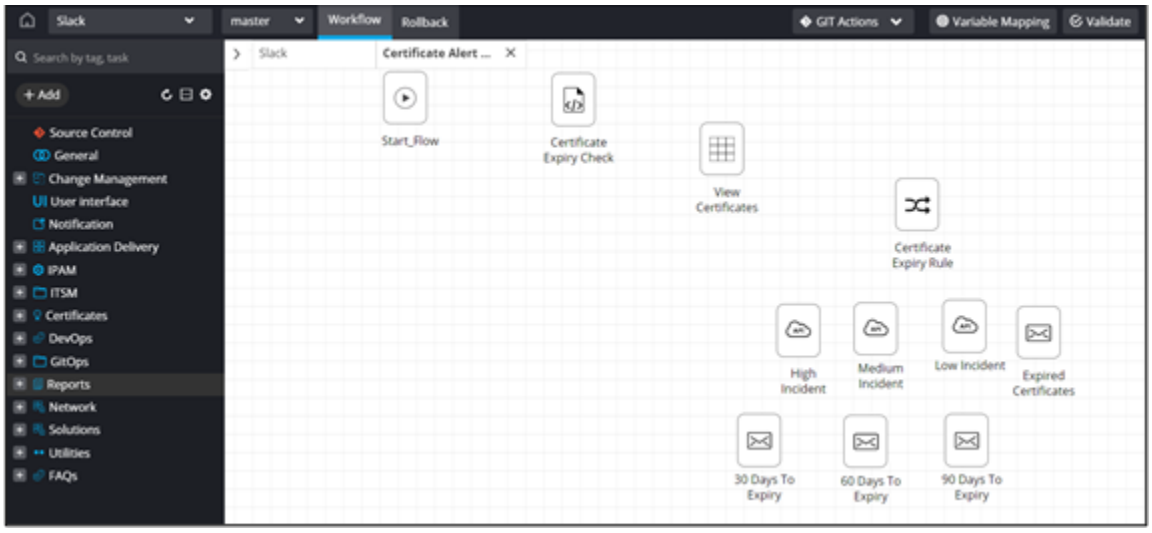
Slack

This task allows you to send notifications on a Slack channel as part of the workflow process.

- Provision to drag and drop the slack task from the Notification section.
- Provision to define custom webhook, method in order to trigger slack notification.
- Provision to refer global variables and embed them as part of the payload .

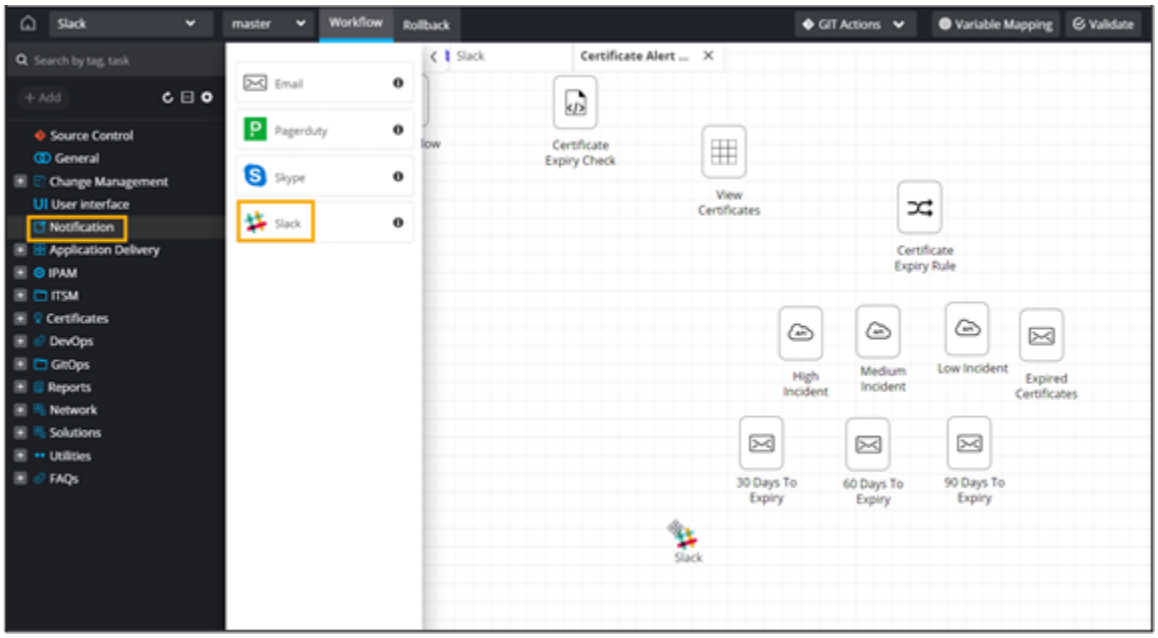
To send a notification through Slack:

1. Design/Modify a workflow.
2. Drag and drop relevant workflow tasks.

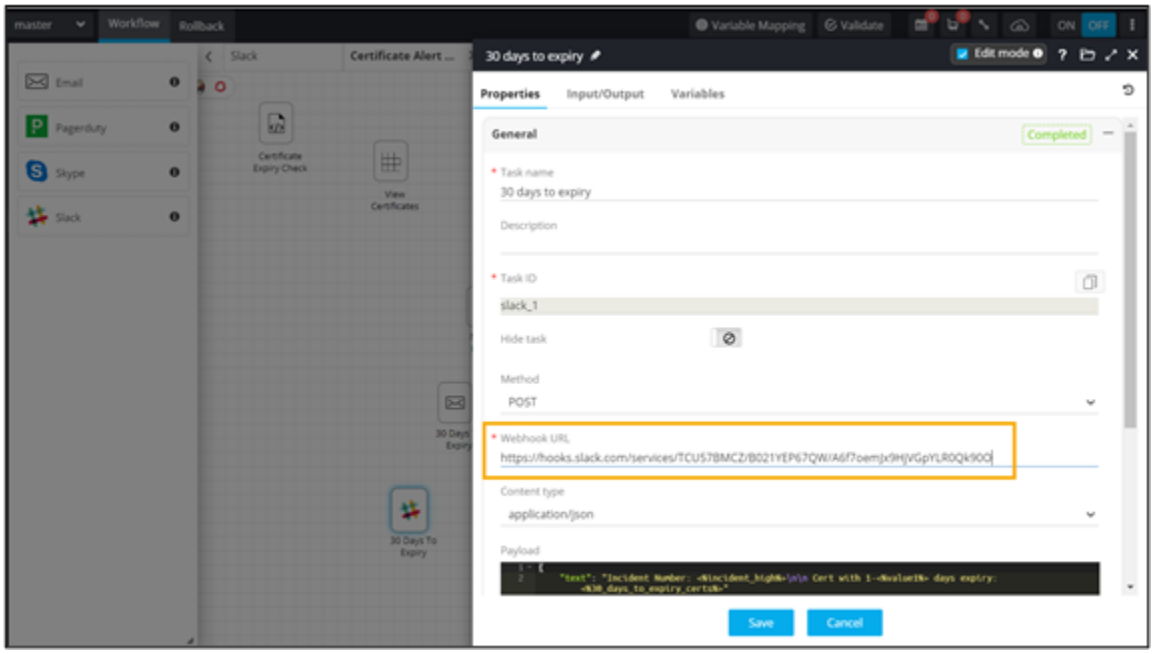


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

3. From the **Notification** section, drag and drop the **Slack** task.

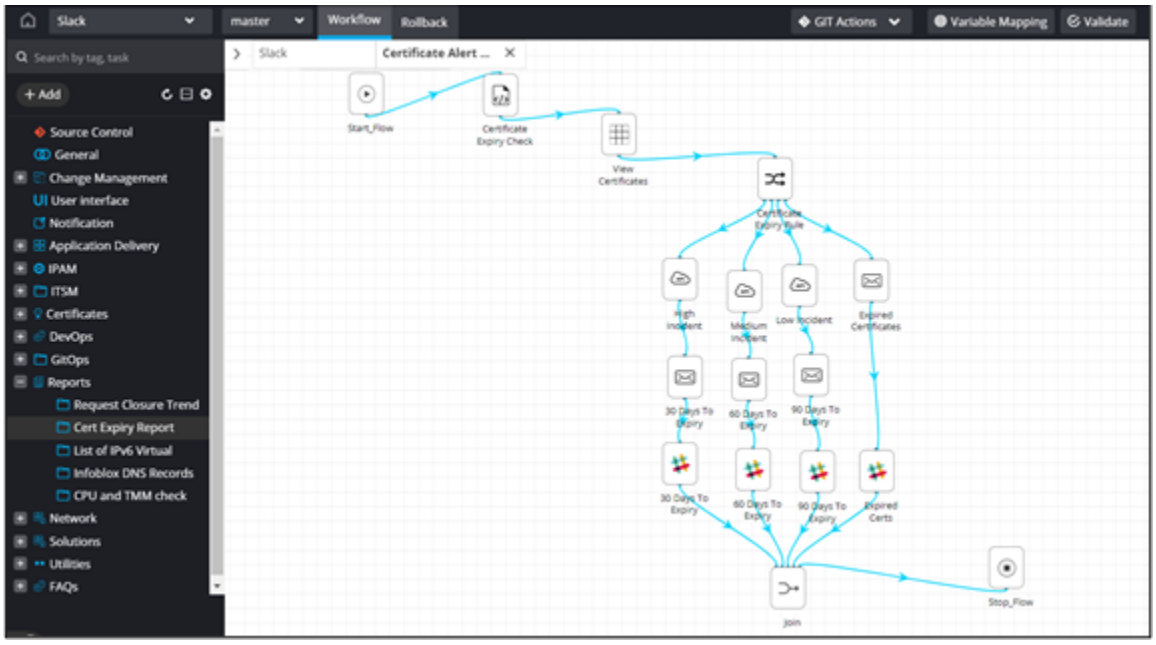


4. Define the appropriate webhook.

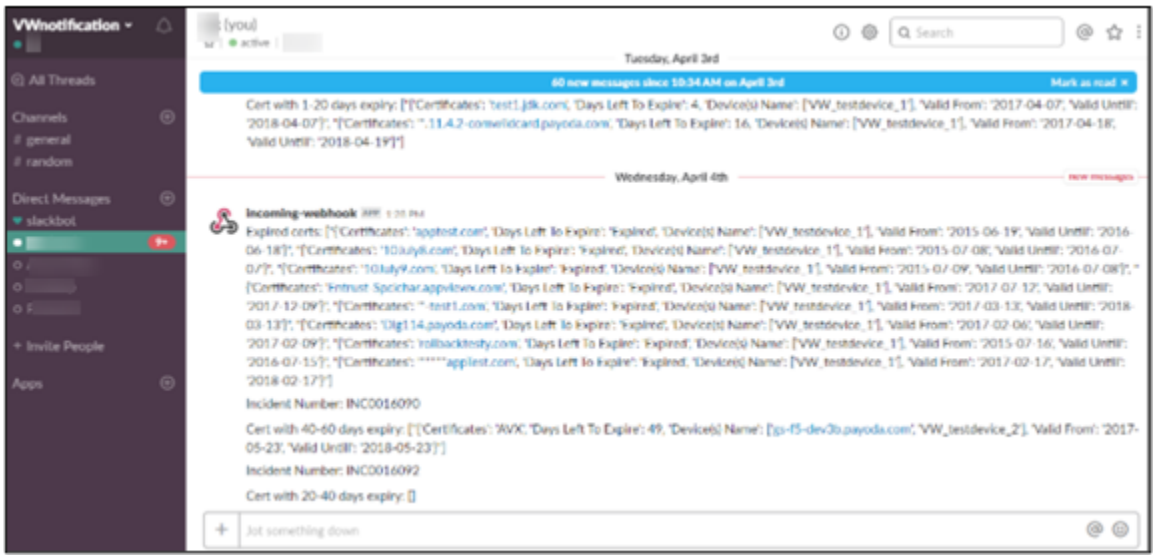


5. Provide the intended payload with global variables, if any.
6. Connect and enable the workflow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



7. Trigger the workflow from the Request :: View/Run page.
8. View Slack notifications.

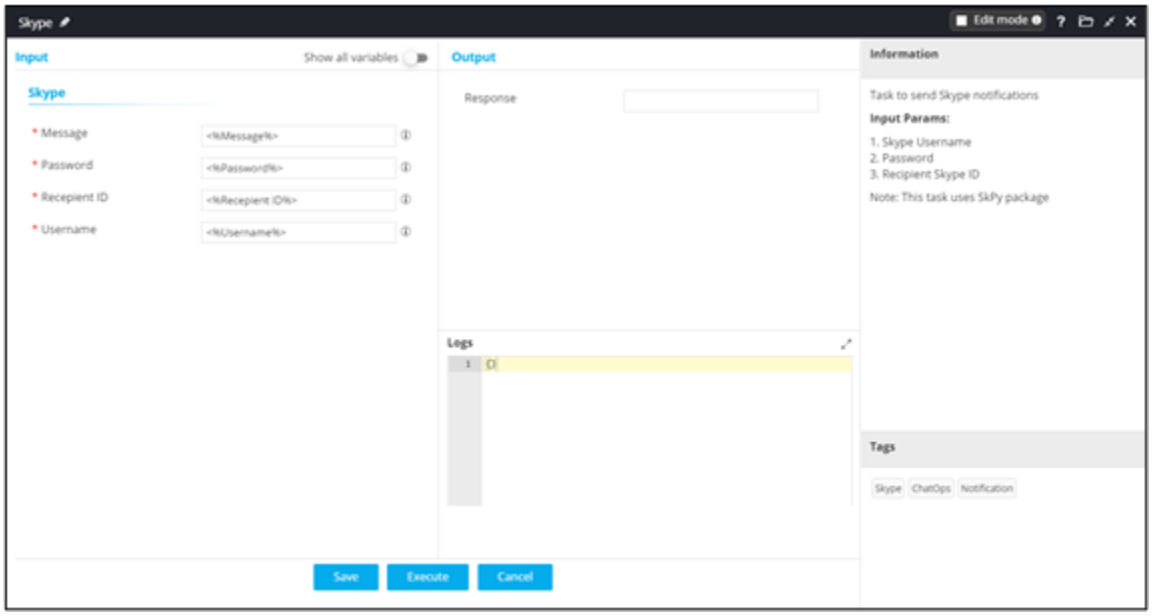


Skype

This task allows you to send Skype notifications as part of the workflow process.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

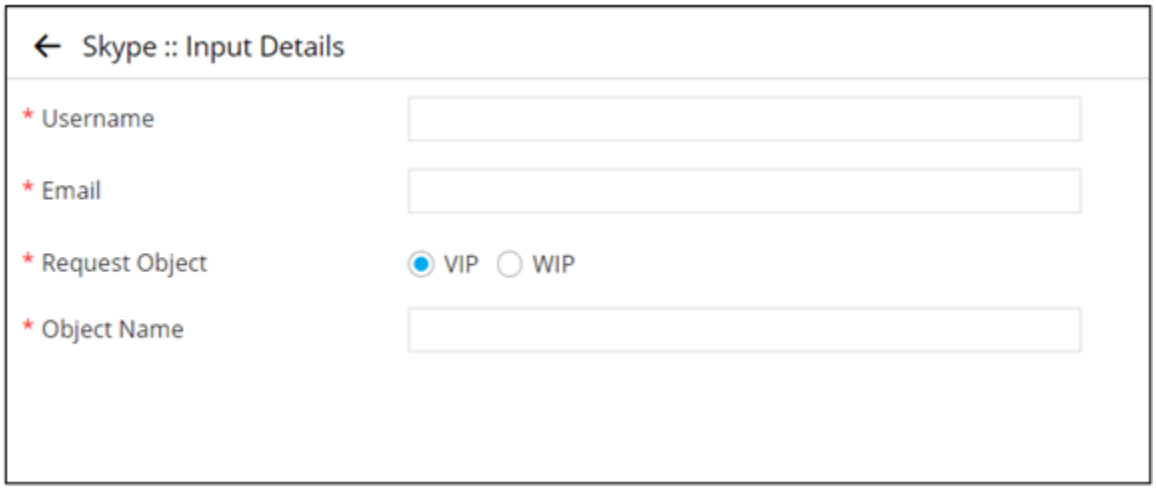
- Provision to drag and drop the Skype task from the Notification section.
- Provision to define custom webhook, method in order to trigger Skype notification.
- Provision to refer global variables and embed them as part of the payload .



To send a notification through Skype:

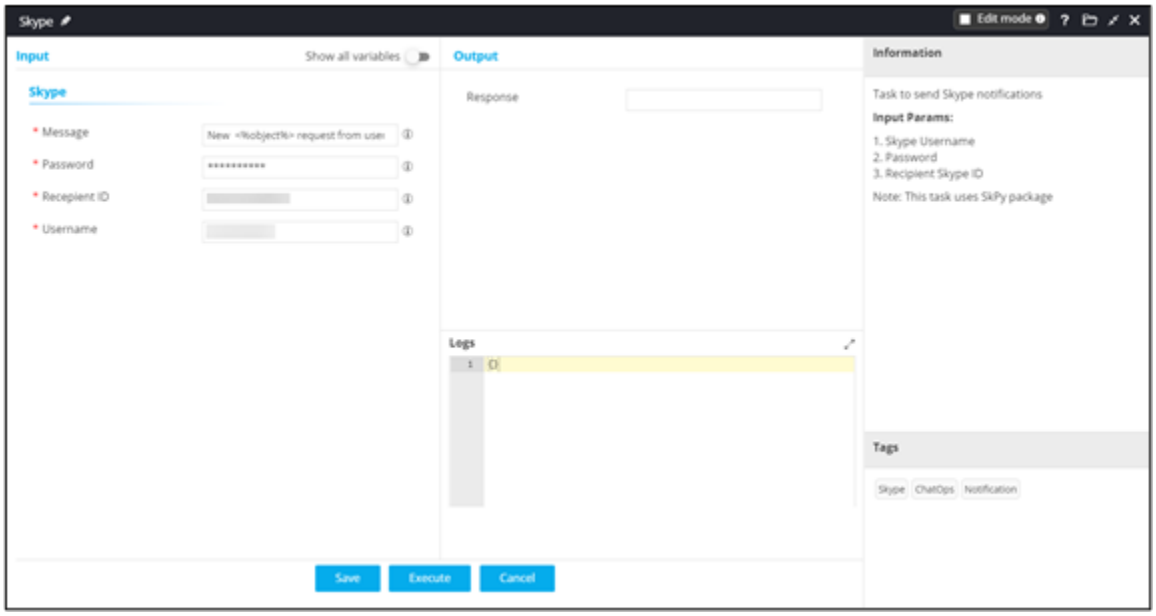
1. Design a new workflow.
2. From the **User Interface** section, drag and drop the **Form** task.
3. Define the relevant form fields as shown here.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



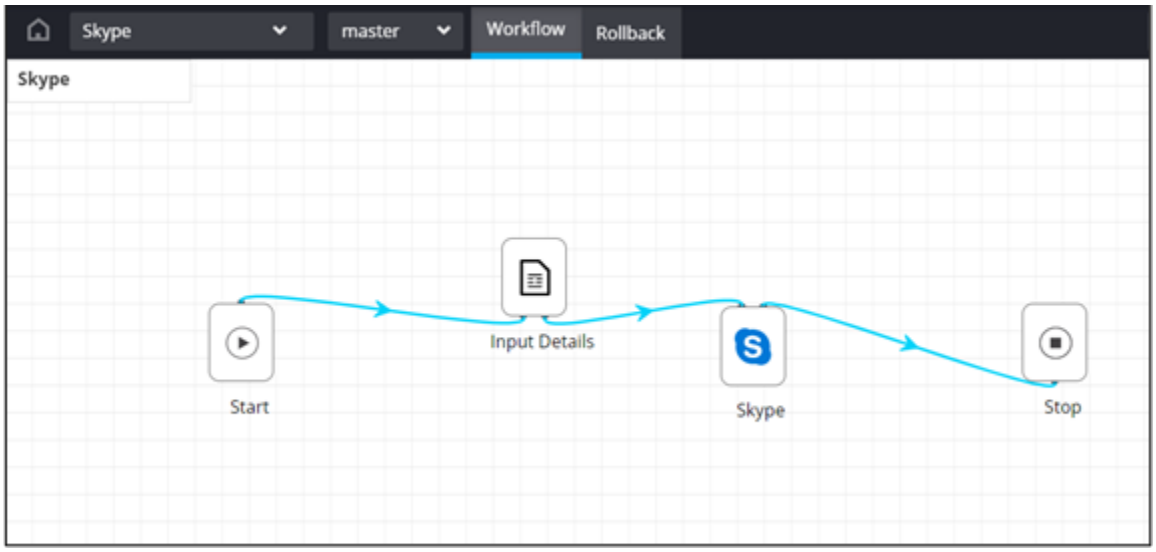
Note: For more information on adding form fields, click here.

4. From the **Notification** section, drag and drop the **Skype** task.
5. In the **Skype** task window, define the input parameters to send Skype notification to the recipient.



6. Connect and enable the workflow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



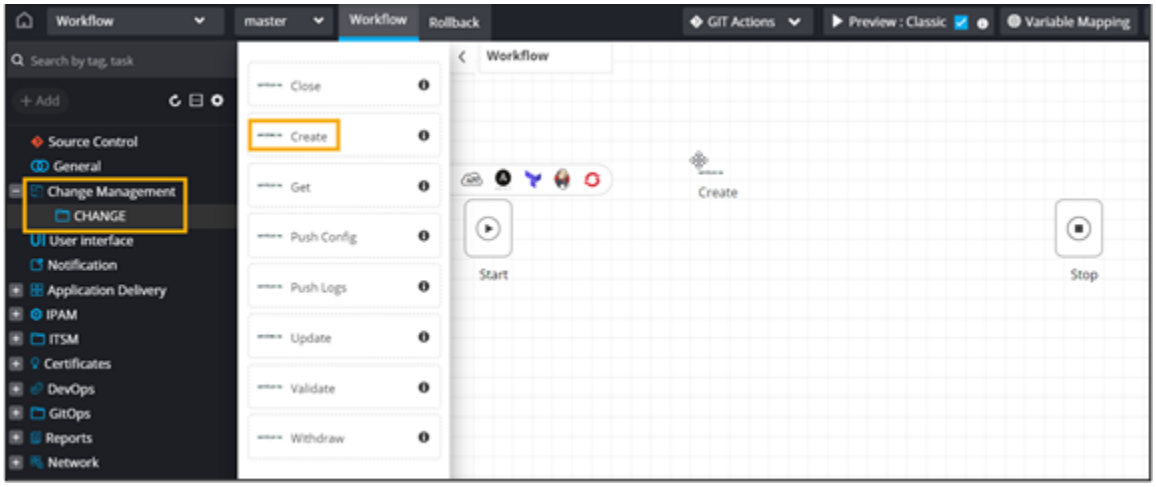
7. Trigger the workflow from the Request :: View/Run page to receive Skype notification.

Task Category - Change Management

When creating a new workflow, you can leverage the Change management Tasks to ensure change management integration with workflow automation. It allows you to plug and play various components required to integrate a workflow automation flow with Change Management.

To create a workflow with ServiceNow tasks:

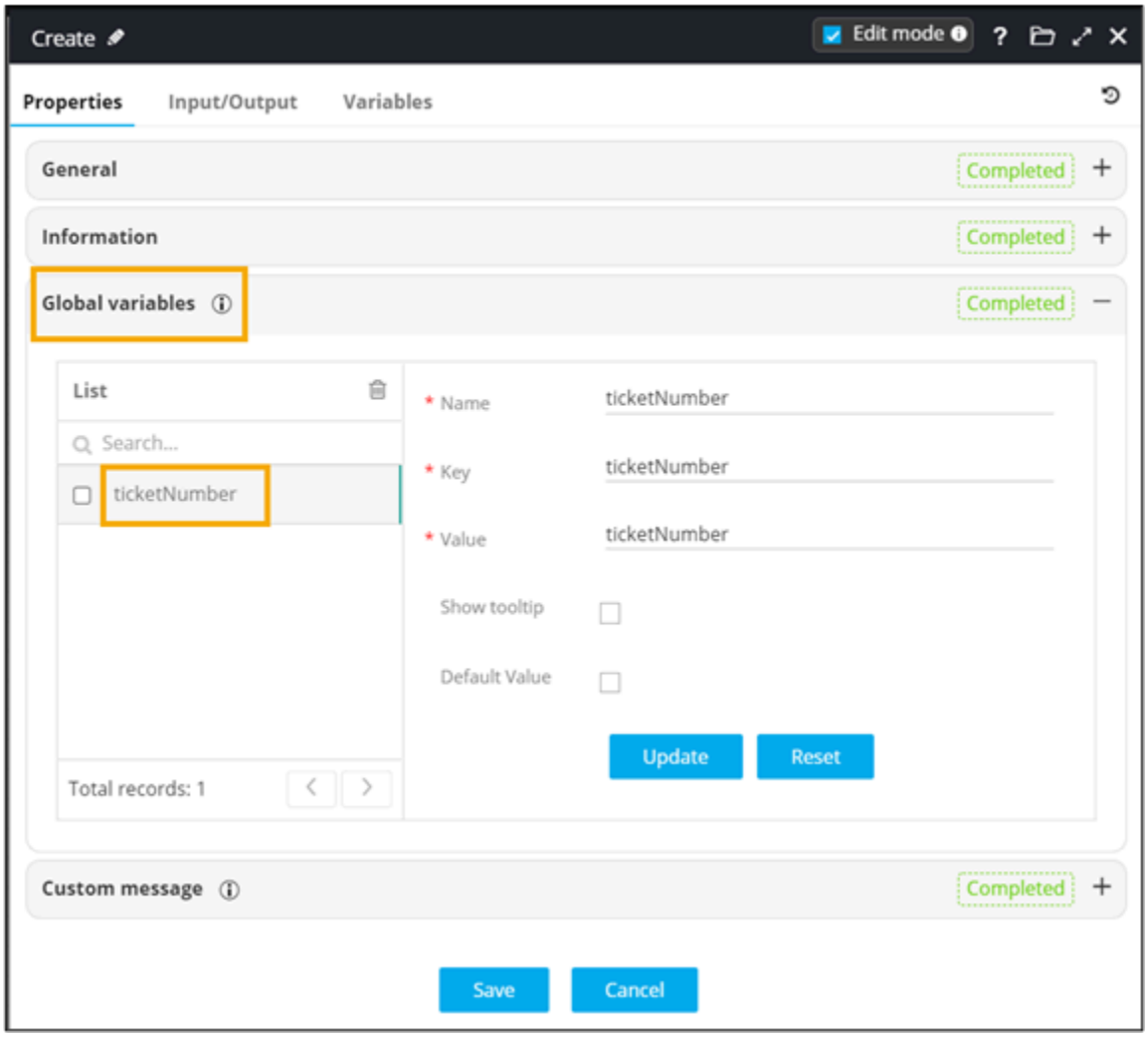
1. Design a workflow.
2. Under **Change Management**, from the **CHANGE** folder, drag and drop the **Create** task.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

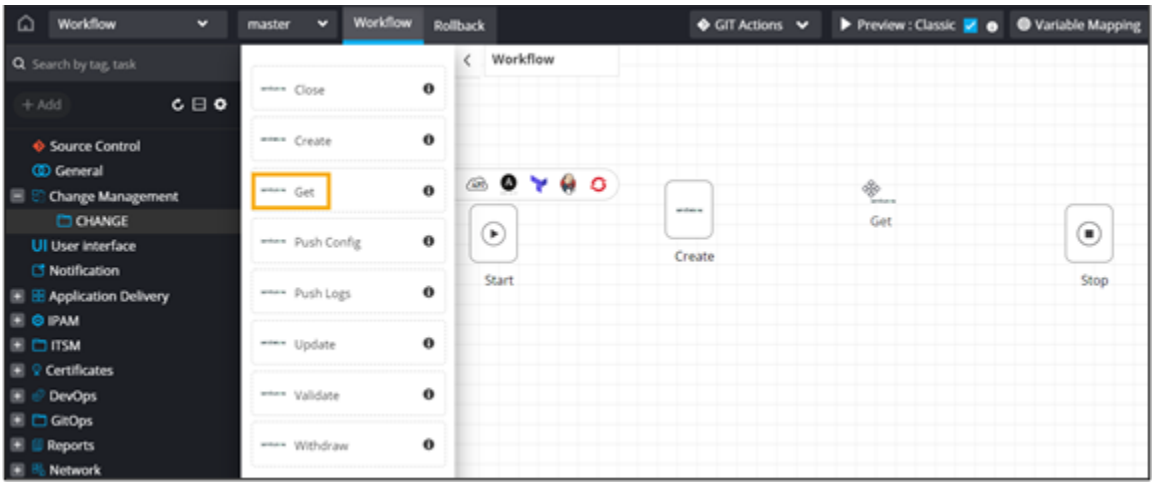
Note: Timestamp must be in milliseconds.

3. Under the **Create** task window, under **Properties**, in the **Global variables** section, declare the necessary global variables.

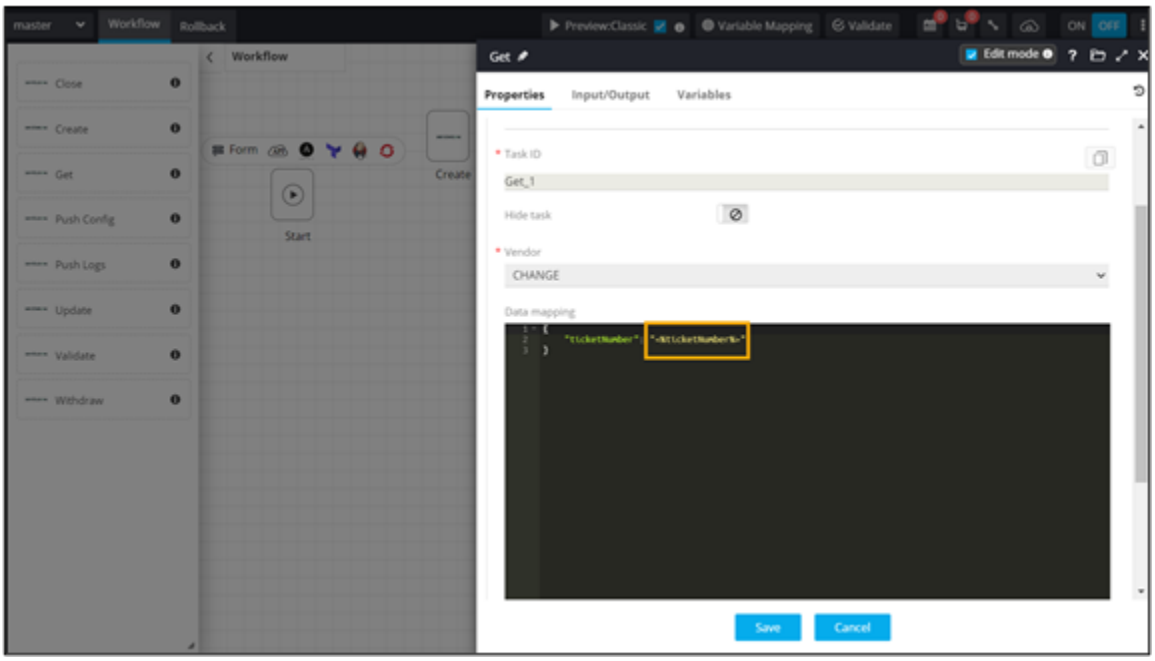


4. Under **Change Management**, from the **CHANGE** folder, drag and drop the **Get** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

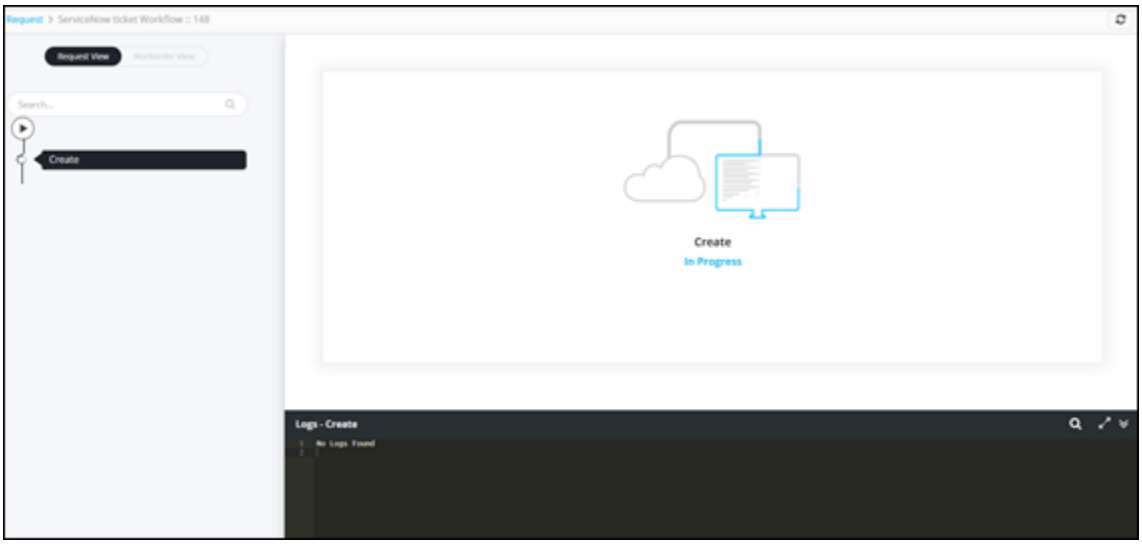


5. In the **Get** task window, under **Properties**, pass the global variable declared in the previous task.

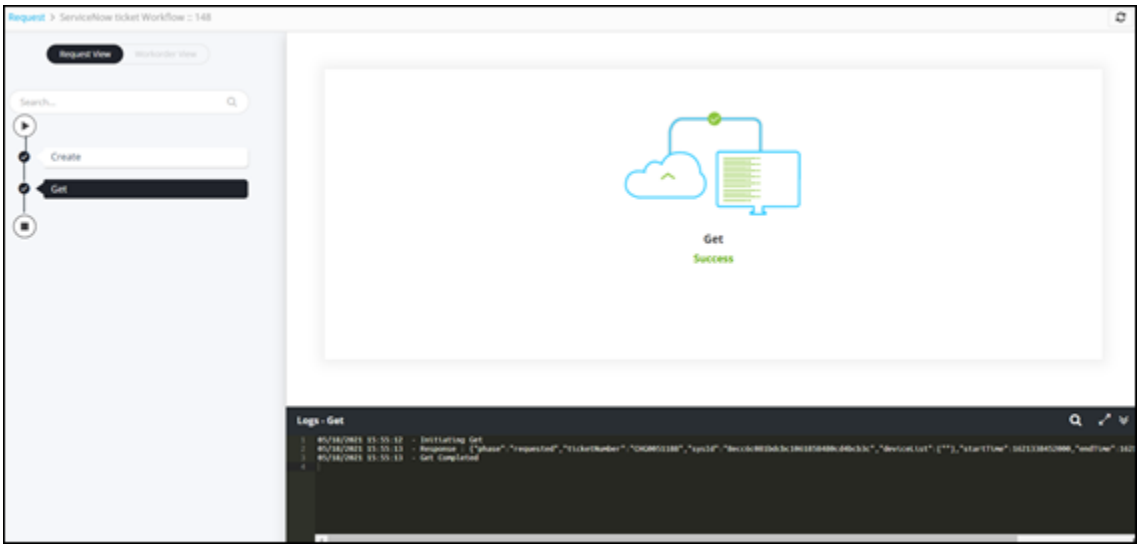


6. Connect and trigger the workflow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Workflow is executed successfully.

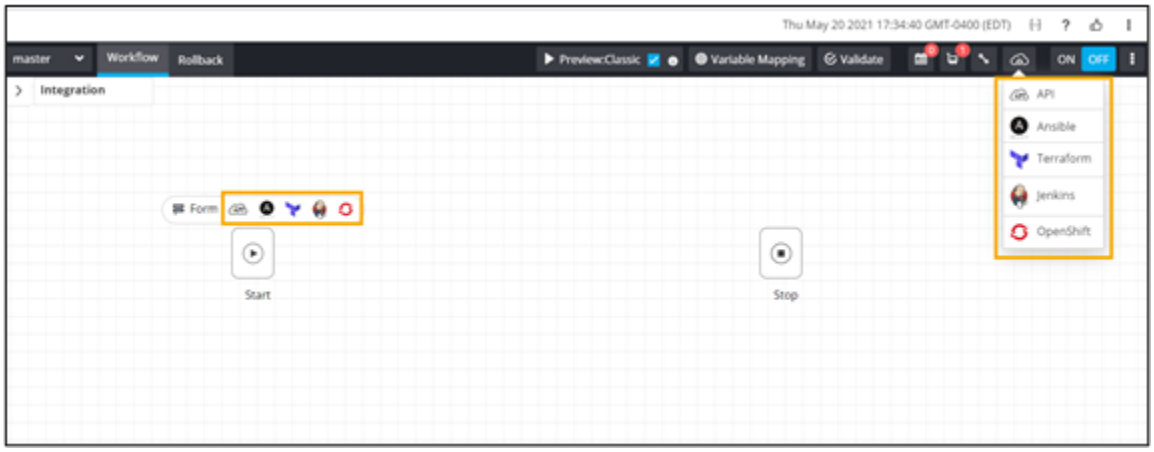


Integrations

Organizations have multi-vendor tools for various network needs. AppViewX allows you to integrate with automation tools such as Ansible, Jenkins, Terraform, and Openshift in order to automate and orchestrate the network.

- Provision for native Northbound integration with multiple platforms
- Provision to generate an API and trigger a workflow from an external source (Python code, Javascript, Web browsers, Postman)

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



- Postman - Visual Workflow Northbound Integration
- Jenkins - Visual Workflow Northbound Integration
- Terraform - Visual Workflow Northbound Integration
- Ansible - Visual Workflow Northbound Integration
- Openshift PaaS Orchestration
- Ansible Southbound Integration

Postman - Visual Workflow Northbound Integration

This integration enables dynamic API generation from a third-party payload generator such as Postman, which can be used to design custom automation workflows.

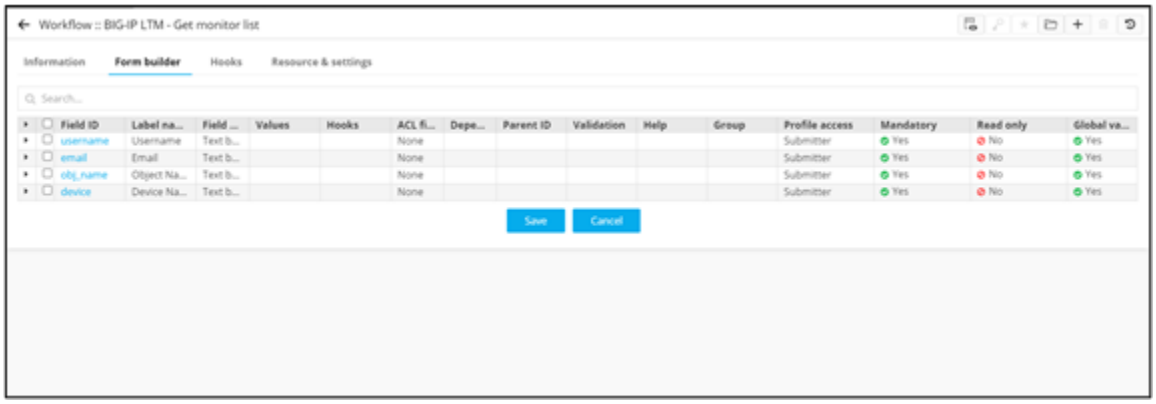
- Dynamic API Generation

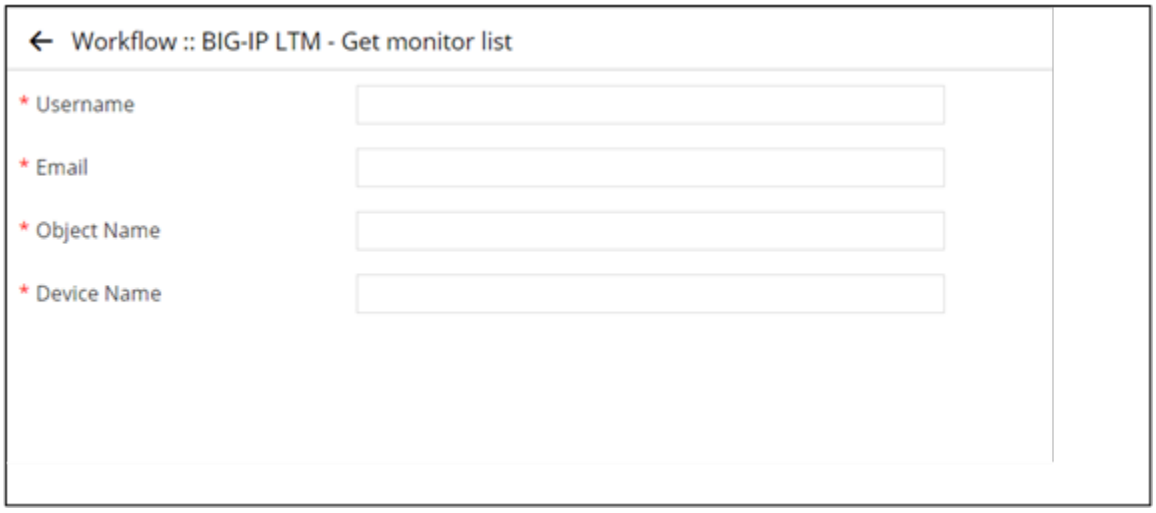
Dynamic API Generation

You can design and enable a workflow and dynamically generate custom API to ease the automation and orchestration process. This allows a simple API payload builder/generator to aid you in integrating with external tools and systems.

1. Design a new workflow.
2. From the **User Interface** section, drag and drop a **Form** task.
3. Define the form fields.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

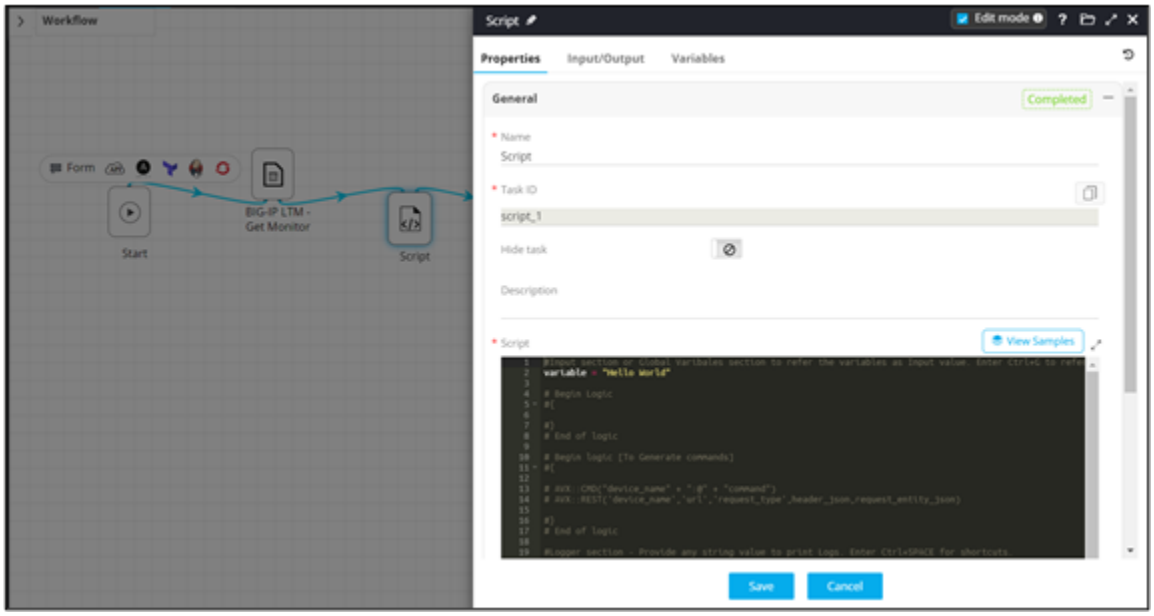




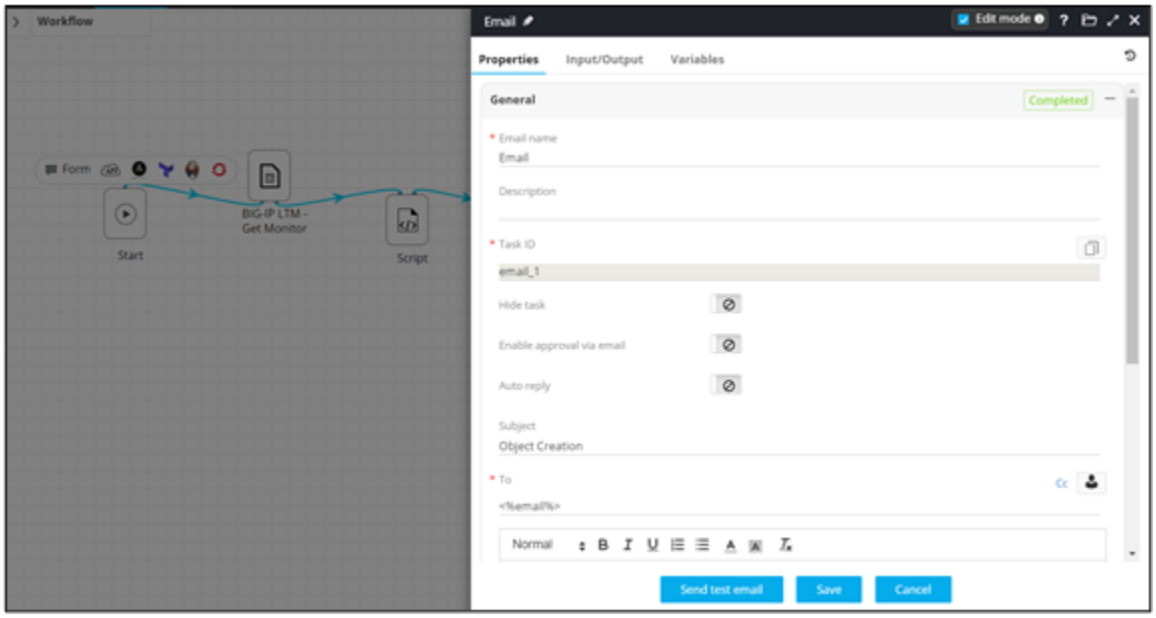
Note: For more information on adding form fields, click here.

4. From the **General** section, drag and drop a **Script** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

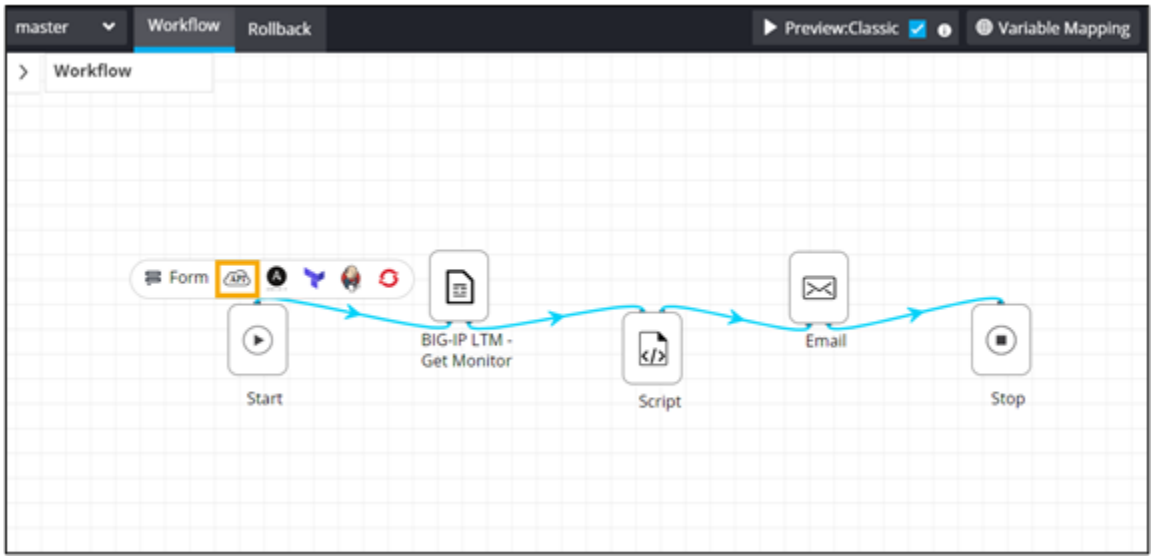


5. From the **User Interface** section, drag and drop an **Email** task.

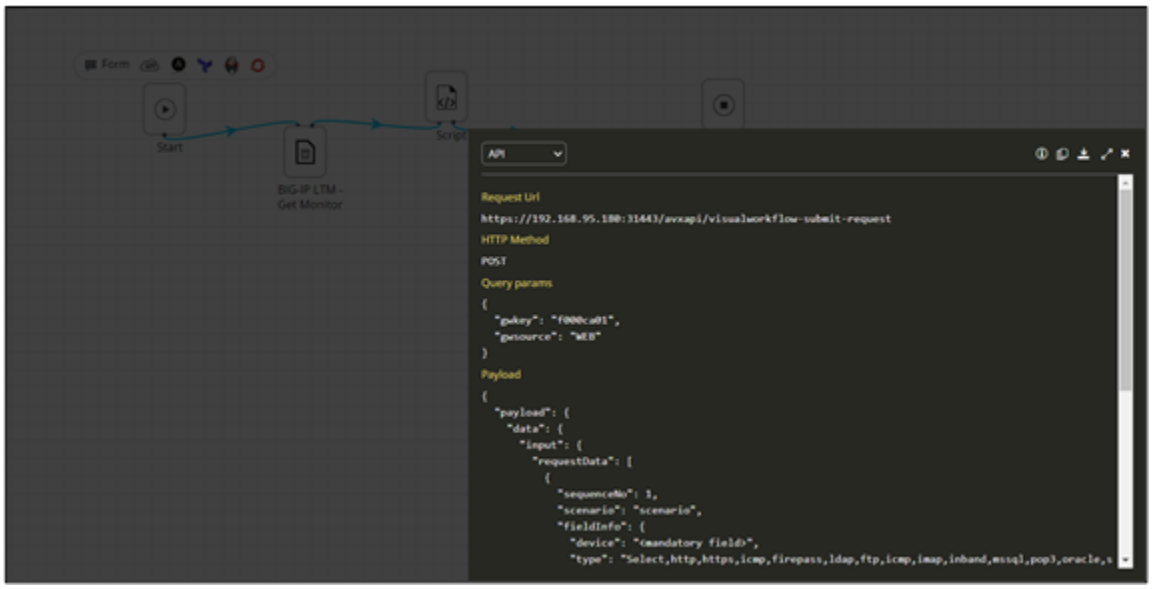


6. To generate an API for this workflow, click API above the **Start** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



The URL, HTTP method, Query params, Payload, and the Header for the API are displayed.



Note: The API can also be generated as cURL or Python.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

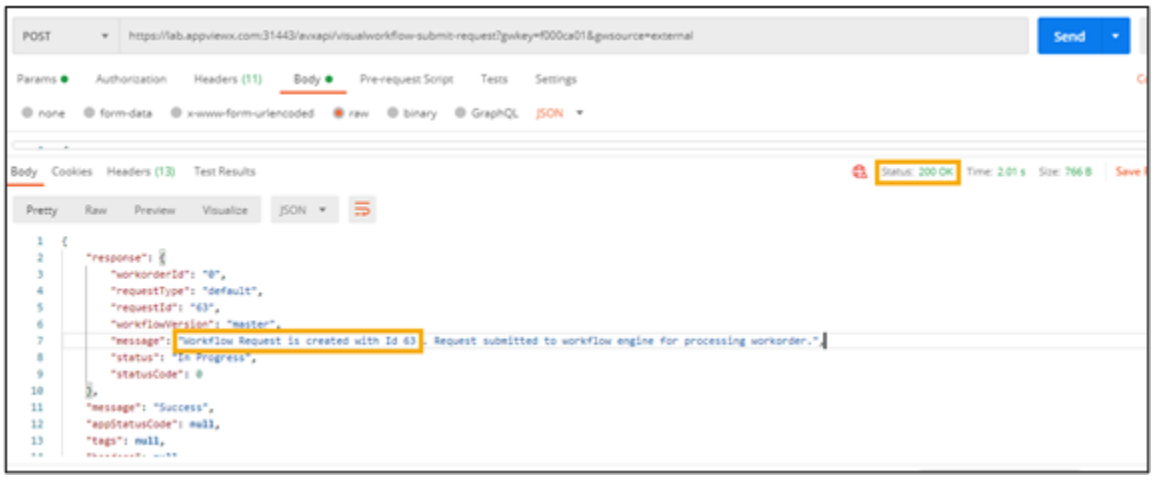
7. To download this API, form the top right corner of the API window, click .

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

8. Connect and enable the workflow.
9. To trigger the workflow from Postman, copy the URL and payload from the Workflow Studio.
10. On the Postman Utility tool, under **Body**, paste the URL and Payload details.

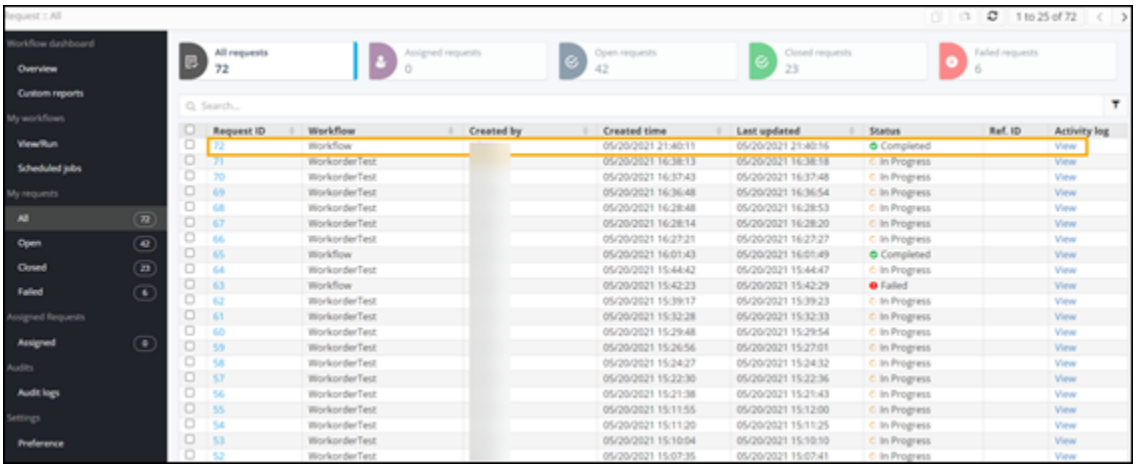
11. Under **Headers**, update the username and password details.
12. To trigger the workflow from Postman, click **Send**.

    Response with a Request ID is generated.

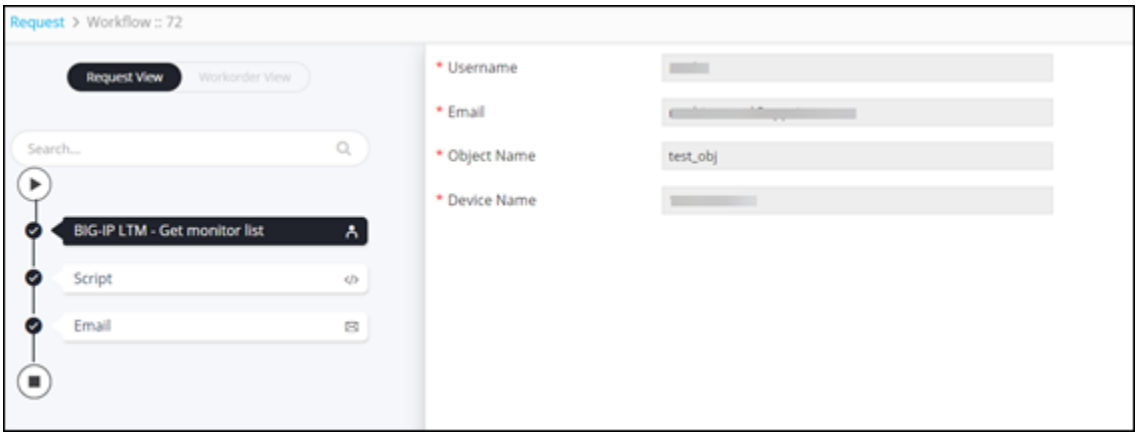


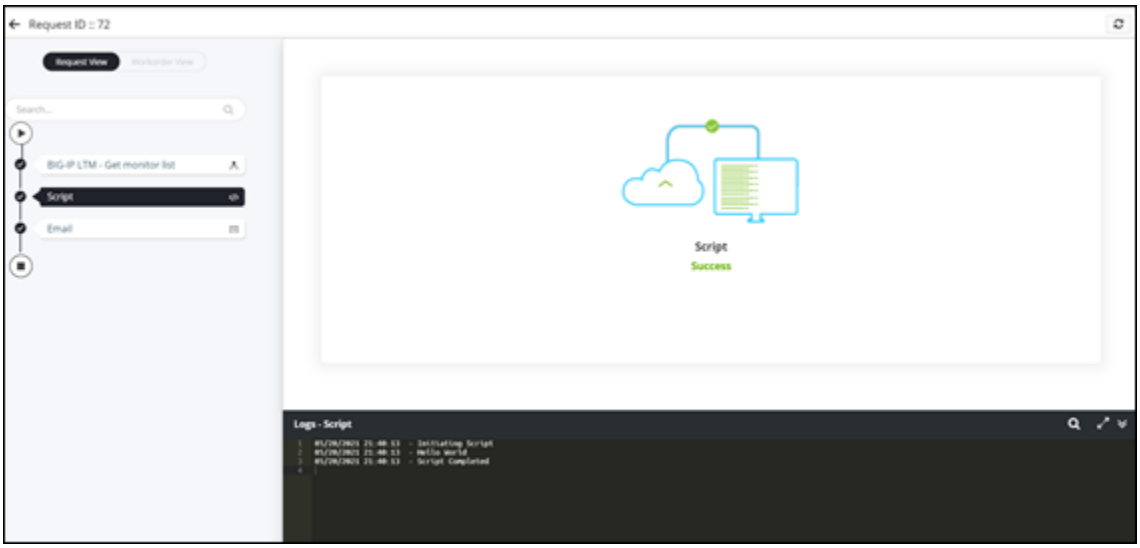
13. On the Workflow **Request :: Overview** page, under **My Requests**, select **All**.
14. On the **Request :: All** page, click on the **Request ID**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

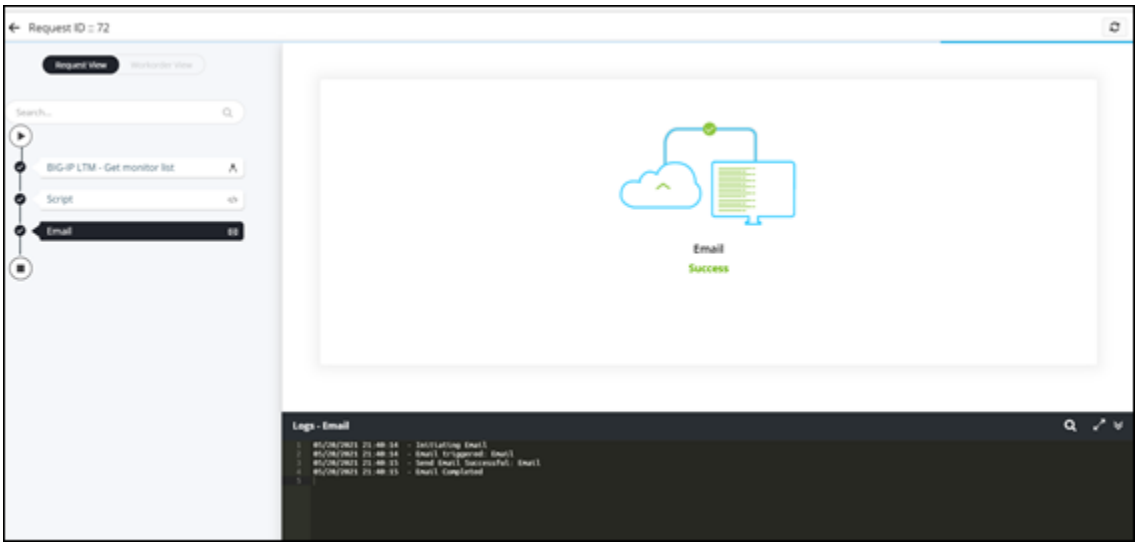


Stage-wise view of workflow task execution.





Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Jenkins - Visual Workflow Northbound Integration

This integration enables rapid network service orchestration of CI/CD Pipelines, using a Jenkins plugin, and AppViewX's Visual workflow in order to automate and orchestrate application and network security services. This section will guide you to

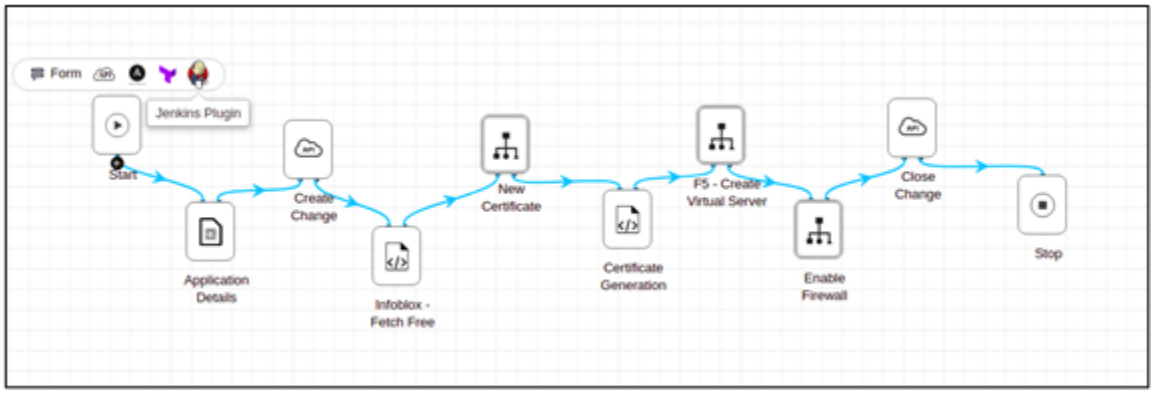
- Download a Jenkins plugin from Visual workflow
- Install the Jenkins plugin and set it up
- Execute Jenkins job to trigger workflows
- Create Pipelines
- Build the Application Service Framework via Jenkins

- Prerequisites
- Global Configuration
- Creating a Freestyle Job
- Creating a Pipeline Job
- Jenkins pipeline for Application provisioning

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Prerequisites

Download the AppViewX plugin for Jenkins, design a sample workflow, and click on the Jenkins logo (AppViewX_Jenkins_Plugin.zip):



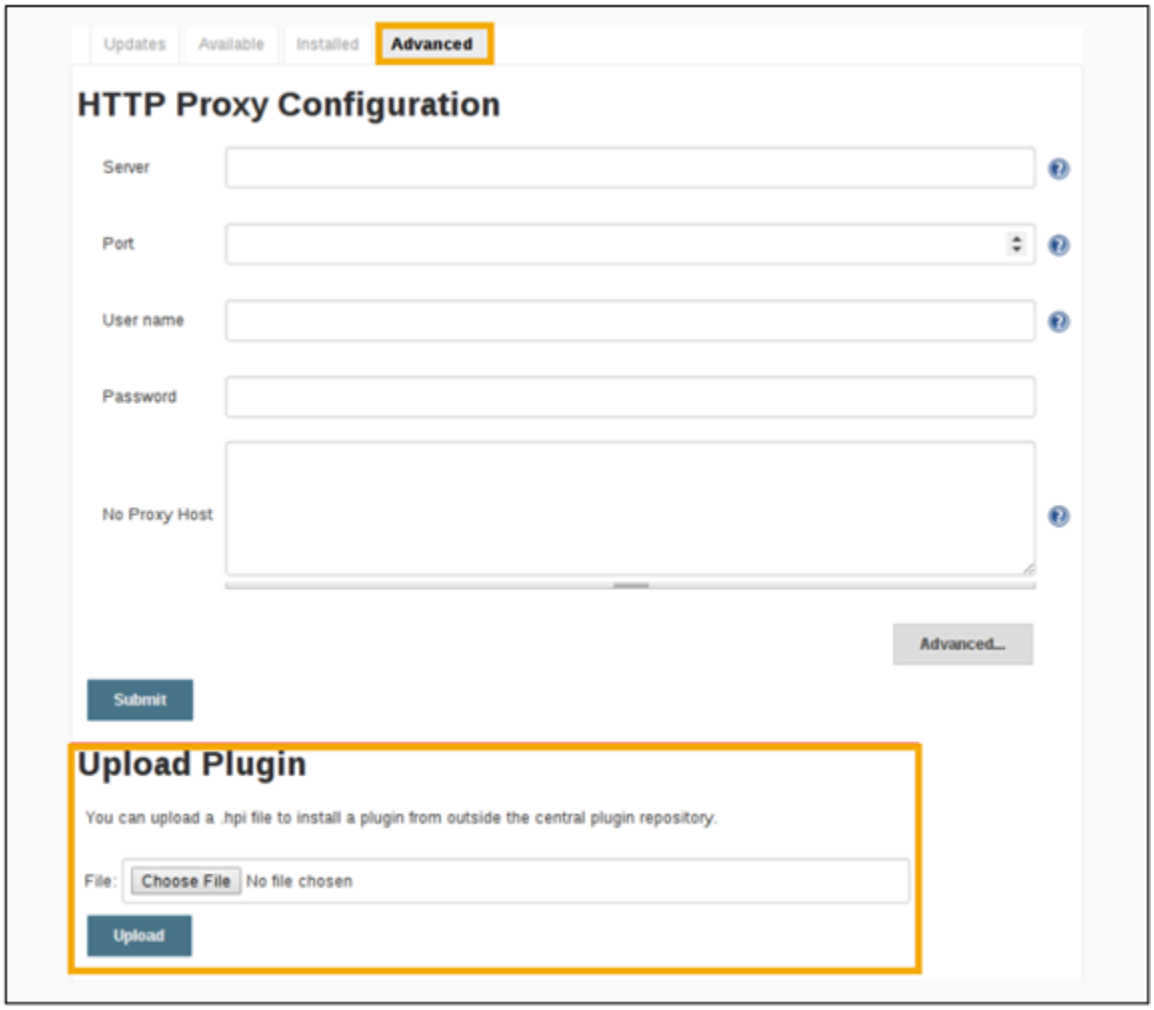
Note: Files present within the downloaded zip file are:

- AppViewX.hpi (the plugin to be installed in the Jenkins)
- Jenkins and AppViewX Integration Guide.html
- Jenkins and AppViewX Integration Guide.pdf

To install the AppViewX plugin in Jenkins:

1. Navigate to the **Manage Jenkins** > **Manage Plugins** page in the web UI (available to administrators of a Jenkins environment).
2. Click on the **Advanced** tab.
3. Under the Upload Plugin section, click the .hpi file.
4. **Upload** the plugin file.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Global Configuration

In Jenkins, credentials can be added in global configuration settings. This information can be used in the job configuration page. Click on **Check connection** to validate connectivity i.e; hostname, port and username and password.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Creating a Freestyle Job

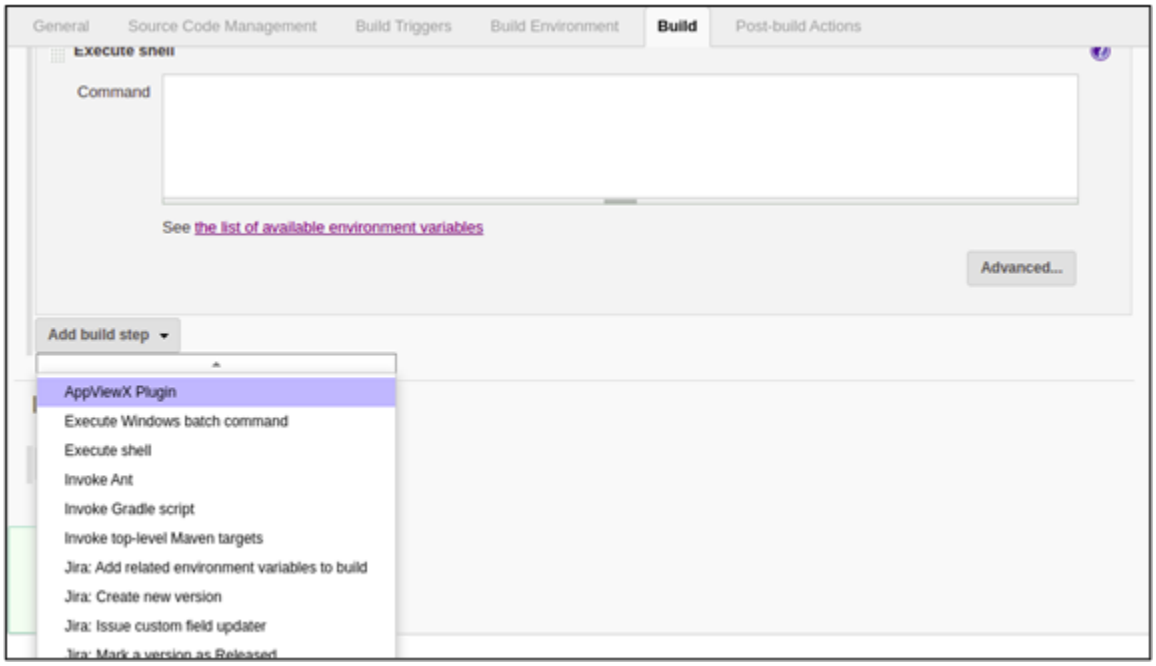
The freestyle build job is a highly flexible and easy-to-use option. It is a repeatable build job which contains steps and post-build actions.

To create a freestyle job in Jenkins:

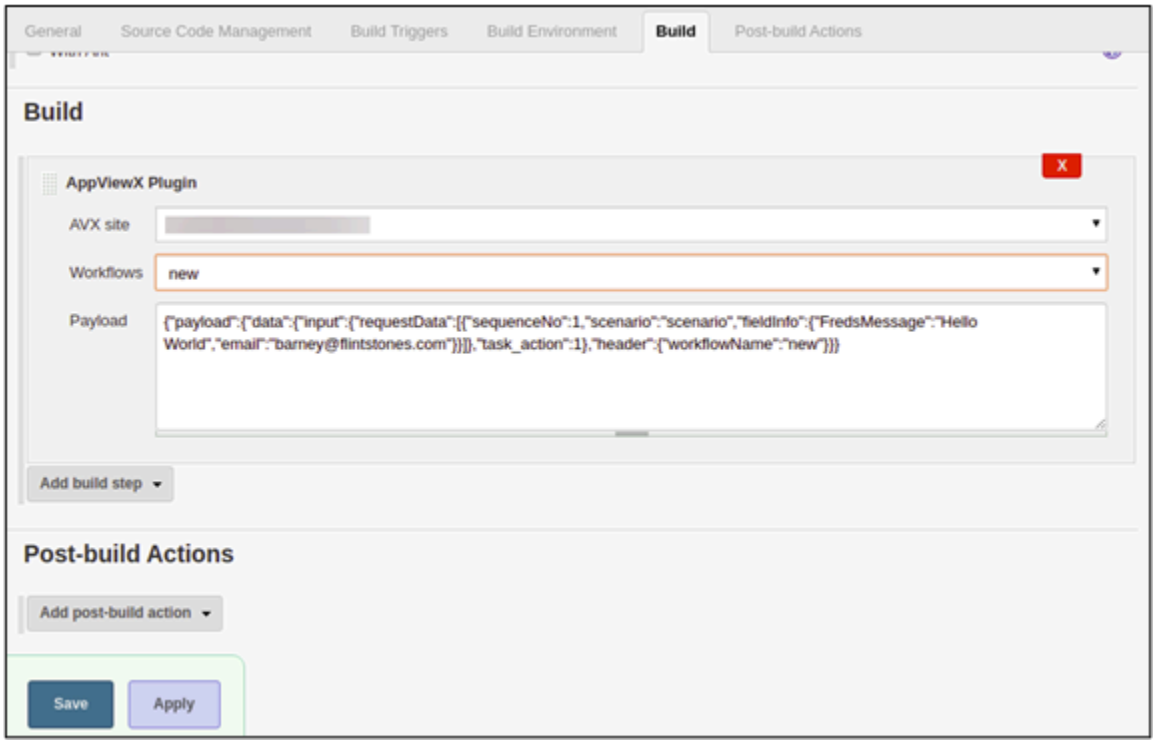
1. Click **New Item** in the upper left corner of the admin console.
2. Enter the name of the project.
3. Select **Freestyle Project**.

4. Create a build step using the AppViewX plugin in order to execute the automation workflow. The AppViewX plugin that has been installed will be available in the build step in the job configuration page.
5. Under **Add build step**, select **AppViewX Plugin** from the dropdown list.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



6. Select the AppViewX site and the required workflow from the list. On saving the configuration, the payload will be auto-populated in the payload text area.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Creating a Pipeline Job

In Jenkins, a pipeline is a group of events or jobs which are interlinked with one another in a sequence.

Use the following exposed extension points in the pipeline script in the configuration page:

- To run the workflow in AppViewX and show each task with its name, logs and status in the console output:

```
appViewXAutomation(String siteName, String workflow, String payload)
```

- To submit a workflow request in AppViewX and print the request Id in the console output:

```
appViewXRaiseRequest(String siteName, String payload, String workflow)
```

- To show the details of a set of tasks in Jenkins pipeline job (It shows the name, logs and the status of the tasks from the start task and end task):

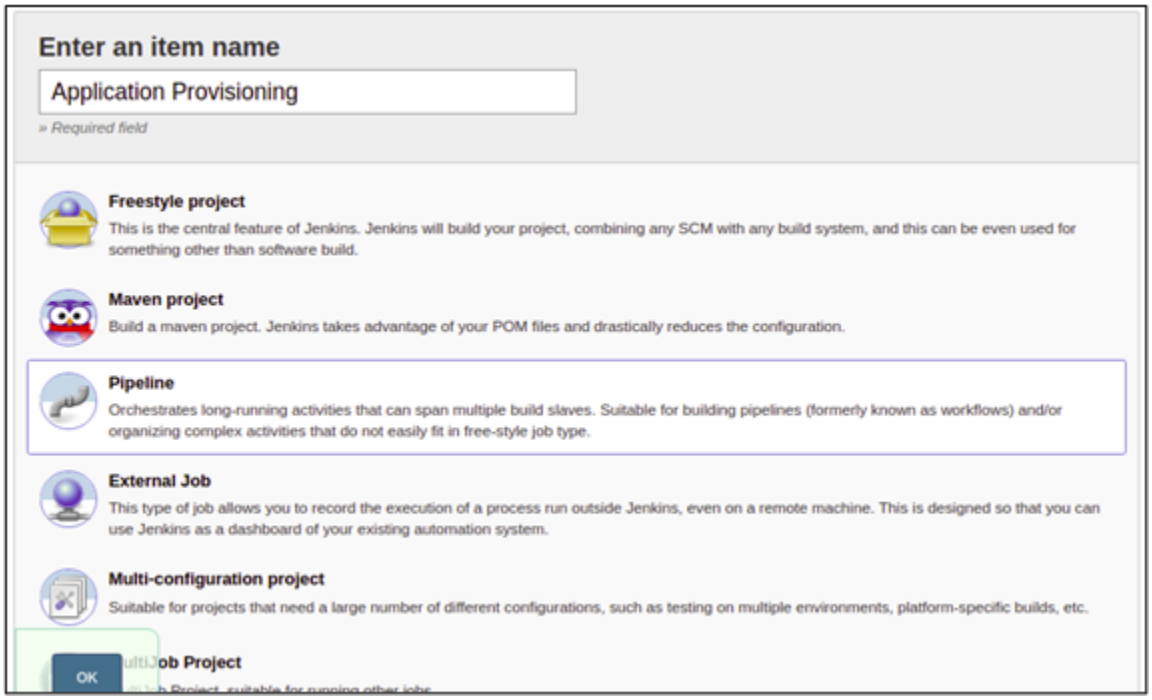
```
appViewXCheckStageStatus(String siteName, String requestId, String startTask, String endTask)
```

Jenkins pipeline for Application provisioning

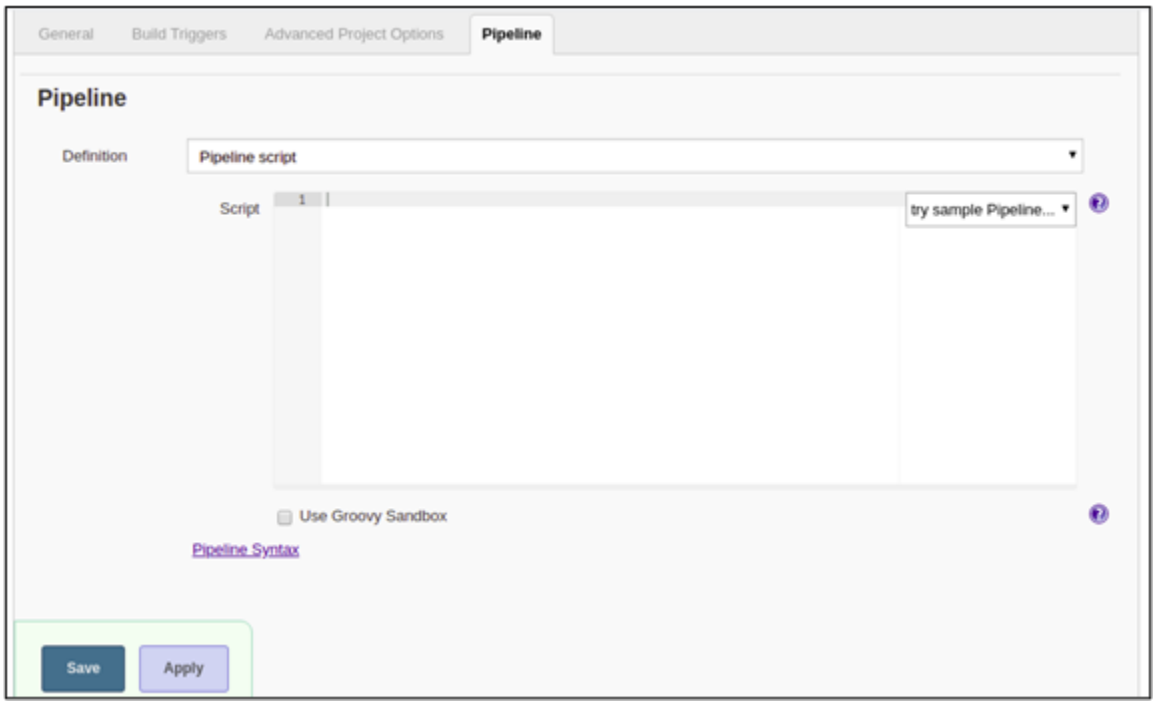
To create a pipeline job in Jenkins:

1. Click **New Item** in the upper left corner of the admin console.
2. Enter the name of the project.
3. Select **Pipeline**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



4. Under **Pipeline**, enter the pipeline script (refer to the example pipeline script given here).



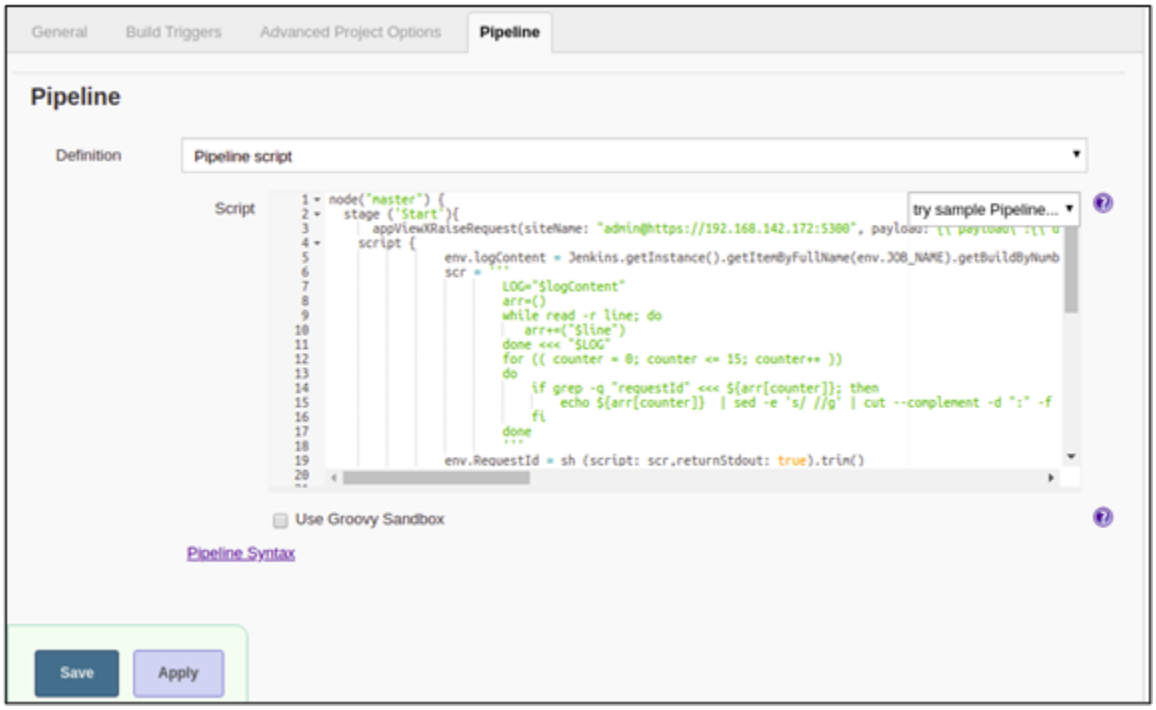
Example Pipeline Script:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

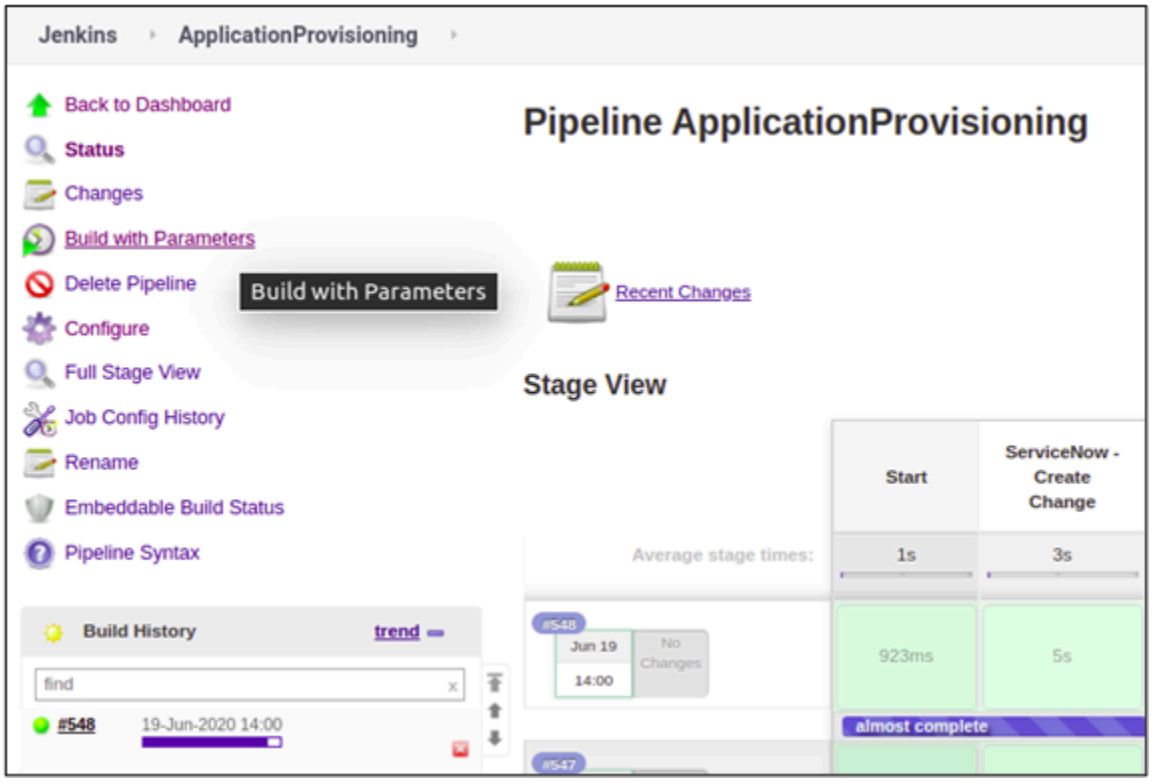
```
node("master") { stage ('Start')
{ appViewXRaiseRequest(siteName: "admin@https://192.168.142.172:5300",
payload:"{\"payload\":{\"data\":{\"input\":{\"requestData\":[{\"sequenceNo\":1,
\"scenario\":\"scenario\",\"fieldInfo\":{\"fqdn\":\"${AppName}\",\"port\":\"${Port}
\",\"name\":\"admin\"}}]},\"task_action\":1},\"header\":{\"workflowName\":
\"Application Provisioning\"}}}",workflow:"Application Provisioning")
script { env.logContent = Jenkins.getInstance().getItemByFullName(env.JOB_NAME)
.getBuildByNumber(Integer.parseInt(env.BUILD_NUMBER)).logFile.text scr = '''
LOG="$logContent" arr=() while read -r line; do arr+=("$line") done <<< "$LOG" for
(( counter = 0; counter <= 15; counter++ ))
do if grep -q "requestId" <<< ${arr[counter]};
then echo ${arr[counter]} | sed -e 's/ //g' | cut --complement -d ":" -f 1 fi done
''' env.RequestId = sh (script: scr,returnStdout: true).trim()
echo "Request Id - ${RequestId}" } } stage ('ServiceNow - Create Change')
 { appViewXCheckStageStatus(siteName:"admin@https://192.168.142.172:5300",
requestId: "${RequestId}", startTask: "",
endTask: "servicenow_create_change_2_1_1_2_2_1_3_1_1_1:Create Change Ticket") }
stage('DNS Provisioning') { appViewXCheckStageStatus
(siteName:"admin@https://192.168.142.172:5300",
requestId: "${RequestId}", startTask: "", endTask:
"script_1:Infoblox - Fetch Free IP's") }
stage('Certificate Generation') { appViewXCheckStageStatus
(siteName:"admin@https://192.168.142.172:5300", requestId: "${RequestId}",
startTask: "", endTask: "script_2:Certificate Generation") }
stage('F5 BIG IP Provisioning')
{ appViewXCheckStageStatus(siteName:"admin@https://192.168.142.172:5300",
requestId: "${RequestId}", startTask: "", endTask: "endWorkOrder_1:WorkOrder") }
stage('Firewall Provisioning') { appViewXCheckStageStatus
(siteName:"admin@https://192.168.142.172:5300",
requestId: "${RequestId}", startTask: "",
endTask: "bigip_firewall_policy_1:bigip_firewall_policy") }
stage ('ServiceNow - Close Change')
{ appViewXCheckStageStatus(siteName:"admin@https://192.168.142.172:5300",
requestId: "${RequestId}", startTask: "",
endTask: "servicenow_close_change_1_1_1_2_1_1:Close Change Ticket") } }
```

5. Add the contents and click **Save**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

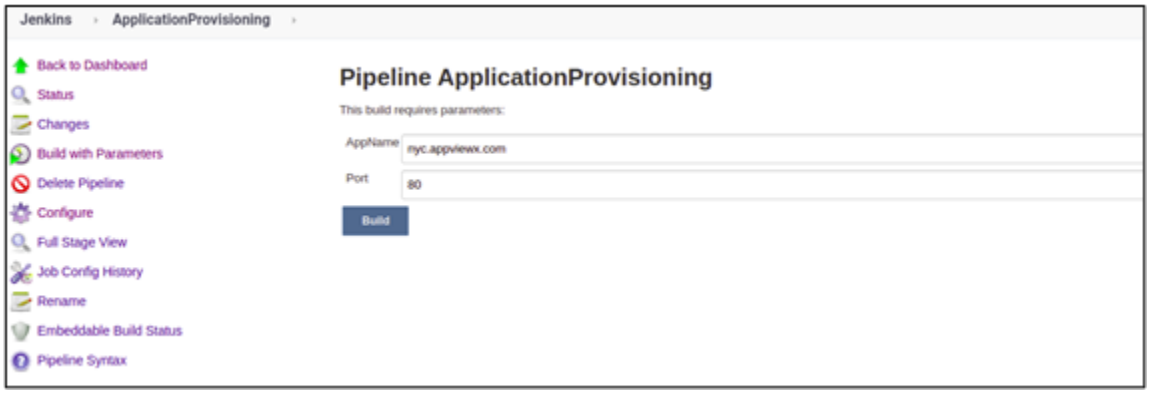


6. On the stage view page, where the Pipeline build will be executed, choose **Build with Parameters** option.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

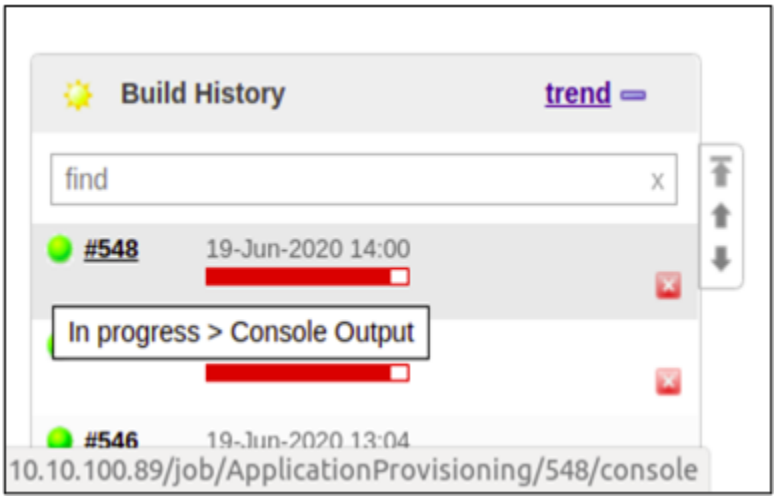
7. Provide the required inputs to run the workflow and click **Build**.



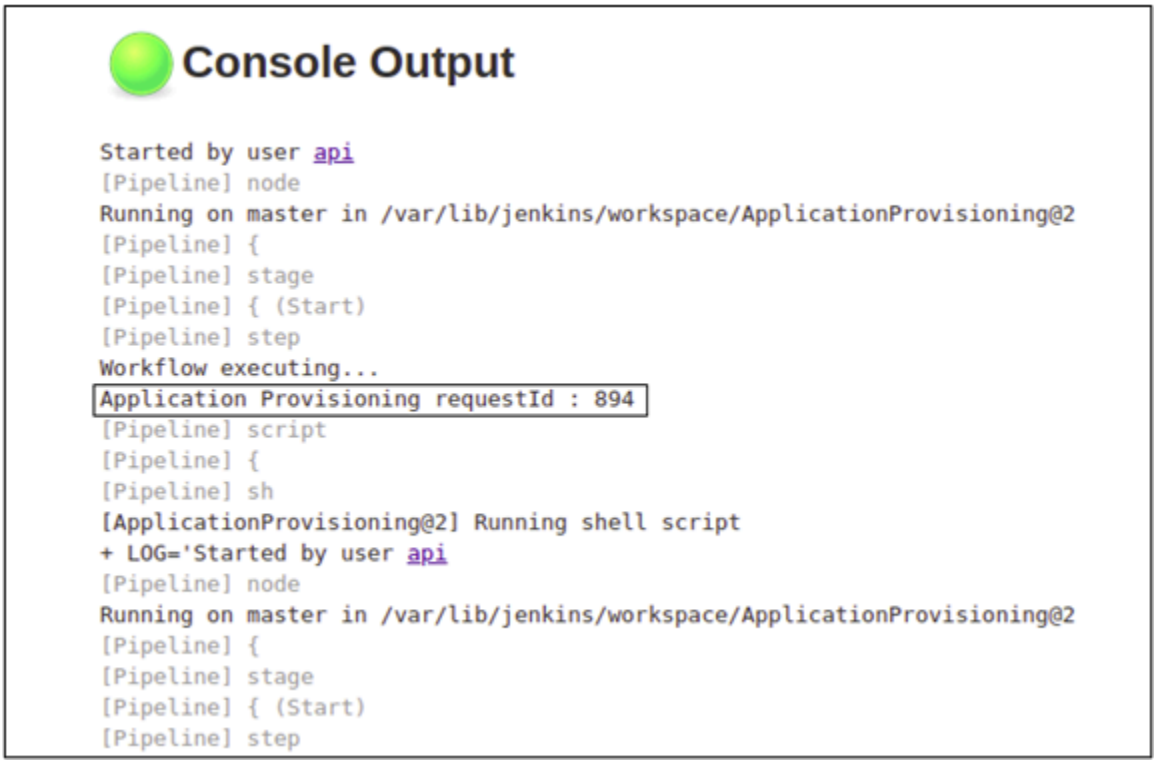
8. The stage view and console output can be seen when the pipeline is executed.
   - Pipeline stages

   - Console Output: Select the globe near the build number under Build History to see the outputs.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



- The request Id of the workflow is shown in the console logs.

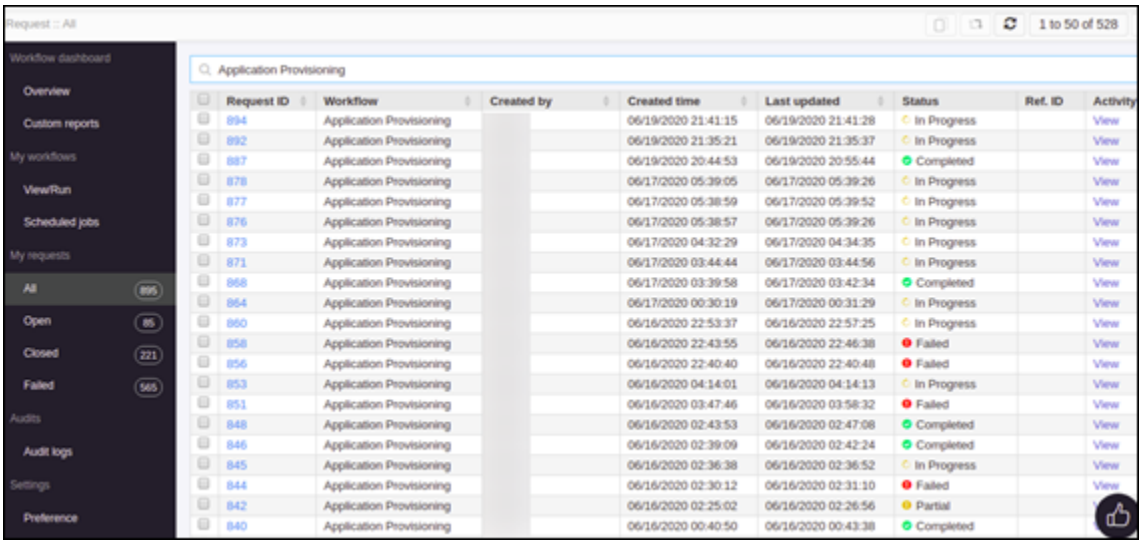


9. The task name, task logs and task status for each task in the workflow also shown in the console logs.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

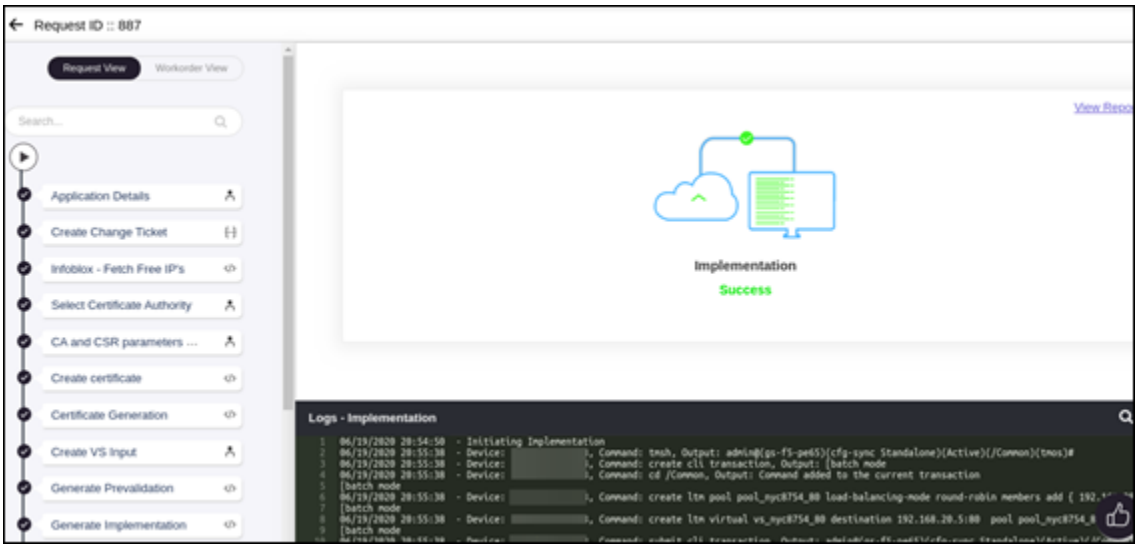
```
task name : Infoblox - Fetch Free IP's
Initiating Infoblox - Fetch Free IP's
reserved
Ip's - [                l', '                ', '                ']
Infoblox - Fetch Free IP's Completed
task status : Success


task name : Select Certificate Authority
Select Certificate Authority Completed
task status : Success


task name : CA and CSR parameters - AppViewX
Form has been submitted by user:admin
CA and CSR parameters - AppViewX Completed
task status : Success
```

10. To see all the workflows triggered from Jenkins in AppViewX, on the Workflow **Request** page, under **My Requests**, select **All**.

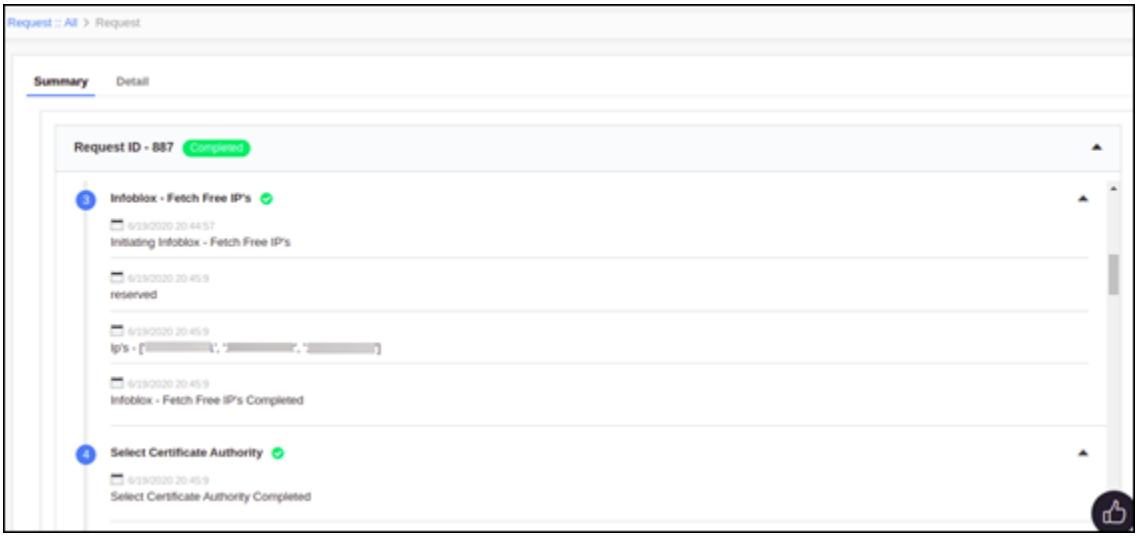


11. Click on the **Request ID** of any workflow to see the execution of each task in the workflow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



12. To see the summary of each task in the workflow, under **Activity**, click **View**.

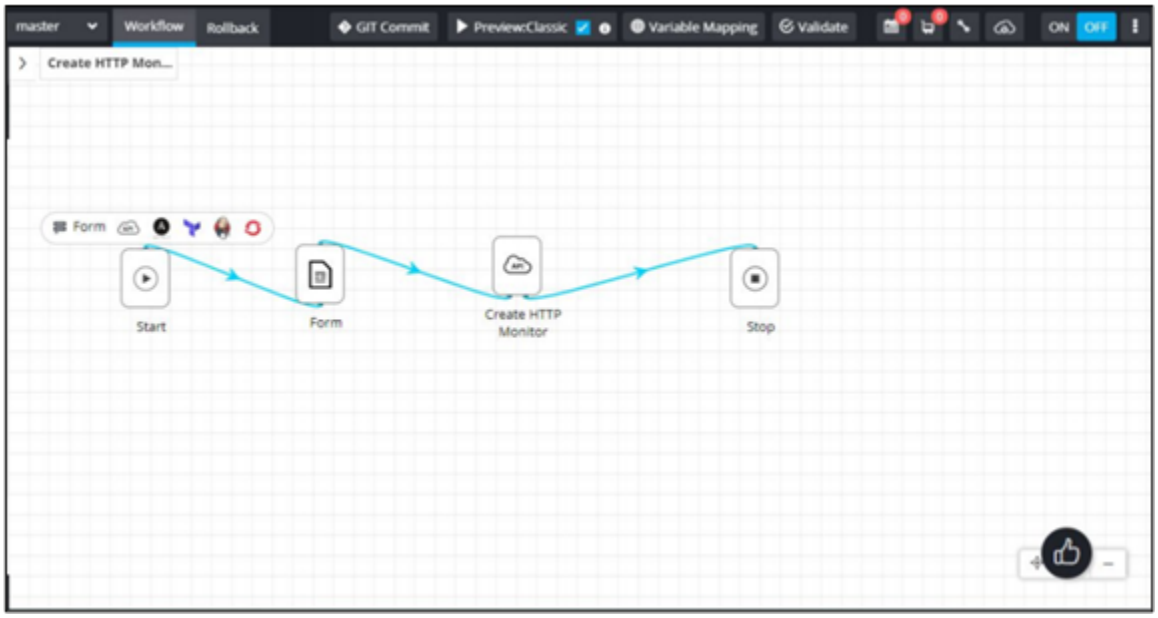


Terraform - Visual Workflow Northbound Integration

AppViewX Visual Workflow provides you with an interface to download Terraform (TF) files for all automation workflows. A TF file can be used with AppViewX Provider to submit automation requests on AppViewX Visual Workflow from Terraform.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. Design a new workflow.



2. To download the .zip file, click above the **Start** task.

   You can also click on the Service Orchestration icon to download the .zip file.

3. Extract the .zip file.
4. From the extracted .zip file, copy the <workflowname>.tf file to the installation path of AppViewX Terraform provider.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

5. In <workflowname>.tf file, update AppViewX instance details - AppViewX ID/FQDN, Username, Password, Protocol.

Sample:

```
provider "appviewx" {
  appviewx_username =
  appviewx_password = "
  appviewx_environment_is_https = true
  appviewx_environment_ip = "
  appviewx_environment_port =
}
```

6. Update Terraform meta arguments.

Sample:

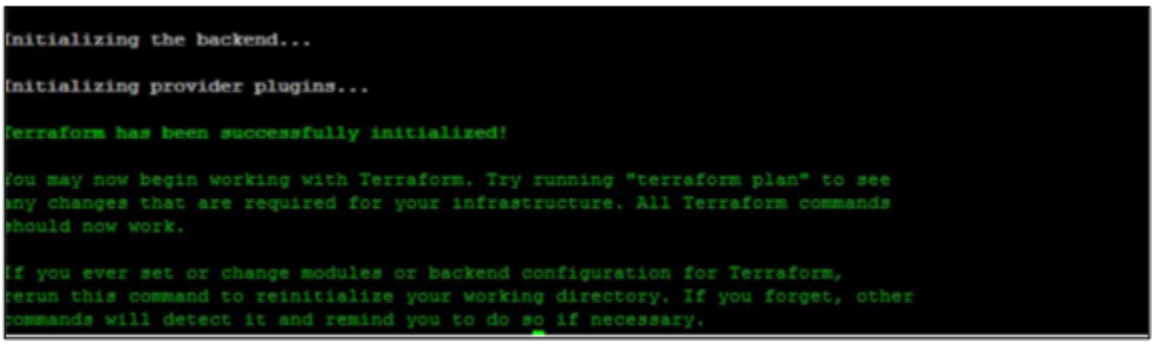
```
resource "appviewx_automation" "vip_lab.appviewx.com"
```

7. Enter workflow request details under the payload section.

Sample:

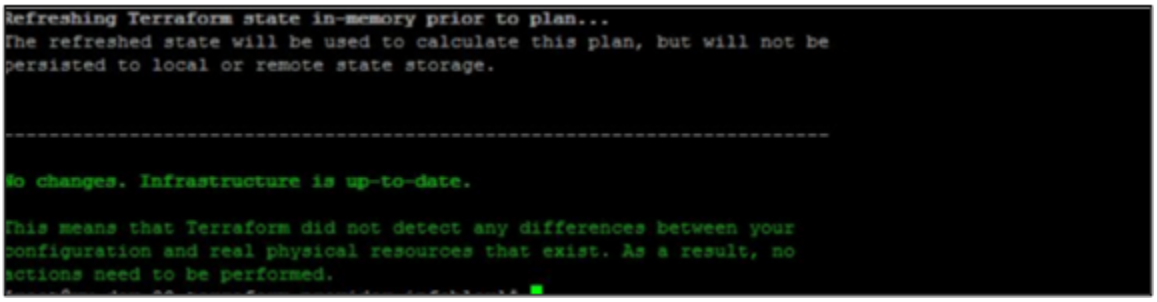
```
 payload= <<EOF
  {
 {
  "payload" : {
    "header" : {
      "workflowName" : "command task"
    }
  }
}
EOF
action_id= "visualworkflow-submit-request"
  }
```

8. Save the changes and execute the file via Terraform commands.
   - terraform init



   - terraform plan

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



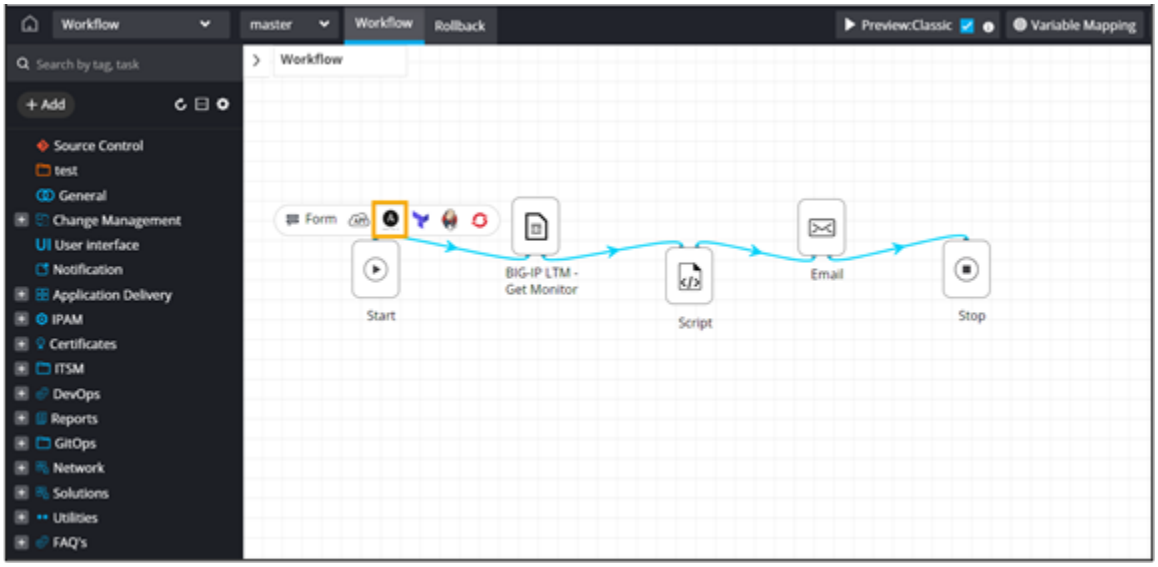
- terraform apply

Ansible - Visual Workflow Northbound Integration

You can automate network configurations through Ansible as the Northbound.

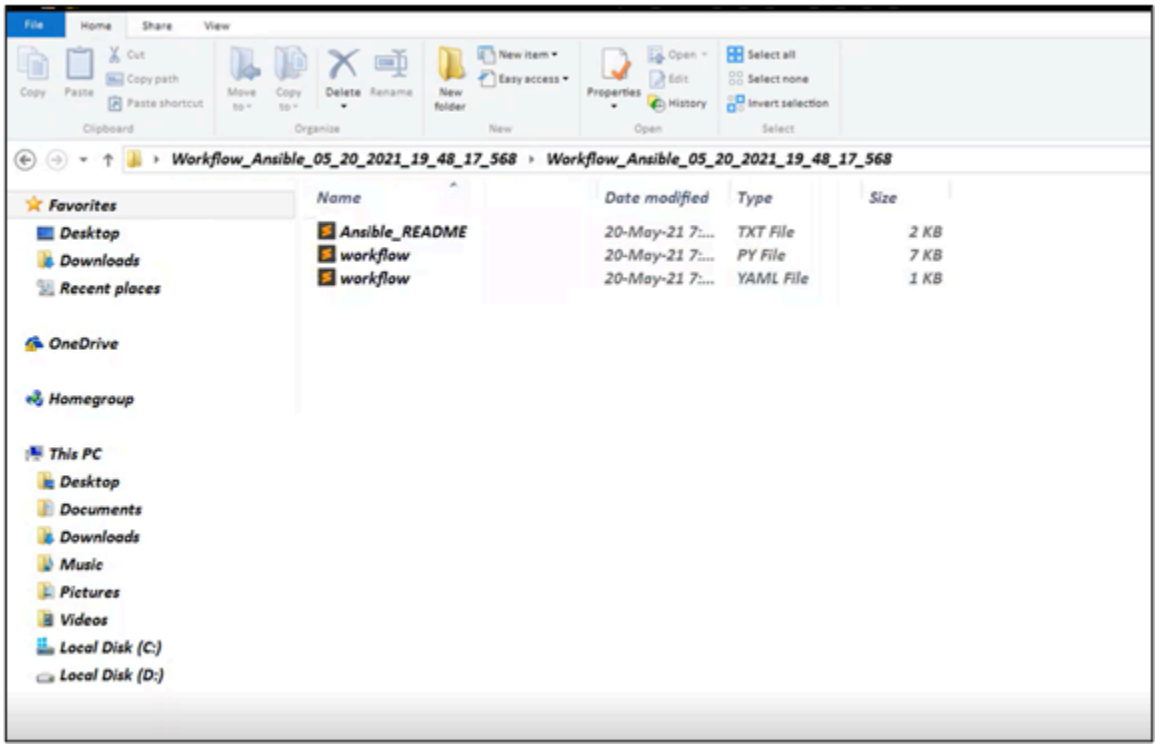
To trigger a workflow by creating an Ansible playbook:

1. Design a workflow.
2. Connect all tasks and enable the workflow.
3. To download the Ansible .zip file, click above the **Start** task.

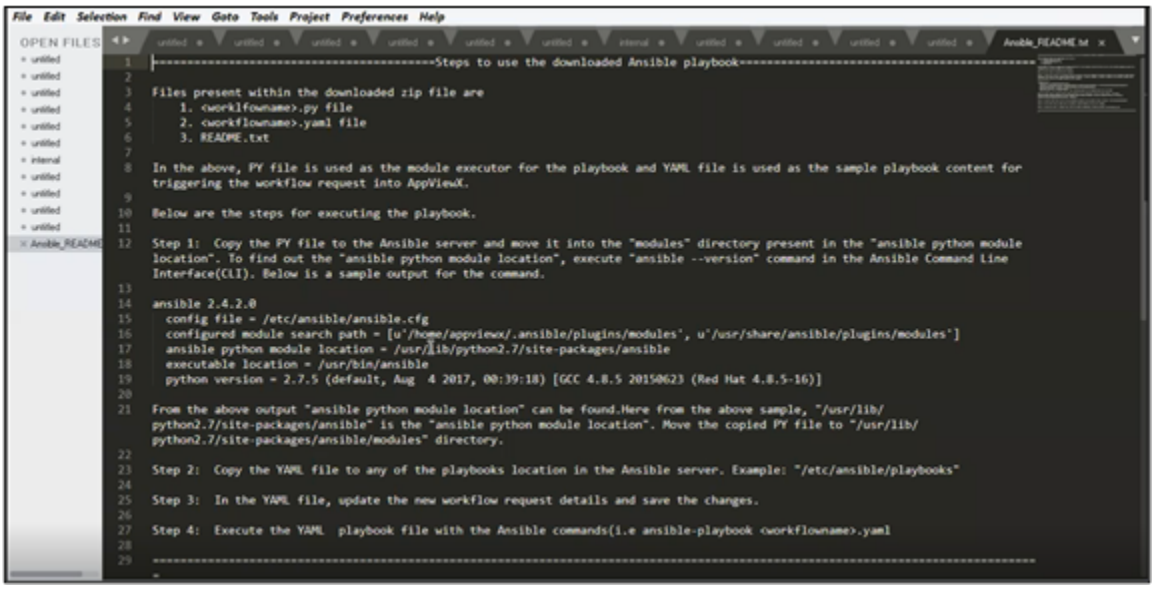


4. Extract the files.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

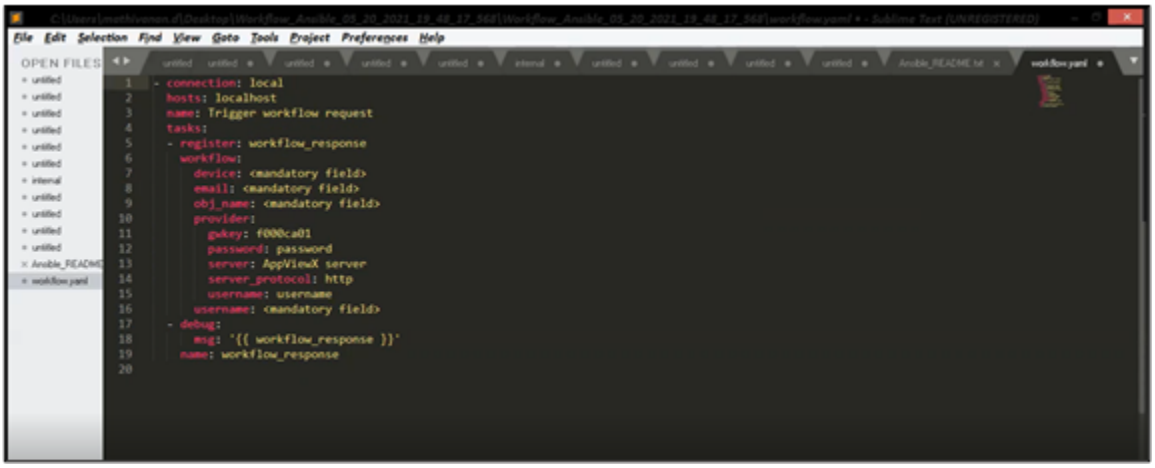


- Ansible_README txt file: This contains the steps to use the downloaded Ansible playbook.



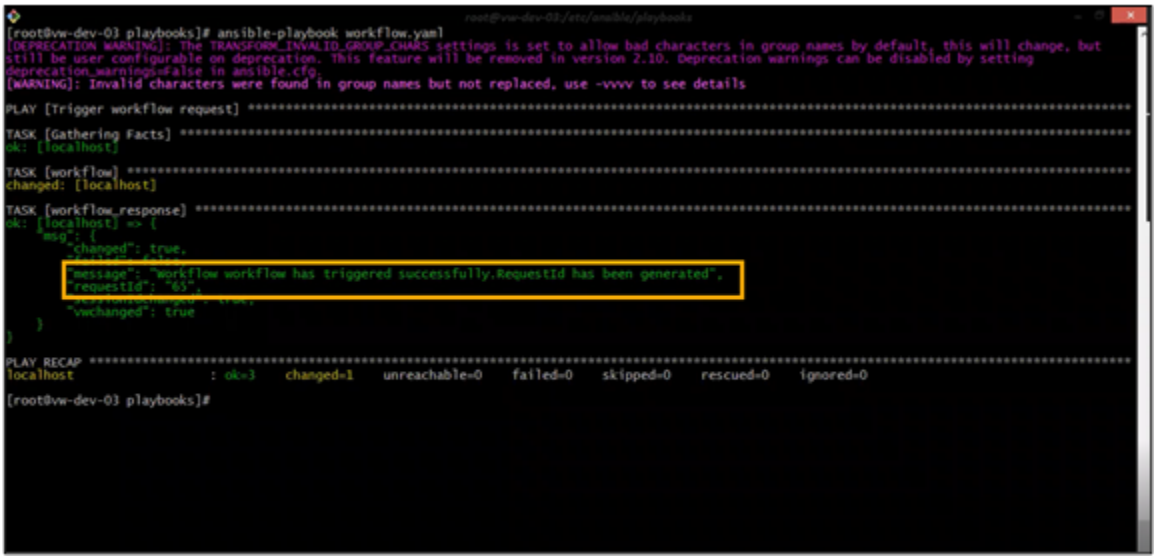
- Workflow Python file (Ansible executor file)
- Workflow YAML file

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



```
- connection: local
  hosts: localhost
  name: Trigger workflow request
  tasks:
  - register: workflow_response
    workflow:
      device: <mandatory field>
      email: <mandatory field>
      obj_name: <mandatory field>
      provider:
        gwkey: f000ca01
        password: password
        server: AppViewX server
        server_protocol: http
        username: username
      username: <mandatory field>
  - debug:
      msg: '{{ workflow_response }}'
    name: workflow_response
```

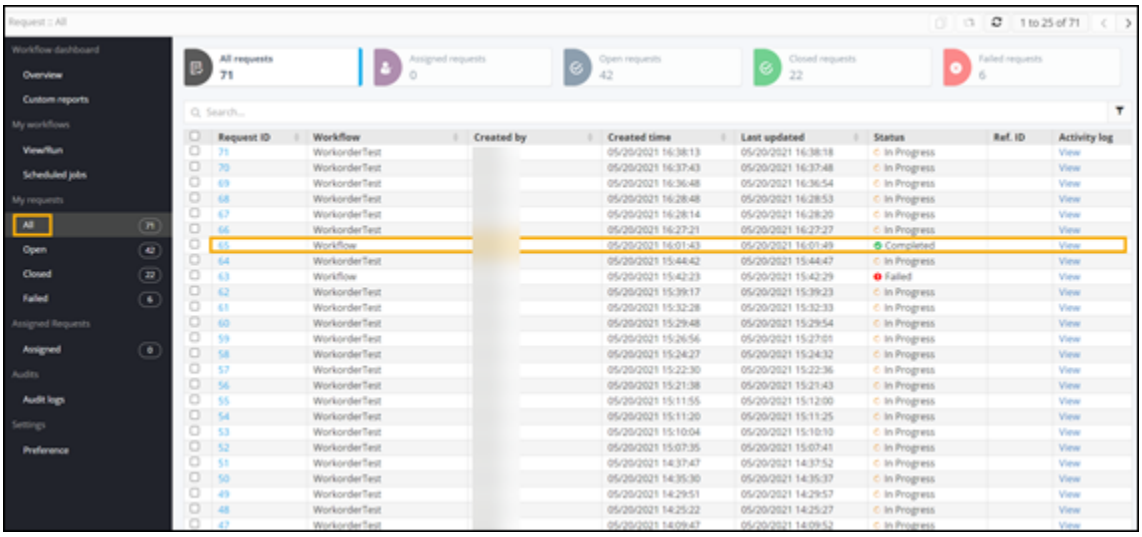
5. To trigger the workflow and generate request ID, copy the files into the Ansible server.



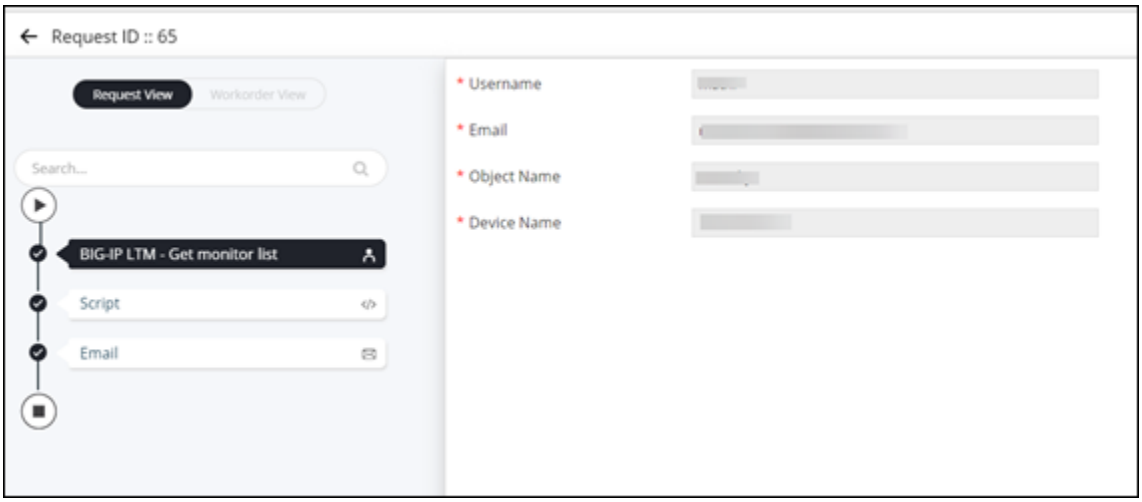
```
[root@vw-dev-03 playbooks]# ansible-playbook workflow.yaml
[DEPRECATION WARNING]: The TRANSFORM_INVALID_GROUP_CHARS settings is set to allow bad characters in group names by default, this will change, but
still be user configurable on deprecation. This feature will be removed in version 2.10. Deprecation warnings can be disabled by setting
deprecation_warnings=False in ansible.cfg.
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details

PLAY [Trigger workflow request] ************************************************

TASK [Gathering Facts] *********************************************************
ok: [localhost]

TASK [workflow] ****************************************************************
changed: [localhost]

TASK [workflow_response] *******************************************************
ok: [localhost] => {
    "msg": {
        "changed": true,
        "message": "workflow workflow has triggered successfully.RequestId has been generated",
        "requestId": "63",
        "vwchanged": true
    }
}

PLAY RECAP *********************************************************************
localhost                  : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[root@vw-dev-03 playbooks]#
```

6. On the Request :: Overview page, from the left menu, select **All**.
7. On the **Request :: All** page, click on the **Request ID** generated for the workflow.

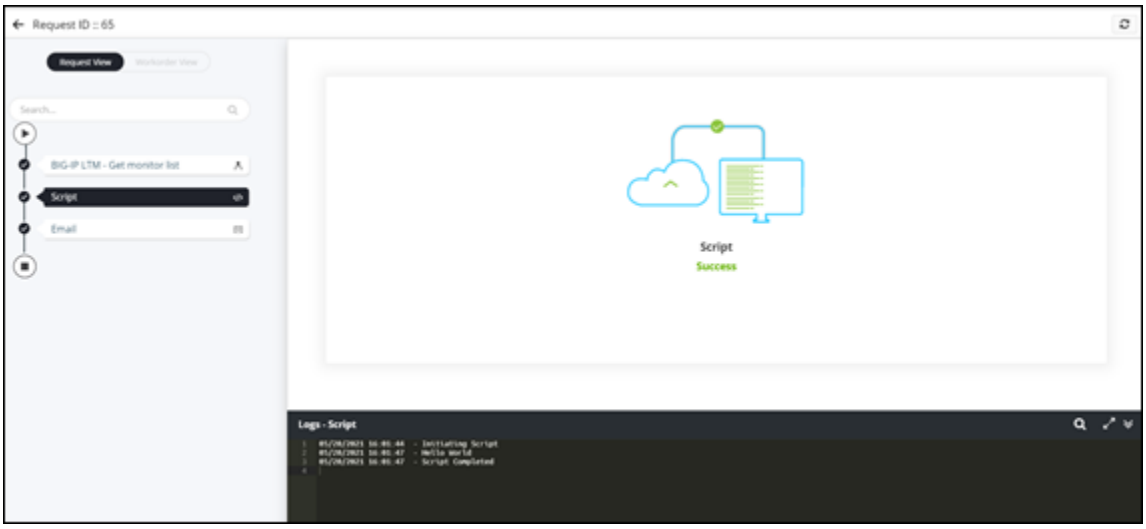
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

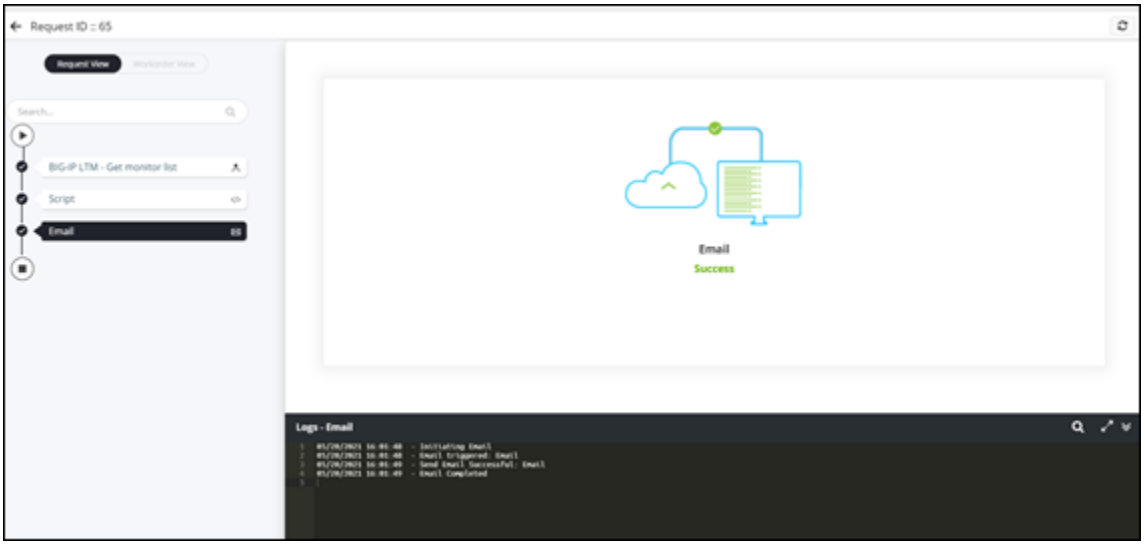


Stage-wise view of the workflow.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.





Openshift PaaS Orchestration

AppViewX's Visual Workflow provides you with an interface to install Open service Broker (OSB) that manages applications defined by AppViewX catalogs and accelerates application deployment.

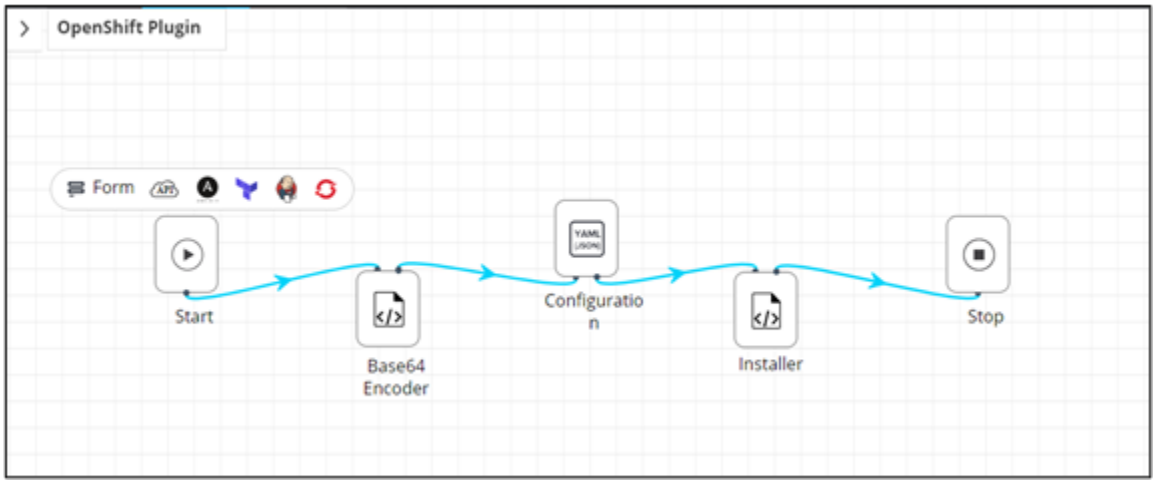
- Offers a Kubernetes container platform to manage hybrid cloud deployments.
- Allows you to connect application deployed in OSE Container Platform to a variety of service brokers.
- OSB API manages applications defined by AppView catalogs.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Enables PaaS Orchestration of application and PKI services.
- Accelerated application deployment

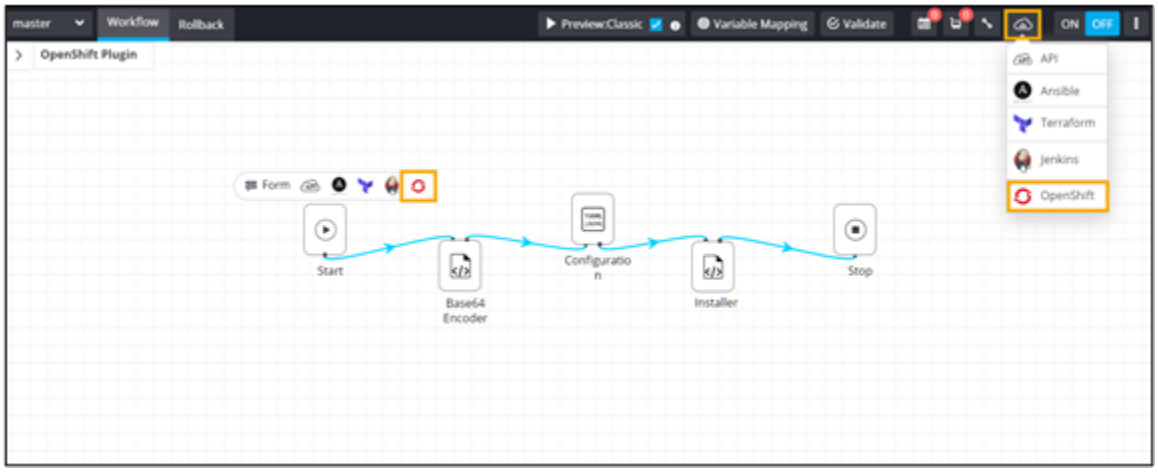
To facilitate Openshift PaaS Orchestration:

1. Design a new workflow.



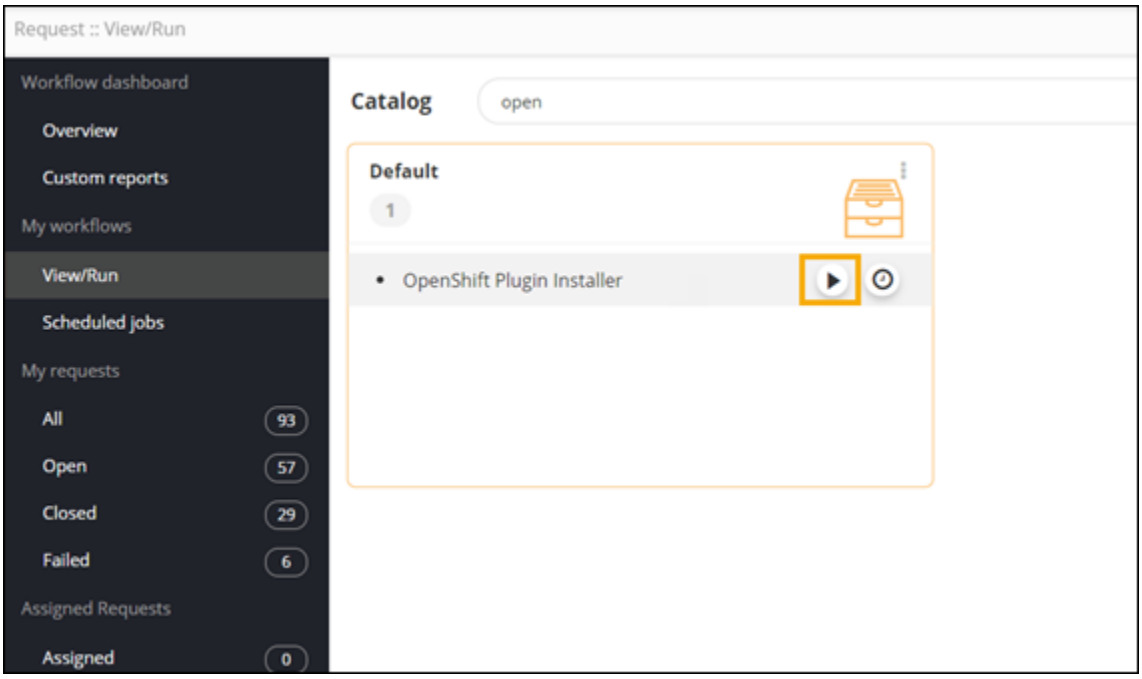
2. To download the OSB Plugin manually, click above the **Start** task.

> **Tip:** You can also click on the Service Orchestration icon to install the plugin.

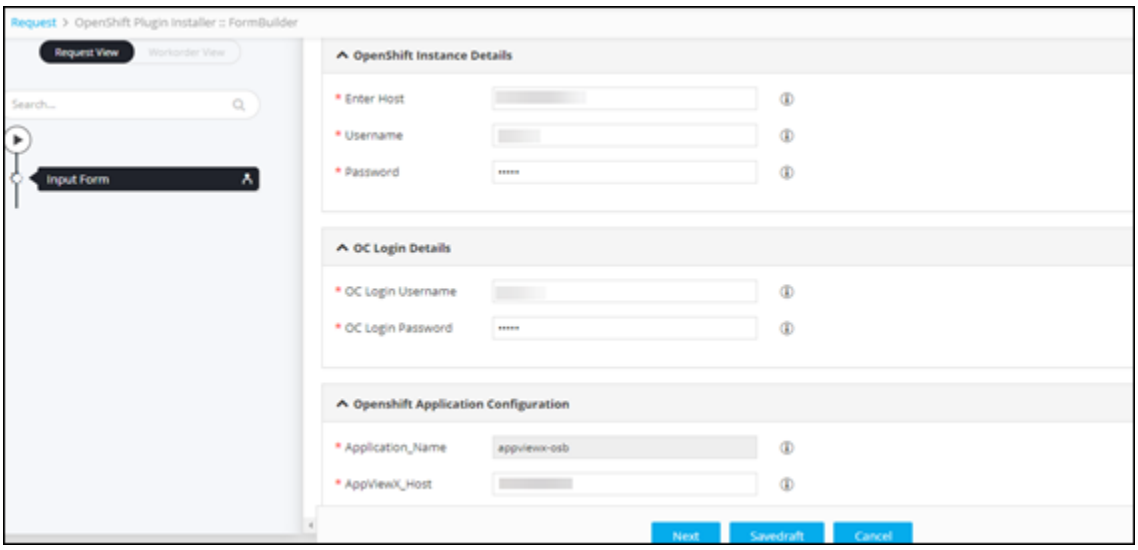


3. To install the AppViewX OSB plugin automatically, from the upper left corner of the screen, click .
4. Trigger the OpenShift Plugin Installer workflow from the Request :: View/Run page.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

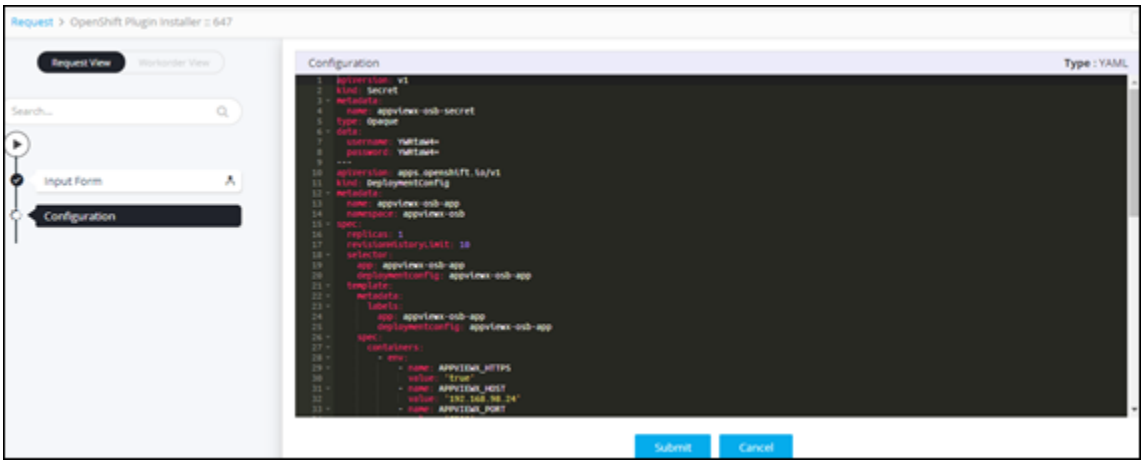


5. Enter the OSB details in the fields.

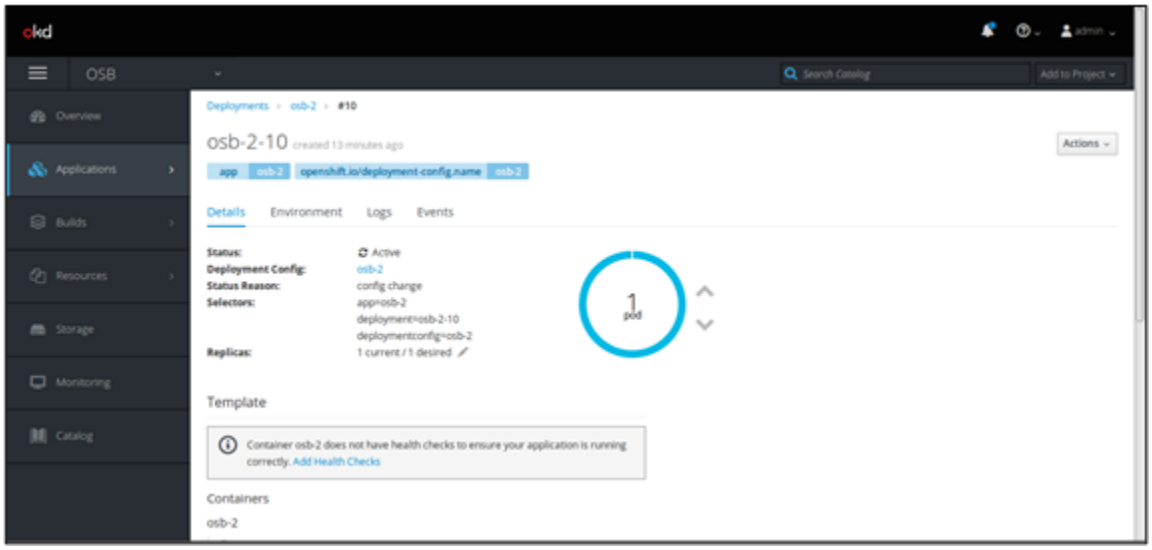


6. To proceed with plugin installation, click **Next**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

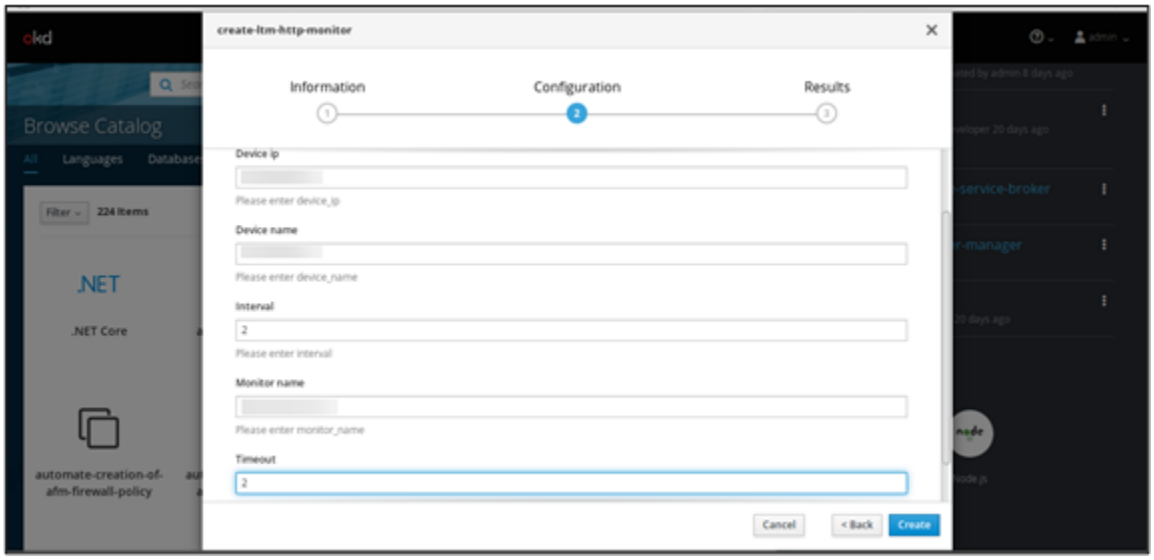


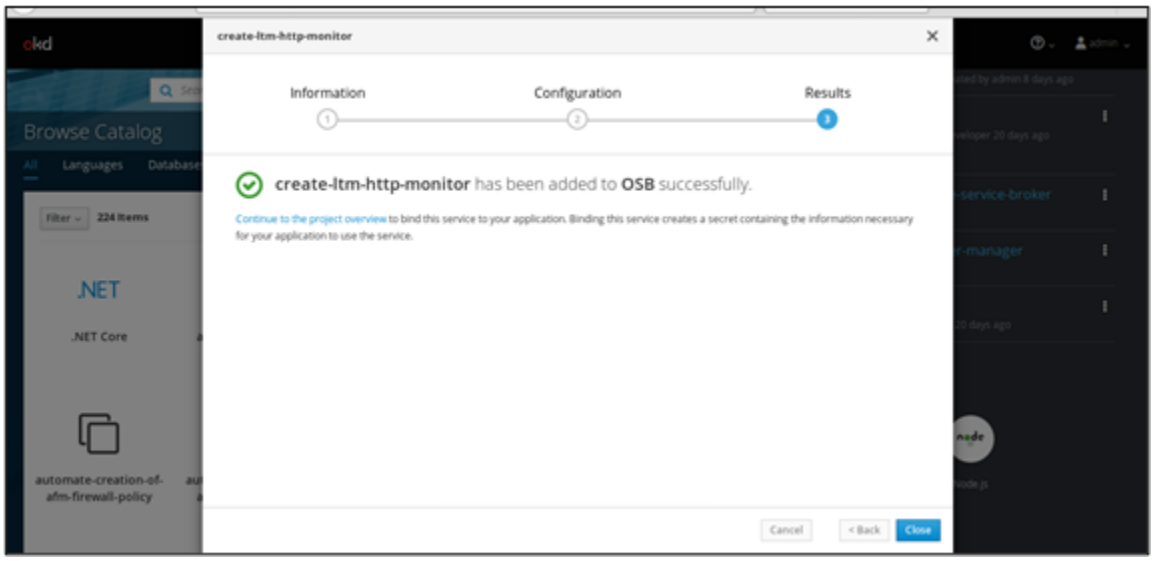
7. Open the AppViewX Open Service Broker.



8. View all automation catalogs in OpenShift once the AppViewX broker is installed.
9. To provision application and security services through AppViewX, click on any Automation catalog.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.





Note: The following REST API is used to orchestrate and provision services between openshift and appviewx.

- **API**: visualworkflow-get-all-workflows

  Description: To fetch the list of workflows in the AppViewX environment

  Method: POST

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Type	Name	Mandatory	Description
Header	username	yes	AppViewX username
Header	password	yes	AppViewX password
Header	Content-Type	yes	application/json
Header	Accept	yes	application/json
Query	gwkey	yes	Tenant key
Query	gwsource	yes	Source from which the request is triggered (Example, external)

Format: https://<application ip>:<gatewayport>/avxapi/visualworkflow-get-all-workflows?gwkey=<tenant key>&gwsource=<gateway source>

Sample Payload: {"payload":{"sSearch":""}}

Sample Response:{"response":{"workflowTemplateList":[{"workflowName":"Create LTM HTTP Monitor","accessType":"RW","properties":{"version":"19.3.0","category":38,"categoryName":"F5 BIG-IP","subCategory":2,"subCategoryName":"Ansible","internalWorkflow":false,"readOnly":false,"hidden":false,"locked":false},"settings":{"disableRequestorSubmission":false,"tags":[],"hideWorkflow":false,"enableRequest":true,"confirmationAlert":true},"globalData":{},"_id":"5f1011f4b958f463955d2b6e","description":"","created_by":"admin","status":"Enabled","workflowCategory":"Default","workflowVersion":"mas

- **API**: visualworkflow-generate-api

  Description: To generate the payload for specific workflow

  Method: POST

Type	Name	Mandatory	Description
Header	username	yes	AppViewX username
Header	password	yes	AppViewX password
Header	Content-Type	yes	application/json
Header	Accept	yes	application/json
Query	gwkey	yes	Tenant key
Query	gwsource	yes	Source from which the request is triggered (Eg: external)

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Format: https://<application ip>:<gateway port>/avxapi/visualworkflow-generate-api?gwkey=<tenant key>&gwsource=<gateway source>

Sample Payload: {"payload":{"workflowName":"Create LTM HTTP Monitor","workflowType":"Default","workflowVersion":"master"}}

Sample Response: {"response":{"url":"<protocol>://<IP>:<port>/avxapi/visualworkflow-submit-request?gwkey=f000ca01&gwsource=WEB","actionId":"visualworkflow-submit-request","httpMethod":"POST","payload":{"payload":{"data":{"input":{"requestData":[{"sequenceNo":1,"scenario":"scenario","fieldInfo":{"device_ip":"192.168.40.214","device_name":"192.168.40.214","interval":"5","timeout":"2","monitor_name":"demmoo_monitor"}}]},"task_action":1},"header":{"workflowName":"Create LTM HTTP Monitor"}}}},"message":null,"appStatusCode":null,"tags":null,"headers":null}

- **API**: visualworkflow-submit-request

  Description: To submit a workflow request in AppViewX

  Method: POST

Type	Name	Mandatory	Description
Header	username	yes	AppViewX username
Header	password	yes	AppViewX password
Header	Content-Type	yes	application/json
Header	Accept	yes	application/json
Query	gwkey	yes	Tenant key
Query	gwsource	yes	Source from which the request is triggered (Eg: external)

  Format: https://<application ip>:<gateway port>/avxapi/visualworkflow-submit-request?gwkey=<tenant key>&gwsource=<gateway source>

  Sample Payload: "payload": { "data": { "input": { "requestData": [{ "sequenceNo": 1, "scenario": "scenario", "fieldInfo": { "device_ip": "192.168.40.214", "device_name": "192.168.40.214", "interval": "5", "timeout": "2", "monitor_name": "ltm_monitor_test" } }] }, "task_action": 1 }, "header": { "workflowName": "Create LTM HTTP Monitor" } } }

  Sample Response: {"response":{"workorderId":"0","requestType":"default","requestId":"38","workflowVersion":"master","message":"Workflow Request is created with Id 33 . Request submitted to workflow engine for processing workorder.","status":"In Progress","statusCode":0}}

Ansible Southbound Integration

AppViewX allows you to integrate with Ansible in order to automate and orchestrate the network.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Provision to integrate with an existing Ansible instance.
- Provision to use the Ansible executor task when there is a need for automating network configurations through Ansible as the southbound.
- Provision to Discover and Import playbooks from a specific path within the visual workflow studio.
- Drag & drop playbooks and automate using Ansible modules via visual workflow.

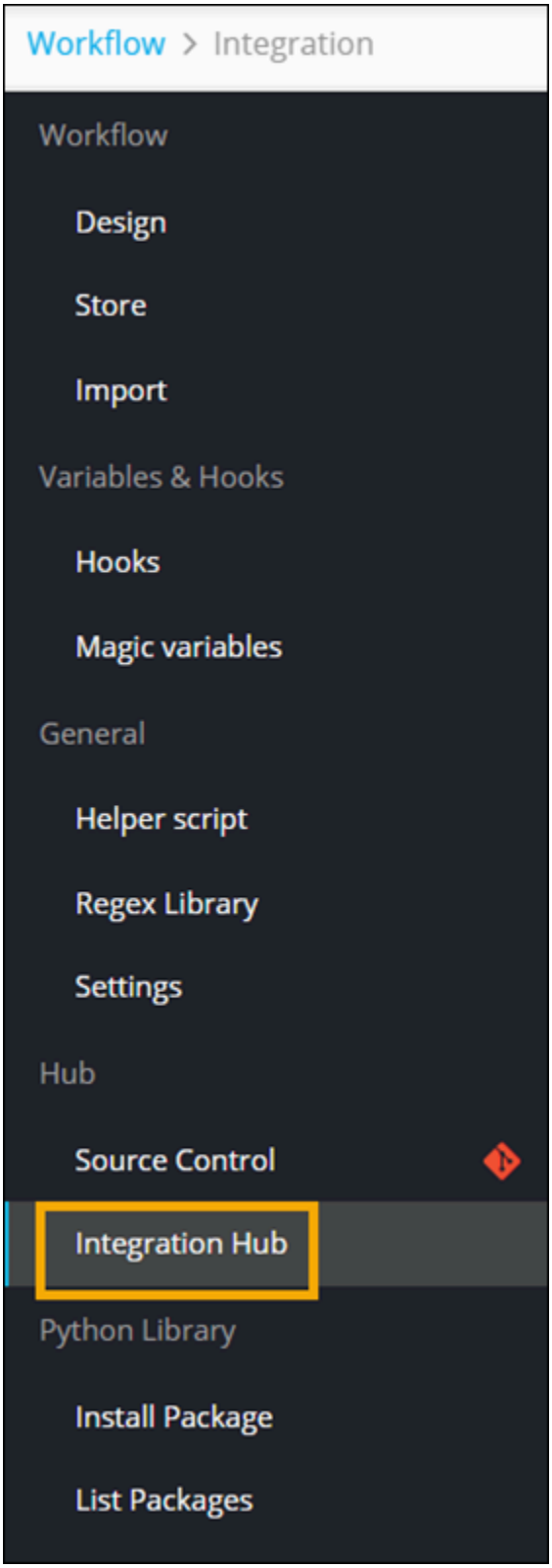
- Configuring Ansible Instance
- Discovering Playbooks
- Using Variables within a YAML Task
- Getting Device Credentials Dynamically
- Ansible Executor

Configuring Ansible Instance

An existing Ansible instance can be configured and integrated, which allows for reusing playbooks, and for south bound implementation via AppViewX. One or more instances can be configured.

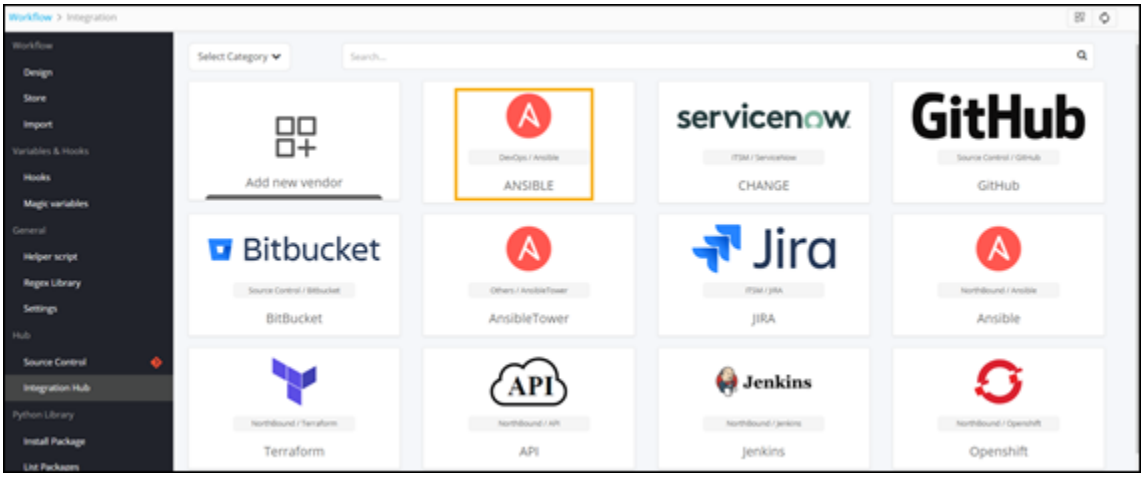
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. On the Workflow Inventory page, from the navigation pane on the left, click **Integration Hub**.

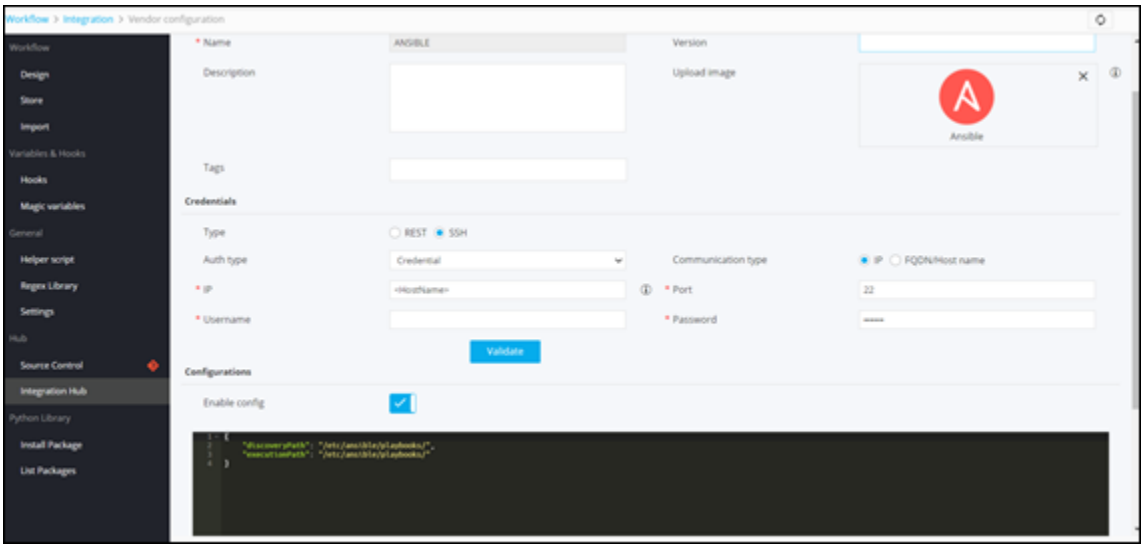


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

2. To configure an Ansible instance, on the **Integration** page, click **Ansible**.



3. On the **Vendor configuration** page, enter the field information under the **General** section.



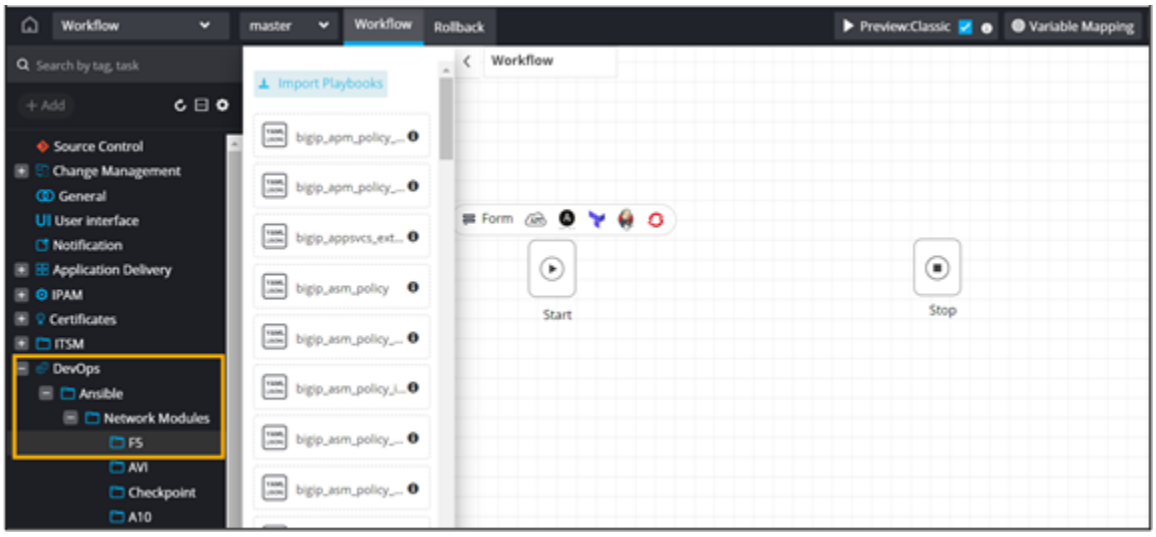
4. Define the folder/path from where existing playbooks can be discovered within the workflow studio.
5. Click **Save**.

Discovering Playbooks

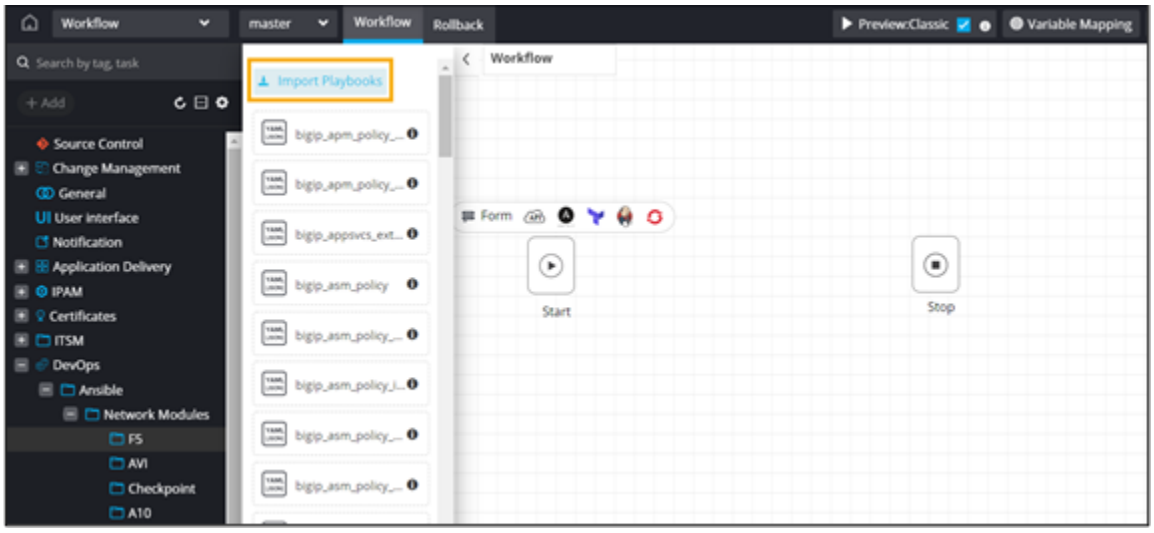
Once an Ansible instance is configured, playbooks can be discovered from the defined path and reused within the workflow design studio for further automation.

1. Design/Modify a workflow.
2. From the menu on the left, under **DevOps**, select **Ansible** > **Network Modules** > **F5**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



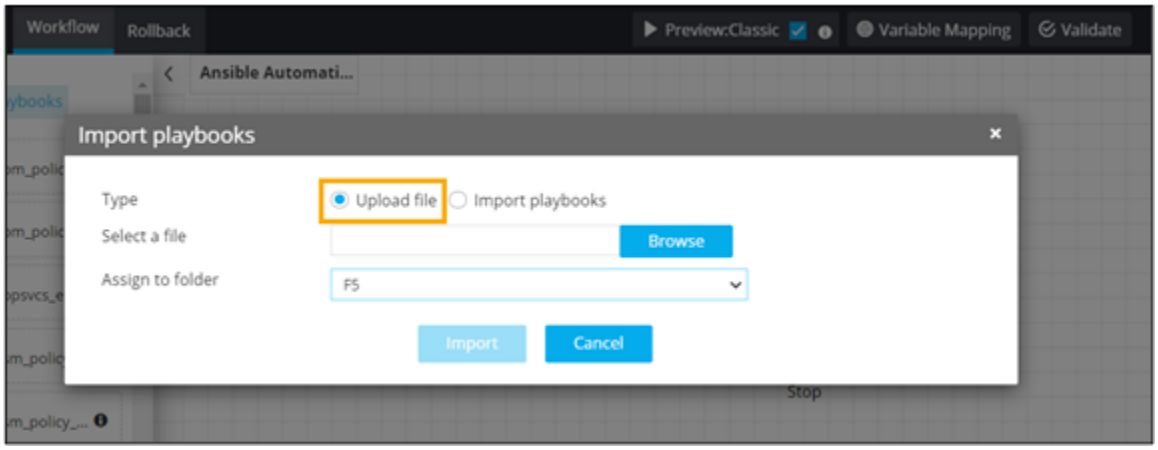
3. From the **F5** folder, click **Import Playbooks**.



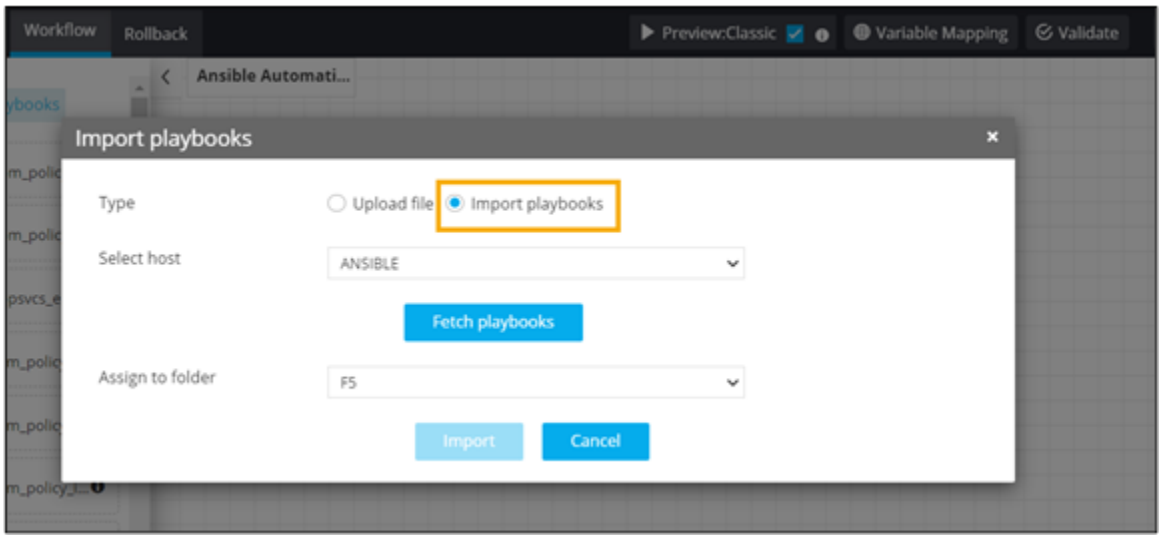
4. In the **Import playbooks** window, select import **Type**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- **Upload file**: Upload playbooks from a local folder

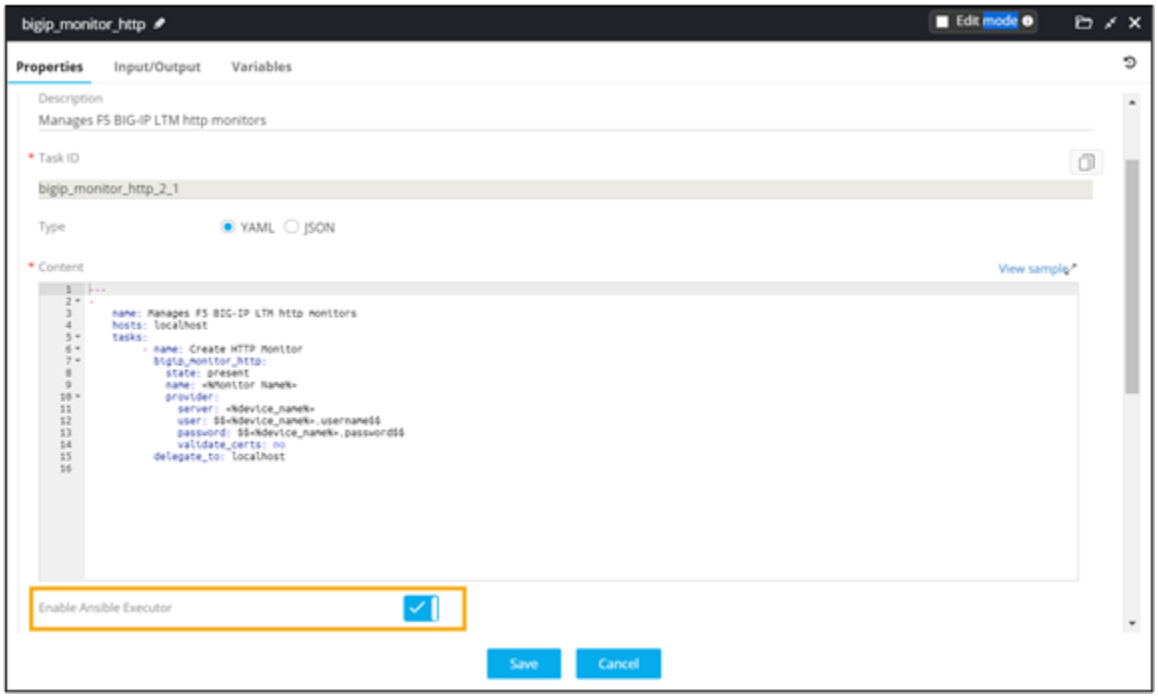


- **Import playbooks**: Discover playbooks from Ansible folder

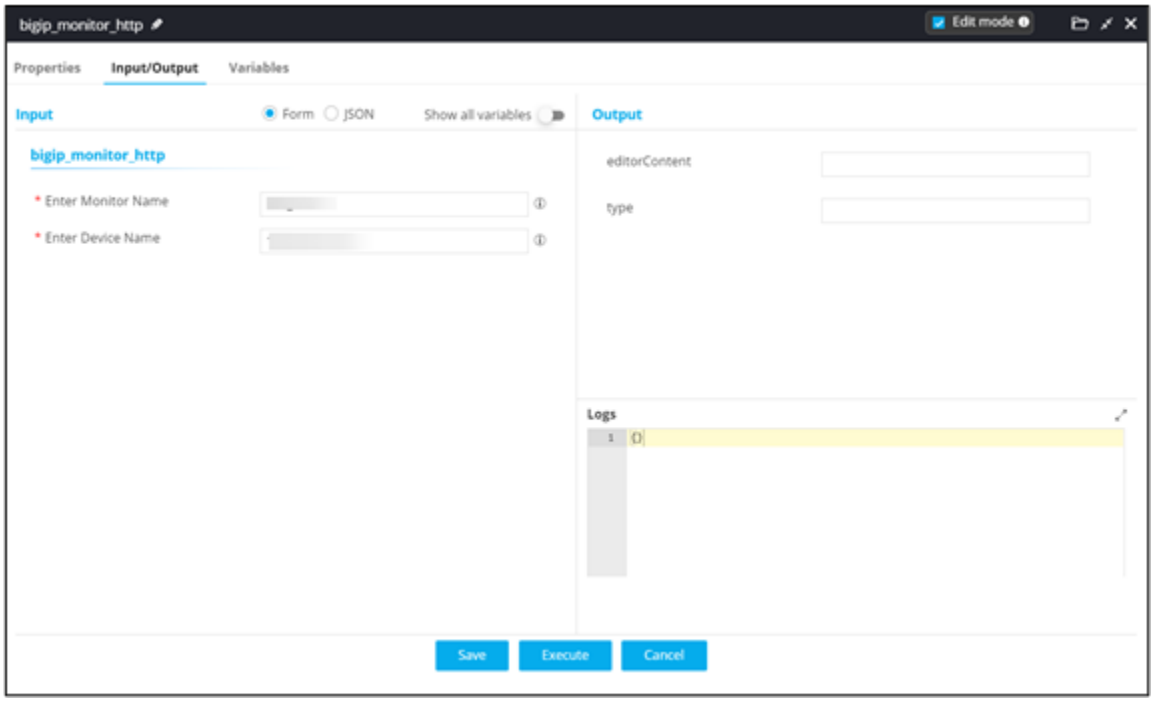


5. Save playbooks to a custom folder.
6. Drag and drop the playbooks into the workspace.
7. To enable Ansible instance, turn on the toggle.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

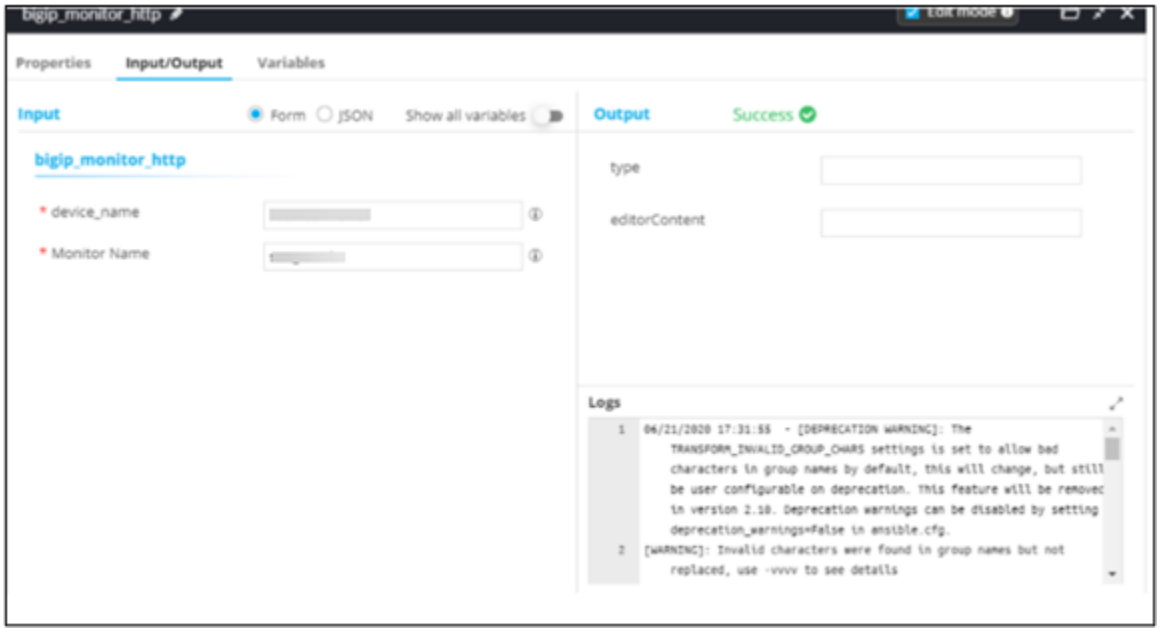


8. Provide **Input/Output** and **Variables** details.

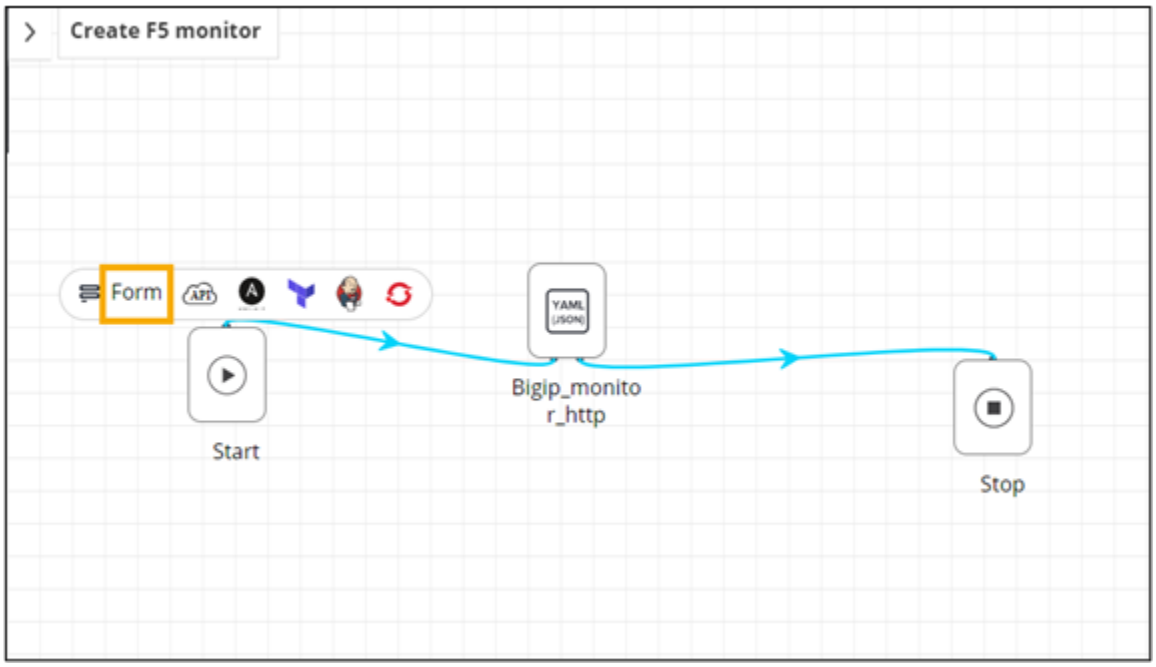


9. Execute the task to view the execution log.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

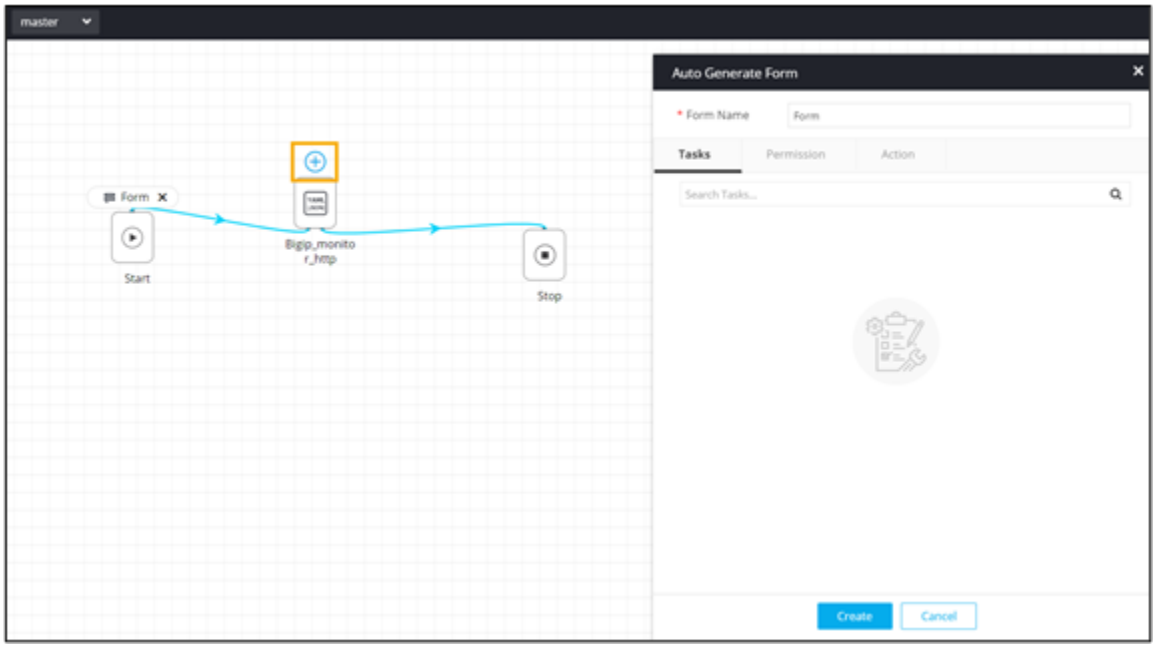


10. To auto generate a self-service form, click **Form** above the **Start** task.

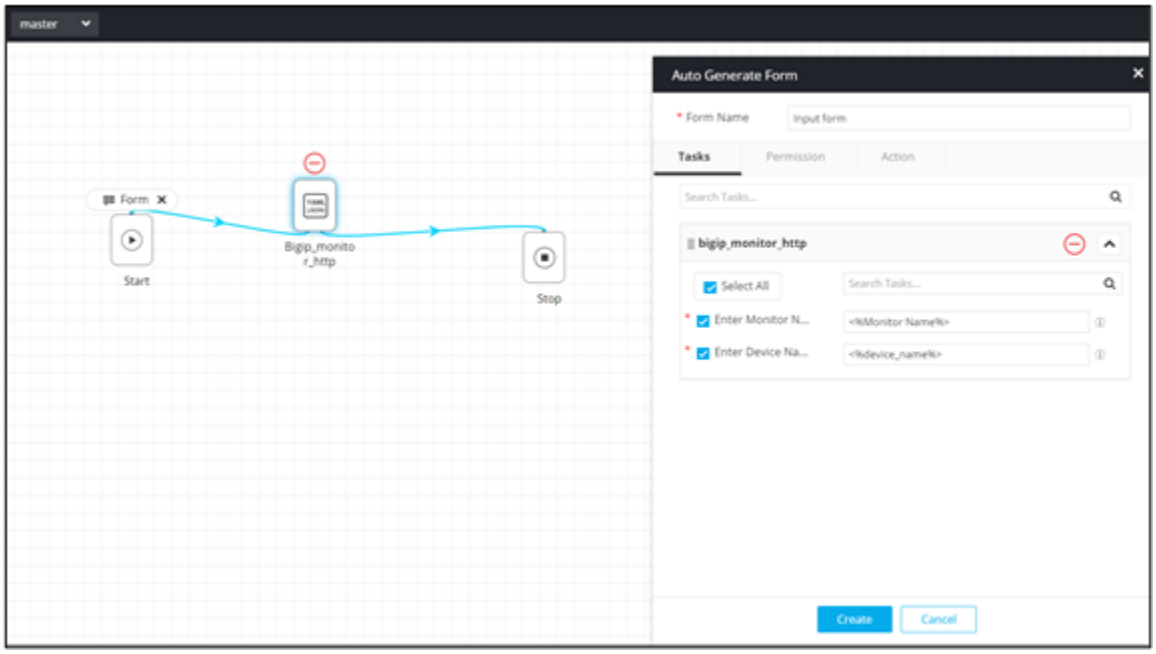


11. To auto-populate form fields, click ⊕ above the workflow task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



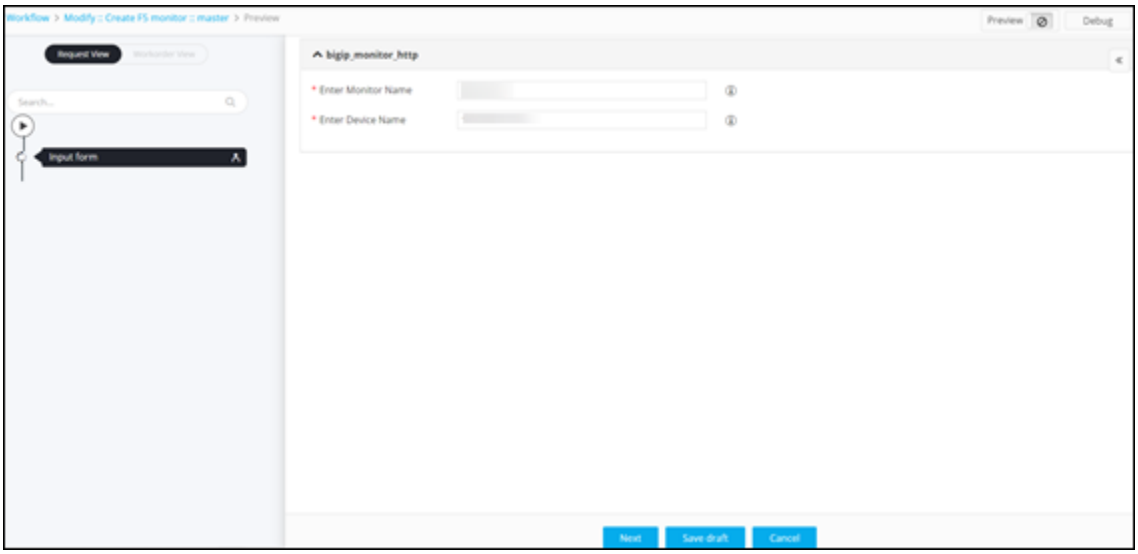
12. Define the variables to be used in the self-service form.

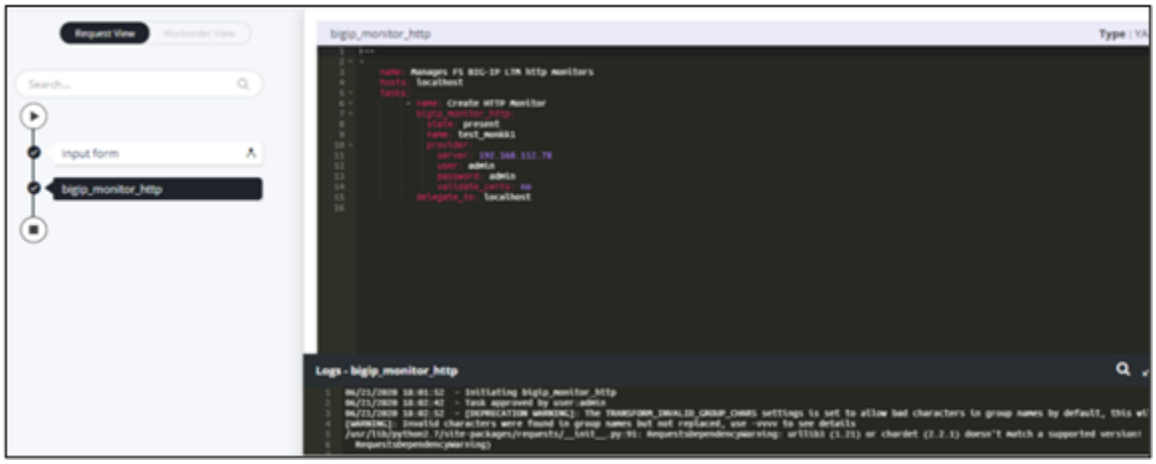


13. Click **Create**.
14. Connect the tasks and trigger the workflow from the Request :: View/Run page.
15. To validate the workflow, click **Preview**.

    Workflow is executed.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.





16. Enable the workflow.
17. To view the Request activity log, on the Request :: View/Run page, under **Activity**, click **View**.

    A **Summary** of the workflow request execution is displayed.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Using Variables within a YAML Task

Variables from any workflow task can be referred within the YAML task as part of the automation process.

- Use the following variable syntax in the YAML task to refer value from a previous task:

`<%variable_name%>`

Example: Following is an illustration for referring the ‘fqdn’ and ‘FreeIP’ values from an Infoblox task into a YAML task:

```
---
-
 name: Creating a virtual server in a F5 device
 hosts: local
 tasks:
      - name: Add virtual server
        bigip_virtual_server:
          server: <%device%>
          user: $$<%device%>.username$$
          password: $$<%device%>.password$$
          state: present
          partition: Common
          name: <%fqdn%>
          destination: <%free_ip%>
          port: 480
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

```
    snat: Automap
    description: Test Virtual Server
    validate_certs: no
    all_profiles:
      - http
      - tcp
  delegate_to: localhost
```

Getting Device Credentials Dynamically

Device credentials can be dynamically retrieved as part of the workflow execution process through a YAML (JSON) task. The following syntax must be used in the task in order to get device credentials.

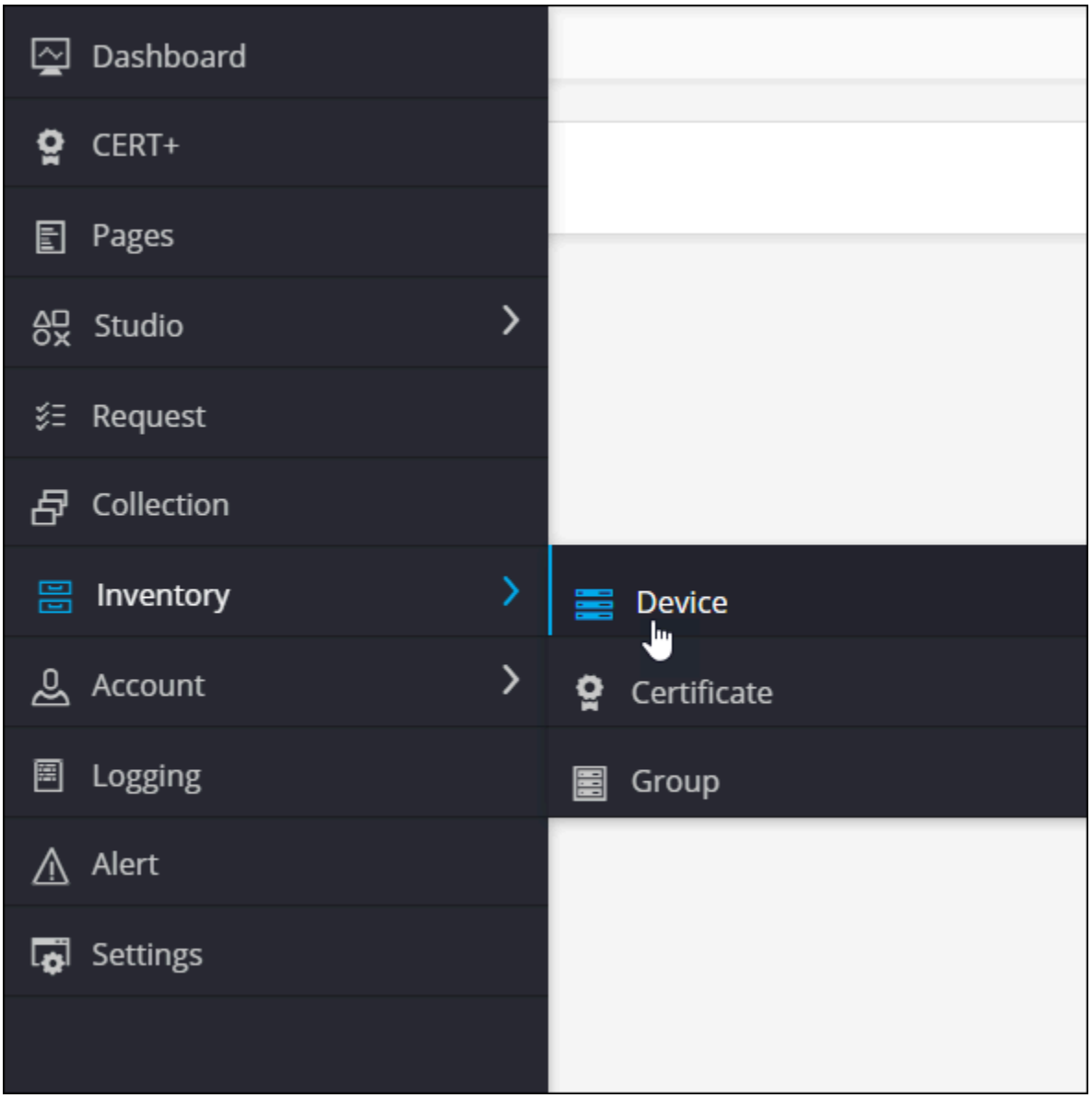
- Get device detail: `$$devicename.username$$`
- Get device credential: `$$devicename.password$$`

Note: Device(s) must be added in the Appviewx inventory

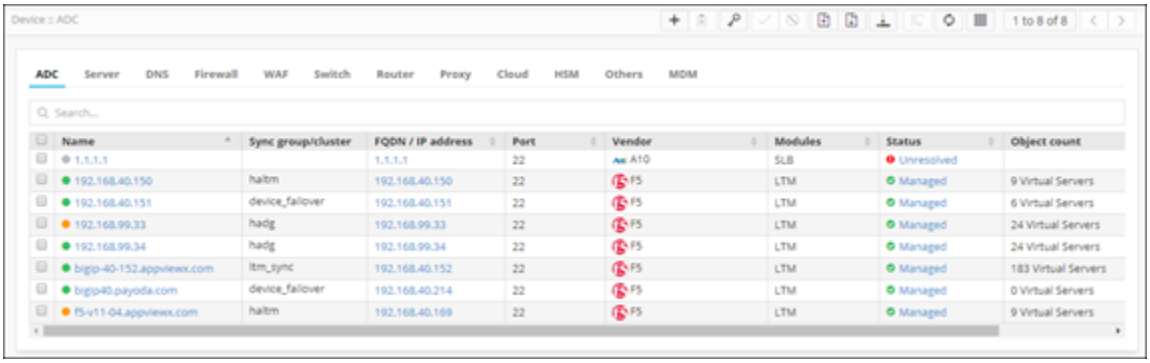
To get the device credentials dynamically during the workflow execution process:

1. From the top left corner of the screen, click ☰.
2. Select **Inventory** > **Device**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



3. On the Device Inventory page, ensure there is at least one device added with relevant credentials.

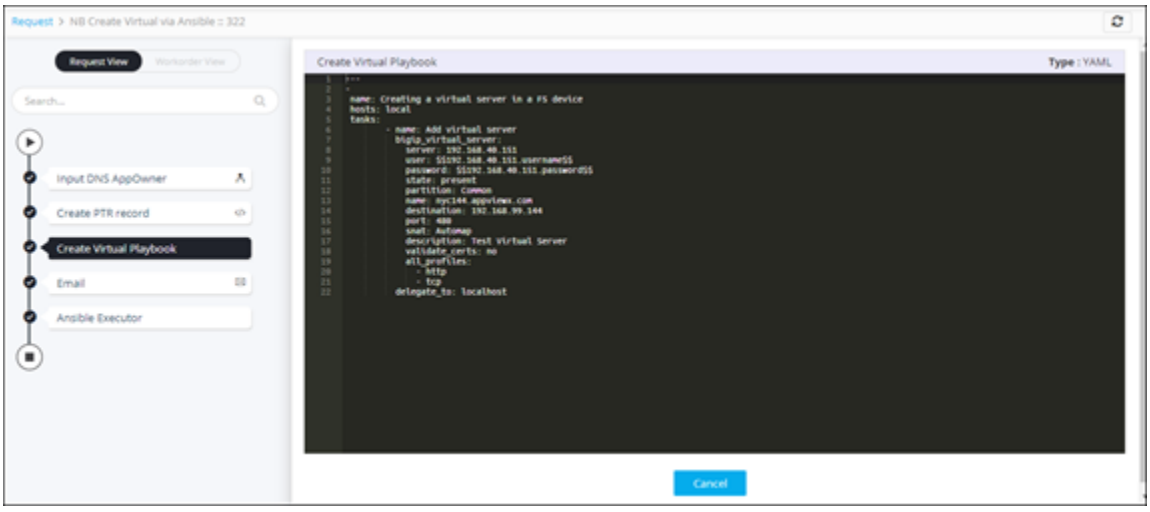


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

4. Design a new workflow.
5. Drag and drop relevant tasks.
6. From the **User Interface** section, drag and drop the **YAML** task.
7. Add input data in the YAML format.
8. Enter the following syntax to get the device credentials dynamically.

Syntax	Usage Sample
--- - name: Creating a virtual server in a F5 device hosts: local tasks: - name: Add virtual server bigip_virtual_server: device: <Devicename or IP address> username: $$devicename.username$$ password: $$devicename.password$$ state: present	--- - name: Creating a virtual server in a F5 device hosts: local tasks: - name: Add virtual server bigip_virtual_server: device: bigip.ltm.12.1 username: $$ bigip.ltm.12.1.username$$ password: $$ bigip.ltm.12.1.password$$ state: present

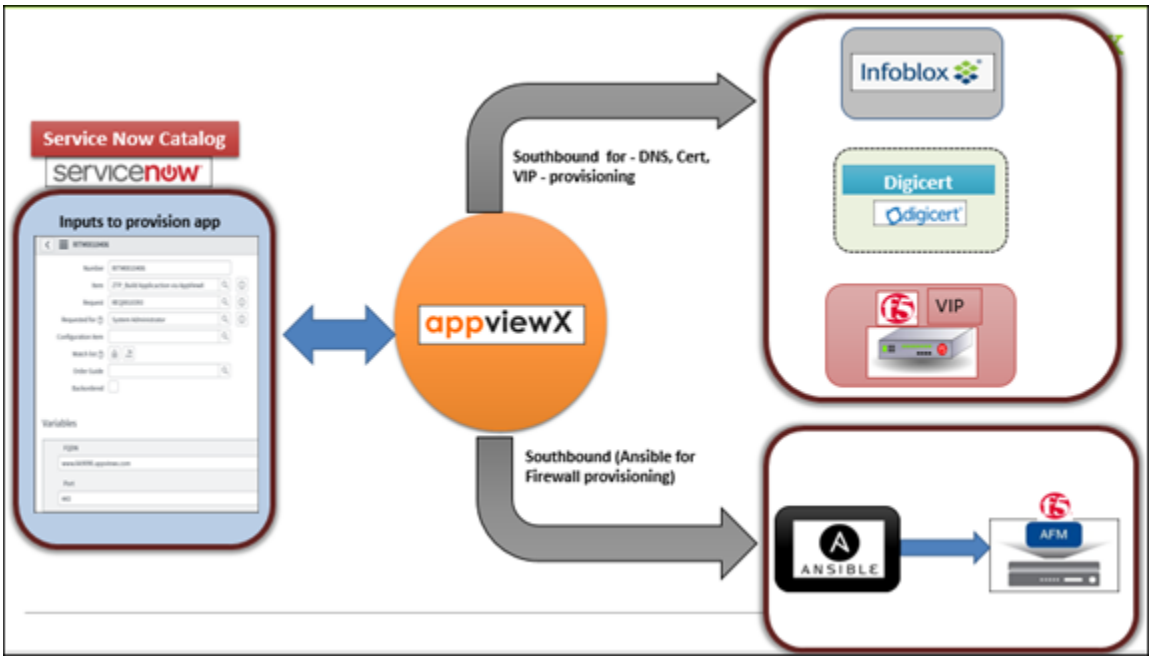
9. Save and enable the workflow.
10. Trigger the workflow from the Request :: View/Run page.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Ansible Executor

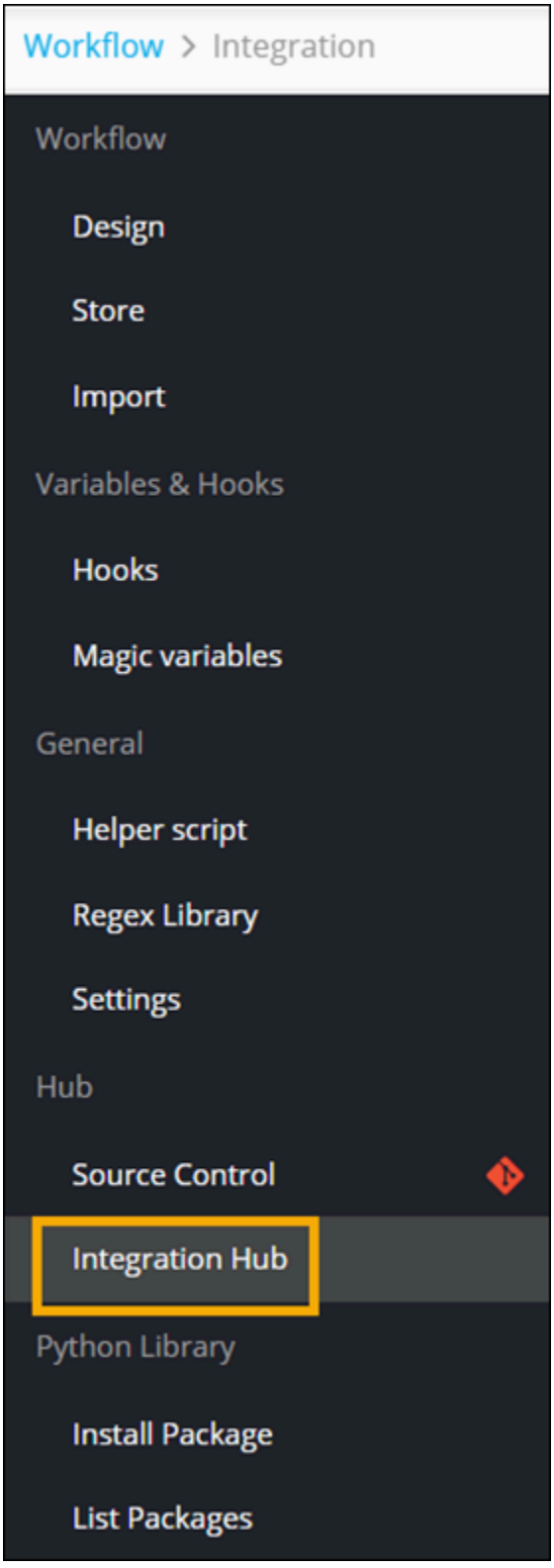
The following is a sample illustration to integrate Visual workflow and Ansible with Ansible as a southbound. This will allow discovery of any existing playbooks and further automate and orchestrate with Visual Workflow.



To connect to an Ansible instance, discover and reuse a BIG-IP playbook and integrate with Infoblox IPAM via Visual Workflow:

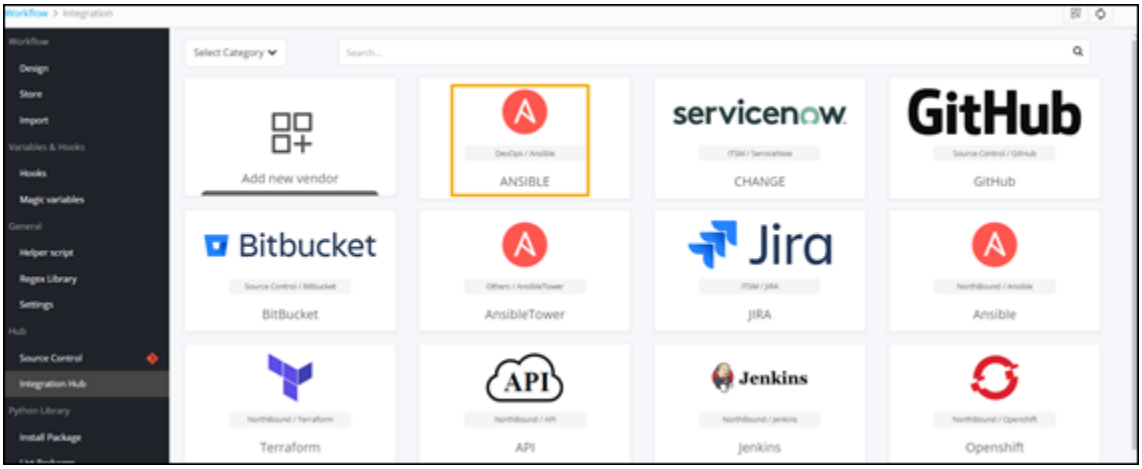
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. On the Workflow Inventory page, from the navigation pane on the left, click **Integration Hub**.

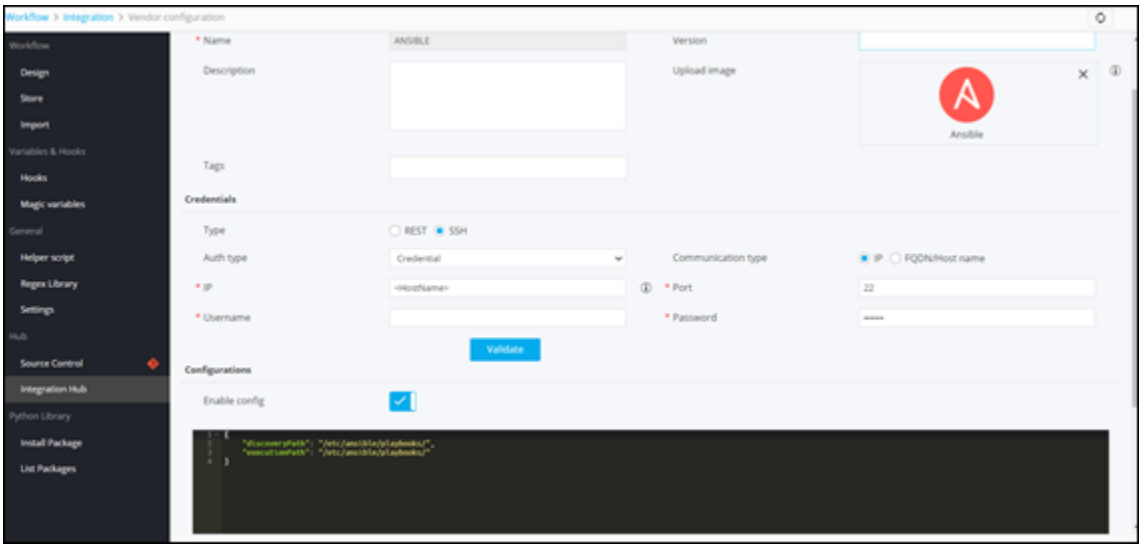


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

2. To configure an Ansible instance, click **Ansible**.

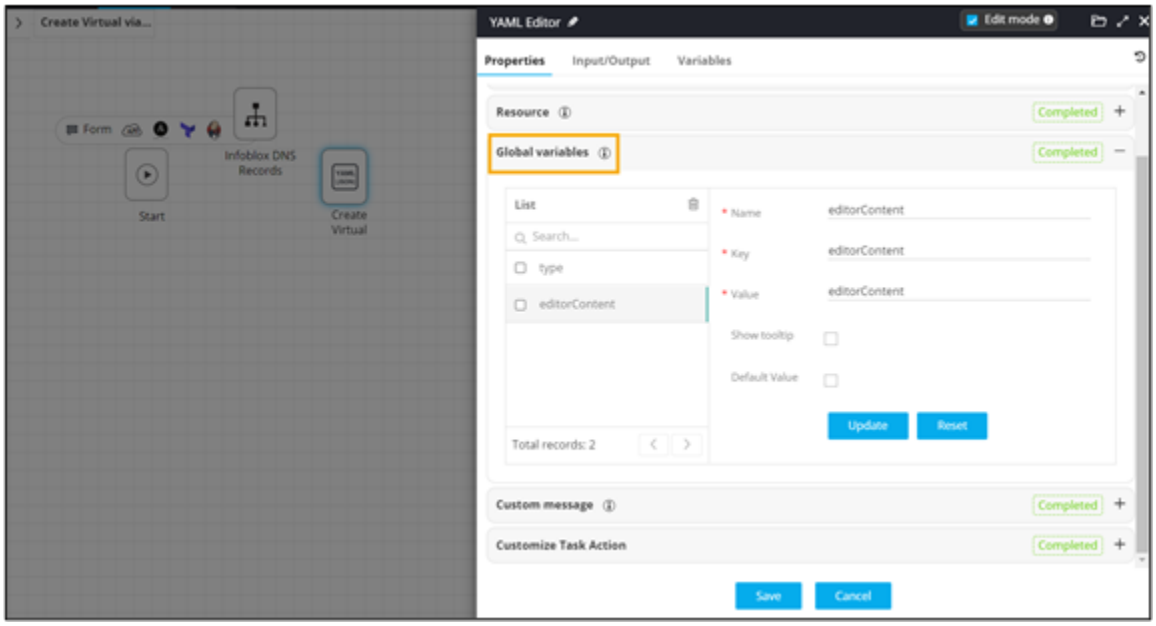


3. Enter the field information under the **General** section.

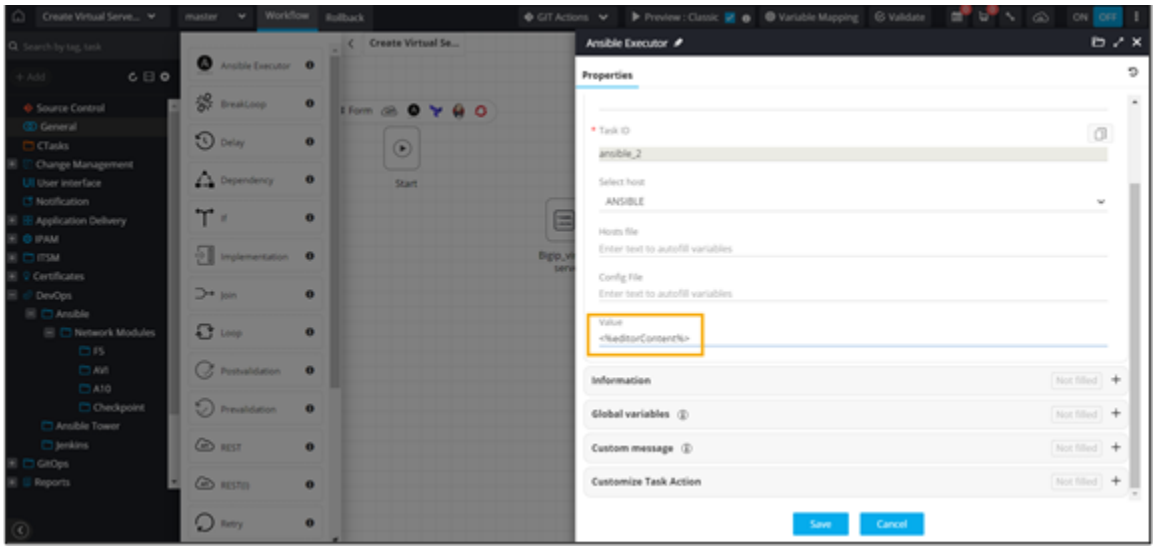


4. Define the folder/path from where existing playbooks can be discovered within the workflow studio.
5. Click **Save**.
6. Design a new workflow.
7. From the **DevOps** folder, select **Ansible** > **Network Modules** > **F5**.
8. From the **F5** folder, drag and drop the **BigIP_Virtual_Server** task into the workspace.
9. In the **YAML Editor** task window, under **Properties**, in the **Global variables** section, declare the global variable in this task to pass data from the YAML task to the Ansible Executor on the southbound.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



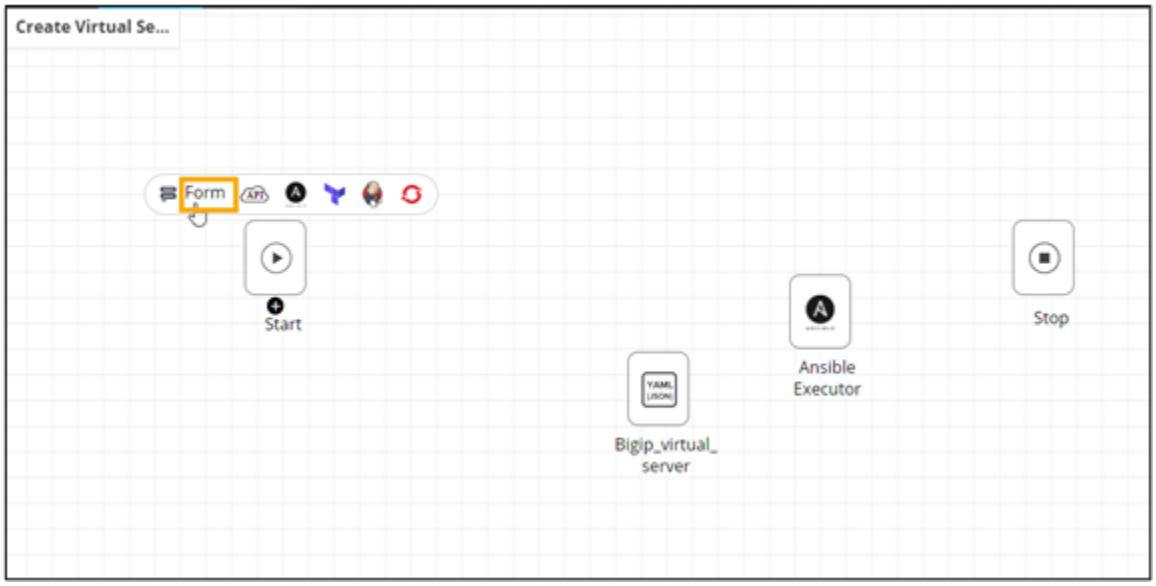
10. From the **General** section, drag and drop the **Ansible Executor** task.
11. In the **Ansible Executor** task window, under **Properties**, select the Ansible host on which implementation is to be done.
12. Map the global variable from the YAML task.



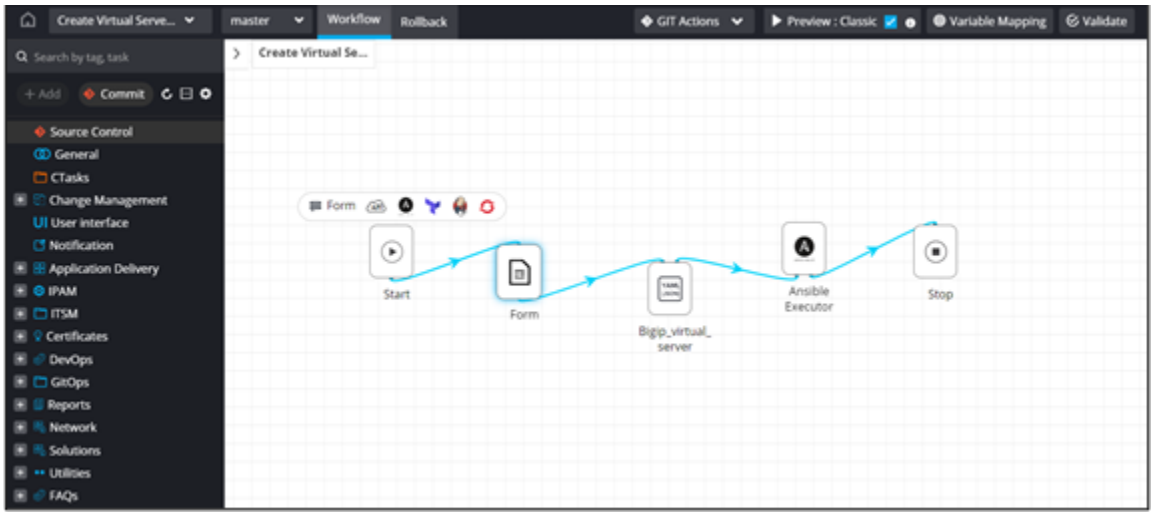
13. To auto-generate a self-service form, click **Form** above **the** Start task.

> **Note:** For more information on how to design self-service forms, refer to the section on Auto-generate forms.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

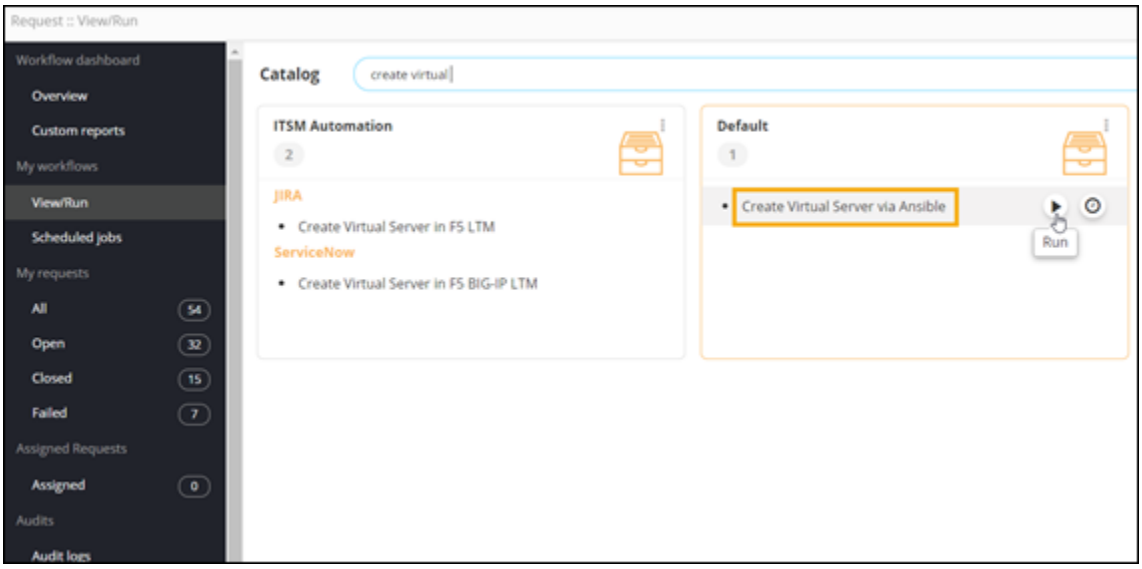


14. Connect the workflow tasks and enable the workflow.

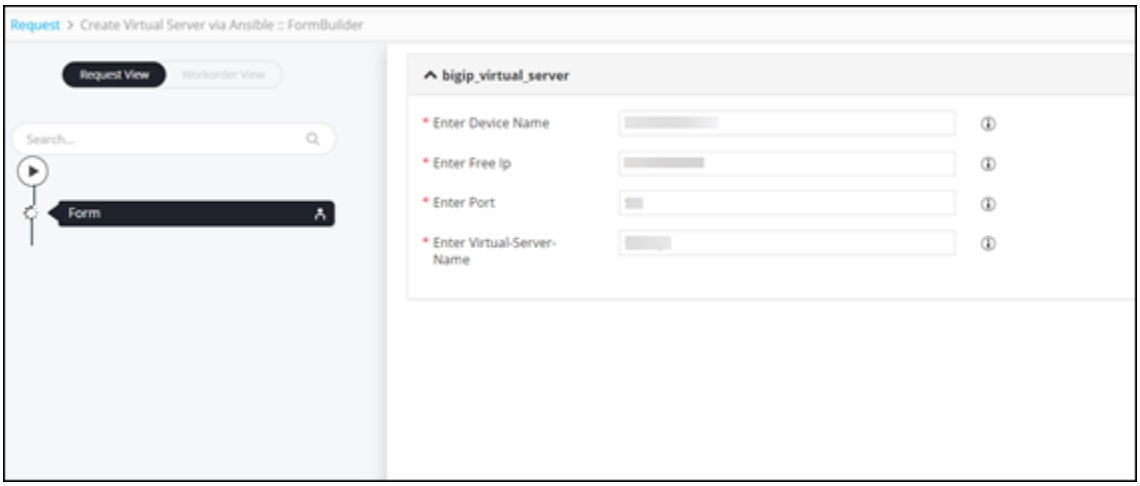


15. Trigger the workflow from the **Request :: View/Run** page.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



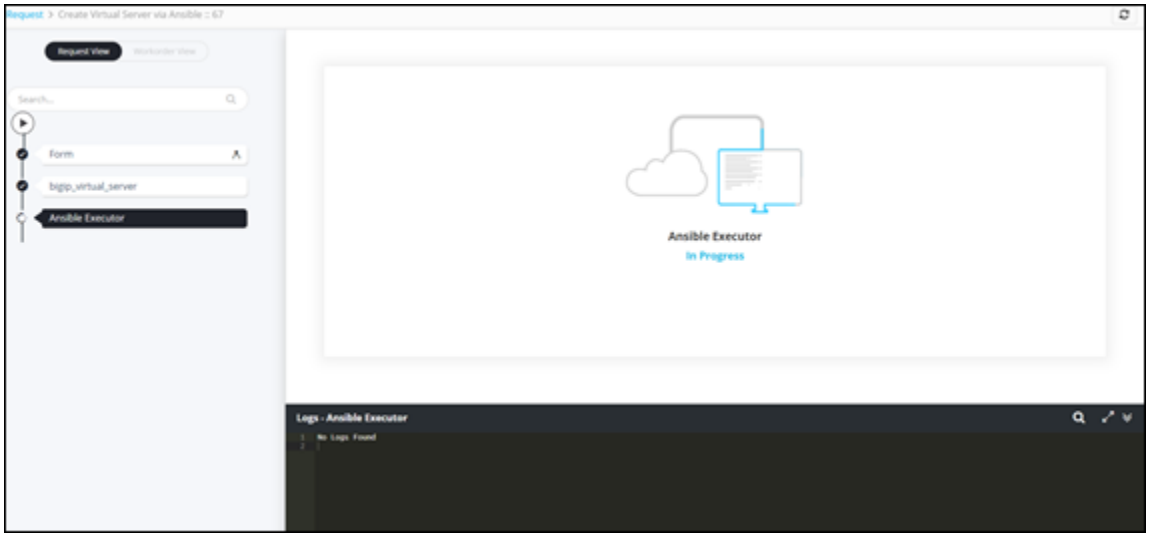
16. Enter the input details in the Form.



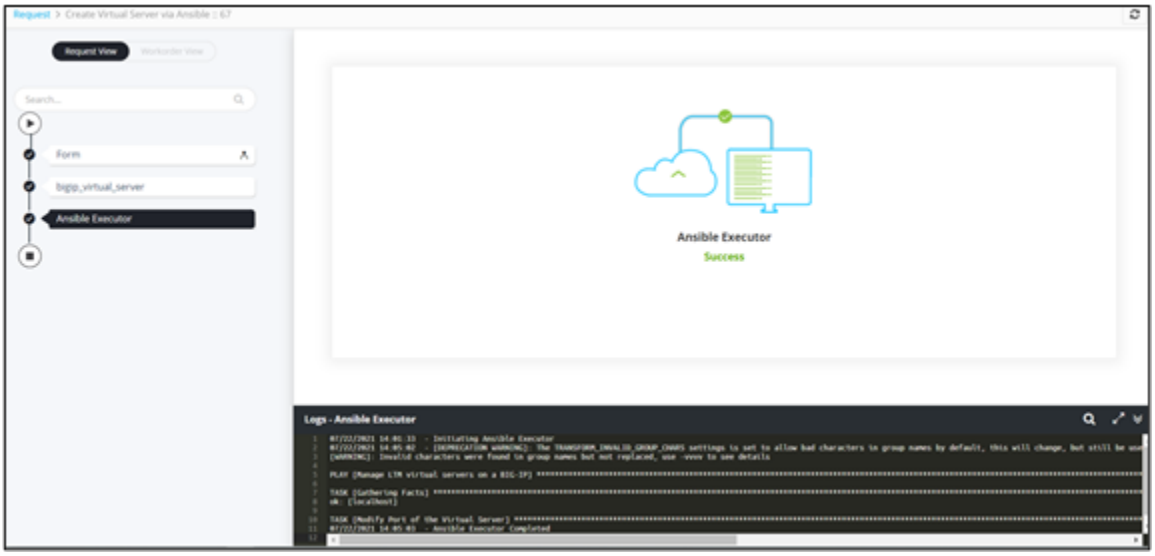
17. Click **Next**.
18. Submit the YAML task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Ansible Executor task in progress.



- Ansible Executor task completed.



Automation Collision

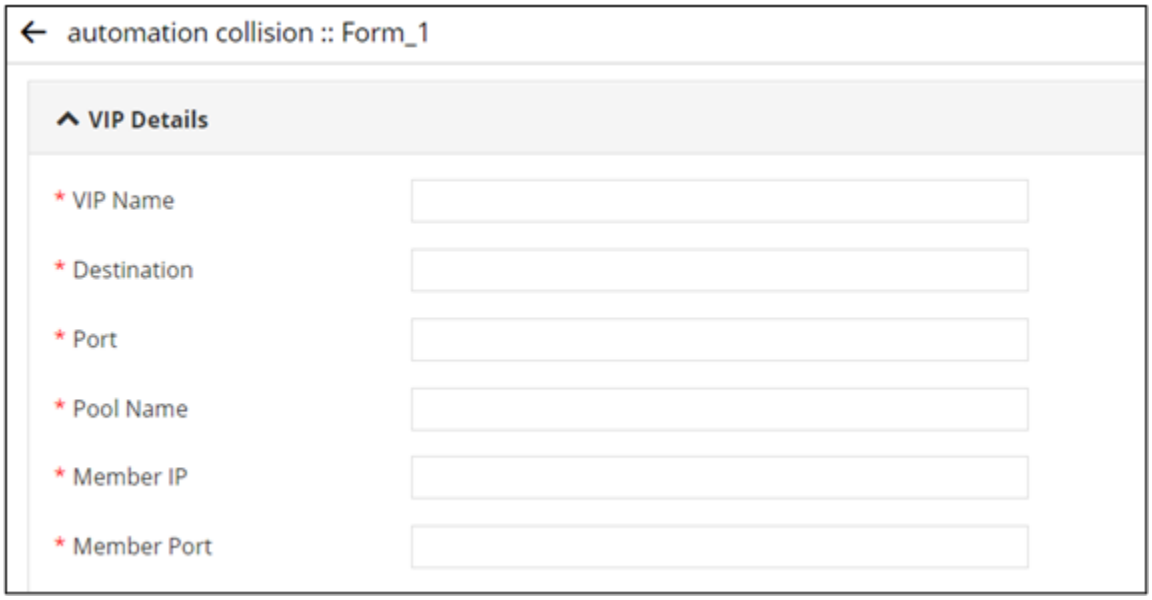
This feature allows you to check for duplicate configs across workflow requests.

- Check all the pending workflow requests for duplicate object names such as virtual server, pool member, and monitors.
- Check all the pending workflow requests for any variables used within a workflow.
- Approve or reject the workflow requests based on duplicate config validation.

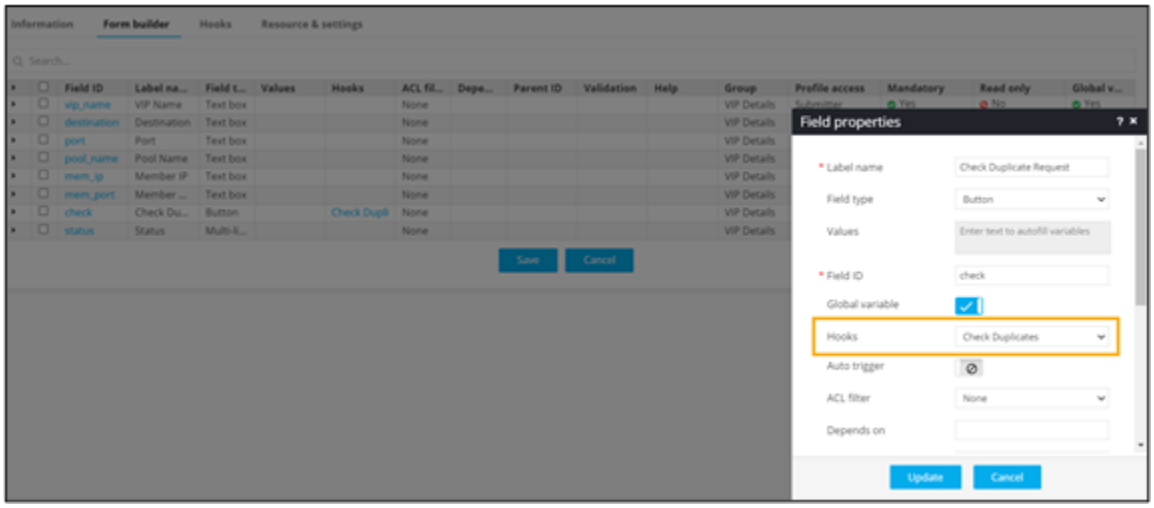
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

To check for duplicate configs when designing a workflow to create a virtual server:

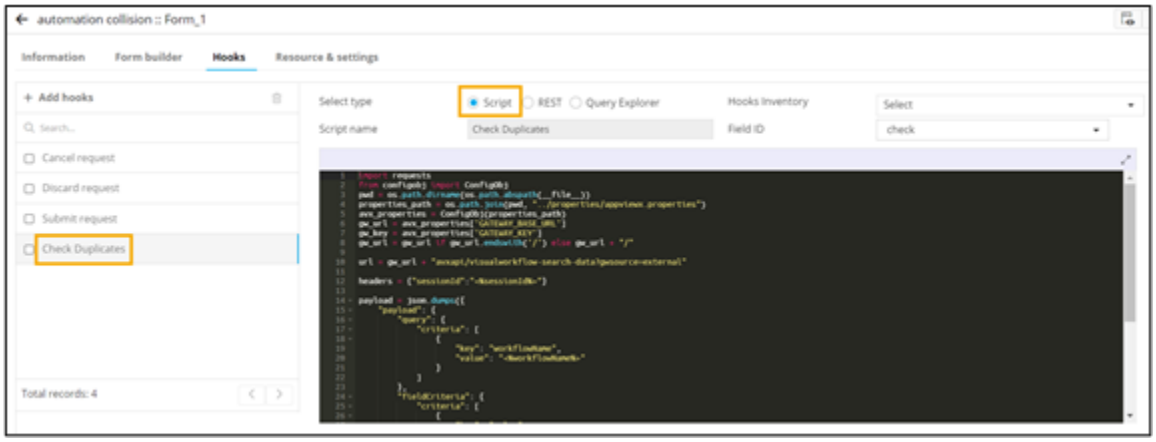
1. Design a workflow.
2. From the **User Interface** section, drag and drop a **Form** task.
3. Add the form fields for creating a virtual server.



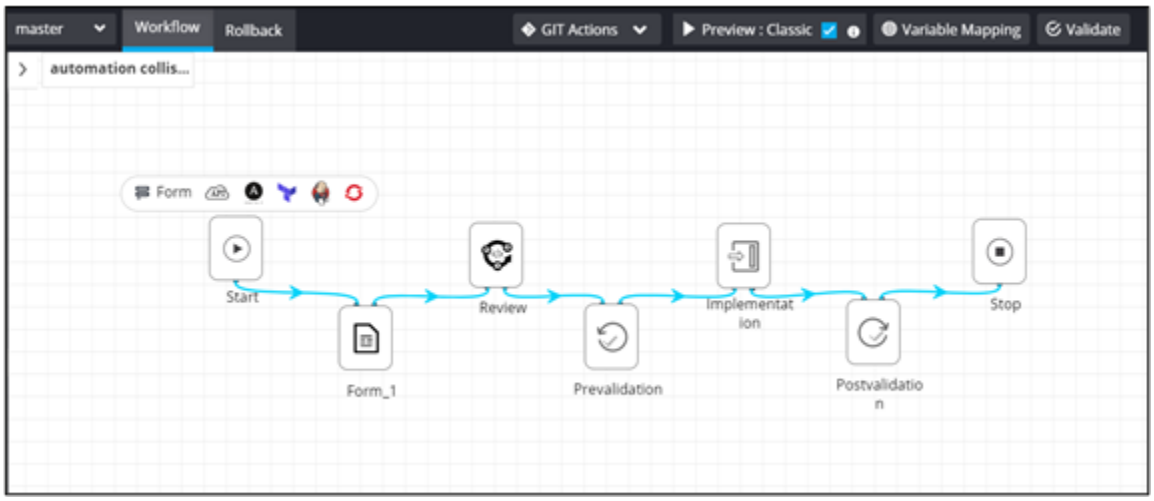
4. Add a button field to check for duplicate configs using a Script logic.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.



5. Connect all workflow tasks.



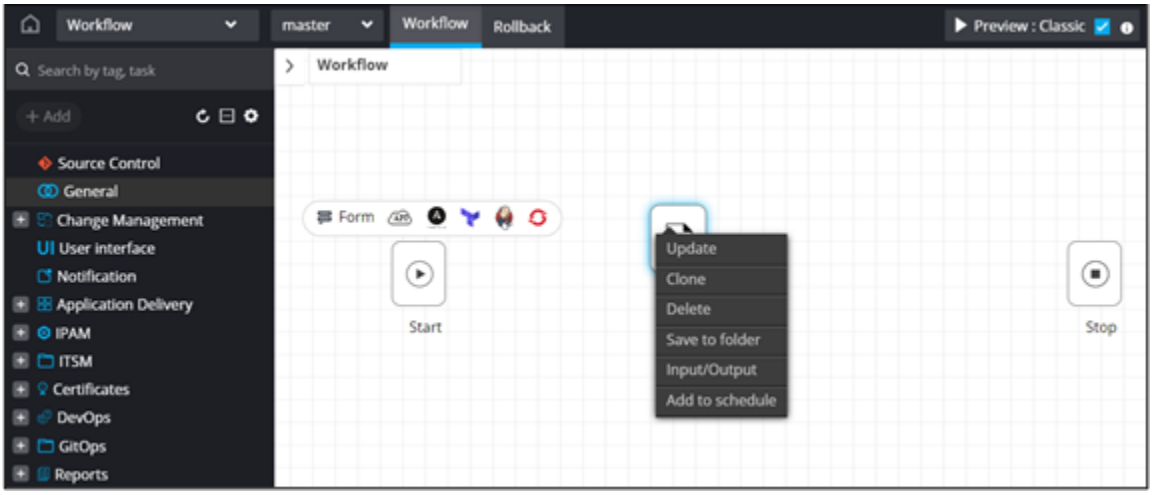
6. Trigger the workflow from the Request :: View/Run page.

Workflow Task Actions

You can right-click any workflow task to perform the following actions:

- **Update**: Allows you to update configuration settings within the task.
- **Clone**: Allows you to clone the selected task.
- **Delete**: Allows you to delete the selected task.
- **Save to folder**: Allows you to save the task to an existing or new folder.
- **Input/Output**: Allows quick and easy variable mapping.
- **Add to schedule**: Allows you to add the task as a scheduled job.

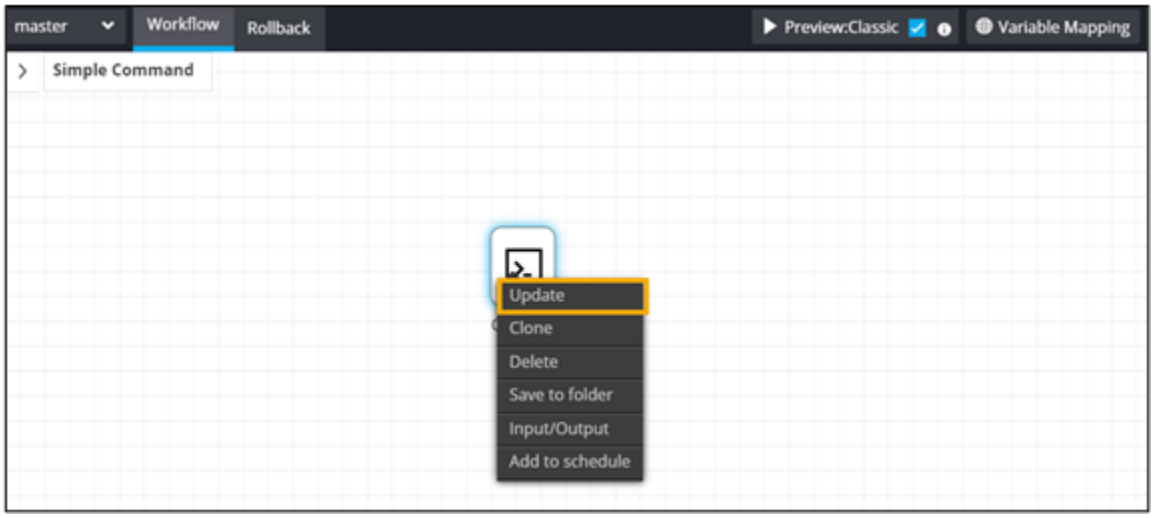
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



- Updating a Task
- Cloning a Task
- Deleting a Task
- Saving and Reusing a Task
- Input/Output Variable Mapping

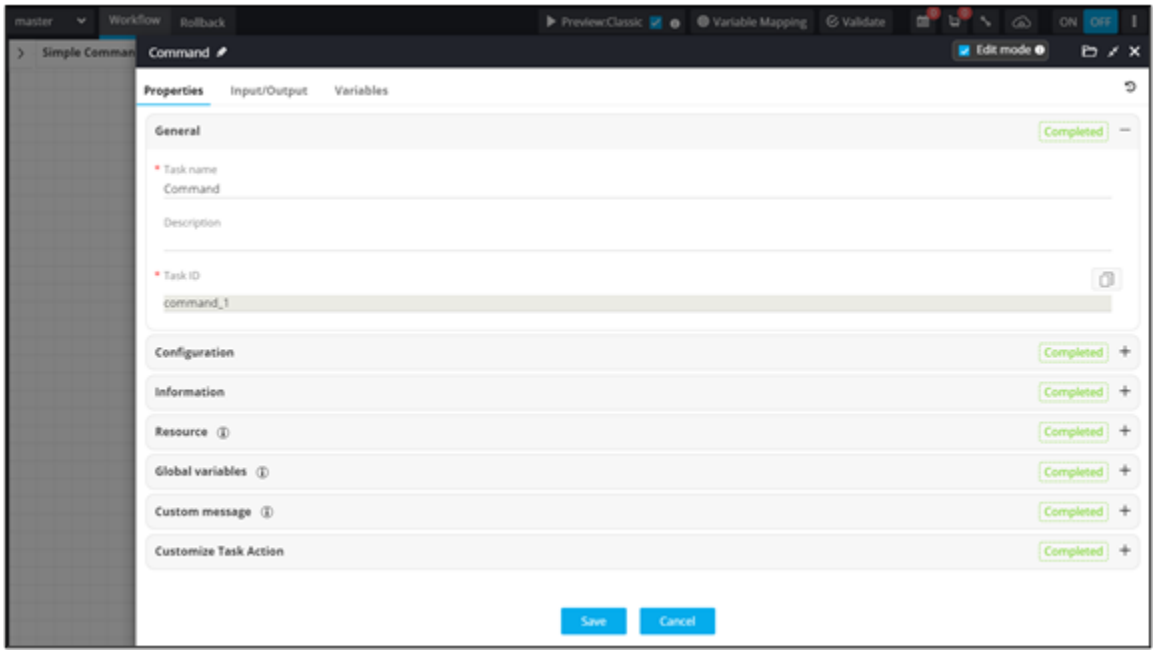
Updating a Task

1. Right-click the task to see options.
2. Click **Update**.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

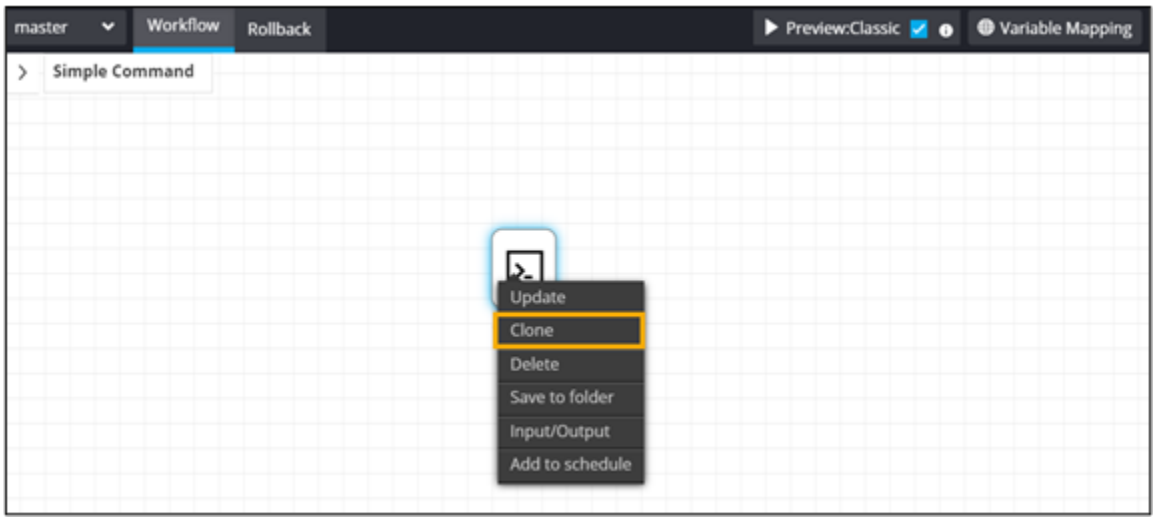
3. Edit the task properties to update the task.



Cloning a Task

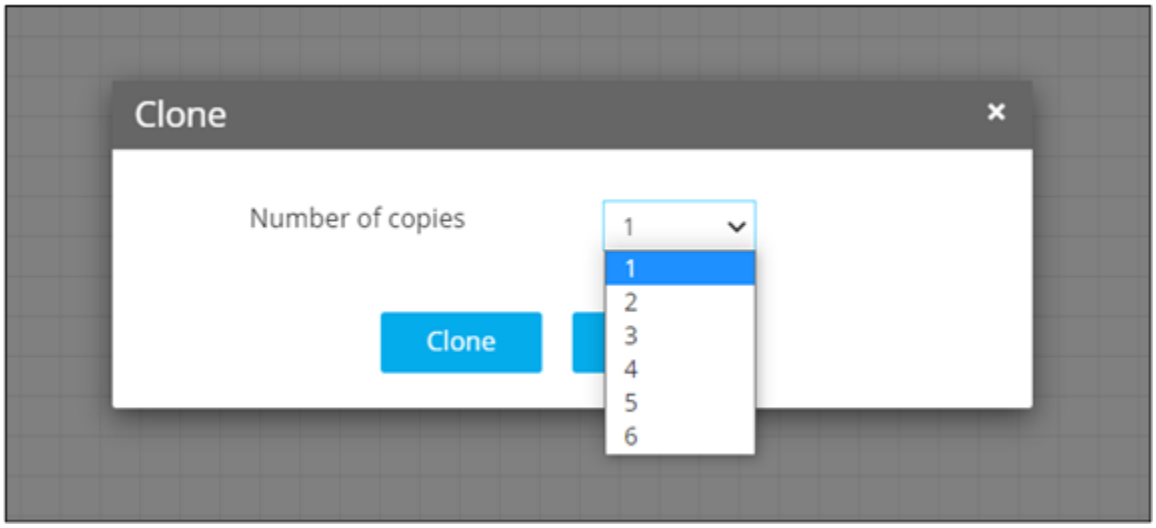
You can clone a task in the workspace instead of having to drag and drop the same task from the menu.

1. Right-click a task to see the options.
2. Click **Clone**.



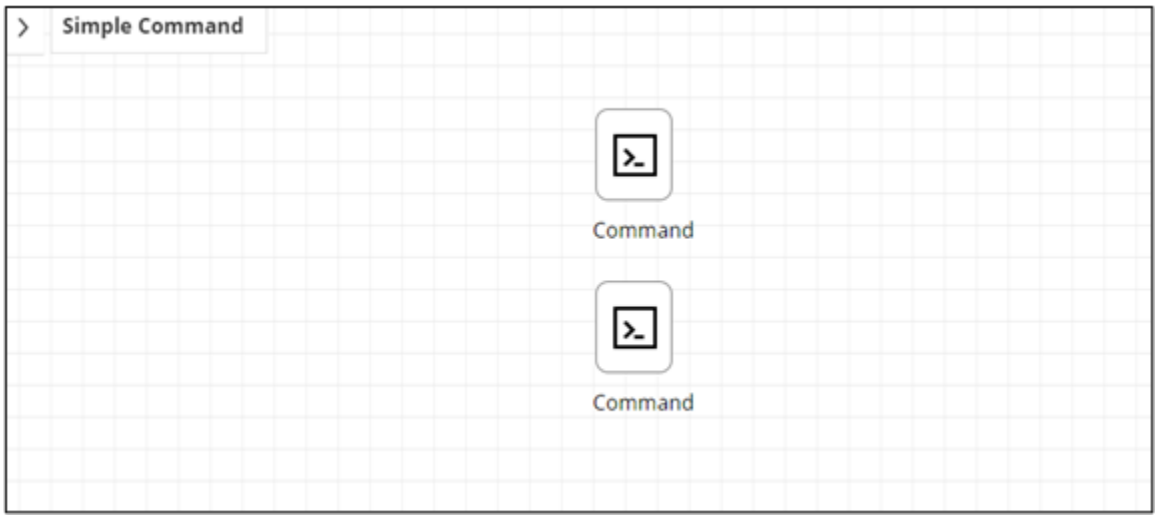
3. In the **Clone** window, select the number of copies from the dropdown list.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



4. Click **Clone**.

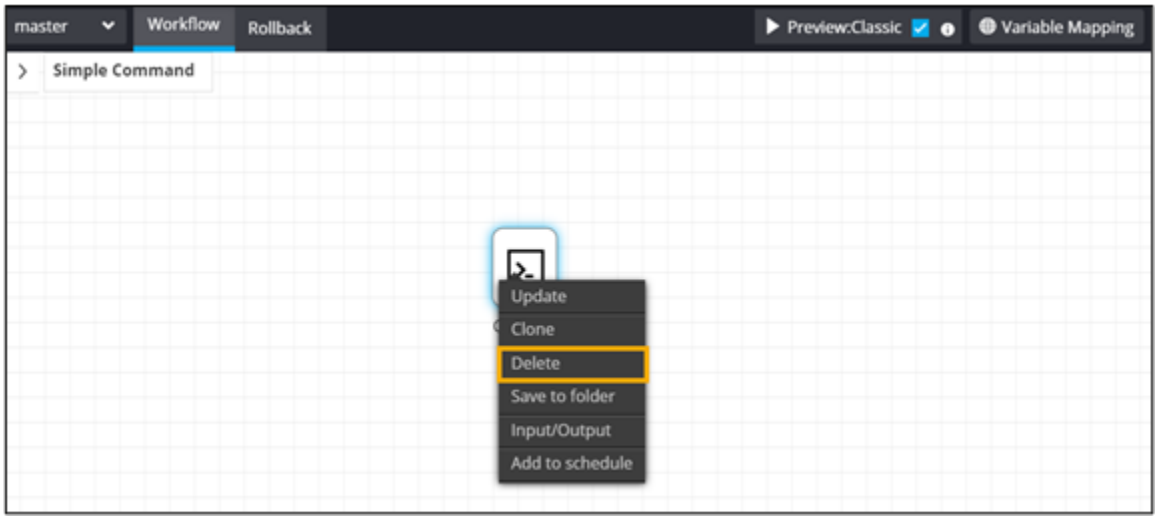
   A copy of the task is created in the workspace.



Deleting a Task

1. Right-click the task to view options.
2. Click **Delete**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

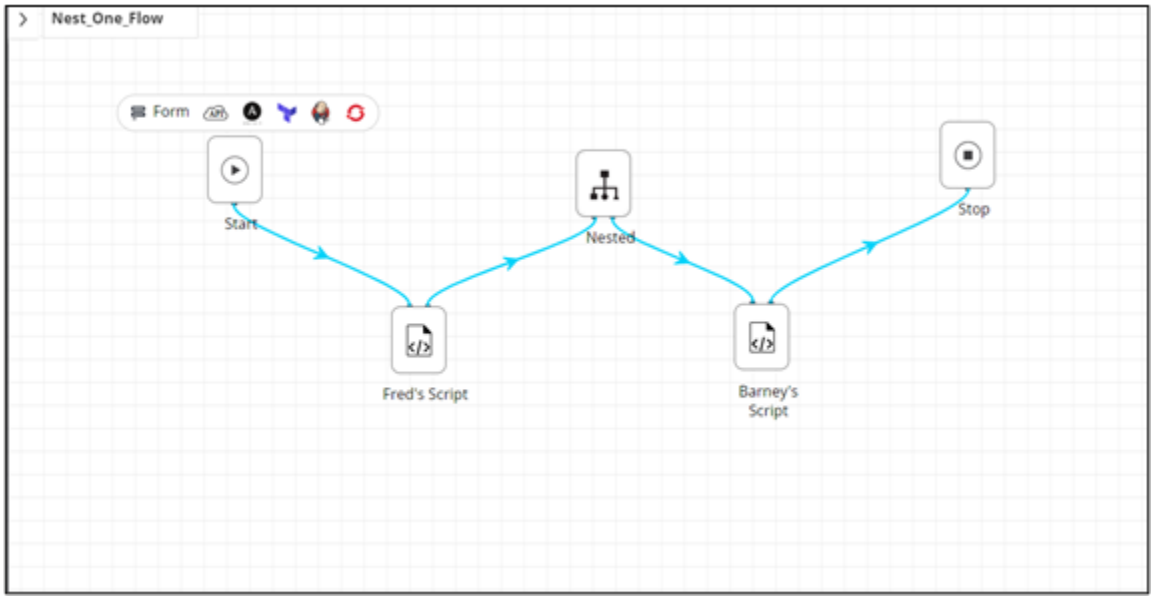


3. Click **Yes** in the **Confirmation** pop-up window.
   The task is deleted.

Saving and Reusing a Task

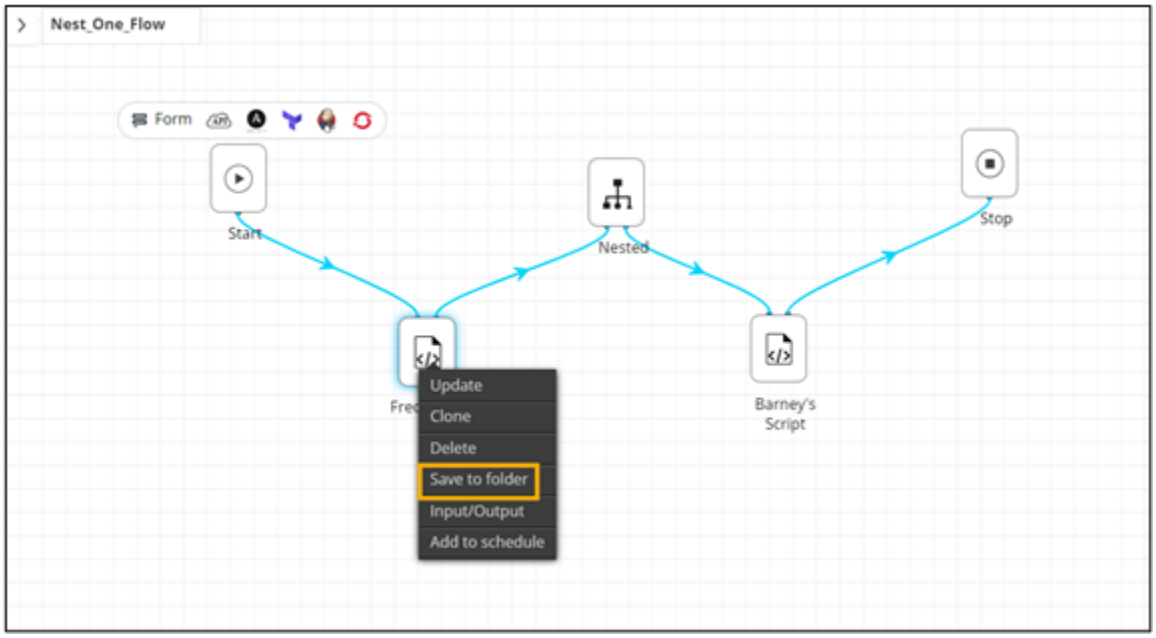
You can save specific tasks to a folder(s) in order to import/reuse them in other workflows.

1. Open an existing workflow in the Workflow Studio.

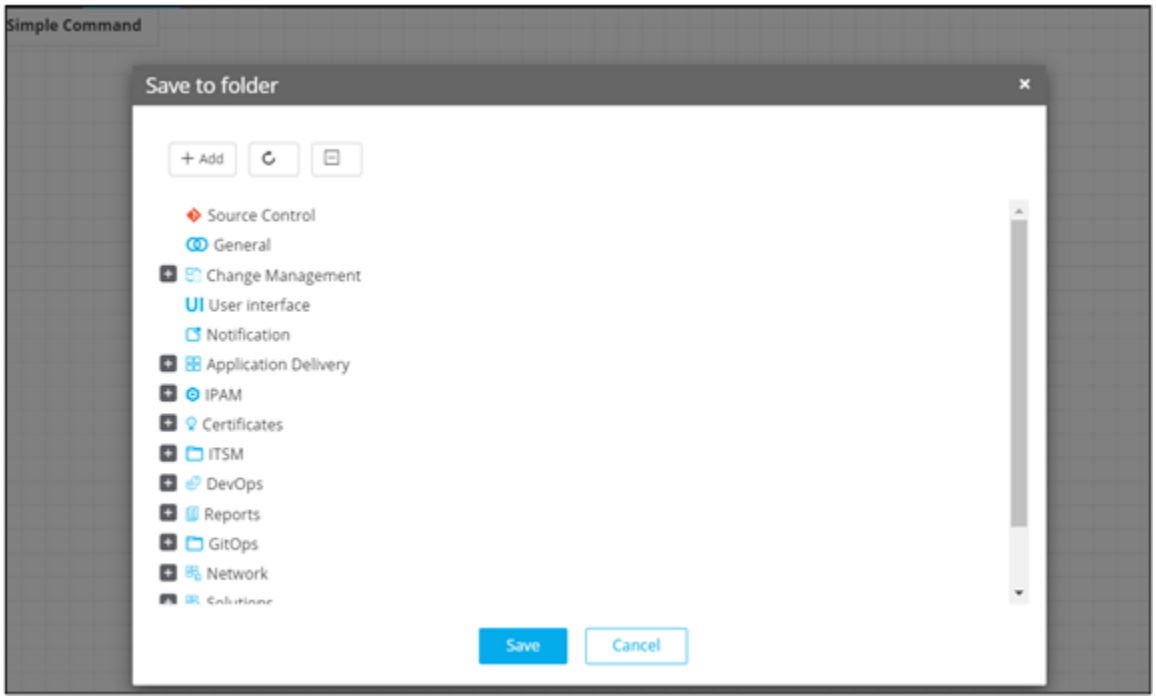


2. Right-click the task to be saved.
3. Select **Save to folder**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

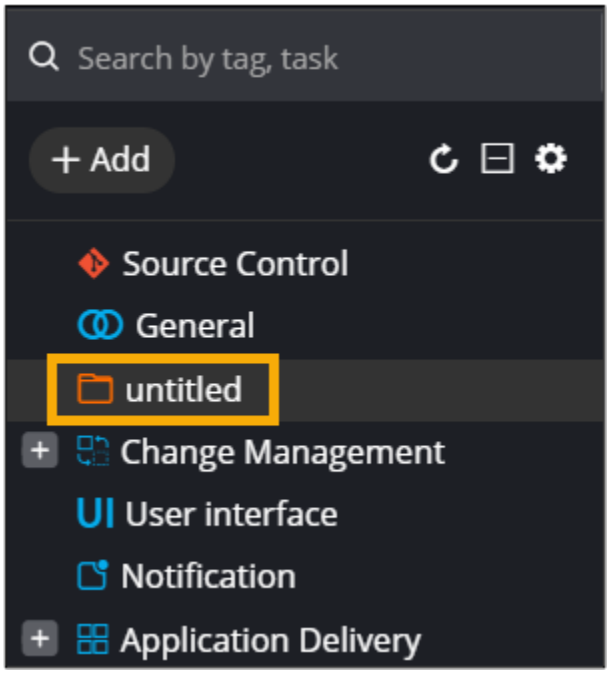


4. To save the task to an existing folder, in the **Save to folder** window, select a folder from the list.



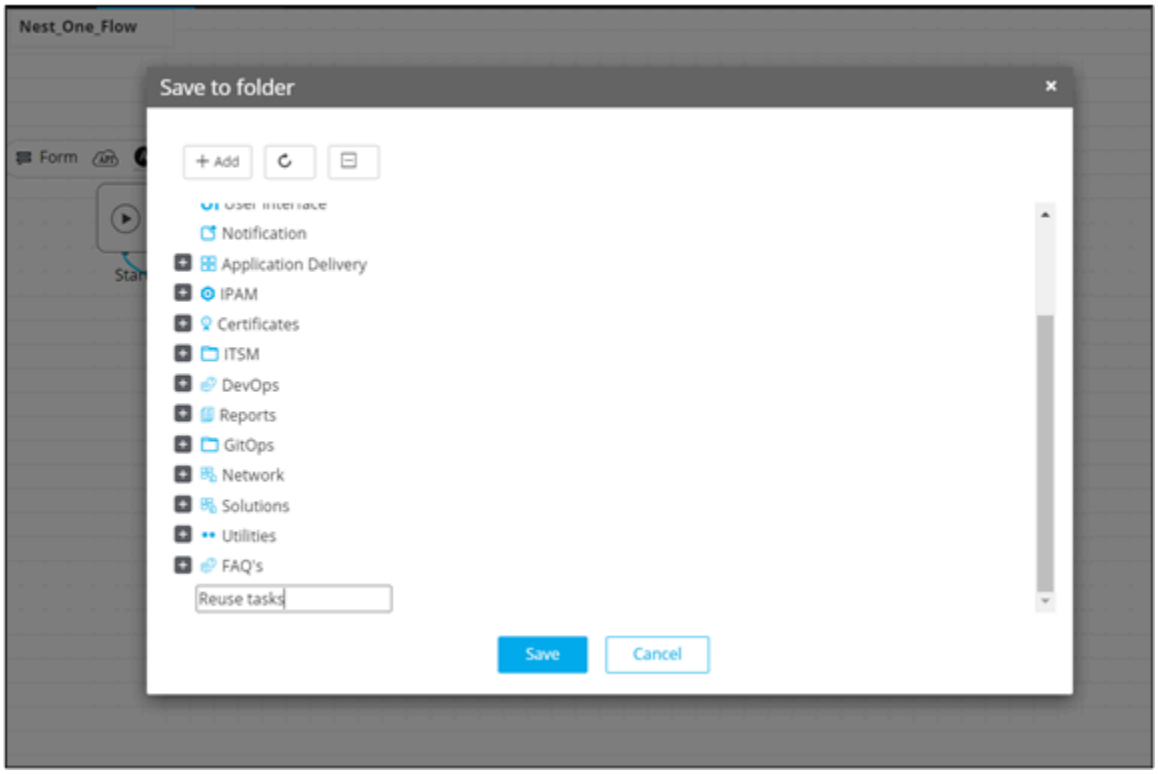
Note: When naming a folder, the same naming conventions must be followed as those when naming a workflow. If a folder is created without providing a name, it will show up as **untitled**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

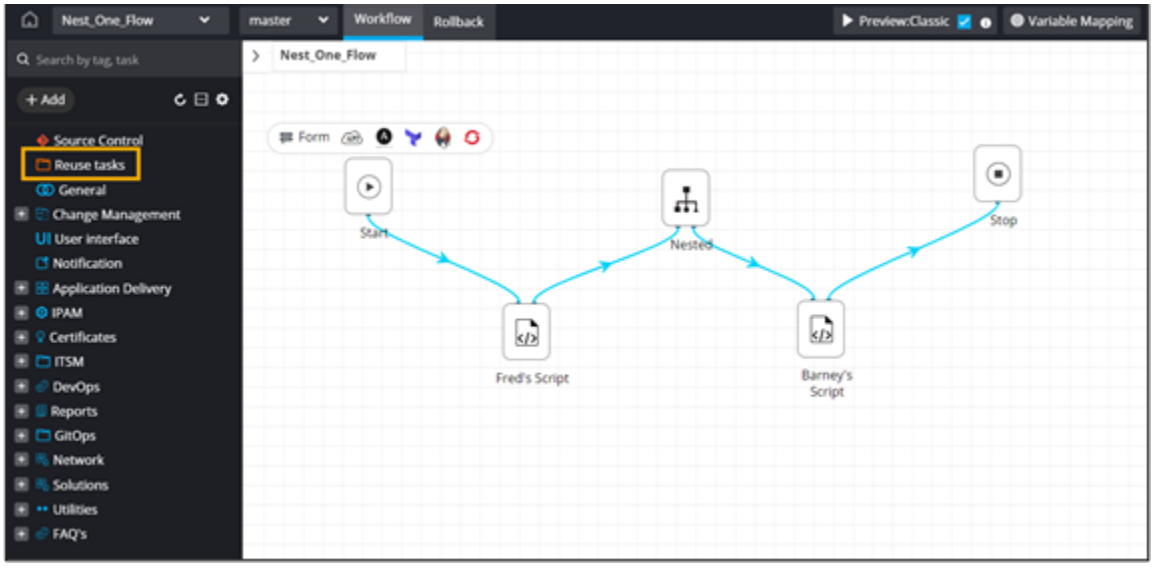


5. Click **Save**.
6. To save the task to a new folder, click **Add**.
7. Enter the folder name and click **Save**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

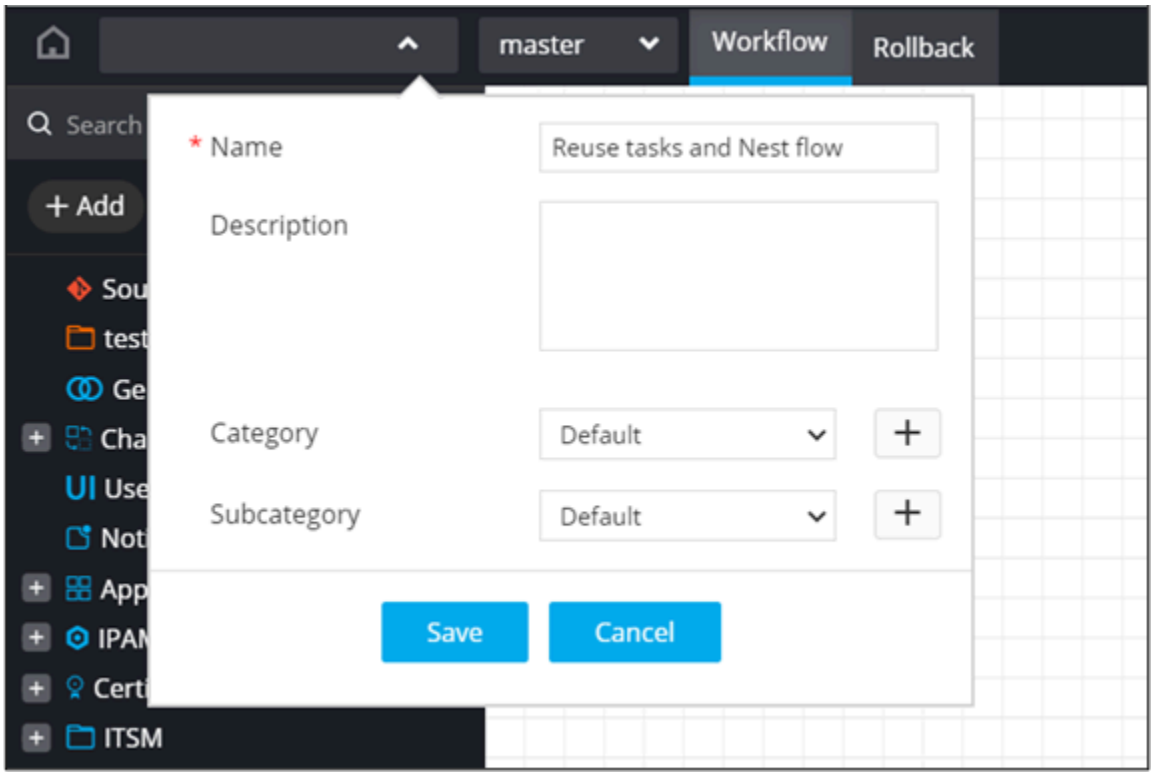


The new folder is added to the left menu.

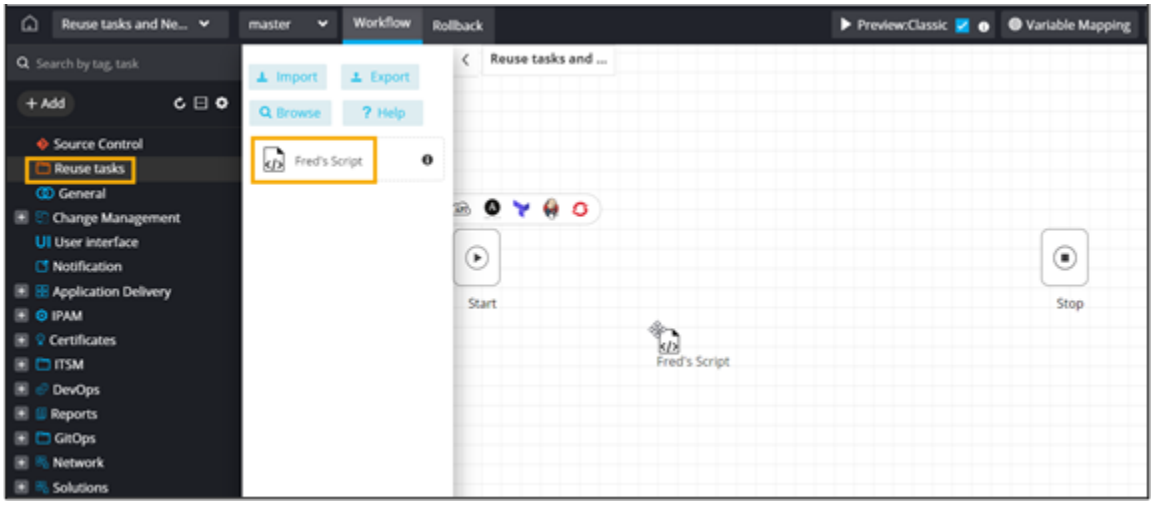


8. Exit the workflow.
9. Design a new workflow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

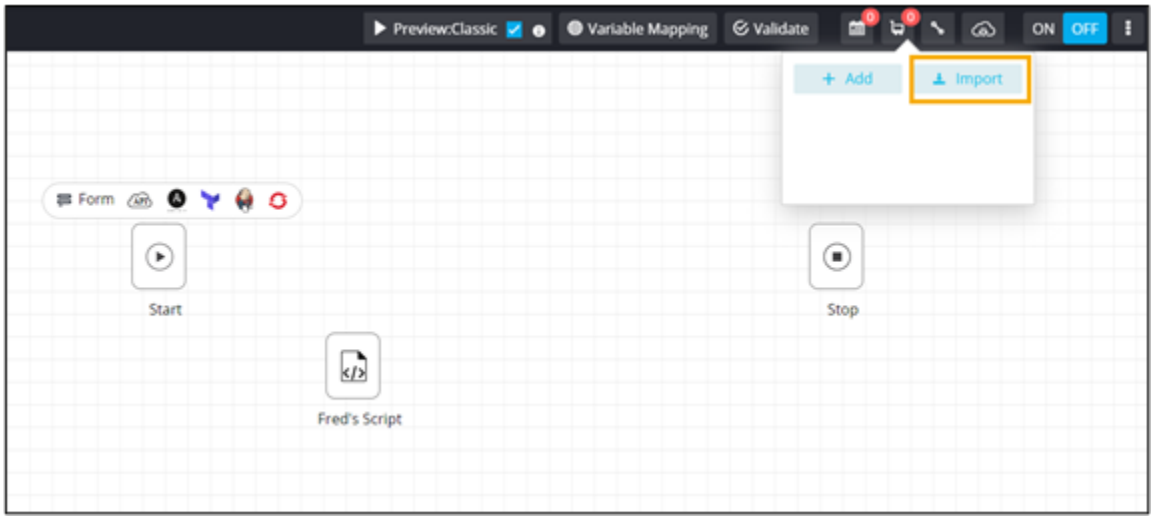


10. Drag and drop the task saved to the folder in the previous workflow and reuse in the current workflow.

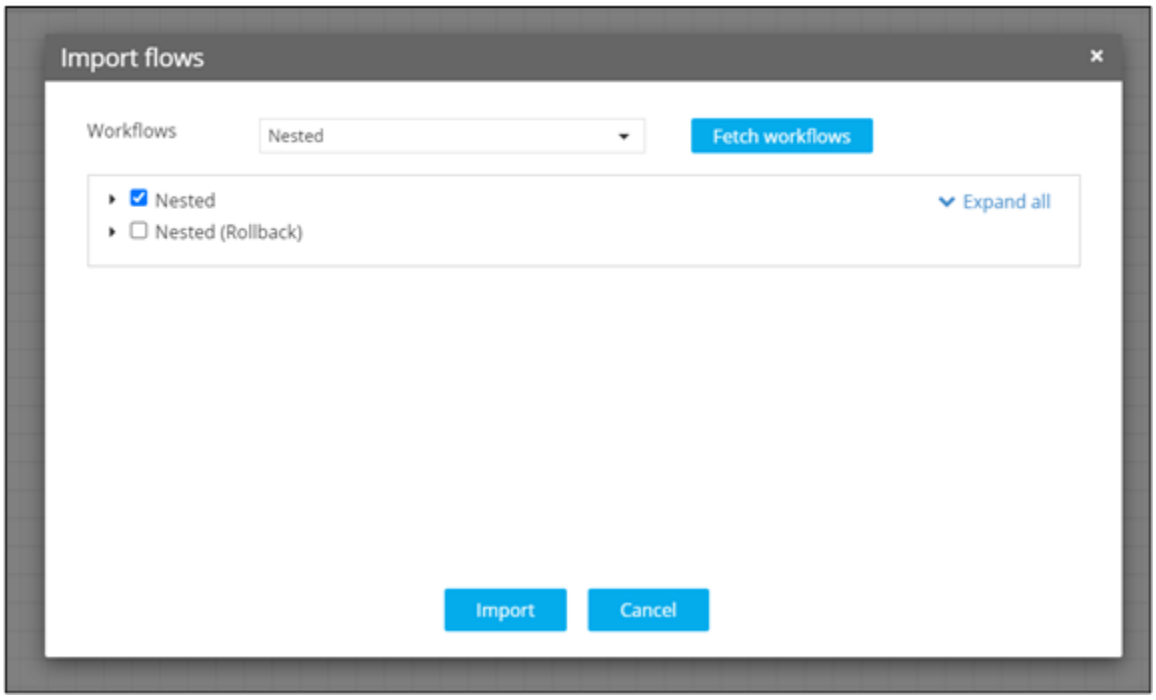


11. To reuse a workflow, click .
12. Click **Import**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

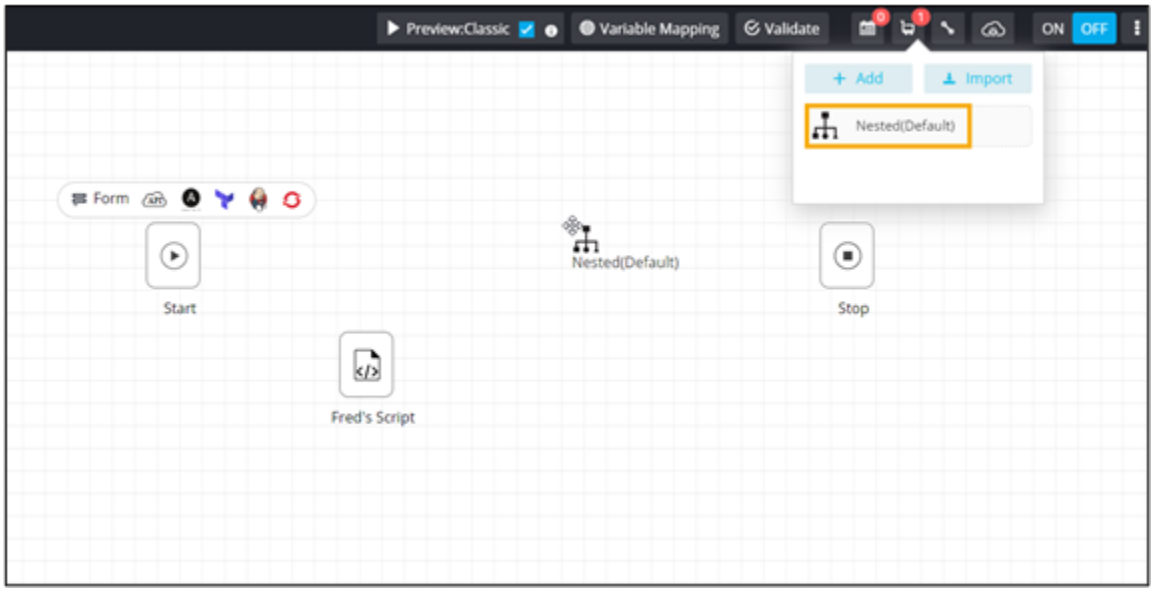


13. Import the workflow.

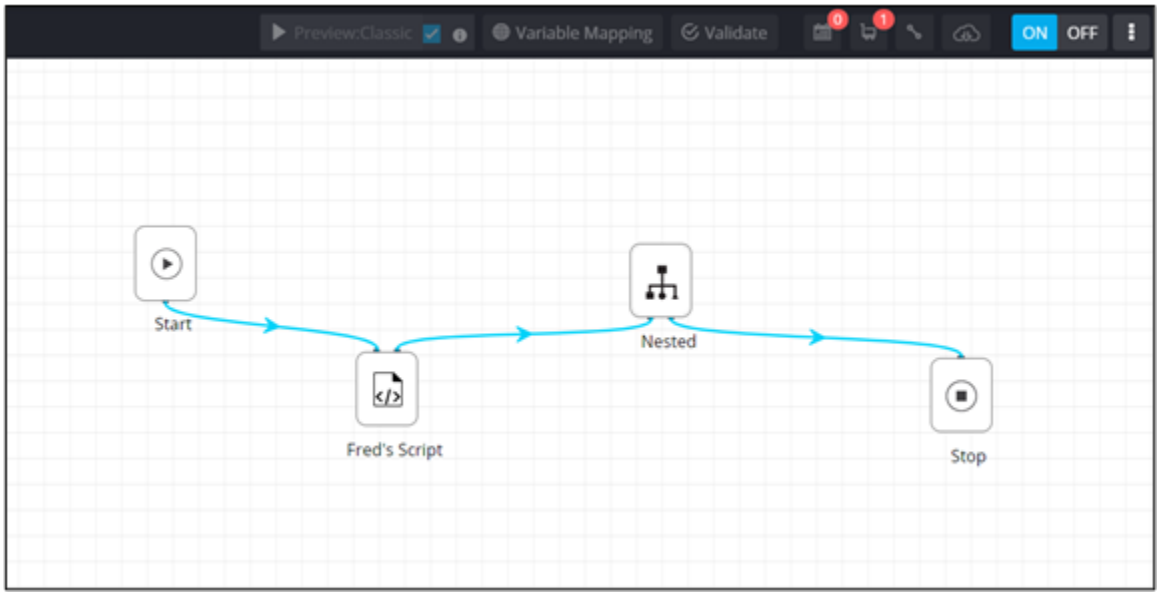


14. Drag and drop the imported workflow into the workspace.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

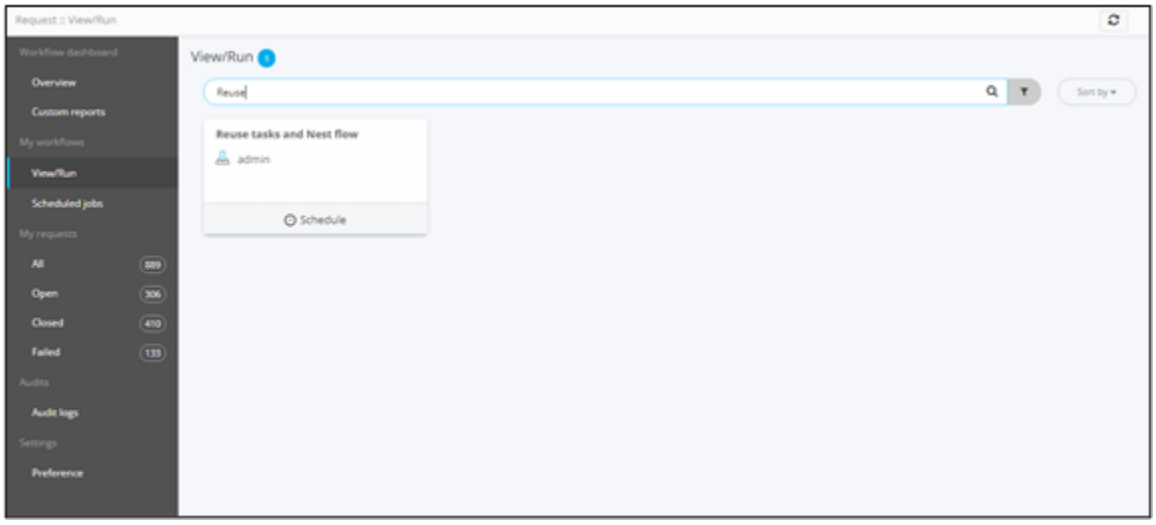


15. Refer the Global variables if needed.
16. Connect the tasks and enable the workflow.

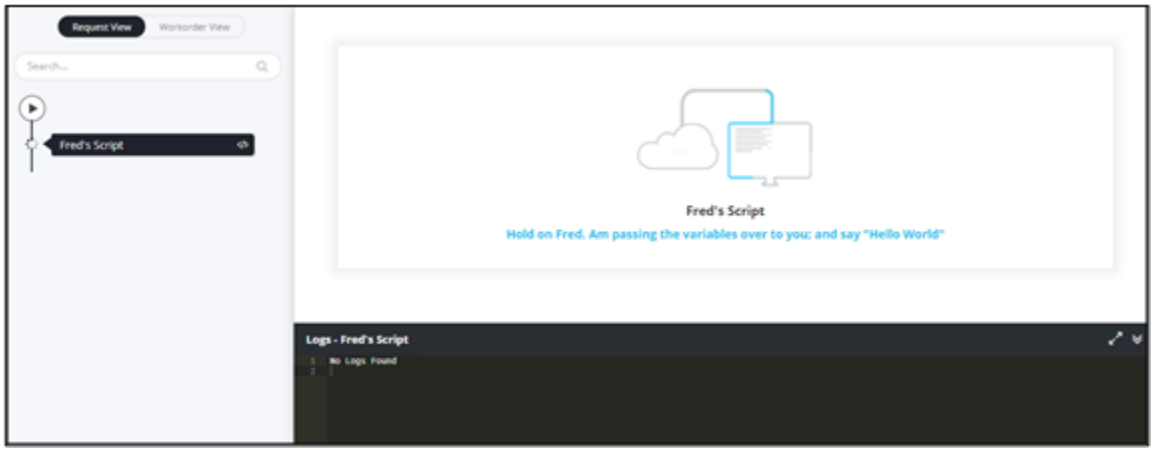


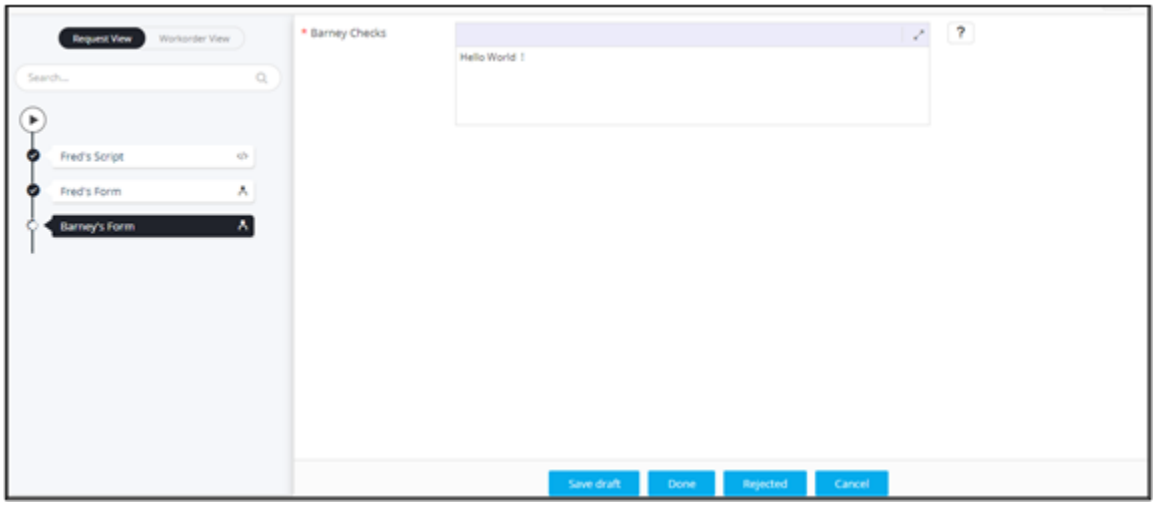
17. Trigger the workflow from the Request :: View/Run page.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Workflow executed with the reused task.



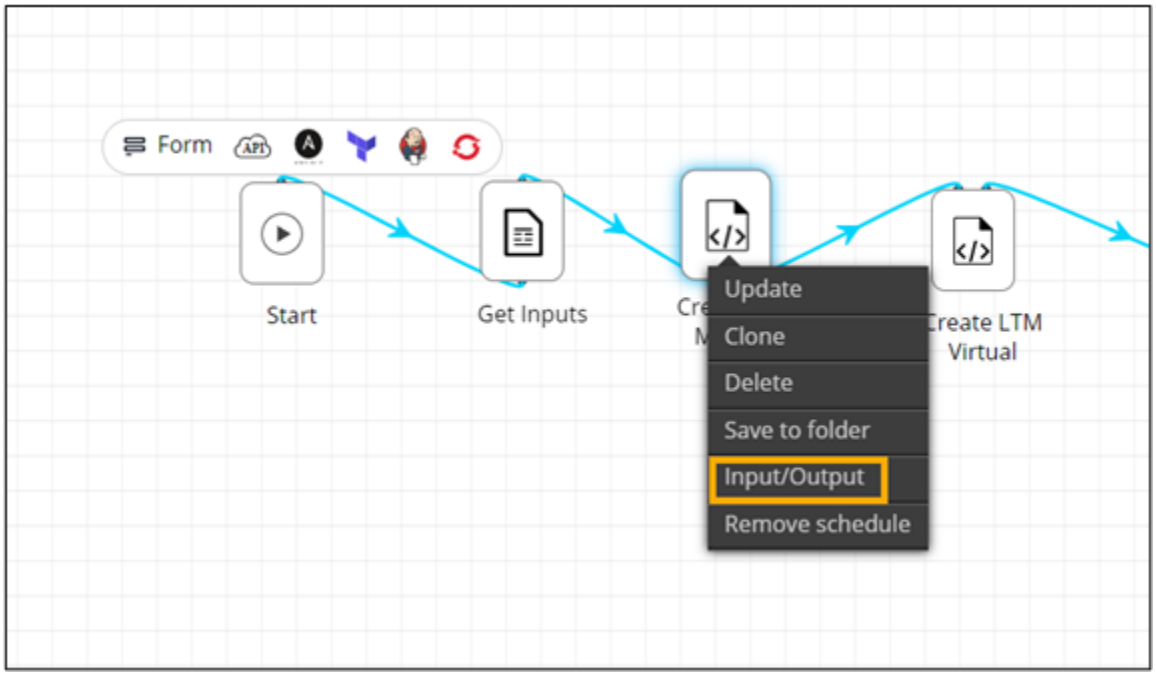


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

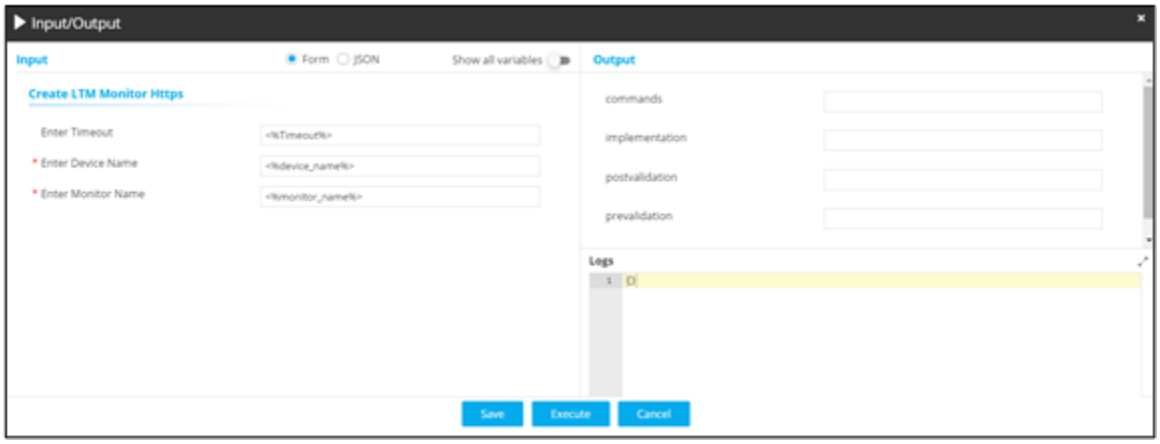
Input/Output Variable Mapping

1. To see the mapping of input/output variables in a task, right-click on the task to see the options.
2. Select **Input/Output** from the list of options.

   The input/output variables defined in the task are displayed.



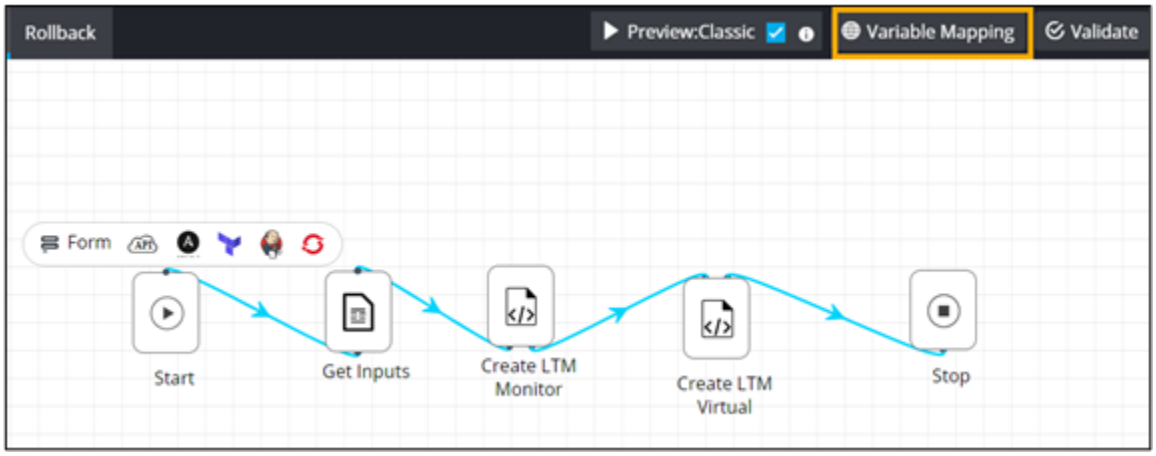
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Variable Mapping

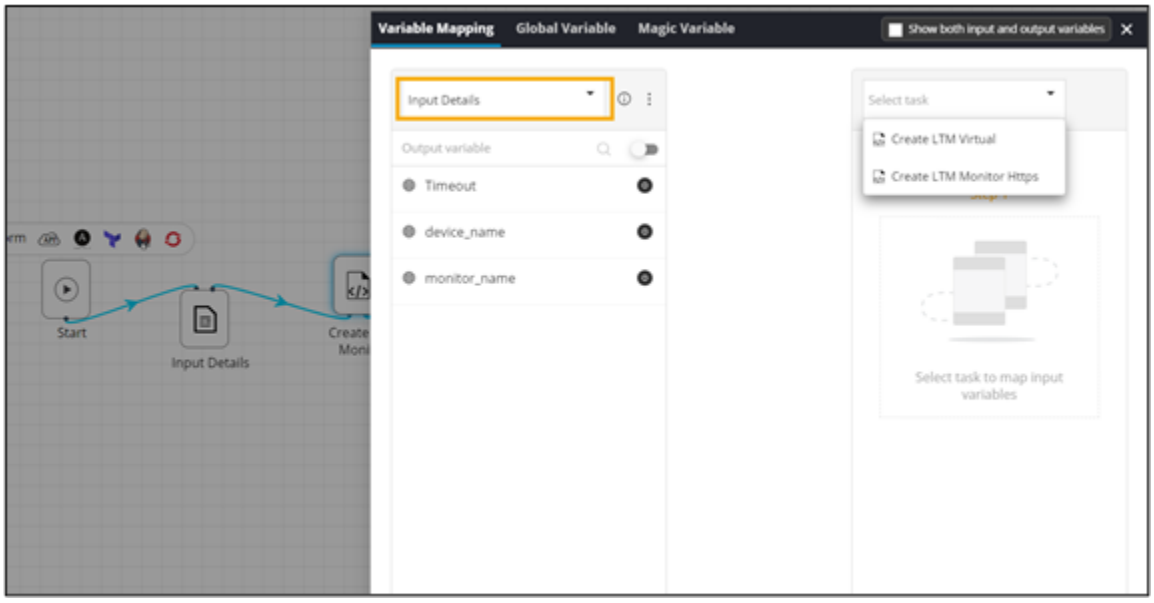
Variable mapping allows you to map the variables between tasks. You can select one or more tasks in order to map variables and get a quick view of the number of variable associations across tasks.

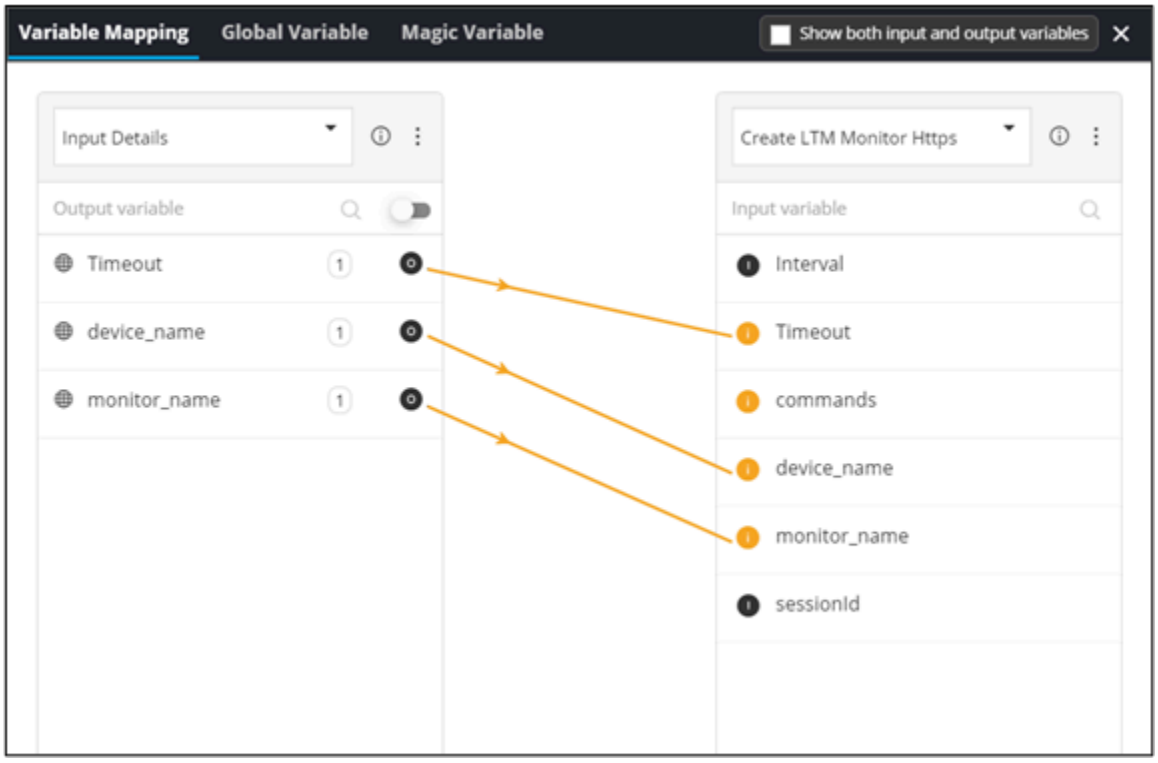
1. Design a workflow or open an existing workflow,
2. Click **Variable Mapping**.



3. Under **Variable Mapping**, select the tasks for variable mapping.

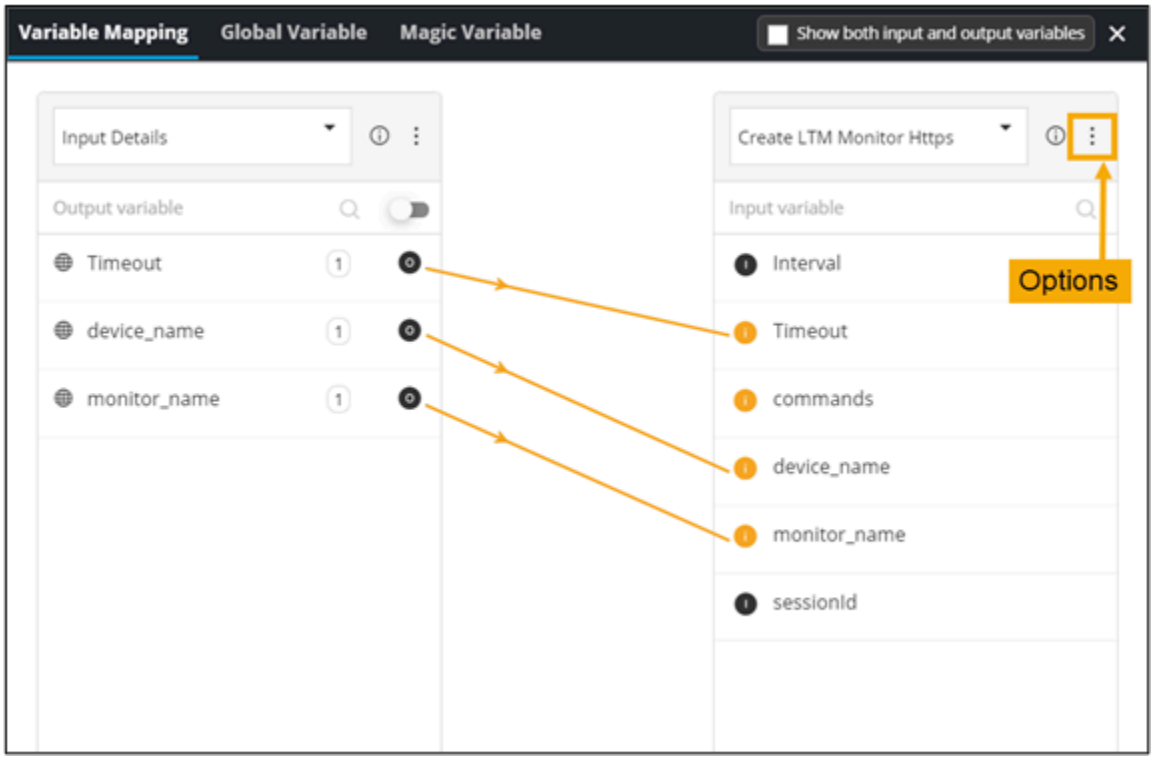
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



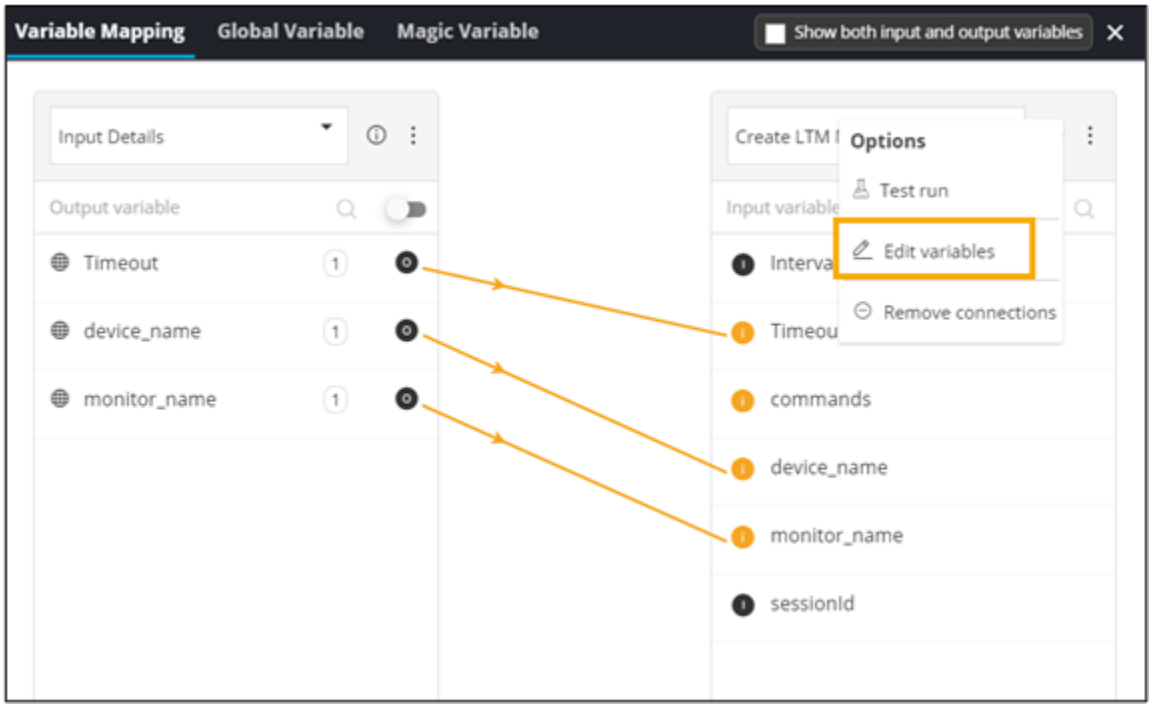


4. To see options, click ⋮.

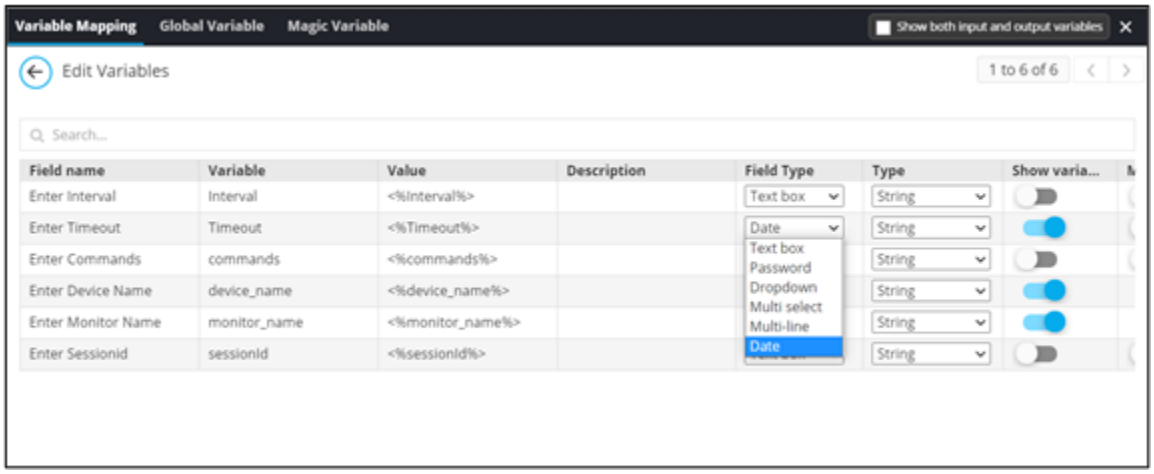
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



5. To make changes in the field parameters, select **Edit variables**.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

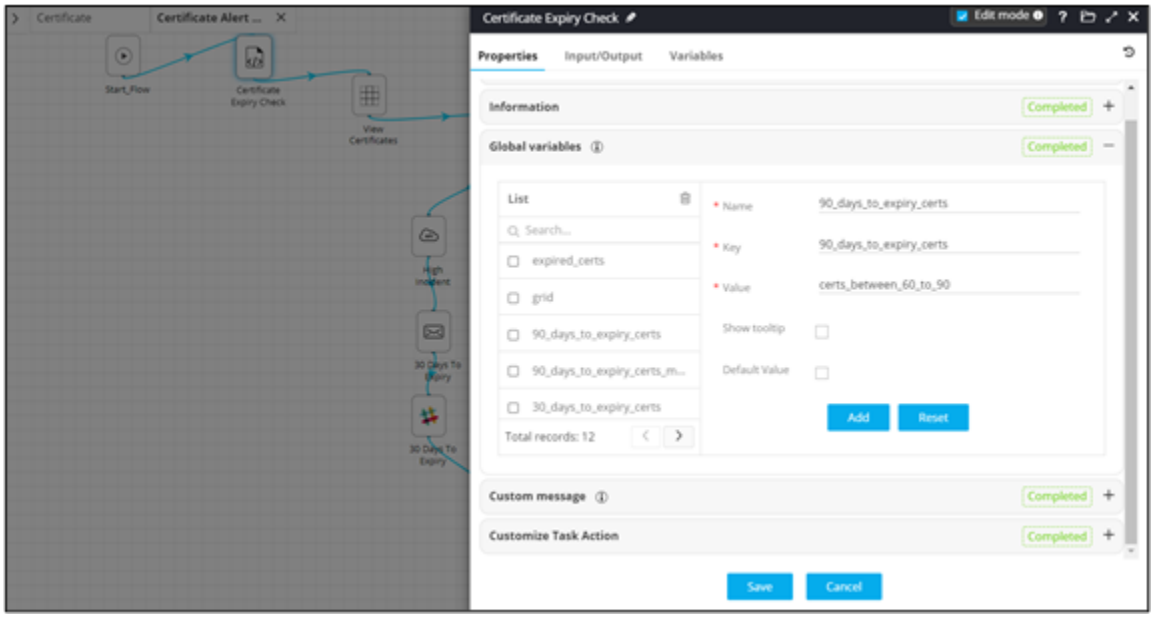


- Global Variables
- Magic Variable

Global Variables

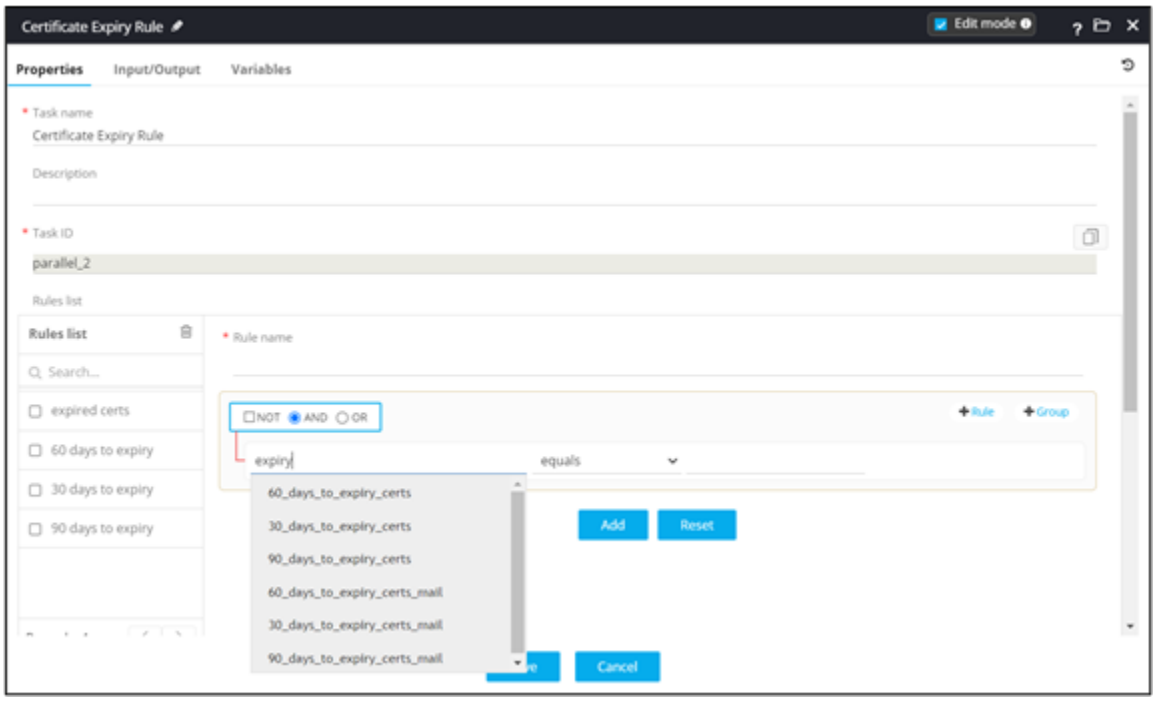
Variables can be declared globally and referenced across any stage of the workflow thus allowing data to flow seamlessly across the entire workflow process.

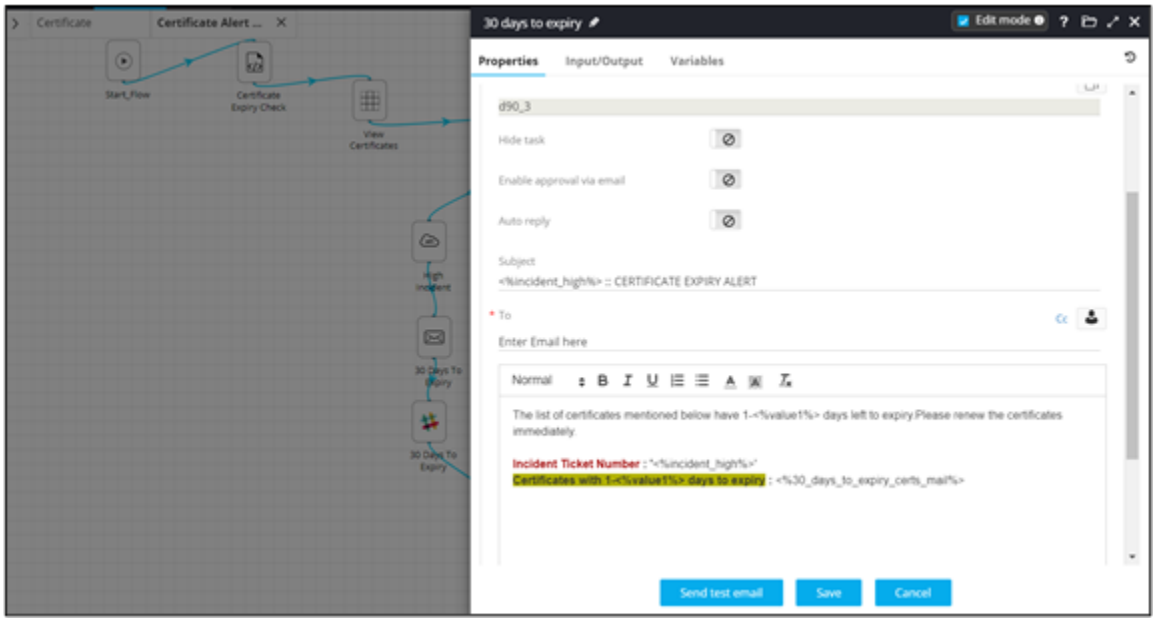
1. Design/modify a workflow.
2. Click on any workflow task.
3. In the task window, under **Properties**, in the the **Global Variables** tab, define the **Key** and **Value**.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

4. Click **Add**.
5. Auto-populate and reference the global variables using keyword match.



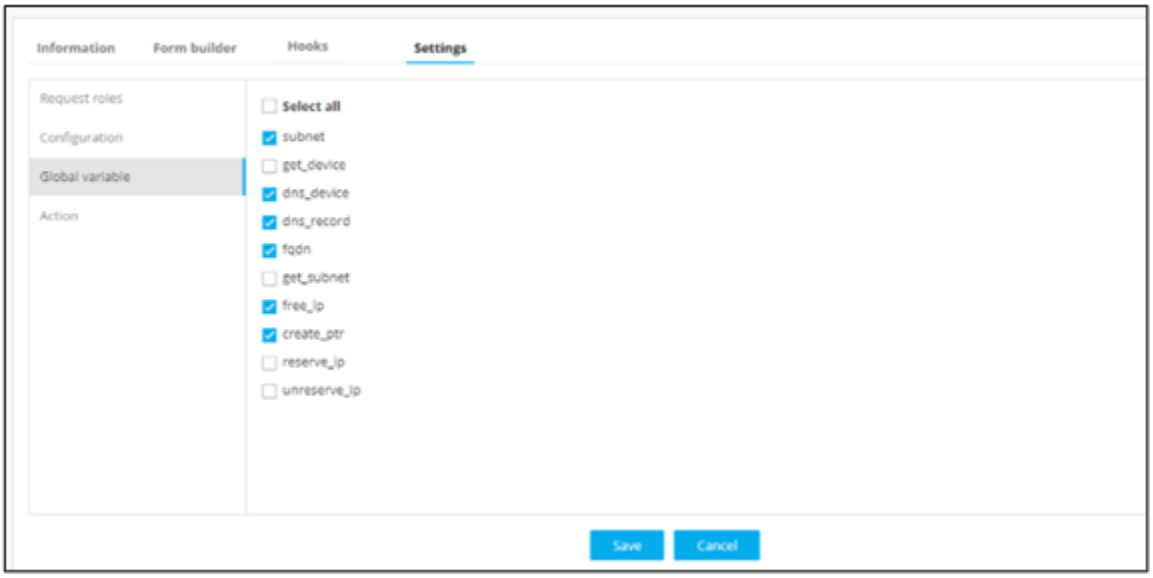


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Declaring Variables from Tasks
- Referring Variables
- Global Variable Inventory
- Configuring Tooltips

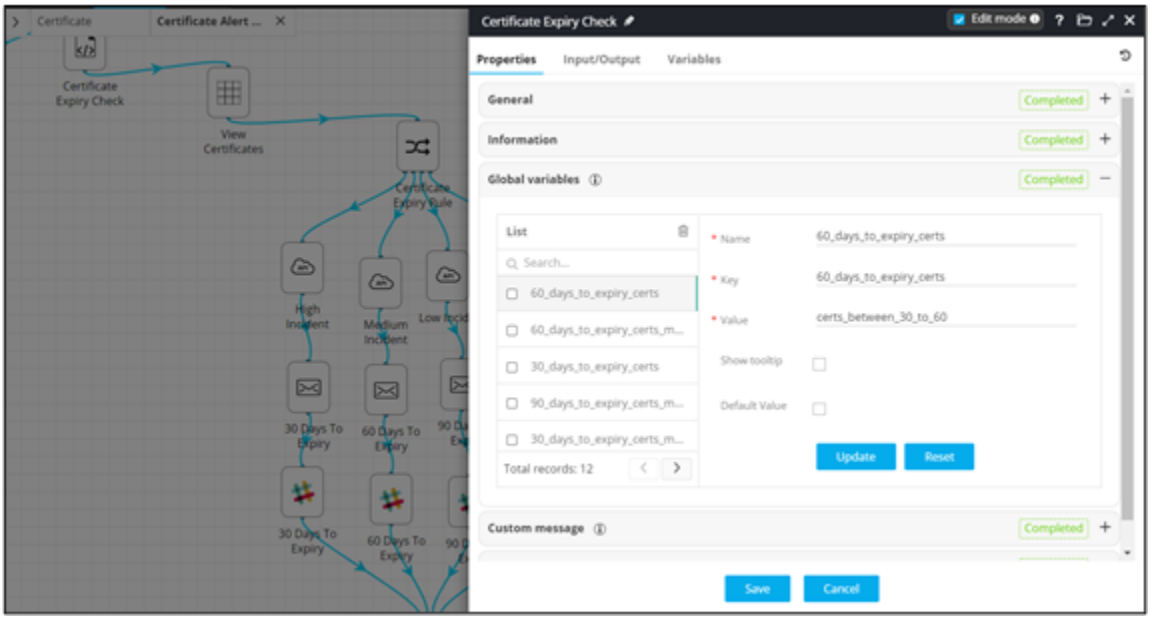
Declaring Variables from Tasks

- Declaring variables from a Form task

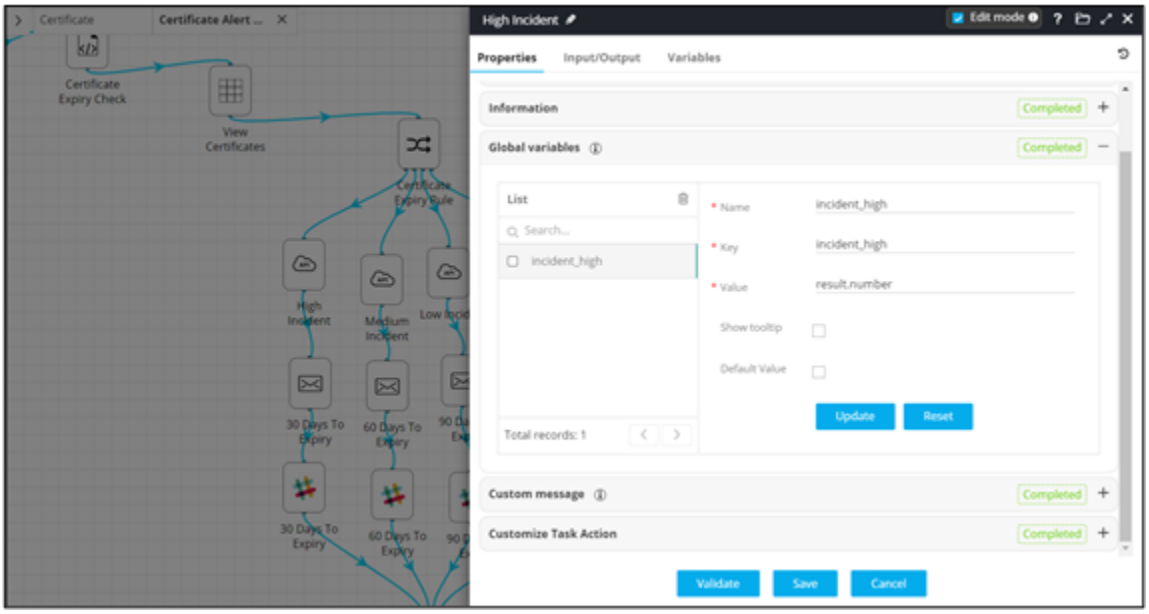


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

• Declaring variables from a Script task



• Declaring variables from a REST API task



Referring Variables

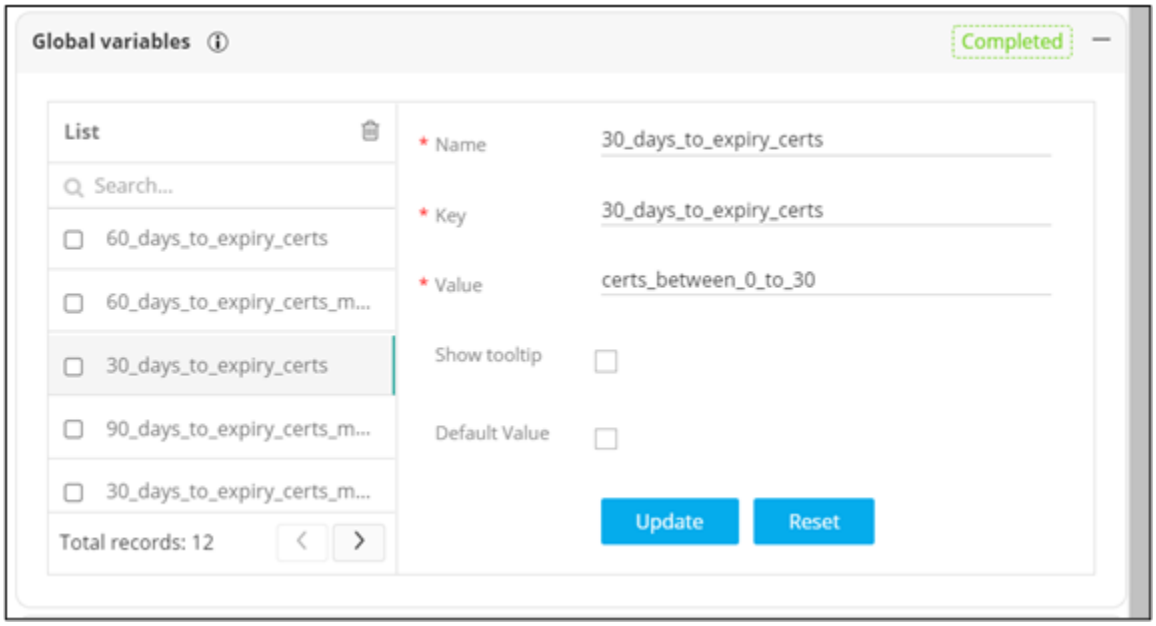
Once declared, global variables can be simply referred across any other stage using the following syntax:

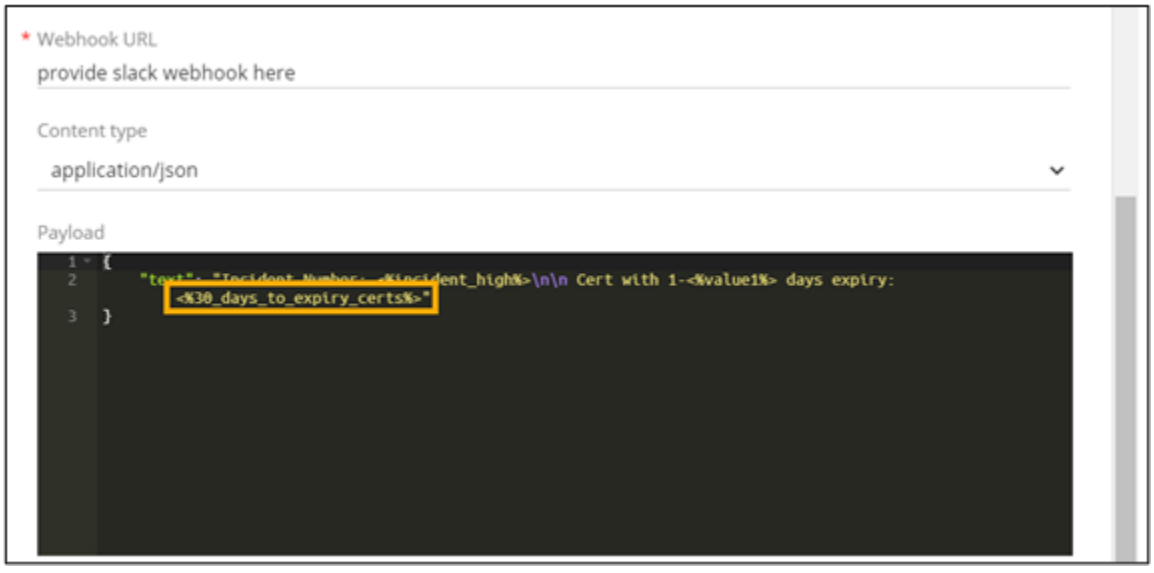
`<%variable%>`

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

For example, once the global variable “30_days_to_expiry_certs” is declared in the Script task, it can be referenced anywhere in the workflow as

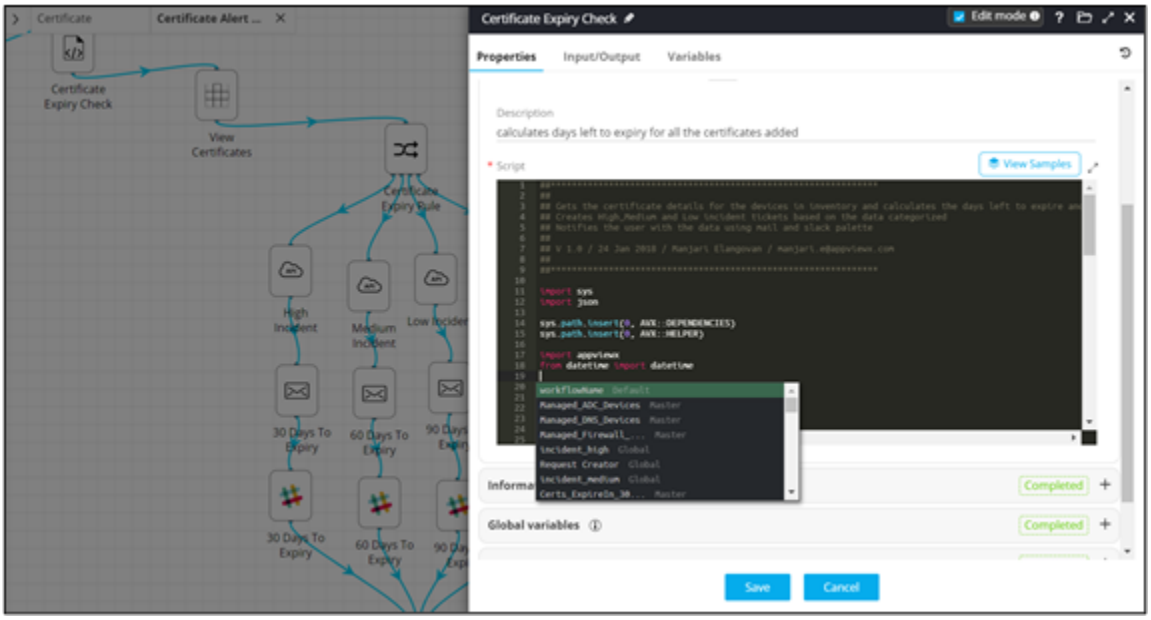
<%30_days_to_expiry_certs%>





Note: Use keyboard shortcut Ctrl+g to view and refer global variables within a Script task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

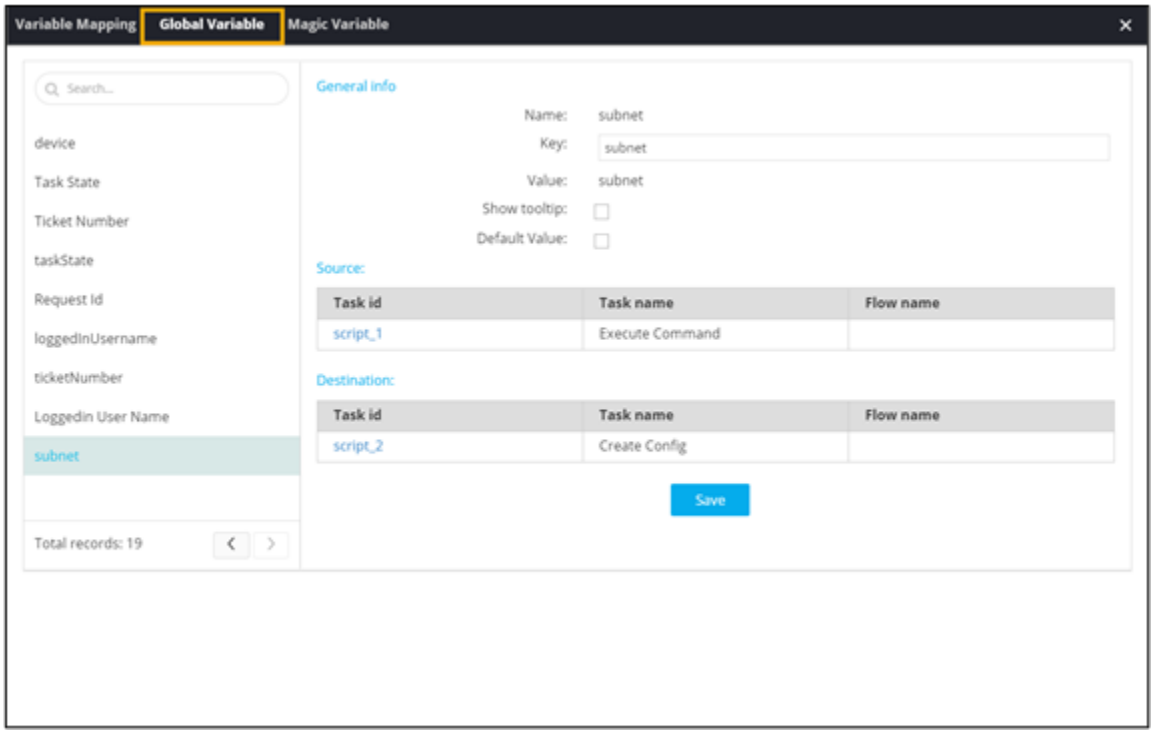


Global Variable Inventory

The Global variable inventory shows a single view of the source and destination tasks and stages across which a global variable has been referenced within a workflow.

- Provision to view and search the list of global variables declared and referenced across various stages of the workflow
- Provision to view the source and destination tasks, nested workflow name across which a global variable has been declared and referred
- Provision to click and update the specific workflow task (via Task id)
- Provision to define one or more global variables as ‘tooltip’ information for a workflow request

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Note: Any updates to the Key field will be reflected globally across the workflow.

Configuring Tooltips

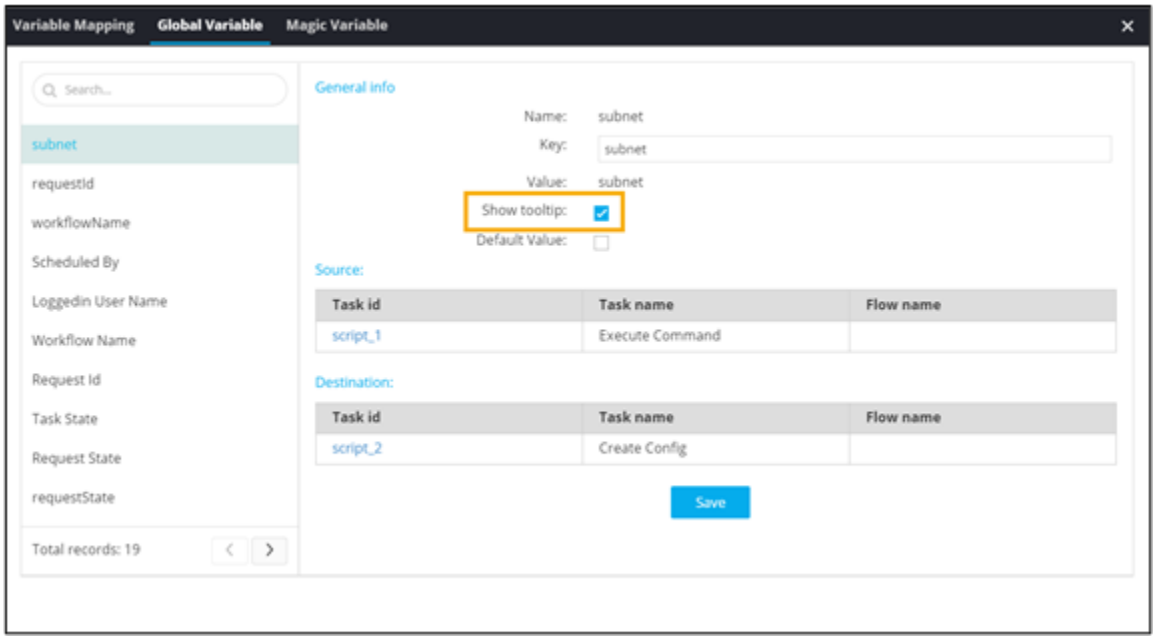
This allows you to define one or more global variables as 'tooltip' information. Once the tooltip is defined, you can right-click a workflow request and view details. Since typically workflows carry a significant amount of data, configuring a tooltip provides the flexibility to select one or more global variable values to be displayed as tooltip information.

To configure a tooltip:

1. Design/Modify a workflow.
2. Drag and drop the required workflow tasks.
3. Assign the required values as global variables.
4. To view all global variables, click **Variable Mapping**.

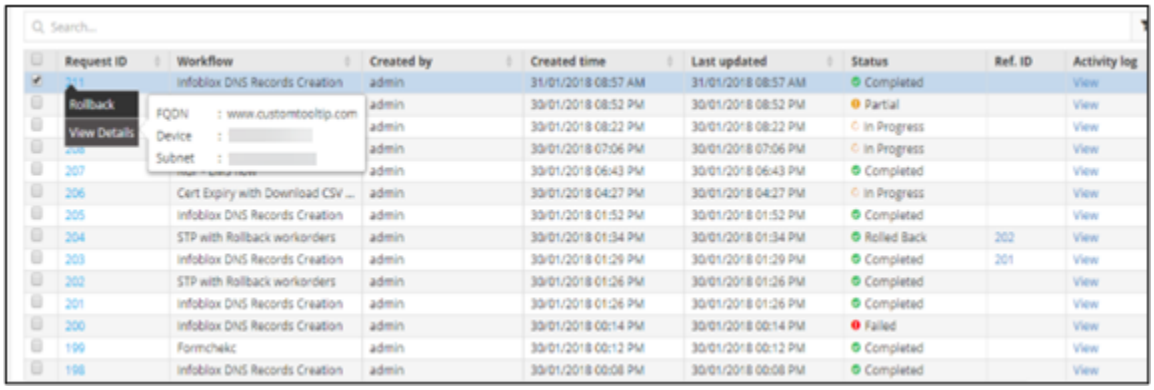
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

5. Under **Global Variable**, select the **Show tooltip** checkbox.



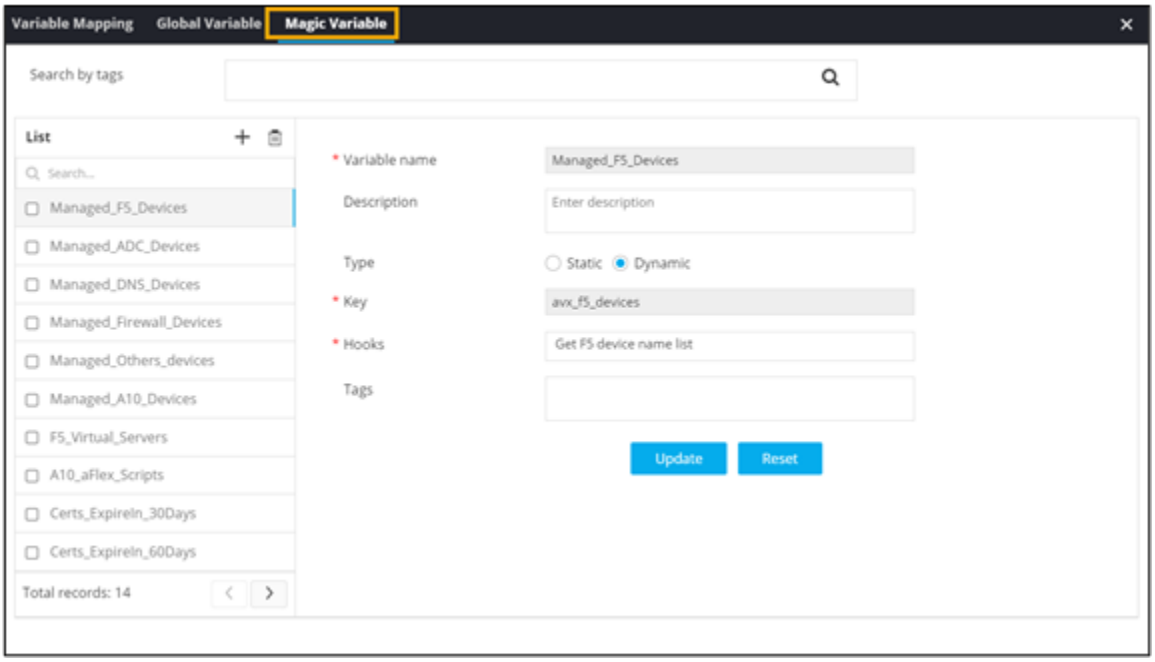
6. Save and enable the workflow.
7. Trigger the workflow from the Request :: View/Run page.
8. On the Request :: All page, right-click the Request ID of the workflow.
9. To see the tooltip information, click **View Details**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Magic Variable

This shows a list of all the Magic variables defined in various tasks across workflows.



Rollback Workflow

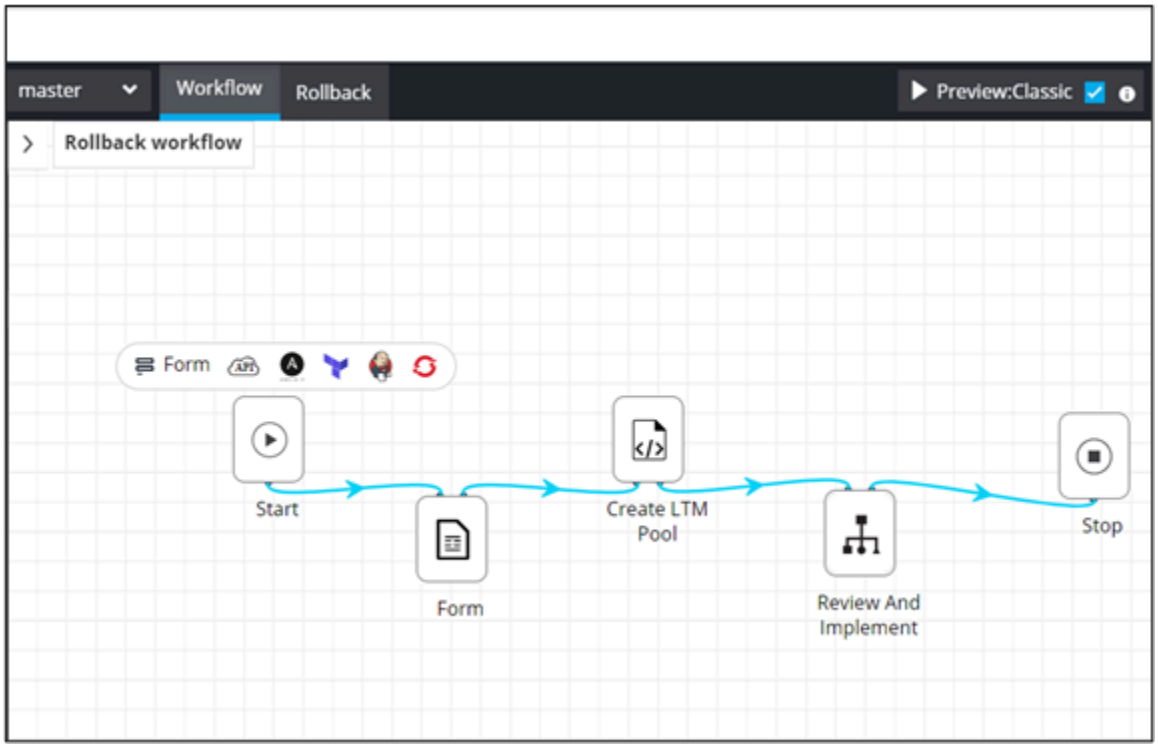
You can either design new rollback workflow(s) or import existing rollback workflow(s) under a parent workflow. A manual rollback can be triggered by accessing Workflow Request and performing a right-click against a workflow request ID.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

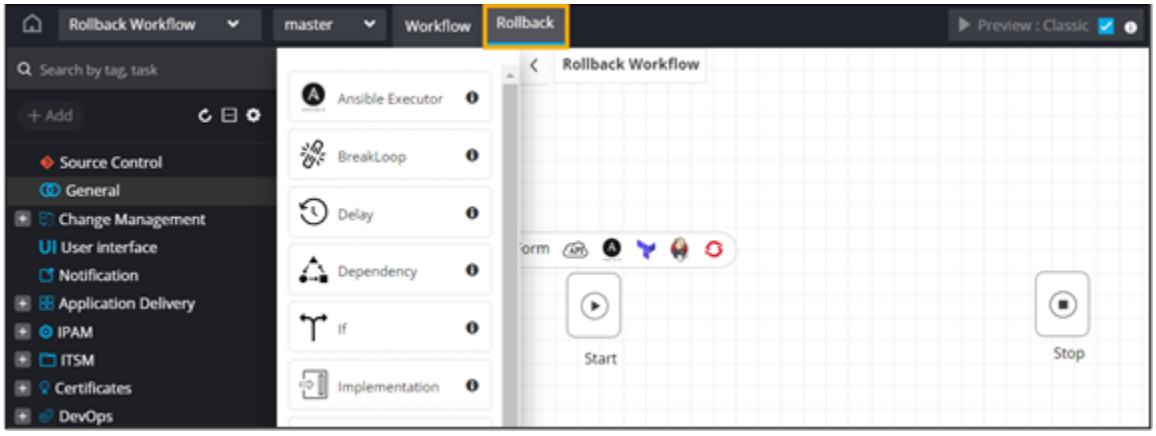
- Manual Rollback
- Auto Rollback

Manual Rollback

1. Design a workflow using workflow tasks.

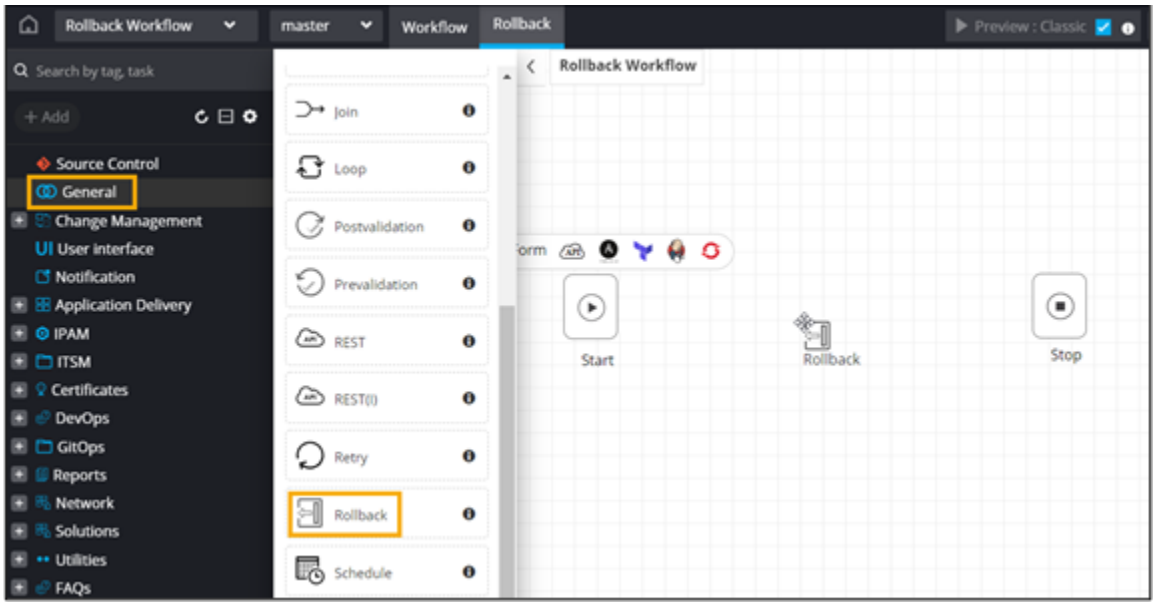


2. To add a rollback workflow, click **Rollback**.

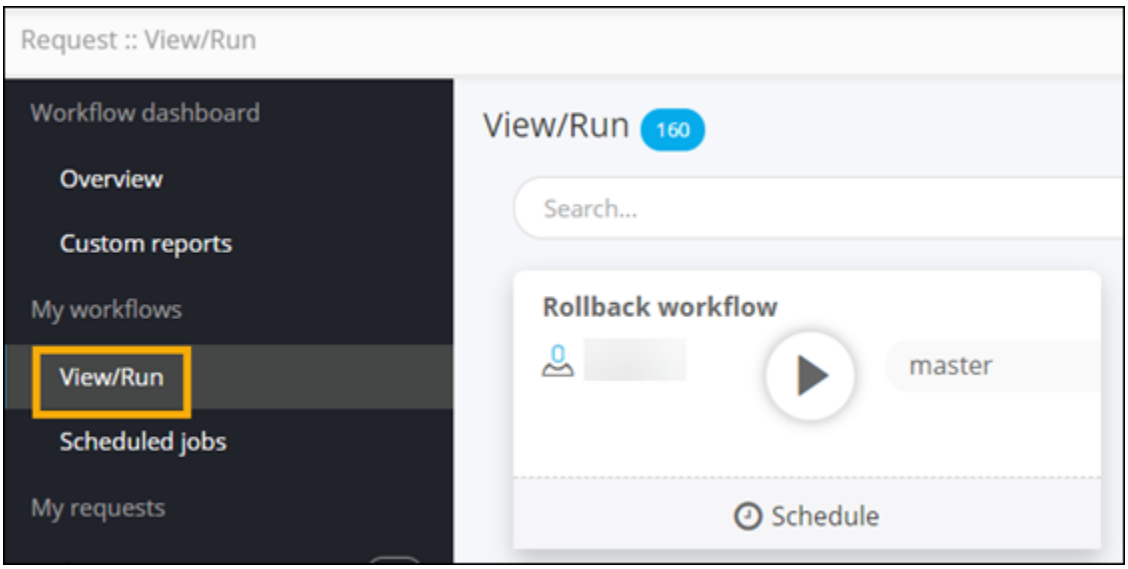


3. From the **General** section, drag and drop the **Rollback** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

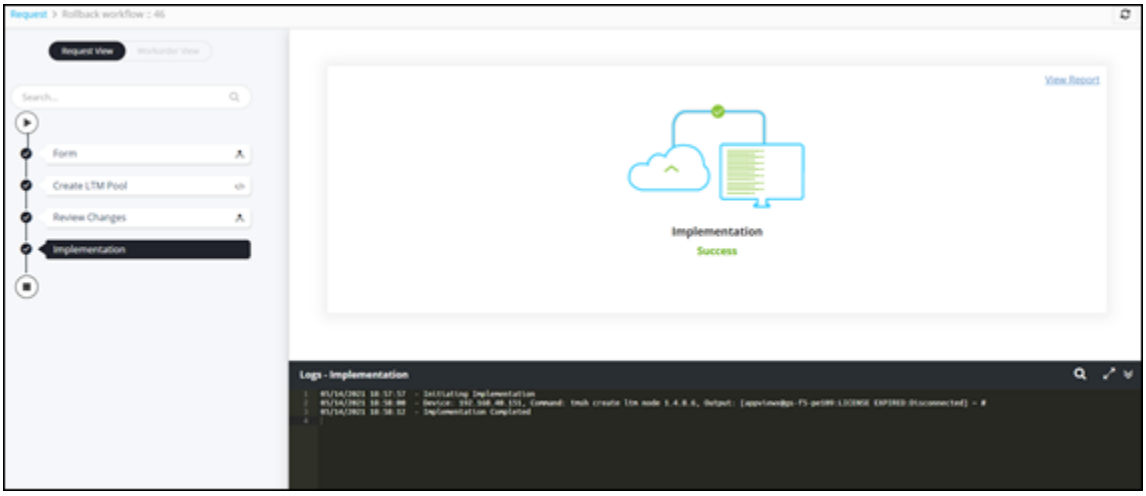


4. Connect and enable the workflow.
5. From the upper left corner of the screen, click ☰.
6. Trigger the workflow from the Request :: View/Run page.

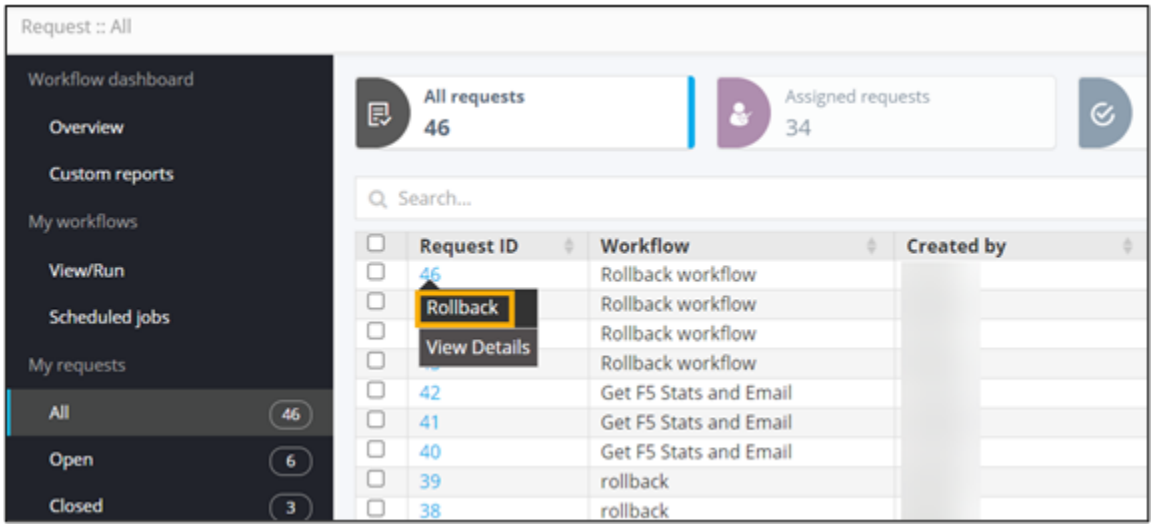


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

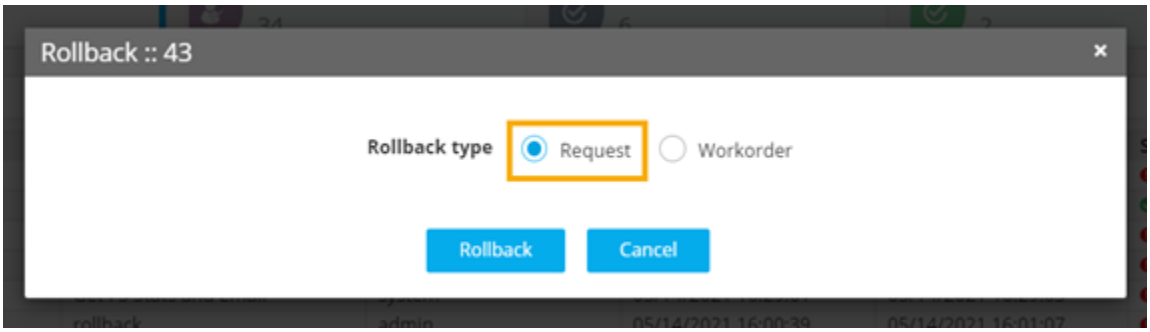
Workflow is executed.



7. On the Request : All page. right-click on the workflow Request ID and select **Rollback**.

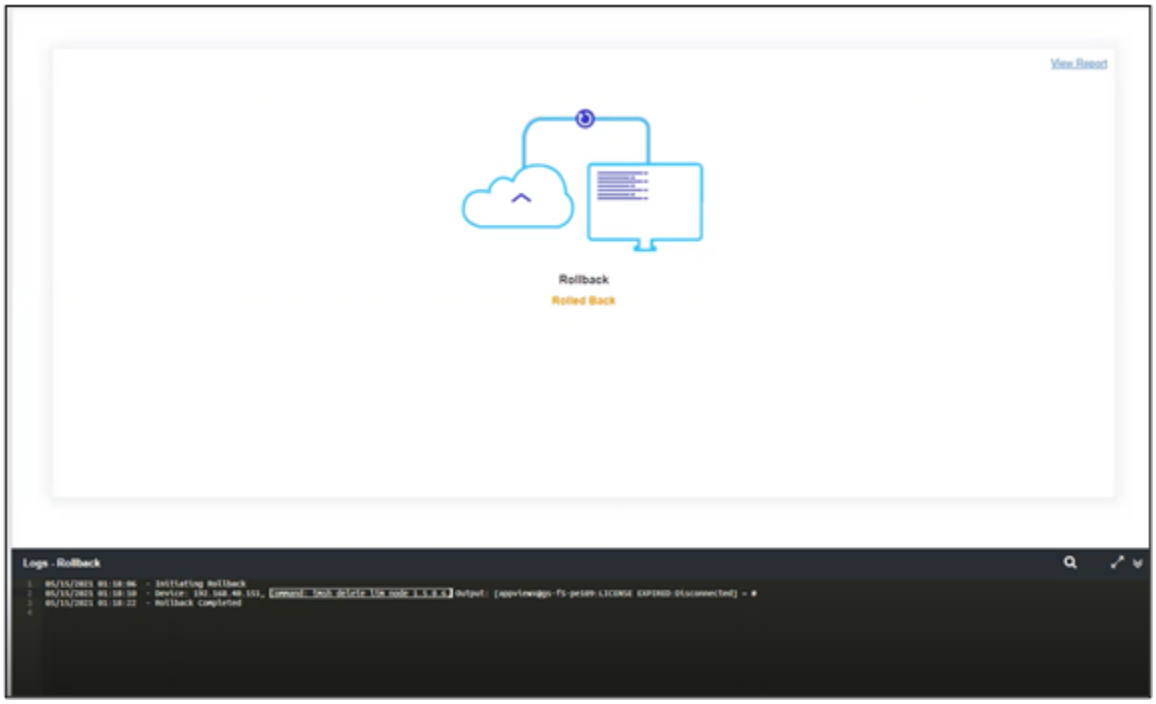


8. Select **Rollback type** as **Request**.



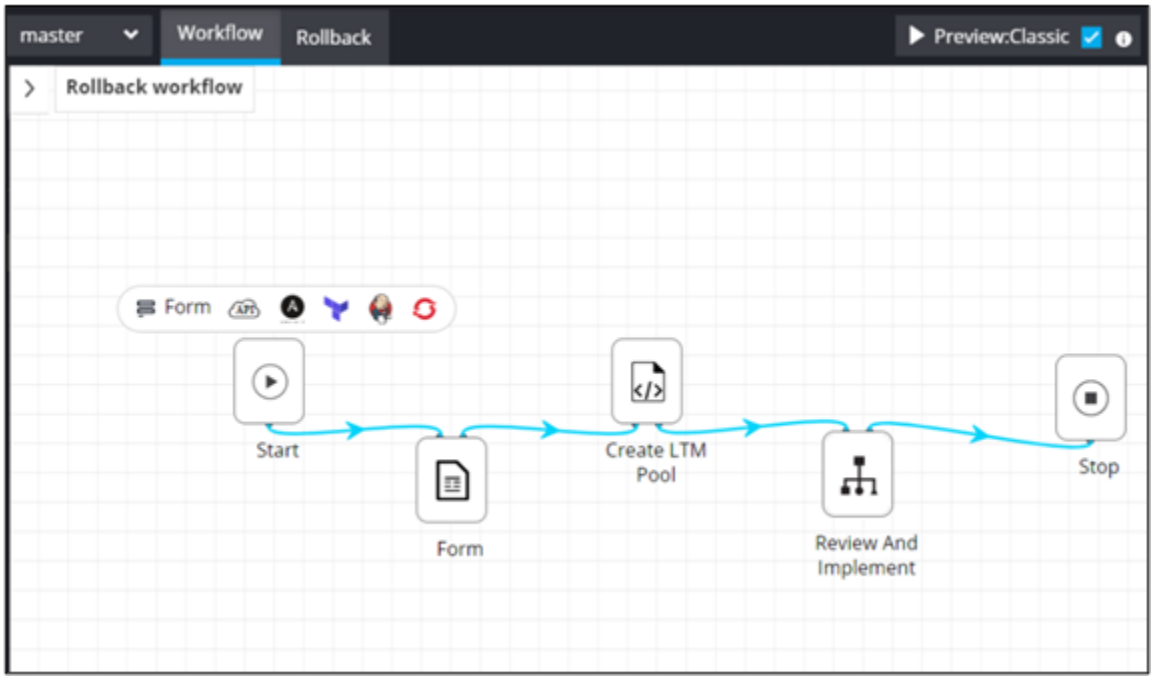
Workflow is **Rolled Back**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



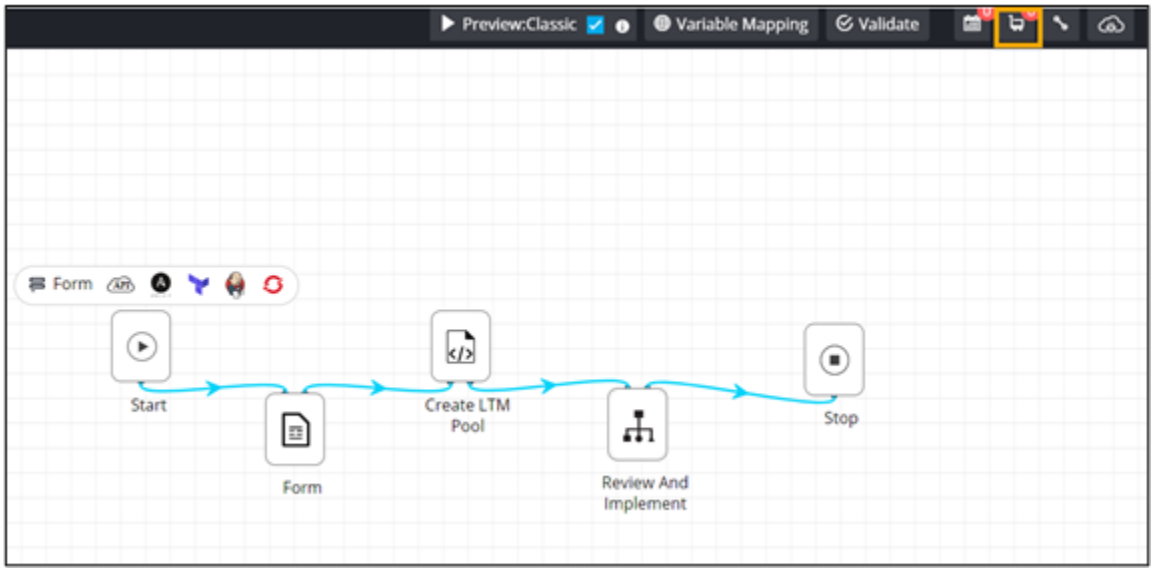
Auto Rollback

1. Design a workflow using workflow tasks.

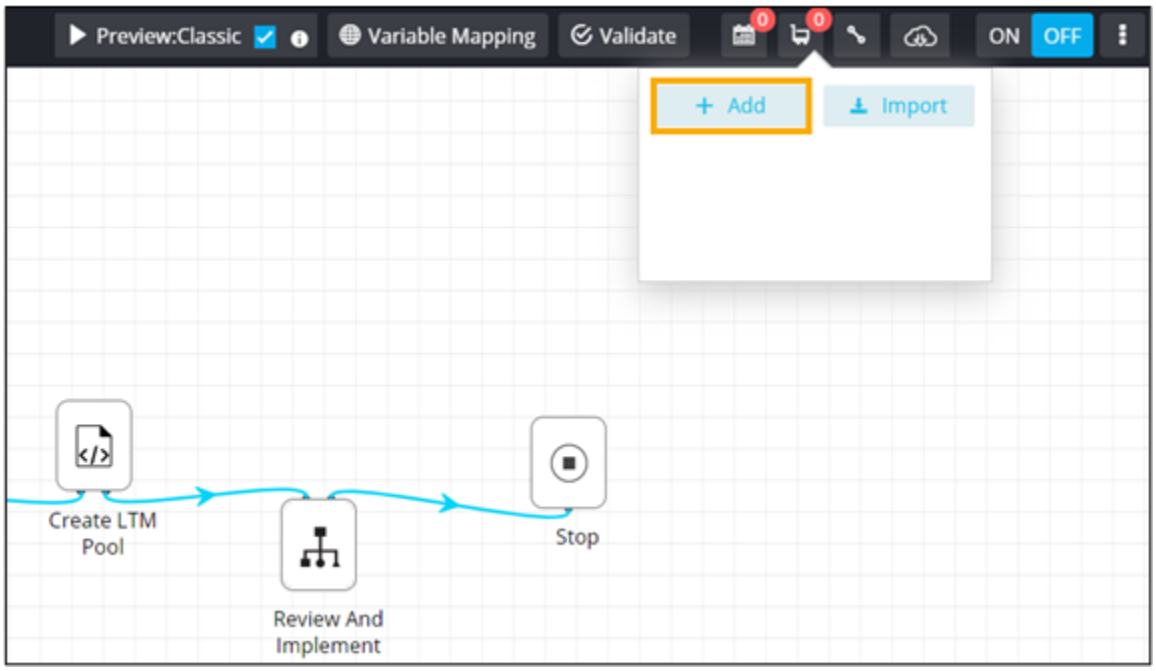


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

2. To create a subflow, click .

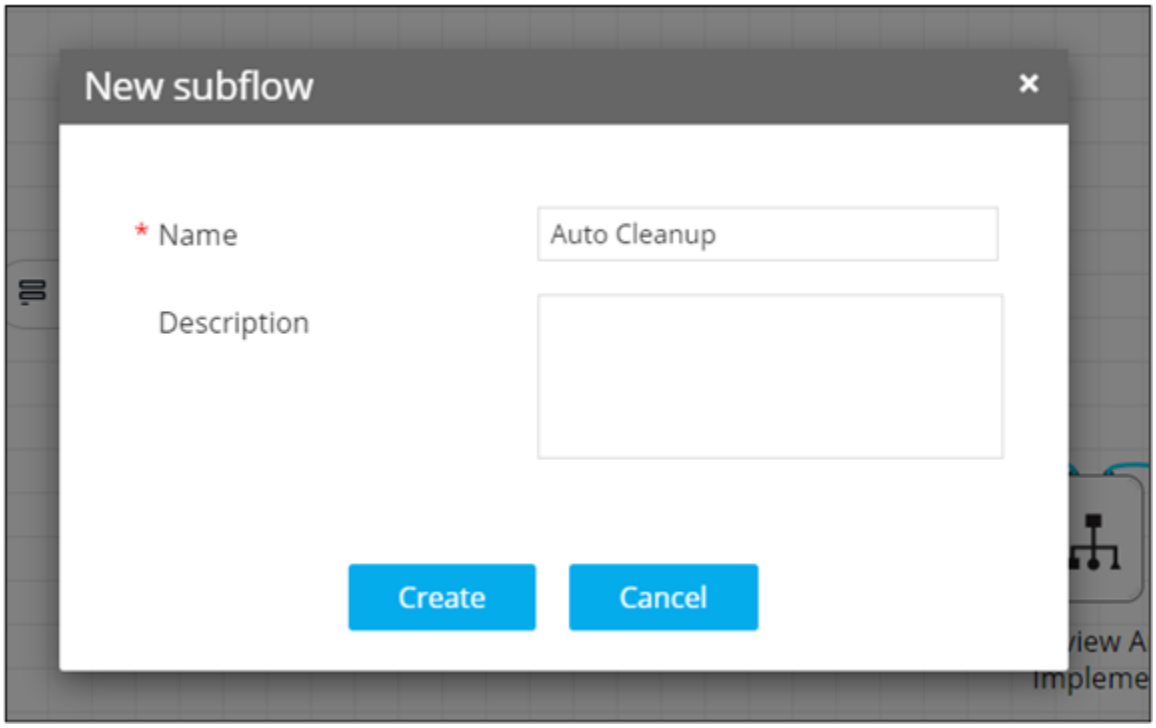


3. Click **Add**.

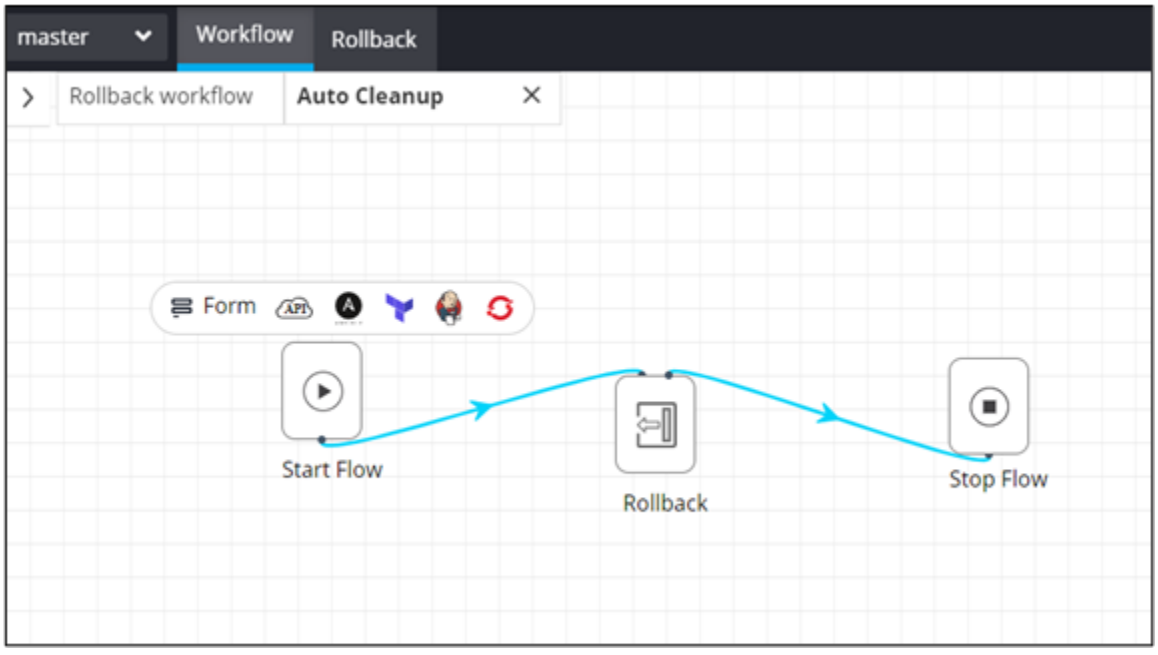


4. Provide a suitable workflow **Name**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

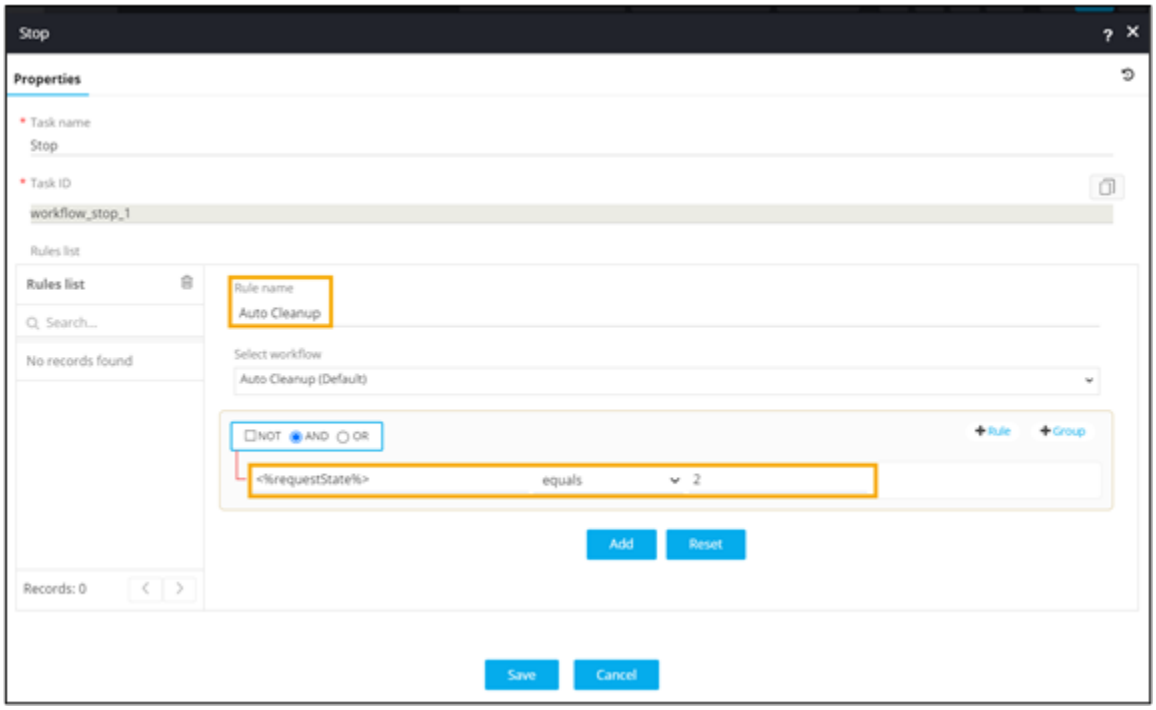


5. From the **General** section, drag and drop the **Rollback** task.
6. Connect the workflow.



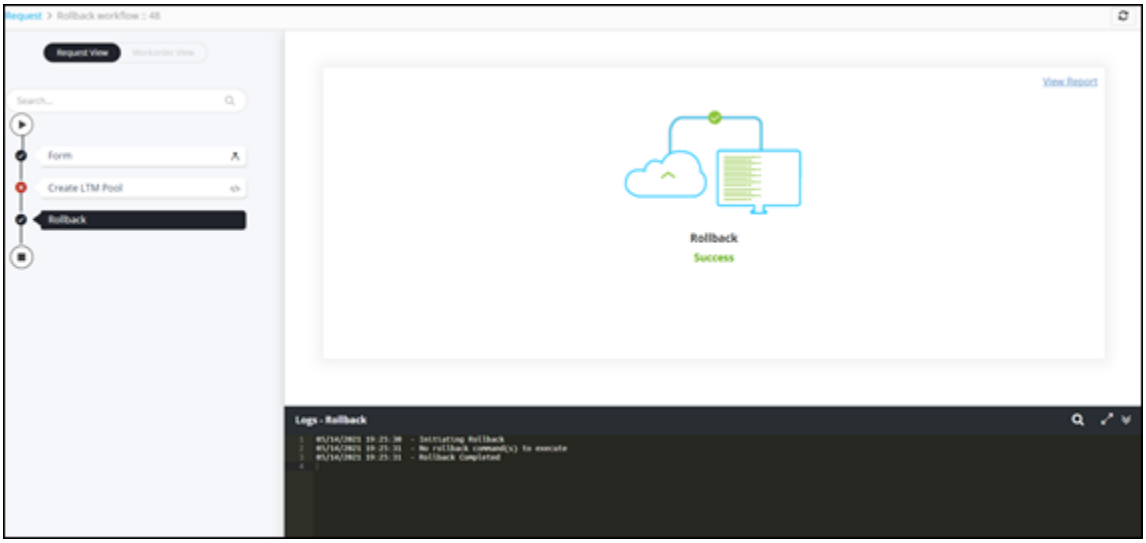
7. Double-click on the **Stop flow** task of your main workflow.
8. Define a rule to trigger the auto rollback.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



9. Click **Add**.
10. Save and enable the workflow.
11. Trigger the workflow from the Request :: View/Run page.

    Auto Rollback is initiated based on workflow state.

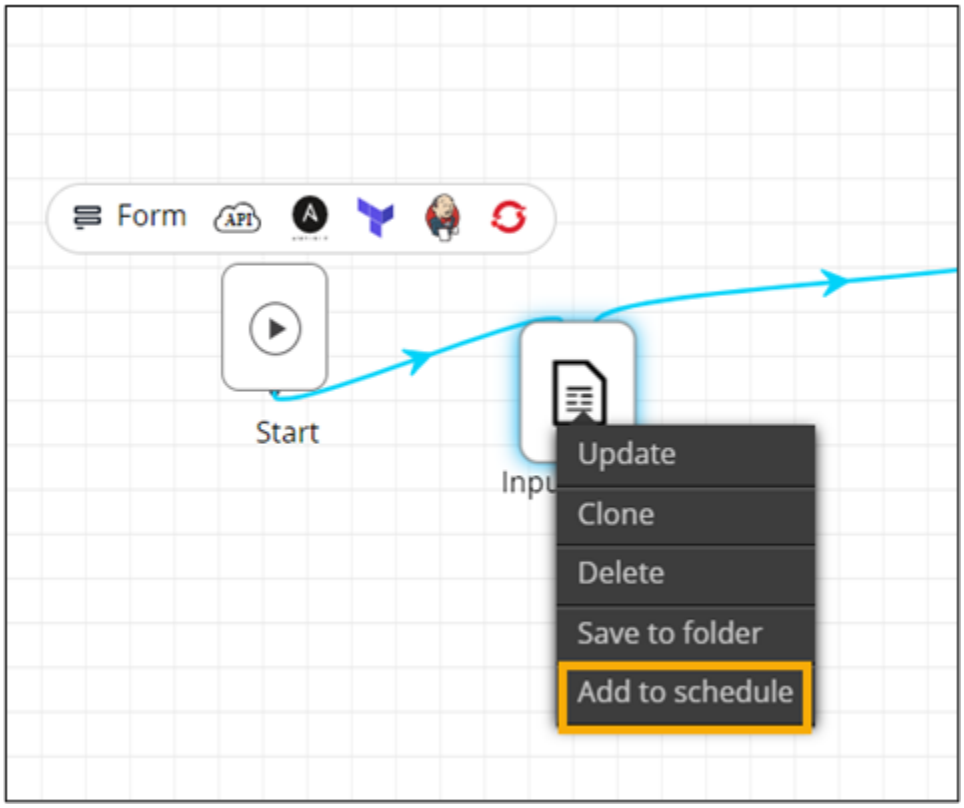


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Task Scheduler

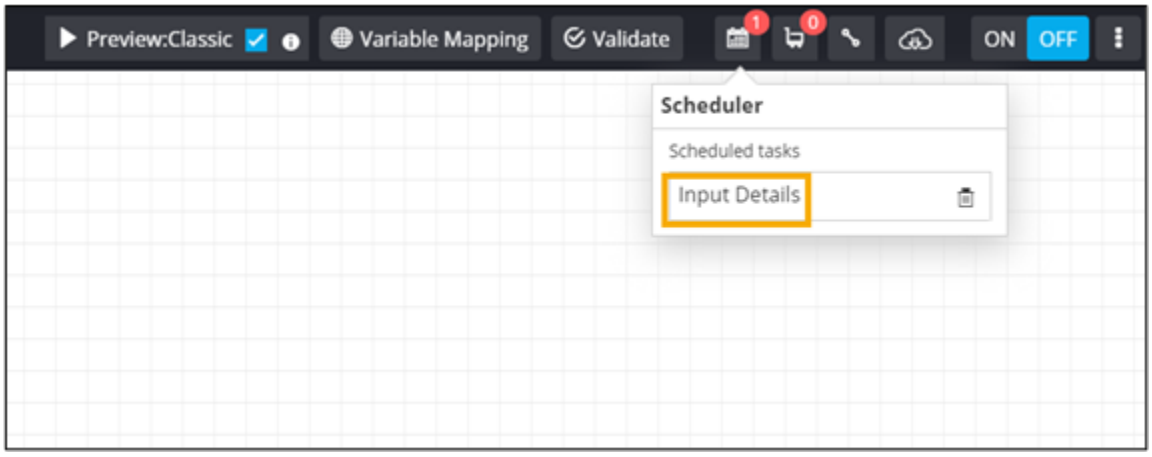
You can schedule workflows and also schedule specific tasks within the workflows that require user inputs.

1. Design a new workflow or open an existing workflow.
2. Right-click the task to be scheduled to view options.
3. Select **Add to schedule**.

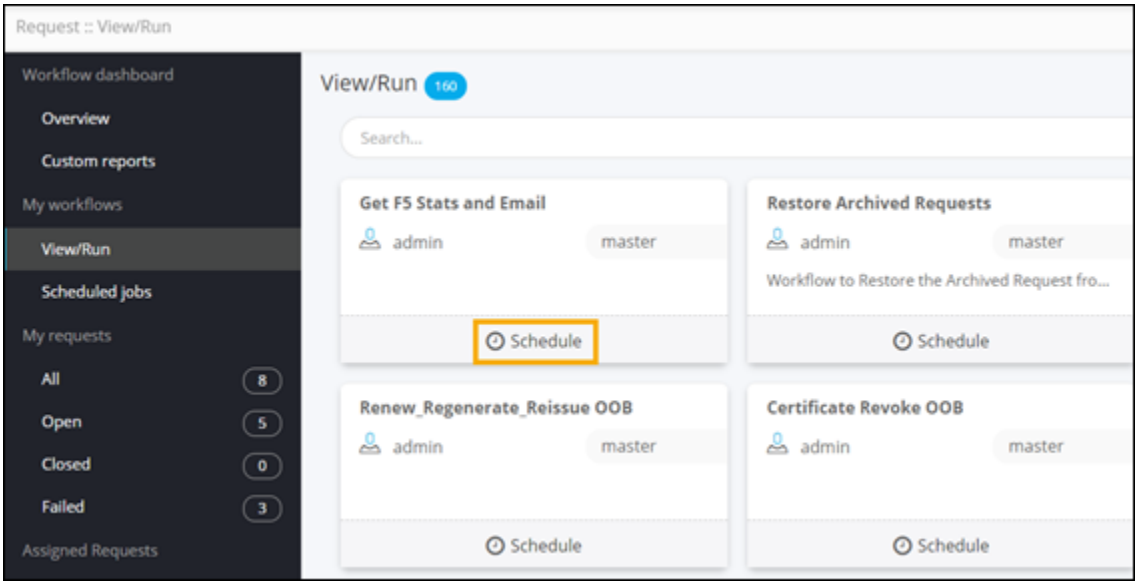


4. Click **Yes** in the **Confirmation** pop-up window.
   The task is added to the **Scheduler**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

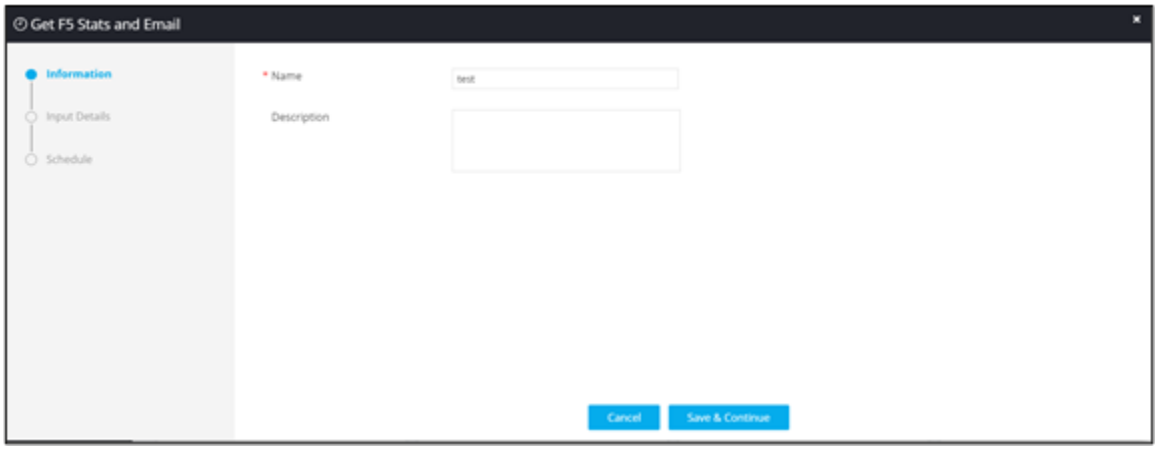


5. Enable the workflow.
6. On the Request :: View/Run page, search for the workflow and click **Schedule**.

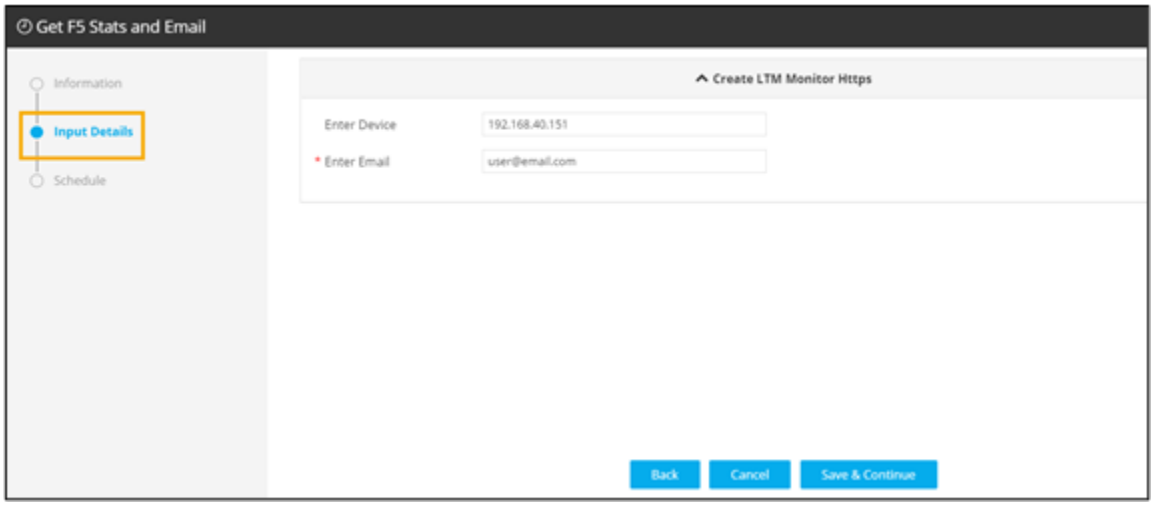


7. Enter a valid **Name**.

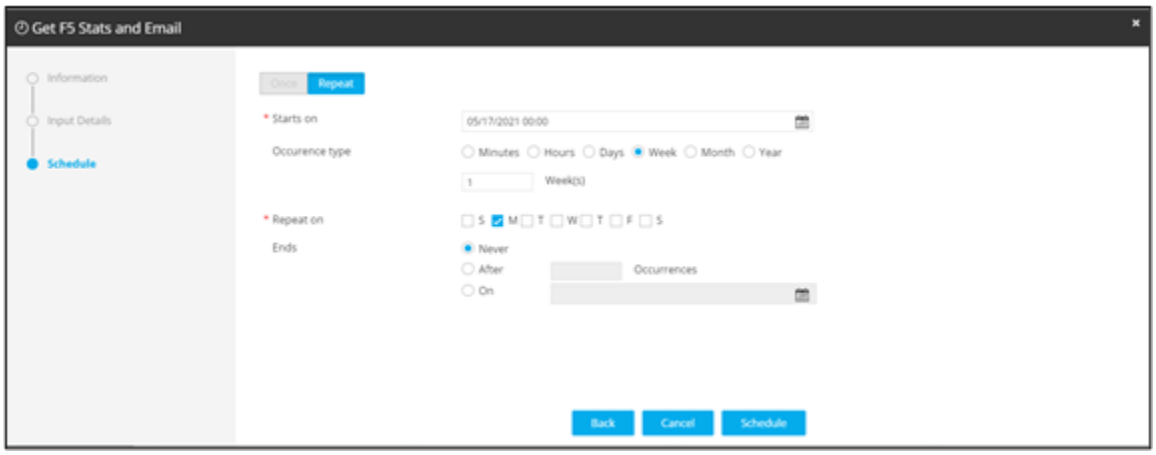
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



8. Enter the field information in the **Input Details** section.



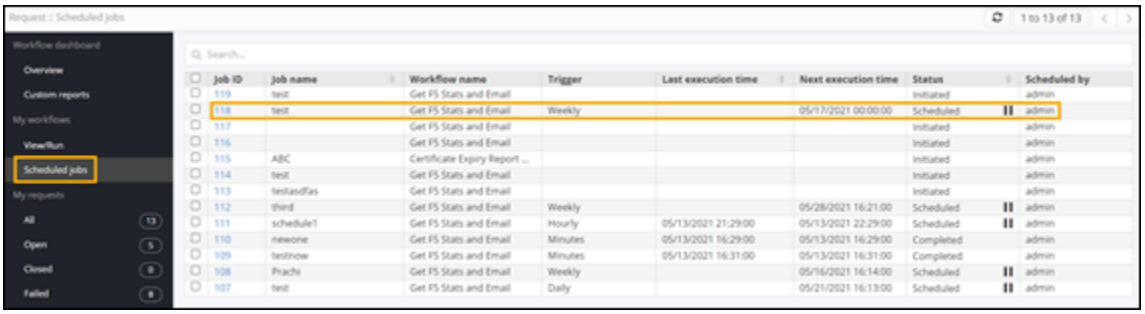
9. Enter the scheduling details.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

10. Click **Schedule**.

    The request is listed under **Scheduled Jobs**.



Workflow Cart

Workflow cart allows you to associate or nest one or more flow(s) within a parent workflow

- **Nested flow**: Provision to design new nested workflows or sub workflows
- **Reuse workflow**: Provision to import, reuse one or more workflow(s); and nest them under the parent flow.
- **Rollback**: Provision to design a new Rollback workflow for a parent workflow
- Nested Workflows

Nested Workflows

“Design once, reuse forever…”

Nested workflows allow you to reuse existing workflows and automate quickly. You can design new workflow(s) or import workflow(s) under a parent workflow. Once nested, the workflows are depicted in green. You can double click on the nested workflow(s) and modify them.

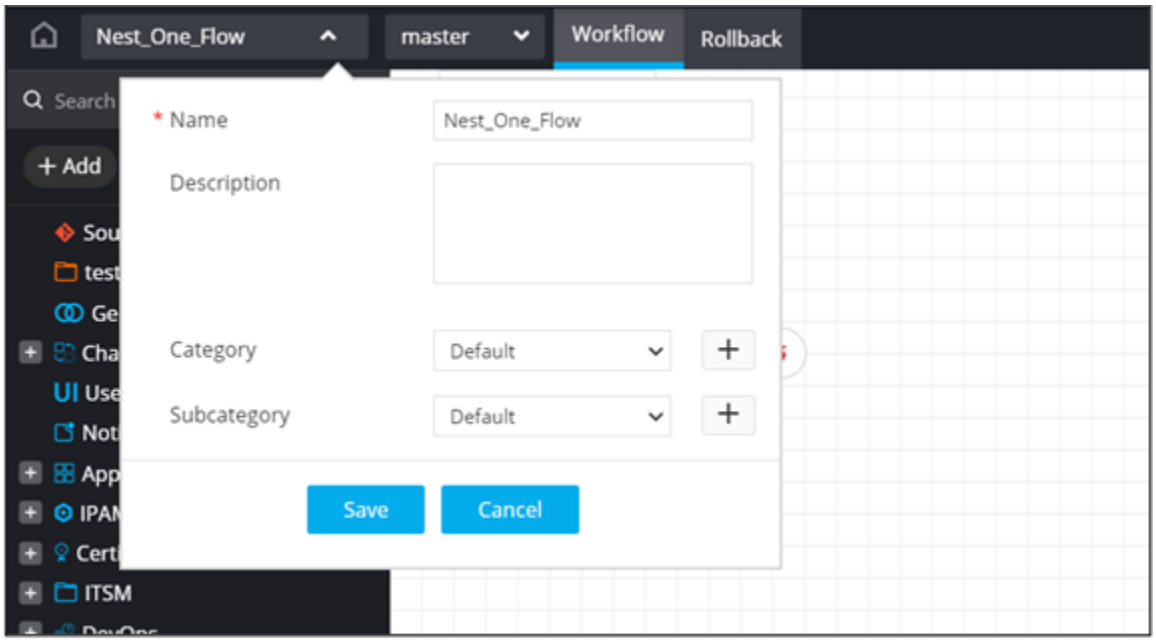
Note: Relevant global variables must be declared and referenced in order for nested workflows to work seamlessly.

- How to Add/Import a Nested Workflow?

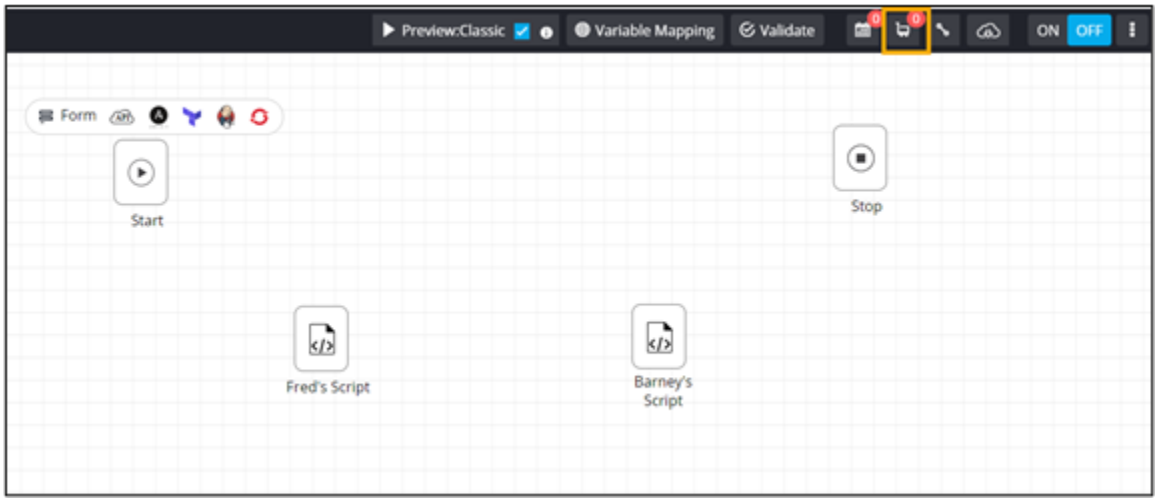
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

How to Add/Import a Nested Workflow?

1. Design a new workflow.

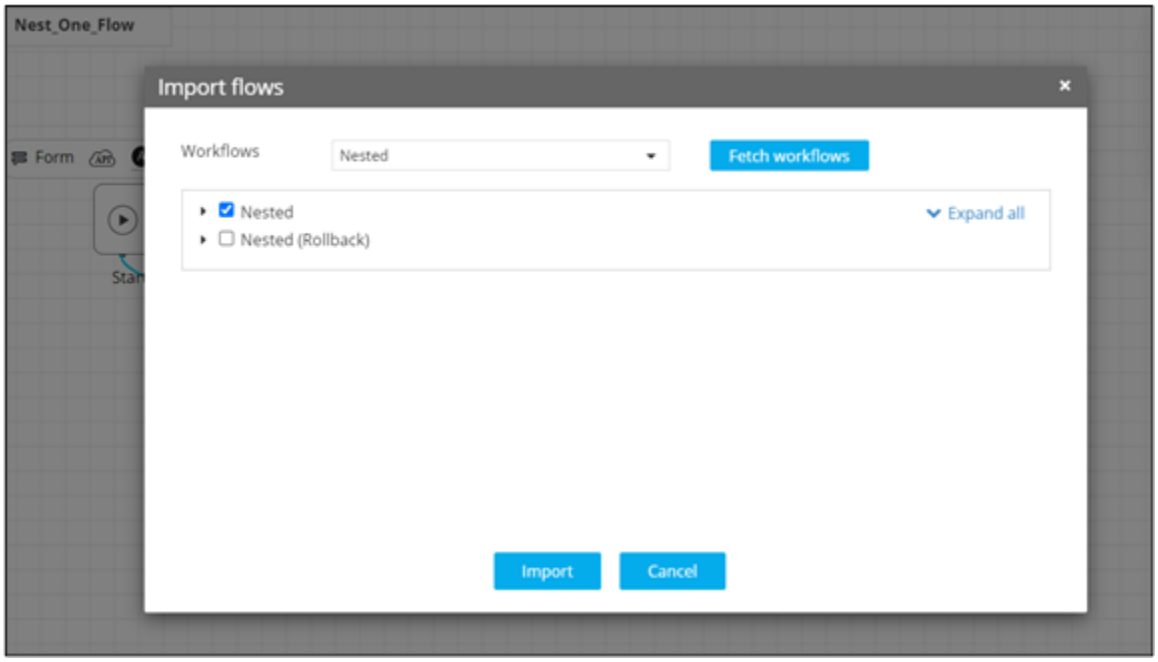


2. To add or import a nested workflow, click .

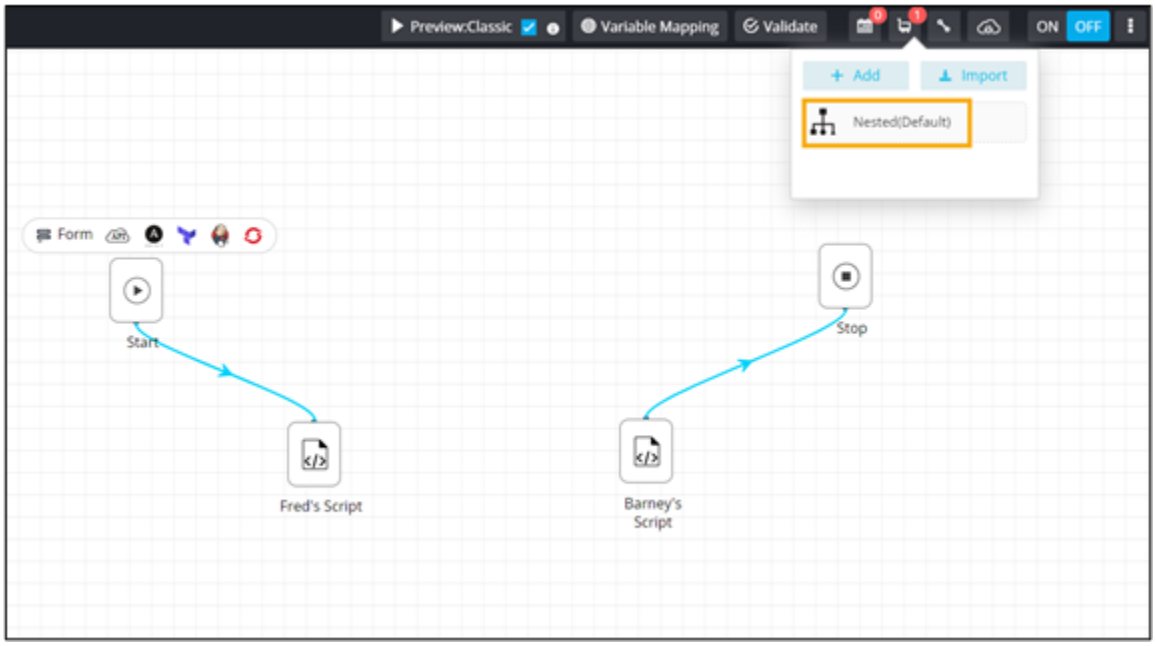


3. To define a new subflow, click **Add** or to reuse an existing workflow, click **Import**.
4. In the **Import flows** window, select the workflow to be reused or nested.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

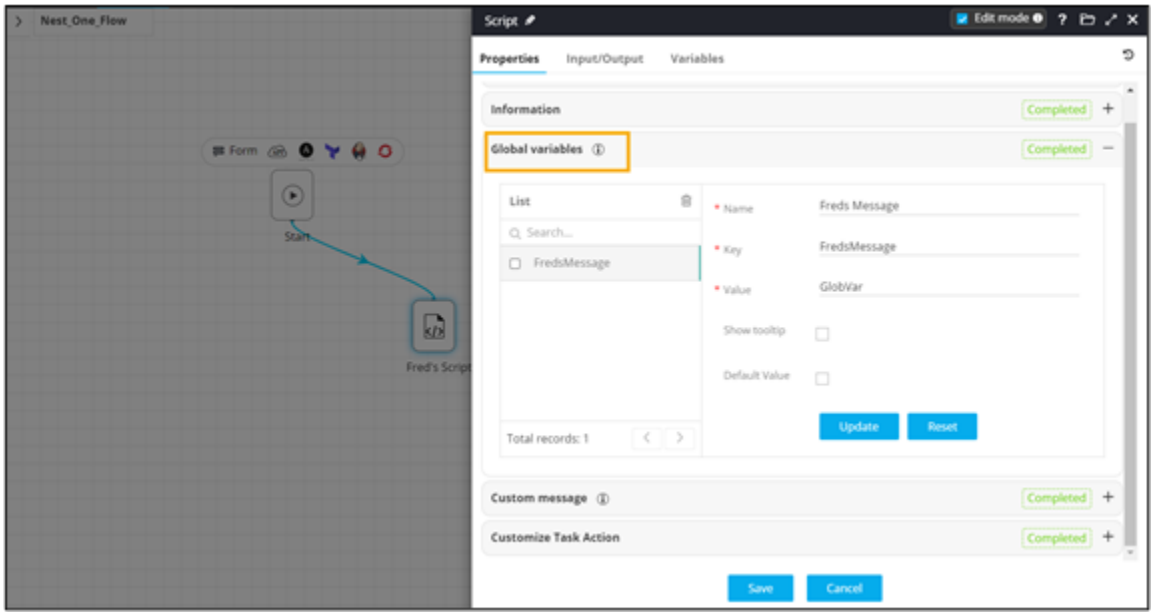


5. Drag and drop the workflow to be nested into the workspace.

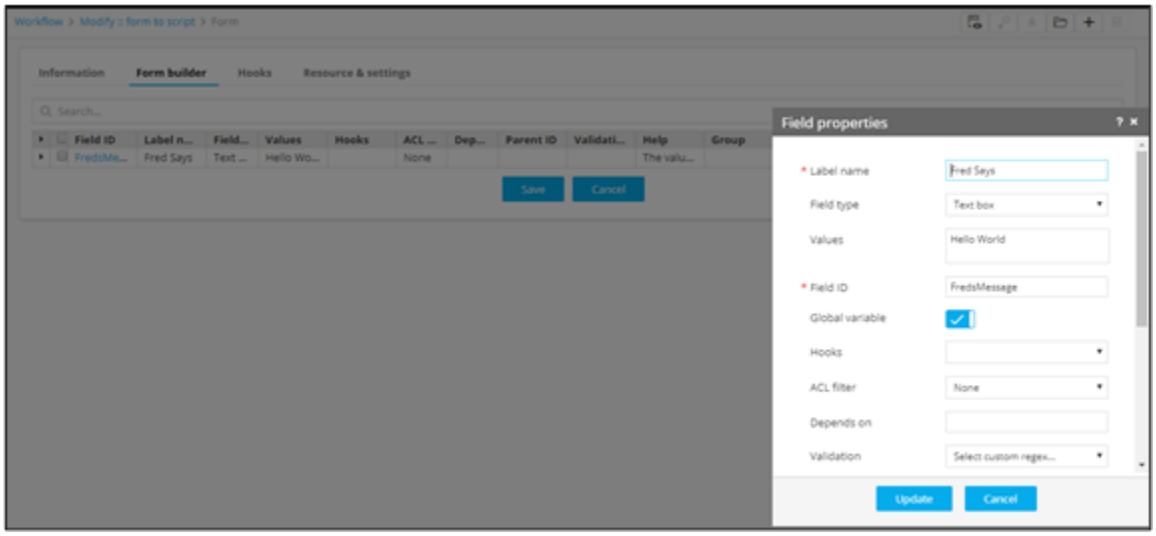


6. In the **Script** task window, under **Properties**, in the **Global variables** section, define the global variables in order to pass the values between task(s) in the existing flow and the nested flow.
   For example, The value ‘Fred says: Hello world’ from the script is declared as a global variable and used as an input into the form task of the nested flow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

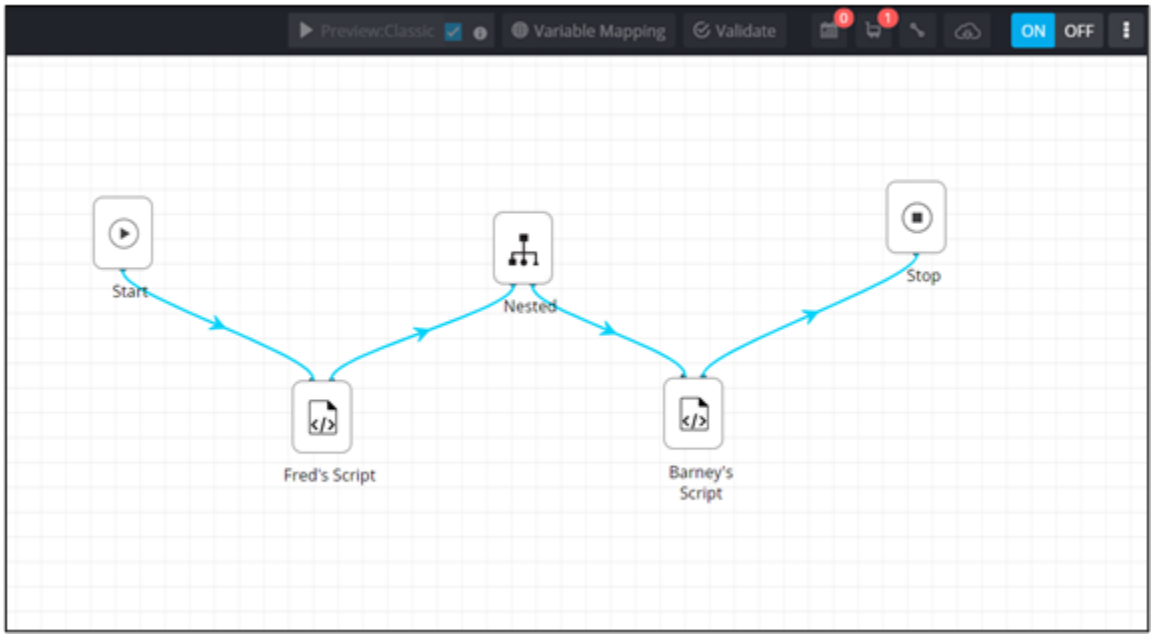


7. Refer the Global Variable key into one or more task(s) of the nested flow.

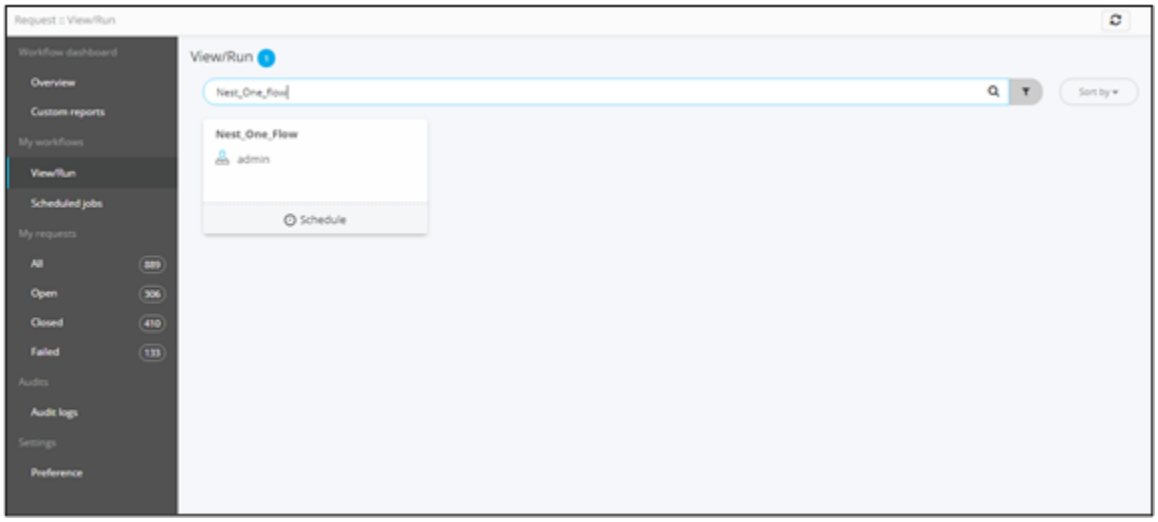


8. Connect the tasks and enable the workflow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

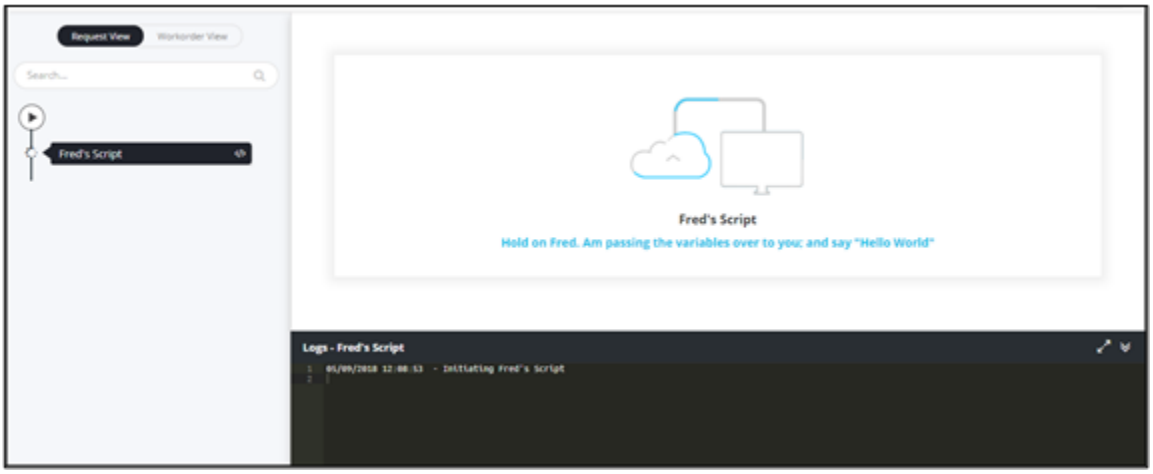


9. Trigger the workflow from the Request :: View/Run page.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

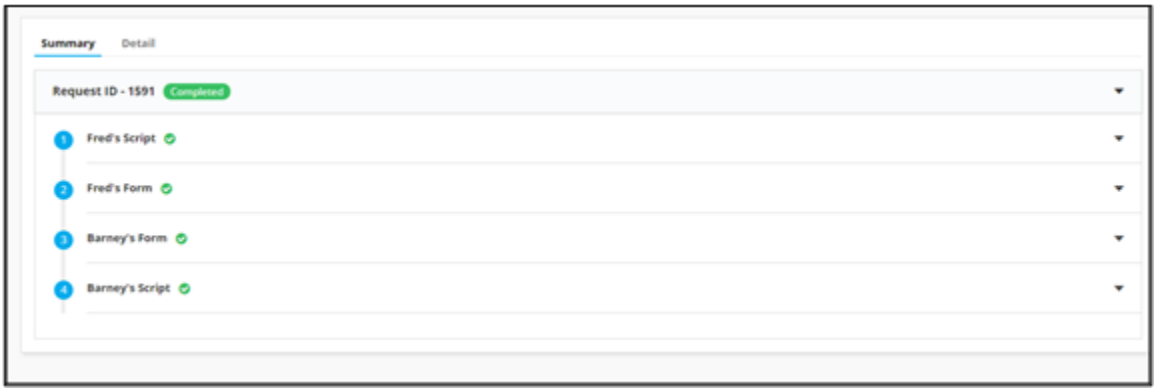
• Variable values being passed from one task to another task within a nested flow.



• The value being passed into the destination task where it is referenced.

• Summary of the workflow tasks executed.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Workflow Options

This section talks about the different features available in the Workflow Studio.

- Provision to add/remove versions of a workflow
- Provision to customize workflow task connectors
- Provision to modify the workflow alignment
- Provision to view the task history within a workflow
- Provision to validate a workflow
- Provision to switch between edit mode and citizen mode
- Provision to preview a workflow in classic or wizard mode
- Provision to configure access control for user(s)
- Version Control
- Preview
- Customizable Connectors
- Workflow Alignment
- History
- Workflow Settings
- Validating a Workflow
- Switching between Edit mode/Citizen mode

Version Control

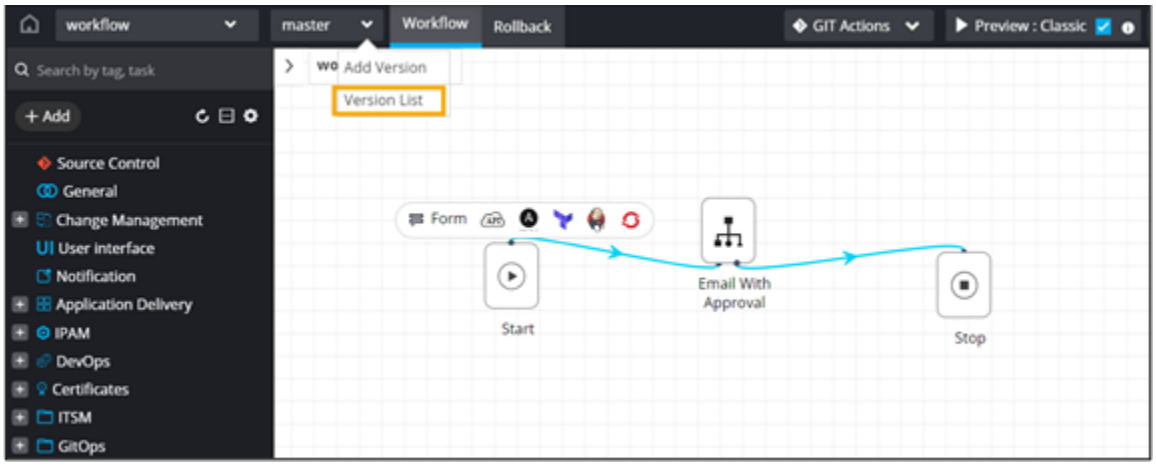
You can design and have multiple versions of a workflow and enable one master version.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

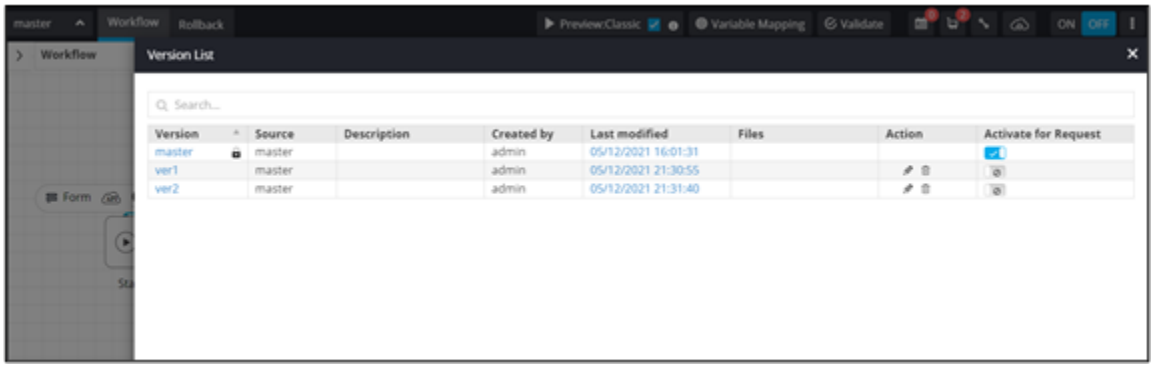
- Provision to version control the workflows.
- Provision to upload and audit requirements/artifacts against each version.
- Version control allows for tracking up to ten versions of the workflow.
- Provision to update, modify version.
- Provision to audit changes made to workflow tasks via ‘Task history’

To see the version list of a workflow:

1. Open an existing workflow.
2. From the master dropdown menu,select **Version List**.

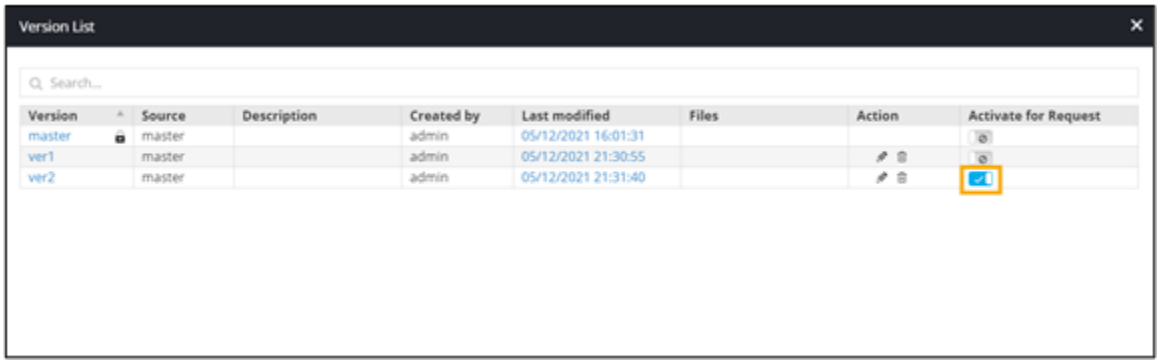


The **Version List** window displays the list of all versions existing for the workflow.



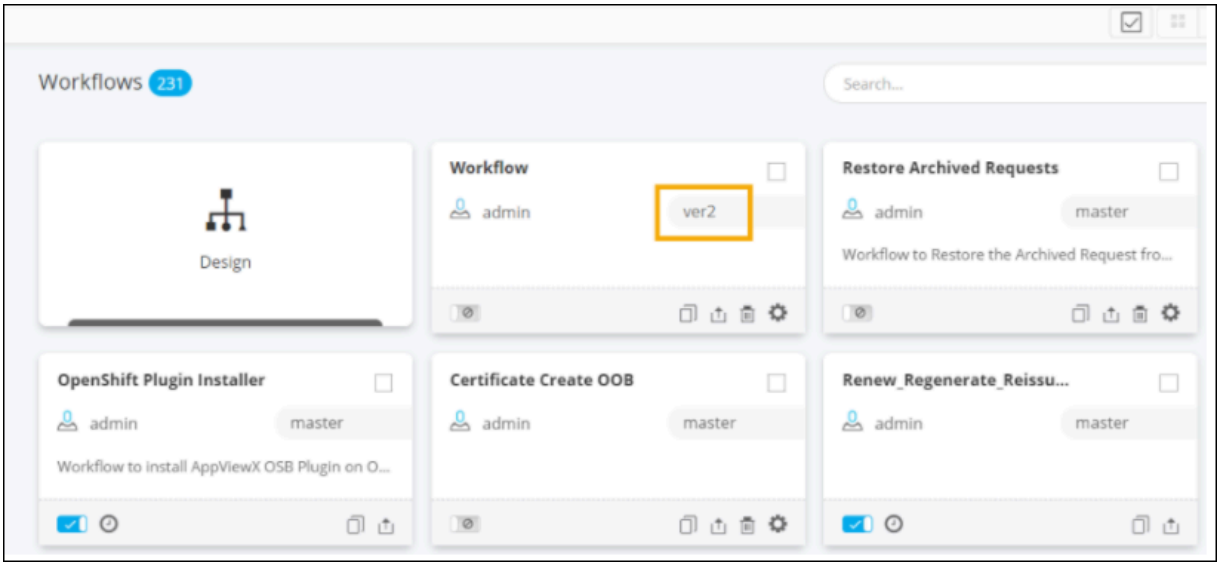
3. To enable a particular workflow version, turn on the toggle.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Note: Only one version can be enabled at a time.

The enabled version is added to the **Workflow** inventory page.

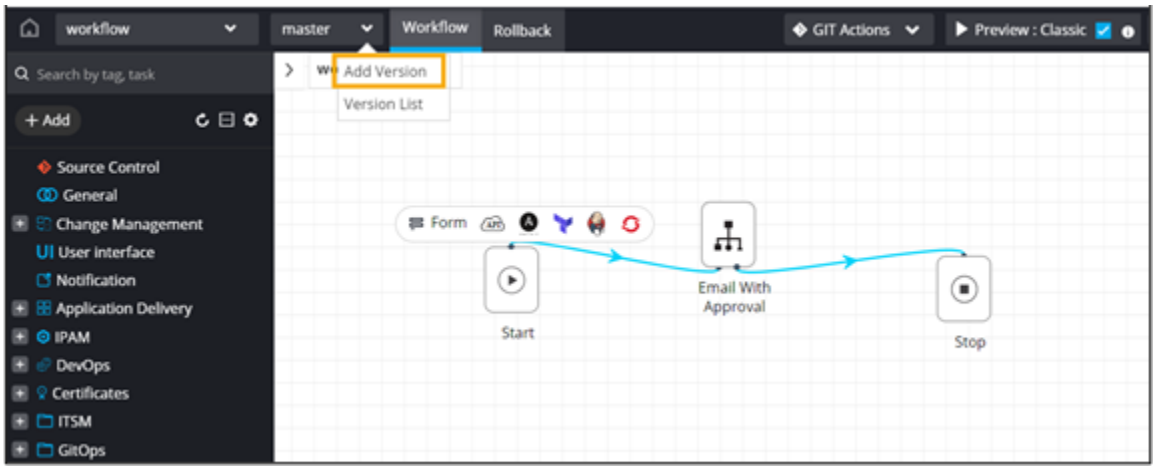


- Adding a Version

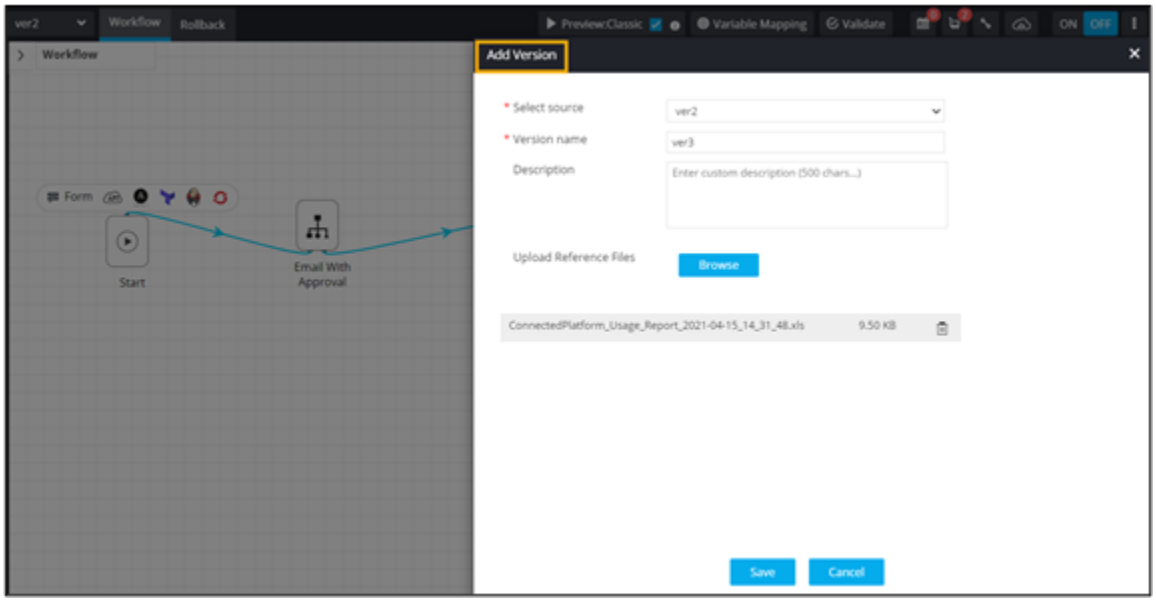
Adding a Version

1. Design a workflow.
2. From the master dropdown menu, select **Add Version**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

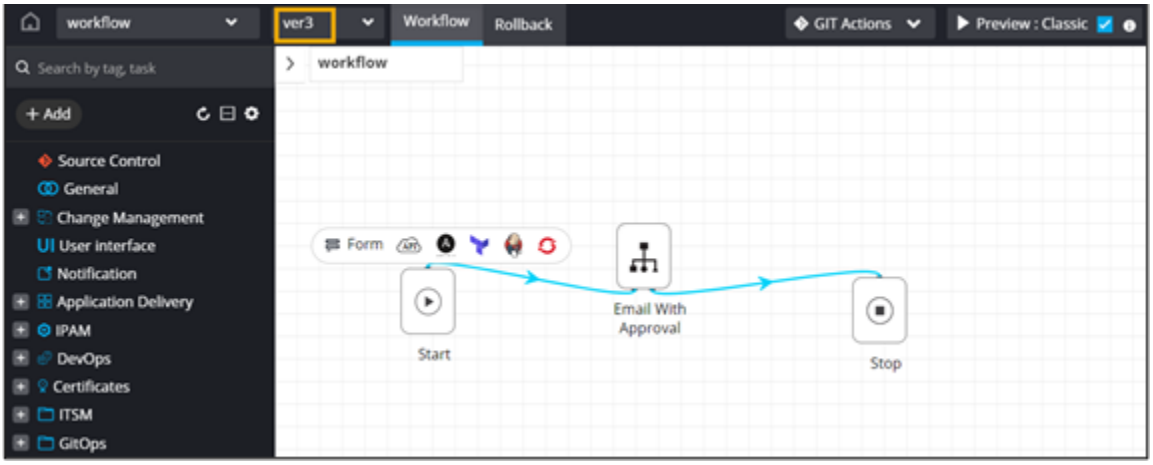


3. In the **Add Version** window, add the version details and click **Save**.

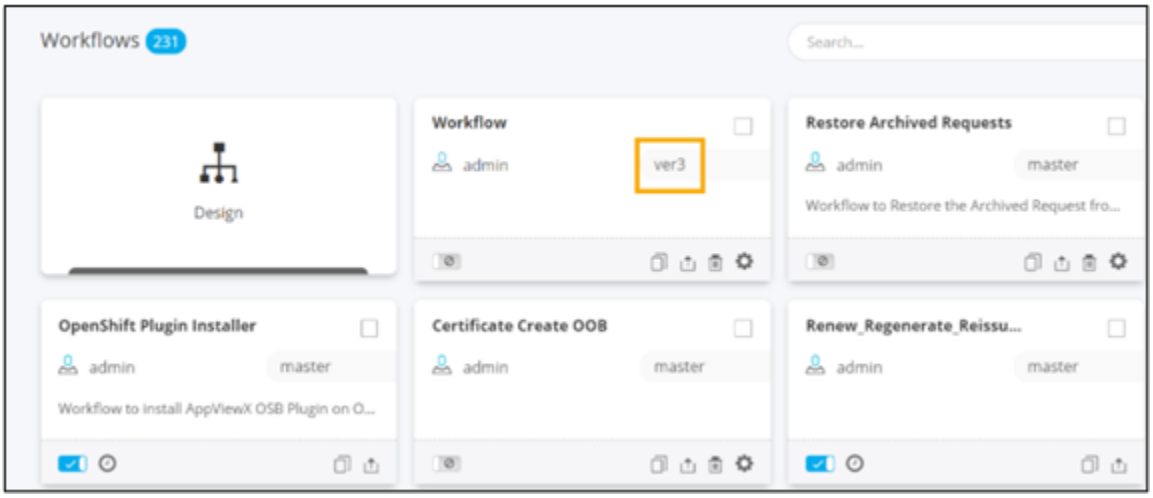


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- The newly added workflow version is activated.



- Workflow version displayed on the Workflow Inventory page.



Preview

You can perform a quick preview of the workflow to validate the outcome before triggering the workflow request. Preview action cannot be performed on enabled workflows. Two preview modes are available:

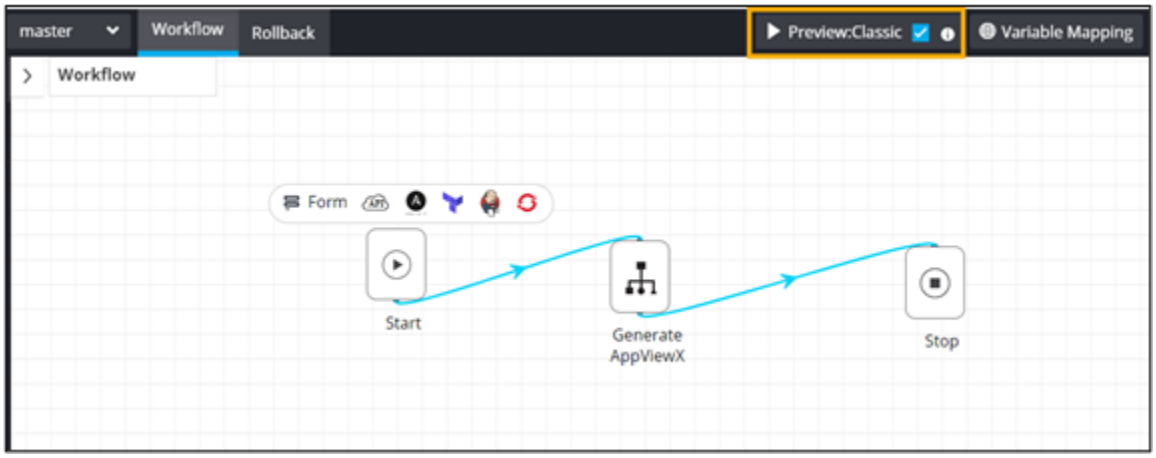
- **Classic**: This is the default Preview option
- **Wizard**: This option can be used when you want to validate individual workflow tasks without enabling the workflow. A pseudo form is generated for stage-wise execution of tasks without triggering the workflow. A pseudo form is only supported for the following tasks:
  - Script
  - REST

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- REST (I)
- Schedule
- Email
- Slack

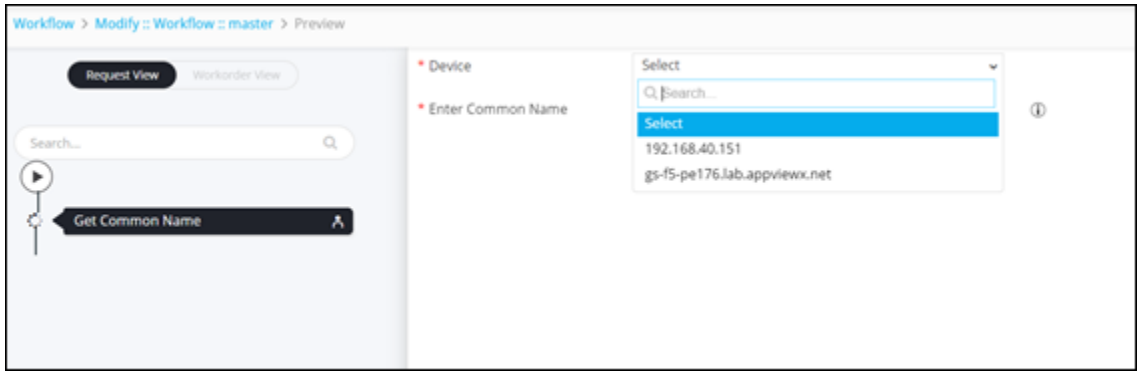
To preview a workflow in Classic mode:

1. Open a workflow.
2. Select the **Preview:Classic** checkbox.

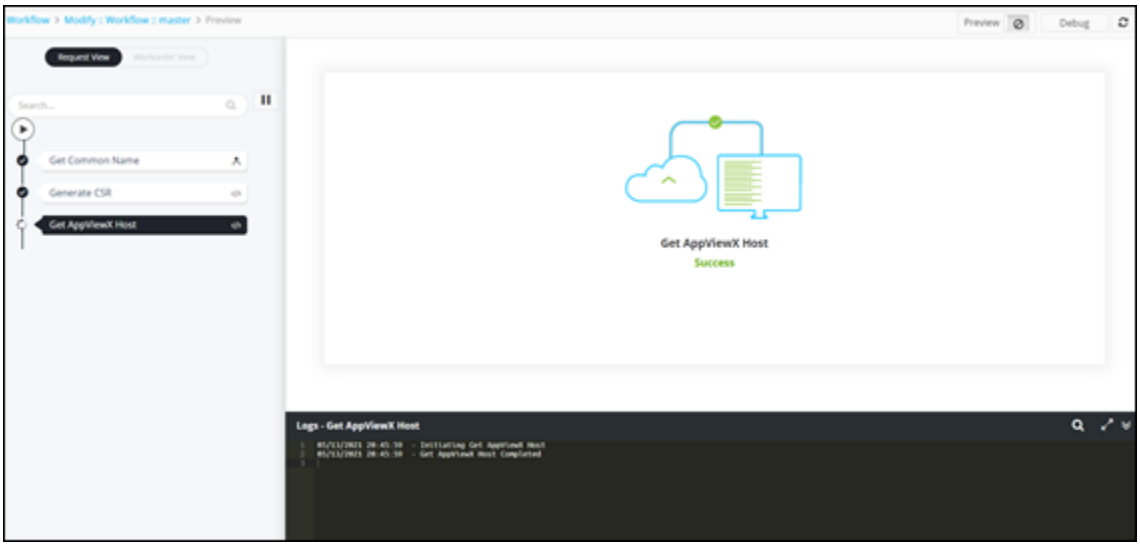


3. Click **Preview**.

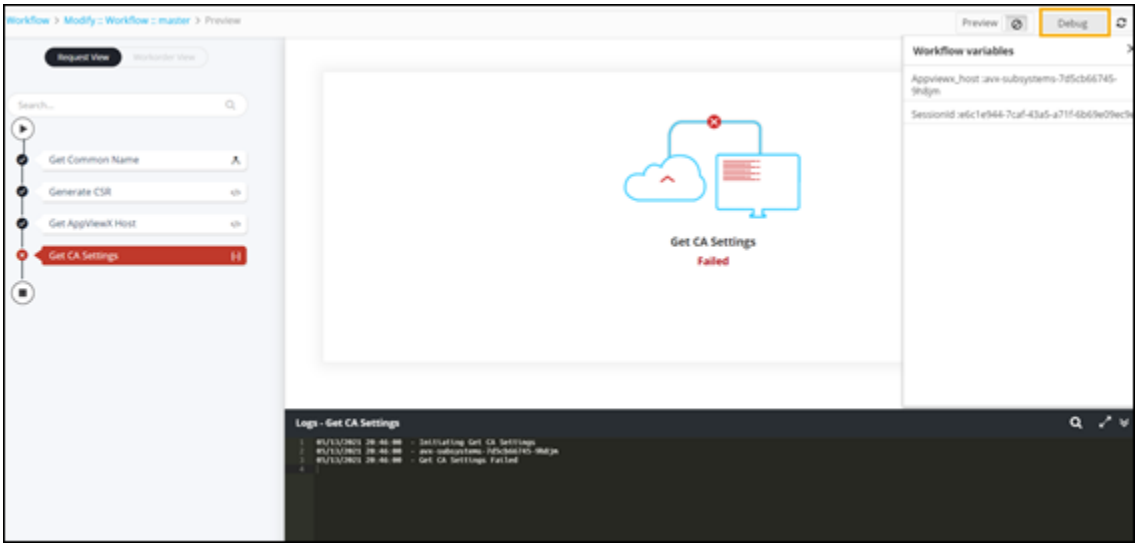
   Workflow preview is displayed.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

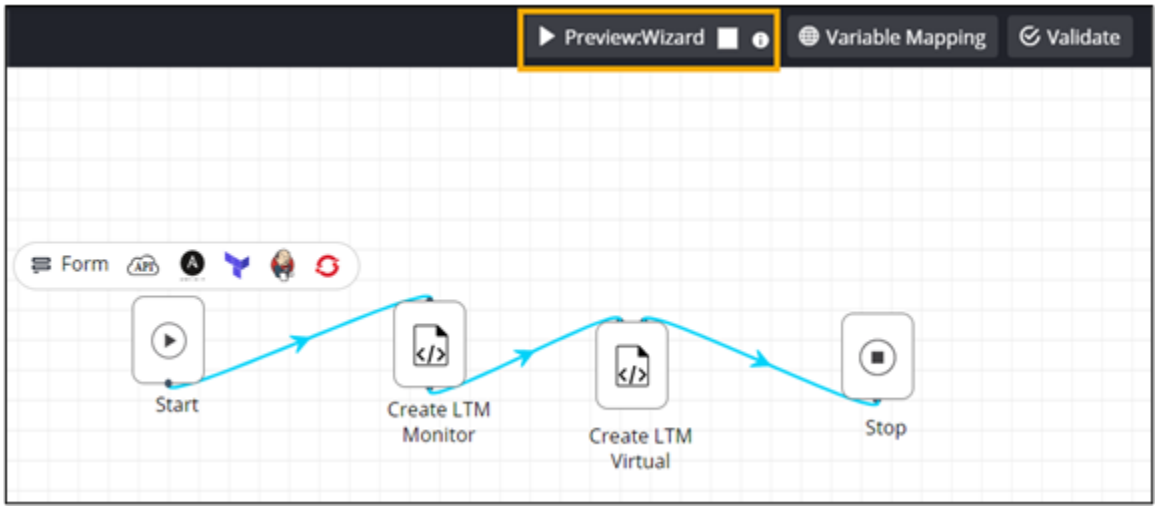


4. Click **Debug** for troubleshooting.



5. To perform Preview in wizard mode, clear the **Preview:Wizard** checkbox.

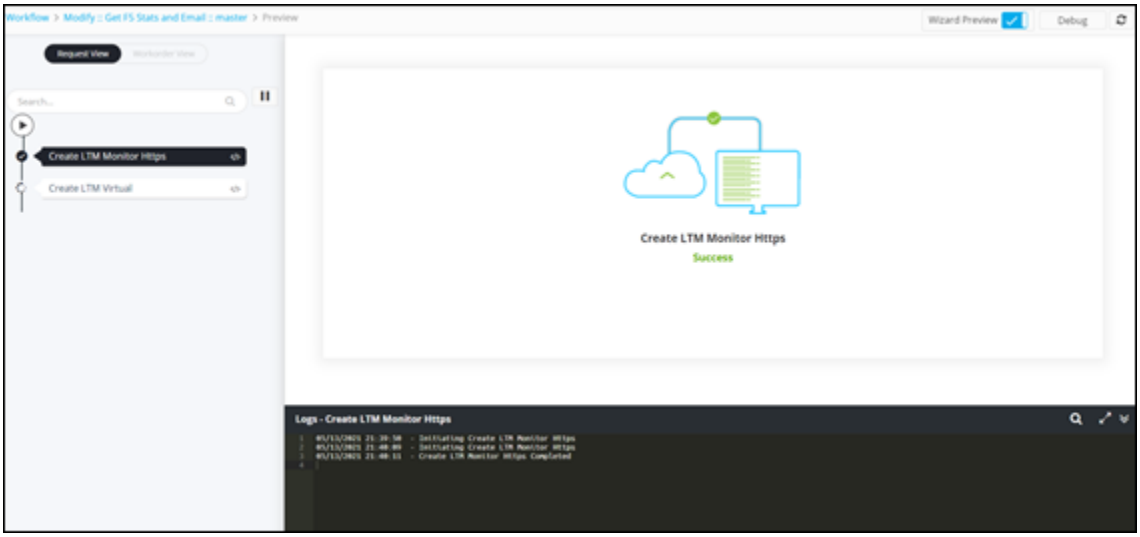
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



6. Click **Preview**.

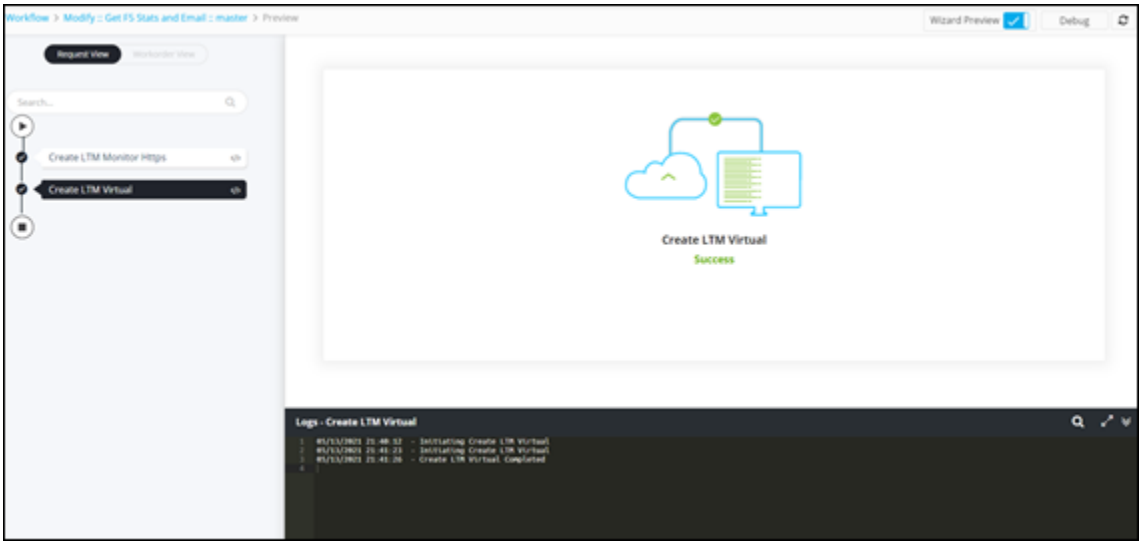
   A pseudo form is generated with the input variables in the Script task.

7. Enter the field information to execute the task.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

The second task executes automatically once the first task is successful.



Customizable Connectors

Connectors or links allow you to connect one task to the other in order to stitch the workflow. The connector option allows you to customize the connectors depending on the nature of the workflow.

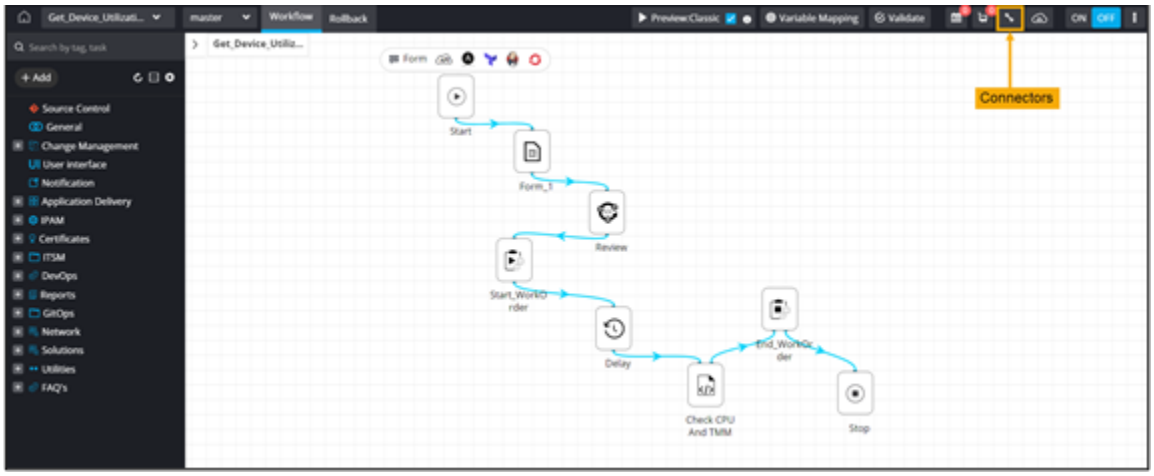
The following types of connectors are available:

- Default
- Flow
- Curved
- Straight

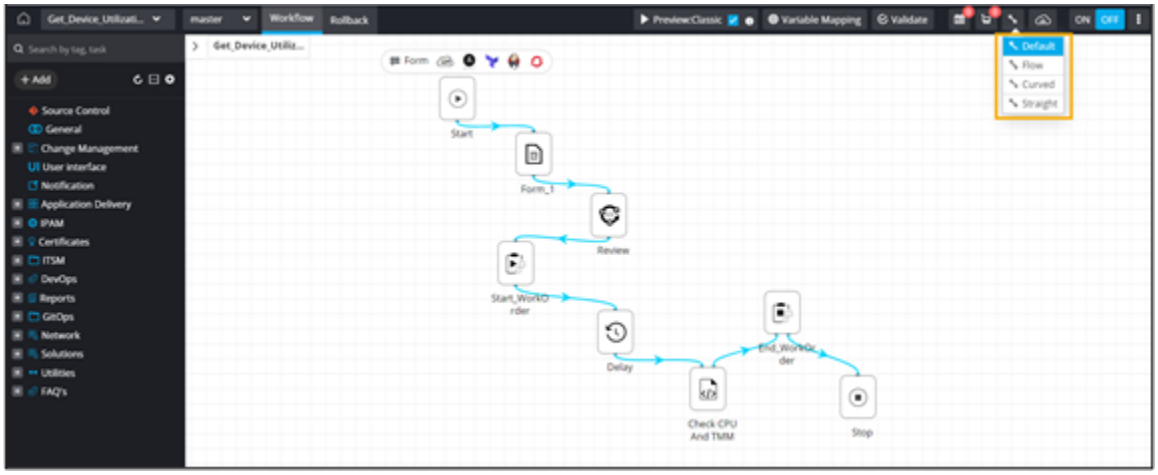
To customize connectors:

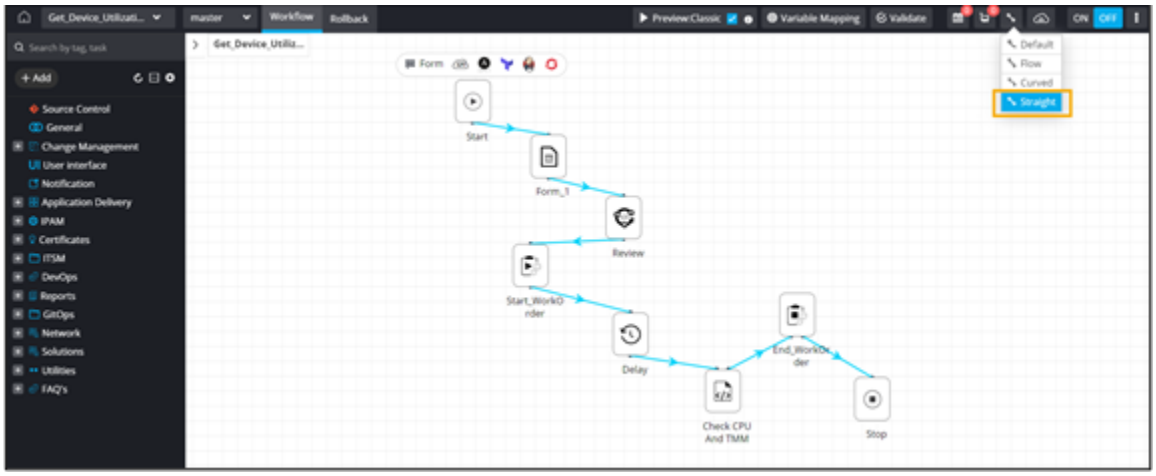
1. Open an existing workflow.
2. To view the different connector options, click .

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

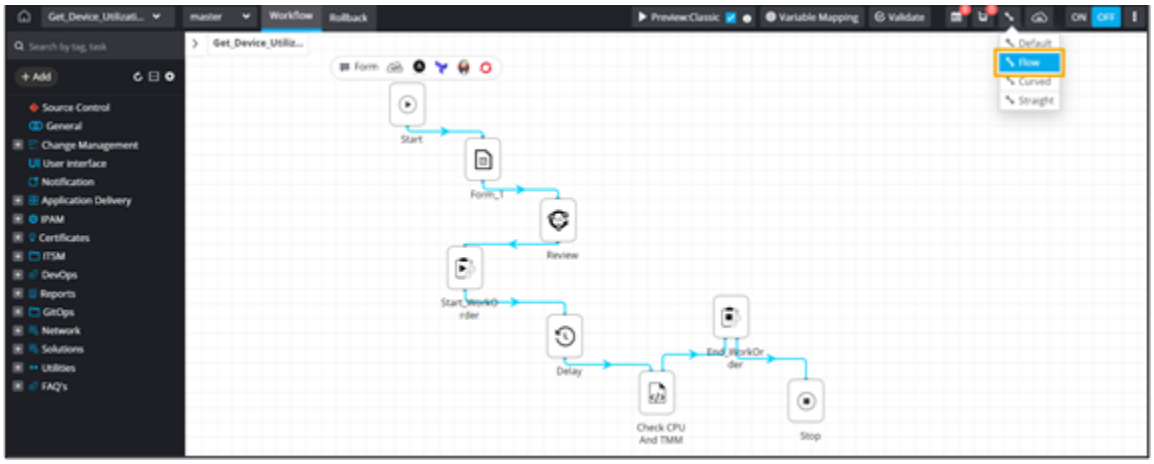


3. Select the connector from the available options.





Copyright © 2020 AppViewX, Inc. All Rights Reserved.



- Decisions via Connectors/Links

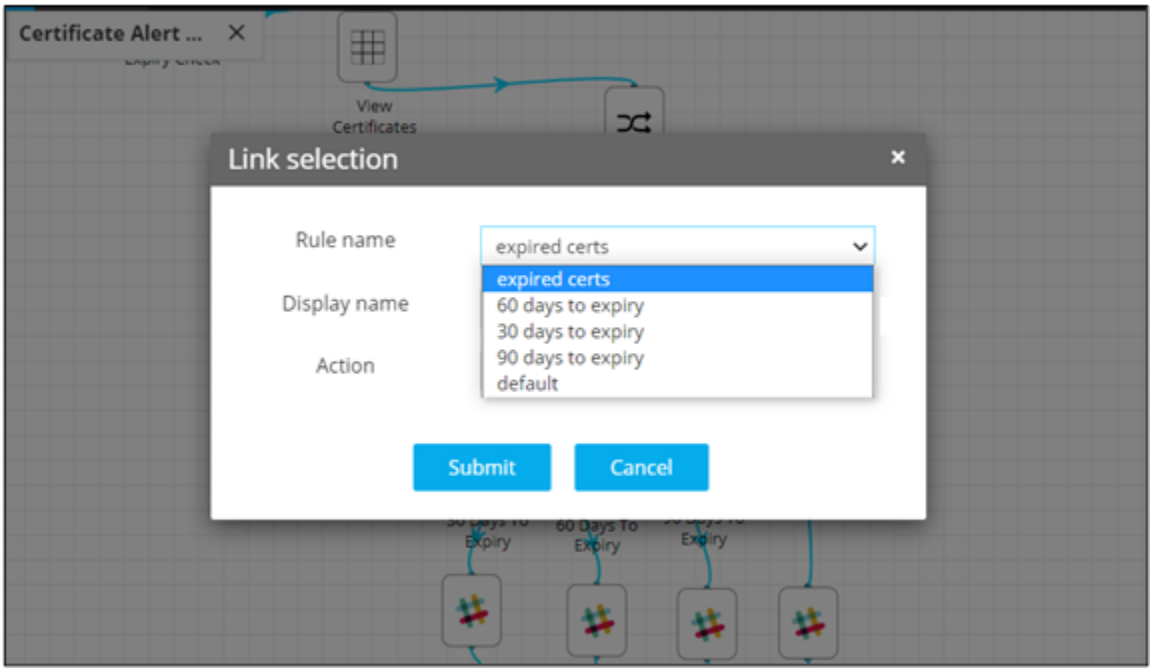
Decisions via Connectors/Links

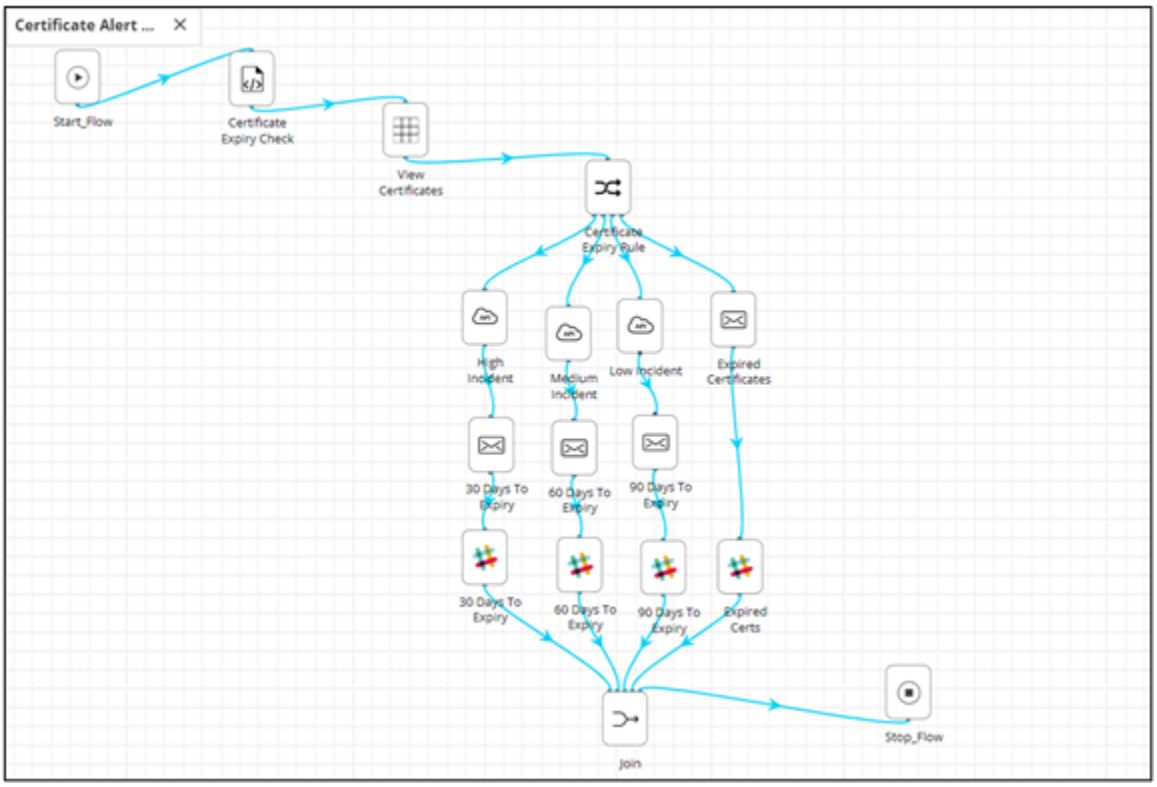
Links allows users to specify decisions (success, failover) based on which a workflow task can be routed intelligently. These links can be updated or deleted as per requirement.

Decisions via links can be used when constructing a workflow with task(s) that allow one or more decisions to be taken.

- **Script**: Allows one or more routes to be taken
- **If/else**: Allows maximum of two decisions to be taken
- **Switch**: Allows more than two decisions to be taken

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



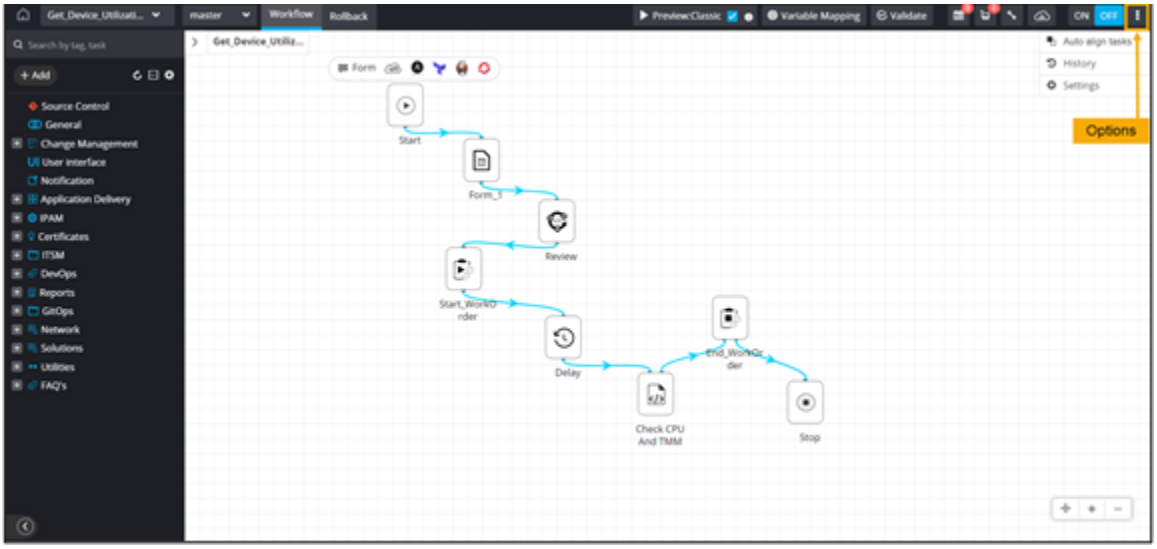


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

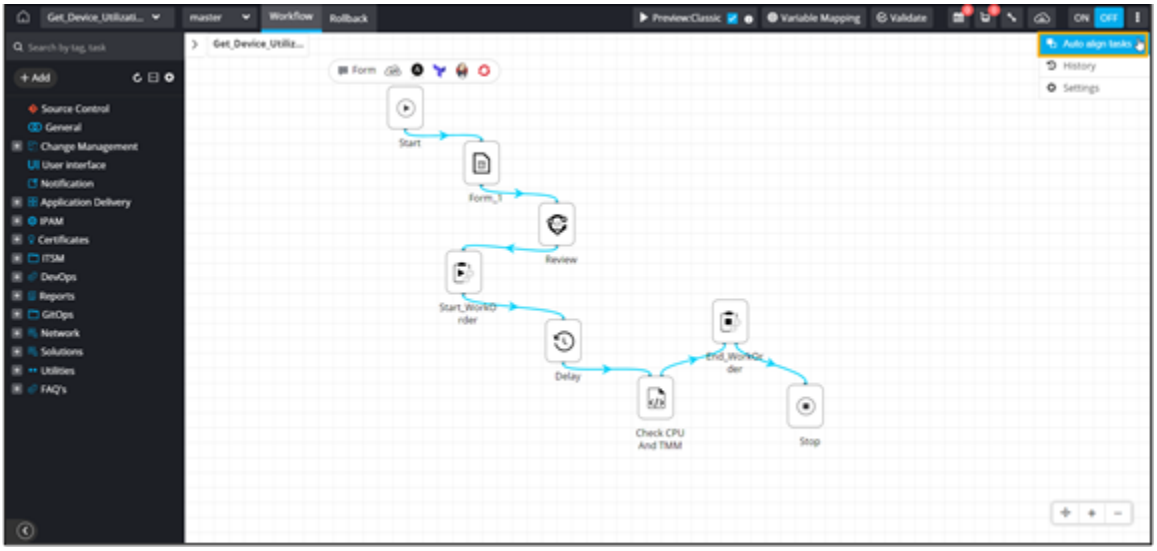
Workflow Alignment

You can align a workflow either vertically or horizontally.

1. To see alignment options, open a workflow and click in the right upper corner of the screen.

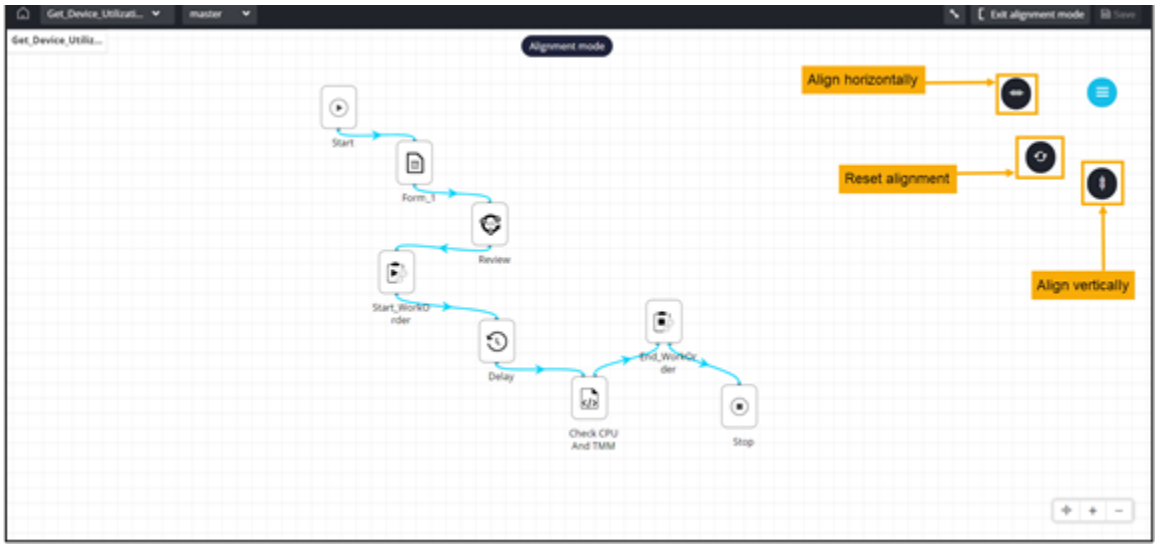


2. From the options displayed, select **Auto align tasks**.



3. Click **Yes** in the **Confirmation** pop-up window.
4. Select the alignment for the workflow from the options available.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



• Horizontal alignment

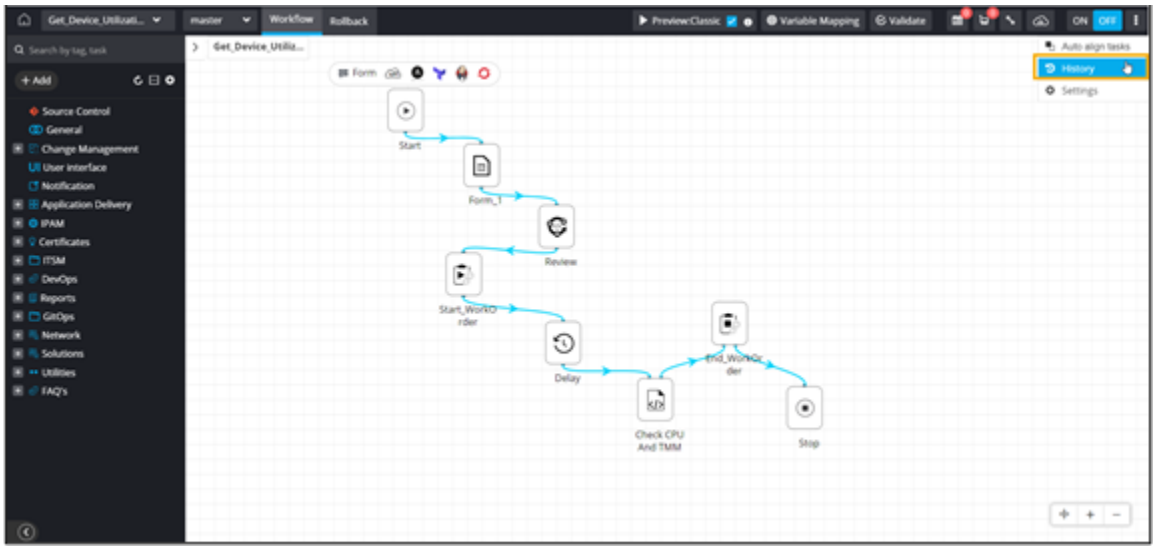
• Vertical alignment

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

History

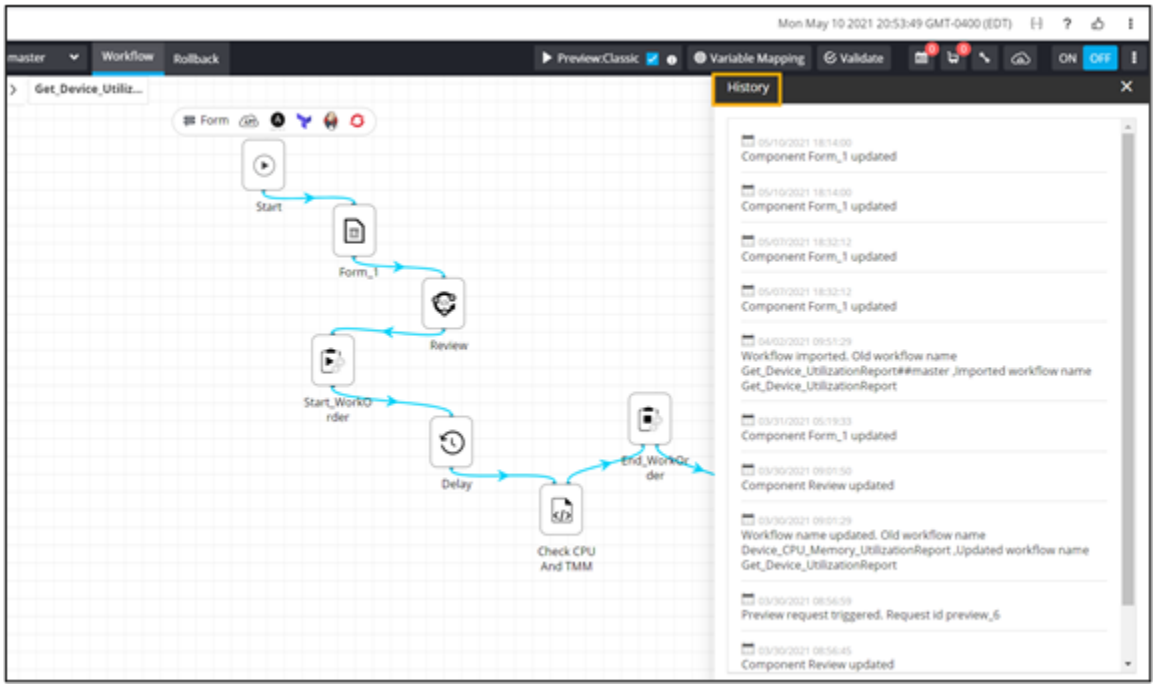
You can track the complete audit of changes made against a workflow.

1. Open a workflow and click in the right upper corner of the screen.
2. From the options displayed, select **History**.



The **History** window opens displaying the complete history of actions for this workflow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Workflow Settings

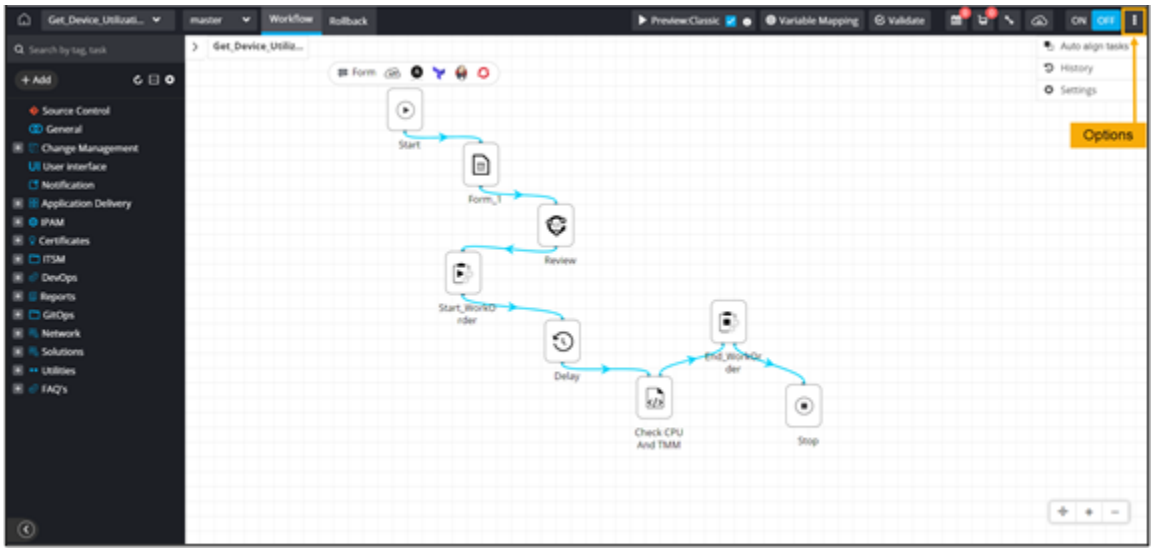
On creating a new workflow, the workflow settings option allows users to perform the following actions: Provision to generate a unique workflow request ID for every workflow triggered.

- Role based access control (RBAC) to assign workflows.
- Role based access control (RBAC) to assign workflow requests.
- Provision to exclude specific workflow task(s) for calculating Request status.
- Provides an option to opt-out specific task(s) from being considered for calculating the overall workflow request status.

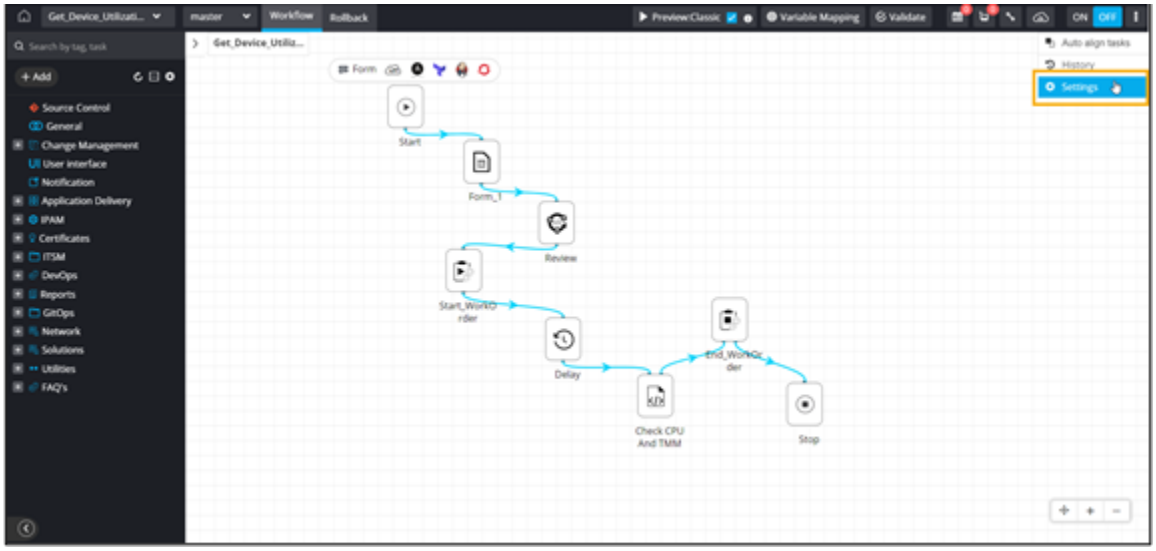
To access workflow settings:

1. Open an existing workflow.
2. Click in the right upper corner of the screen.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



3. From the options displayed, select **Settings**.

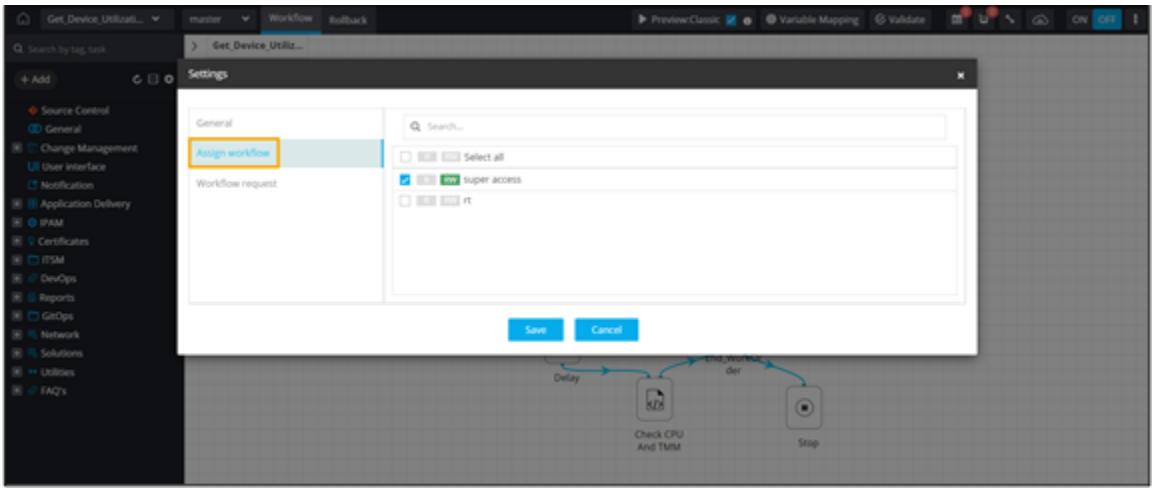


- Assign Workflow
- General
- Workflow Request

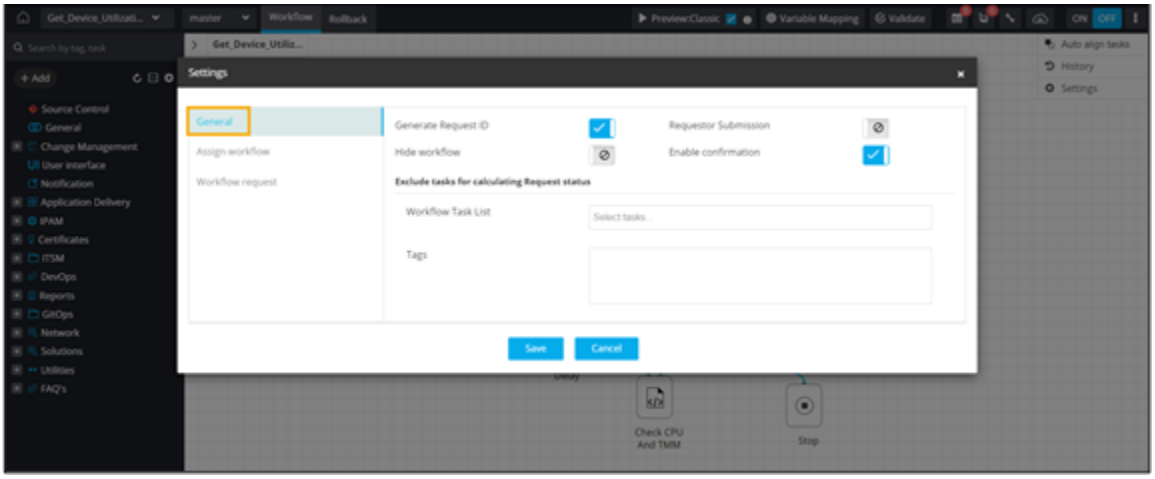
Assign Workflow

Under Workflow Settings, you can set up role based control (RBAC) for workflows and assign workflows to specific users or roles.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



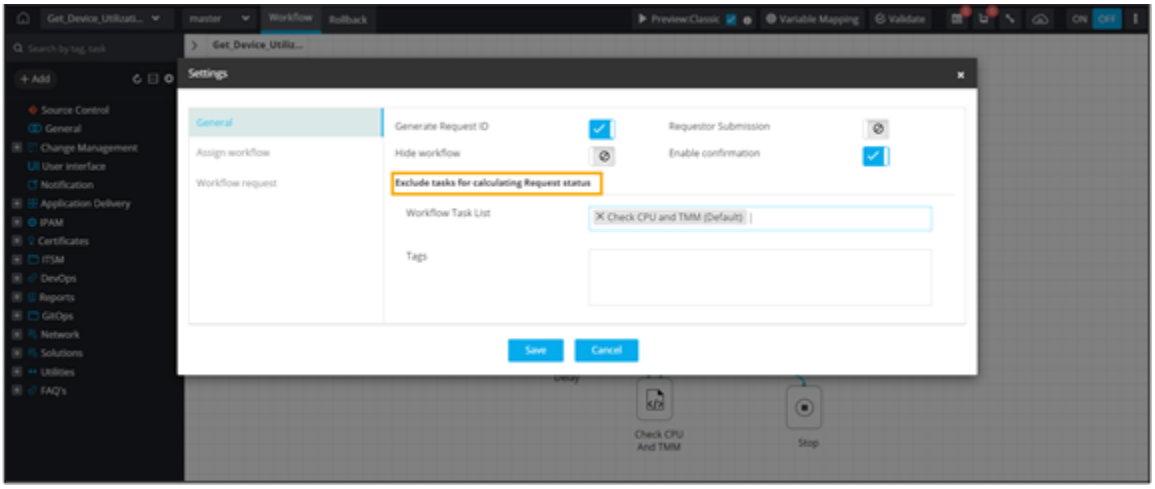
General



Under Workflow Settings, the following options are available under the General section:

- **Generate Request ID**: Allows you to choose generating or not generating a request ID for a workflow
- **Requestor Submission**: Allows you to restrict/allow actions/approvals for the user who requested the task
- **Hide workflow**: Allows you to keep the workflow hidden or visible
- **Enable confirmation**: Allows you to enable or disable the Confirmation popup window
- **Exclude tasks for calculating Request status**: Allows you to select workflow tasks that will be excluded when calculating request status

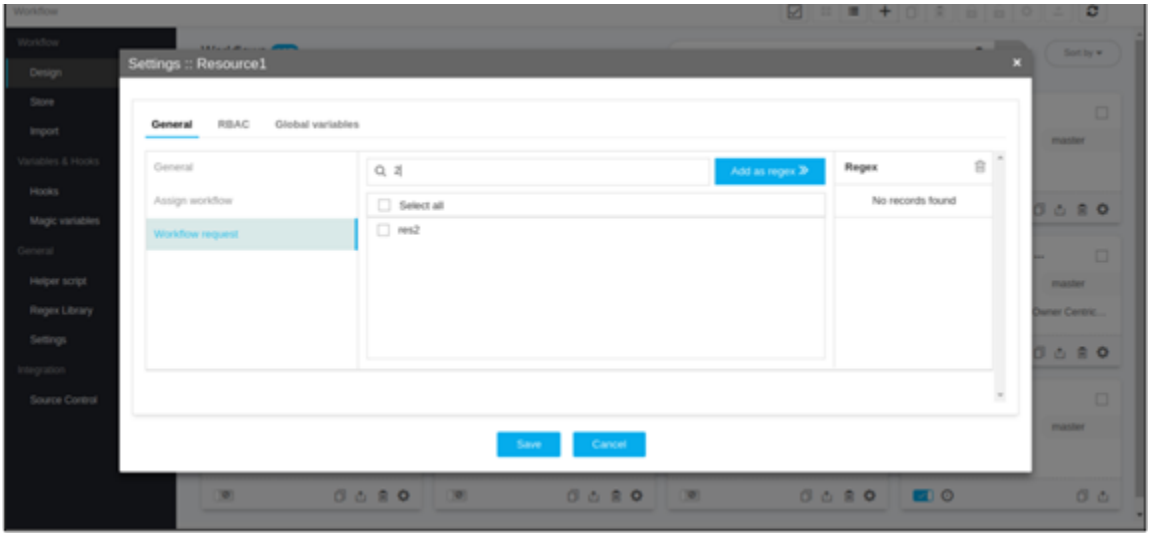
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Workflow Request

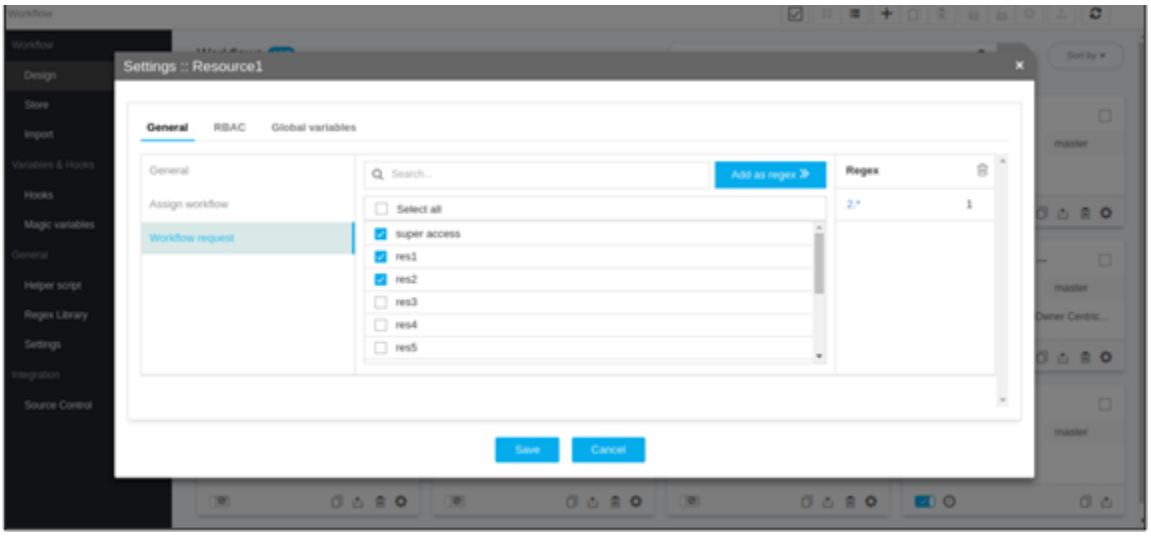
Under Workflow Settings, you can set up role based control (RBAC) for workflow requests.

- Provision to search for resources by entering values in the search field.

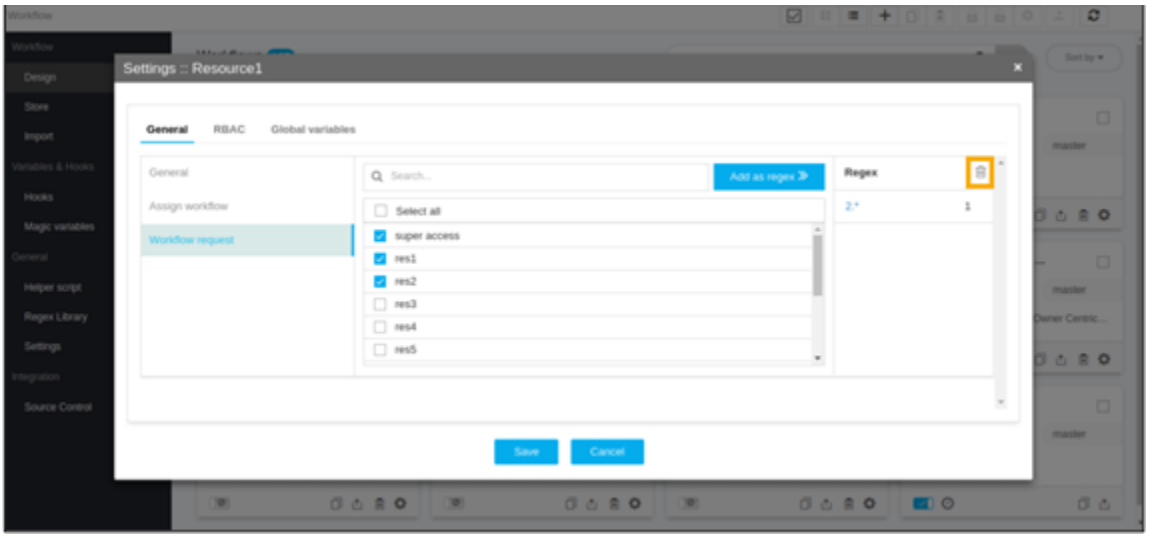


- Provision to Add a Regex and select the list of available resources using a regex pattern. For example, (.*) regex can be used to select the list of all available resources.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



- Provision to delete the regex.



Validating a Workflow

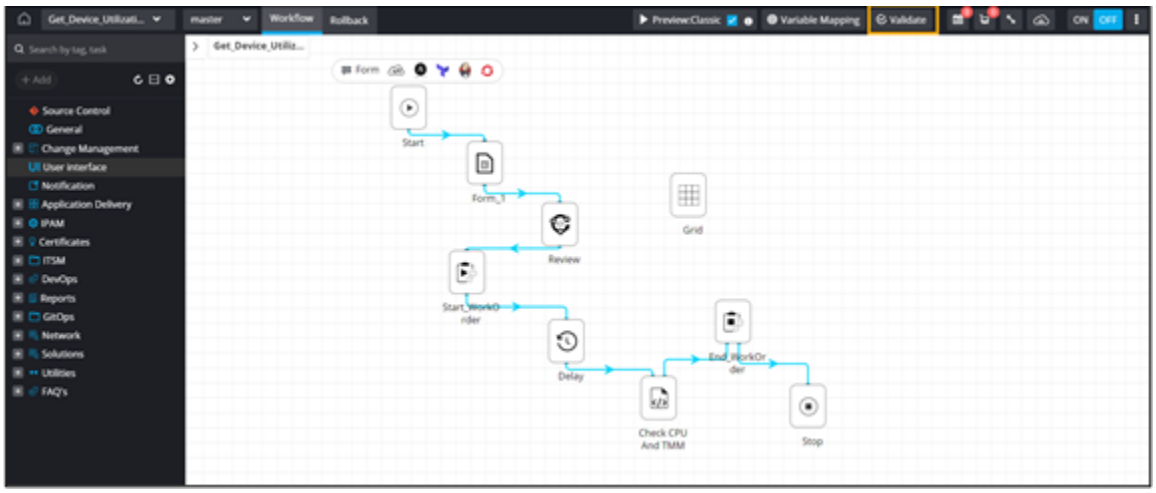
Once a workflow is designed, users can validate the workflow and its elements. Basic workflow validation includes:

- Tasks with no relevant connectors associated.
- Missing tasks which are dependent on other workflow tasks.
- Invalid links or connections.

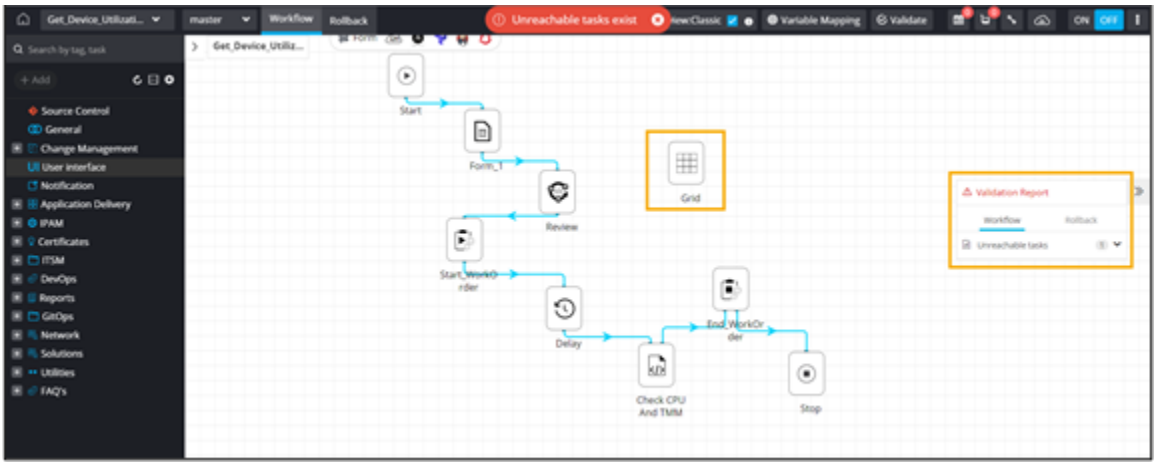
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

To validate a workflow:

1. Design a workflow.
2. Click **Validate**.

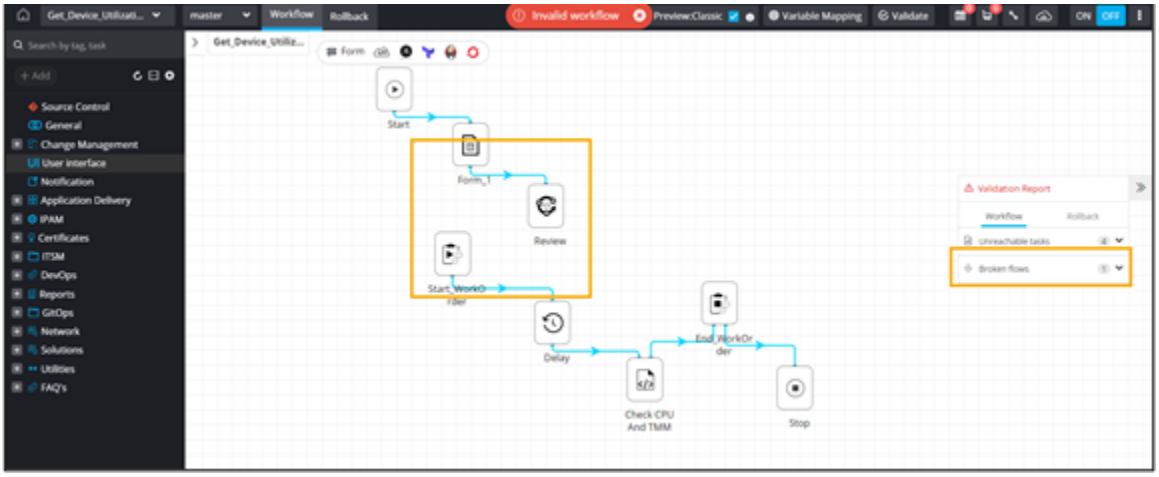


- Validation Report shows unreachable tasks.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

• Validation report shows broken flows.

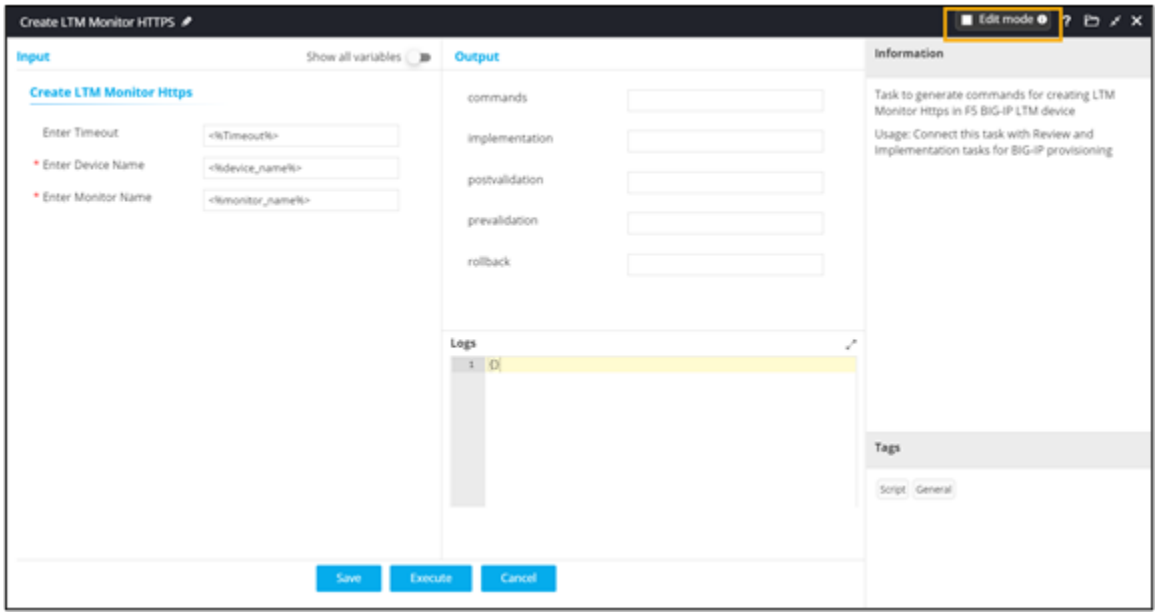


Switching between Edit mode/Citizen mode

Users can switch between Edit mode and Citizen mode when creating/updating tasks. All tasks are displayed in Edit mode by default.

1. Design a workflow with workflow tasks.
2. To switch to citizen mode, in the task window, clear the **Edit mode** checkbox.

   Variables referred in the script are listed under **Input**, while the values declared as Global Variables are listed under **Output**.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

3. To switch back to Edit mode, select the **Edit mode** checkbox.

Connecting tasks in Workflow Studio

This section shows how to connect the different tasks in the Visual Workflow Studio.

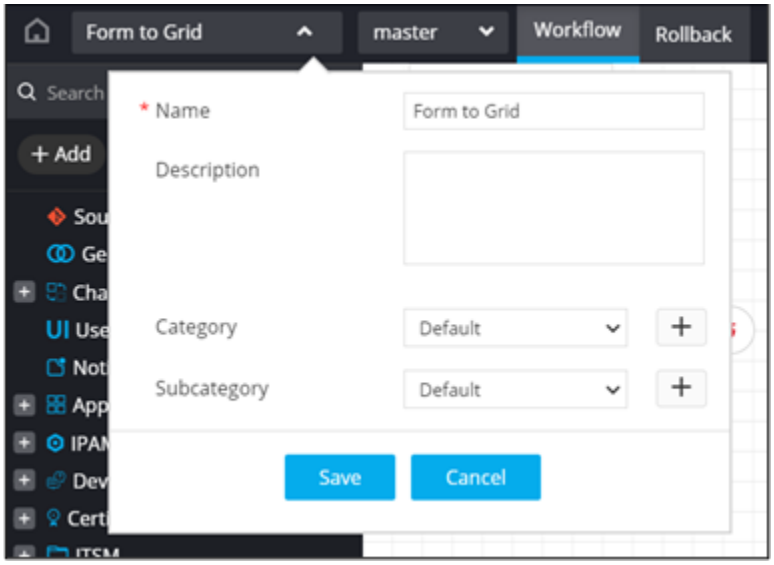
- How to connect Form to Grid
- How to connect Form to Grid to CSV
- How to connect Form to Decision to Form
- How to connect Script to Grid
- How to connect Script to Review
- How to connect Form to Create Ticket to Close Change Ticket
- How to connect Form to Delay to Email
- How to connect Form to Retry to Form
- How to connect Script to Bar or Pie Chart
- How to connect Script to Stacked Chart
- How to connect Form with Loop and Printing
- How to connect Form to Schedule to Script

How to connect Form to Grid

You can design a workflow to get inputs from a Form task and pass them into a Grid using global variables.

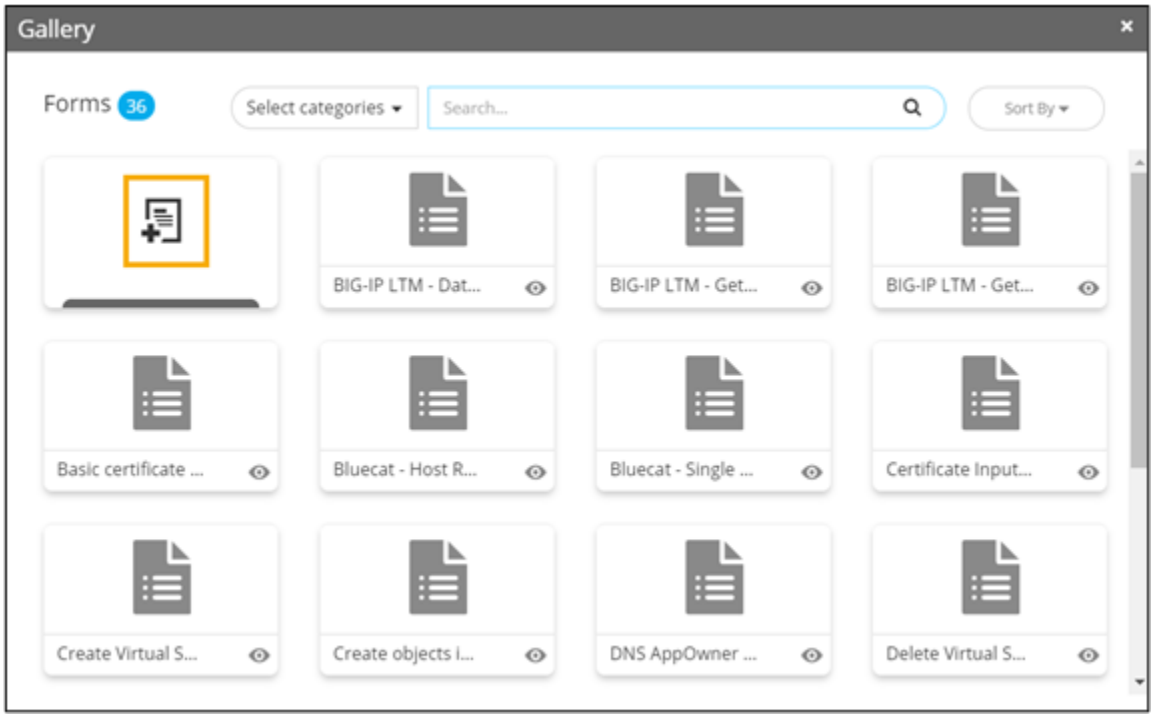
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. Design a new workflow.



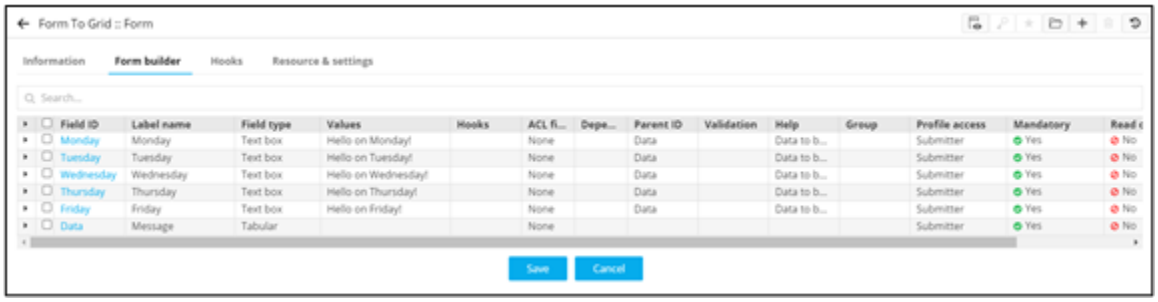
2. From the **User Interface** section, drag and drop the **Form** task.

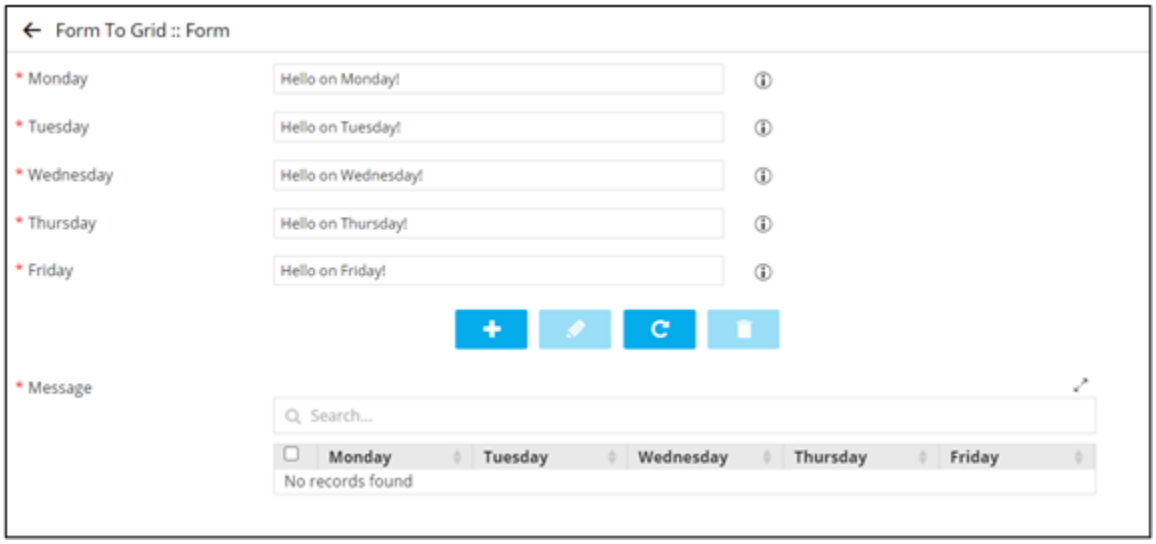
3. In the **Gallery** window, to design a new form, click .



4. Under the **Form builder** tab, define the form fields to get inputs from the user.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

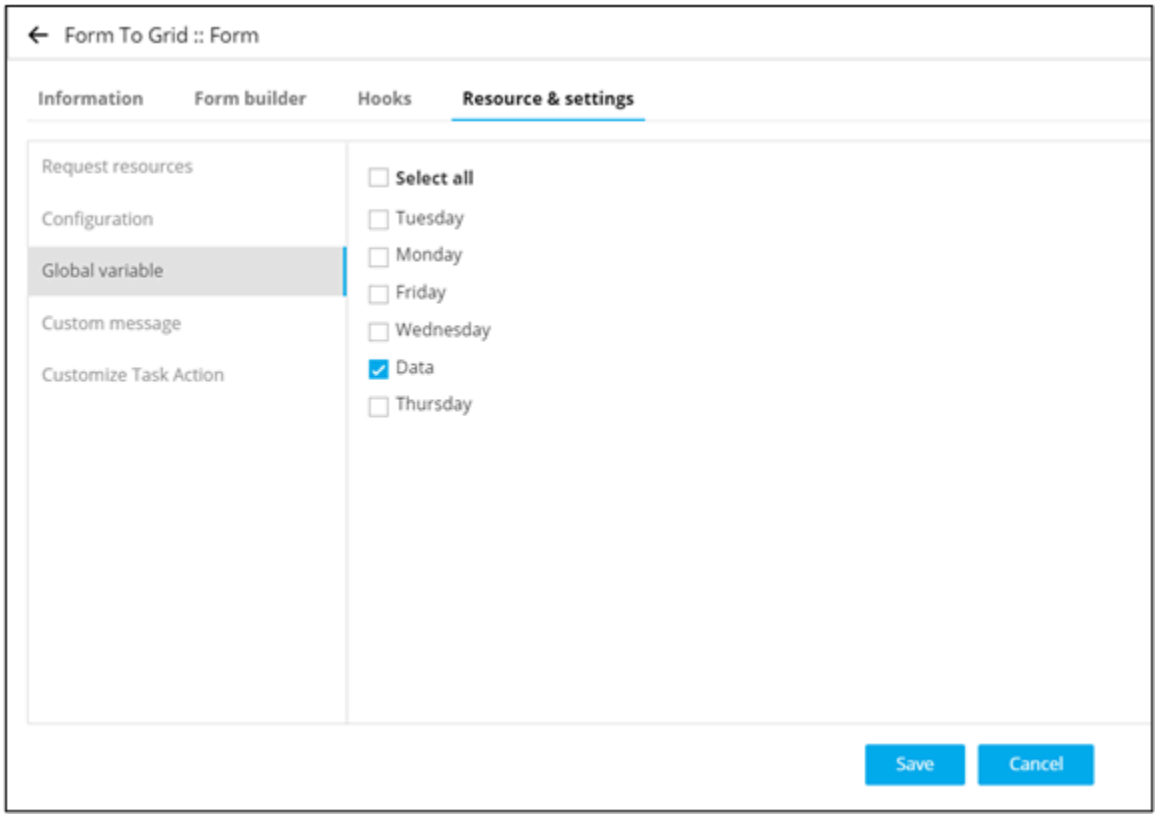




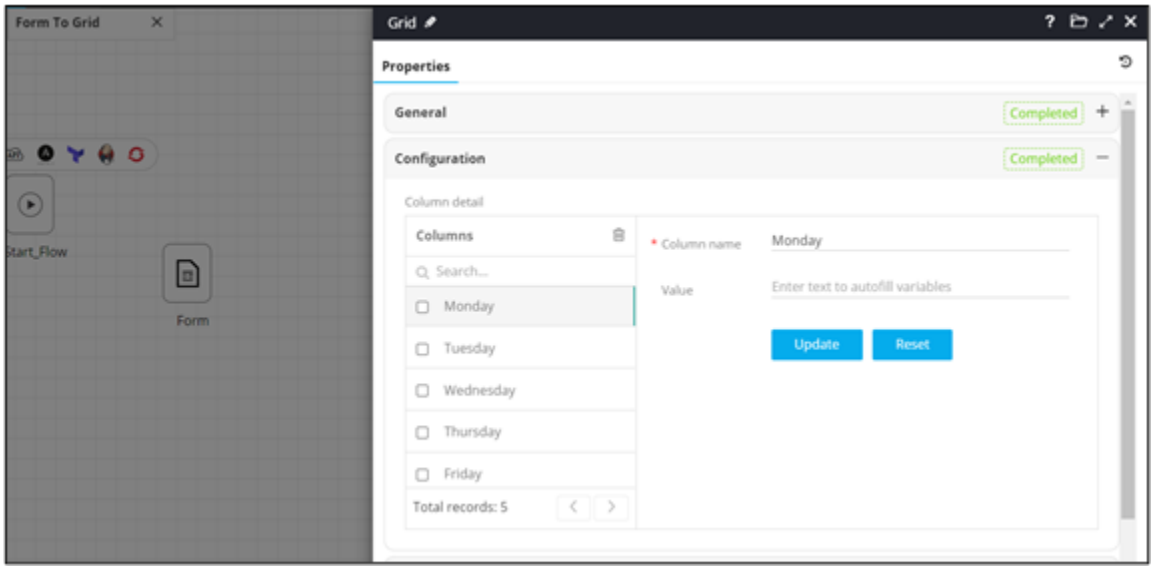
Note: For more information on adding form fields, click here.

5. Under the **Resource & Settings** tab, define the form field values as Global Variables.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

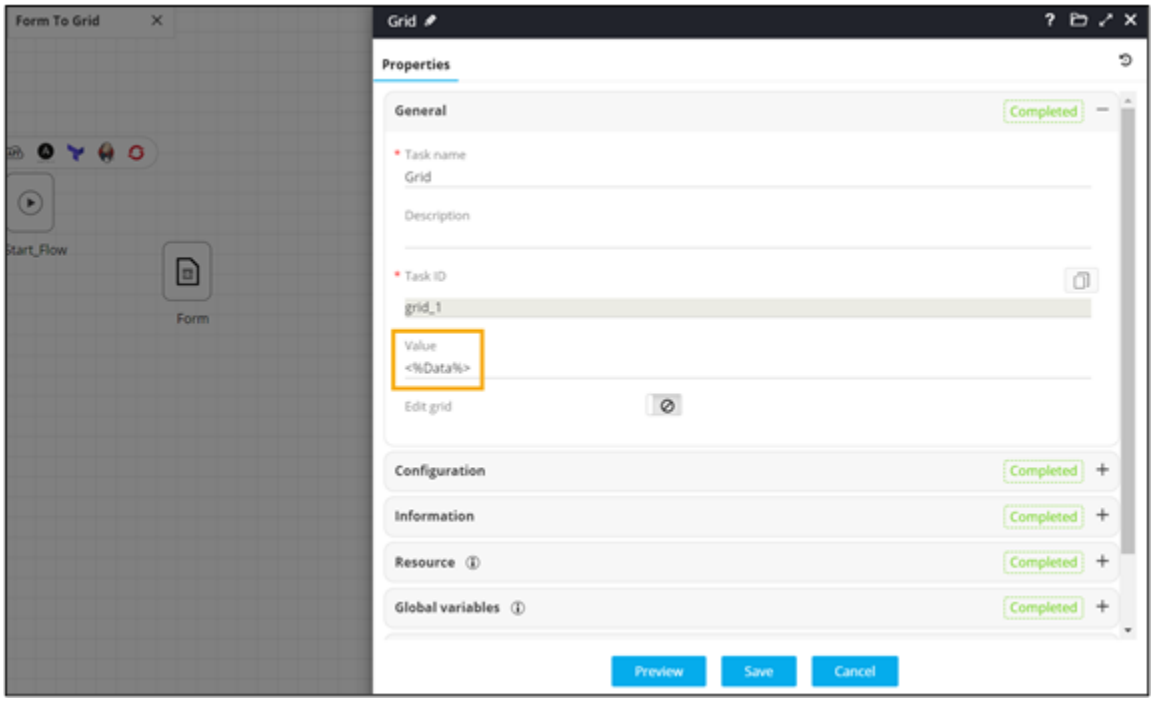


6. From the **User Interface** section, drag and drop the **Grid** task.
7. In the **Grid** task window, under **Properties**, in the **Configuration** section, define the **Columns** required for the Grid.

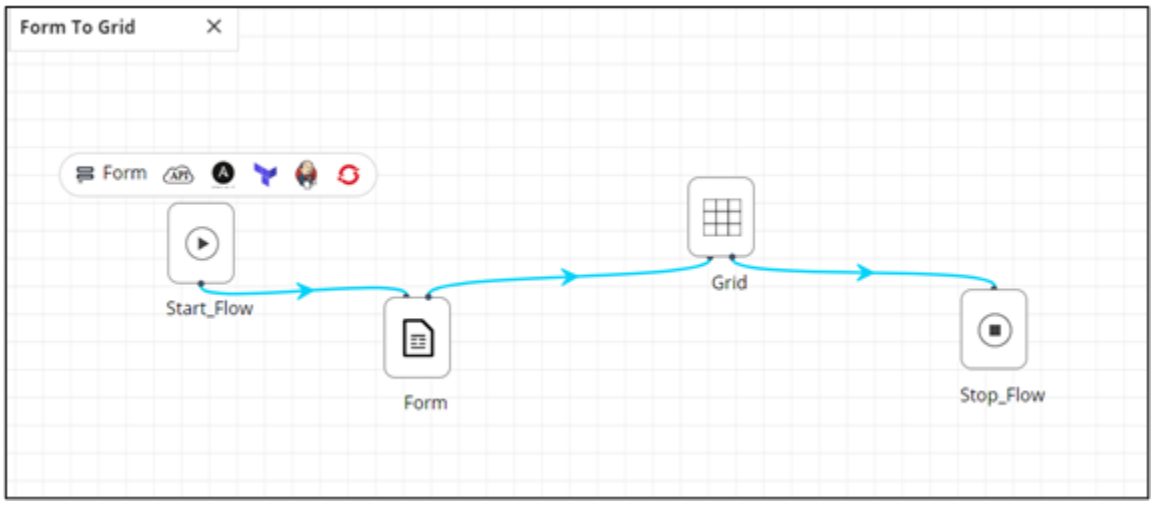


8. Refer the variables defined in the Form task as a value in the Grid task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

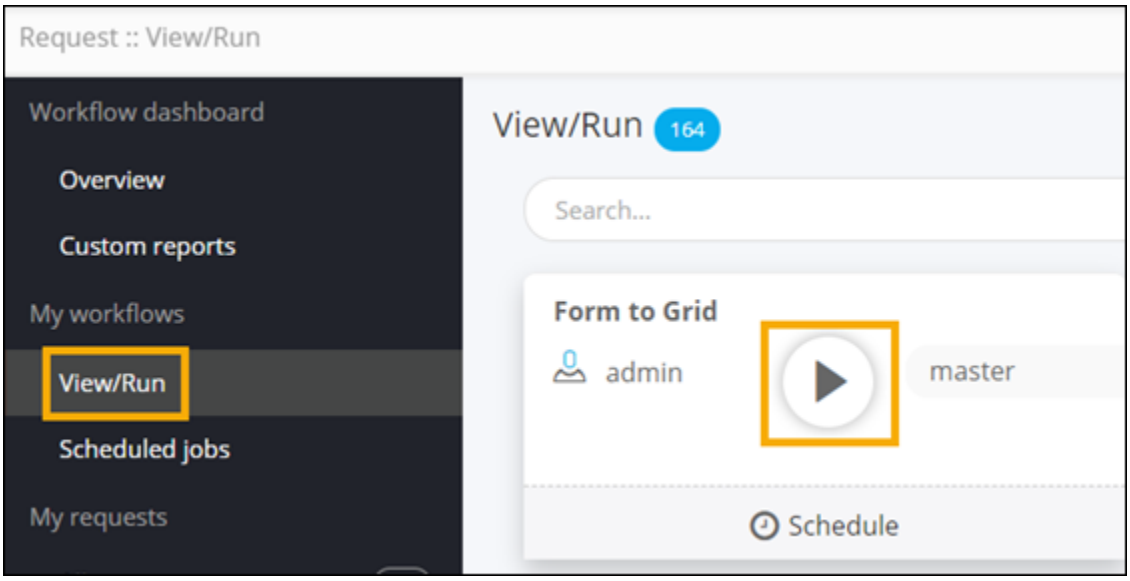


9. Connect and enable the workflow.

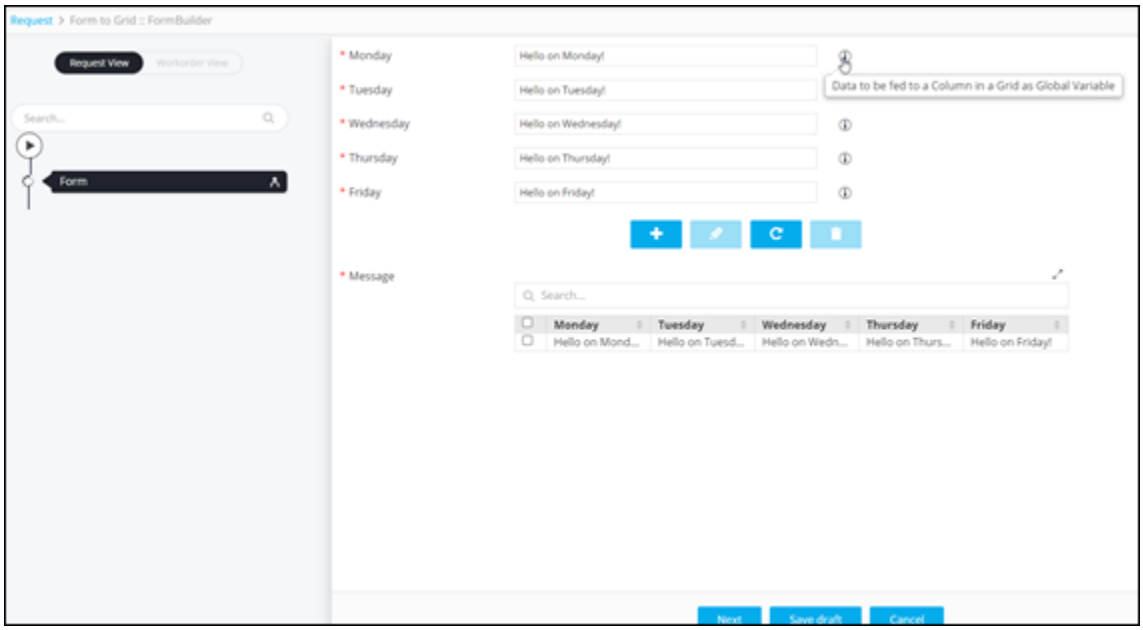


10. Trigger the workflow from the Request : View/Run page.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

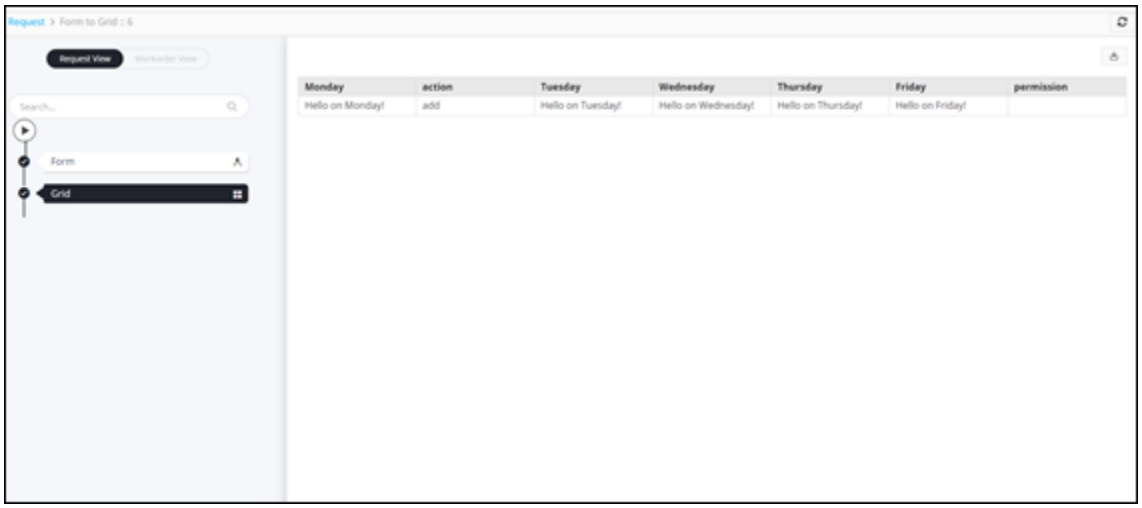


- User inputs received through the **Form**.



- Data displayed as a Grid.

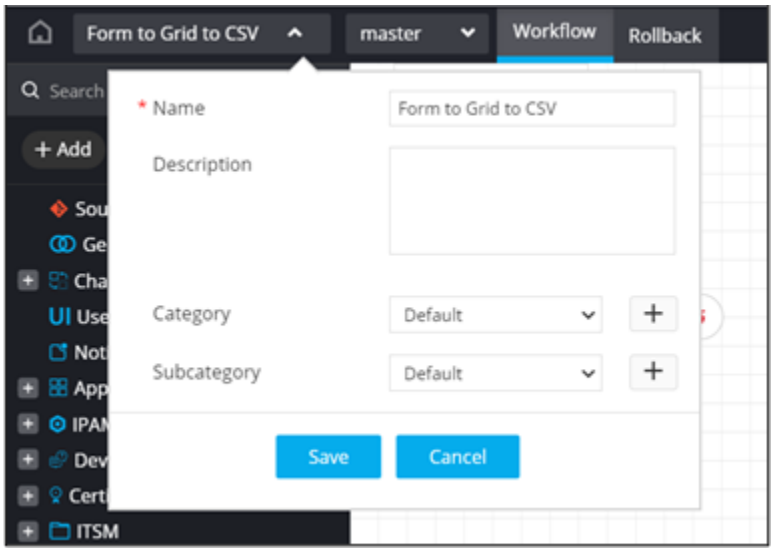
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



How to connect Form to Grid to CSV

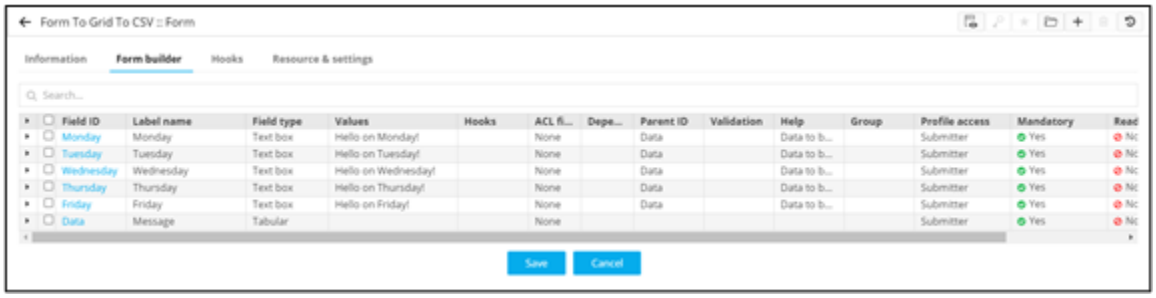
You can design a workflow to get inputs from a Form task, pass the data into a Grid and download the content as a .csv file.

1. Design a new workflow.



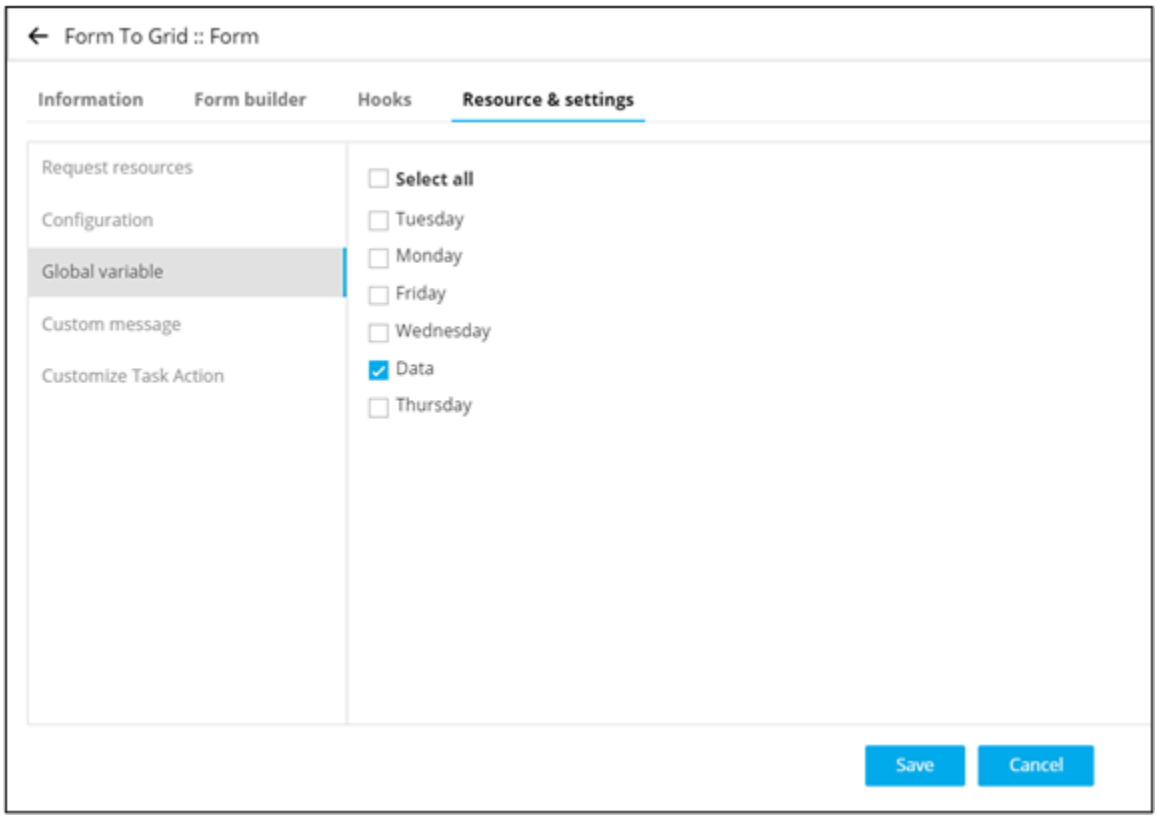
2. From the **User Interface** section, drag and drop the **Form** task.
3. In the **Gallery** window, to design a new form, click .
4. Under the **Form builder** tab, define the form fields to get multiple inputs from the user.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



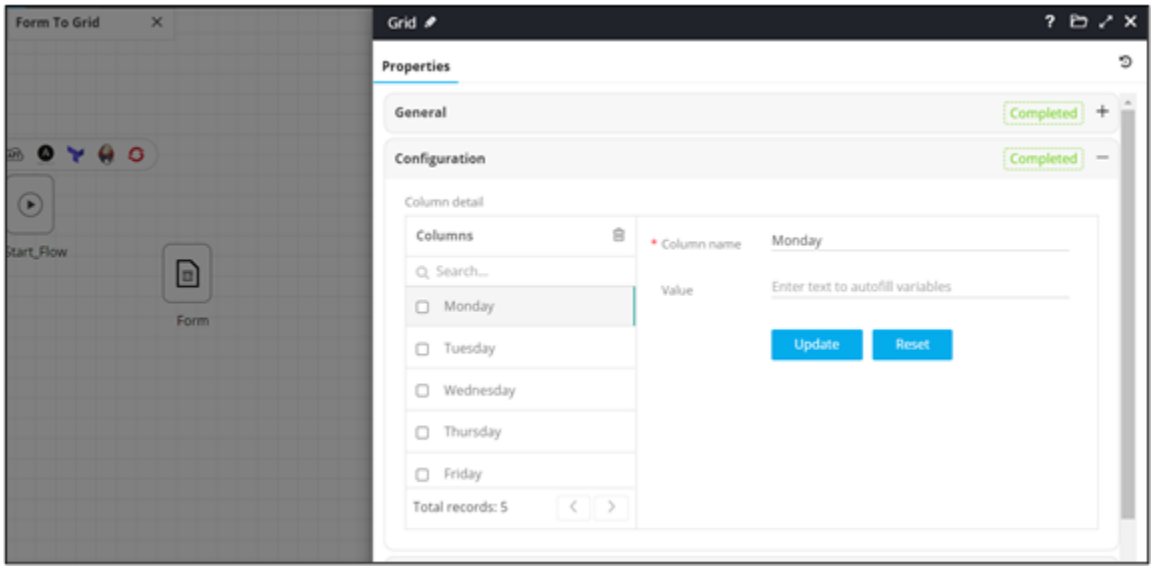
Note: For more information on adding form fields, click here.

5. Under the **Resource & Settings** tab, define form field values as global variables.

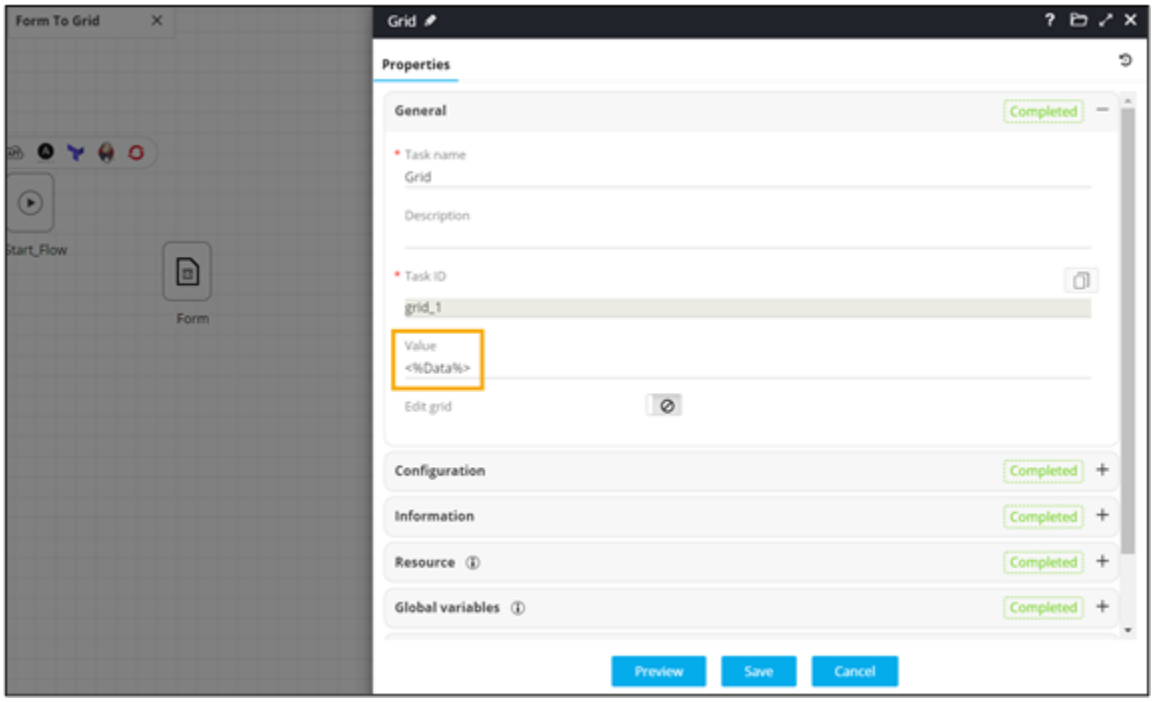


6. From the **User Interface** section, drag and drop the **Grid** task.
7. In the **Grid** task window, under **Properties**, in the **Configuration** section, define the columns required for the Grid.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



8. Refer the variables defined in the Form task as **Value** in the Grid task.

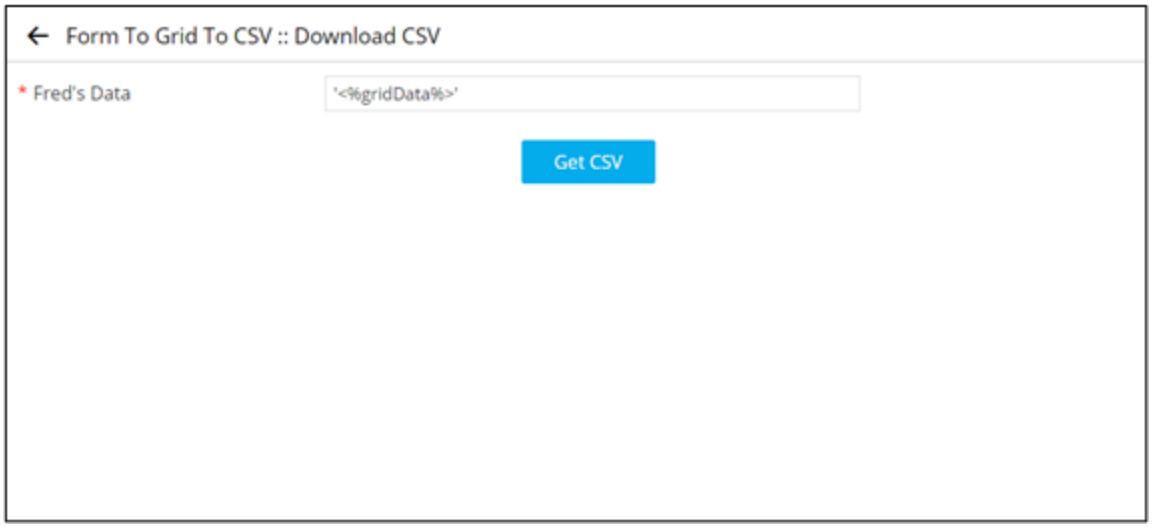


9. Drag and drop another **Form** task.
10. Under the **Form builder** tab, define the form fields with an option to generate a .csv file.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

11. Define associate script to generate a .csv file dynamically.

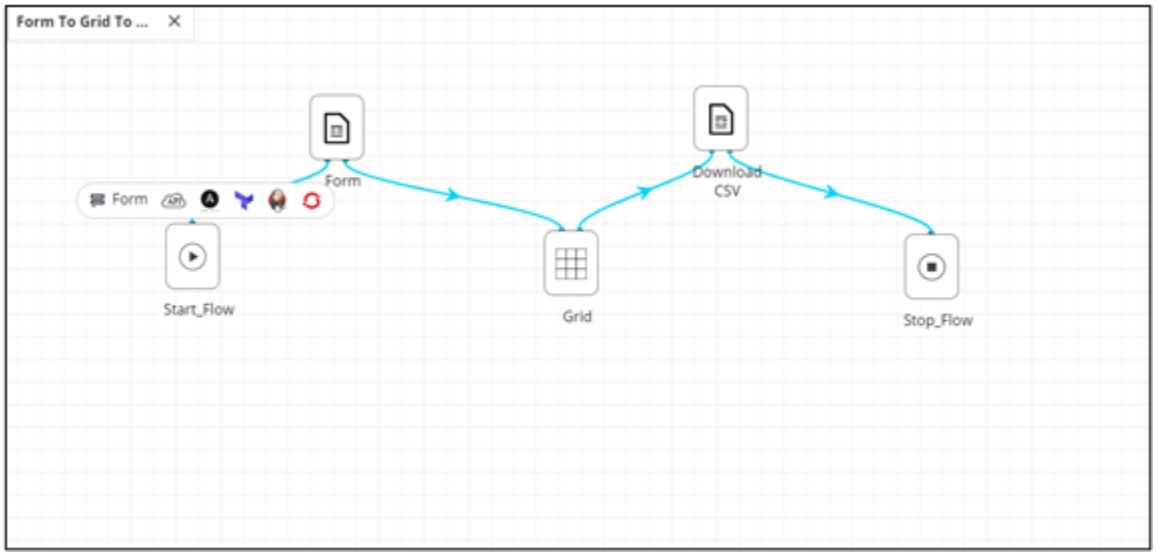
```
import sys

import json

sys.path.insert(0,AVX::DEPENDENCIES)

sys.path.insert(0,AVX::HELPER)

excelvalue ='<FredsData>'

AVX::LOG(str(excelvalue))

excellist=json.loads(excelvalue)

column_names = ['Monday','Tuesday','Wednesday','Thursday','Friday']

cmd = {"AVX::FileDownload" :{"linkData": excellist ,"fileName":'Excel_file', "columnHeaders":column_names}}

print(json.dumps(cmd))
```



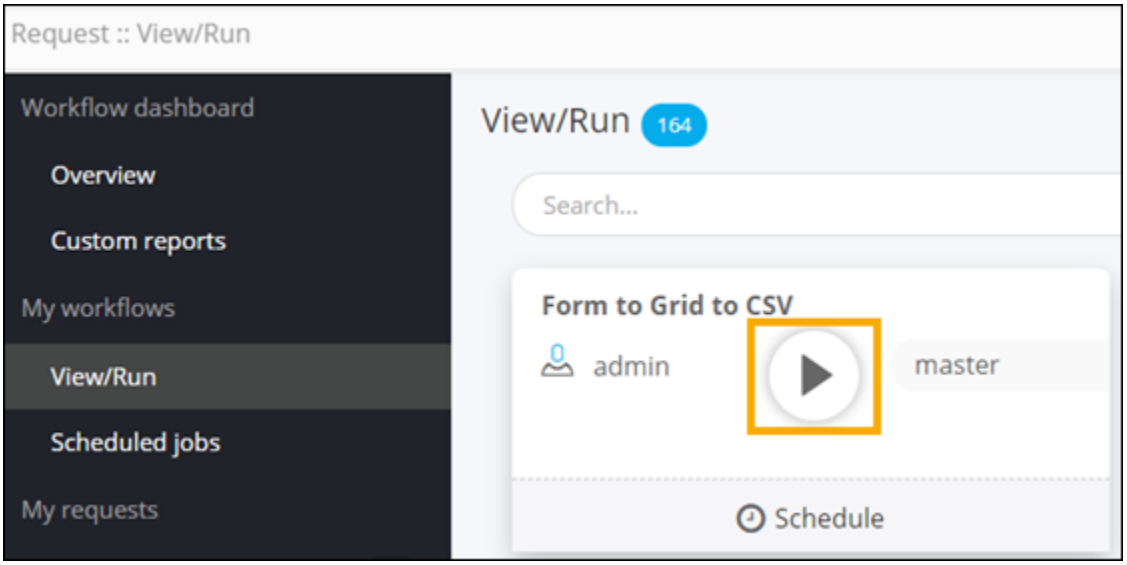
Note: For more information on using scripts within forms, click here.

12. Connect and enable the workflow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

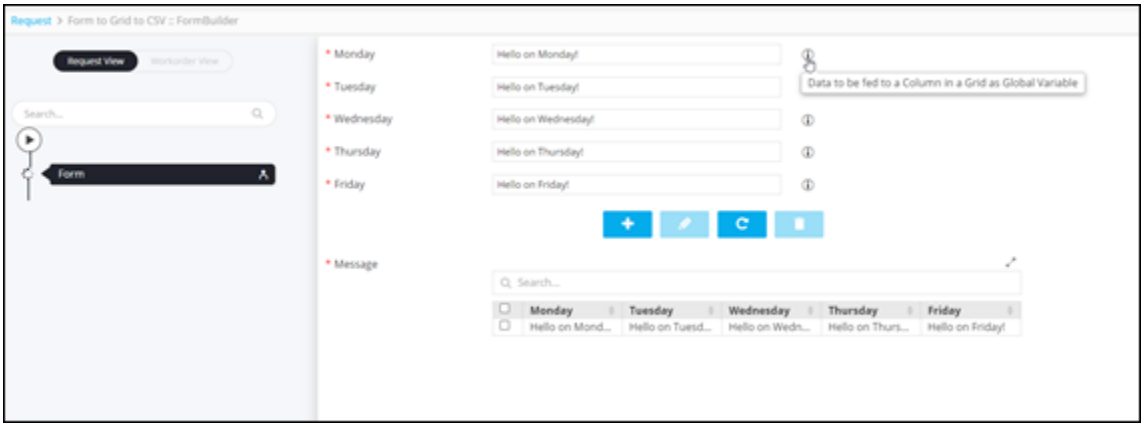


13. Trigger the workflow from the Request :: View/Run page.



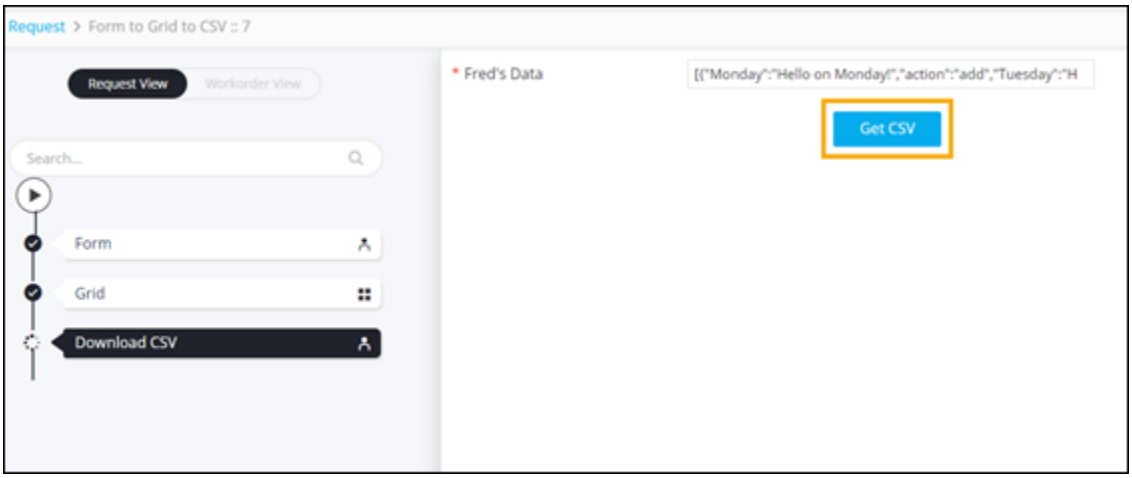
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

User inputs received through the **Form**task.

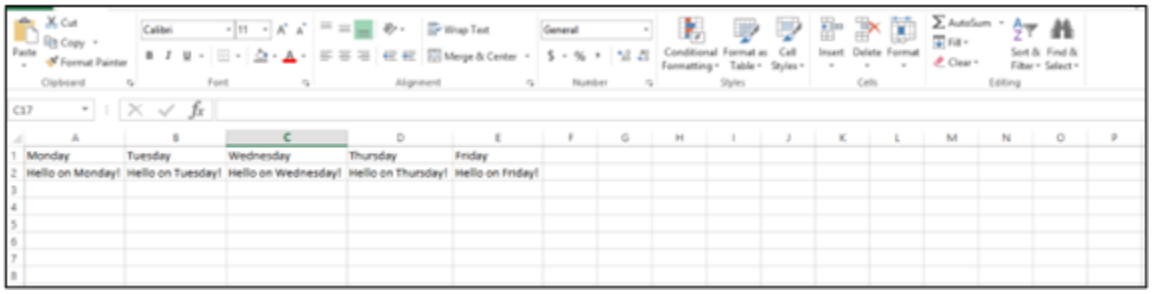


14. To pass the data into a Grid, click **Next**.

15. To download the .csv file, click **Get CSV**.



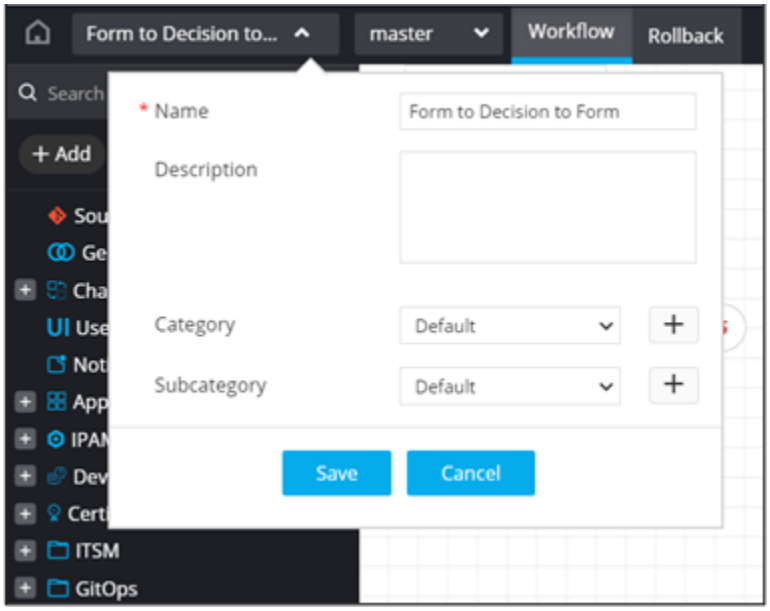
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



How to connect Form to Decision to Form

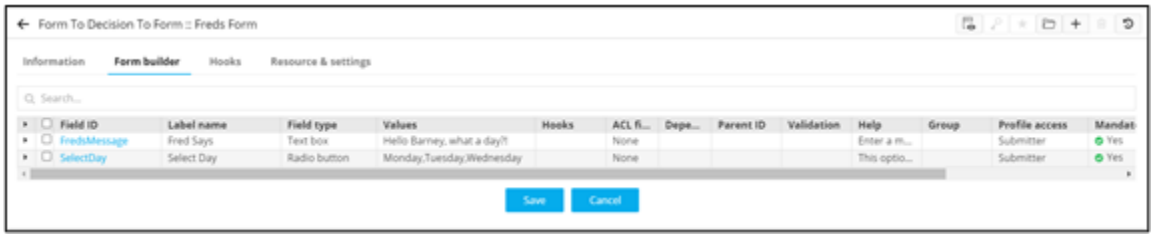
You can create a workflow to get inputs from the user via the Form task and take a decision based on the inputs. In this workflow, the user will be able to select one option, based on which the output will be displayed.

1. Design a new workflow.
2. Provide a **Name** for the workflow.

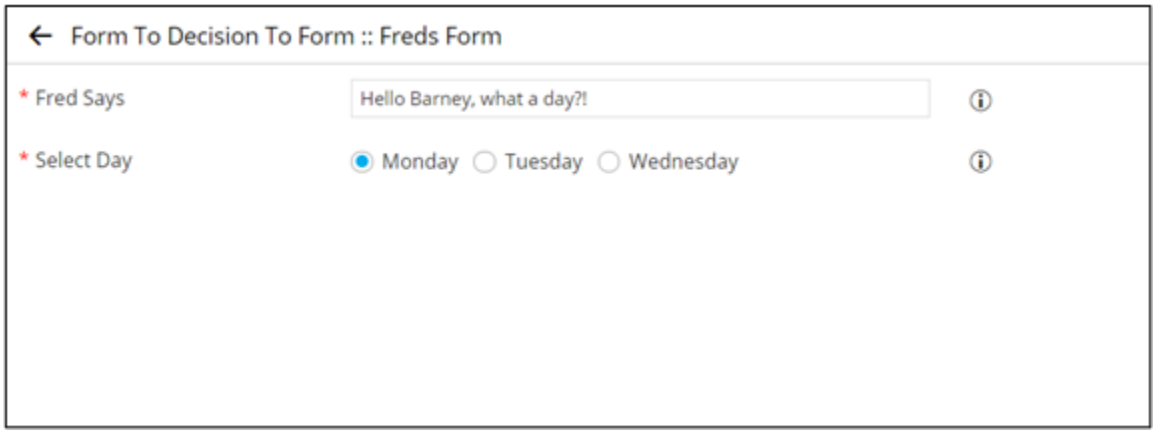


3. Click **Save**.
4. From the **User Interface** section, drag and drop the **Form** task.
5. In the **Gallery** window, to design a new form, click .
6. Under the **Form builder** tab, define the form fields to get simple inputs from the user.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



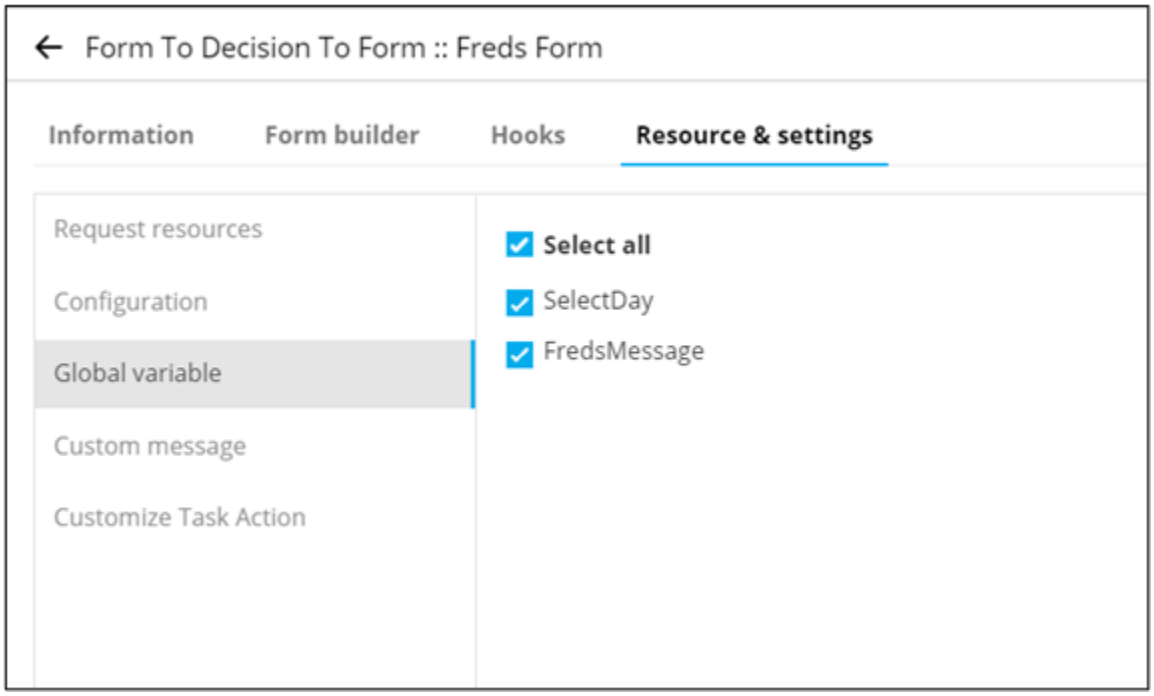
The user will be able to select from the multiple options, based on which a decision will be made.



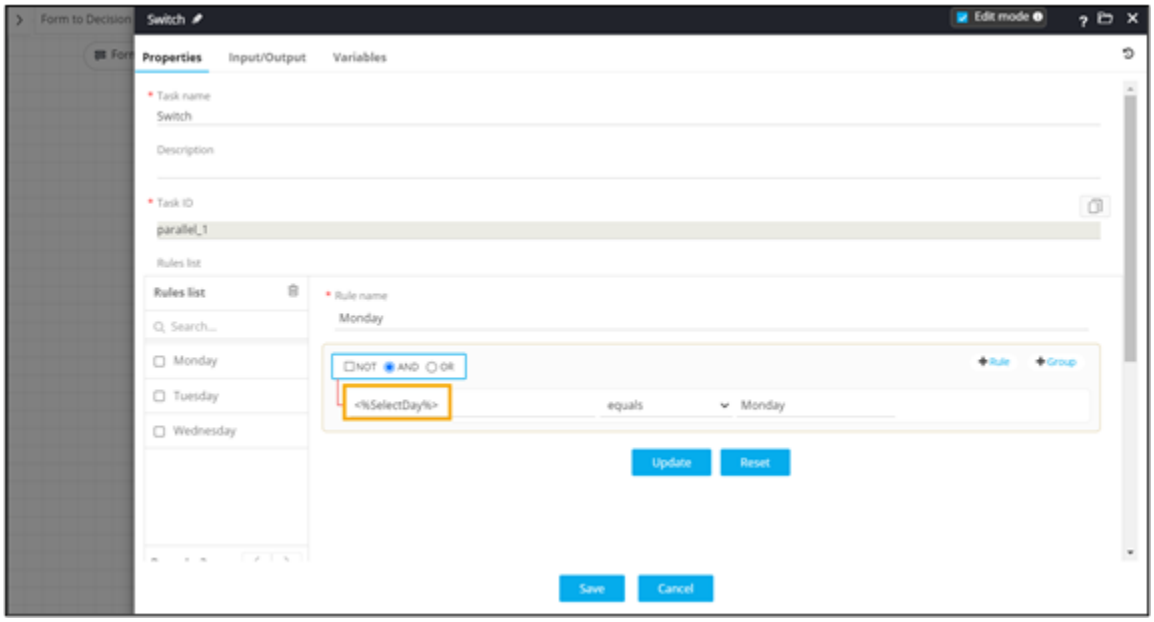
Note: For more information on adding form fields, click here.

7. Under the **Resource & Settings** tab, define and declare the global variable to be referenced in the Form task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

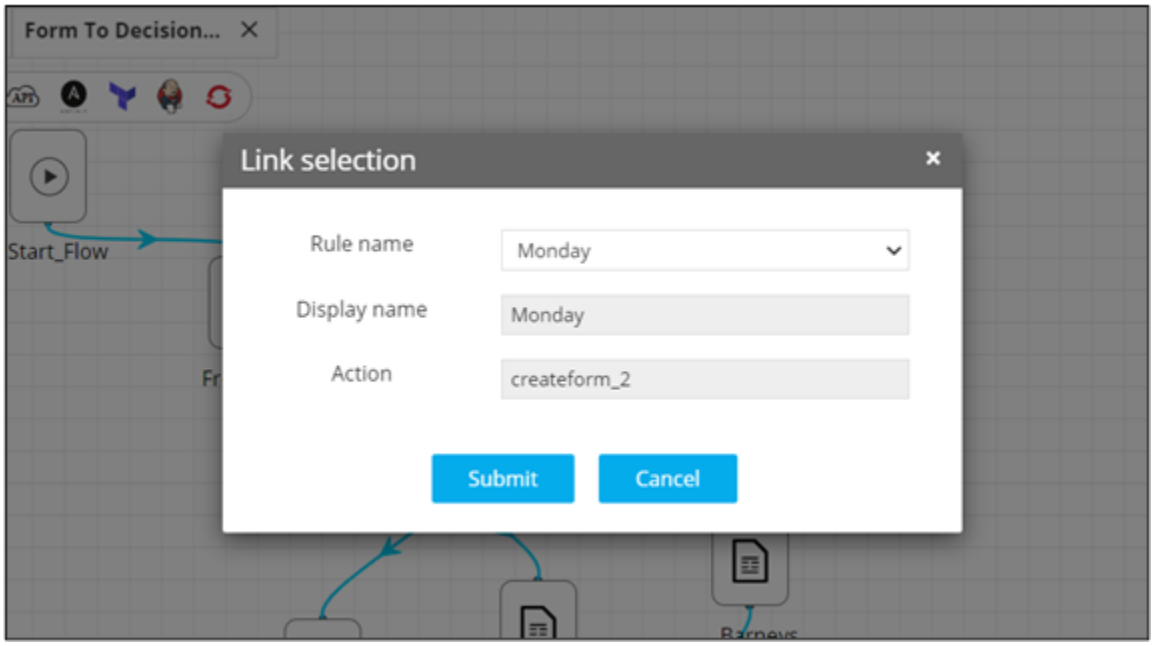


8. From the **General** section, drag and drop the **Switch** task.
9. In the **Switch** task window, under **Properties**, define the rule based on which the decision will be made.
10. Refer the global variable (`<%SelectDay%>`) from the Form task.

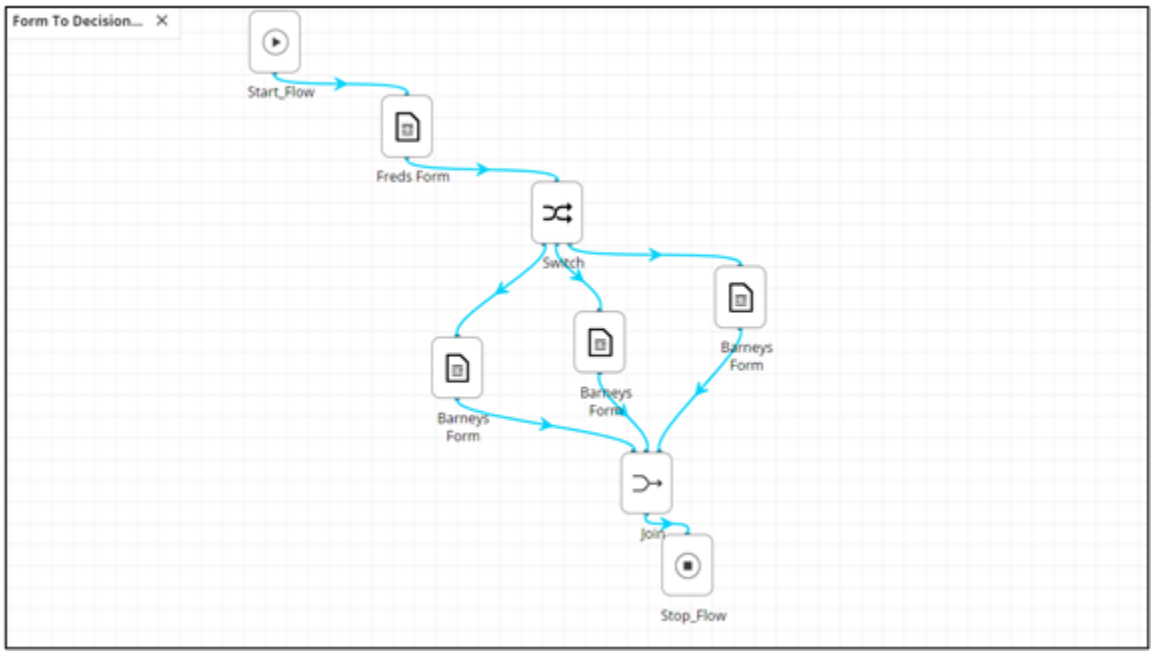


11. Define the Form tasks that will print the output based on the decision.
12. Connect the tasks based on the rules defined in the Switch task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

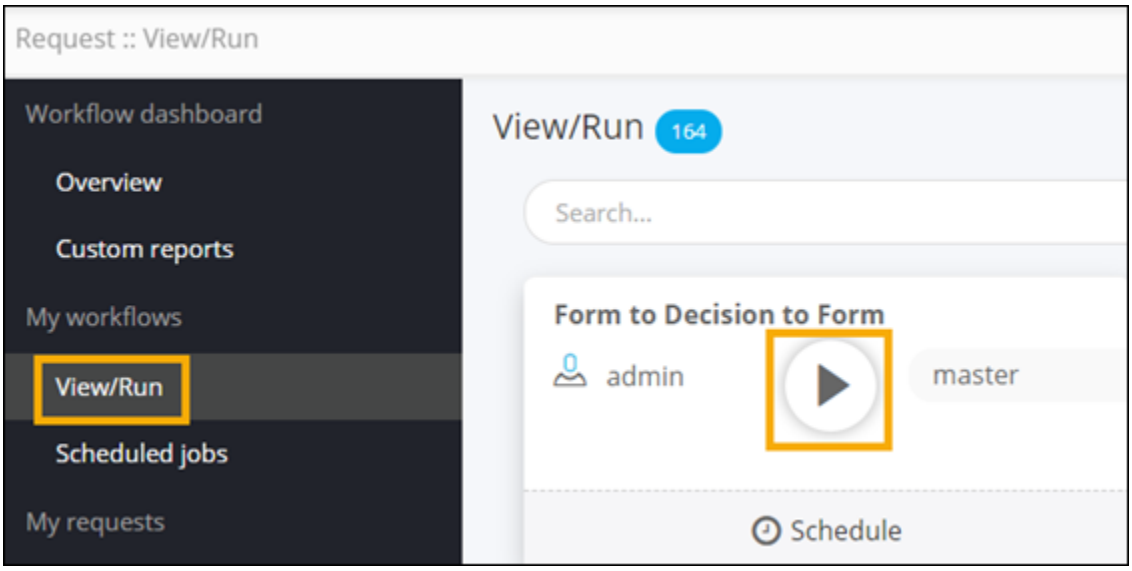


13. To bring the flow to a logical end, from the **General** section, drag and drop the **Join** task.
14. Enable the workflow.

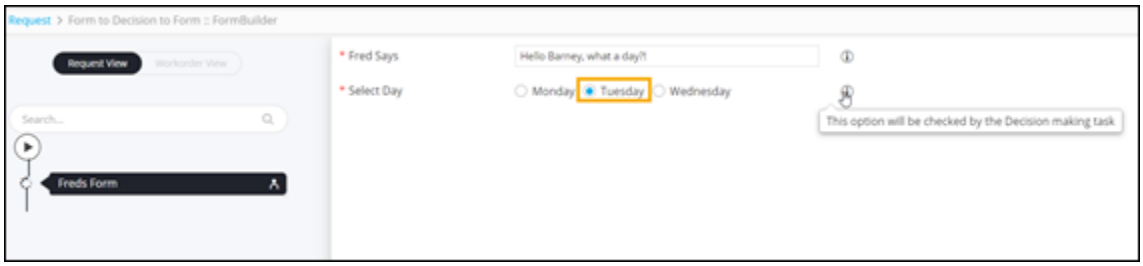


15. Trigger the workflow from the Request :: View/Run page.

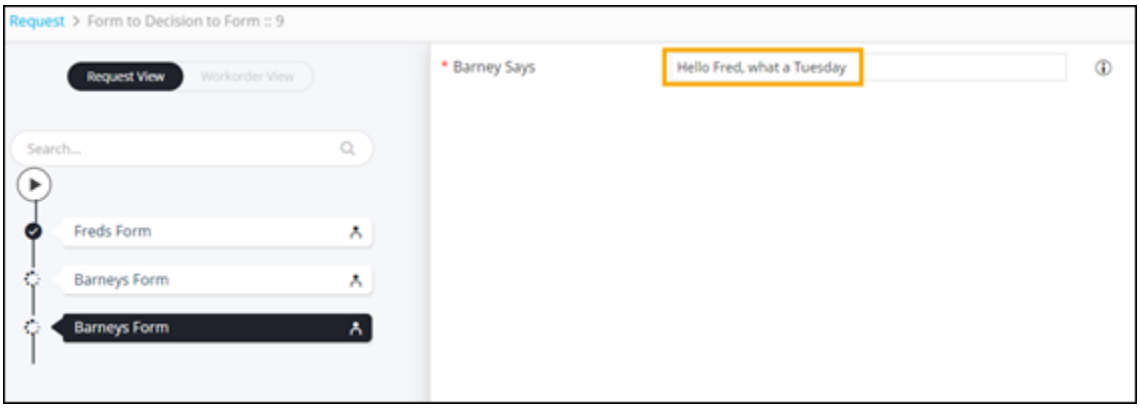
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



- Get inputs from the user to select a day.

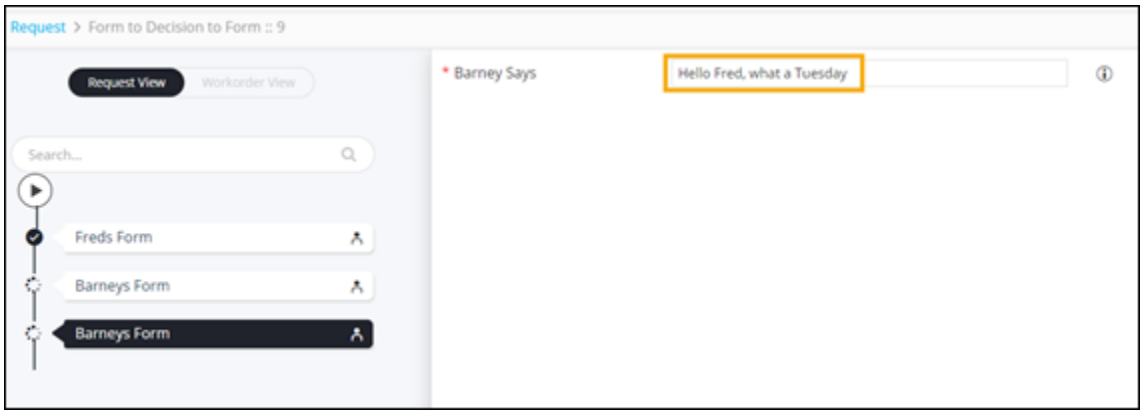


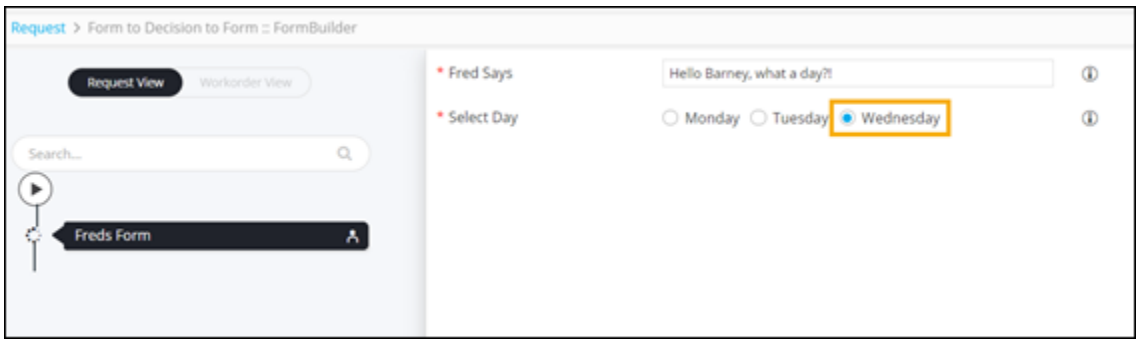
- Based on the selection, a decision will be made and the day printed accordingly.
  - Printing output based on the selection - Tuesday.



  - Printing output based on the selection - Wednesday.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



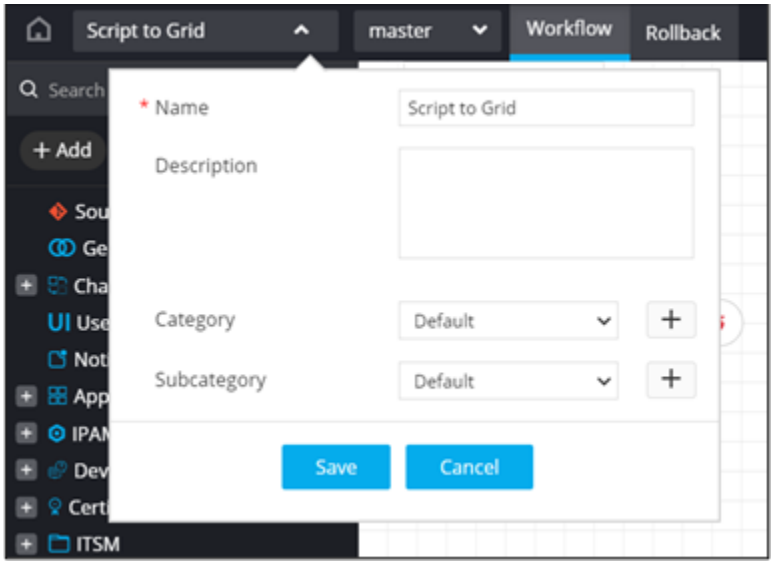


How to connect Script to Grid

You can define a script to get multiple values (in a list) and pass them into a Grid component using Global Variables.

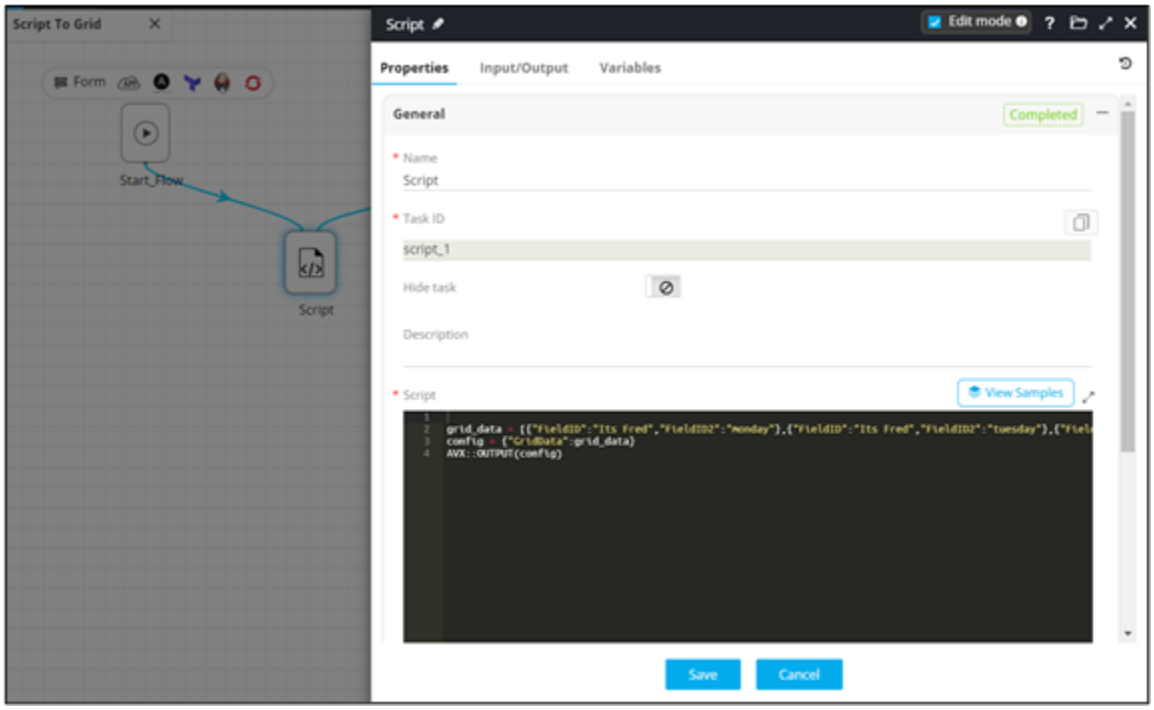
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. Design a new workflow.



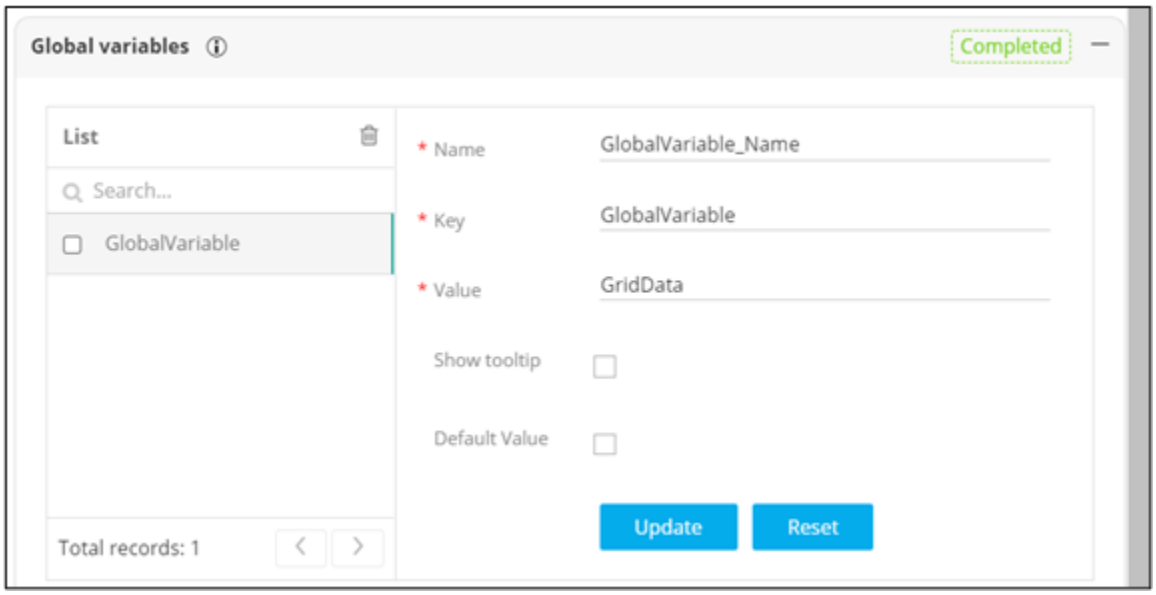
2. From the **General** section, drag and drop a **Script** task.
3. In the **Script** task window, under **Properties**, in the **General** section, define the list of values.

```
grid_data = [{"FieldID":"Its Fred","FieldID2":"monday"},{"FieldID":"Its Fred","FieldID2":"tuesday"},{"FieldID":"Its Barney","FieldID2":"wednesday"}]

config = {"GridData":grid_data}

AVX::OUTPUT(config)
```

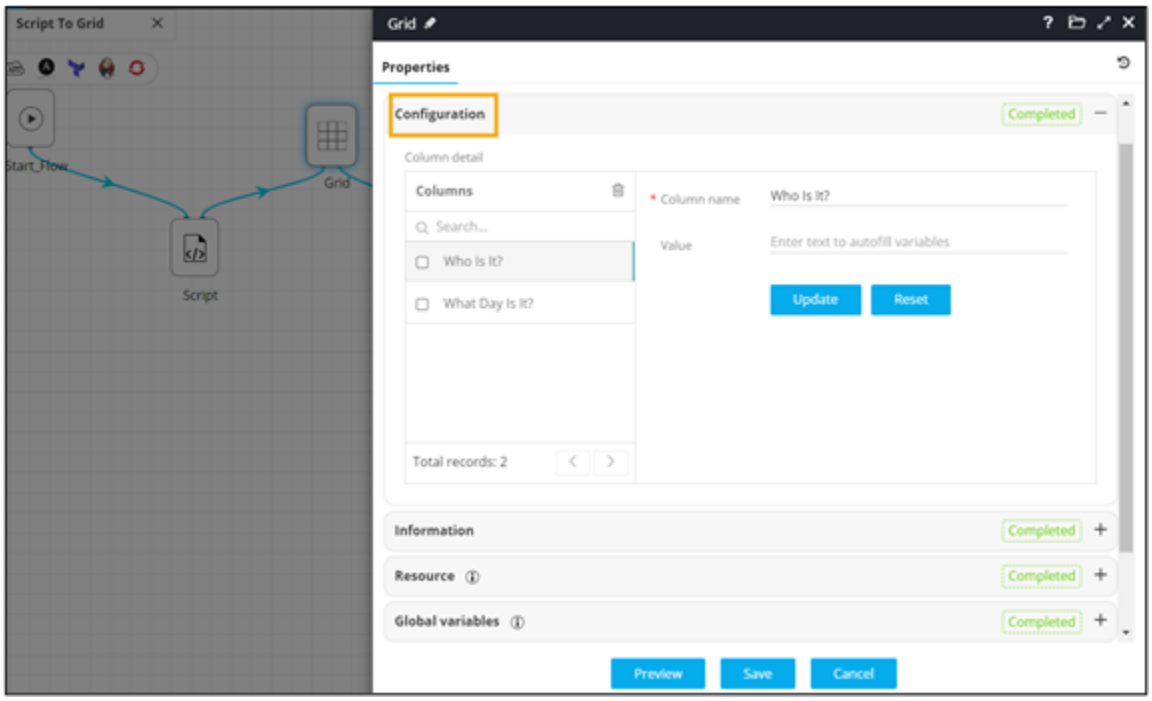


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

4. In the **Script** task window, under **Properties**, in the **Global variables** section, define and declare the grid data value as a global variable to be referenced in the Grid task.

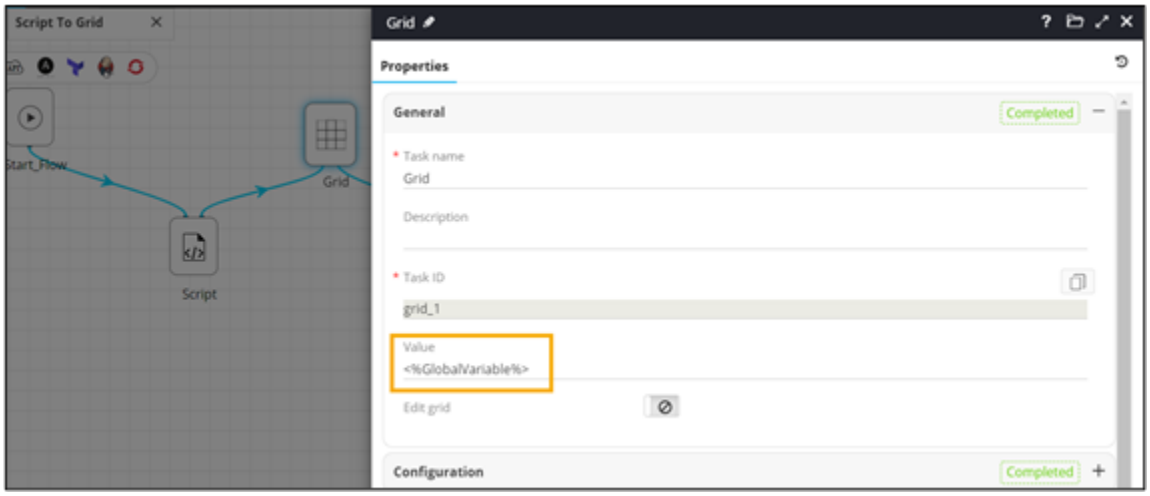


5. From the **User Interface** section, drag and drop the **Grid** task.
6. In the **Grid** task window, under **Properties** in the **Configuration** section, define the **Columns** of the Grid.

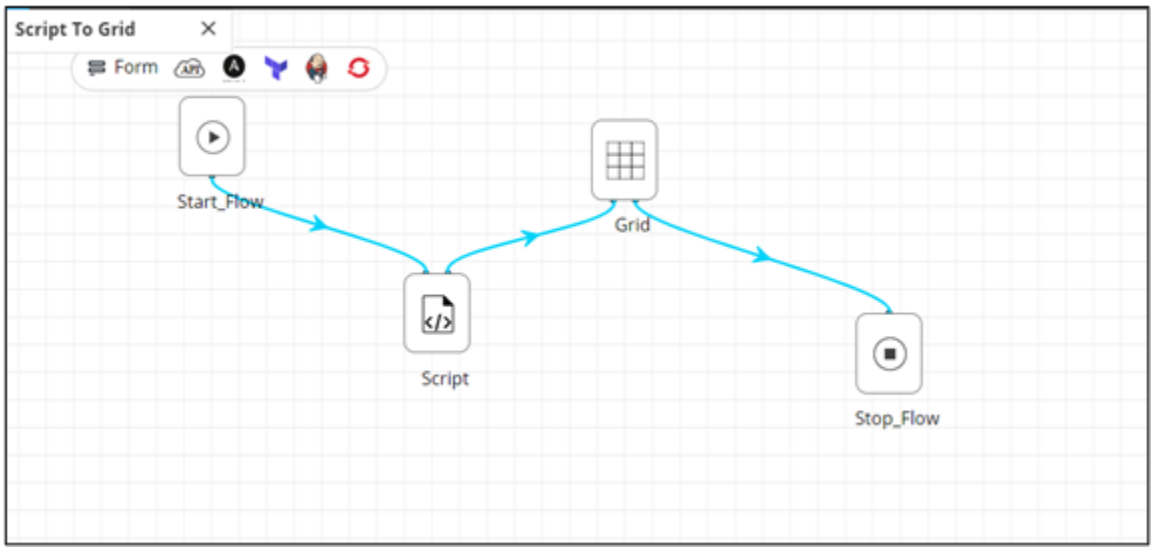


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

7. Refer the global variable to be used.

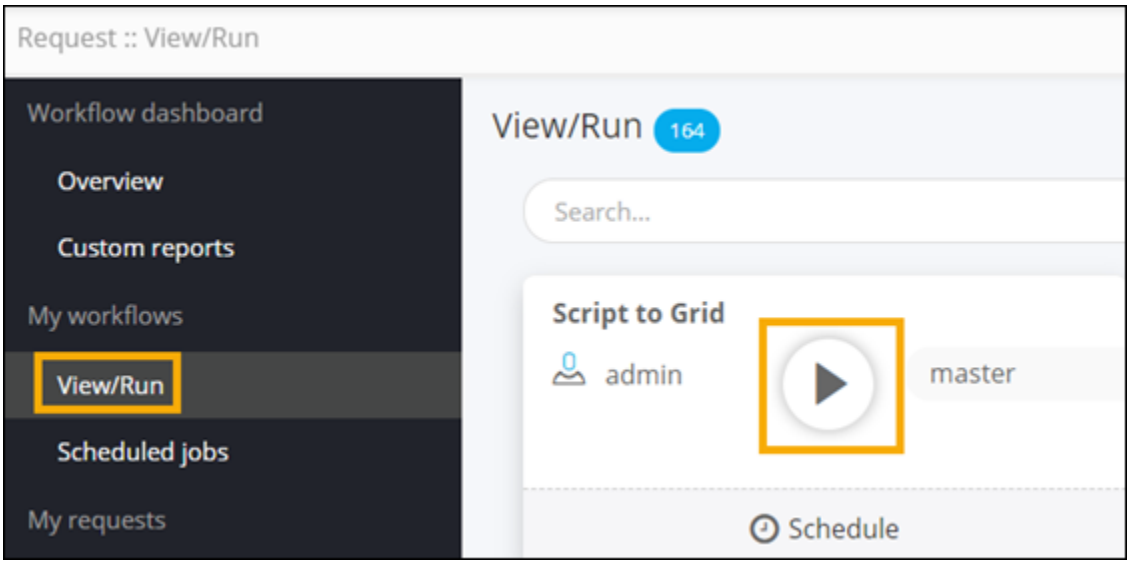


8. Connect and enable the workflow.

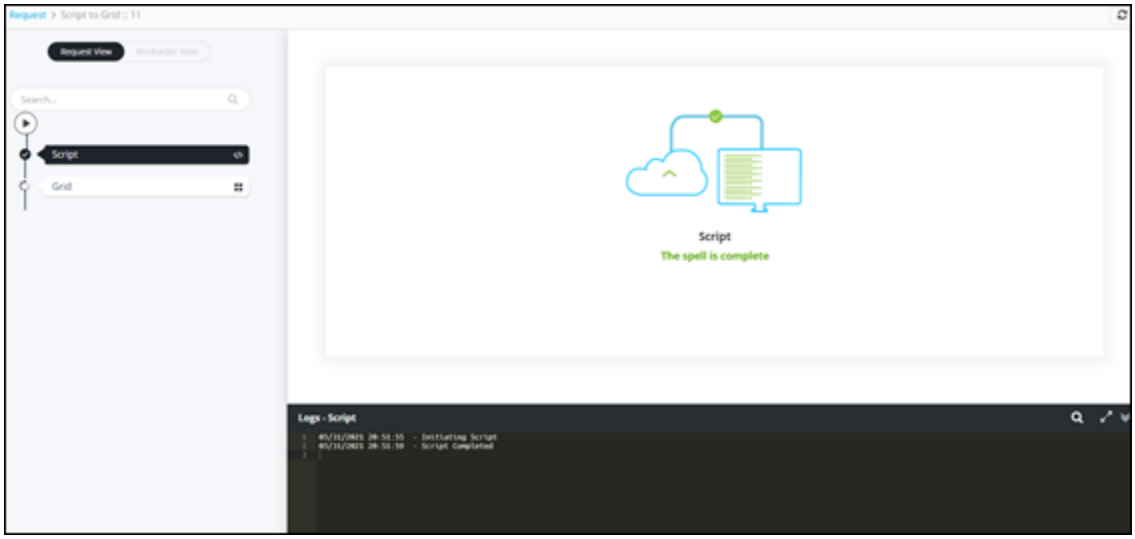


9. Trigger the workflow from the Request :: View/Run page.

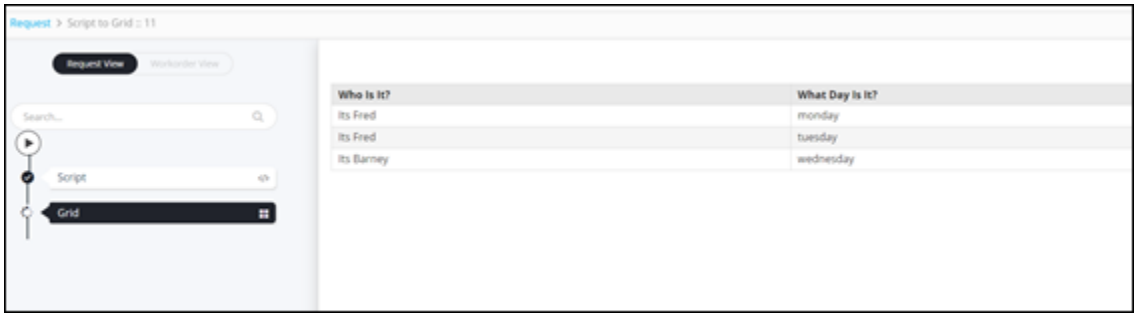
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



- Script executed with custom messages.



- Data displayed in a Grid using global variables from the Script task.

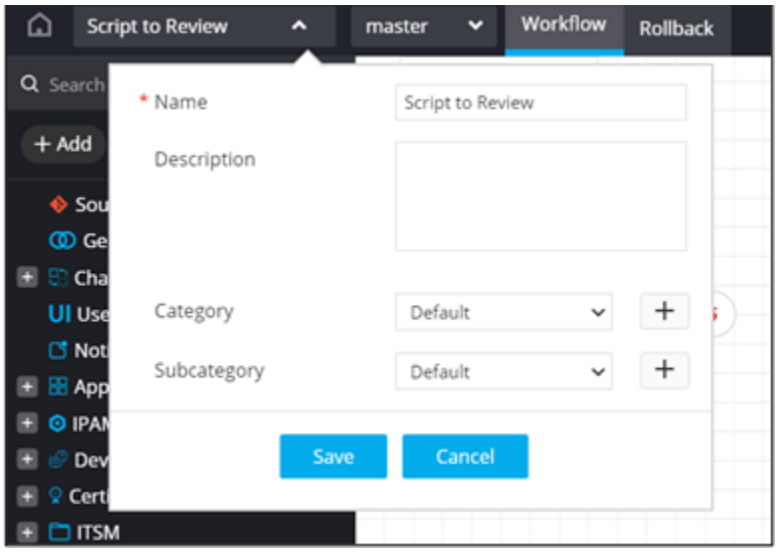


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

How to connect Script to Review

You can design a workflow to pass values from a Script into a Review task using global variables. The Review component is used for approvals.

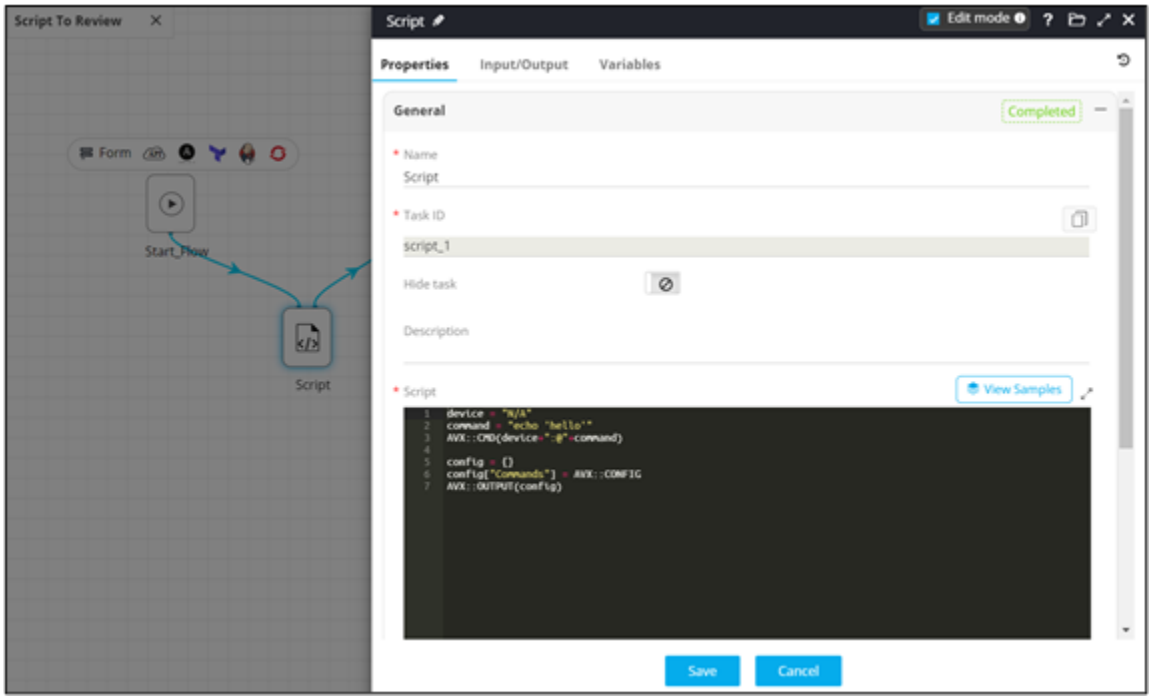
1. Design a new workflow.



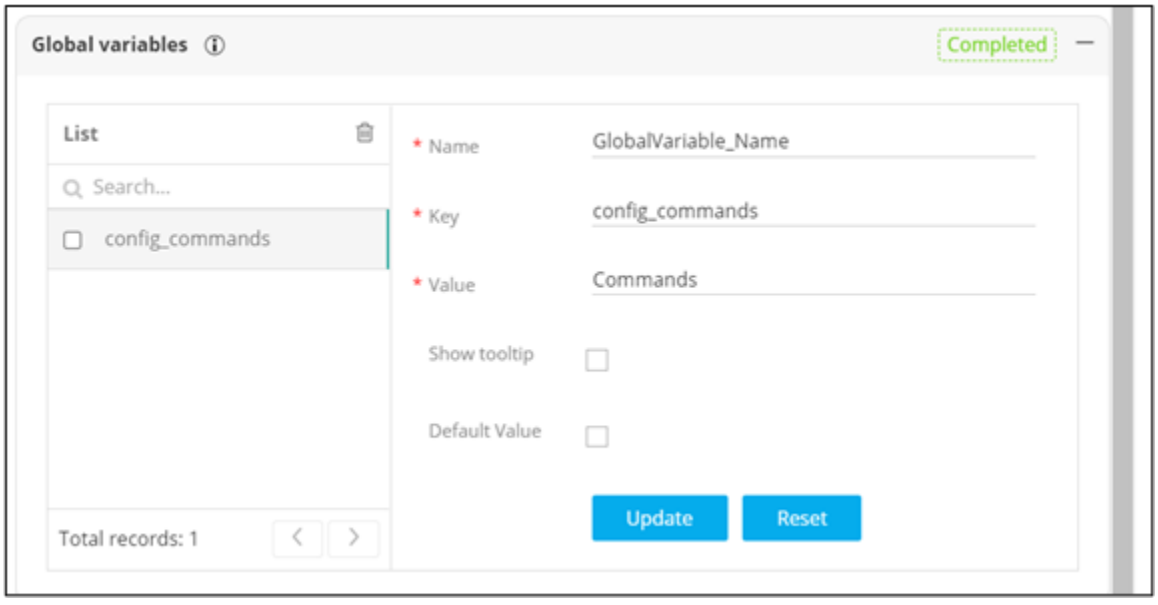
2. From the **General** section, drag and drop a **Script** task.
3. In the **Script** task window, under **Properties**, in the **General** section, define a command/configuration.

```
device = "N/A"

command = "echo 'hello'"

AVX::CMD(device+":@"+command)

config = {}

config["Commands"] = AVX::CONFIG

AVX::OUTPUT(config)
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

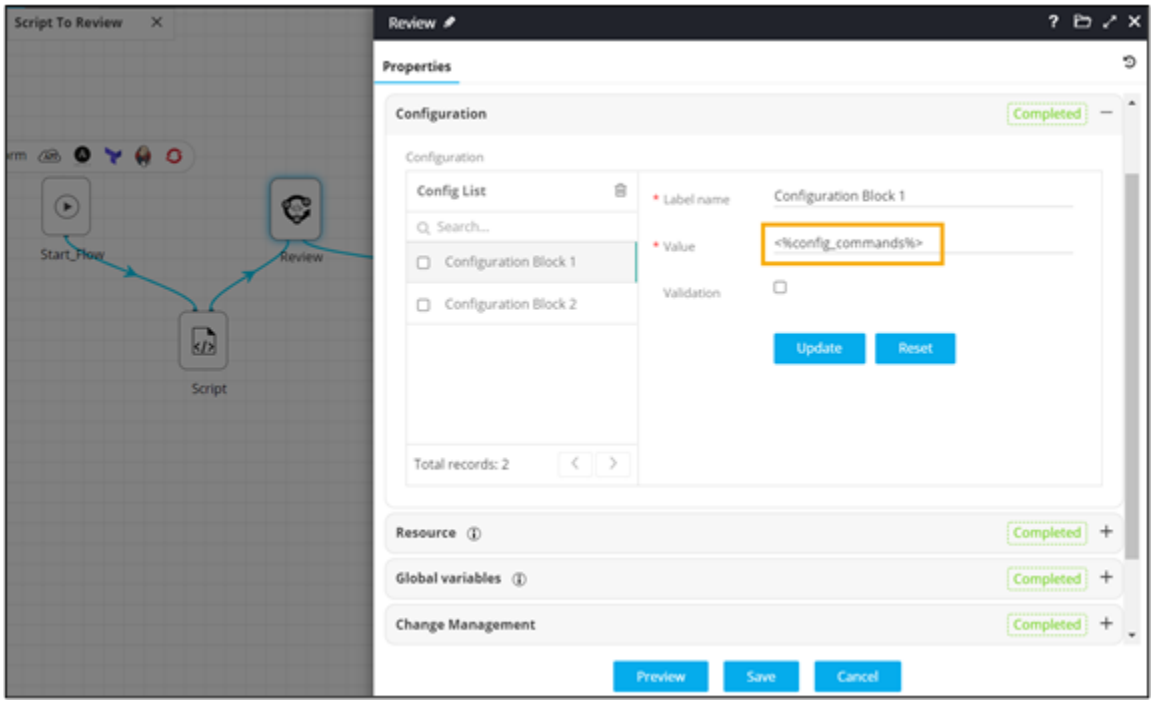


4. In the **Script** task window, under **Properties**, in the **Global variables** section, declare the commands as global variables to be passed into the next (Review) task.

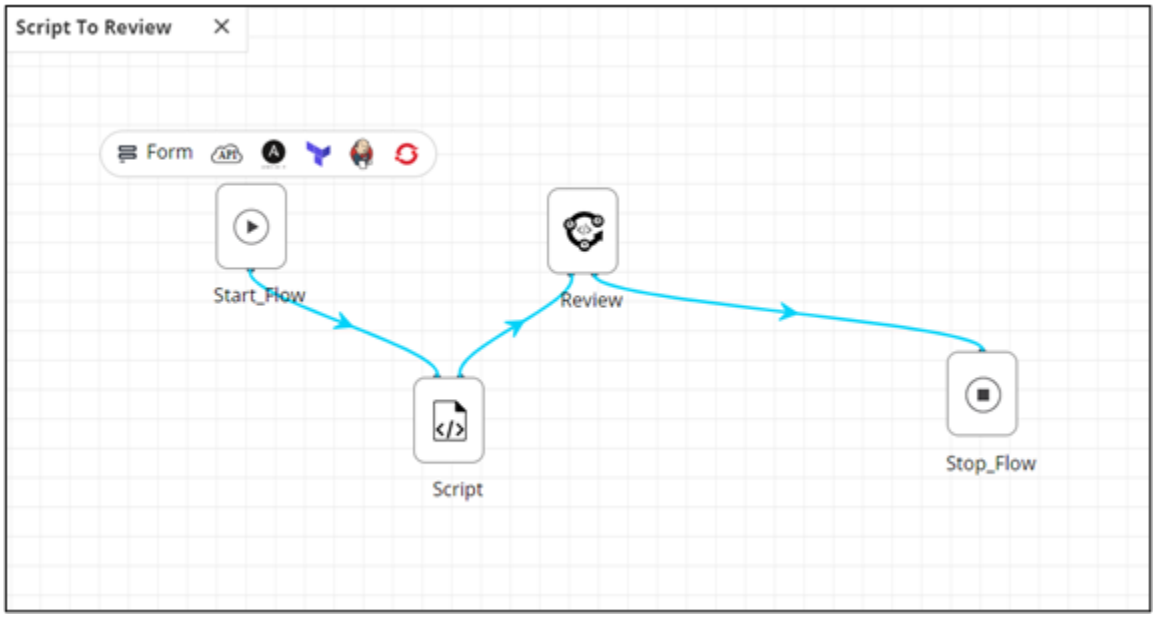


5. From the **General** section, drag and drop the **Review** task.
6. In the **Review** task window, under **Properties**, in the **Configuration** section, define the configuration blocks.
7. Refer the global variables defined in the Script.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

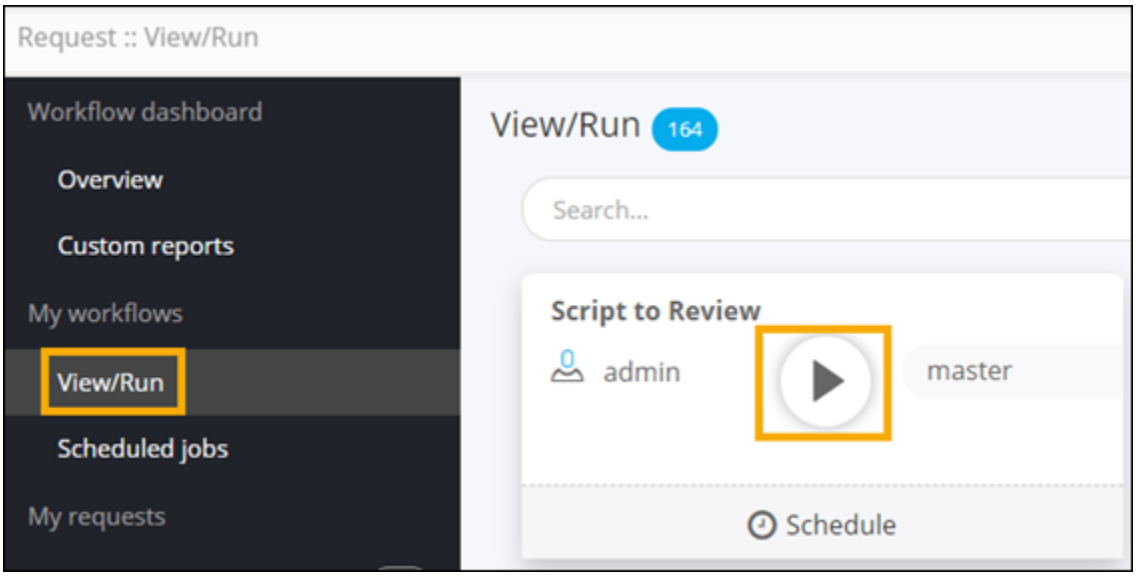


8. Connect and enable the workflow.

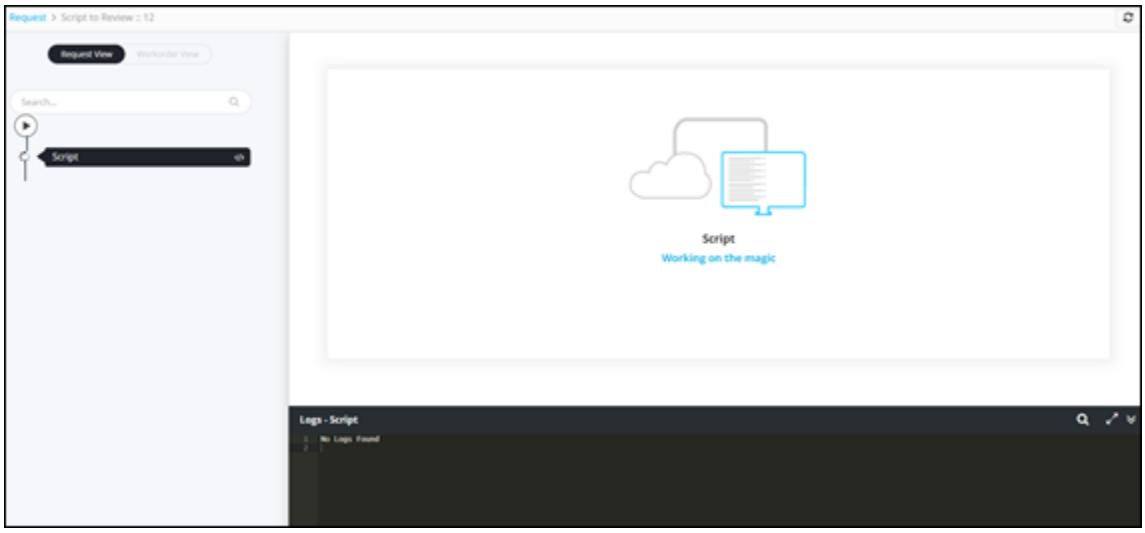


9. Trigger the workflow from the Request :: View/Run page.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

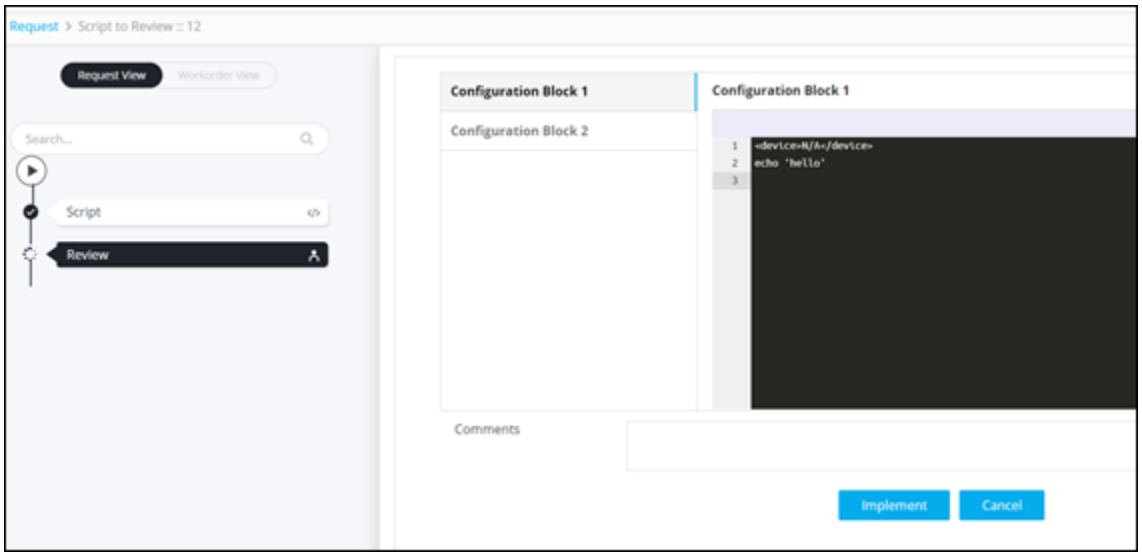


- Script executed with custom messages.



- **Review** task showing configuration blocks.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



How to connect Form to Create Ticket to Close Change Ticket

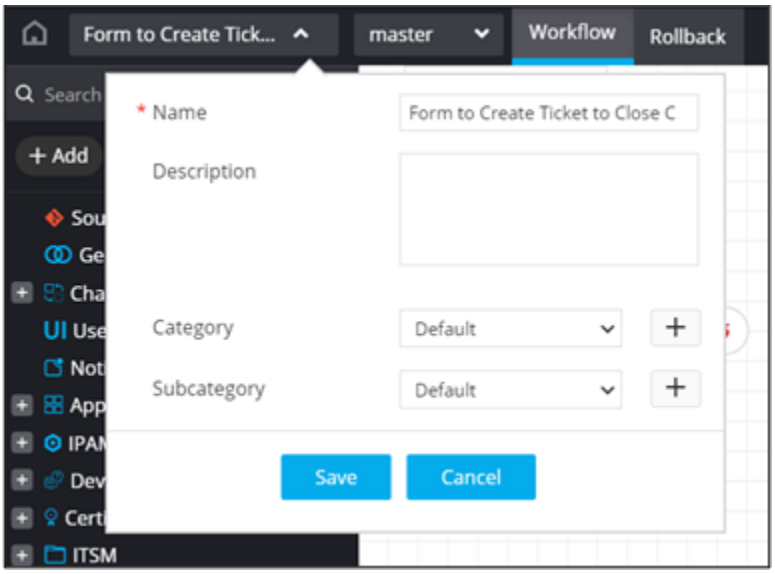
You can create a workflow to get user inputs and create a simple change ticket on ServiceNow (ITSM) using a REST API task and global variables.

1. Define the ITSM configuration for ServiceNow to get credentials and for field mapping between AppViewX and ServiceNow.

   Note: For more information, refer to the section on ITSM Vendor Configuration.

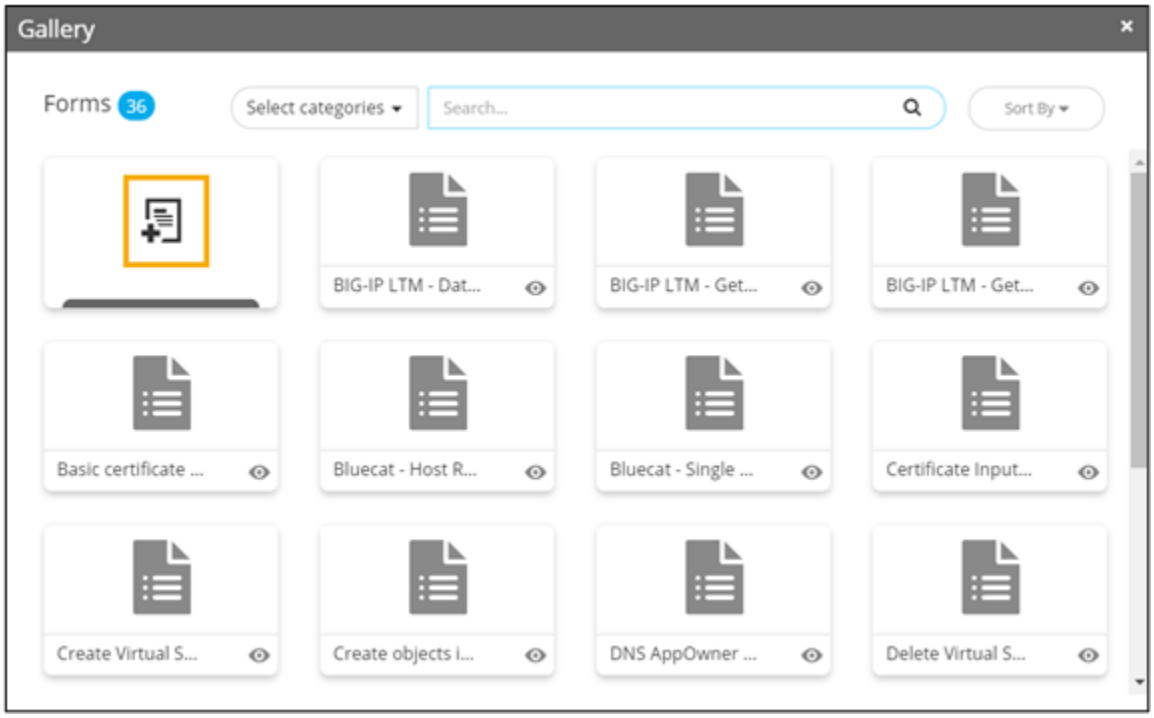
2. Design a new workflow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



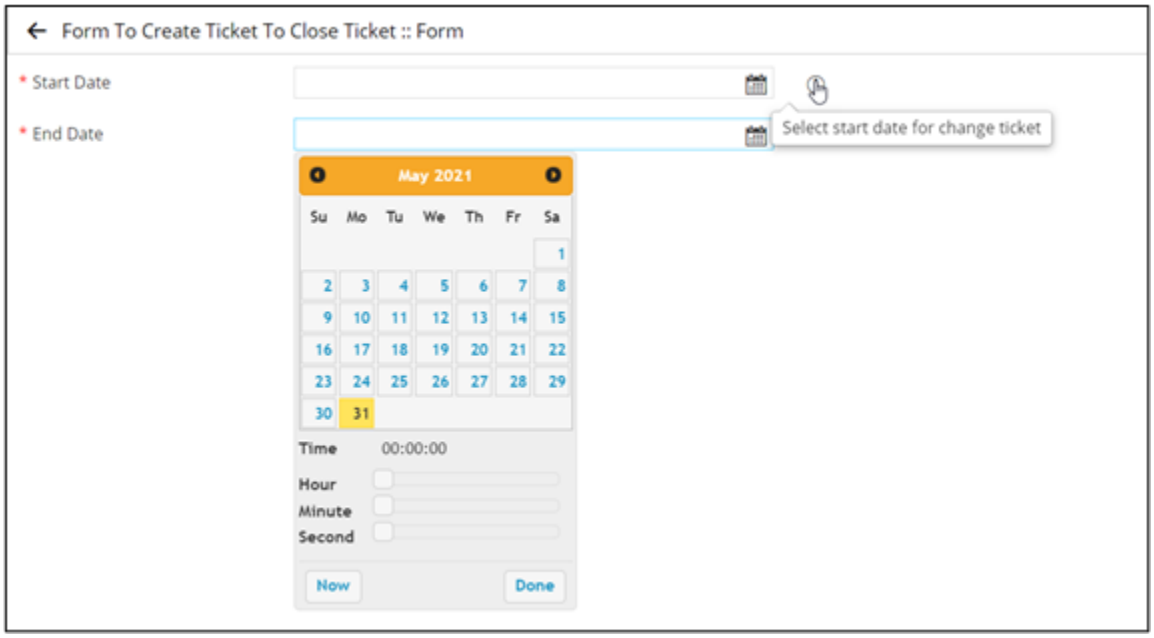
3. From the **User Interface** section, drag and drop the **Form** task.

4. In the **Gallery** window, to design a new form, click .



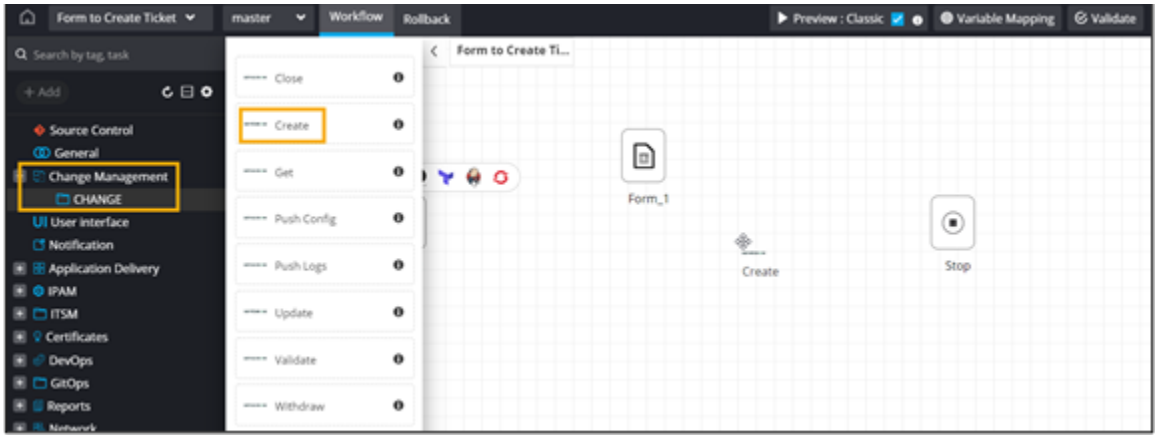
5. Under the **Form builder** tab, define the form fields to get inputs from the user.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



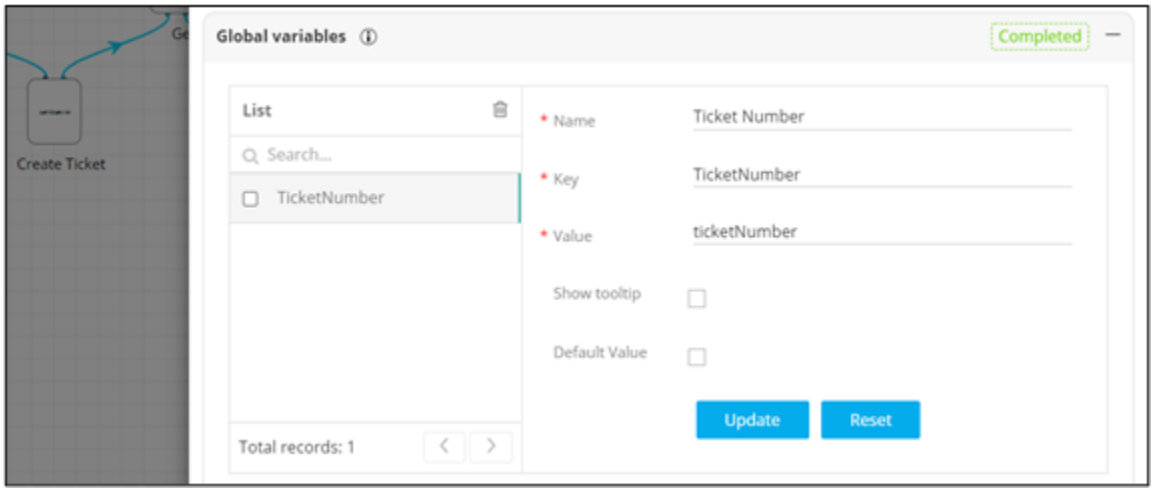
Note: For more information on adding form fields, click here.

6. Under **Change Management**, from the **Change** folder, drag and drop the **Create** task.

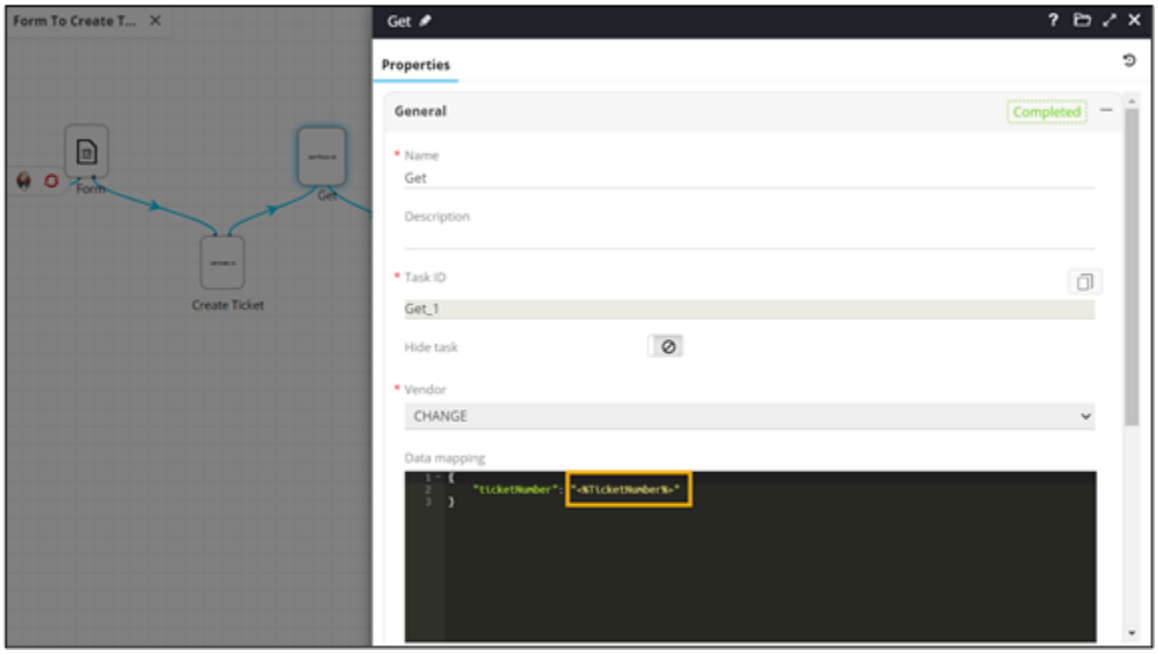


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

7. In the **Create** task window, under **Properties**, in the **Global variables** section, define the global variable to be referenced in the next task.

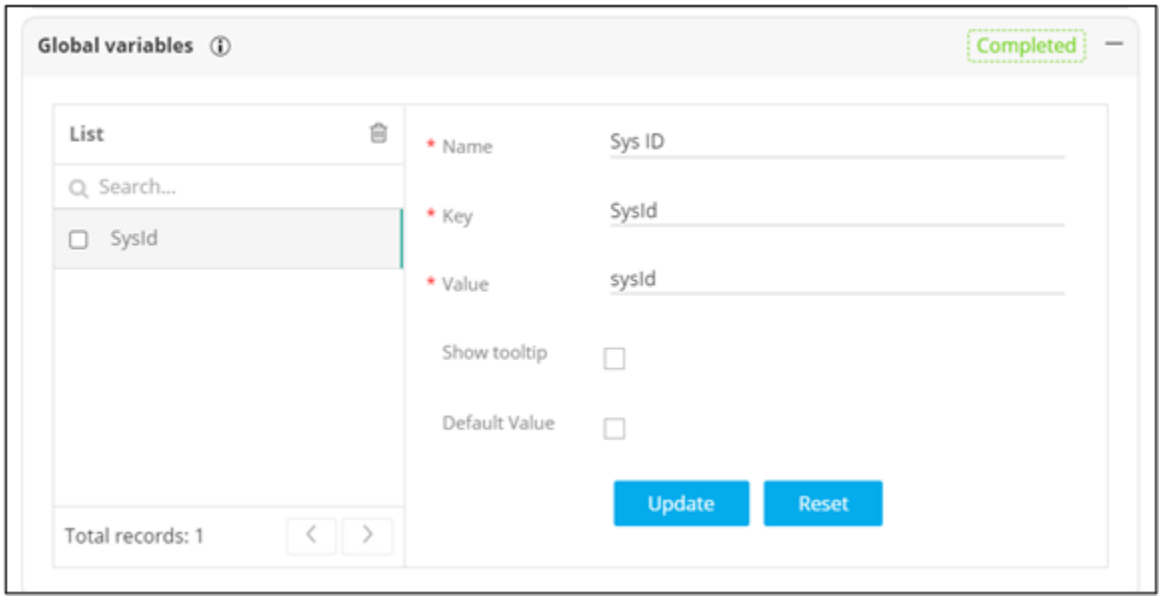


8. To fetch the ticket number, under **Change Management**, from the **Change** folder, drag and drop the **Get** task.
9. In the **Get** task window, under **Properties**, in the **General** section, refer the global variable (`<%ticketNumber%>`) defined in the previous task.

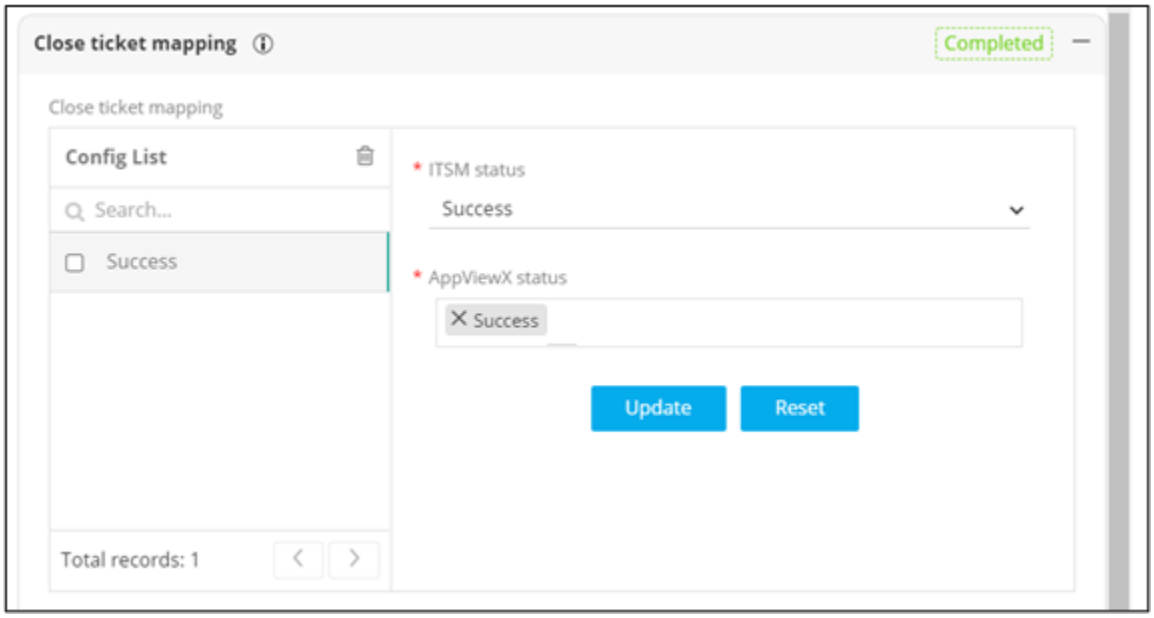


10. In the **Get** task window, under **Properties**, in the **Global variables** section, define the global variable (sysId) to be referenced in the next task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

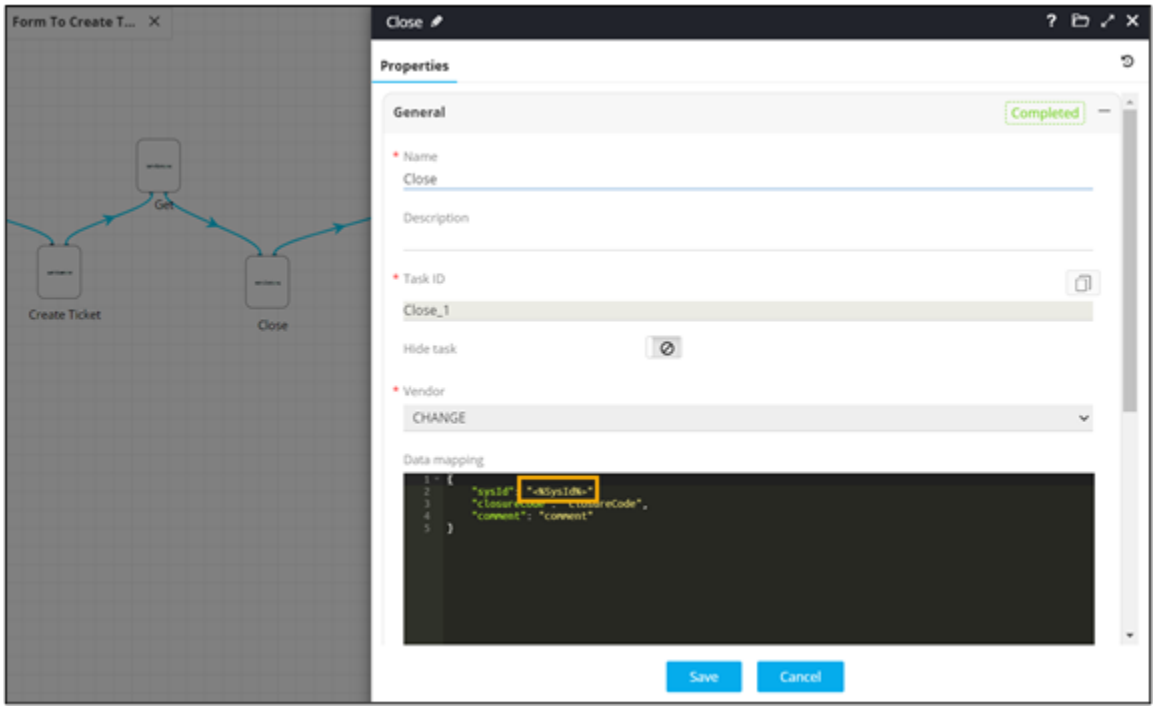


11. Under **Change Management**, from the **Change** folder, drag and drop the **Close** ticket task.
12. In the **Close** task window, under **Properties**, in the **Close ticket mapping** section, define the closure state mapping between AppViewX and ITSM.

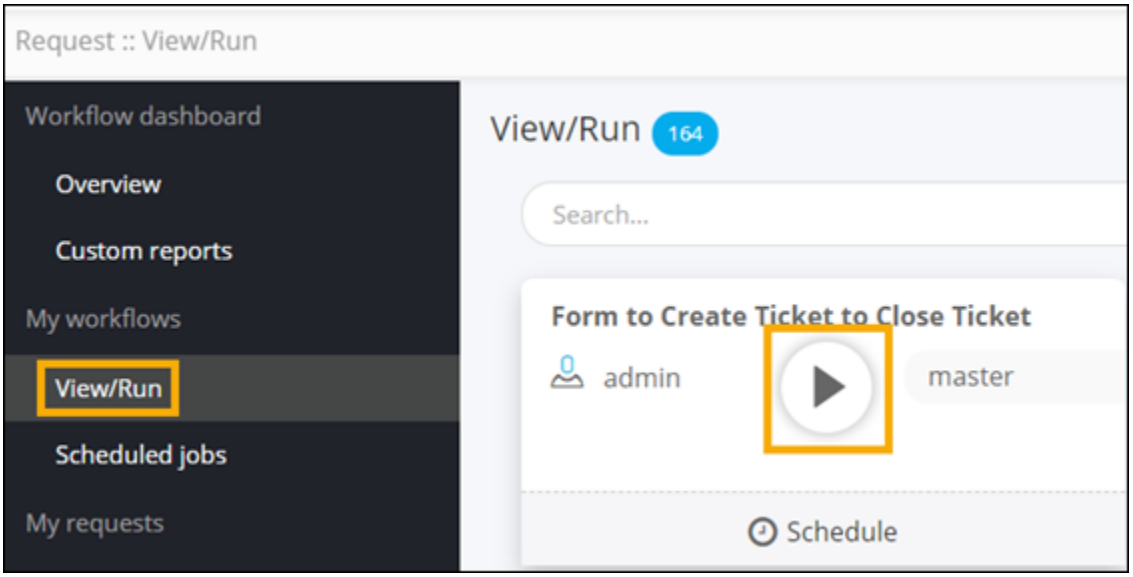


13. In the **Close** task window, under **Properties**, in the **General** section, refer the global variable (`<%sysID %>`) defined in the previous task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

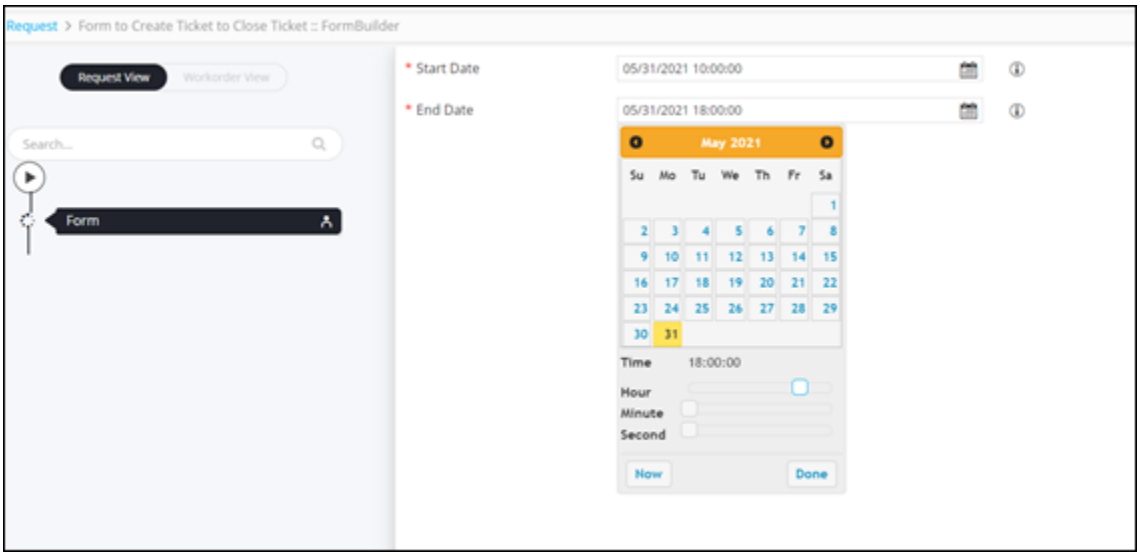


14. Connect and enable the workflow.
15. Trigger the workflow from the Request :: View/Run page.

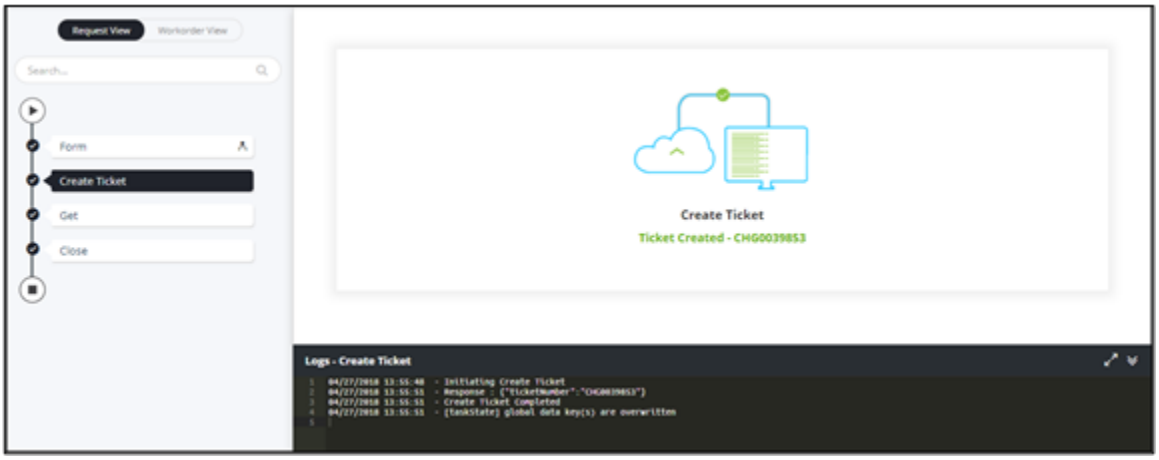


16. Enter the start and end date via the form.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

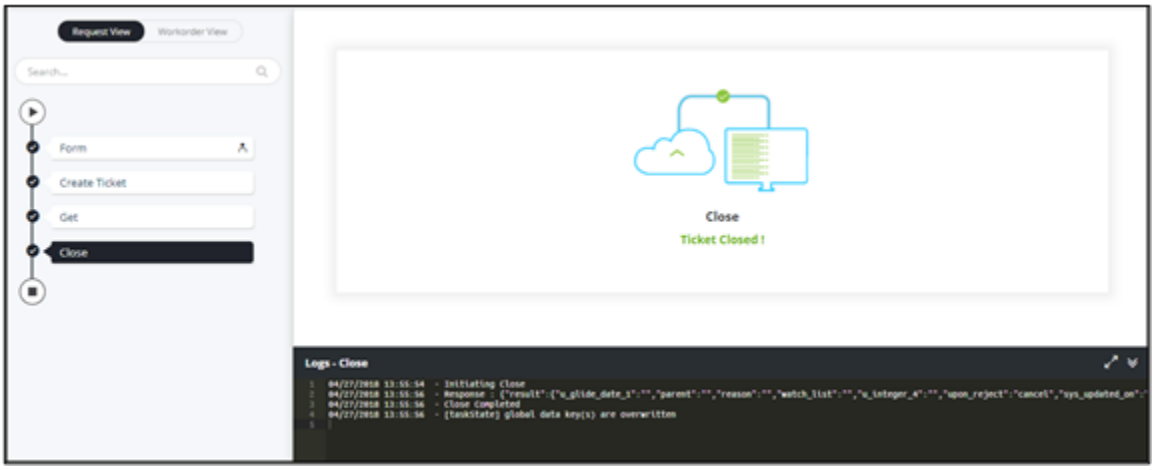


17. Click **Submit**.
    - Change ticket created.



    - Change ticket closed.

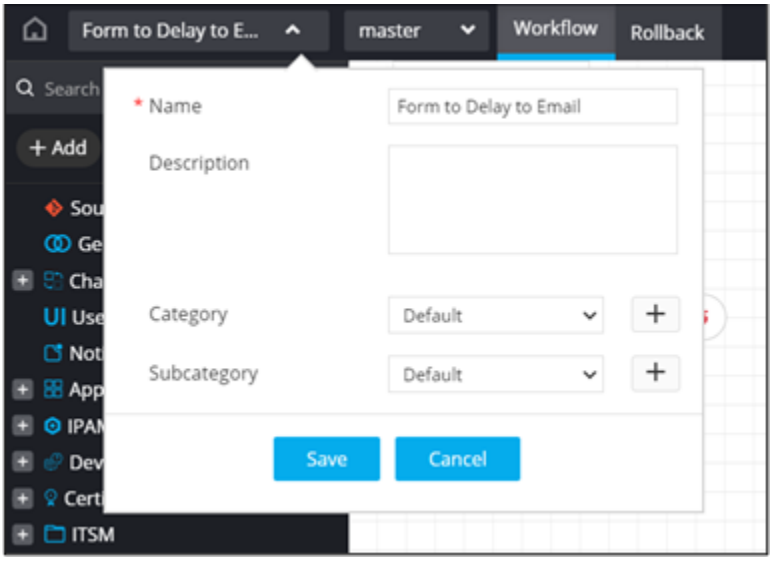
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



How to connect Form to Delay to Email

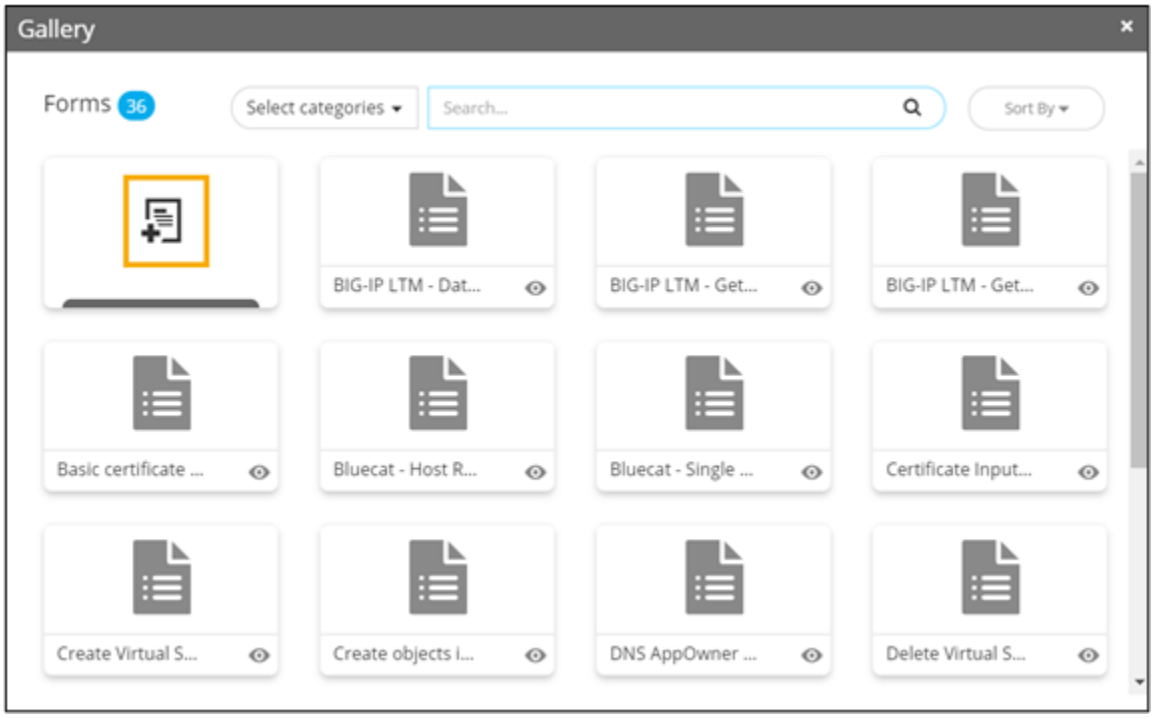
You can design workflows to get inputs from a user and trigger an email with a delay of say 1 minute.

1. Design a new workflow.



2. From the **User Interface** section, drag and drop a **Form** task.
3. In the **Gallery** window, to design a new form, click .

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

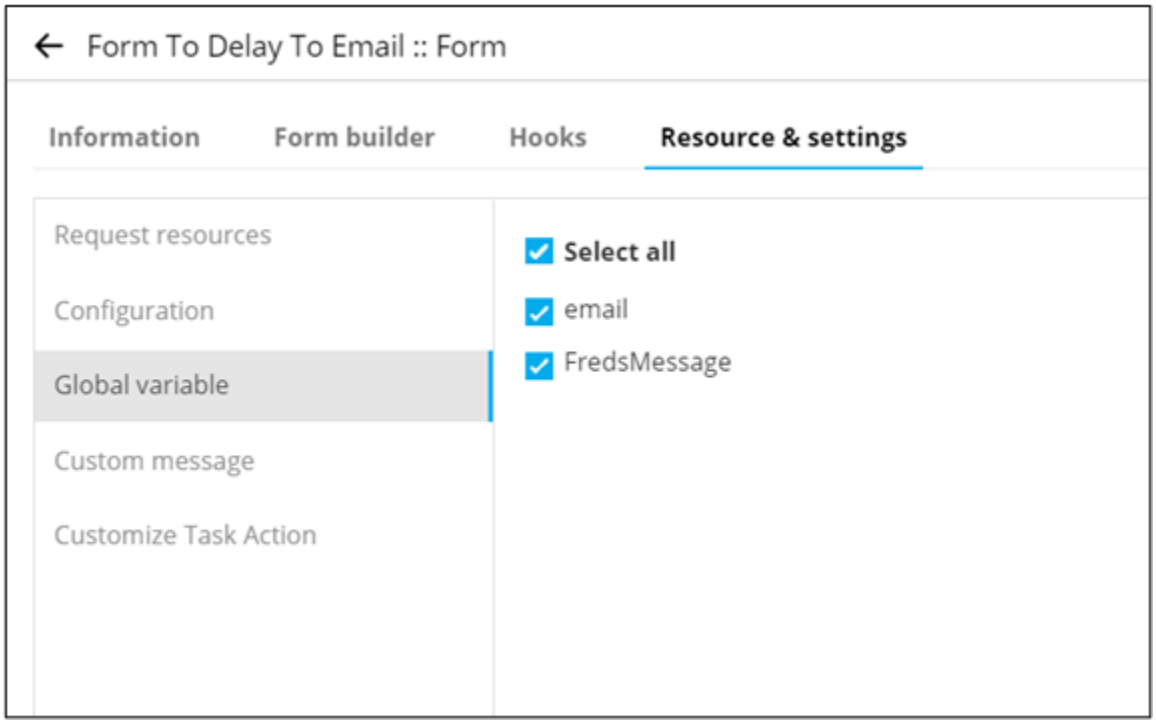


4. Under the **Form builder** tab, define the form fields to get basic inputs from the user.

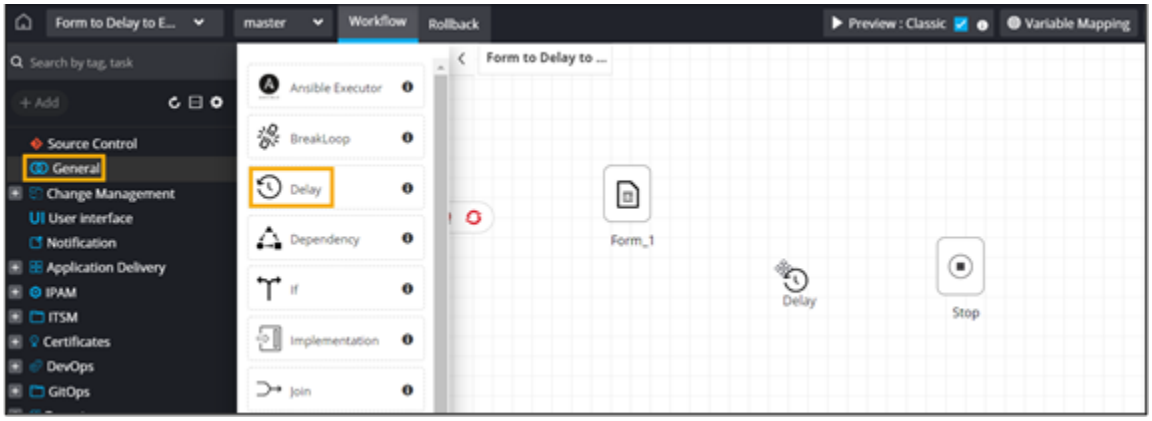
Note: For more information on adding form fields, click here.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

5. Under the **Resource & Settings** tab, define the field values as global variables.

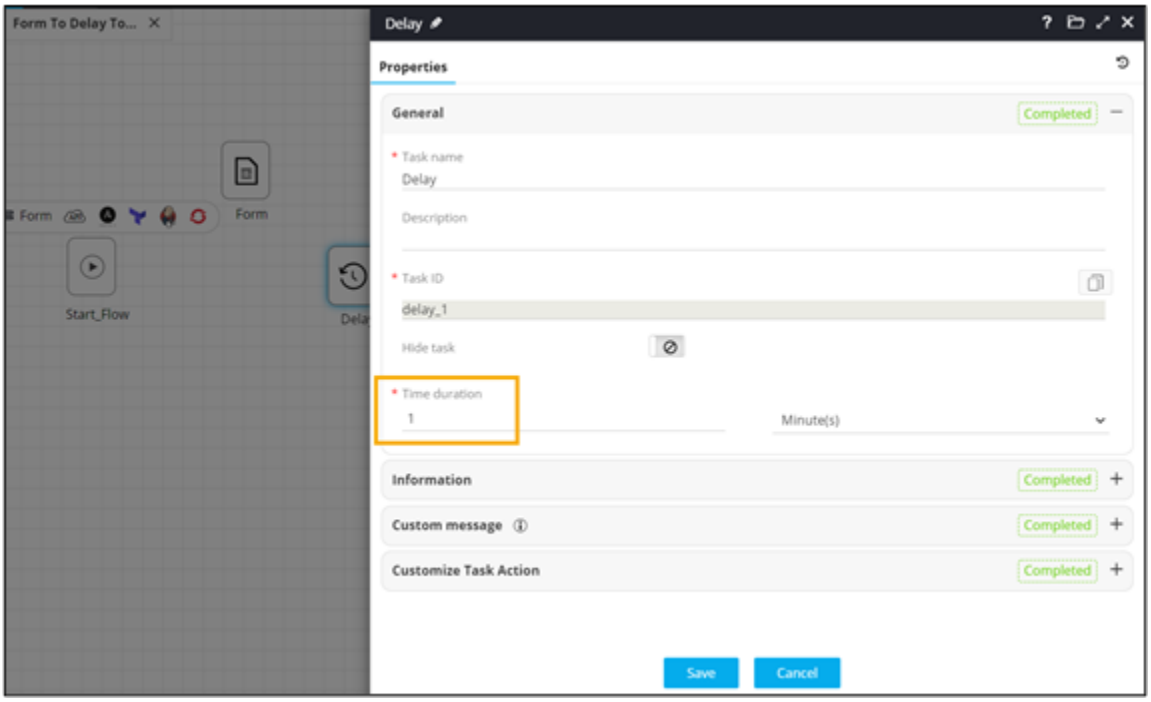


6. From the **General** section, drag and drop the **Delay** task.

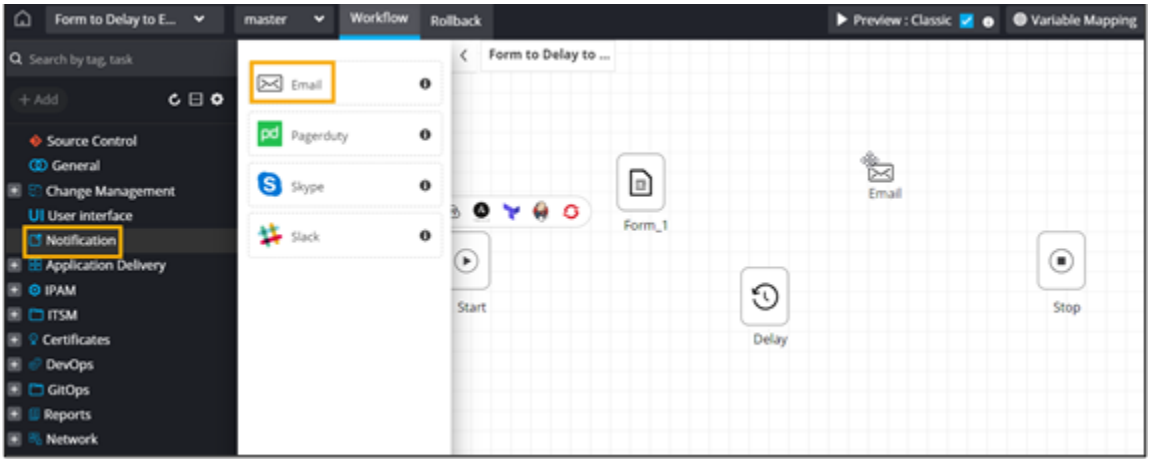


7. In the **Delay** task window, under **Properties**, in the **General** section, set the delay to 1 minute.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

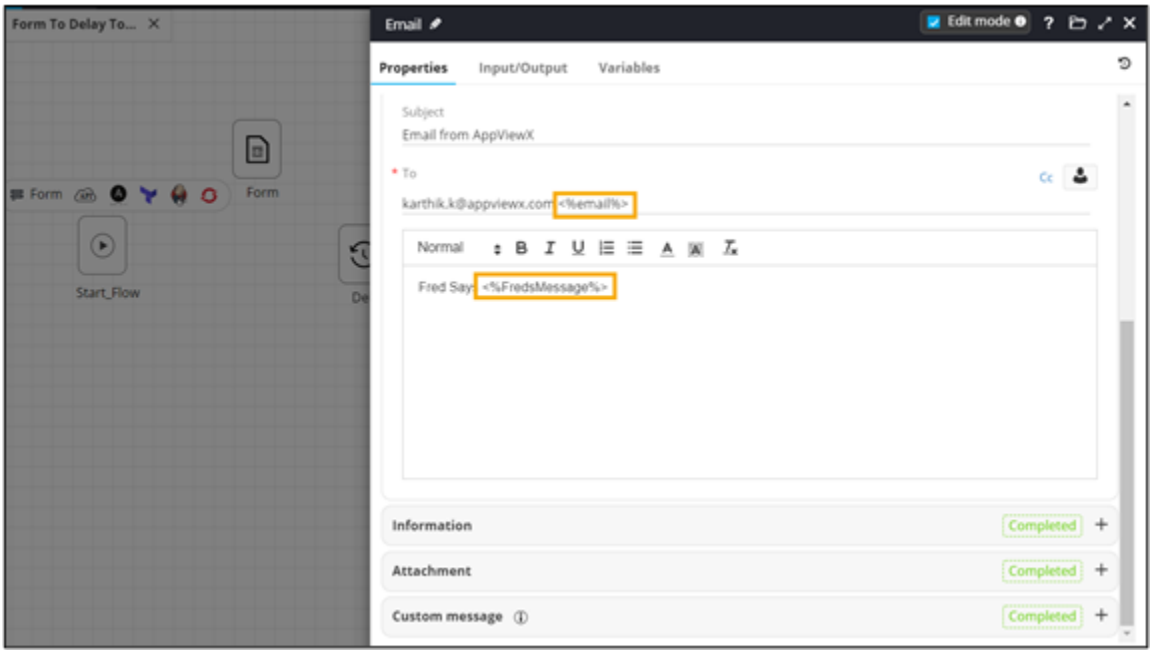


8. From the **Notification** section, drag and drop the **Email** task.

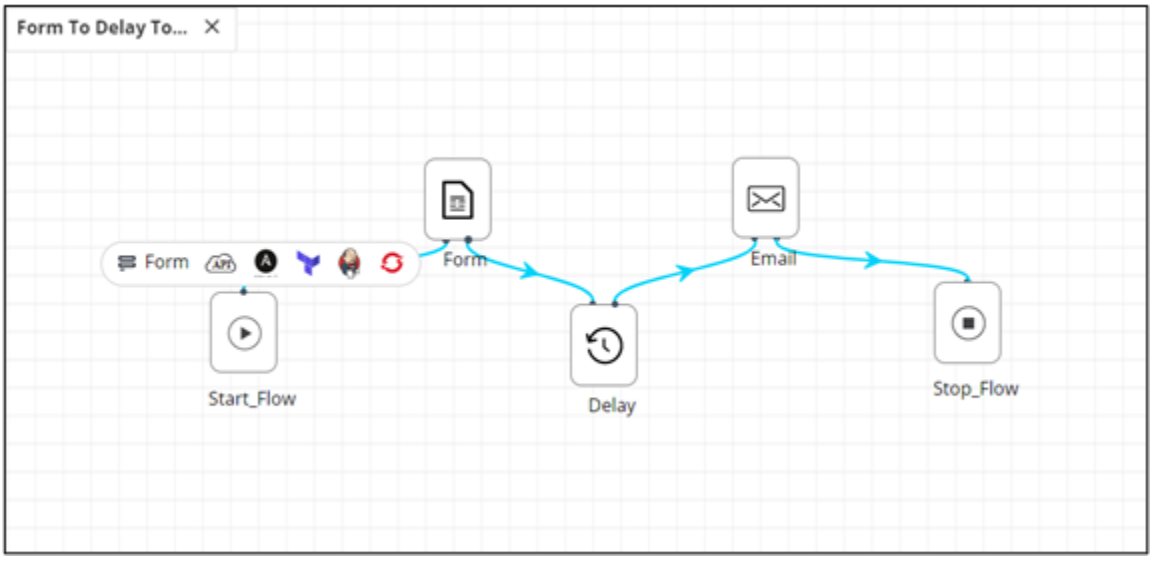


9. In the **Email** task window, under **Properties**, refer the global variables from the previous (form) task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

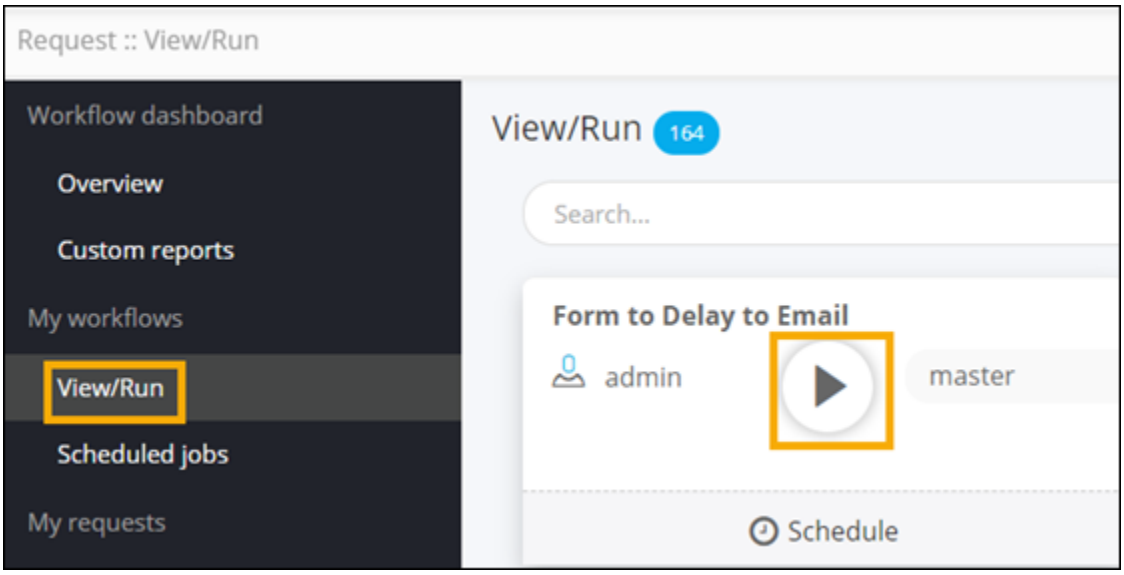


10. Connect and enable the workflow.



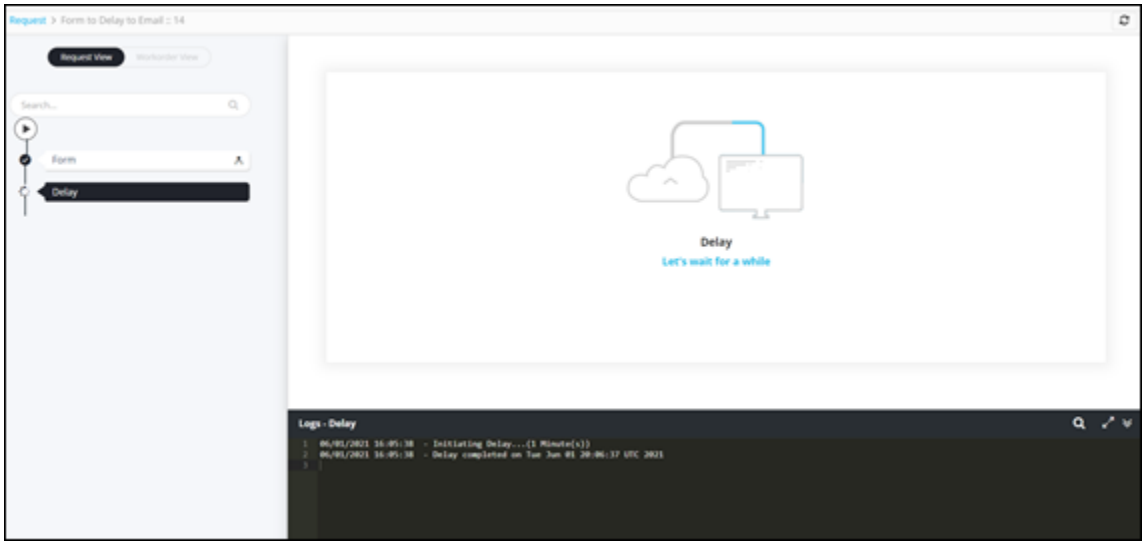
11. Trigger the workflow from the Request :: View/Run page.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



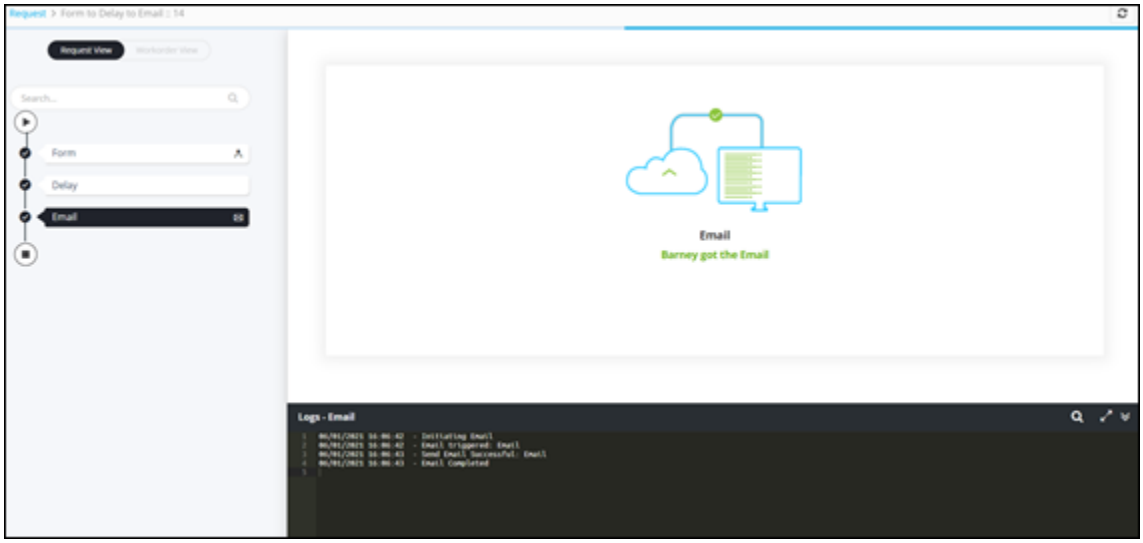
- Inputs received through the form.

- Delay of 1 minute is initiated.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

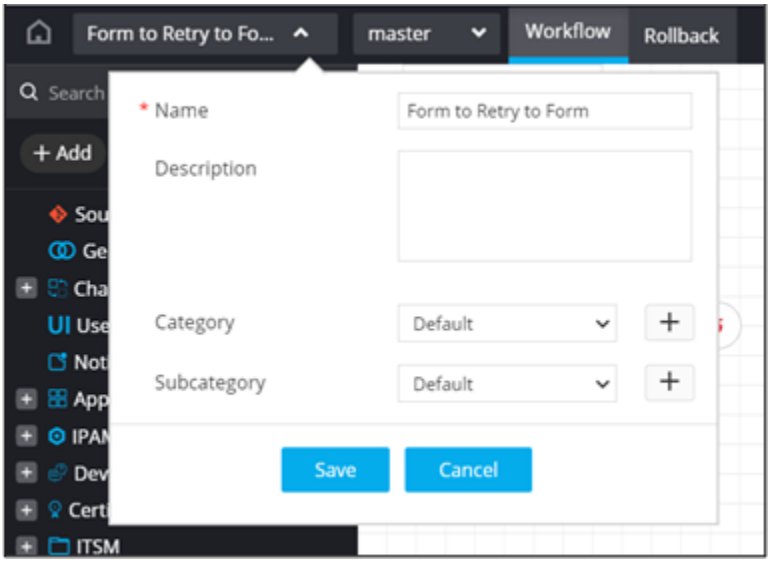
- Email notification sent after the 1 minute delay.



How to connect Form to Retry to Form

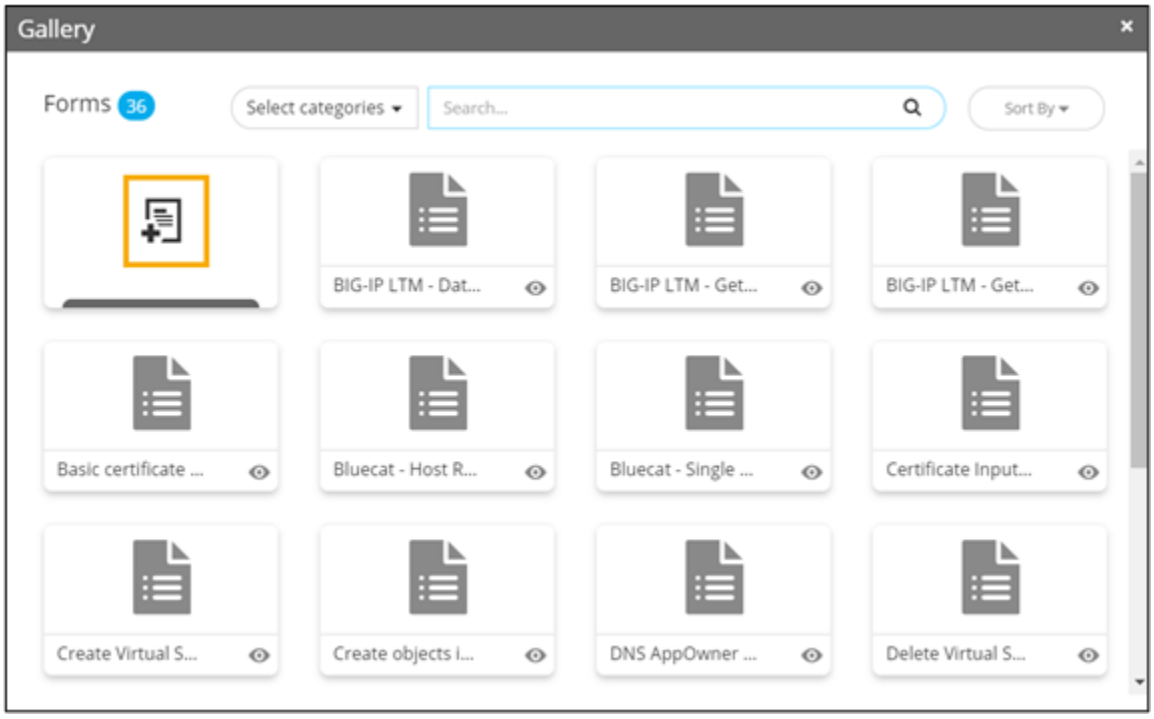
You can create a workflow to check for a specific input in the form, based on which a decision will be made to retry and pass the value to the next form task.

1. Design a new workflow.

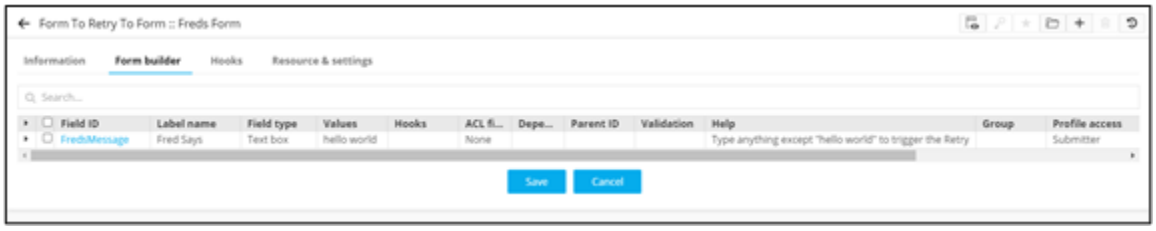


2. From the **User Interface** section, drag and drop a **Form** task.
3. In the **Gallery** window, to design a new form, click .

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



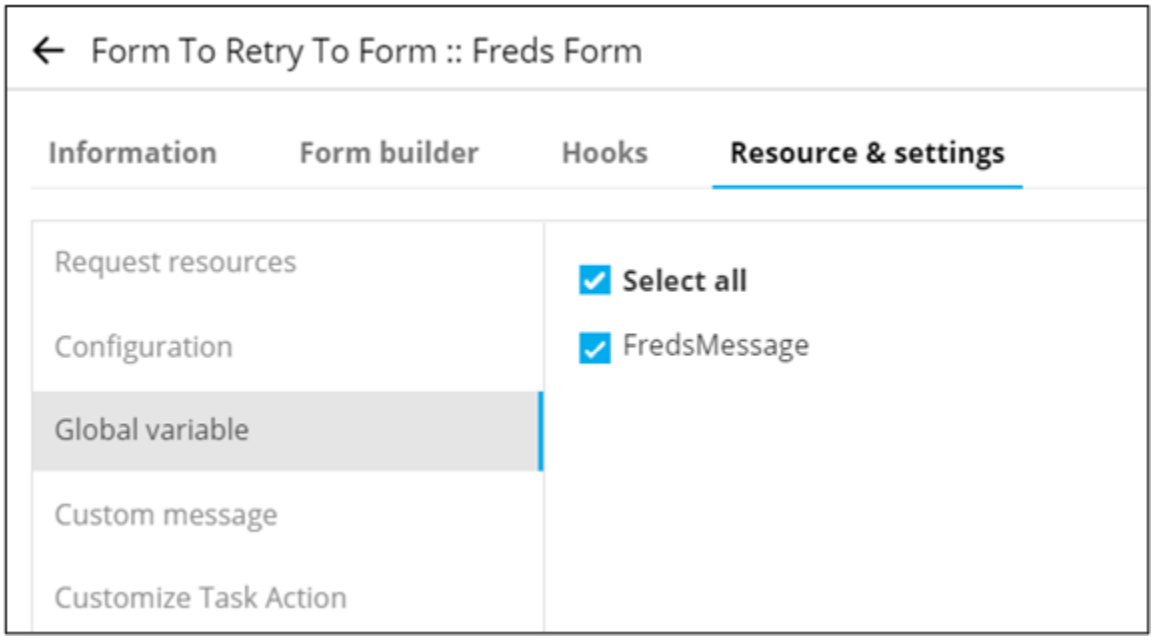
4. Under the **Form builder** tab, define the form fields to get inputs from a user.



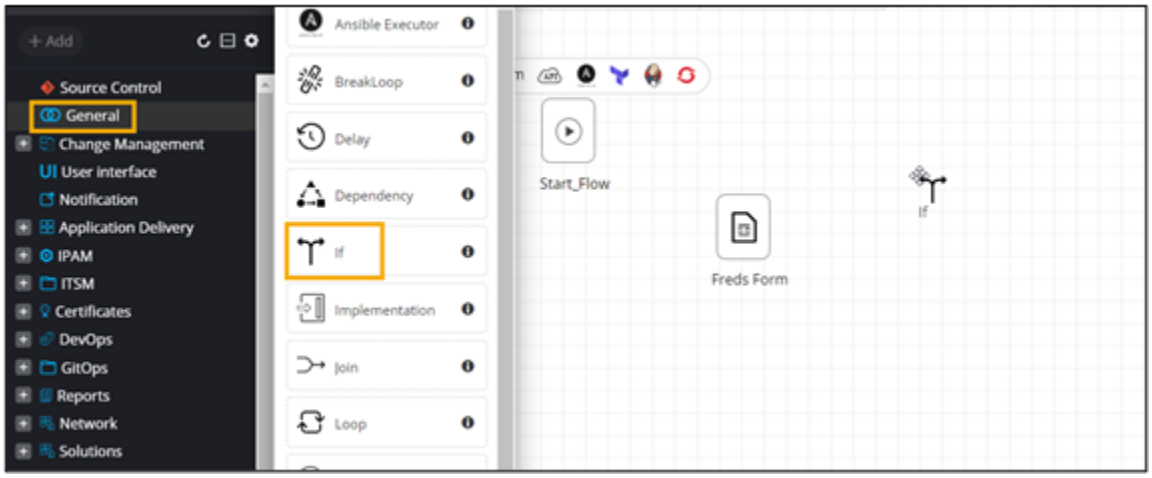
Note: For more information on adding form fields, click here.

5. Under the **Resource & Settings** tab, declare the input value as a global variable to be used for initiating retry.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

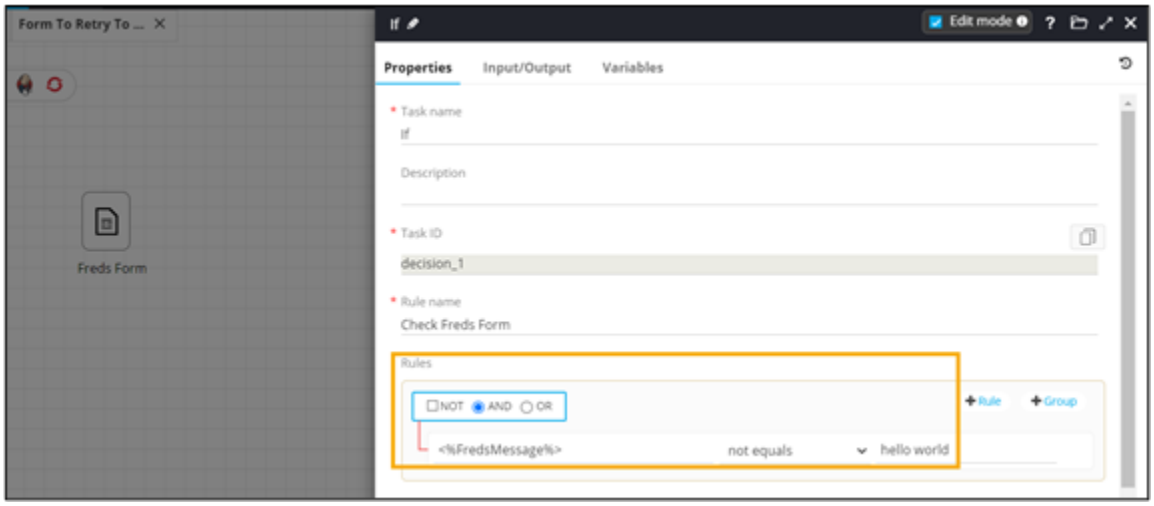


6. From the **General** section, drag and drop the **If** task (decision task).

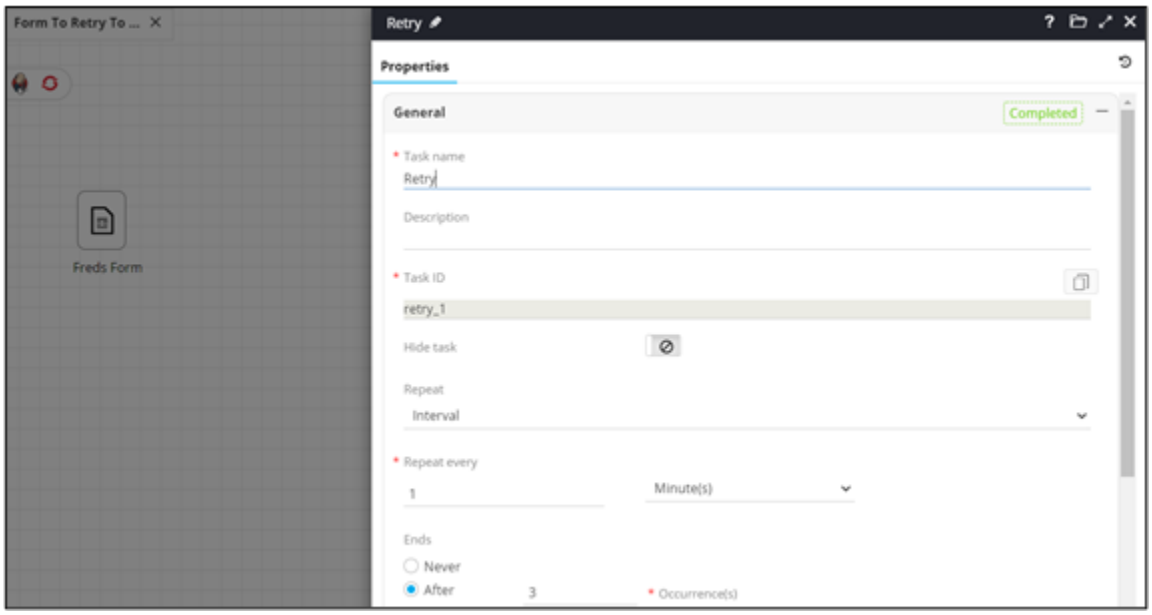


7. In the **If** task window, under **Properties**, define a rule to check if the input string entered in the form matches the value defined in the form field (Hello world). If not, initiate a call to try.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



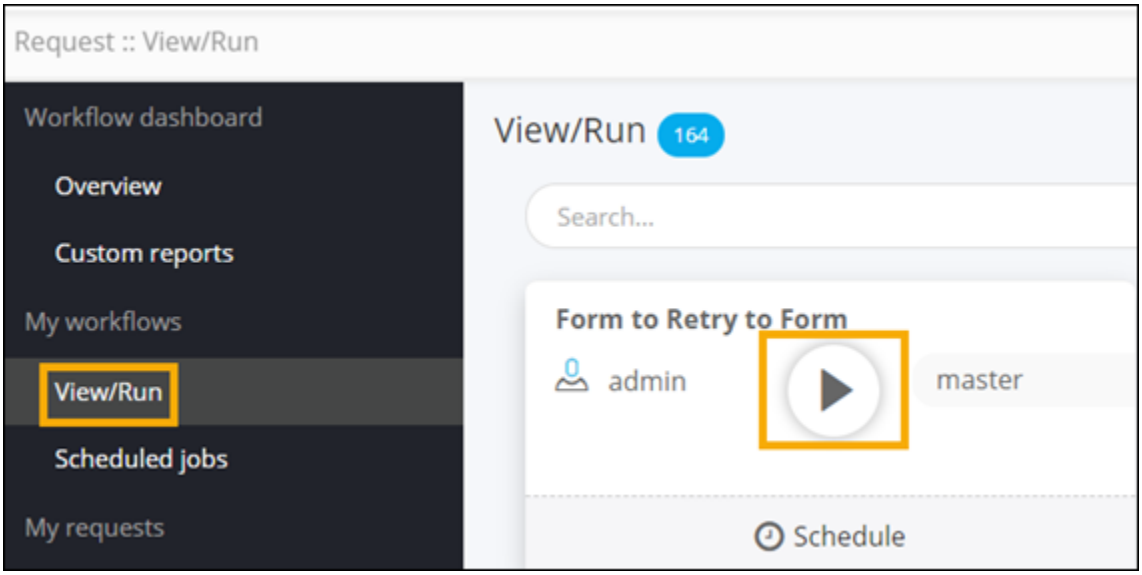
8. To initiate retry based on the input string, from the **General** section, drag and drop the **Retry** task.
9. In the **Retry** task window, under **Properties**, in the **General** section, define the parameters for the **Retry** task.



10. To publish the values from the first form task, drag and drop another **Form** task.
11. Under the **Form builder** tab, refer the global variable value.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

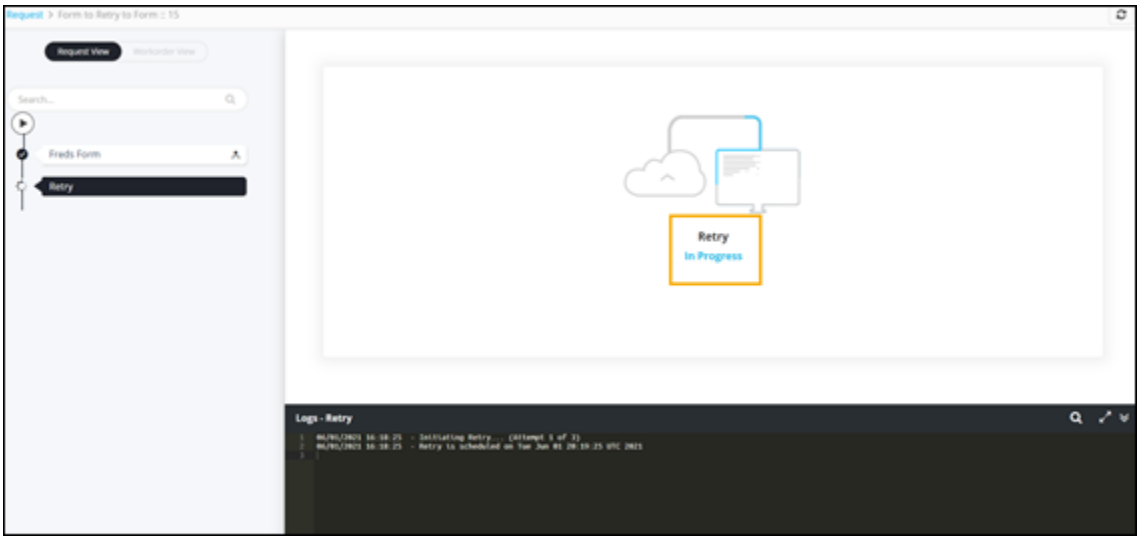
12. Connect and enable the workflow.
13. Trigger the workflow from the Request :: View/Run page.



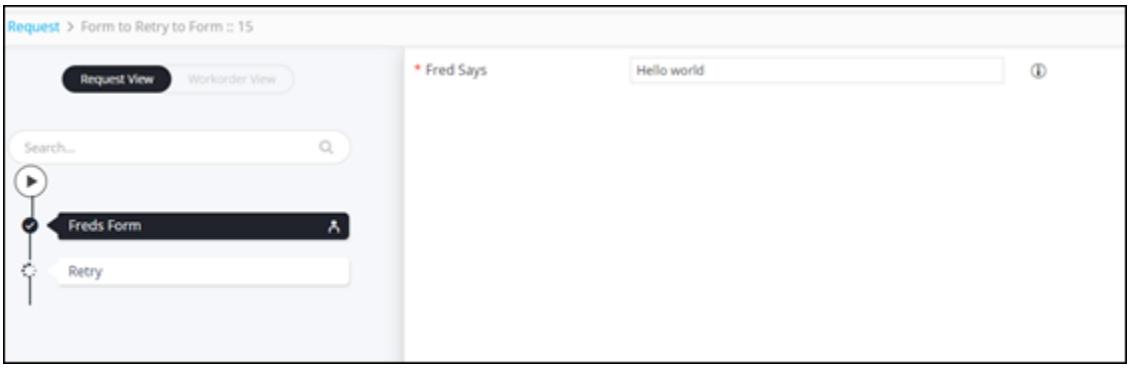
14. Type any input except 'Hello world' to initiate the retry.

Retry is initiated.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



15. Type an input to match the defined input - hello world.



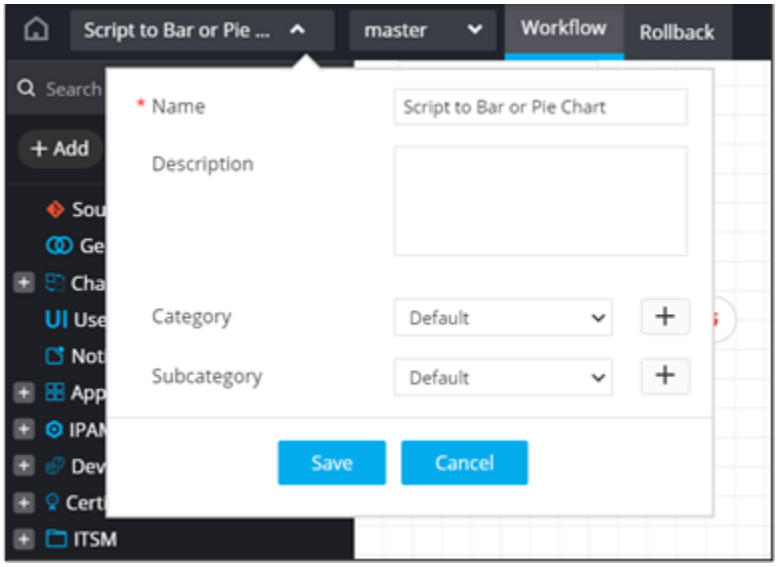
Output is published based on the given input.

How to connect Script to Bar or Pie Chart

You can create a workflow to design a bar/pie chart based on the output from a Script logic.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

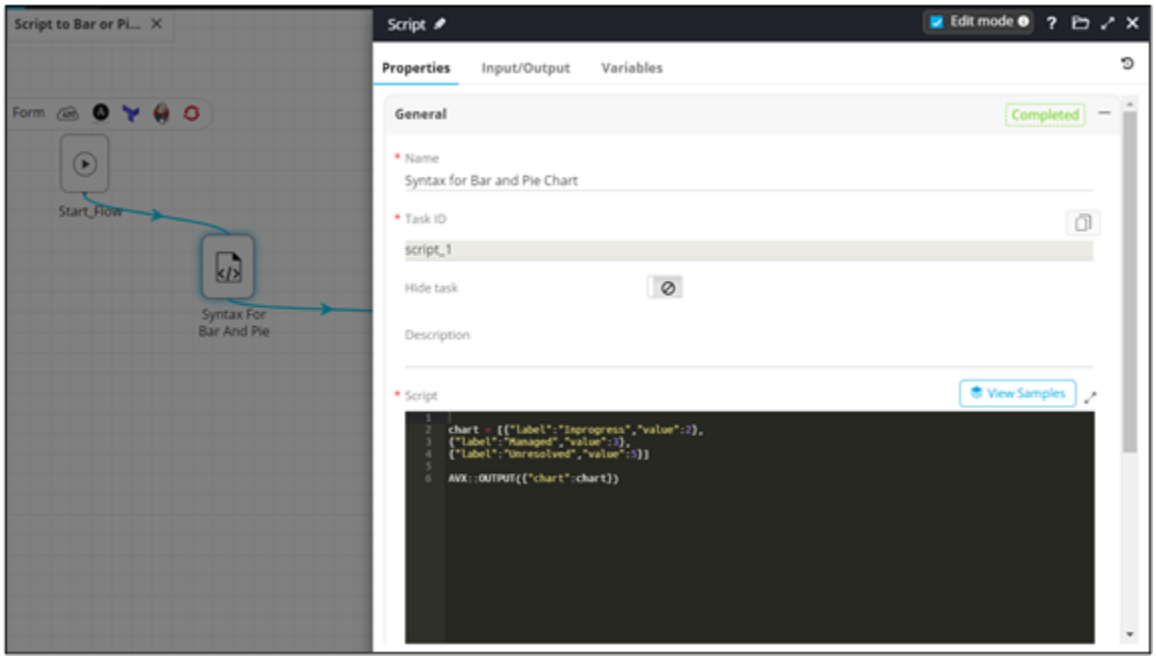
1. Design a new workflow.



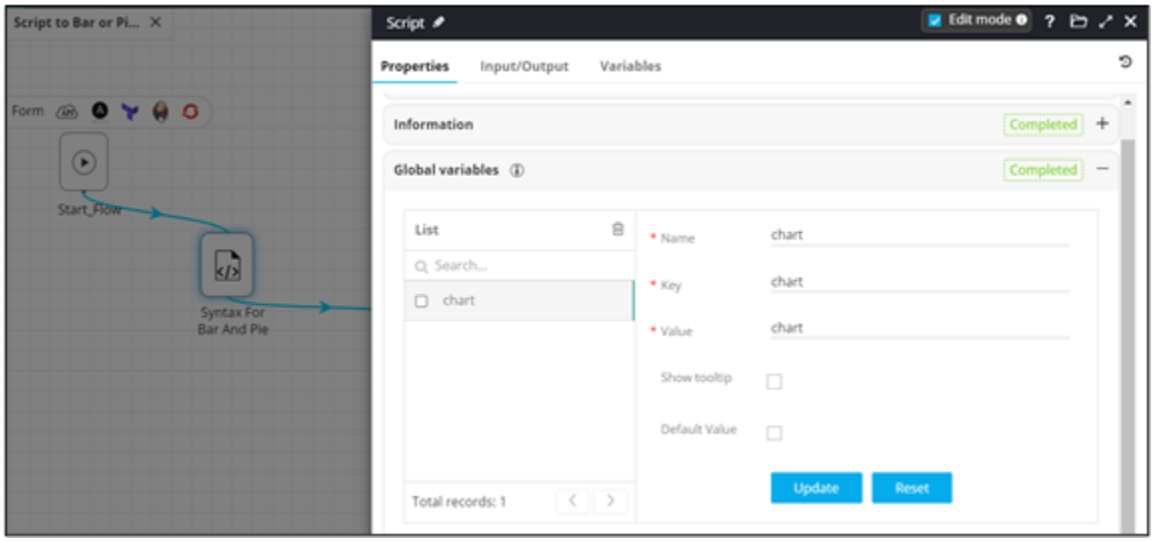
2. From the **General** section, drag and drop a **Script** task.
3. In the **Script** task window, under **Properties**, in the **General** section, define the script logic.

```
chart = [{"label":"Inprogress","value":2},

{"label":"Managed","value":3},

{"label":"Unresolved","value":5}]


AVX::OUTPUT({"chart":chart})
```

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

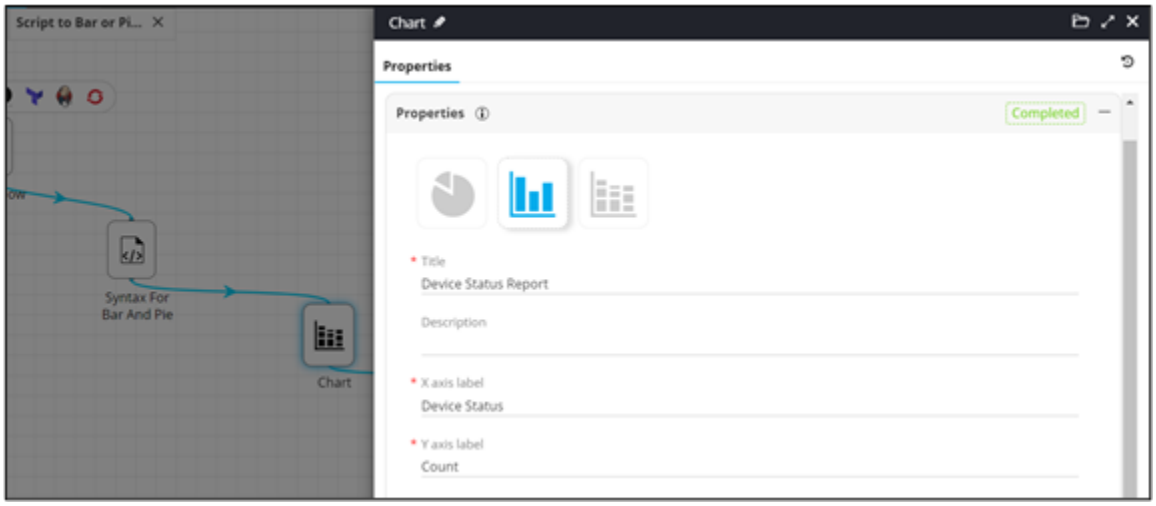


4. In the **Script** task window, under **Properties**, in the **Global variables** section, define the output of the Script task (chart) as a global variable.

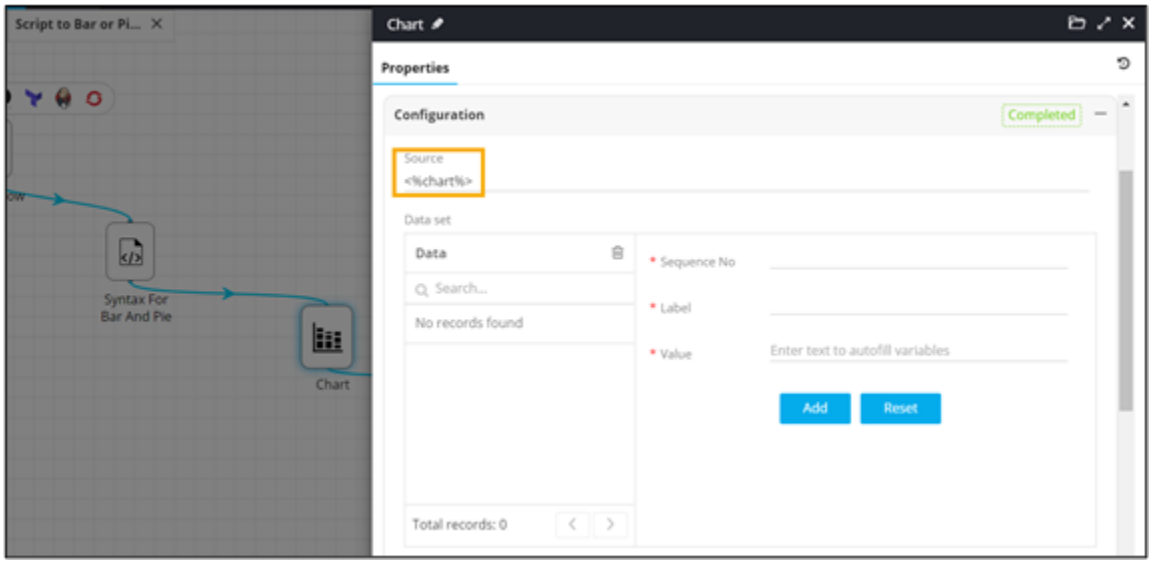


5. From the **User Interface** section, drag and drop the **Chart** task.
6. In the **Chart** task window, under **Properties**, select chart type as .
7. Define the chart title.
8. Define the X-axis and Y-axis labels.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

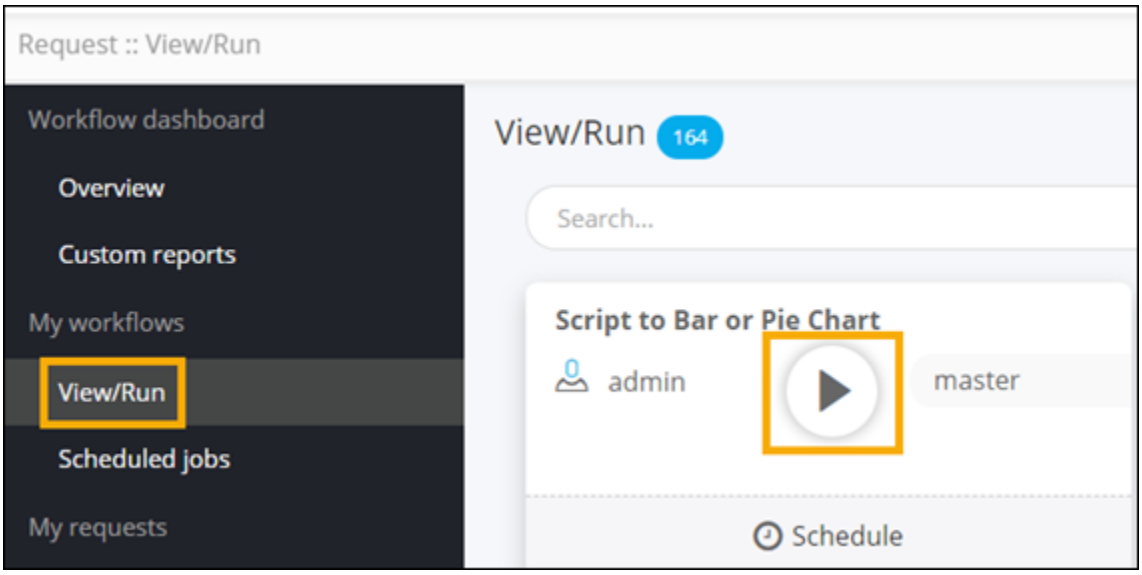


9. In the **Chart** task window, under **Properties**, in the **Configuration** section, refer the variable defined in the Script task (`<%chart%>`).



10. Connect and enable the workflow.
11. Trigger the workflow from the Request :: View/Run page.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



• Script is executed.

• Output is displayed in the form of a Bar chart.

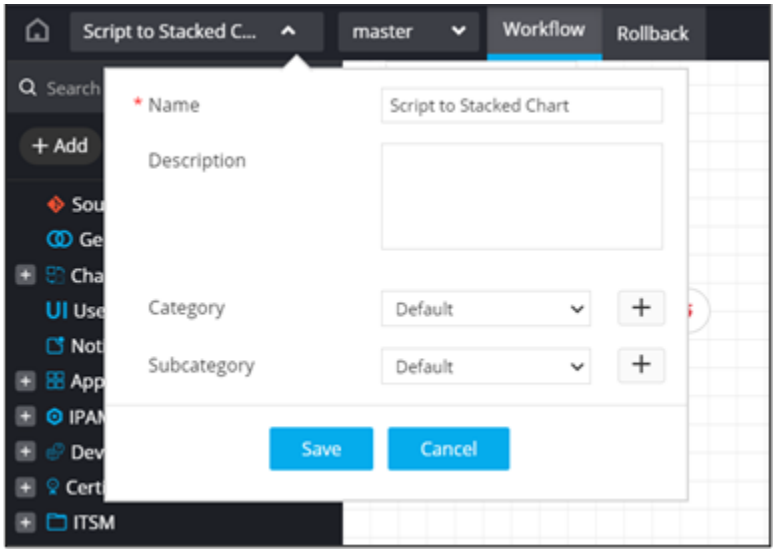
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Note: To view the output as a Pie chart, select Pie when defining the parameters for the Chart task.

How to connect Script to Stacked Chart

You can design a workflow to design a stacked chart based on inputs from a script logic.

1. Design a new workflow.

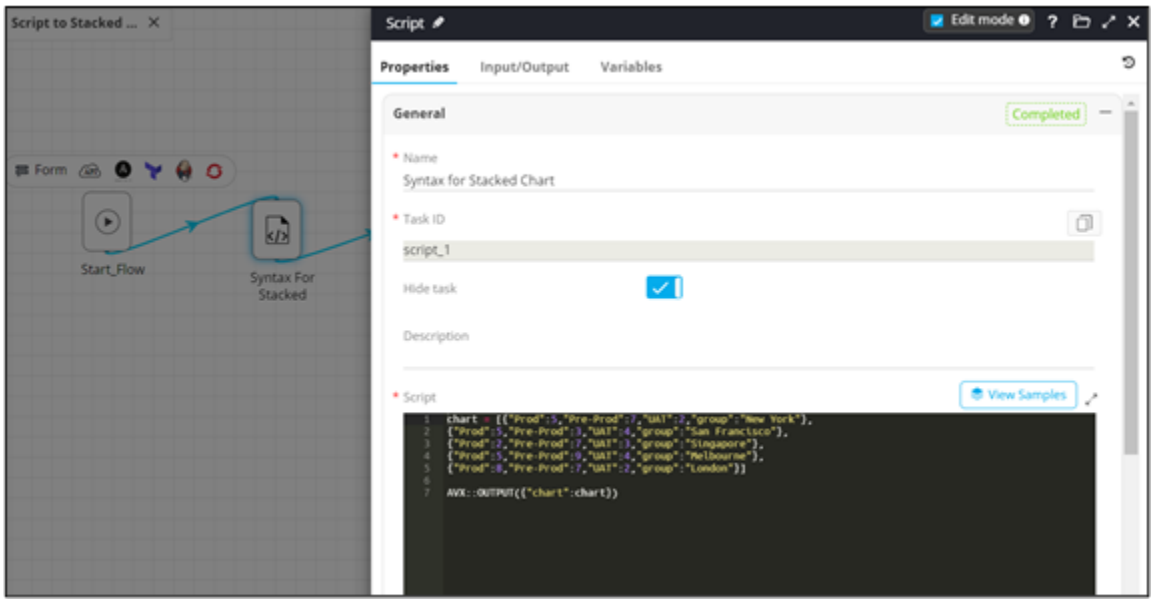


2. From the **General** section, drag and drop a **Script** task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

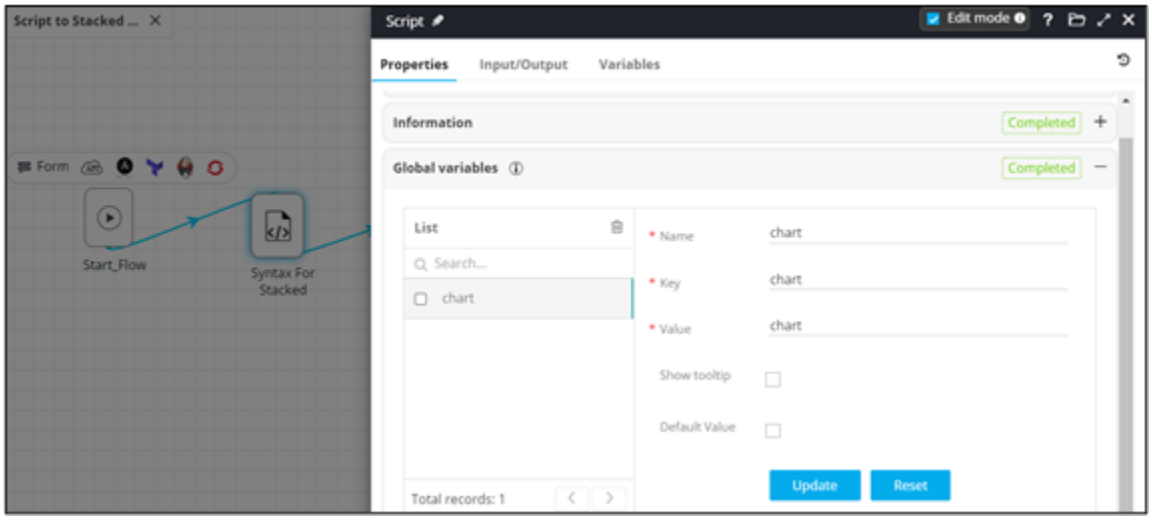
3. In the **Script** task window, under **Properties**, in the **General** section, define the script logic.

```
chart = [{"Prod":5,"Pre-Prod":7,"UAT":2,"group":"New York"},

{"Prod":5,"Pre-Prod":3,"UAT":4,"group":"San Francisco"},

{"Prod":2,"Pre-Prod":7,"UAT":3,"group":"Singapore"},

{"Prod":5,"Pre-Prod":9,"UAT":4,"group":"Melbourne"},

{"Prod":8,"Pre-Prod":7,"UAT":2,"group":"London"}]


AVX::OUTPUT({"chart":chart})
```

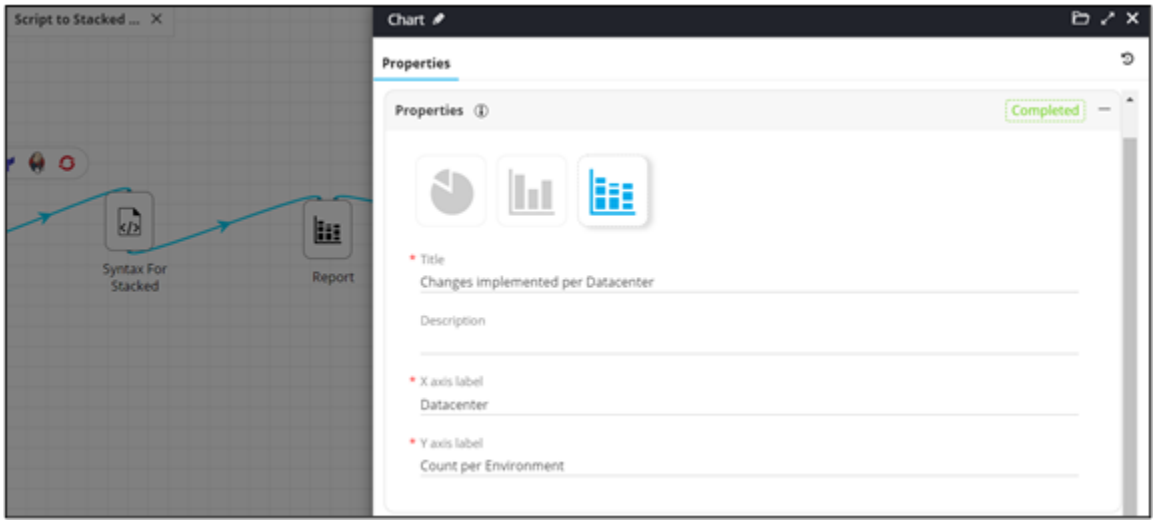


4. In the **Script** task window, under **Properties**, in the **Global variables** section, define the output of the script as a global variable to be referenced in the Chart task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

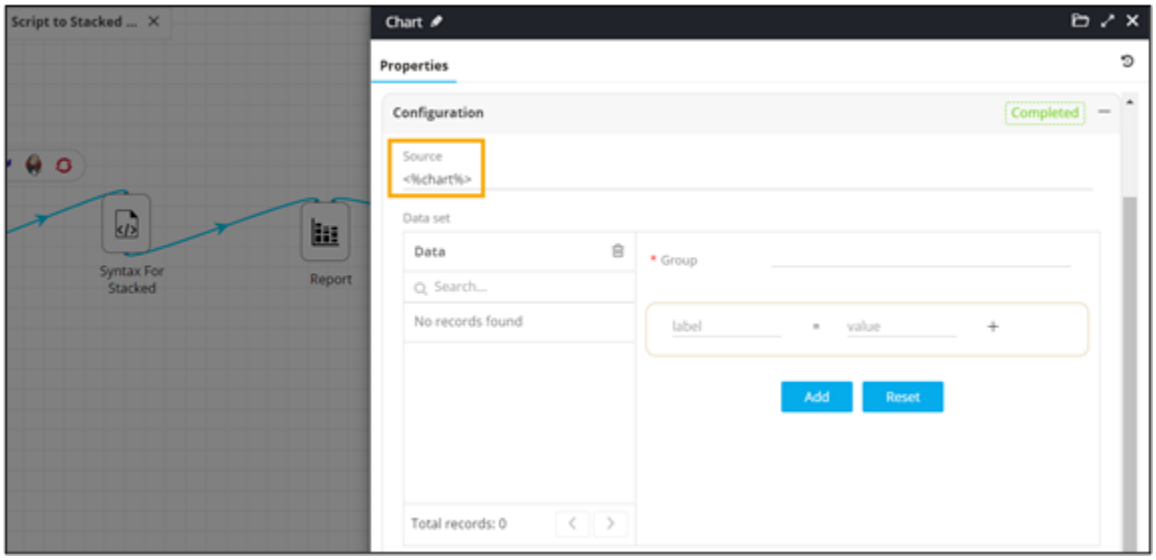


5. From the **User Interface** section, drag and drop the **Chart** task.
6. In the **Chart** task window, under **Properties**, select the Chart type as .
7. Define the Chart title.
8. Define the values for X-axis and Y-axis.

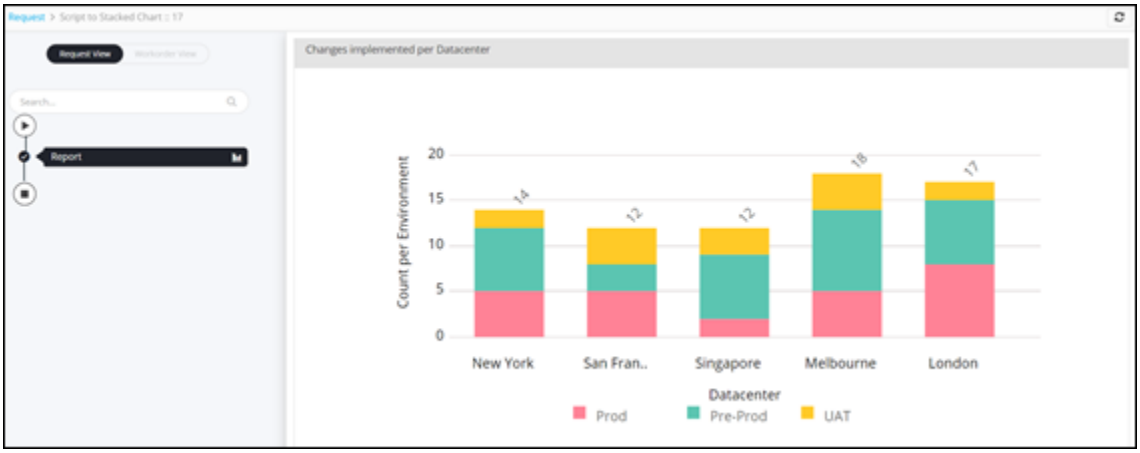


9. In the **Script** task window, under **Properties**, in the **Configuration** section, refer the global variable declared in the previous (Script) task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



10. Enable and trigger the workflow.
11. Execute the report to view Chart data.

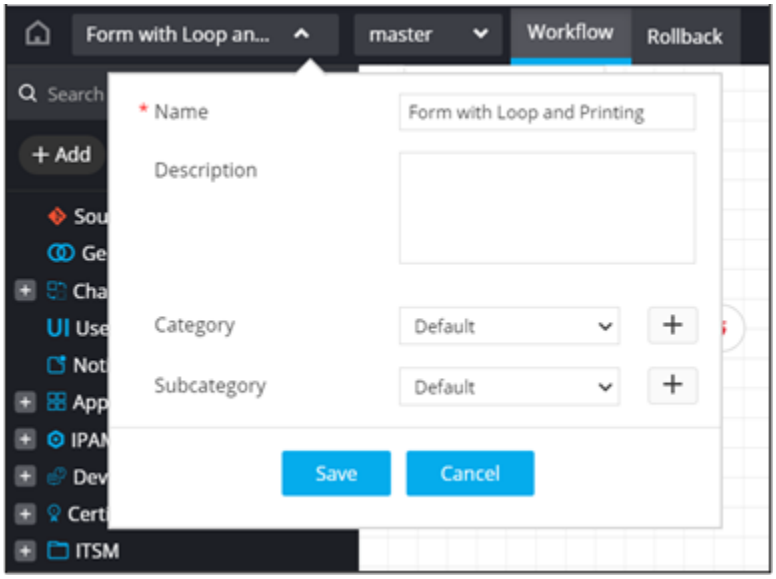


How to connect Form with Loop and Printing

You can design a workflow to get data from a form as input, iterate it based on the count and print it.

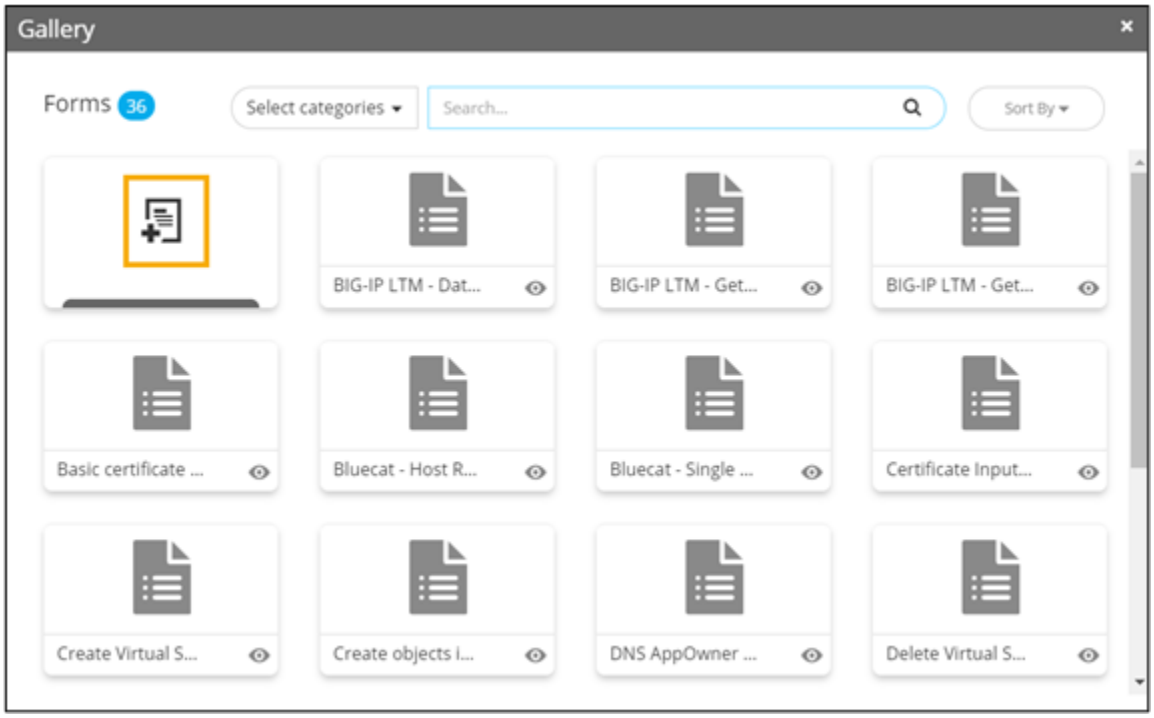
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. Design a new workflow.



2. From the **User Interface** section, drag and drop a **Form** task.

3. In the **Gallery** window, to design a new form, click .



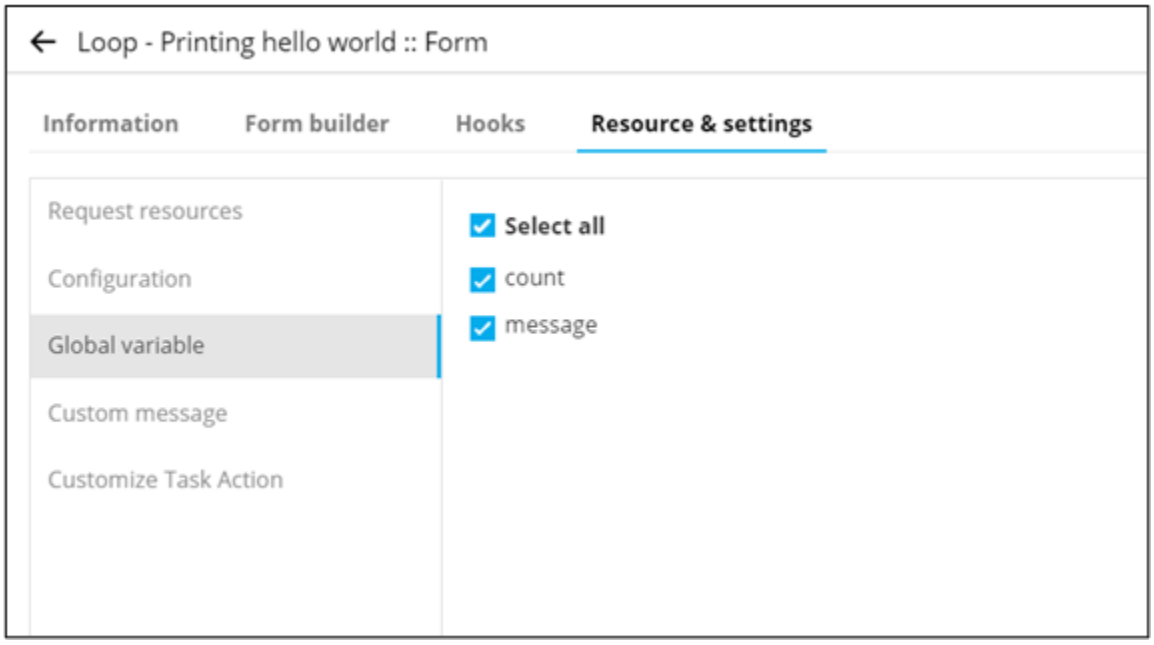
4. Under the **Form builder** tab, add a form field to define the number of times the message needs to be iterated for and then printed.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

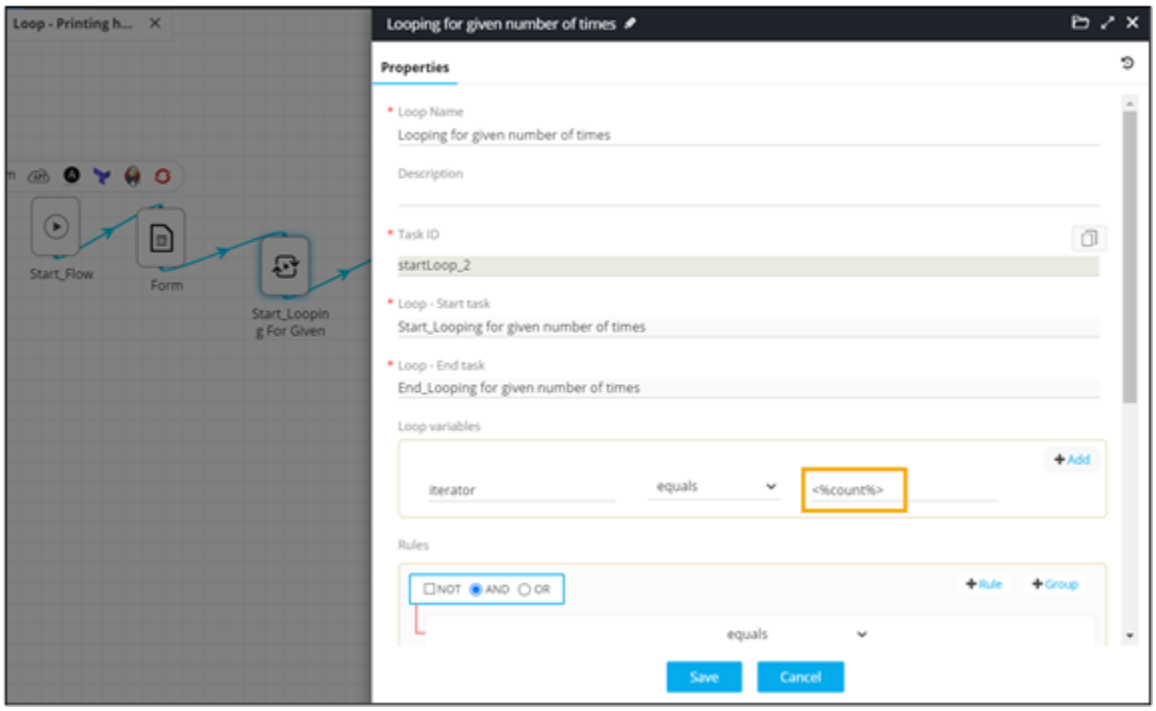
Note: For more information on adding form fields, click here.

5. Under the **Resource & Settings** tab, declare the field values as global variables to be referenced into the next task(s).

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



6. From the **General** section, drag and drop the **Loop** task.
7. In the Loop task window, under **Properties**, define the number of times the iteration needs to happen in the Loop task.
8. Refer the variable captured as input from the Form task.

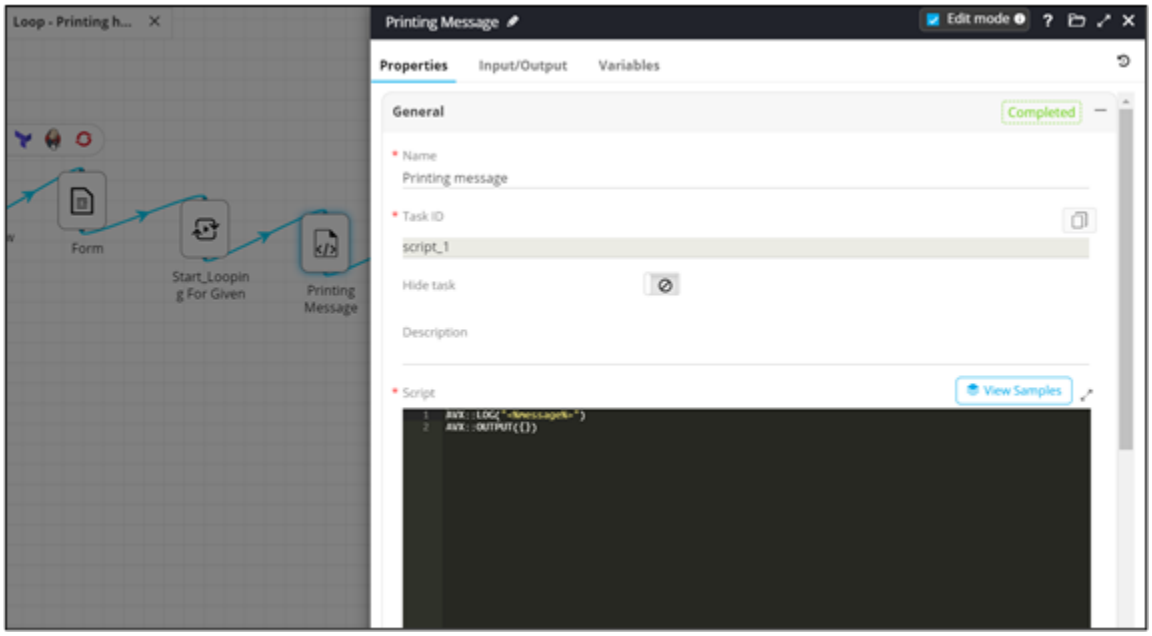


9. From the **General** section, drag and drop a **Script** task.

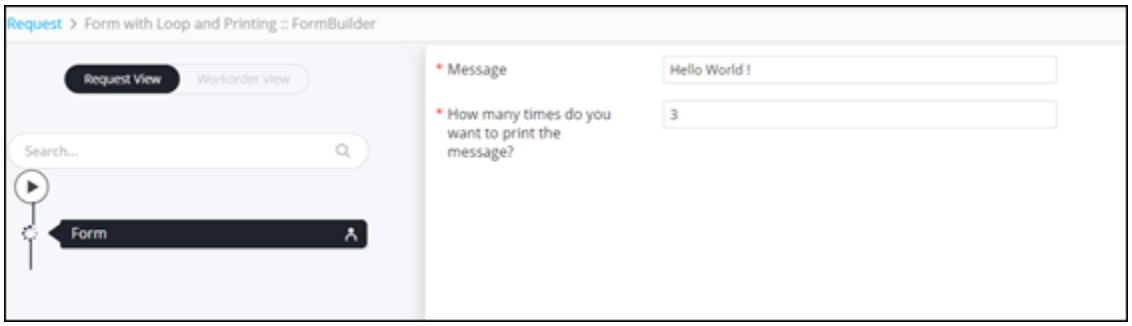
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

10. In the **Script** task window, under **Properties**, in the **General** section, define the script logic to print the message (captured from the Form task) based on the iteration count.

```
AVX::LOG("<%message%>")

AVX::OUTPUT({})
```

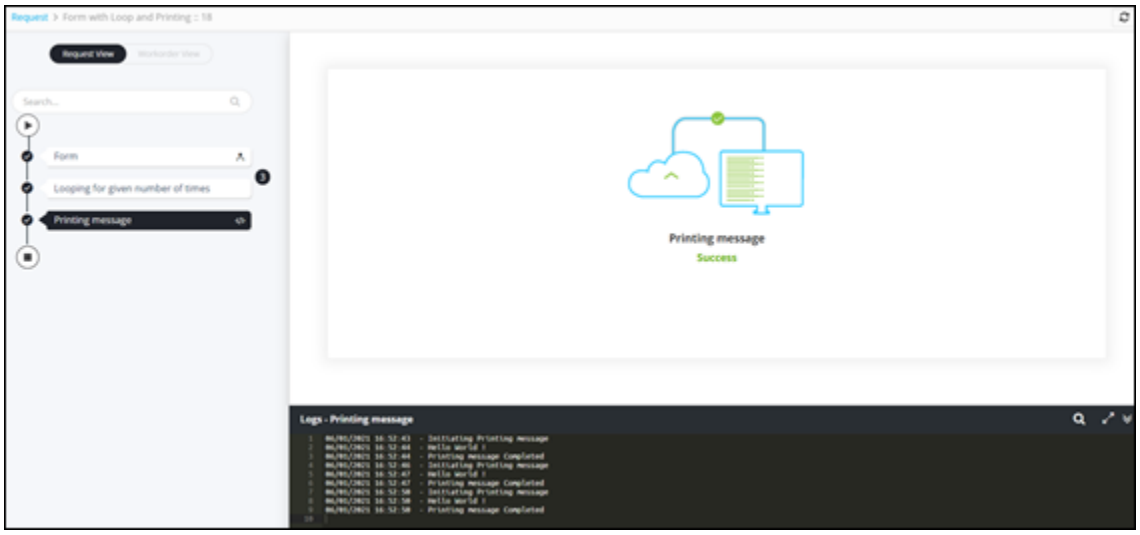


11. Enable and trigger the workflow.
    - Input and iteration count is received via the form.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

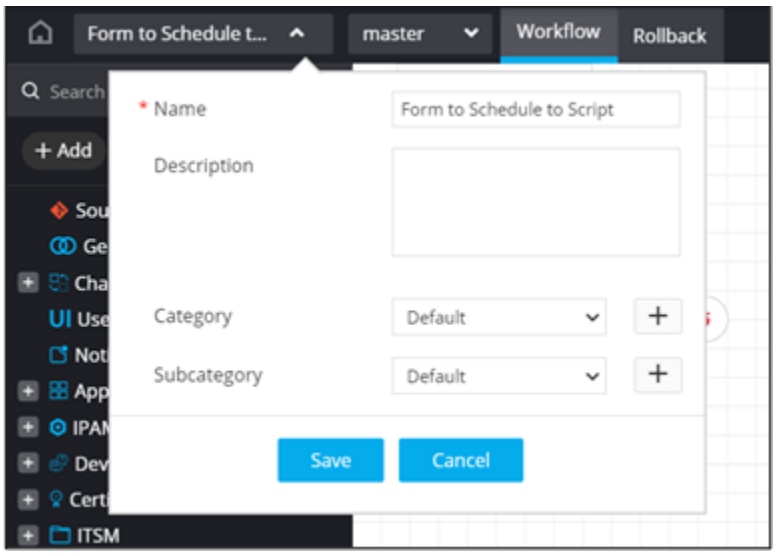
- Message is printed as per the defined iterations.



How to connect Form to Schedule to Script

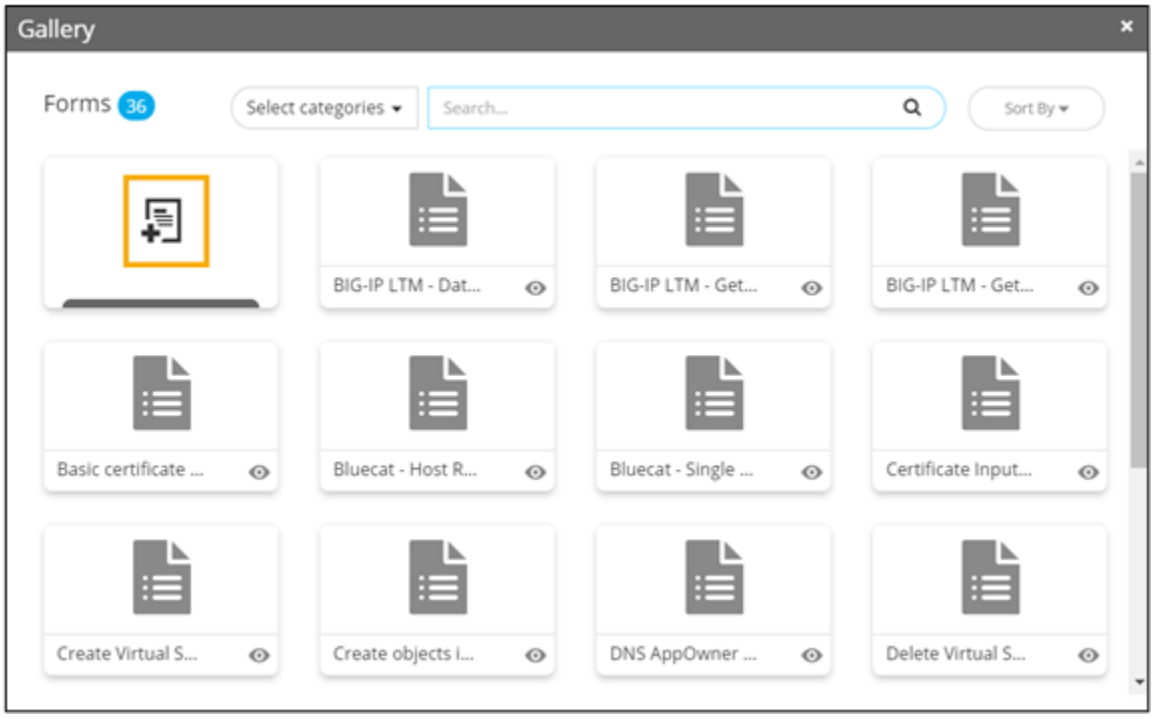
You can design a workflow to schedule one or more task(s) in between a workflow process.

1. Design a new workflow.

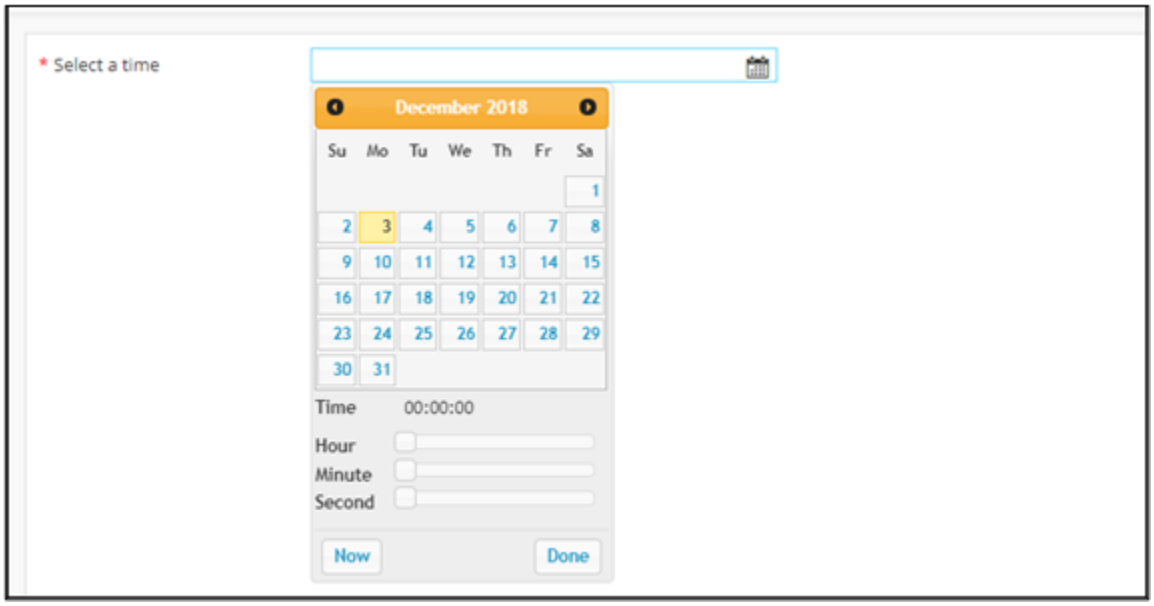


2. From the **User Interface** section, drag and drop a **Form** task.
3. In the **Gallery** window, to design a new form, click .

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



4. Under the **Form builder** tab, define a date element to manually get inputs via the form.

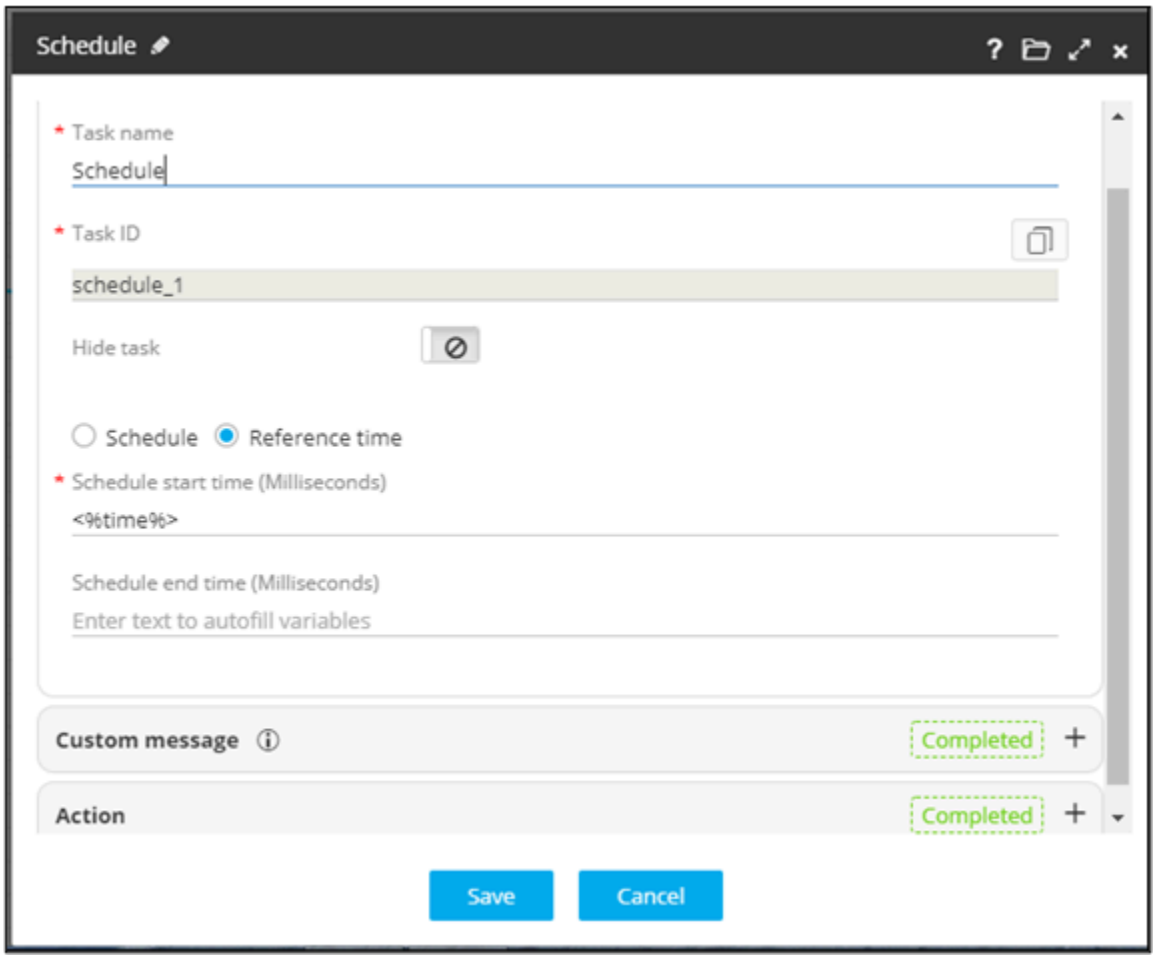


Note: For more information on adding form fields, click here.

5. Declare the date/time value as a global variable.

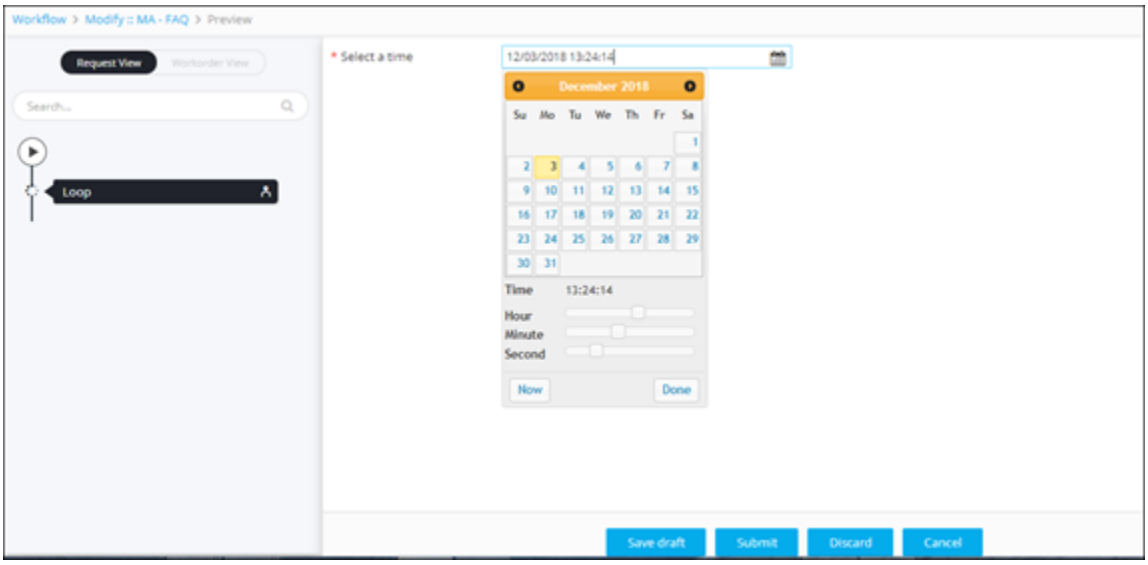
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

6. From the **General** section, drag and drop the **Schedule** task.
7. In the **Schedule** task window, refer the global variables defined in the form task into the Schedule task.

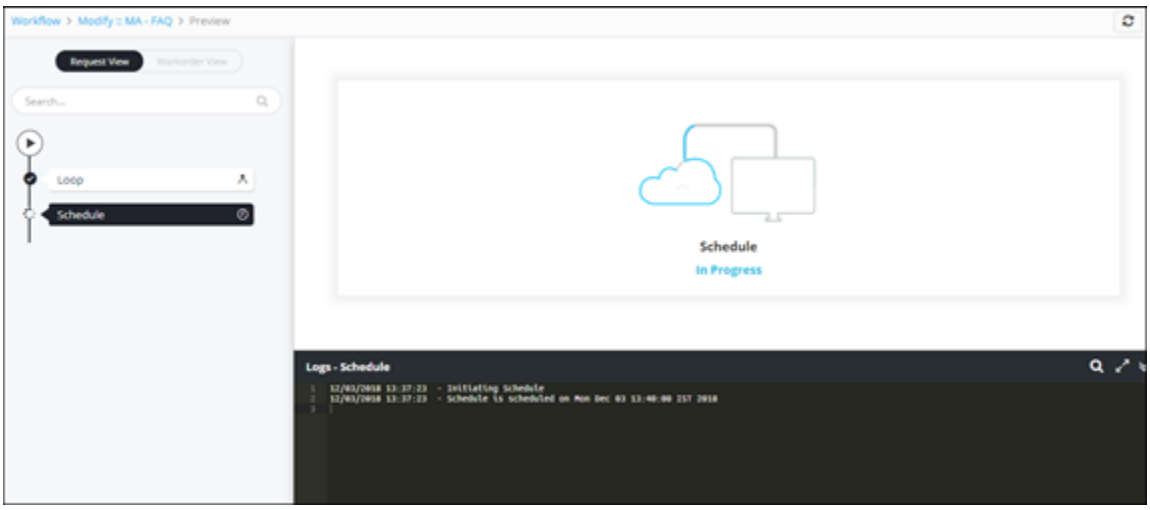


8. Drag and drop a **Script** task that is to be scheduled in the workflow.
9. Enable and trigger the workflow.
10. Select a time for scheduling the task.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



The task is scheduled.



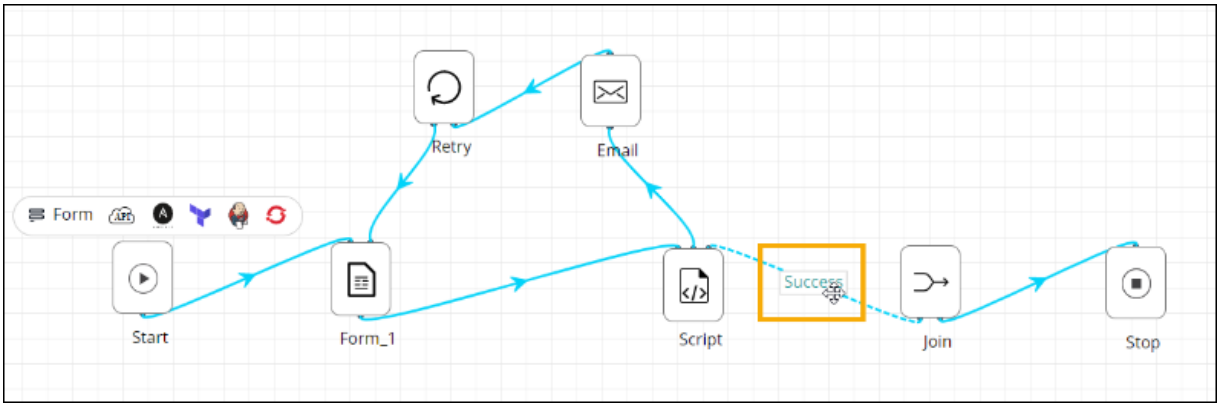
Connecting Workflow Tasks with Failover - RGF Flow

When connecting tasks in a workflow, links can be used to specify decisions (success or failure), based on which a workflow can be routed intelligently. A Script task can be used to make decisions based on the outcome of the script to perform specific workflow actions.

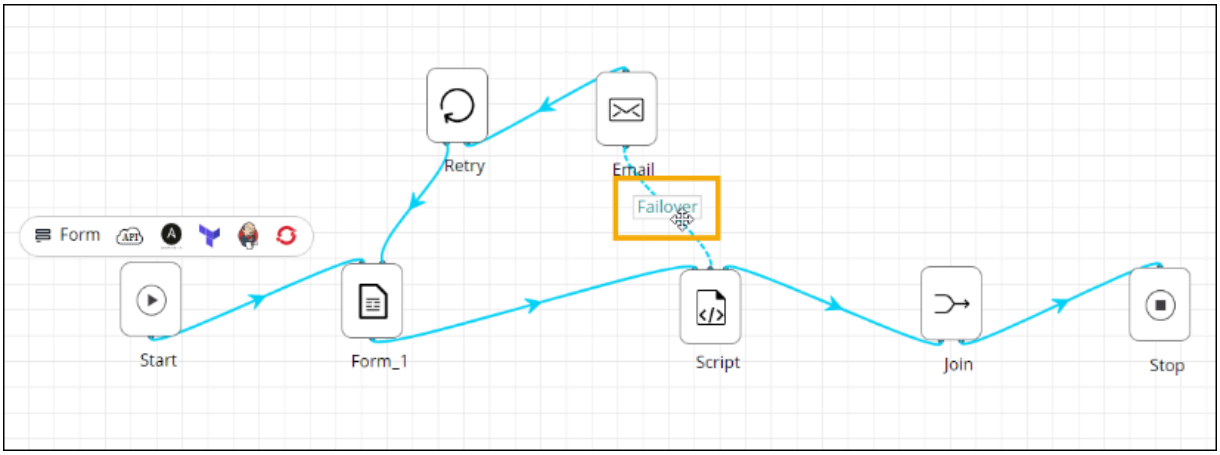
1. Design a new workflow.
2. In the Script task, define two outputs for ‘success’ and ‘failover’ outcomes.
3. Based on the script output, connect the links to the relevant workflow tasks.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

4. In case of success, proceed to the next task.



5. In case of failover, send an email notification and perform a retry.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Chapter 10: Visual Workflow Request

- Overview
- Accessing the Workflow Request Page
- Workflow Dashboard
- My Workflows
- My Requests
- Assigned Requests
- Audit
- Settings
- Archive Workflow Requests
- Restore Workflow Archives
- Configuring RBAC for Archive and Restore Requests

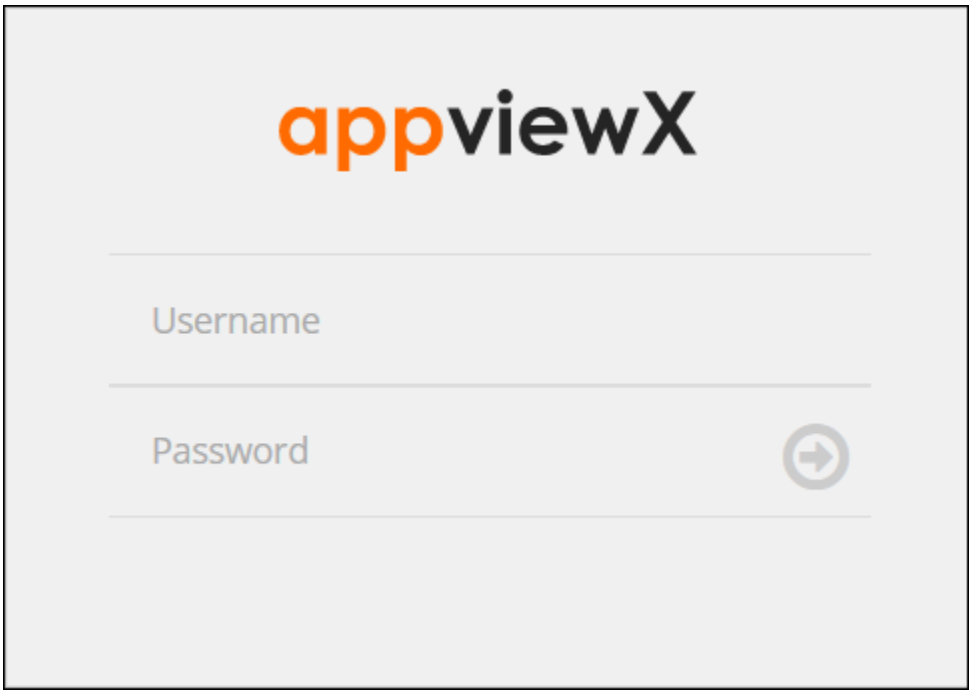
Overview

The Request console allows you to execute workflow(s). Workflows can be executed manually or can be scheduled.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

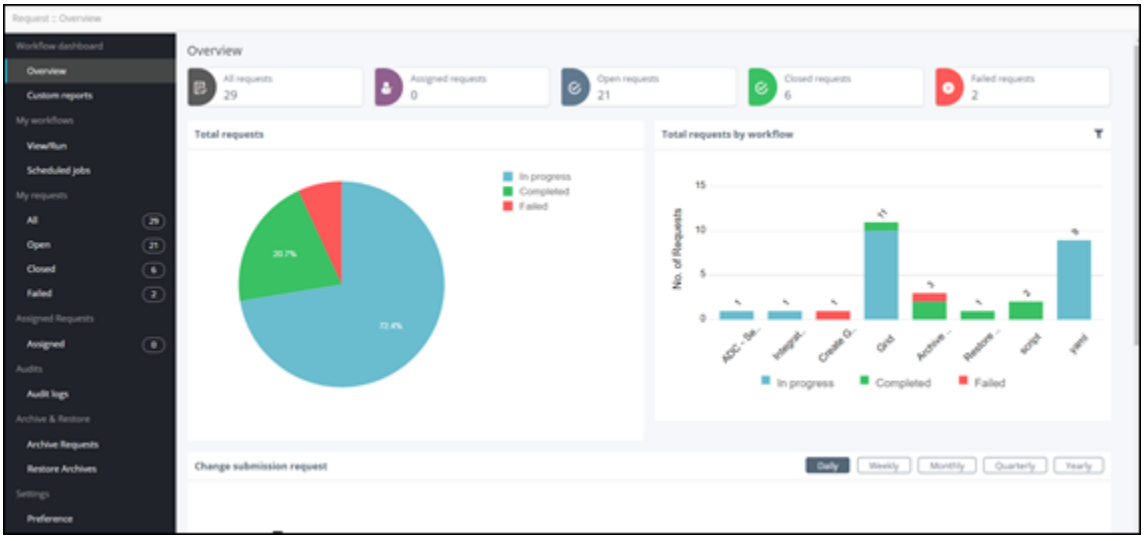
Accessing the Workflow Request Page

1. Log into AppViewX with valid credentials.



2. To access the navigation pane, hover the mouse over the ☰ Menu icon.
3. From the menu displayed, click **Request**.

   The Workflow **Request :: Overview** page is displayed.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Workflow Dashboard

The workflow dashboard gives you an overview of the status of all automation workflow requests, generate dynamic status reports, access workflow request logs, and manage catalog preferences.

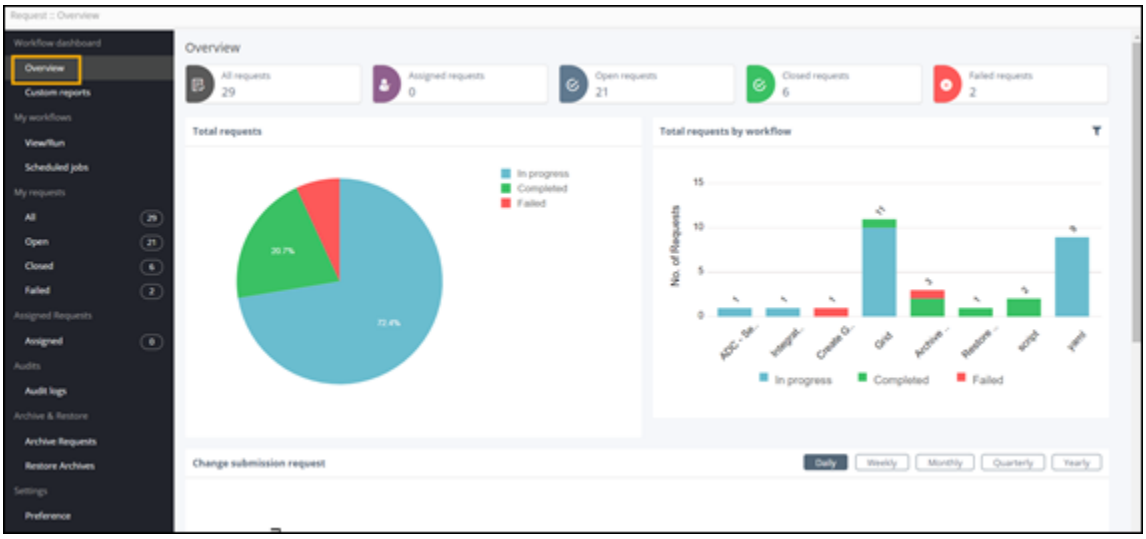
- Request :: Overview
- Custom Reports

Request :: Overview

The workflow dashboard provides a one-stop view of all workflow requests by status - opened, closed, failed and total workflow requests. It allows you to generate dynamic reports on the status of the workflow requests.

To access the **Request :: Overview** page:

1. From the top left corner of the screen, click ☰.
2. From the menu displayed, click **Request**.
   The **Request :: Overview** page is displayed.



Note: Click on any section of the chart or metric and navigate directly to the inventory. Following are the possible statuses for a workflow request:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

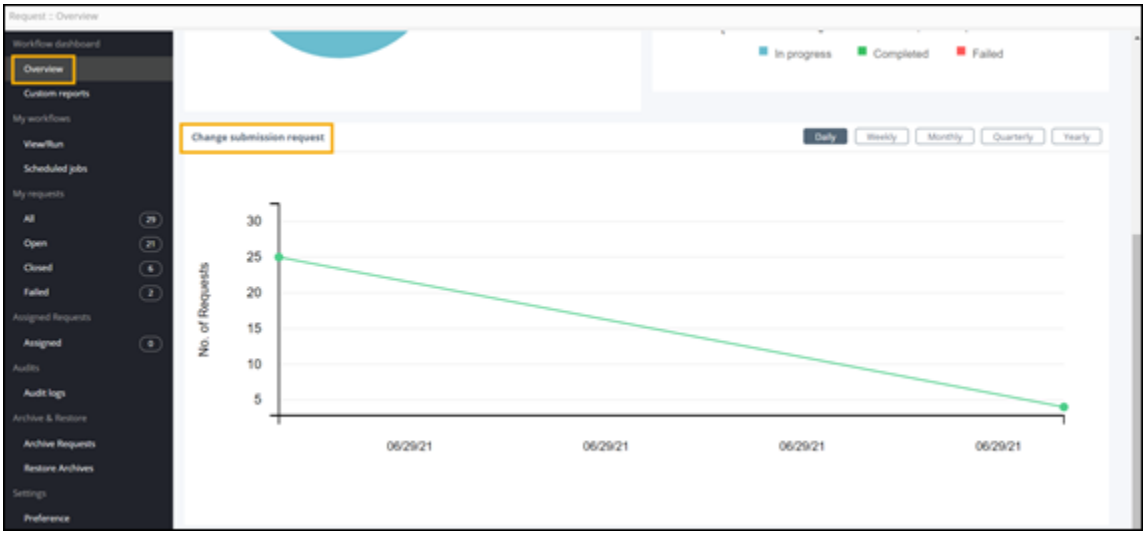
Status	Description
In-progress	The request is awaiting a response to proceed.
Completed	The request is complete and successful.
Failed	The request failed during execution.
Partial	The request is partially executed.
Rolled Back	The request is complete and has rolled back to its previous state.
Paused	The request is suspended/paused.
Aborted	The request is aborted.

- Change Submission Trend

Change Submission Trend

This allows you to visualize the request trend over a period of time.

- Provision to perform trend analysis on the requests triggered over a period of time
- Provision to create daily, weekly, monthly and yearly views for the requests triggered

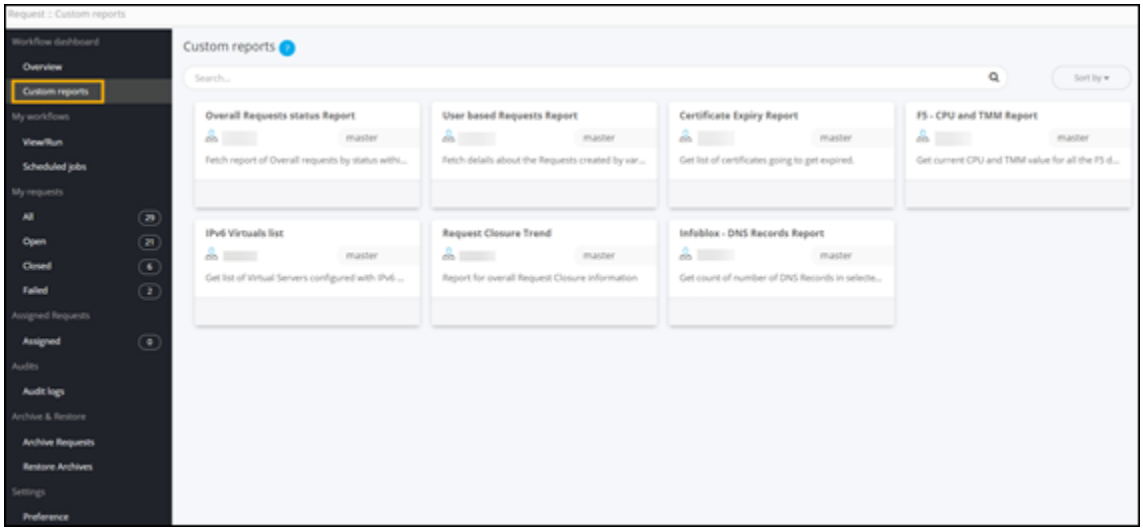


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Custom Reports

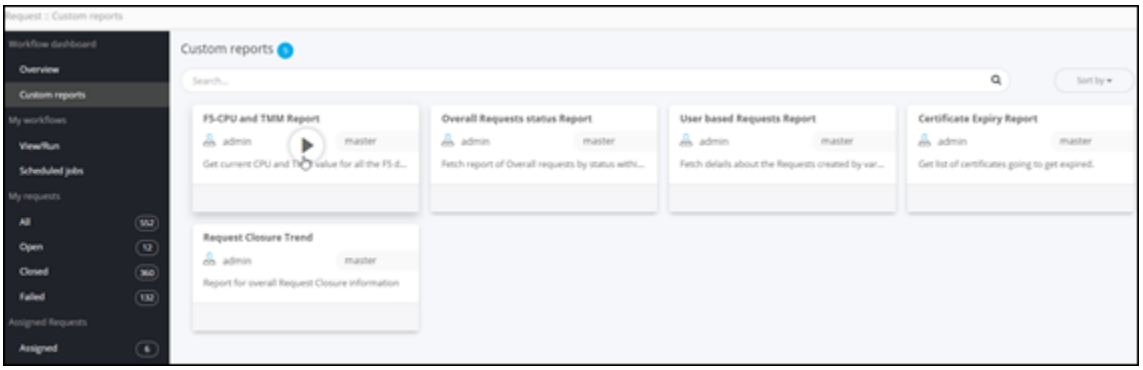
Allows the user to view and trigger custom reports.

- Any workflow created under the category Reports will be available under Custom report section.
- Provision to search for a custom report.
- Provision to apply custom sort based on ascending, descending, recently created.
- Provision to execute/run a custom report workflow.



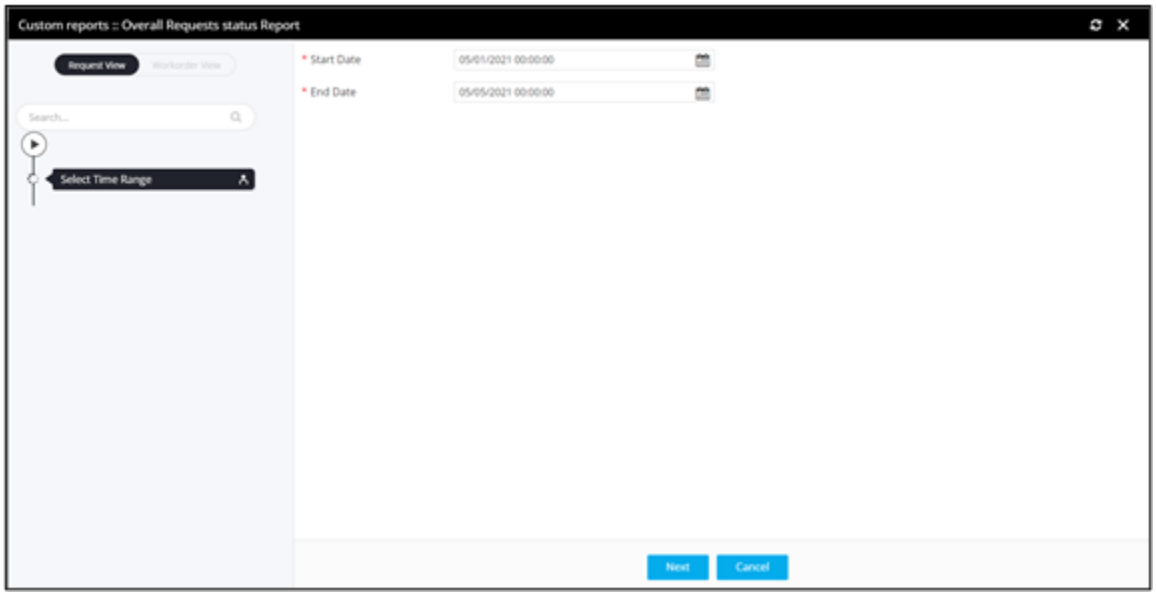
To execute a custom report workflow:

1. On the Request :: Overview page, from the navigation pane on the left, click **Custom Reports**.
   The Request :: Custom Reports page is displayed.
2. To execute any workflow under **Custom Reports**, hover your mouse over the workflow and click .



3. Enter the **Start Date** and **End Date**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



4. Click **Ok** in the **Confirmation** pop-up window.

   Custom report is ready.

My Workflows

In this section you can trigger the workflows that are enabled in the Workflow Studio and also schedule them to trigger at a later point in time.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- Request :: View/Run
- Scheduled Jobs

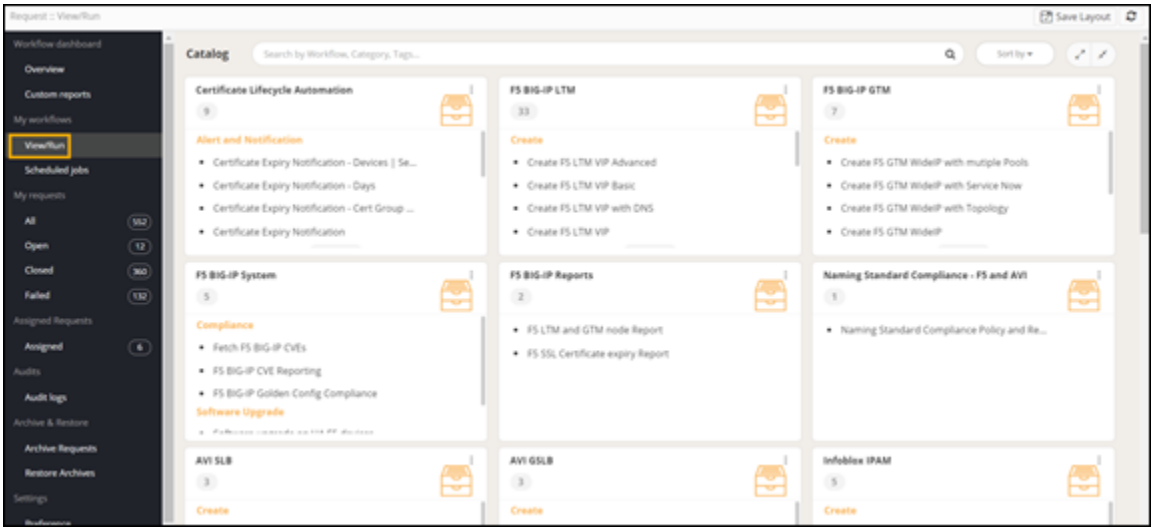
Request :: View/Run

The **Request :: View/Run** page displays the workflow catalogs.

- Provision to search for a workflow by keyword.
- Provision to trigger a workflow on demand.
- Provision to schedule a workflow.

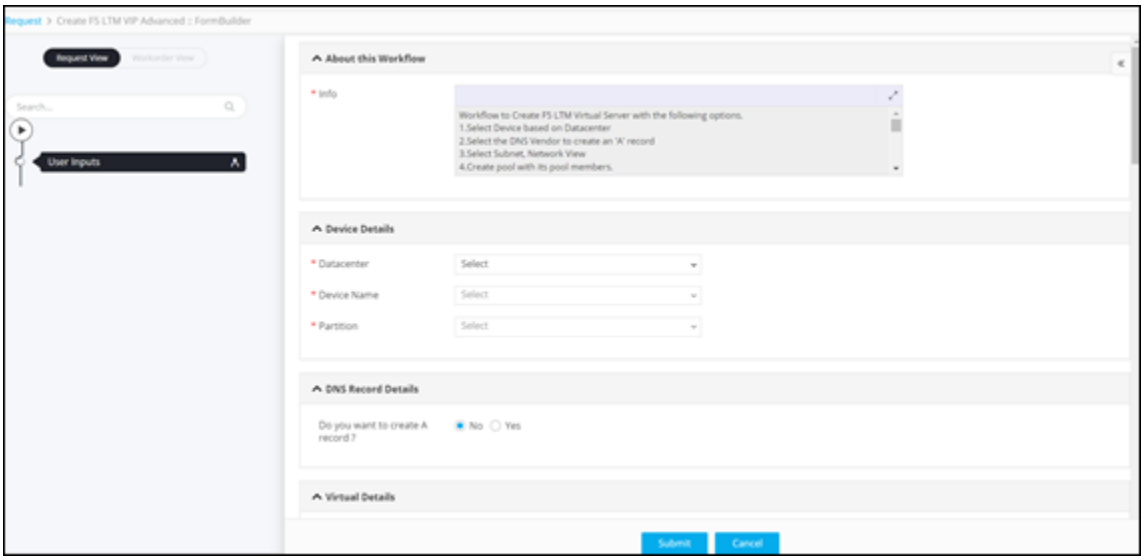
To trigger a workflow from the Workflow **Request :: View/Run** page:

1. From the top left corner of the screen, click ☰.
2. From the menu displayed, click **Request**.
3. On the **Request :: Overview** page, from the navigation page pane on the left, click **View/Run**.
   The **Request :: View/Run** page displays all the workflows that have been enabled in the Workflow Studio.



4. On the **Request :: View/Run** page, to trigger a workflow search for the workflow name and click ▶.
   The workflow is executed with the user inputs being requested at the first stage.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



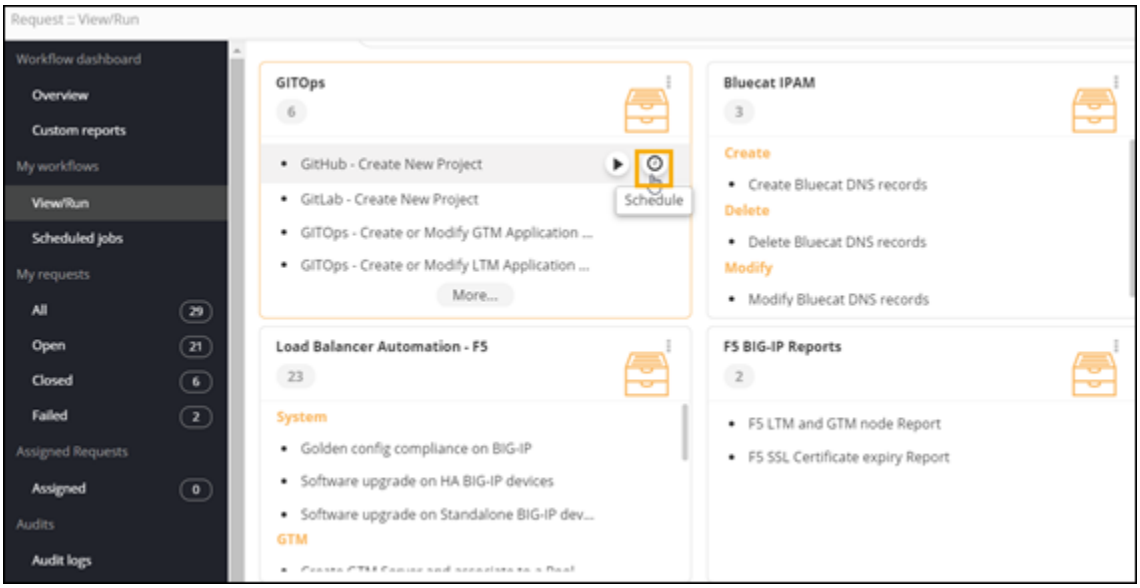
- Scheduling a Workflow
- Editing the Workflow Catalog Layout
- Editing Subcategories and Workflows within a Catalog

Scheduling a Workflow

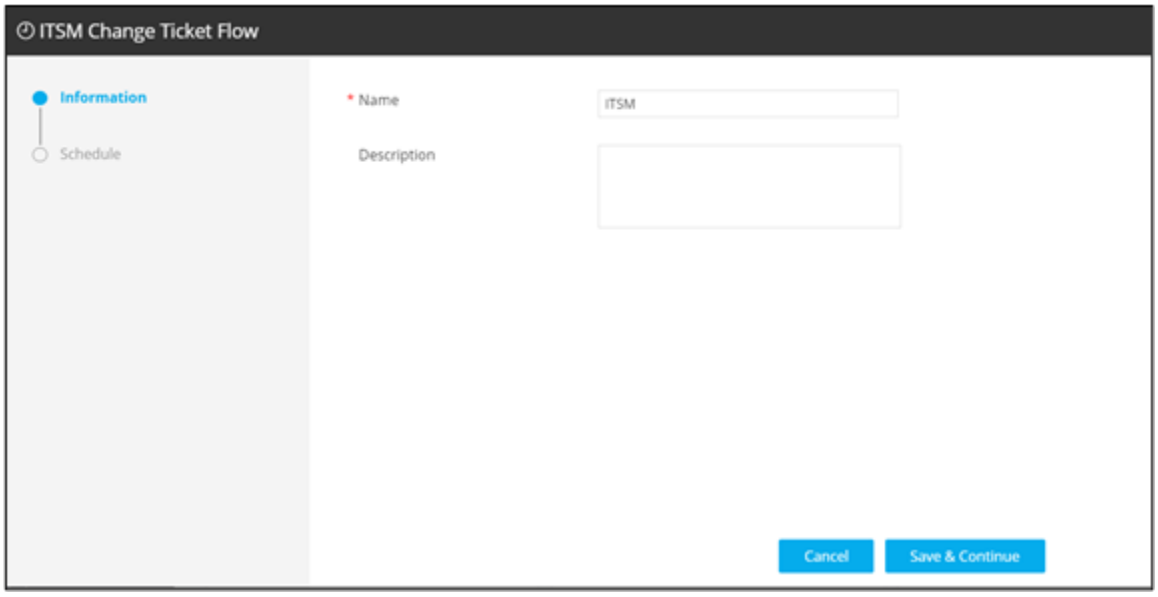
This task allows you to schedule a workflow to trigger once or daily, weekly, monthly and yearly as per requirement.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. To schedule a workflow, on the Request :: View/Run page, hover your mouse over a workflow and click .



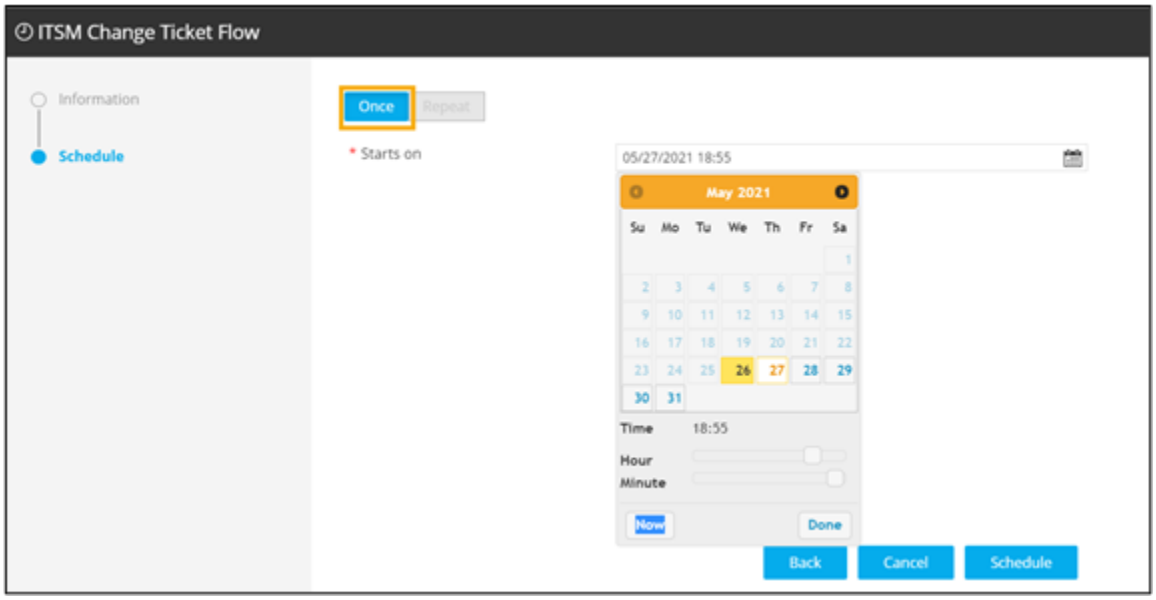
2. Enter the field information.



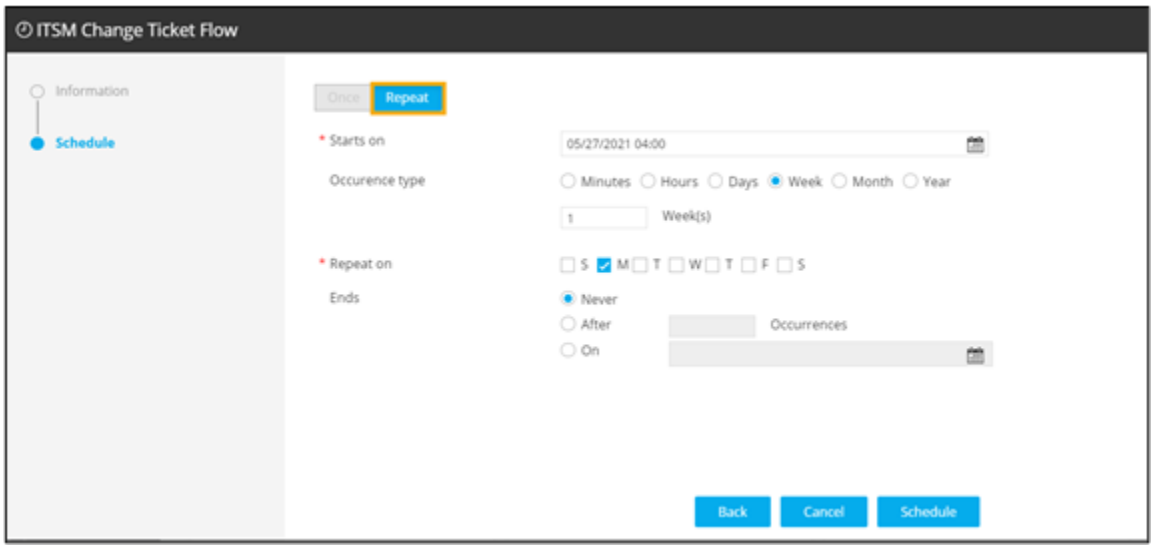
3. Assign the time for which the workflow is to be scheduled.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

- **Once**



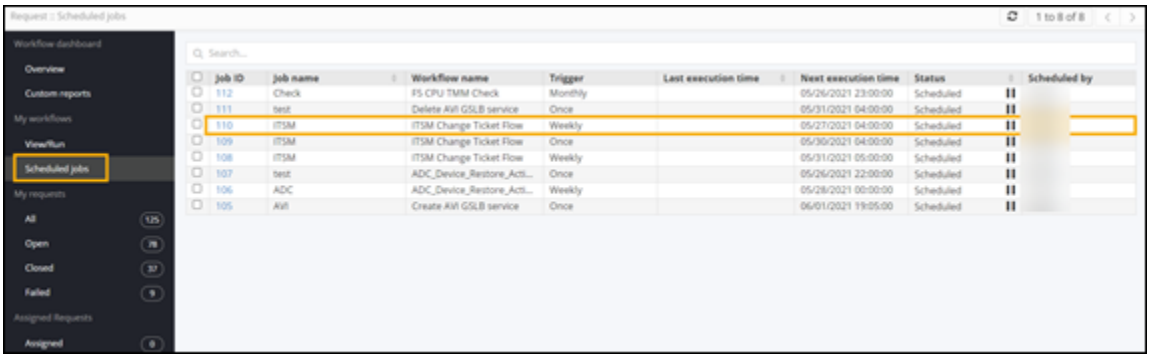
- **Repeat** - Provision to schedule the workflow to trigger daily, weekly, monthly and yearly.



4. Click **Schedule**.

   The scheduled workflow is added to the **Scheduled jobs** section.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Editing the Workflow Catalog Layout

You can move the catalogs around and place them according to your preference on the View/Run page. You can also expand or collapse all workflow catalogs. Expanding the catalogs displays the list of workflows with a catalog.

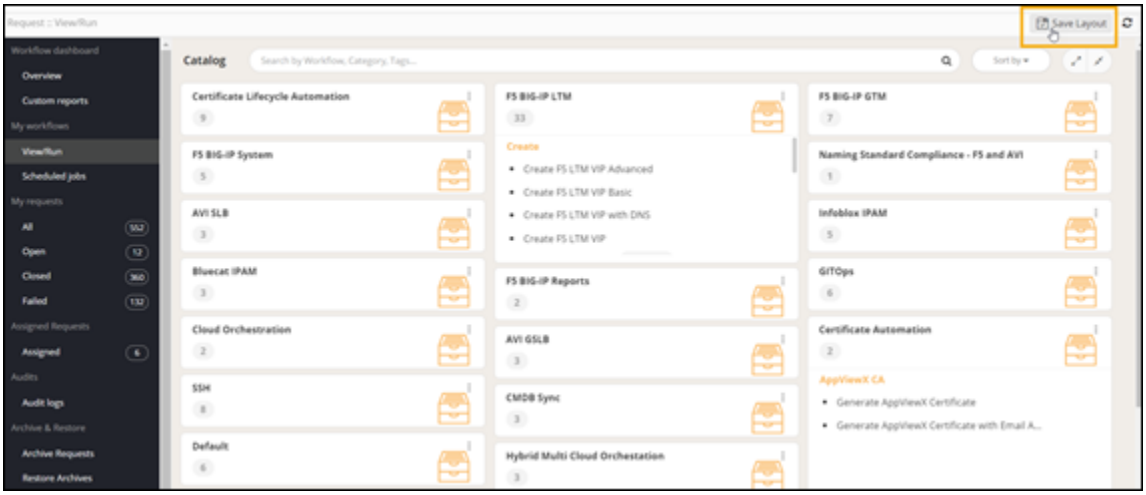
To edit the layout:

1. On the Request :: View/Run page, from the top right corner of the screen, click **Edit Layout**.

2. Hold and drag the workflow catalog to another area on the page.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

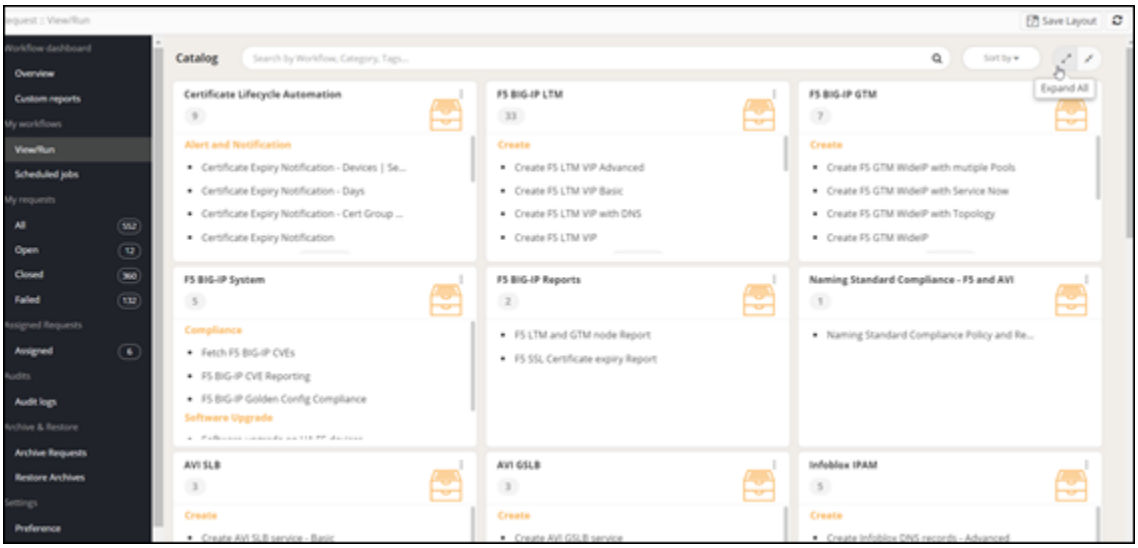
3. To save your layout preferences, from the top right corner of the page, click **Save Layout**.



4. To expand all the workflow catalogs, from the top right corner of the screen, click .

   Expanded view of the catalogs.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



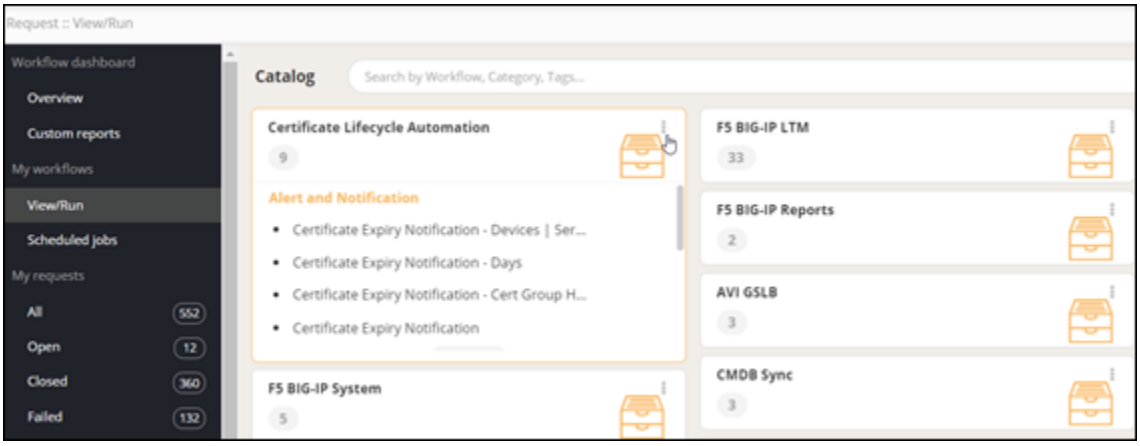
5. To collapse the workflow catalogs, from the top right corner of the screen, click . Collapsed view of workflow catalogs.

Editing Subcategories and Workflows within a Catalog

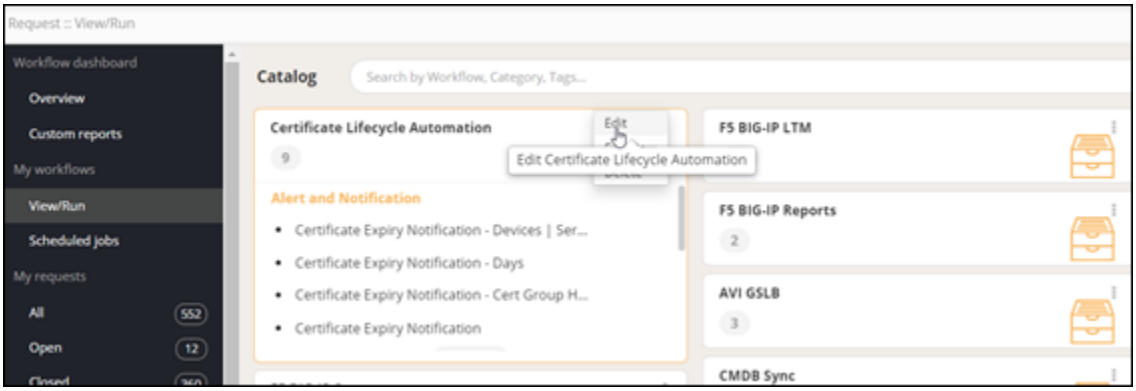
You can select the workflows to be placed under a workflow catalog and modify the order of sub-categories within the workflow catalog.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. On the **Request :: View/Run** page, from the top right corner of the screen, click .

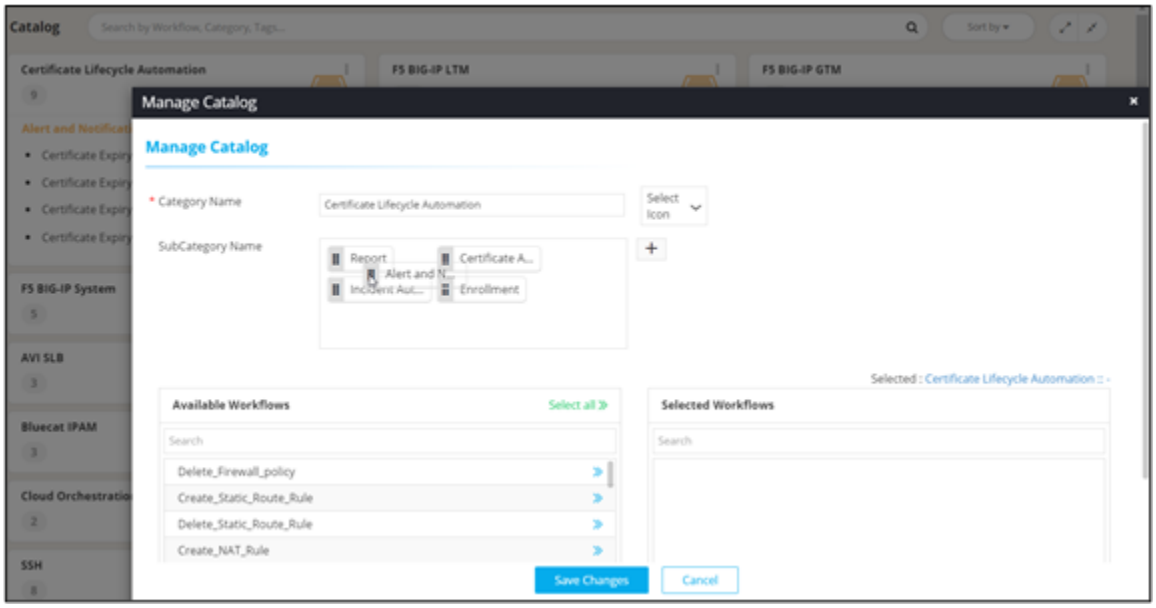


2. From the options displayed, select **Edit**.



3. In the **Manage Catalog** window, under **SubCategory Name**, hold and drag folders to modify the order in which they are displayed on the workflow card.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



4. To add more workflows under a subcategory, select the subcategory and click » next to the workflow name under **Available Workflows**.
5. To move a workflow out of the selected workflows, click « next to the workflow under **Selected Workflows**.
6. Click **Save Changes**.

Scheduled Jobs

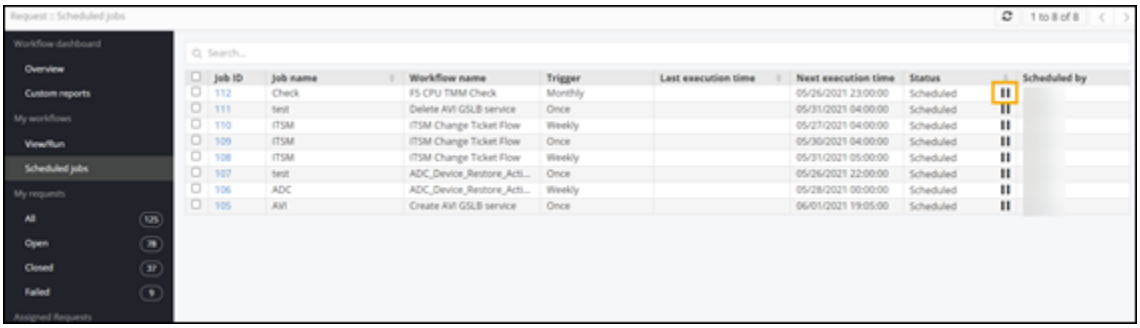
This section shows the inventory of scheduled workflows.

- Provides an option to ‘pause’ and ‘play’ a workflow.
- Provides details of execution time, trigger time and logs.

To schedule a job:

1. On the Request :: Overview page, from the navigation pane in the left, click Scheduled Jobs. The **Request :: Scheduled Jobs** page is displayed.
2. On the **Request :: Scheduled Jobs** page, to pause a scheduled job click ⏸.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



The scheduled job is paused.

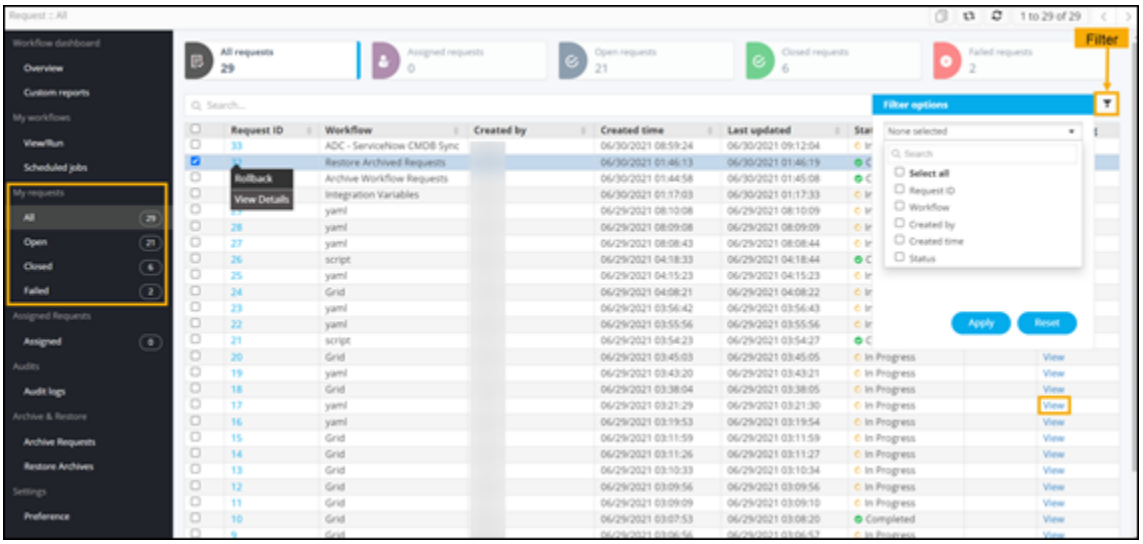
3. To resume a paused scheduled job, click ▶ .
   The scheduled job resumes.

My Requests

This section provides a single point view of all the assigned workflows, workflow requests triggered and workflow(s) that are scheduled. Requests are logically categorized based on All, Open, Closed and Failed requests.

- Provision to search for a request workflow by keyword.
- Provision to filter and search by categories
- Provision to view requests by logged in user, user role(s)
- Provision to view the workflow activity log summary and detail
- Provision to pause or abort a ‘In Progress’ workflow
- Provision to ‘view details of the workflow as a tooltip information
- Provision to manually rollback a workflow request

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



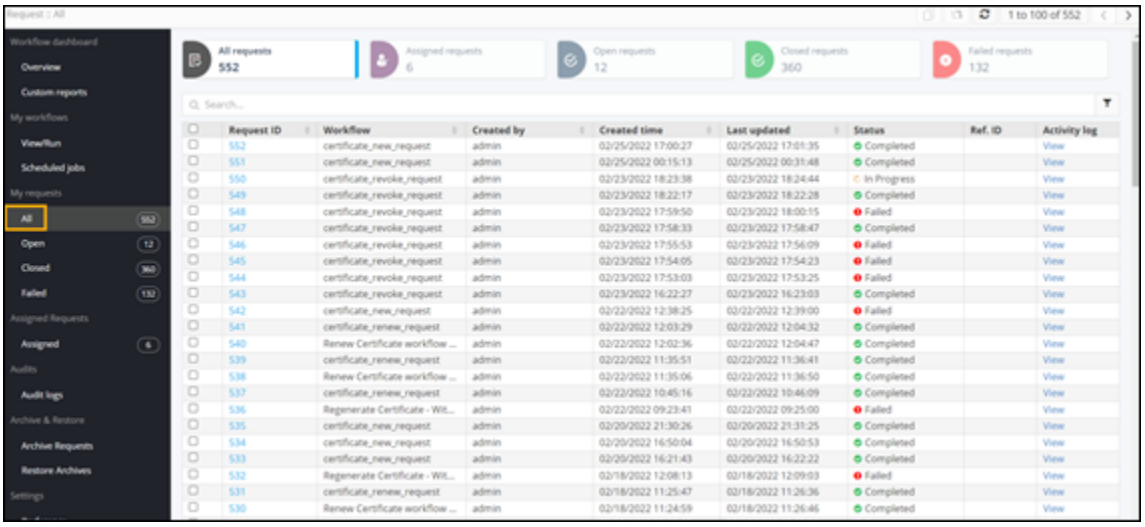
- Request :: All
- Viewing the Workflow Log Summary
- Viewing workflow details
- Pausing/Resuming a Workflow
- Aborting a Workflow
- Rollback a Workflow

Request :: All

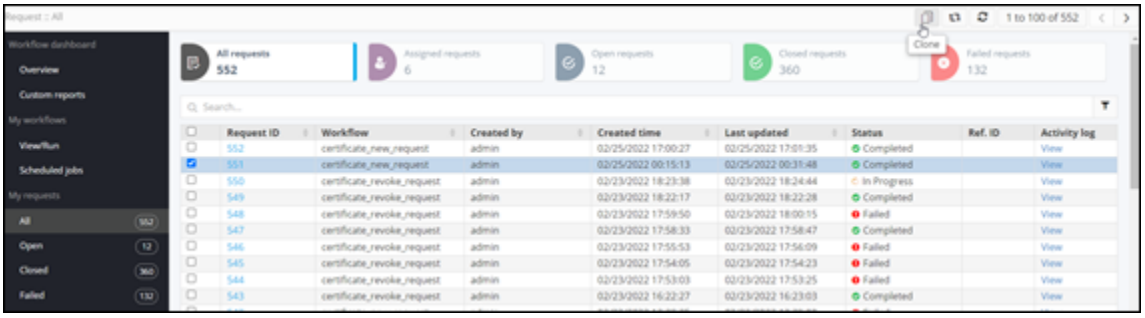
You can see a list of all (open, closed, failed) requests on the **Request :: All** page.

1. On the **Request :: Overview** page, from the navigation pane on the left, click **All**.
   The **Request :: All** page is displayed.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



2. To see the stage-wise execution of a workflow, click on the **Request ID** for that workflow.
3. To clone a workflow request, select the workflow and click from the command bar in the top right corner of the screen.



.

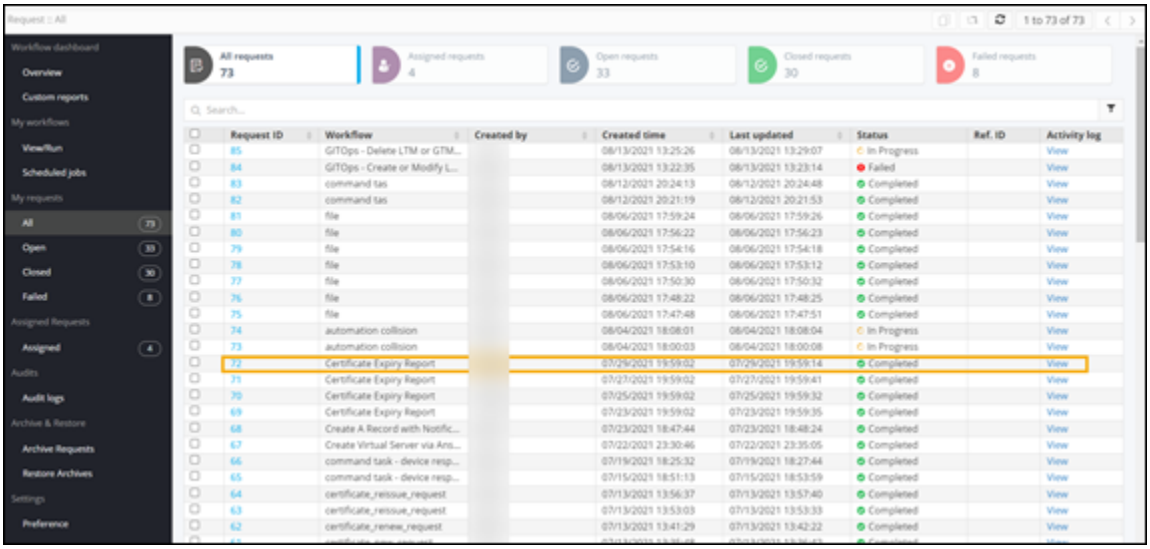
4. To refresh the **Request :: All** page, from the command bar, click .

Viewing the Workflow Log Summary

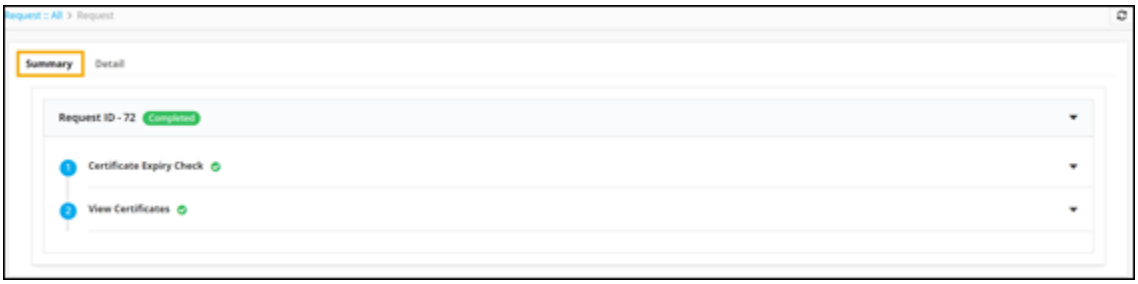
To view the activity log of a specific workflow:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. On the Request :: All page, click **View** next to the selected workflow.

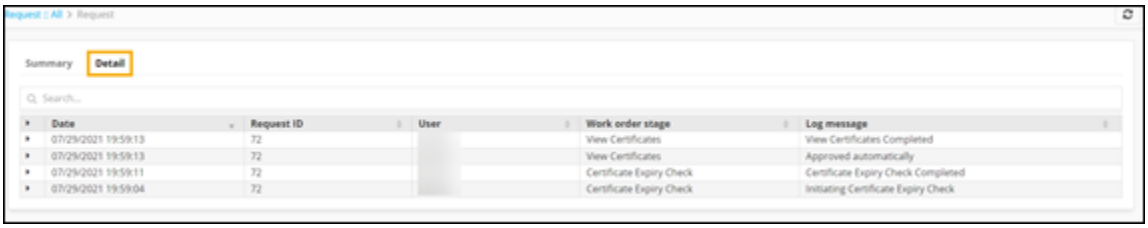


A summary of the workflow stages is displayed.



2. To view the workflow log requests details, click **Detail**.

A detailed log of the workflow stages is displayed.

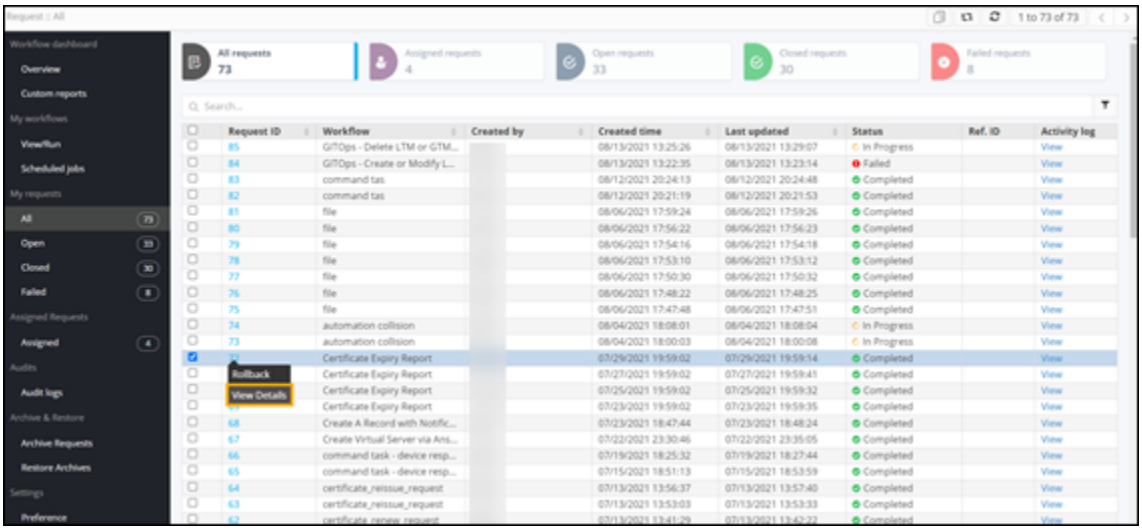


Viewing workflow details

You can view any pertinent information related to the specific workflow.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. On the Request :: All page, right-click a workflow **Request ID** to see options.
2. From the options displayed, select **View Details**.

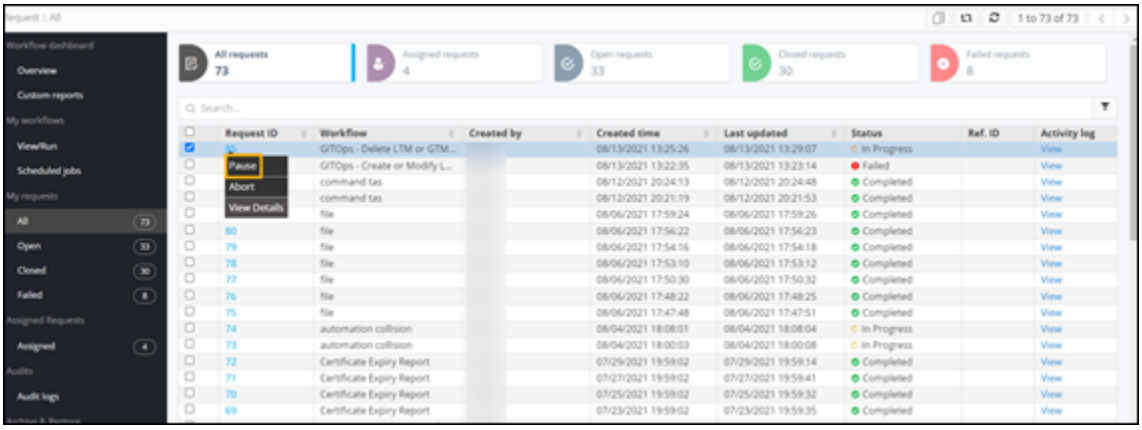


Note: For more information, refer to the section on Tooltip data configuration.

Pausing/Resuming a Workflow

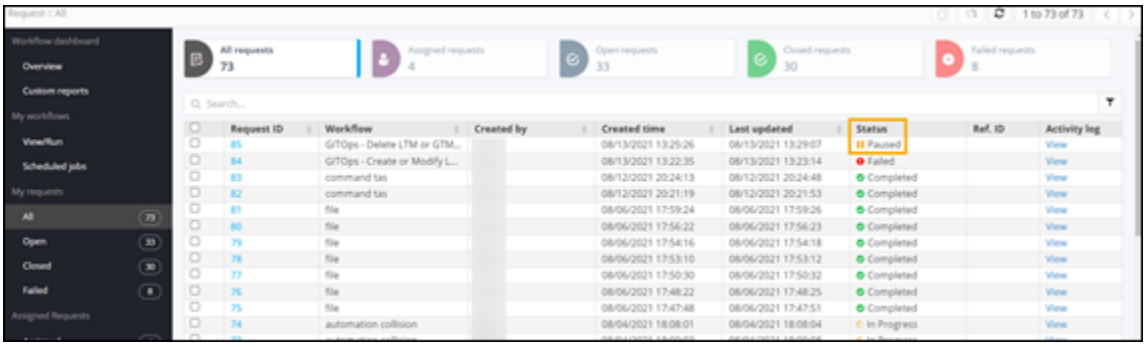
You can pause and resume a workflow request which is in the **In Progress** status.

1. On the Request :: All page, right-click the **Request ID** of a workflow to see the options.
2. To pause the workflow, select **Pause** from the options.

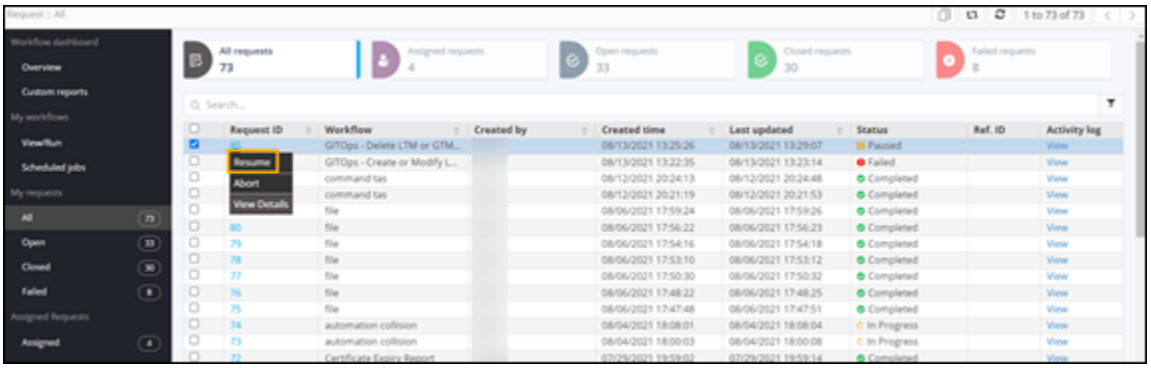


3. Click **Yes** in the pop-up window. The workflow is paused.

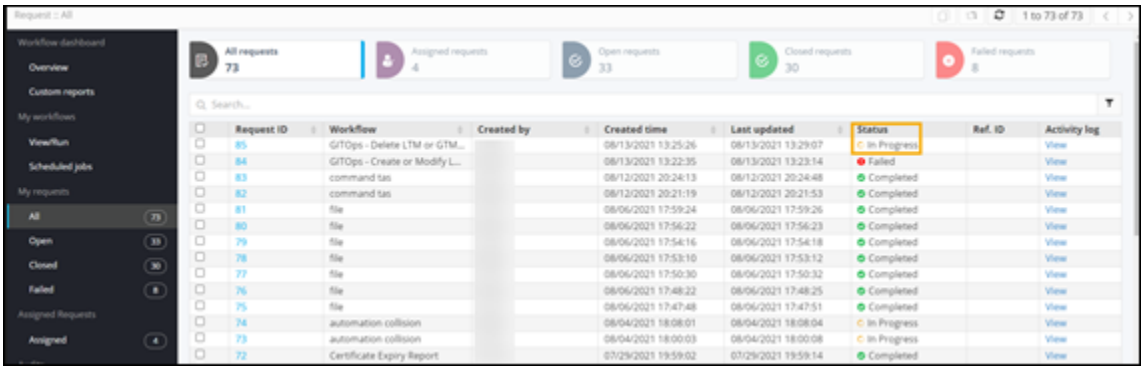
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



4. To resume the workflow, select **Resume**.



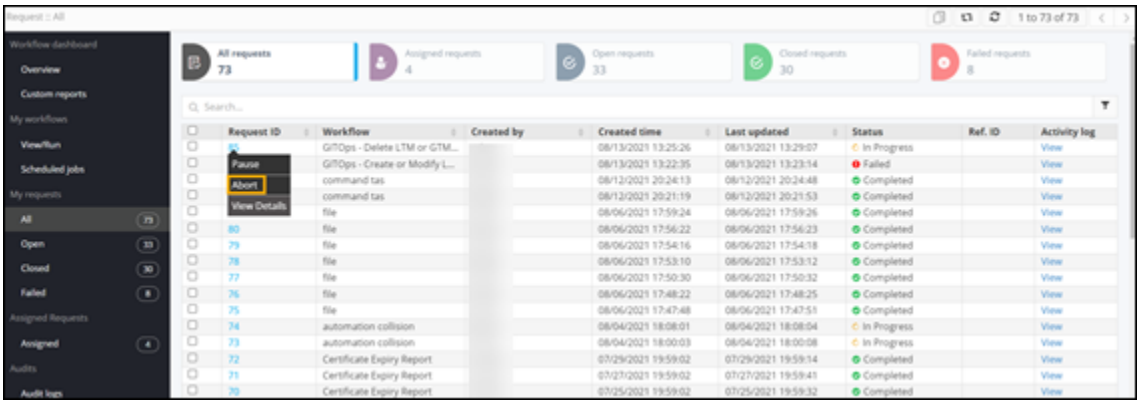
5. Click **Yes** in the confirmation pop-up window. The workflow resumes.



Aborting a Workflow

1. On the Request :: All page, right-click the **Request ID** of the workflow to be aborted.
2. Select **Abort** from the available options.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



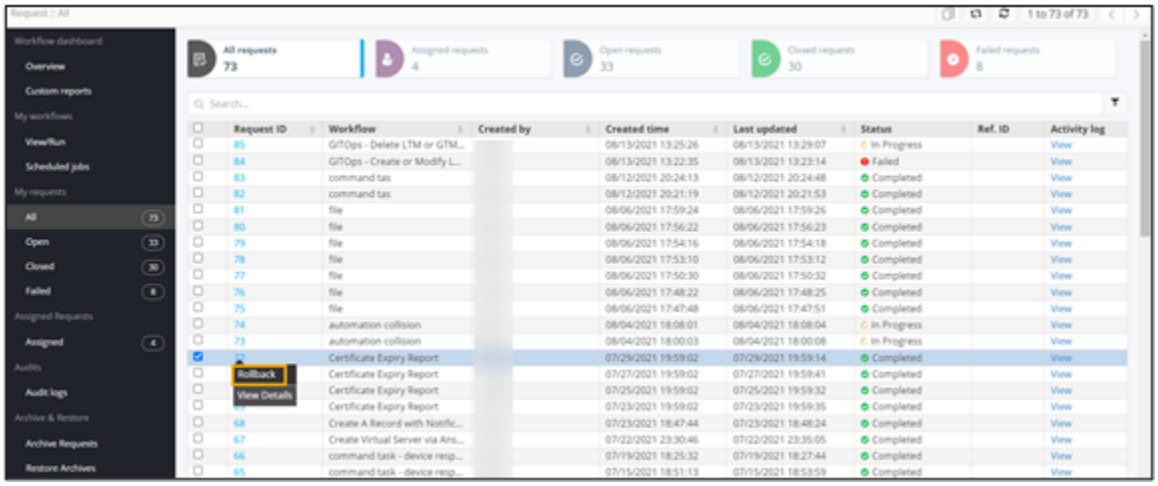
3. Click **Yes** in the confirmation pop-up window.

> **Note:** You can abort a workflow only if it is in the **In progress** state.

Rollback a Workflow

You can manually rollback a workflow request that has been completed.

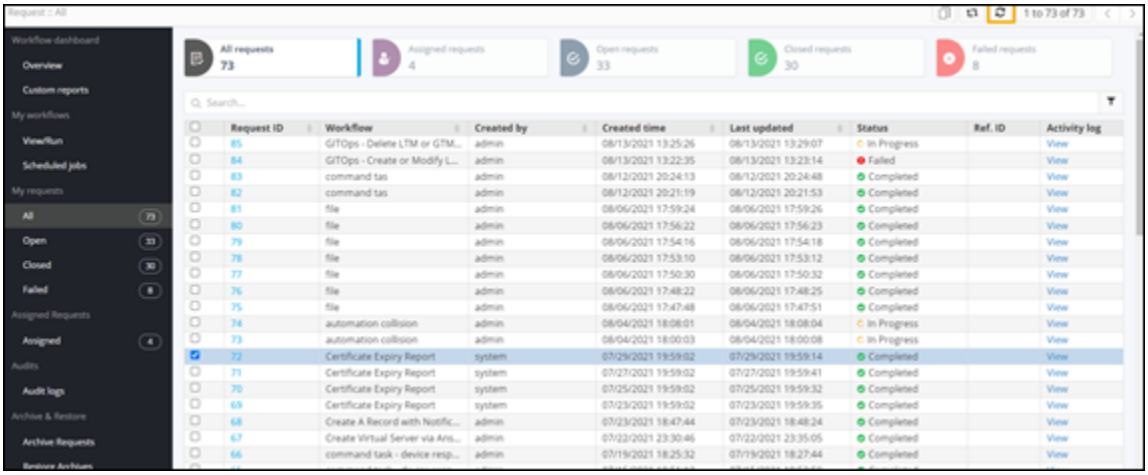
1. On the Request :: All page, right-click the workflow **Request ID** to see options.
2. From the options, select **Rollback**.



3. Click **Yes** in the Confirmation pop-up window.

> **Tip:** You can also rollback a workflow, by selecting the workflow and clicking in the upper right corner of the screen.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



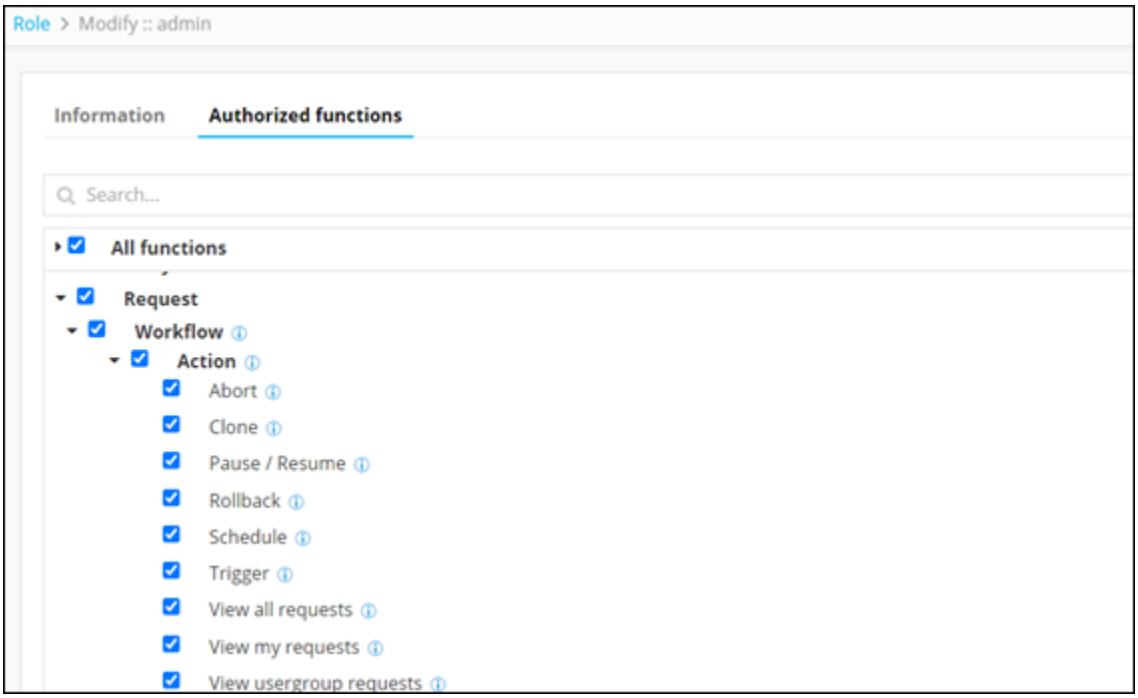
Assigned Requests

The following settings under RBAC allow for viewing and accessing of workflow requests in the Request Inventory based on:

- View my requests: To view workflow requests by logged in user
- View user group request: To view workflow requests based on the user group
- View all requests: To view all workflow requests

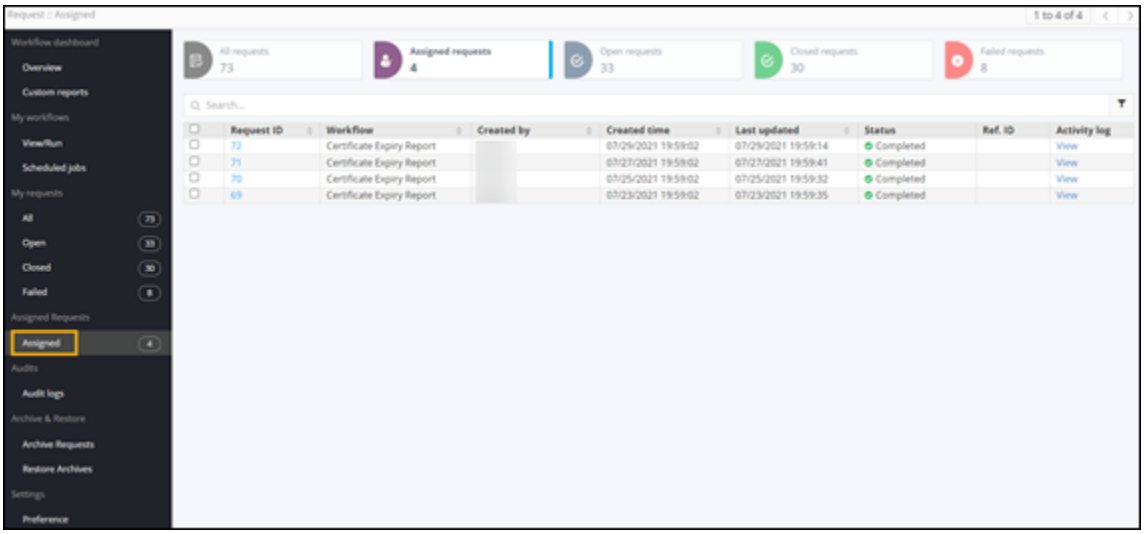
1. From the top left corner of the screen, click ☰.
2. Select **Account** > **Role**.
3. Select admin from the list of roles.
4. To view the actions authorized for the admin, click **Authorized functions**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



5. On the Request :: Overview page, from the navigation pane on the left, click From the top left corner of the screen, click **Assigned requests** ☰.

   The **Request :: Assigned** page is displayed showing the requests assigned to the particular role.



- Configuring RBAC for Assigned Requests

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Configuring RBAC for Assigned Requests

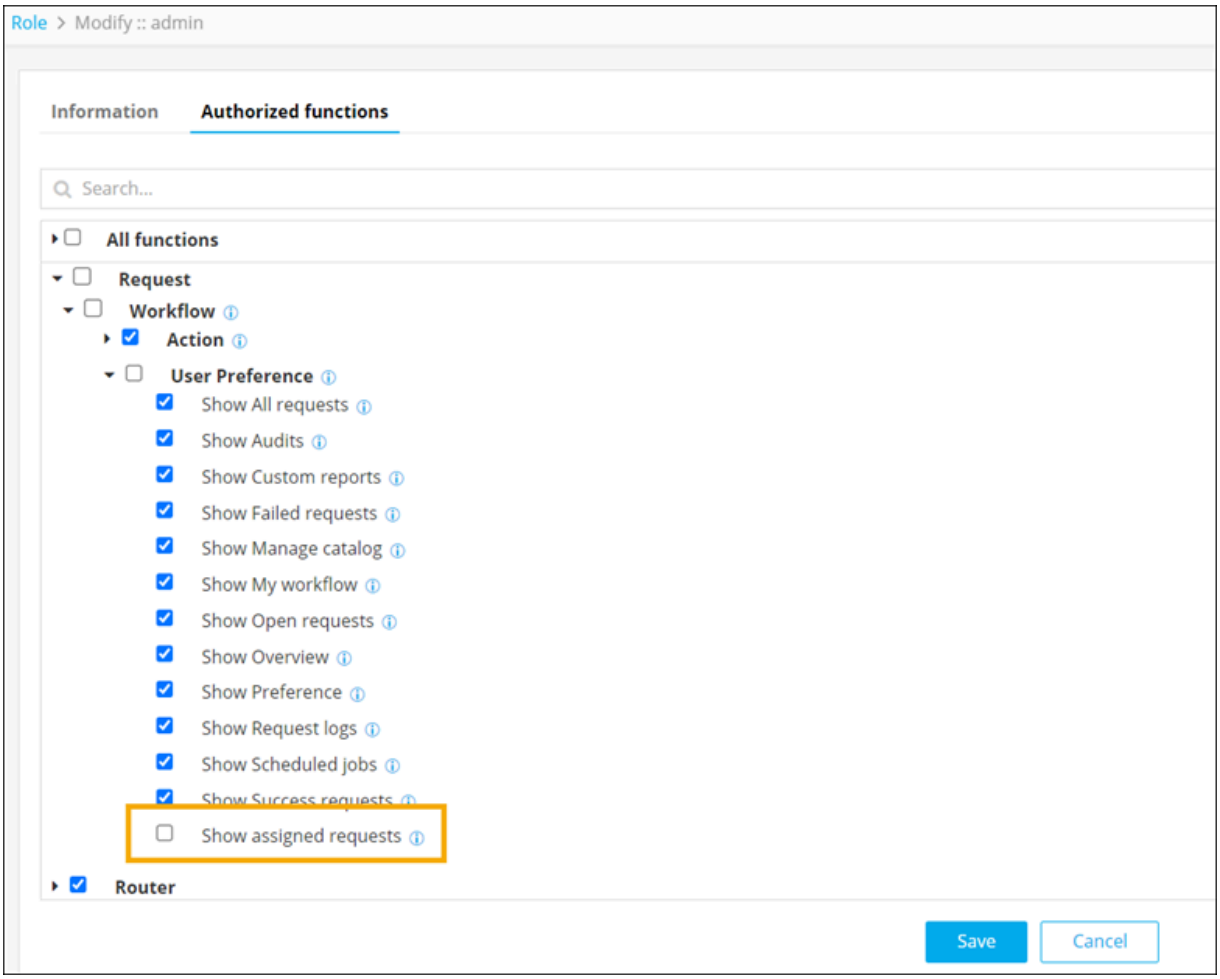
The following settings under RBAC allows for viewing the assigned workflow requests in the Request Inventory based on:

- View my requests: To view workflow requests by logged in user
- View user group request: To view workflow requests based on the user group
- View all requests: To view all workflow requests

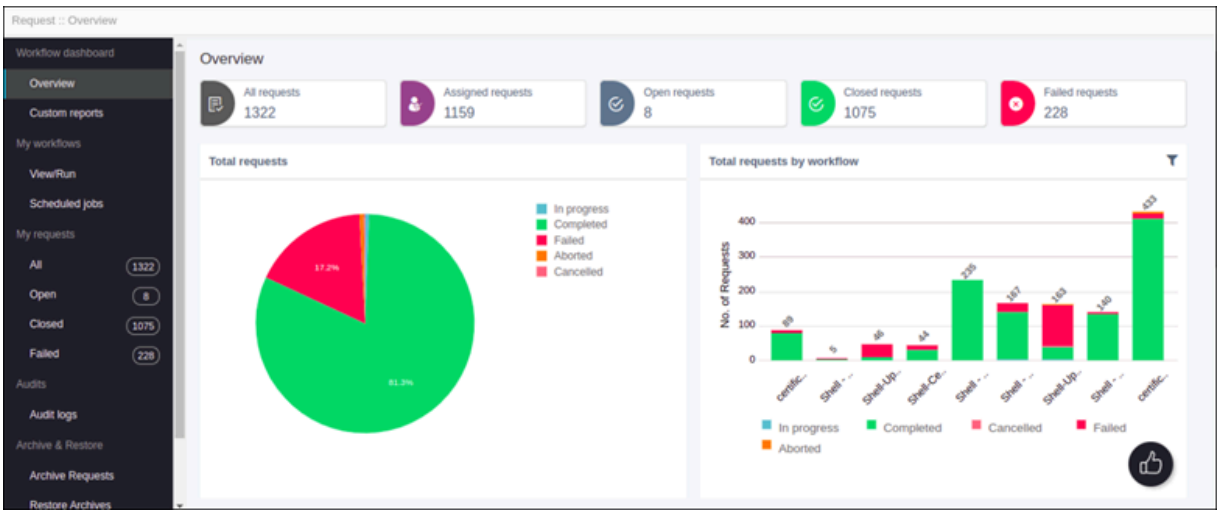
To configure the RBAC for assigned requests:

1. From the upper left corner of the screen, click ☰.
2. Select **Account** > **Role**.
3. Select **admin** from the list of roles.
4. To view the actions authorized for the **admin**, click **Authorized functions**.
5. To restrict a user from viewing the assigned requests, under **Request** > **Workflow** > **User Preference**, unselect the **Show assigned requests** checkbox.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

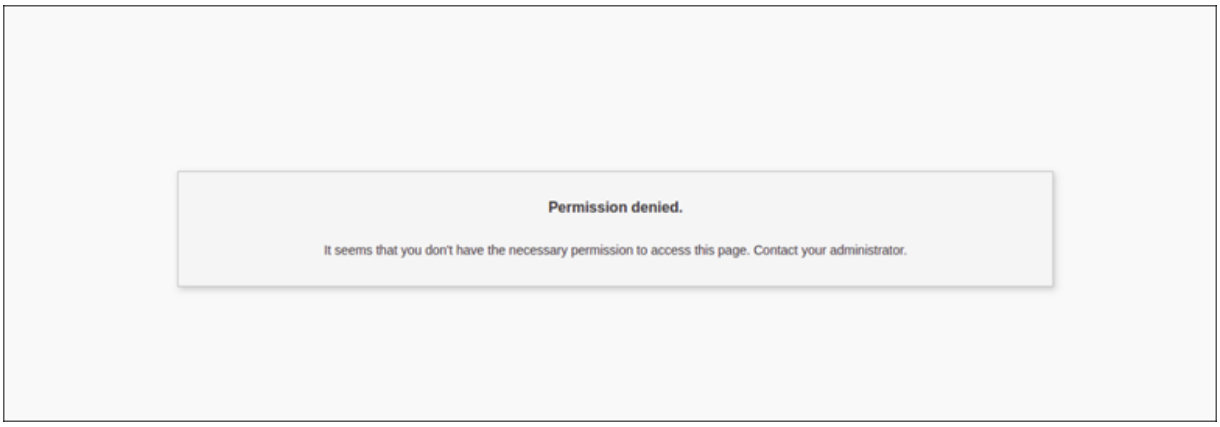


The **Assigned Requests** menu is not displayed on the **Request :: Overview** page.

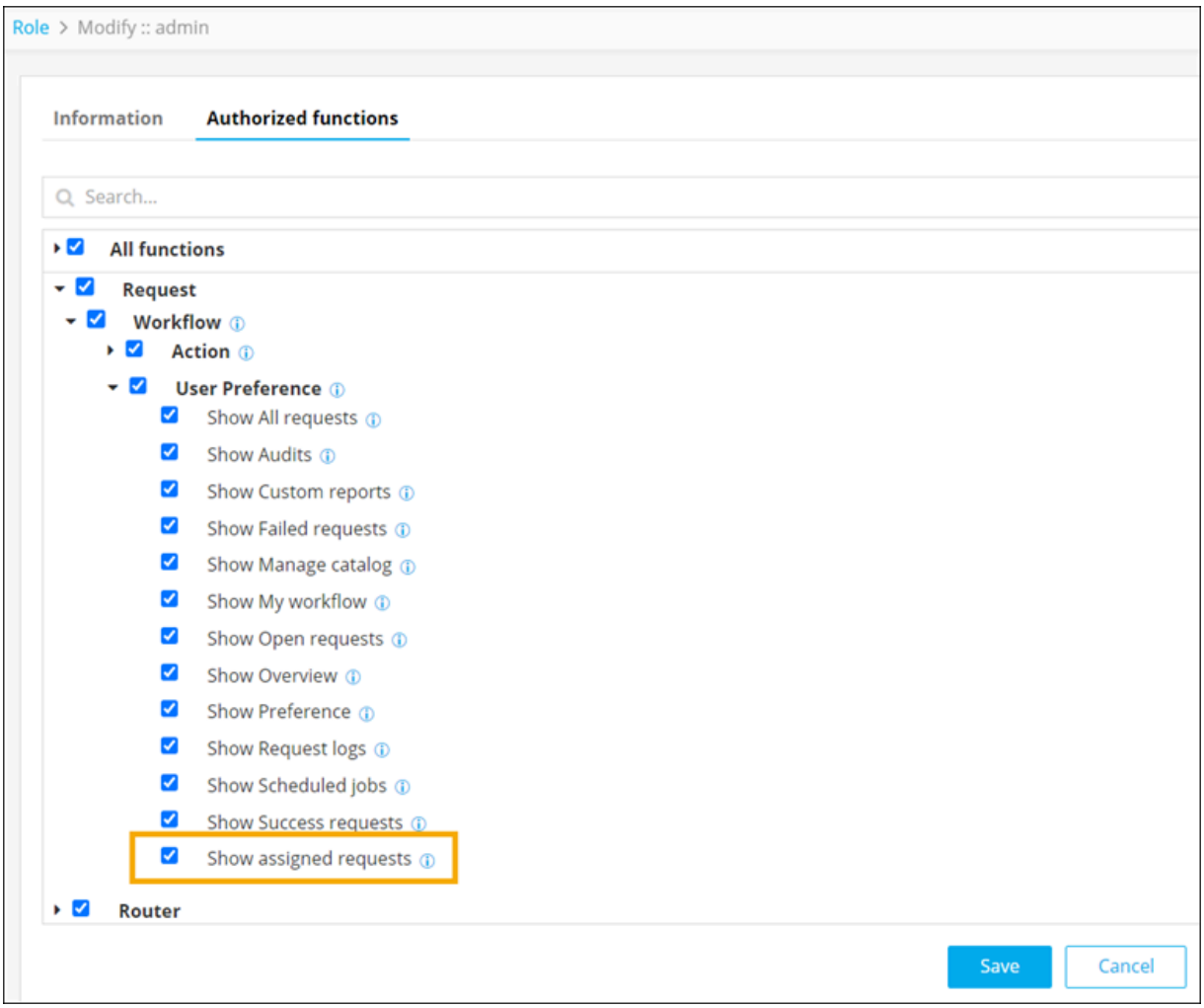


The following error message is displayed if the user tries to access the Assigned Requests.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

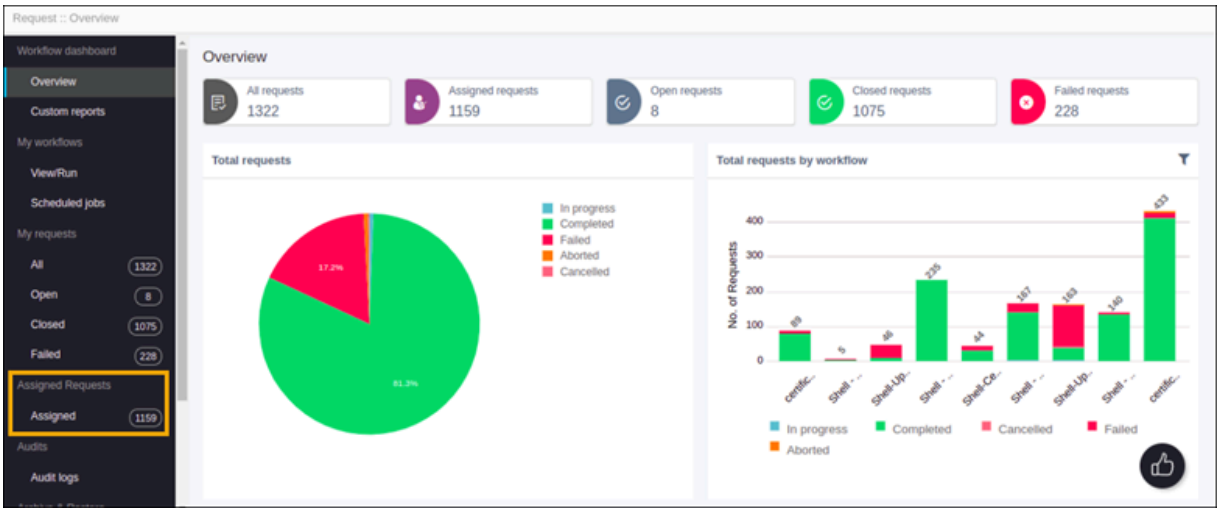


6. To allow a user to view the assigned requests, under **Request** > **Workflow** > **User Preference**, select the **Show assigned requests** checkbox.



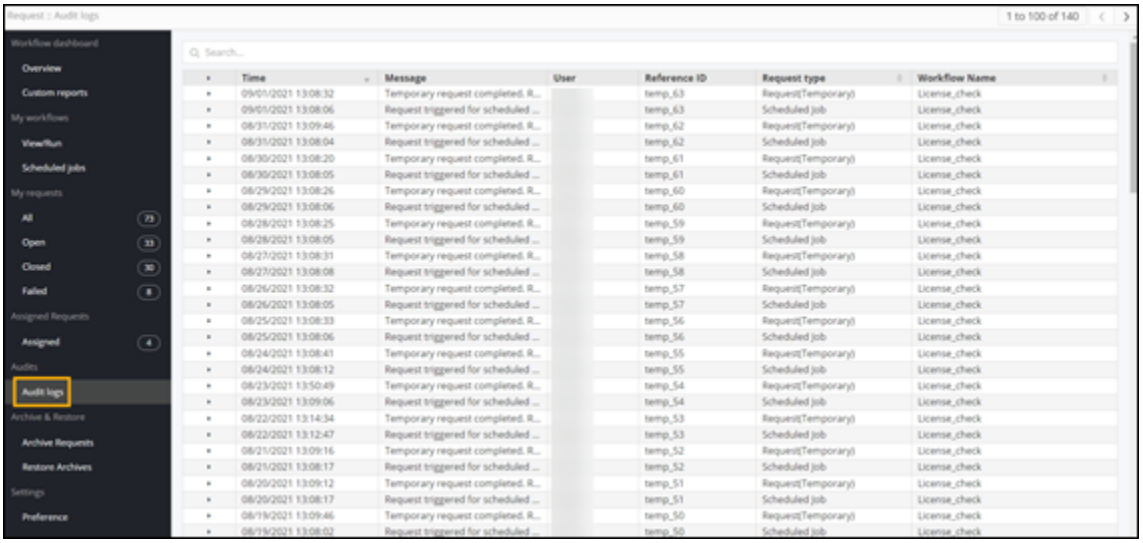
The **Assigned Requests** menu is displayed on the **Request :: Overview** page.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



Audit

You can view the audit logs of all scheduled workflows and report-based workflows that were triggered.



Settings

This section talks about the different settings available on the Workflow Request page.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

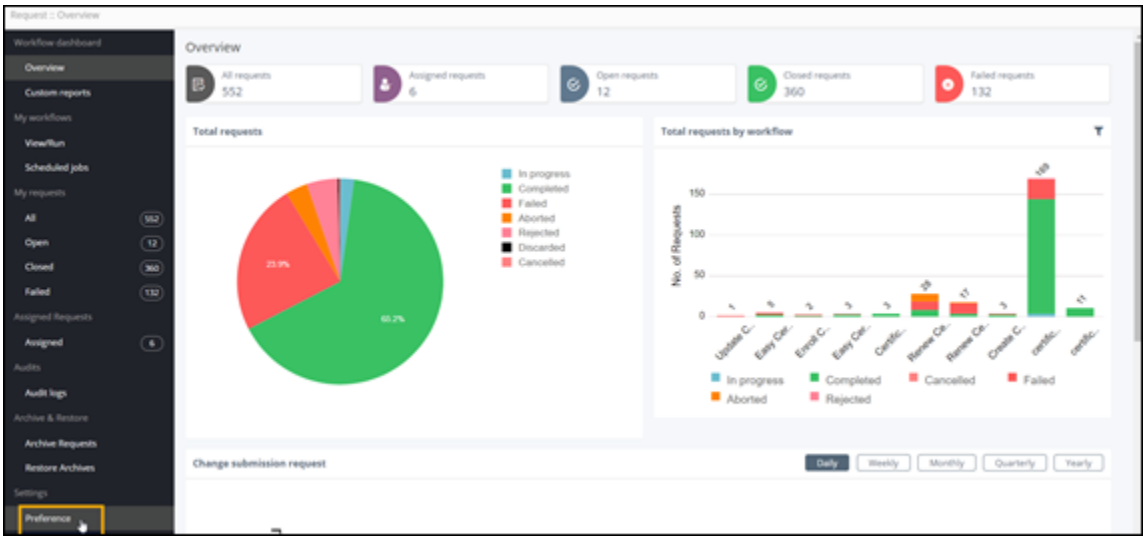
- Provision to select the Landing page.
- Provision to view the View/Run page in Classic mode.
- Provision to manage catalogs on the Request page.

- Selecting the Landing Page
- Viewing the View/Run page in Classic mode
- Managing the Catalogs

Selecting the Landing Page

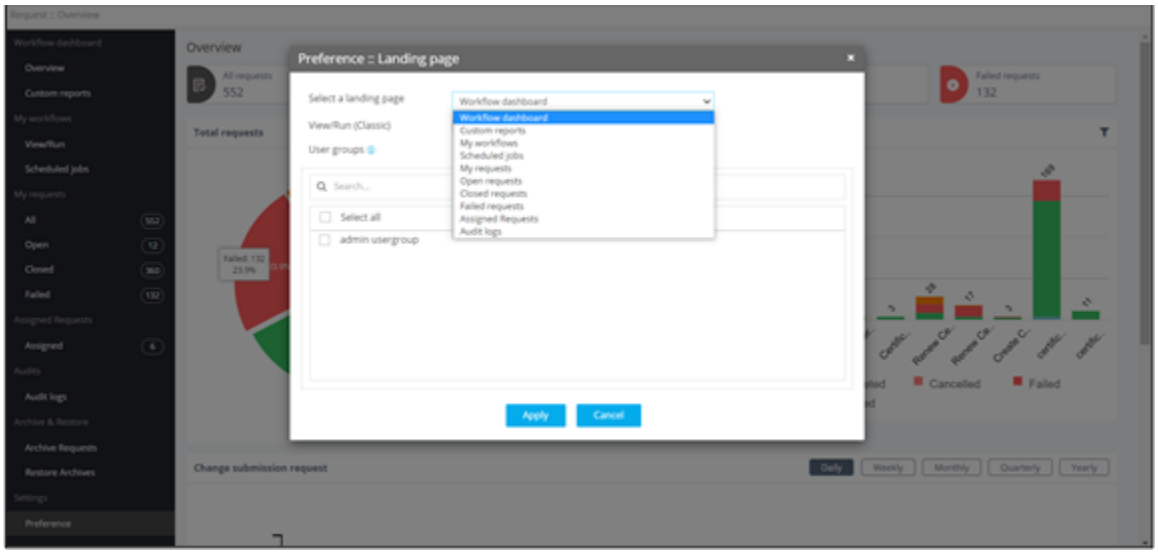
Users can set the page that is displayed when they access the Workflow Request page.

1. On the Request :: Overview page, from the navigation pane on the left, under **Settings**, click **Preference**.



2. In the **Preference :: Landing page** window, **Select a Landing page** from the list of options.

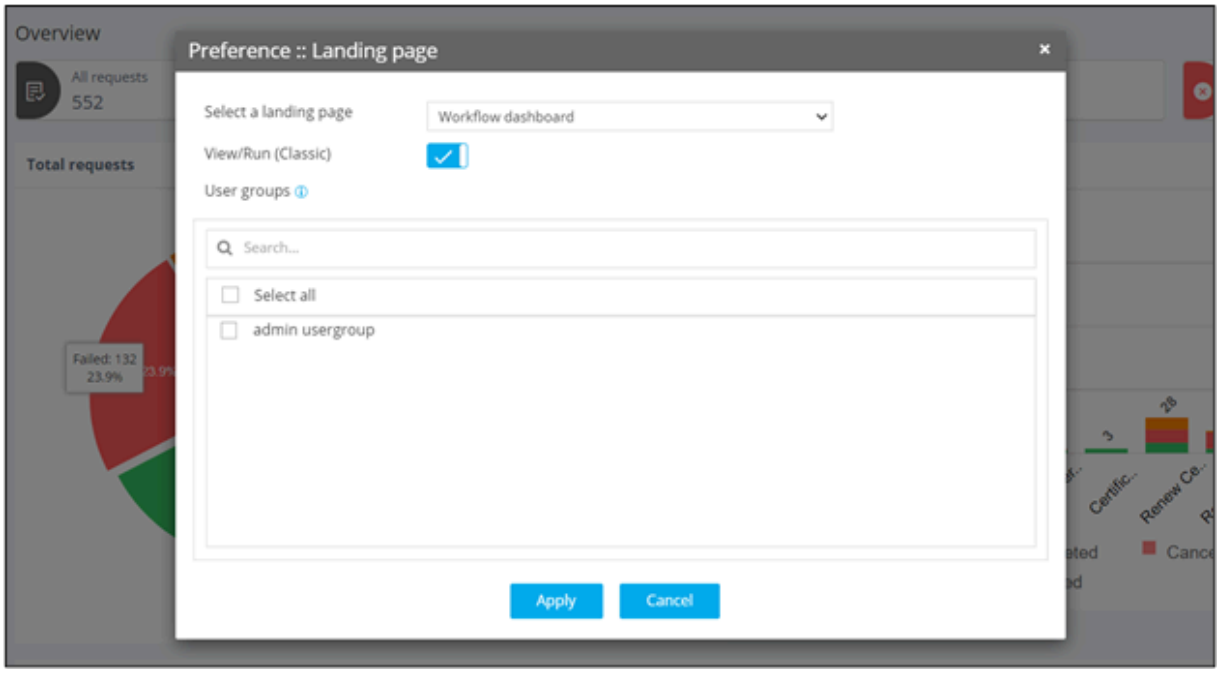
Copyright © 2020 AppViewX, Inc. All Rights Reserved.



3. Click **Apply**.

Viewing the View/Run page in Classic mode

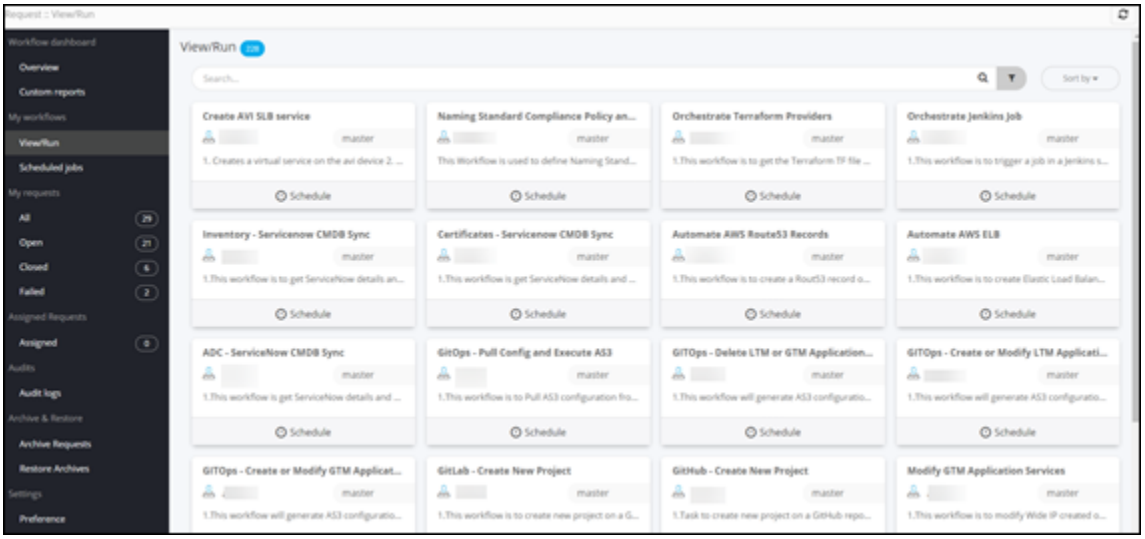
1. On the Request :: Overview page, from the navigation pane on the left, under **Settings**, click **Preference**.
2. To display the **View/Run** page in **Classic** mode, in the **Preference :: Landing page** window, turn on the **View/Run (Classic)** toggle.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

3. Click **Apply**.

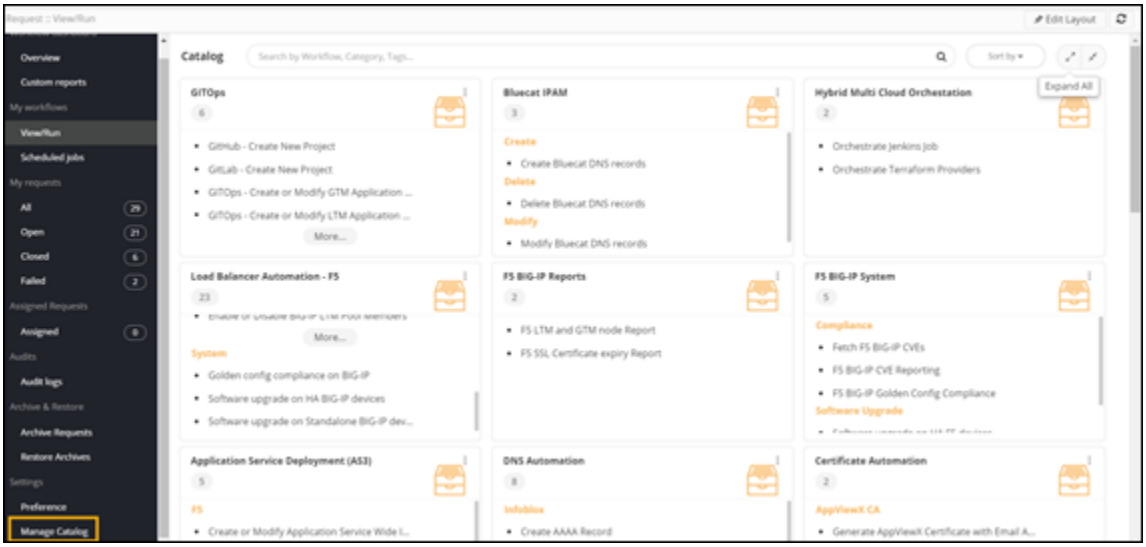
   The **View/Run** page is displayed in Classic mode.



Managing the Catalogs

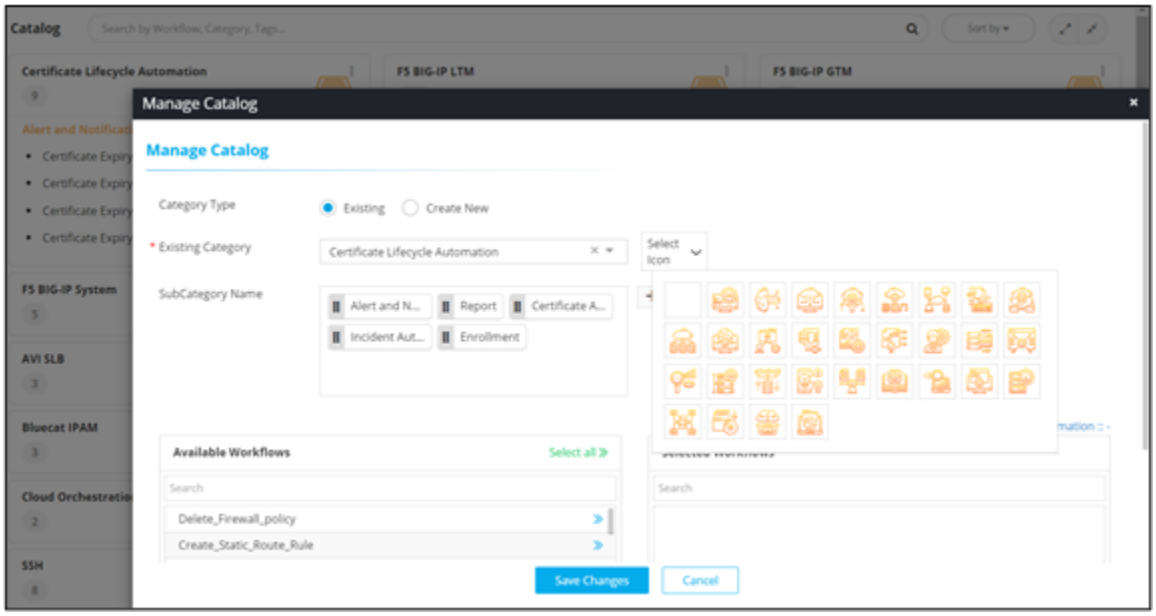
You can edit the catalog properties when the View/Run page is not in Classic mode. This feature allows you to create/modify catalog categories, select icons for catalogs etc.

1. On the Request :: View/Run page, from the navigation pane on the left, under **Settings**, click **Manage Catalog**.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

The **Manage Catalog** window is displayed.



2. To modify the existing workflow categories, under **Category Type**, select the **Existing** option.
3. To create a new category, under **Category Type**, select the **Create New** option.

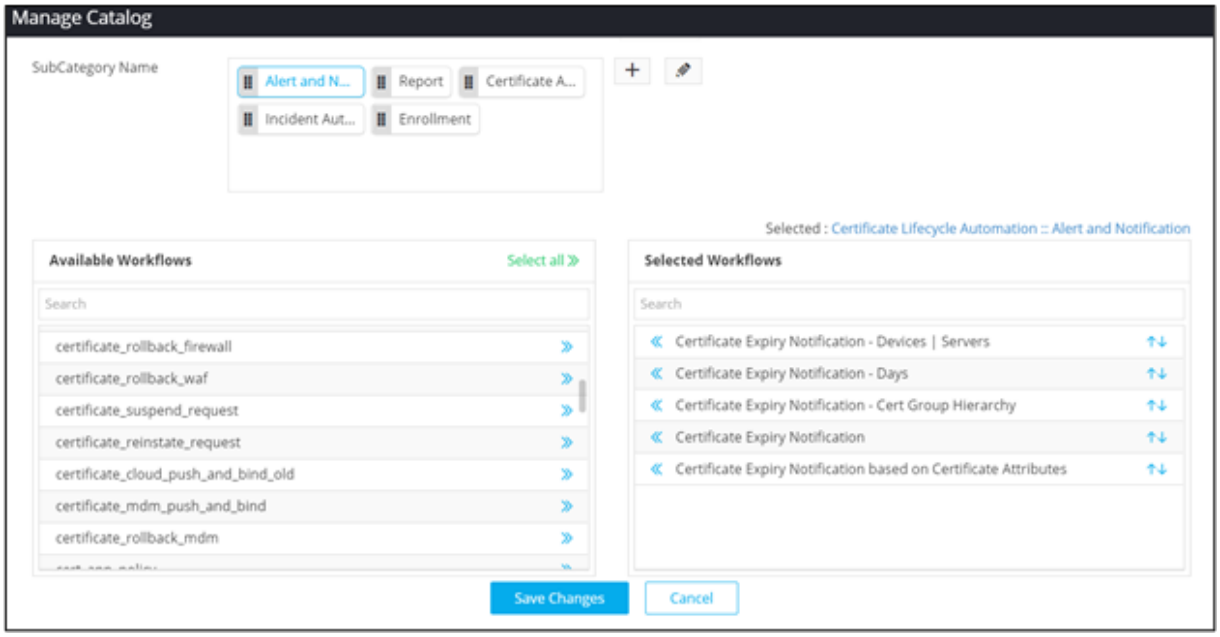
   The following table describes the options available in the **Manage Catalogs** window:

Option	Description
Category Type	Select the **Category Type** as: • **Existing**: Selecting this option allows you to select the category of workflows from the options available in the Existing Category dropdown. • **Create New**: Selecting this option allows you to create a new category of workflows.
***Existing Category**	Select the category of workflows from the options displayed in the dropdown. **Note:** If the **Create New** option is selected in the **Category Type** field, then this field becomes a text box and allows you to type a new category name.
Select Icon	Select an icon for the workflow card from the options available in the dropdown.
SubCategory Name	This field is auto-populated when you select the Existing Category option in the Category Type field.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Option	Description
+	Allows you to add more subcategories of workflows.
All asterisk (*) marked fields are mandatory.	

4. To add more workflows to a subcategory, select the subcategory in the **SubCategory Name** field and click » next to the workflows to be added.
5. To move a workflow out of the selected workflows, click « next to the workflow under **Selected Workflows**.



6. To modify the order in which the workflows are displayed within the subcategory, click ↑↓.
7. Click **Save Changes**.

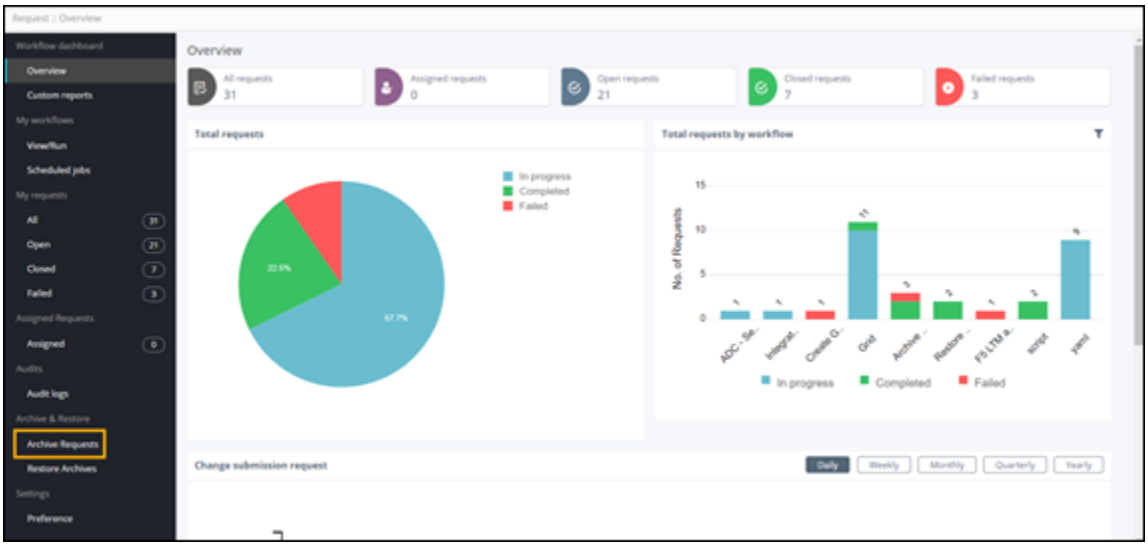
Archive Workflow Requests

You can archive workflow service requests as per your requirement. The requests can be archived by status, time period and so on. You can also schedule archivals and also purge service requests.

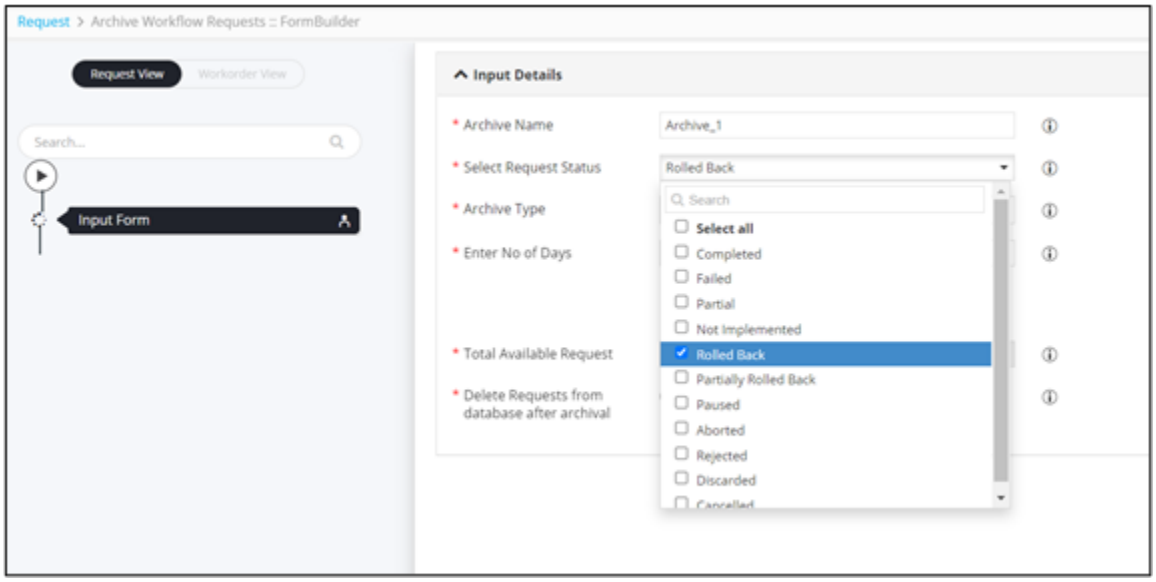
To trigger an archive workflow request:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. On the Request :: Overview page, from the navigation pane on the left, under **Archive & Restore**, click **Archive Requests**.

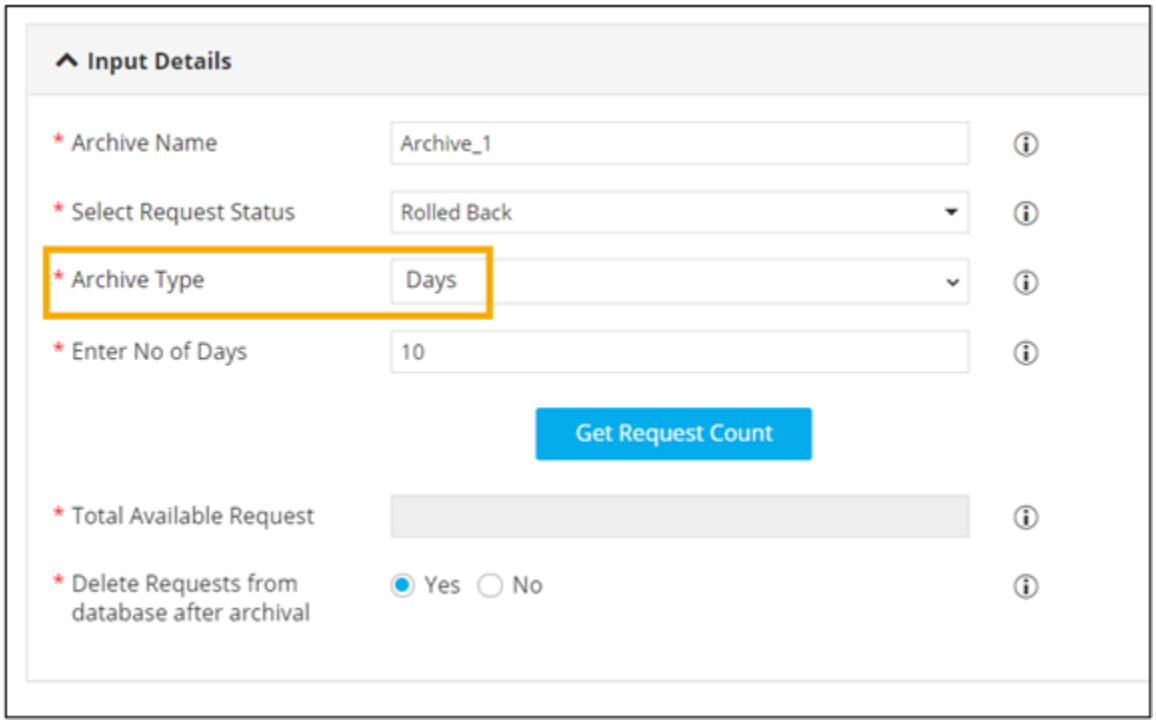


2. In the Input form, provide a suitable **Archive Name**.
3. Select one or more request status to archive workflows.



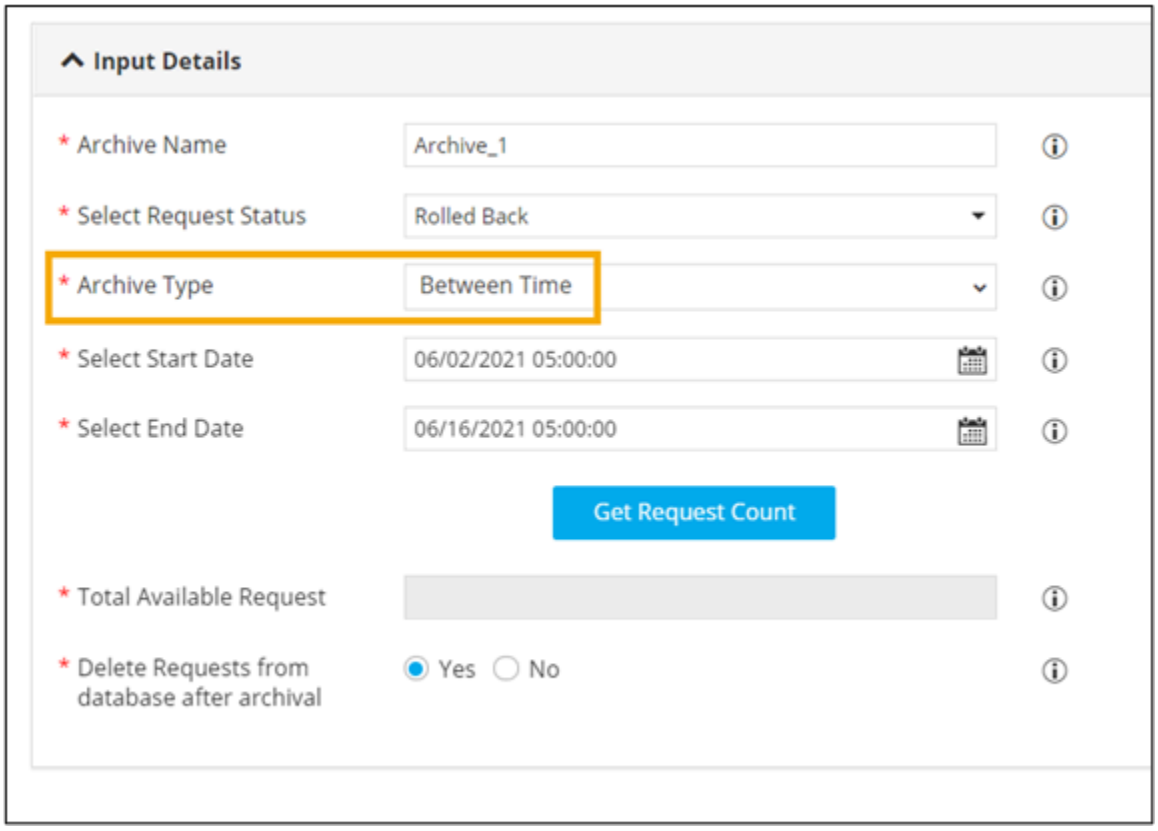
4. Select the duration for which the requests will be archived.
   For example, selecting the archive type as Days and number of days as 10 will archive requests older than 10 days.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



5. To archive workflow requests between specific dates, under **Archive Type**, select **Between Time**.
6. Select the Start and End dates.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



7. To get the total number of requests that match the selection, click **Get Request Count**.
8. To delete the requests from the database after archival, select **Yes**.
9. Click **Submit**.

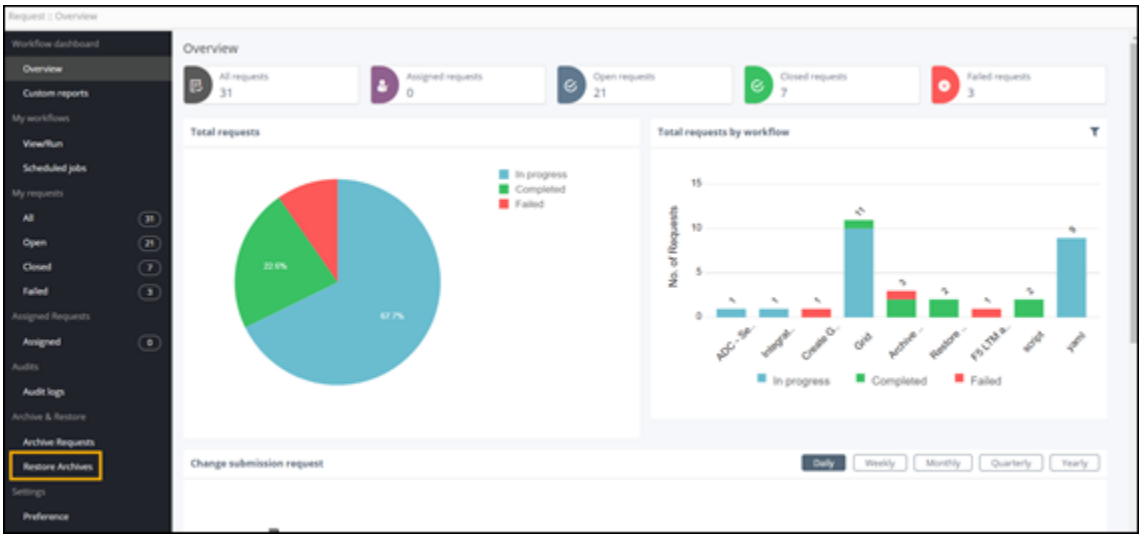
Restore Workflow Archives

You can restore workflow service requests from the Workflow Request page as per your requirement. The requests can be restored either in bulk or by selecting individual workflow requests.

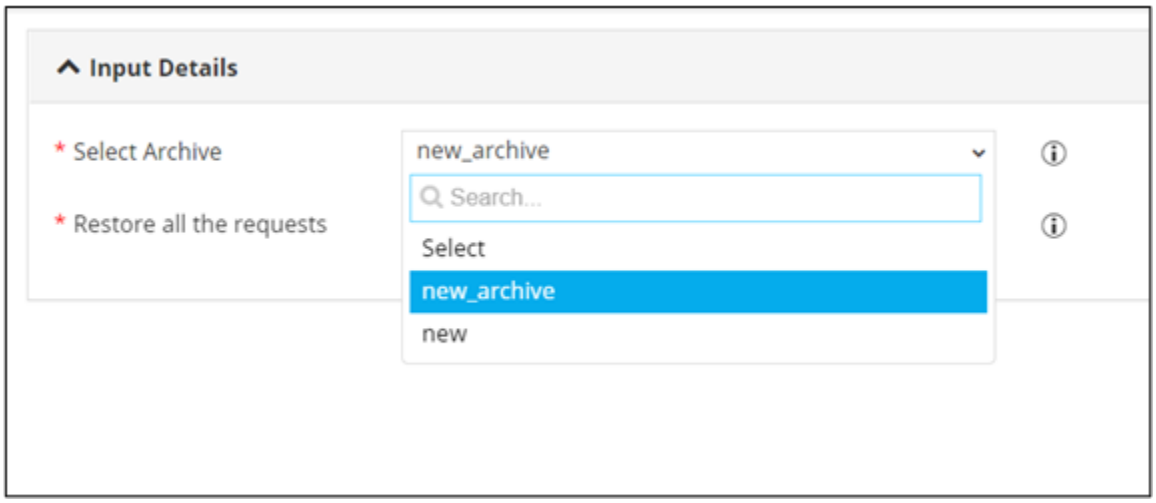
To trigger a restore workflow request:

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

1. On the Request :: Overview page, from the navigation pane on the left, under **Archive & Restore**, click **Restore Archives**.

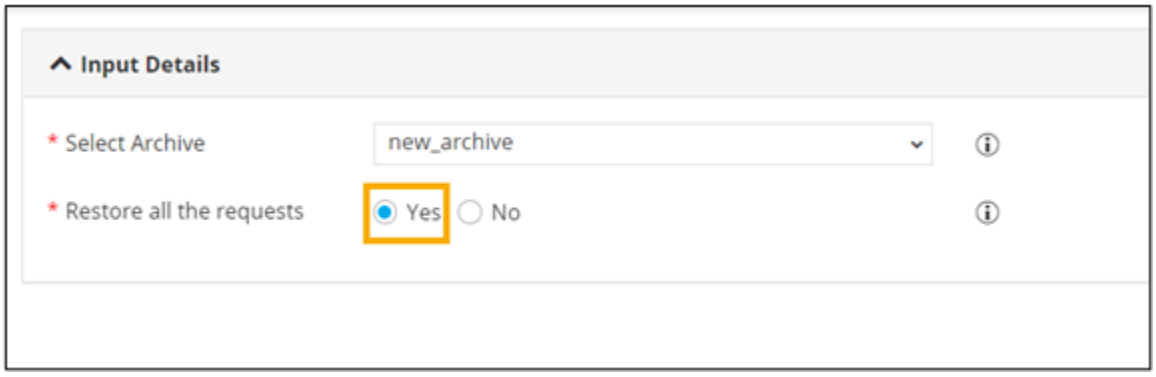


2. Select the archive that is to be restored.

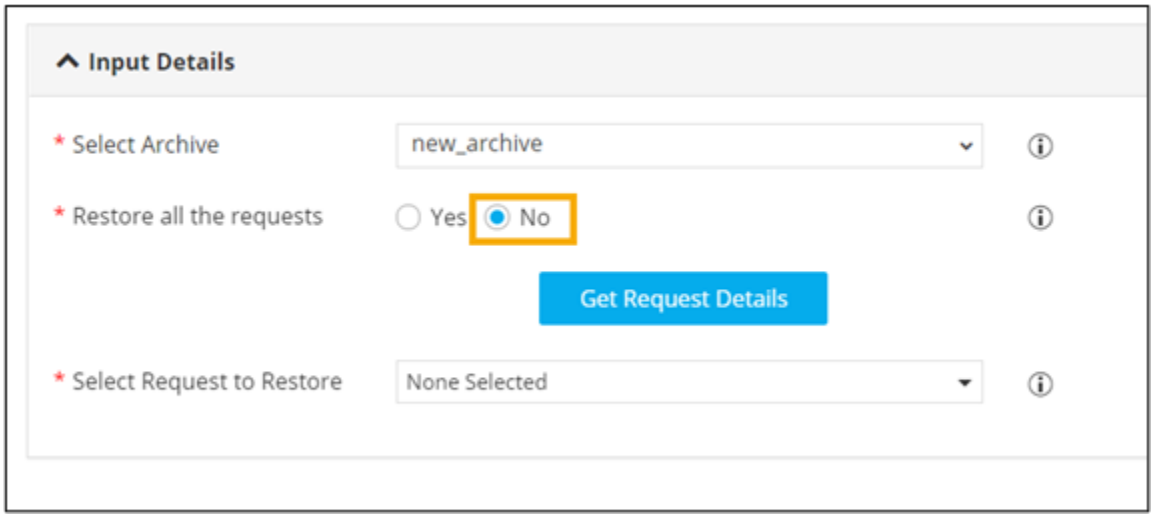


3. To restore all the requests from the archive, select **Yes**.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.

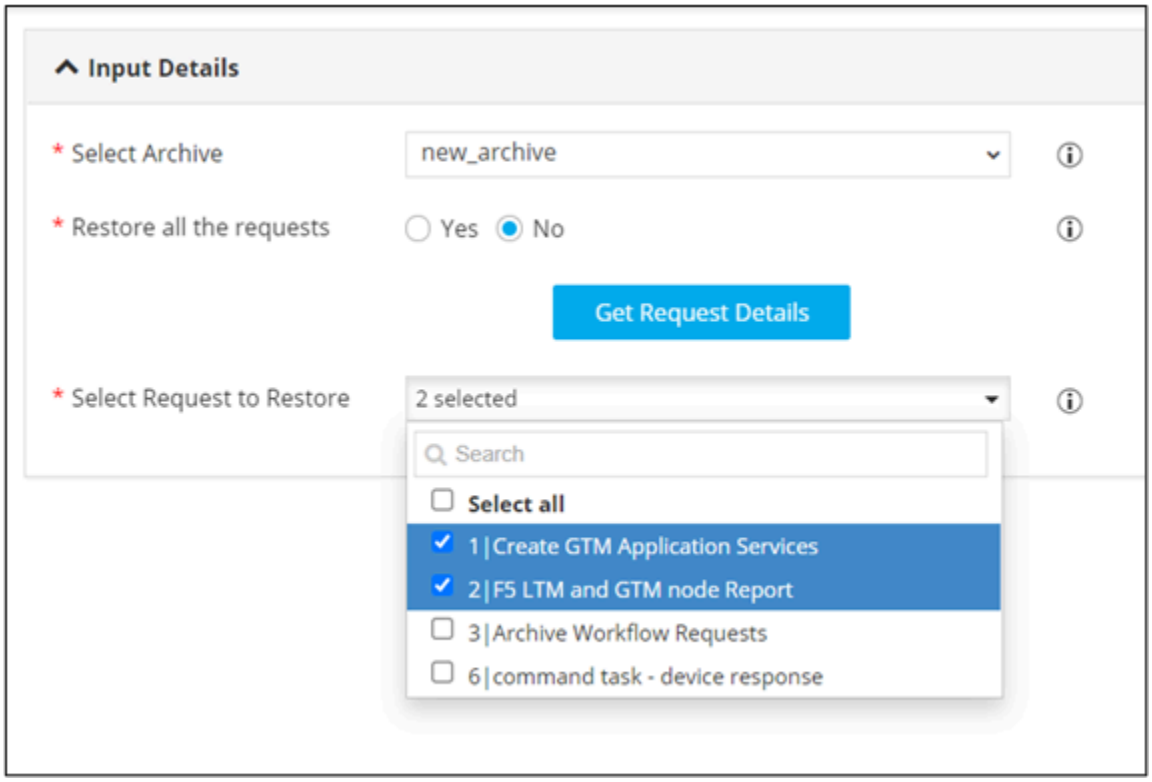


4. To select specific request(s) from the archive, select **No**.



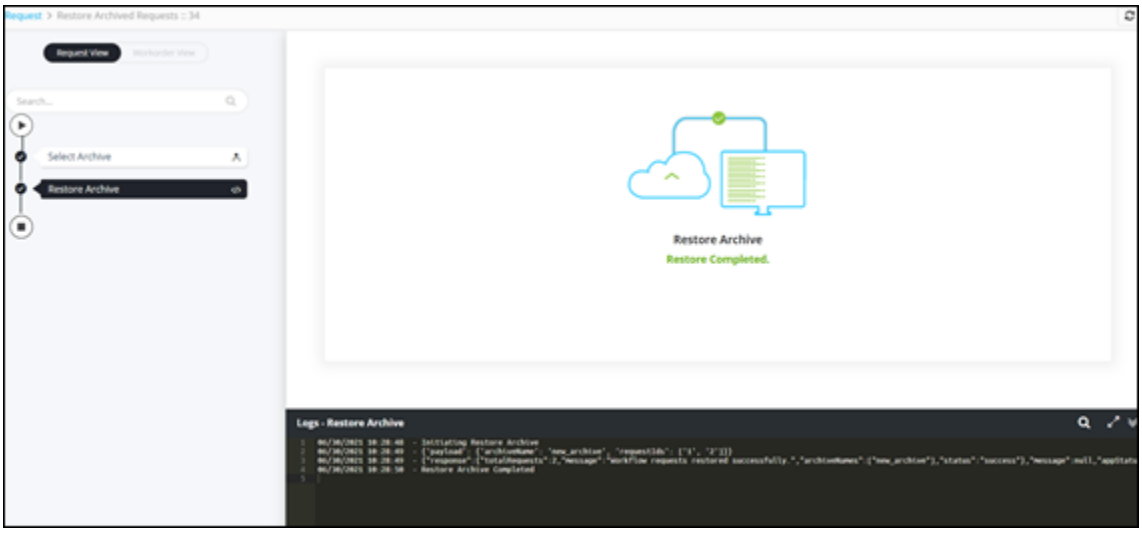
5. To auto populate the list of workflow requests, click **Get Request Details**.
6. Select the workflows that you want to restore.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.



7. To restore the selected workflow request(s) from the archive, click **Submit** .
8. In the **Confirmation** window, click **Ok**.

   The workflow to Restore Archive is completed.

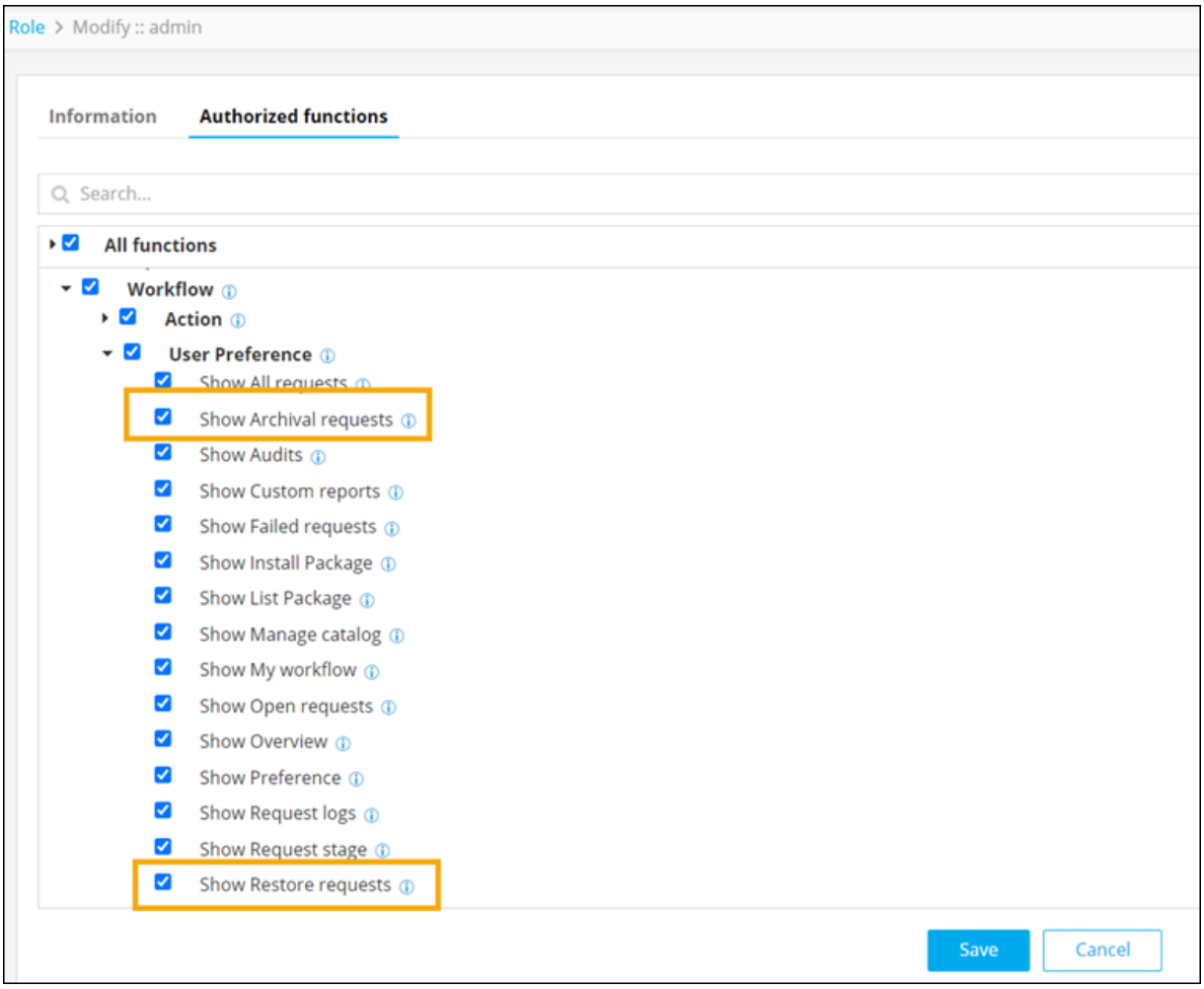


Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Configuring RBAC for Archive and Restore Requests

You can allow users to access the **Archive & Restore** section or restrict them from viewing this section on the Workflow Request page.

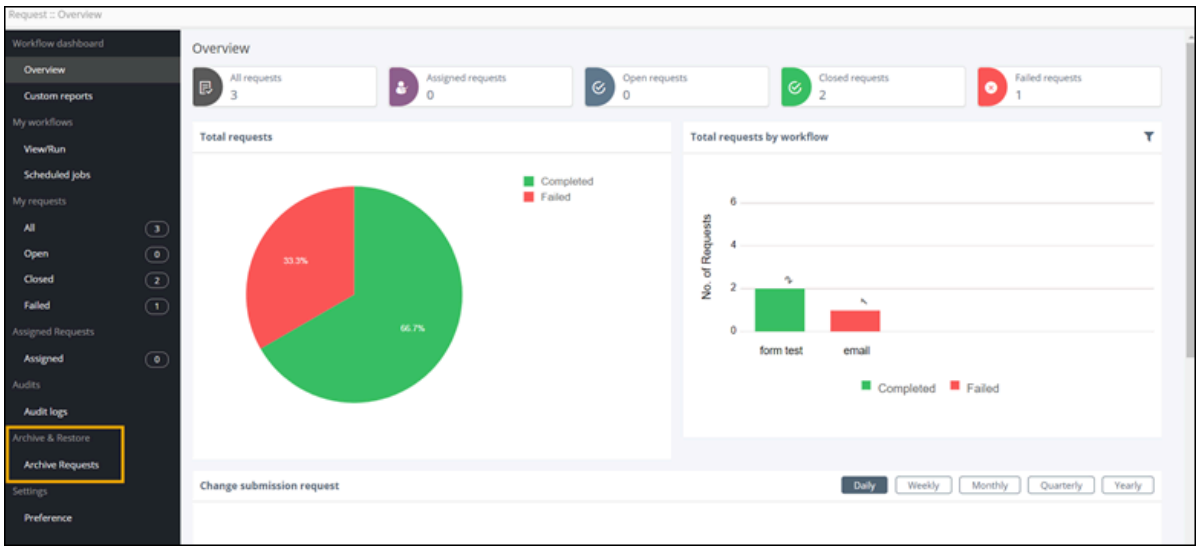
1. To allow users to access the **Archive & Restore**, from the top left corner of the screen, click ☰.
2. From the menu displayed, select **Account** > **Role**.
3. Select the role for which you want to enable access, for example, admin.
4. Under **Authorized functions**, select **Request** > **Workflow** > **User Preference**.
5. Under **User Preference**, ensure that the following checkboxes are selected:
   - **Show Archival requests**
   - **Show Restore requests**



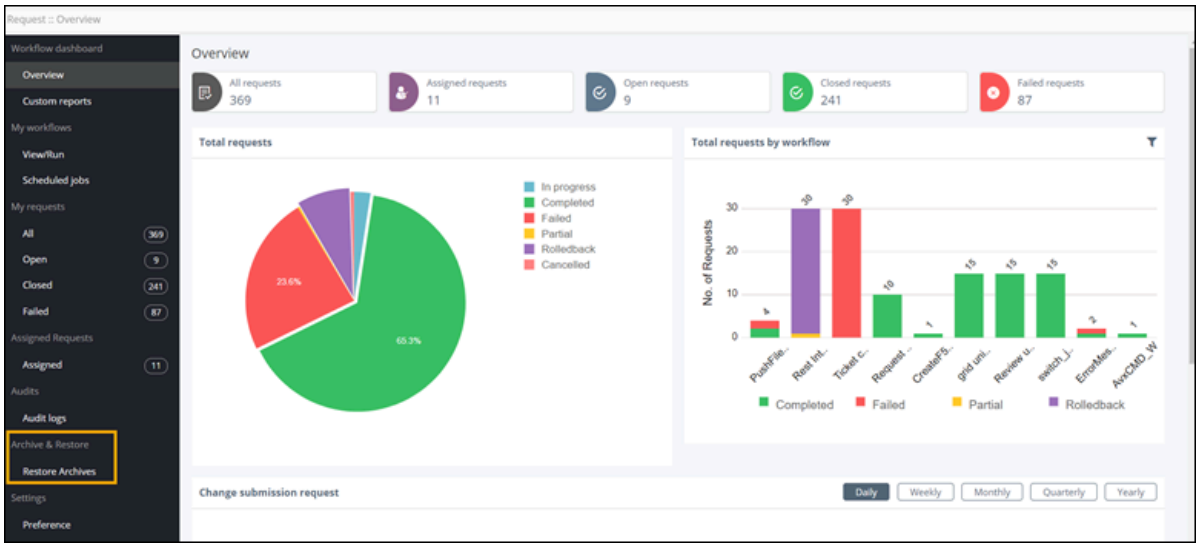
Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Note: Only the option that is selected under **User Preference** will be displayed as a submenu under the **Archive & Restore** section. If both options are not selected the Archive & Restore section will not be displayed at all.

- When only the **Show Archival requests** option is selected under **User Preference**.



- When only the **Show Restore requests** option is selected under **User Preference**.



Copyright © 2020 AppViewX, Inc. All Rights Reserved.

Chapter 11: Visual Workflow and Pages

Once you have designed your workflows, you can view them all in one place through AppViewX's self-service catalog pages. To access AppViewX Pages, navigate to **Studio** > **Pages**.

- Provision to enable self-service automation workflows by persona
- Provision to logically select and group workflows such as CLM Automation, Application Delivery and so on
- Provision to drag and drop workflow(s) to design custom self-service catalogs
- Provision to view workflows through a single pane of glass
- Provision to add automation, self service request alerts to a customized page
- Provision to view workflow request status (open, closed, failed) from the self-service catalog
- Provision to run/trigger and schedule workflows from self-service catalog pages

Note: For more information, refer to the Pages User Guide.

Copyright © 2020 AppViewX, Inc. All Rights Reserved.